User Guide

AWS CodeBuild

API Version 2016-10-06

AWS CodeBuild: User Guide

Table of Contents

What is AWS CodeBuild?

AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy. CodeBuild eliminates the need to provision, manage, and scale your own build servers. It provides prepackaged build environments for popular programming languages and build tools such as Apache Maven, Gradle, and more. You can also customize build environments in CodeBuild to use your own build tools. CodeBuild scales automatically to meet peak build requests.

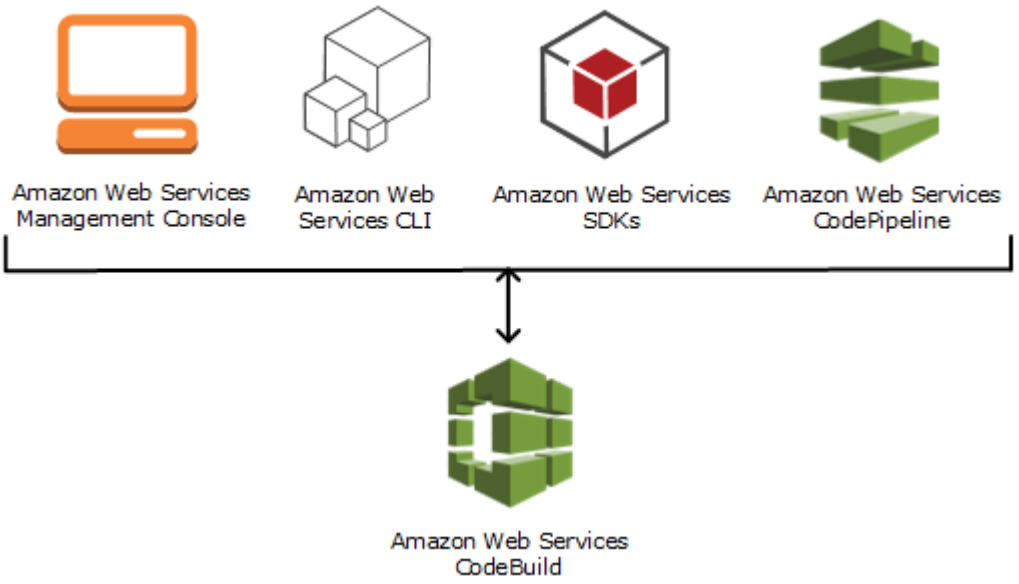
CodeBuild provides these benefits:

- **Fully managed** – CodeBuild eliminates the need to set up, patch, update, and manage your own build servers.
- **On demand** – CodeBuild scales on demand to meet your build needs. You pay only for the number of build minutes you consume.
- **Out of the box** – CodeBuild provides preconfigured build environments for the most popular programming languages. All you need to do is point to your build script to start your first build.

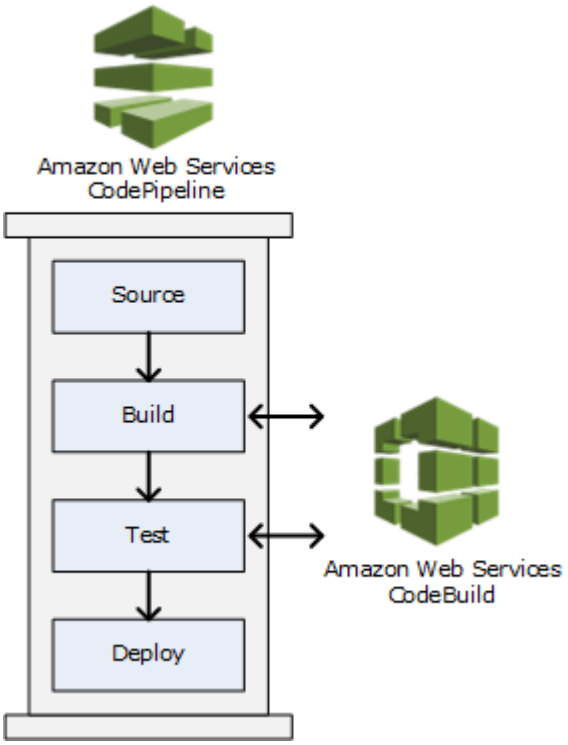
For more information, see AWS CodeBuild.

How to run CodeBuild

You can use the AWS CodeBuild or AWS CodePipeline console to run CodeBuild. You can also automate the running of CodeBuild by using the AWS Command Line Interface (AWS CLI) or the AWS SDKs.



As the following diagram shows, you can add CodeBuild as a build or test action to the build or test stage of a pipeline in AWS CodePipeline. AWS CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your code. This includes building your code. A *pipeline* is a workflow construct that describes how code changes go through a release process.



To use CodePipeline to create a pipeline and then add a CodeBuild build or test action, see Use CodeBuild with CodePipeline. For more information about CodePipeline, see the AWS CodePipeline User Guide.

The CodeBuild console also provides a way to quickly search for your resources, such as repositories, build projects, deployment applications, and pipelines. Choose **Go to resource** or press the / key, and then enter the name of the resource. Any matches appear in the list. Searches are case insensitive. You only see resources that you have permissions to view. For more information, see Viewing resources in the console.

Pricing for CodeBuild

For information, see CodeBuild pricing.

How do I get started with CodeBuild?

We recommend that you complete the following steps:

1. **Learn** more about CodeBuild by reading the information in Concepts.
2. **Experiment** with CodeBuild in an example scenario by following the instructions in Getting started using the console.
3. **Use** CodeBuild in your own scenarios by following the instructions in Plan a build.

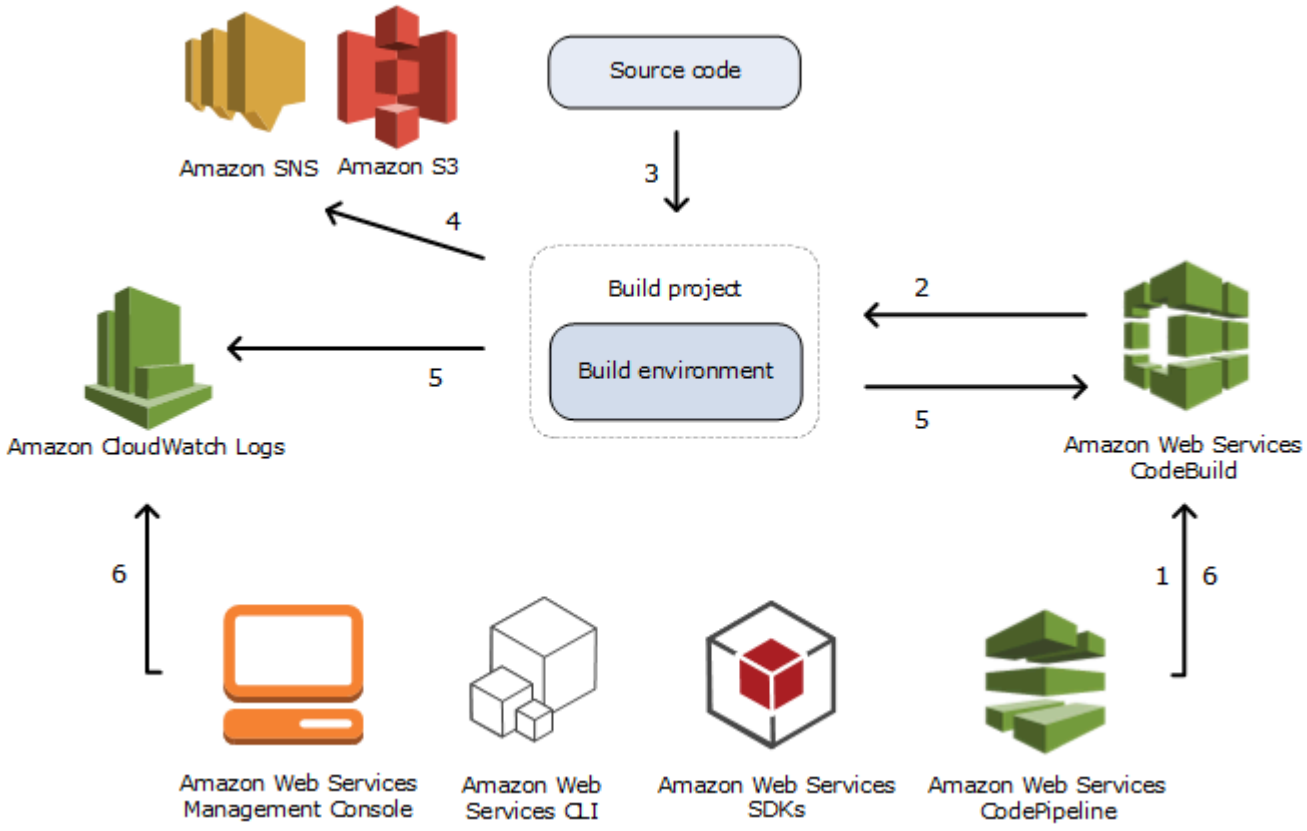
AWS CodeBuild concepts

The following concepts are important for understanding how CodeBuild works.

Topics

How CodeBuild works

The following diagram shows what happens when you run a build with CodeBuild:



1. As input, you must provide CodeBuild with a build project. A *build project* includes information about how to run a build, including where to get the source code, which build environment to use, which build commands to run, and where to store the build output. A *build environment* represents a combination of operating system, programming language runtime, and tools that CodeBuild uses to run a build. For more information, see:

   - Create a build project
   - Build environment reference

2. CodeBuild uses the build project to create the build environment.
3. CodeBuild downloads the source code into the build environment and then uses the build specification (buildspec), as defined in the build project or included directly in the source code. A *buildspec* is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build. For more information, see the Buildspec reference.
4. If there is any build output, the build environment uploads its output to an S3 bucket. The build environment can also perform tasks that you specify in the buildspec (for example, sending build notifications to an Amazon SNS topic). For an example, see Build notifications sample.

5. While the build is running, the build environment sends information to CodeBuild and Amazon CloudWatch Logs.
6. While the build is running, you can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to get summarized build information from CodeBuild and detailed build information from Amazon CloudWatch Logs. If you use AWS CodePipeline to run builds, you can get limited build information from CodePipeline.

Next steps

Now that you know more about AWS CodeBuild, we recommend these next steps:

1. **Experiment** with CodeBuild in an example scenario by following the instructions in Getting started using the console.
2. **Use** CodeBuild in your own scenarios by following the instructions in Plan a build.

Getting started with CodeBuild

In the following tutorials, you use AWS CodeBuild to build a collection of sample source code input files into a deployable version of the source code.

Both tutorials have the same input and results, but one uses the AWS CodeBuild console and the other uses the AWS CLI.

> **Important**
>
> We do not recommend that you use your AWS root account to complete this tutorial.

Topics

Getting started with AWS CodeBuild using the console

In this tutorial, you use AWS CodeBuild to build a collection of sample source code input files (*build input artifacts* or *build input*) into a deployable version of the source code (*build output artifact* or *build output*). Specifically, you instruct CodeBuild to use Apache Maven, a common build tool, to build a set of Java class files into a Java Archive (JAR) file. You do not need to be familiar with Apache Maven or Java to complete this tutorial.

You can work with CodeBuild through the CodeBuild console, AWS CodePipeline, the AWS CLI, or the AWS SDKs. This tutorial demonstrates how to use the CodeBuild console. For information about using CodePipeline, see Use CodeBuild with CodePipeline.

> **Important**
>
> The steps in this tutorial require you to create resources (for example, an S3 bucket) that might result in charges to your AWS account. These include possible charges for CodeBuild and for AWS resources and actions related to Amazon S3, AWS KMS, and CloudWatch Logs. For more information, see AWS CodeBuild pricing, Amazon S3 pricing, AWS Key Management Service pricing, and Amazon CloudWatch pricing.

Topics

Step 1: Create the source code

(Part of: Getting started with AWS CodeBuild using the console)

In this step, you create the source code that you want CodeBuild to build to the output bucket. This source code consists of two Java class files and an Apache Maven Project Object Model (POM) file.

1. In an empty directory on your local computer or instance, create this directory structure.

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

2. Using a text editor of your choice, create this file, name it `MessageUtil.java`, and then save it in the `src/main/java` directory.

```
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;
  }
}
```

This class file creates as output the string of characters passed into it. The `MessageUtil` constructor sets the string of characters. The `printMessage` method creates the output. The `salutationMessage` method outputs `Hi!` followed by the string of characters.

3. Create this file, name it `TestMessageUtil.java`, and then save it in the `/src/test/java` directory.

```
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());
  }
```

```
}
```

This class file sets the `message` variable in the `MessageUtil` class to `Robert`. It then tests to see if the `message` variable was successfully set by checking whether the strings `Robert` and `Hi!Robert` appear in the output.

4. Create this file, name it `pom.xml`, and then save it in the root (top level) directory.

```
<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 uses the instructions in this file to convert the `MessageUtil.java` and `TestMessageUtil.java` files into a file named `messageUtil-1.0.jar` and then run the specified tests.

At this point, your directory structure should look like this.

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

Step 2: Create the buildspec file

(Previous step: Step 1: Create the source code)

In this step, you create a build specification (build spec) file. A *buildspec* is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build. Without a build spec, CodeBuild cannot successfully convert your build input into build output or locate the build output artifact in the build environment to upload to your output bucket.

Create this file, name it `buildspec.yml`, and then save it in the root (top level) directory.

```
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

Because a build spec declaration must be valid YAML, the spacing in a build spec declaration is important. If the number of spaces in your build spec declaration does not match this one, the build might fail immediately. You can use a YAML validator to test whether your build spec declaration is valid YAML.

Note

Instead of including a build spec file in your source code, you can declare build commands separately when you create a build project. This is helpful if you want to build your source code with different build commands without updating your source code's repository each time. For more information, see Buildspec syntax.

In this build spec declaration:

- `version` represents the version of the build spec standard being used. This build spec declaration uses the latest version, `0.2`.
- `phases` represents the build phases during which you can instruct CodeBuild to run commands. These build phases are listed here as `install`, `pre_build`, `build`, and `post_build`. You cannot change the spelling of these build phase names, and you cannot create more build phase names.

  In this example, during the `build` phase, CodeBuild runs the `mvn install` command. This command instructs Apache Maven to compile, test, and package the compiled Java class files into a build output artifact. For completeness, a few `echo` commands are placed in each build phase in this example. When you view detailed build information later in this tutorial, the output of these `echo` commands can help you better understand how CodeBuild runs commands and in which order. (Although all build phases are included in this example, you are not required to include a build phase if you do not plan to run any commands during that phase.) For each build phase, CodeBuild runs each specified command, one at a time, in the order listed, from beginning to end.
- `artifacts` represents the set of build output artifacts that CodeBuild uploads to the output bucket. `files` represents the files to include in the build output. CodeBuild uploads the single `messageUtil-1.0.jar` file found in the `target` relative directory in the build environment.

The file name `messageUtil-1.0.jar` and the directory name `target` are based on the way Apache Maven creates and stores build output artifacts for this example only. In your own builds, these file names and directories are different.

For more information, see the Buildspec reference.

At this point, your directory structure should look like this.

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

Step 3: Create two S3 buckets

(Previous step: Step 2: Create the buildspec file)

Although you can use a single bucket for this tutorial, two buckets makes it easier to see where the build input is coming from and where the build output is going.

- One of these buckets (the *input bucket*) stores the build input. In this tutorial, the name of this input bucket is `codebuild-`*`region-ID`*`-`*`account-ID`*`-input-bucket`, where *`region-ID`* is the AWS Region of the bucket and *`account-ID`* is your AWS account ID.
- The other bucket (the *output bucket*) stores the build output. In this tutorial, the name of this output bucket is `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`.

If you chose different names for these buckets, be sure to use them throughout this tutorial.

These two buckets must be in the same AWS Region as your builds. For example, if you instruct CodeBuild to run a build in the US East (Ohio) Region, these buckets must also be in the US East (Ohio) Region.

For more information, see Creating a Bucket in the *Amazon Simple Storage Service User Guide*.

> **Note**
>
> Although CodeBuild also supports build input stored in CodeCommit, GitHub, and Bitbucket repositories, this tutorial does not show you how to use them. For more information, see Plan a build.

Step 4: Upload the source code and the buildspec file

(Previous step: Step 3: Create two S3 buckets)

In this step, you add the source code and build spec file to the input bucket.

Using your operating system's zip utility, create a file named `MessageUtil.zip` that includes `MessageUtil.java`, `TestMessageUtil.java`, `pom.xml`, and `buildspec.yml`.

The `MessageUtil.zip` file's directory structure must look like this.

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

> **Important**
>
> Do not include the *(root directory name)* directory, only the directories and files in the *(root directory name)* directory.

Upload the `MessageUtil.zip` file to the input bucket named `codebuild-`*region-ID*`-`*account-ID*`-input-bucket`.

Important

For CodeCommit, GitHub, and Bitbucket repositories, by convention, you must store a build spec file named `buildspec.yml` in the root (top level) of each repository or include the build spec declaration as part of the build project definition. Do not create a ZIP file that contains the repository's source code and build spec file.
For build input stored in S3 buckets only, you must create a ZIP file that contains the source code and, by convention, a build spec file named `buildspec.yml` at the root (top level) or include the build spec declaration as part of the build project definition.
If you want to use a different name for your build spec file, or you want to reference a build spec in a location other than the root, you can specify a build spec override as part of the build project definition. For more information, see Buildspec file name and storage location.

Step 5: Create the build project

(Previous step: Step 4: Upload the source code and the buildspec file)

In this step, you create a build project that AWS CodeBuild uses to run the build. A *build project* includes information about how to run a build, including where to get the source code, which build environment to use, which build commands to run, and where to store the build output. A *build environment* represents a combination of operating system, programming language runtime, and tools that CodeBuild uses to run a build. The build environment is expressed as a Docker image. For more information, see Docker overview on the Docker Docs website.

For this build environment, you instruct CodeBuild to use a Docker image that contains a version of the Java Development Kit (JDK) and Apache Maven.

To create the build project

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Use the AWS region selector to choose an AWS Region where CodeBuild is supported. For more information, see AWS CodeBuild endpoints and quotas in the *Amazon Web Services General Reference*.
3. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.

4. On the **Create build project** page, in **Project configuration**, for **Project name**, enter a name for this build project (in this example, `codebuild-demo-project`). Build project names must be unique across each AWS account. If you use a different name, be sure to use it throughout this tutorial.

> **Note**
>
> On the **Create build project** page, you might see an error message similar to the following: **You are not authorized to perform this operation.**. This is most likely because you signed in to the AWS Management Console as an user who does not have permissions to create a build project.. To fix this, sign out of the AWS Management Console, and then sign back in with credentials belonging to one of the following IAM entities:
>
> - An administrator user in your AWS account. For more information, see Creating your first AWS account root user and group in the *user Guide*.
> - An user in your AWS account with the `AWSCodeBuildAdminAccess`, `AmazonS3ReadOnlyAccess`, and `IAMFullAccess` managed policies attached to that user or to an IAM group that the user belongs to. If you do not have an user or group in your AWS account with these permissions, and you cannot add these permissions to your user or group, contact your AWS account administrator for assistance. For more information, see AWS managed (predefined) policies for AWS CodeBuild.
>
> Both options include administrator permissions that allow you to create a build project so you can complete this tutorial. We recommend that you always use the minimum permissions required to accomplish your task. For more information, see AWS CodeBuild permissions reference.

5. In **Source**, for **Source provider**, choose **Amazon S3**.
6. For **Bucket**, choose **codebuild-*region-ID*-*account-ID*-input-bucket**.
7. For **S3 object key**, enter **`MessageUtil.zip`**.
8. In **Environment**, for **Environment image**, leave **Managed image** selected.
9. For **Operating system**, choose **Amazon Linux**.
10. For **Runtime(s)**, choose **Standard**.

11. For **Image**, choose **aws/codebuild/amazonlinux-x86_64-standard:corretto11**.
12. In **Service role**, leave **New service role** selected, and leave **Role name** unchanged.
13. For **Buildspec**, leave **Use a buildspec file** selected.
14. In **Artifacts**, for **Type**, choose **Amazon S3**.
15. For **Bucket name**, choose **codebuild-*region-ID*-*account-ID*-output-bucket**.
16. Leave **Name** and **Path** blank.
17. Choose **Create build project**.

Step 6: Run the build

(Previous step: Step 5: Create the build project)

In this step, you instruct AWS CodeBuild to run the build with the settings in the build project.

To run the build

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. In the list of build projects, choose **codebuild-demo-project**, and then choose **Start build**. The build starts immediately.

Step 7: View summarized build information

(Previous step: Step 6: Run the build)

In this step, you view summarized information about the status of your build.

To view summarized build information

1. If the **codebuild-demo-project:*<build-ID>*** page is not displayed, in the navigation bar, choose **Build history**. Next, in the list of build projects, for **Project**, choose the **Build run** link for **codebuild-demo-project**. There should be only one matching link. (If you have completed this tutorial before, choose the link with the most recent value in the **Completed** column.)
2. On the **Build status** page, in **Phase details**, the following build phases should be displayed, with **Succeeded** in the **Status** column:

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

In **Build Status**, **Succeeded** should be displayed.

If you see **In Progress** instead, choose the refresh button.

3. Next to each build phase, the **Duration** value indicates how long the build phase lasted. The **End time** value indicates when that build phase ended.

Step 8: View detailed build information

(Previous step: Step 7: View summarized build information)

In this step, you view detailed information about your build in CloudWatch Logs.

> **Note**
>
> To protect sensitive information, the following are hidden in CodeBuild logs:
>
> - AWS access key IDs. For more information, see Managing Access Keys for IAM Users in the *AWS Identity and Access Management User Guide*.
> - Strings specified using the Parameter Store. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the *Amazon EC2 Systems Manager User Guide*.

- Strings specified using AWS Secrets Manager. For more information, see Key management.

To view detailed build information

1. With the build details page still displayed from the previous step, the last 10,000 lines of the build log are displayed in **Build logs**. To see the entire build log in CloudWatch Logs, choose the **View entire log** link.
2. In the CloudWatch Logs log stream, you can browse the log events. By default, only the last set of log events is displayed. To see earlier log events, scroll to the beginning of the list.
3. In this tutorial, most of the log events contain verbose information about CodeBuild downloading and installing build dependency files into its build environment, which you probably don't care about. You can use the **Filter events** box to reduce the information displayed. For example, if you enter `"[INFO]"` in **Filter events**, only those events that contain `[INFO]` are displayed. For more information, see Filter and pattern syntax in the *Amazon CloudWatch User Guide*.

Step 9: Get the build output artifact

(Previous step: Step 8: View detailed build information)

In this step, you get the `messageUtil-1.0.jar` file that CodeBuild built and uploaded to the output bucket.

You can use the CodeBuild console or the Amazon S3 console to complete this step.

To get the build output artifact (AWS CodeBuild console)

1. With the CodeBuild console still open and the build details page still displayed from the previous step, choose the **Build details** tab and scroll down to the **Artifacts** section.

> **Note**
>
> If the build details page is not displayed, in the navigation bar, choose **Build history**, and then choose the **Build run** link.

2. The link to the Amazon S3 folder is under the **Artifacts upload location**. This link opens the folder in Amazon S3 where you find the `messageUtil-1.0.jar` build output artifact file.

To get the build output artifact (Amazon S3 console)

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Open `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`.
3. Open the `codebuild-demo-project` folder.
4. Open the `target` folder, where you find the `messageUtil-1.0.jar` build output artifact file.

Step 10: Delete the S3 buckets

(Previous step: Step 9: Get the build output artifact)

To prevent ongoing charges to your AWS account, you can delete the input and output buckets used in this tutorial. For instructions, see Deleting or Emptying a Bucket in the *Amazon Simple Storage Service User Guide*.

If you are using the IAM user or an administrator IAM user to delete these buckets, the user must have more access permissions. Add the following statement between the markers (*`### BEGIN ADDING STATEMENT HERE ###`* and *`### END ADDING STATEMENTS HERE ###`*) to an existing access policy for the user.

The ellipses (...) in this statement are used for brevity. Do not remove any statements in the existing access policy. Do not enter these ellipses into the policy.

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

```
    }
    ### END ADDING STATEMENT HERE ###
  ]
}
```

Wrapping up

In this tutorial, you used AWS CodeBuild to build a set of Java class files into a JAR file. You then viewed the build's results.

You can now try using CodeBuild in your own scenarios. Follow the instructions in Plan a build. If you don't feel ready yet, you might want to try building some of the samples. For more information, see Use case-based samples for CodeBuild.

Getting started with AWS CodeBuild using the AWS CLI

In this tutorial, you use AWS CodeBuild to build a collection of sample source code input files (called *build input artifacts* or *build input*) into a deployable version of the source code (called *build output artifact* or *build output*). Specifically, you instruct CodeBuild to use Apache Maven, a common build tool, to build a set of Java class files into a Java Archive (JAR) file. You do not need to be familiar with Apache Maven or Java to complete this tutorial.

You can work with CodeBuild through the CodeBuild console, AWS CodePipeline, the AWS CLI, or the AWS SDKs. This tutorial demonstrates how to use CodeBuild with the AWS CLI. For information about using CodePipeline, see Use CodeBuild with CodePipeline.

> **Important**
>
> The steps in this tutorial require you to create resources (for example, an S3 bucket) that might result in charges to your AWS account. These include possible charges for CodeBuild and for AWS resources and actions related to Amazon S3, AWS KMS, and CloudWatch Logs. For more information, see CodeBuild pricing, Amazon S3 pricing, AWS Key Management Service pricing, and Amazon CloudWatch pricing.

Topics

Step 1: Create the source code

(Part of: Getting started with AWS CodeBuild using the AWS CLI)

In this step, you create the source code that you want CodeBuild to build to the output bucket. This source code consists of two Java class files and an Apache Maven Project Object Model (POM) file.

1. In an empty directory on your local computer or instance, create this directory structure.

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

2. Using a text editor of your choice, create this file, name it `MessageUtil.java`, and then save it in the `src/main/java` directory.

```
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;
  }
}
```

This class file creates as output the string of characters passed into it. The `MessageUtil` constructor sets the string of characters. The `printMessage` method creates the output. The `salutationMessage` method outputs `Hi!` followed by the string of characters.

3. Create this file, name it `TestMessageUtil.java`, and then save it in the `/src/test/java` directory.

```
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());
  }
}
```

This class file sets the `message` variable in the `MessageUtil` class to `Robert`. It then tests to see if the `message` variable was successfully set by checking whether the strings `Robert` and `Hi!Robert` appear in the output.

4. Create this file, name it `pom.xml`, and then save it in the root (top level) directory.

```
<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 uses the instructions in this file to convert the `MessageUtil.java` and `TestMessageUtil.java` files into a file named `messageUtil-1.0.jar` and then run the specified tests.

At this point, your directory structure should look like this.

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

Step 2: Create the buildspec file

(Previous step: Step 1: Create the source code)

In this step, you create a build specification (build spec) file. A *buildspec* is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build. Without a build spec, CodeBuild cannot successfully convert your build input into build output or locate the build output artifact in the build environment to upload to your output bucket.

Create this file, name it `buildspec.yml`, and then save it in the root (top level) directory.

```
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

Because a build spec declaration must be valid YAML, the spacing in a build spec declaration is important. If the number of spaces in your build spec declaration does not match this one, the build might fail immediately. You can use a YAML validator to test whether your build spec declaration is valid YAML.

Note

Instead of including a build spec file in your source code, you can declare build commands separately when you create a build project. This is helpful if you want to build your source code with different build commands without updating your source code's repository each time. For more information, see Buildspec syntax.

In this build spec declaration:

- `version` represents the version of the build spec standard being used. This build spec declaration uses the latest version, `0.2`.
- `phases` represents the build phases during which you can instruct CodeBuild to run commands. These build phases are listed here as `install`, `pre_build`, `build`, and `post_build`. You cannot change the spelling of these build phase names, and you cannot create more build phase names.

  In this example, during the `build` phase, CodeBuild runs the `mvn install` command. This command instructs Apache Maven to compile, test, and package the compiled Java class files into a build output artifact. For completeness, a few `echo` commands are placed in each build phase in this example. When you view detailed build information later in this tutorial, the output of these `echo` commands can help you better understand how CodeBuild runs commands and in which order. (Although all build phases are included in this example, you are not required to include a build phase if you do not plan to run any commands during that phase.) For each build phase, CodeBuild runs each specified command, one at a time, in the order listed, from beginning to end.
- `artifacts` represents the set of build output artifacts that CodeBuild uploads to the output bucket. `files` represents the files to include in the build output. CodeBuild uploads the single `messageUtil-1.0.jar` file found in the `target` relative directory in the build environment.

The file name `messageUtil-1.0.jar` and the directory name `target` are based on the way Apache Maven creates and stores build output artifacts for this example only. In your own builds, these file names and directories are different.

For more information, see the Buildspec reference.

At this point, your directory structure should look like this.

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

Step 3: Create two S3 buckets

(Previous step: Step 2: Create the buildspec file)

Although you can use a single bucket for this tutorial, two buckets makes it easier to see where the build input is coming from and where the build output is going.

- One of these buckets (the *input bucket*) stores the build input. In this tutorial, the name of this input bucket is `codebuild-`*`region-ID`*`-`*`account-ID`*`-input-bucket`, where *`region-ID`* is the AWS Region of the bucket and *`account-ID`* is your AWS account ID.
- The other bucket (the *output bucket*) stores the build output. In this tutorial, the name of this output bucket is `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`.

If you chose different names for these buckets, be sure to use them throughout this tutorial.

These two buckets must be in the same AWS Region as your builds. For example, if you instruct CodeBuild to run a build in the US East (Ohio) Region, these buckets must also be in the US East (Ohio) Region.

For more information, see Creating a Bucket in the *Amazon Simple Storage Service User Guide*.

> **Note**
>
> Although CodeBuild also supports build input stored in CodeCommit, GitHub, and Bitbucket repositories, this tutorial does not show you how to use them. For more information, see Plan a build.

Step 4: Upload the source code and the buildspec file

(Previous step: Step 3: Create two S3 buckets)

In this step, you add the source code and build spec file to the input bucket.

Using your operating system's zip utility, create a file named `MessageUtil.zip` that includes `MessageUtil.java`, `TestMessageUtil.java`, `pom.xml`, and `buildspec.yml`.

The `MessageUtil.zip` file's directory structure must look like this.

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

> **Important**
>
> Do not include the *(root directory name)* directory, only the directories and files in the *(root directory name)* directory.

Upload the `MessageUtil.zip` file to the input bucket named `codebuild-`*region-ID*`-`*account-ID*`-input-bucket`.

Important

For CodeCommit, GitHub, and Bitbucket repositories, by convention, you must store a build spec file named `buildspec.yml` in the root (top level) of each repository or include the build spec declaration as part of the build project definition. Do not create a ZIP file that contains the repository's source code and build spec file.
For build input stored in S3 buckets only, you must create a ZIP file that contains the source code and, by convention, a build spec file named `buildspec.yml` at the root (top level) or include the build spec declaration as part of the build project definition.
If you want to use a different name for your build spec file, or you want to reference a build spec in a location other than the root, you can specify a build spec override as part of the build project definition. For more information, see Buildspec file name and storage location.

Step 5: Create the build project

(Previous step: Step 4: Upload the source code and the buildspec file)

In this step, you create a build project that AWS CodeBuild uses to run the build. A *build project* includes information about how to run a build, including where to get the source code, which build environment to use, which build commands to run, and where to store the build output. A *build environment* represents a combination of operating system, programming language runtime, and tools that CodeBuild uses to run a build. The build environment is expressed as a Docker image. For more information, see Docker overview on the Docker Docs website.

For this build environment, you instruct CodeBuild to use a Docker image that contains a version of the Java Development Kit (JDK) and Apache Maven.

To create the build project

1. Use the AWS CLI to run the **create-project** command:

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

JSON-formatted data appears in the output. Copy the data to a file named `create-project.json` in a location on the local computer or instance where the AWS CLI is installed. If you choose to use a different file name, be sure to use it throughout this tutorial.

Modify the copied data to follow this format, and then save your results:

```
{
  "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"
}
```

Replace *serviceIAMRole* with the Amazon Resource Name (ARN) of a CodeBuild service role (for example, `arn:aws:iam::`*account-ID*`:role/`*role-name*). To create one, see Allow CodeBuild to interact with other AWS services.

In this data:

- `name` represents a required identifier for this build project (in this example, `codebuild-demo-project`). Build project names must be unique across all build projects in your account.
- For `source`, `type` is a required value that represents the source code's repository type (in this example, `S3` for an Amazon S3 bucket).
- For `source`, `location` represents the path to the source code (in this example, the input bucket name followed by the ZIP file name).
- For `artifacts`, `type` is a required value that represents the build output artifact's repository type (in this example, `S3` for an Amazon S3 bucket).
- For `artifacts`, `location` represents the name of the output bucket you created or identified earlier (in this example, `codebuild-`*region-ID*`-`*account-ID*`-output-bucket`).

- For `environment`, `type` is a required value that represents the type of build environment (in this example, `LINUX_CONTAINER`).
- For `environment`, `image` is a required value that represents the Docker image name and tag combination this build project uses, as specified by the Docker image repository type (in this example, `aws/codebuild/standard:5.0` for a Docker image in the CodeBuild Docker images repository). `aws/codebuild/standard` is the name of the Docker image. `5.0` is the tag of the Docker image.

  To find more Docker images you can use in your scenarios, see the Build environment reference.
- For `environment`, `computeType` is a required value that represents the computing resources CodeBuild uses (in this example, `BUILD_GENERAL1_SMALL`).

> **Note**
>
> Other available values in the original JSON-formatted data, such as `description`, `buildspec`, `auth` (including `type` and `resource`), `path`, `namespaceType`, `name` (for `artifacts`), `packaging`, `environmentVariables` (including `name` and `value`), `timeoutInMinutes`, `encryptionKey`, and `tags` (including `key` and `value`) are optional. They are not used in this tutorial, so they are not shown here. For more information, see Create a build project (AWS CLI).

2. Switch to the directory that contains the file you just saved, and then run the **create-project** command again.

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

If successful, data similar to this appears in the output.

```
{
  "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` represents information about this build project.
  - `tags` represents any tags that were declared.
  - `packaging` represents how the build output artifact is stored in the output bucket. `NONE` means that a folder is created in the output bucket. The build output artifact is stored in that folder.
  - `lastModified` represents the time, in Unix time format, when information about the build project was last changed.
  - `timeoutInMinutes` represents the number of minutes after which CodeBuild stops the build if the build has not been completed. (The default is 60 minutes.)
  - `created` represents the time, in Unix time format, when the build project was created.
  - `environmentVariables` represents any environment variables that were declared and are available for CodeBuild to use during the build.
  - `encryptionKey` represents the ARN of the customer managed key that CodeBuild used to encrypt the build output artifact.
  - `arn` represents the ARN of the build project.

Note

After you run the **create-project** command, an error message similar to the following might be output: **User: *user-ARN* is not authorized to perform: codebuild:CreateProject**. This is most likely because you configured the AWS CLI with the credentials of an user who does not have sufficient permissions to use CodeBuild to create build projects. To fix this, configure the AWS CLI with credentials belonging to one of the following IAM entities:

- An administrator user in your AWS account. For more information, see Creating your first AWS account root user and group in the *user Guide*.
- An user in your AWS account with the `AWSCodeBuildAdminAccess`, `AmazonS3ReadOnlyAccess`, and `IAMFullAccess` managed policies attached to that user or to an IAM group that the user belongs to. If you do not have an user or group in your AWS account with these permissions, and you cannot add these permissions to your user or group, contact your AWS account administrator for assistance. For more information, see AWS managed (predefined) policies for AWS CodeBuild.

Step 6: Run the build

(Previous step: Step 5: Create the build project)

In this step, you instruct AWS CodeBuild to run the build with the settings in the build project.

To run the build

1. Use the AWS CLI to run the **start-build** command:

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

   Replace *project-name* with your build project name from the previous step (for example, `codebuild-demo-project`).
2. If successful, data similar to the following appears in the output:

   ```
   {
     "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` represents information about this build.
  - `buildComplete` represents whether the build was completed (`true`). Otherwise, `false`.
  - `initiator` represents the entity that started the build.
  - `artifacts` represents information about the build output, including its location.
  - `projectName` represents the name of the build project.
  - `buildStatus` represents the current build status when the **start-build** command was run.
  - `currentPhase` represents the current build phase when the **start-build** command was run.
  - `startTime` represents the time, in Unix time format, when the build process started.
  - `id` represents the ID of the build.
  - `arn` represents the ARN of the build.

Make a note of the `id` value. You need it in the next step.

Step 7: View summarized build information

(Previous step: Step 6: Run the build)

In this step, you view summarized information about the status of your build.

To view summarized build information

- Use the AWS CLI to run the **batch-get-builds** command.

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

Replace *id* with the `id` value that appeared in the output of the previous step.

If successful, data similar to this appears in the output.

```
{
  "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` represents the build IDs for any builds where information is not available. In this example, it should be empty.
- `builds` represents information about each build where information is available. In this example, information about only one build appears in the output.
  - `phases` represents the set of build phases CodeBuild runs during the build process. Information about each build phase is listed separately as `startTime`, `endTime`, and `durationInSeconds` (when the build phase started and ended, expressed in Unix time format, and how long it lasted, in seconds), and `phaseType` such as (`SUBMITTED`, `PROVISIONING`, `DOWNLOAD_SOURCE`, `INSTALL`, `PRE_BUILD`, `BUILD`, `POST_BUILD`, `UPLOAD_ARTIFACTS`, `FINALIZING`, or `COMPLETED`) and `phaseStatus` (such as `SUCCEEDED`, `FAILED`, `FAULT`, `TIMED_OUT`, `IN_PROGRESS`, or `STOPPED`). The first time you run the **batch-get-builds** command, there might not be many (or any) phases. After

subsequent runs of the **batch-get-builds** command with the same build ID, more build phases should appear in the output.

- `logs` represents information in Amazon CloudWatch Logs about the build's logs.
- `md5sum` and `sha256sum` represent MD5 and SHA-256 hashes of the build's output artifact. These appear in the output only if the build project's `packaging` value is set to `ZIP`. (You did not set this value in this tutorial.) You can use these hashes along with a checksum tool to confirm file integrity and authenticity.

  > **Note**
  >
  > You can also use the Amazon S3 console to view these hashes. Select the box next to the build output artifact, choose **Actions**, and then choose **Properties**. In the **Properties** pane, expand **Metadata**, and view the values for **x-amz-meta-codebuild-content-md5** and **x-amz-meta-codebuild-content-sha256**. (In the Amazon S3 console, the build output artifact's **ETag** value should not be interpreted to be either the MD5 or SHA-256 hash.)
  > If you use the AWS SDKs to get these hashes, the values are named `codebuild-content-md5` and `codebuild-content-sha256`.

- `endTime` represents the time, in Unix time format, when the build process ended.

> **Note**
>
> Amazon S3 metadata has a CodeBuild header named `x-amz-meta-codebuild-buildarn` which contains the `buildArn` of the CodeBuild build that publishes artifacts to Amazon S3. The `buildArn` is added to allow source tracking for notifications and to reference which build the artifact is generated from.

Step 8: View detailed build information

(Previous step: Step 7: View summarized build information)

In this step, you view detailed information about your build in CloudWatch Logs.

Note

To protect sensitive information, the following are hidden in CodeBuild logs:

- AWS access key IDs. For more information, see Managing Access Keys for IAM Users in the *AWS Identity and Access Management User Guide*.
- Strings specified using the Parameter Store. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the *Amazon EC2 Systems Manager User Guide*.
- Strings specified using AWS Secrets Manager. For more information, see Key management.

To view detailed build information

1. Use your web browser to go to the `deepLink` location that appeared in the output in the previous step (for example, `https://console.aws.amazon.com/cloudwatch/home?region=region-ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE`).
2. In the CloudWatch Logs log stream, you can browse the log events. By default, only the last set of log events is displayed. To see earlier log events, scroll to the beginning of the list.
3. In this tutorial, most of the log events contain verbose information about CodeBuild downloading and installing build dependency files into its build environment, which you probably don't care about. You can use the **Filter events** box to reduce the information displayed. For example, if you enter `"[INFO]"` in **Filter events**, only those events that contain `[INFO]` are displayed. For more information, see Filter and pattern syntax in the *Amazon CloudWatch User Guide*.

These portions of a CloudWatch Logs log stream pertain to this tutorial.

```
...
[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
```

In this example, CodeBuild successfully completed the pre-build, build, and post-build build phases. It ran the unit tests and successfully built the `messageUtil-1.0.jar` file.

Step 9: Get the build output artifact

(Previous step: Step 8: View detailed build information)

In this step, you get the `messageUtil-1.0.jar` file that CodeBuild built and uploaded to the output bucket.

You can use the CodeBuild console or the Amazon S3 console to complete this step.

To get the build output artifact (AWS CodeBuild console)

1. With the CodeBuild console still open and the build details page still displayed from the previous step, choose the **Build details** tab and scroll down to the **Artifacts** section.

   > **Note**
   >
   > If the build details page is not displayed, in the navigation bar, choose **Build history**, and then choose the **Build run** link.

2. The link to the Amazon S3 folder is under the **Artifacts upload location**. This link opens the folder in Amazon S3 where you find the `messageUtil-1.0.jar` build output artifact file.

To get the build output artifact (Amazon S3 console)

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Open `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`.
3. Open the `codebuild-demo-project` folder.
4. Open the `target` folder, where you find the `messageUtil-1.0.jar` build output artifact file.

Step 10: Delete the S3 buckets

(Previous step: Step 9: Get the build output artifact)

To prevent ongoing charges to your AWS account, you can delete the input and output buckets used in this tutorial. For instructions, see Deleting or Emptying a Bucket in the *Amazon Simple Storage Service User Guide*.

If you are using the IAM user or an administrator IAM user to delete these buckets, the user must have more access permissions. Add the following statement between the markers (*### BEGIN ADDING STATEMENT HERE ###* and *### END ADDING STATEMENTS HERE ###*) to an existing access policy for the user.

The ellipses (...) in this statement are used for brevity. Do not remove any statements in the existing access policy. Do not enter these ellipses into the policy.

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

Wrapping up

In this tutorial, you used AWS CodeBuild to build a set of Java class files into a JAR file. You then viewed the build's results.

You can now try using CodeBuild in your own scenarios. Follow the instructions in Plan a build. If you don't feel ready yet, you might want to try building some of the samples. For more information, see Use case-based samples for CodeBuild.

Use case-based samples for CodeBuild

You can use these use case-based samples to experiment with AWS CodeBuild:

Cross-service samples

A list of cross-service samples to experiment with AWS CodeBuild.

Build badges sample

Shows how to set up CodeBuild with build badges.

Test report sample

Uses the AWS CLI to create, run, and view the results of a test report.

Docker samples for CodeBuild

Shows how to use custom Docker images, publish Docker images to a repository in Amazon ECR, and use Docker images in a private registry.

Host build output in an S3 bucket

Shows how to create a static website in an S3 bucket using unencrypted build artifacts.

Multiple inputs and outputs sample

Shows how to use multiple input sources and multiple output artifacts in a build project.

Parallel test execution samples

Shows how to use the `codebuild-tests-run` CLI command to split and run tests across parallel execution environments.

Runtime versions in buildspec file sample

Shows how to specify runtimes and their versions in the buildspec file.

Source version sample

Shows how to use a specific version of your source in a CodeBuild build project.

Third-party source repository samples for CodeBuild

Shows how to create BitBucket, GitHub Enterprise Server, and GitHub pull requests with webhooks using CodeBuild.

Set artifact names at build time using semantic versioning

Shows how to use semantic versioning to create an artifact name at build time.

Cross-service samples for CodeBuild

You can use these cross-service samples to experiment with AWS CodeBuild:

Amazon ECR sample

Uses a Docker image in an Amazon ECR repository to use Apache Maven to produce a single JAR file. The sample instructions will show you how to create and push a Docker image to Amazon ECR, create a Go project, build the project, run the project, and set up permissions to allow CodeBuild to connect to Amazon ECR.

Amazon EFS sample

Shows how to configure a buildspec file so that a CodeBuild project mounts and builds on an Amazon EFS file system. The sample instructions will show you how to create a Amazon VPC, create file system in the Amazon VPC, create and build a project that uses the Amazon VPC, and then review the generated project file and variables.

AWS CodePipeline samples

Shows how to use AWS CodePipeline to create a build with batch builds as well as multiple input sources and multiple output artifacts. Included in this section are example JSON files that show pipeline structures that create batch builds with separate artifacts, and combined artifacts. An additonal JSON sample is provided that show the pipeline structure with multiple input sources and multiple output artifacts.

AWS Config sample

Shows how to set up AWS Config. Lists which CodeBuild resources are tracked and describes how to look up CodeBuild projects in AWS Config. The sample instructions will show you the prerequisites for integrating with AWS Config, the steps to set up AWS Config, and the steps to look up CodeBuild projects and data in AWS Config.

Build notifications sample

Uses Apache Maven to produce a single JAR file. Sends a build notification to subscribers of an Amazon SNS topic. The sample instructions show you how to set up permissions so that

CodeBuild can communicate with Amazon SNS and CloudWatch, how to create and identify CodeBuild topics in Amazon SNS, how to subscribe recipients to the topic, and how to set up rules in CloudWatch.

Amazon ECR sample for CodeBuild

This sample uses a Docker image in an Amazon Elastic Container Registry (Amazon ECR) image repository to build a sample Go project.

> **Important**
>
> Running this sample might result in charges to your AWS account. These include possible charges for AWS CodeBuild and for AWS resources and actions related to Amazon S3, AWS KMS, CloudWatch Logs, and Amazon ECR. For more information, see CodeBuild pricing, Amazon S3 pricing, AWS Key Management Service pricing, Amazon CloudWatch pricing, and Amazon Elastic Container Registry pricing.

Topics

Run the Amazon ECR sample

Use the following instructions to run the Amazon ECR sample for CodeBuild.

To run this sample

1. To create and push the Docker image to your image repository in Amazon ECR, complete the steps in the Run the 'Publish Docker image to Amazon ECR' sample section of the 'Publish Docker image to Amazon ECR' sample.
2. Create a Go project:

   a. Create the files as described in the Go project structure and Go project files sections of this topic, and then upload them to an S3 input bucket or an AWS CodeCommit, GitHub, or Bitbucket repository.

> **Important**
>
> Do not upload *(root directory name)*, just the files inside of *(root directory name)*.
> If you are using an S3 input bucket, be sure to create a ZIP file that contains the files, and then upload it to the input bucket. Do not add *(root directory name)* to the ZIP file, just the files inside of *(root directory name)*.

b. Create a build project, run the build, and view related build information.

If you use the AWS CLI to create the build project, the JSON-formatted input to the `create-project` command might look similar to this. (Replace the placeholders with your own values.)

```
{
  "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. To get the build output artifact, open your S3 output bucket.

d. Download the *GoOutputArtifact*`.zip` file to your local computer or instance, and then extract the contents of the file. In the extracted contents, get the `hello` file.

3. If one of the following is true, you must add permissions to your image repository in Amazon ECR so that AWS CodeBuild can pull its Docker image into the build environment.

   - Your project uses CodeBuild credentials to pull Amazon ECR images. This is denoted by a value of `CODEBUILD` in the `imagePullCredentialsType` attribute of your `ProjectEnvironment`.
   - Your project uses a cross-account Amazon ECR image. In this case, your project must use its service role to pull Amazon ECR images. To enable this behavior, set the `imagePullCredentialsType` attribute of your `ProjectEnvironment` to `SERVICE_ROLE`.

   1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/.
   2. In the list of repository names, choose the name of the repository you created or selected.
   3. From the navigation pane, choose **Permissions**, choose **Edit**, and then choose **Add statement**.
   4. For **Statement name**, enter an identifier (for example, **`CodeBuildAccess`**).
   5. For **Effect**, leave **Allow** selected. This indicates that you want to allow access to another AWS account.
   6. For **Principal**, do one of the following:
      - If your project uses CodeBuild credentials to pull an Amazon ECR image, in **Service principal**, enter **`codebuild.amazonaws.com`**.
      - If your project uses a cross-account Amazon ECR image, for **AWS account IDs**, enter IDs of the AWS accounts that you want to give access.
   7. Skip the **All IAM entities** list.
   8. For **Action**, select the pull-only actions: **ecr:GetDownloadUrlForLayer**, **ecr:BatchGetImage**, and **ecr:BatchCheckLayerAvailability**.
   9. For **Conditions**, add the following:

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

10. Choose **Save**.

This policy is displayed in **Permissions**. The principal is what you entered for **Principal** in step 3 of this procedure:

- If your project uses CodeBuild credentials to pull an Amazon ECR image, `"codebuild.amazonaws.com"` appears under **Service principals**.
- If your project uses a cross-account Amazon ECR image, the ID of the AWS account that you want to give access appears under **AWS Account IDs**.

The following sample policy uses both CodeBuild credentials and a cross-account Amazon ECR image.

```
{
   "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"
         ]
      }
   ]
}
```

- If your projects use CodeBuild credentials and you would like your CodeBuild projects to have open access to the Amazon ECR repository, you can omit the `Condition` keys and add the following sample policy.

```
{
  "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. Create a build project, run the build, and view build information.

If you use the AWS CLI to create the build project, the JSON-formatted input to the `create-project` command might look similar to this. (Replace the placeholders with your own values.)

```
{
  "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. To get the build output artifact, open your S3 output bucket.
6. Download the *GoOutputArtifact*`.zip` file to your local computer or instance, and then extract the contents of the *GoOutputArtifact*`.zip` file. In the extracted contents, get the `hello` file.

Go project structure

This sample assumes this directory structure.

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

Go project files

This sample uses these files.

`buildspec.yml` (in *(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` (in *(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)
}
```

Amazon Elastic File System sample for AWS CodeBuild

You might want to create your AWS CodeBuild builds on Amazon Elastic File System, a scalable, shared file service for Amazon EC2 instances. The storage capacity with Amazon EFS is elastic, so

it grows or shrinks as files are added and removed. It has a simple web services interface that you can use to create and configure file systems. It also manages all of the file storage infrastructure for you, so you do not need to worry about deploying, patching, or maintaining file system configurations. For more information, see What is Amazon Elastic File System? in the *Amazon Elastic File System User Guide*.

This sample shows you how to configure a CodeBuild project so that it mounts and then builds a Java application to an Amazon EFS file system. Before you begin, you must have a Java application ready to build that is uploaded to an S3 input bucket or an AWS CodeCommit, GitHub, GitHub Enterprise Server, or Bitbucket repository.

Data in transit for your file system is encrypted. To encrypt data in transit using a different image, see Encrypting data in transit.

Topics

Use AWS CodeBuild with Amazon Elastic File System

The sample covers the four high-level steps required to use Amazon EFS with AWS CodeBuild. They are:

1. Create a virtual private cloud (VPC) in your AWS account.
2. Create a file system that uses this VPC.
3. Create and build a CodeBuild project that uses the VPC. The CodeBuild project uses the following to identify the file system:
   - A unique file system identifier. You choose the identifier when you specify the file system in your build project.
   - The file system ID. The ID is displayed when you view your file system in the Amazon EFS console.
   - A mount point. This is a directory in your Docker container that mounts the file system.
   - Mount options. These include details about how to mount the file system.
4. Review the build project to ensure that the correct project files and variables were generated.

> **Note**
>
> A file system created in Amazon EFS is supported on Linux platforms only.

Topics

Step 1: Create a VPC using AWS CloudFormation

Create your VPC with an AWS CloudFormation template.

1. Follow the instructions in AWS CloudFormation VPC template to use AWS CloudFormation to create a VPC.

   > **Note**
   >
   > The VPC created by this AWS CloudFormation template has two private subnets and two public subnets. You must only use private subnets when you use AWS CodeBuild to mount the file system you created in Amazon EFS. If you use one of the public subnets, the build fails.

2. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. Choose the VPC you created with AWS CloudFormation.
4. On the **Description** tab, make a note of the name of your VPC and its ID. Both are required when you create your AWS CodeBuild project later in this sample.

Step 2: Create an Amazon Elastic File System file system with your VPC

Create a simple Amazon EFS file system for this sample using the VPC you created earlier.

1. Sign in to the AWS Management Console and open the Amazon EFS console at https://console.aws.amazon.com/efs/.
2. Choose **Create file system**.
3. From **VPC**, choose the VPC name you noted earlier in this sample.
4. Leave the Availability Zones associated with your subnets selected.
5. Choose **Next Step**.
6. In **Add tags**, for the default **Name** key, in **Value**, enter the name of your Amazon EFS file system.
7. Keep **Bursting** and **General Purpose** selected as your default performance and throughput modes, and then choose **Next Step**.
8. For **Configure client access**, choose **Next Step**.
9. Choose **Create File System**.
10. (Optional) We recommend adding a policy to your Amazon EFS file system that enforces encryption of data in transit. In the Amazon EFS console, choose **File system policy**, choose **Edit**, select the box labeled **Enforce in-transit encryption for all clients**, and then choose **Save**.

Step 3: Create a CodeBuild project to use with Amazon EFS

Create a AWS CodeBuild project that uses the VPC you created earlier in this sample. When the build is run, it mounts the Amazon EFS file system created earlier. Next, it stores the .jar file created by your Java application in your file system's mount point directory.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. From the navigation pane, choose **Build projects**, and then choose **Create build project**.
3. In **Project name**, enter a name for your project.
4. From **Source provider**, choose the repository that contains the Java application you want to build.
5. Enter information, such as a repository URL, that CodeBuild uses to locate your application. The options are different for each source provider. For more information, see Choose source provider.
6. From **Environment image**, choose **Managed image**.

7. From **Operating system**, choose **Amazon Linux 2**.
8. From **Runtime(s)**, choose **Standard**.
9. From **Image**, choose **aws/codebuild/amazonlinux-x86_64-standard:4.0**.
10. From **Environment type**, choose **Linux**.
11. Under **Service role**, choose **New service role**. In **Role name**, enter a name for the role CodeBuild creates for you.
12. Expand **Additional configuration**.
13. Select **Enable this flag if you want to build Docker images or want your builds to get elevated privileges**.

> **Note**
>
> By default, Docker daemon is enabled for non-VPC builds. If you would like to use Docker containers for VPC builds, see Runtime Privilege and Linux Capabilities on the Docker Docs website and enable privileged mode. Also, Windows does not support privileged mode.

14. From **VPC**, choose the VPC ID.
15. From **Subnets**, choose one or more of the private subnets associated with your VPC. You must use private subnets in a build that mounts an Amazon EFS file system. If you use a public subnet, the build fails.
16. From **Security Groups**, choose the default security group.
17. In **File systems**, enter the following information:

    - For **Identifier**, enter a unique file system identifier. It must be fewer than 129 characters and contain only alphanumeric characters and underscores. CodeBuild uses this identifier to create an environment variable that identifies the elastic file system. The environment variable format is `CODEBUILD_<file_system_identifier>` in capital letters. For example, if you enter `my_efs`, the environment variable is `CODEBUILD_MY_EFS`.
    - For **ID**, choose the file system ID.
    - (Optional) Enter a directory in the file system. CodeBuild mounts this directory. If you leave **Directory path** blank, CodeBuild mounts the entire file system. The path is relative to the root of the file system.
    - For **Mount point**, enter the absolute path of the directory in your build container where the file system is mounted. If this directory does not exist, CodeBuild creates it during the build.

- (Optional) Enter mount options. If you leave **Mount options** blank, CodeBuild uses its default mount options:

```
nfsvers=4.1
rsize=1048576
wsize=1048576
hard
timeo=600
retrans=2
```

  For more information, see Recommended NFS Mount Options in the *Amazon Elastic File System User Guide*.

18. For **Build specification**, choose **Insert build commands**, and then choose **Switch to editor**.
19. Enter the following build spec commands into the editor. Replace *<file_system_identifier>* with the identifier you entered in step 17. Use capital letters (for example, CODEBUILD_MY_EFS).

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

20. Use the default values for all other settings, and then choose **Create build project**. When your build is complete, the console page for your project is displayed.
21. Choose **Start build**.

Step 4: Review the build project

After your AWS CodeBuild project is built:

- You have a .jar file created by your Java application that is built to your Amazon EFS file system under your mount point directory.

- An environment variable that identifies your file system is created using the file system identifier you entered when you created the project.

For more information, see Mounting file systems in the *Amazon Elastic File System User Guide*.

Troubleshoot the Amazon EFS integration

The following are errors you might encounter when setting up Amazon EFS with CodeBuild.

Topics

CLIENT_ERROR: mounting '127.0.0.1:/' failed. permission denied

IAM authorization is not supported for mounting Amazon EFS with CodeBuild. If you are using a custom Amazon EFS file system policy, you will need to grant read and write access to all IAM principals. For example:

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

CLIENT_ERROR: mounting '127.0.0.1:/' failed. connection reset by peer

There are two possible causes for this error:

- The CodeBuild VPC subnet is in a different availability zone than the Amazon EFS mount target. You can resolve this by adding a VPC subnet in the same availability zone as the Amazon EFS mount target.
- The security group does not have permissions to communicate with Amazon EFS. You can resolve this by adding an inbound rule to allow all traffic from either the VPC (add the primary CIDR block for your VPC), or the security group itself.

VPC_CLIENT_ERROR: Unexpected EC2 error: UnauthorizedOperation

This error occurs when all of the subnets in your VPC configuration for the CodeBuild project are public subnets. You must have at least one private subnet in the VPC to ensure network connectivity.

AWS CodePipeline samples for CodeBuild

This section describes sample integrations between CodePipeline and CodeBuild.

Sample	Description
Samples of CodePipeline/CodeBuild integrations and batch builds	These samples demonstrate how to use AWS CodePipeline to create a build project that uses batch builds.
Sample of a CodePipeline/CodeBuild integration with multiple input sources and output artifacts	This sample demonstrates how to use AWS CodePipeline to create a build project that uses multiple input sources to create multiple output artifacts.

Samples of CodePipeline/CodeBuild integrations and batch builds

AWS CodeBuild supports batch builds. The following samples demonstrate how to use AWS CodePipeline to create a build project that uses batch builds.

You can use a JSON-formatted file that defines the structure of your pipeline, and then use it with the AWS CLI to create the pipeline. For more information, see AWS CodePipeline Pipeline structure reference in the *AWS CodePipeline User Guide*.

Batch build with individual artifacts

Use the following JSON file as an example of a pipeline structure that creates a batch build with separate artifacts. To enable batch builds in CodePipeline, set the `BatchEnabled` parameter of the `configuration` object to `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
  }
}
```

The following is an example of a CodeBuild buildspec file that will work with this pipeline configuration.

```
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
```

The names of the output artifacts specified in the pipeline's JSON file must match the identifier of the builds and artifacts defined in your buildspec file. The syntax is *buildIdentifier* for the primary artifacts, and *buildIdentifier_artifactIdentifier* for the secondary artifacts.

For example, for output artifact name `build1`, CodeBuild will upload the primary artifact of `build1` to the location of `build1`. For output name `build1_artifact1`, CodeBuild will upload the secondary artifact `artifact1` of `build1` to the location of `build1_artifact1`, and so on. If only one output location is specified, the name should be *buildIdentifier* only.

After you create the JSON file, you can create your pipeline. Use the AWS CLI to run the **create-pipeline** command and pass the file to the `--cli-input-json` parameter. For more information, see Create a pipeline (CLI) in the *AWS CodePipeline User Guide*.

Batch build with combined artifacts

Use the following JSON file as an example of a pipeline structure that creates a batch build with combined artifacts. To enable batch builds in CodePipeline, set the `BatchEnabled` parameter of the `configuration` object to `true`. To combine the build artifacts into the same location, set the `CombineArtifacts` parameter of the `configuration` object to `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": "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
 }
}
```

The following is an example of a CodeBuild buildspec file that will work with this pipeline configuration.

```
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
```

If combined artifacts is enabled for the batch build, there is only one output allowed. CodeBuild will combine the primary artifacts of all the builds into one single ZIP file.

After you create the JSON file, you can create your pipeline. Use the AWS CLI to run the **create-pipeline** command and pass the file to the `--cli-input-json` parameter. For more information, see Create a pipeline (CLI) in the *AWS CodePipeline User Guide*.

Sample of a CodePipeline/CodeBuild integration with multiple input sources and output artifacts

An AWS CodeBuild project can take more than one input source. It can also create more than one output artifact. This sample demonstrates how to use AWS CodePipeline to create a build project that uses multiple input sources to create multiple output artifacts. For more information, see Multiple input sources and output artifacts sample.

You can use a JSON-formatted file that defines the structure of your pipeline, and then use it with the AWS CLI to create the pipeline. Use the following JSON file as an example of a pipeline structure that creates a build with more than one input source and more than one output artifact. Later in this sample you see how this file specifies the multiple inputs and outputs. For more information, see CodePipeline pipeline structure reference in the *AWS CodePipeline User Guide*.

```
{
 "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
 }
}
```

In this JSON file:

- One of your input sources must be designated the `PrimarySource`. This source is the directory where CodeBuild looks for and runs your buildspec file. The keyword `PrimarySource` is used to specify the primary source in the `configuration` section of the CodeBuild stage in the JSON file.

- Each input source is installed in its own directory. This directory is stored in the built-in environment variable `$CODEBUILD_SRC_DIR` for the primary source and `$CODEBUILD_SRC_DIR_yourInputArtifactName` for all other sources. For the pipeline in this sample, the two input source directories are `$CODEBUILD_SRC_DIR` and `$CODEBUILD_SRC_DIR_source2`. For more information, see Environment variables in build environments.
- The names of the output artifacts specified in the pipeline's JSON file must match the names of the secondary artifacts defined in your buildspec file. This pipeline uses the following buildspec file. For more information, see Buildspec syntax.

```
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
```

After you create the JSON file, you can create your pipeline. Use the AWS CLI to run the **create-pipeline** command and pass the file to the `--cli-input-json` parameter. For more information, see Create a pipeline (CLI) in the *AWS CodePipeline User Guide*.

AWS Config sample with CodeBuild

AWS Config provides an inventory of your AWS resources and a history of configuration changes to these resources. AWS Config now supports AWS CodeBuild as an AWS resource, which means

the service can track your CodeBuild projects. For more information about AWS Config, see What is AWS Config? in the *AWS Config Developer Guide*.

You can see the following information about CodeBuild resources on the **Resource Inventory** page in the AWS Config console:

- A timeline of your CodeBuild configuration changes.
- Configuration details for each CodeBuild project.
- Relationships with other AWS resources.
- A list of changes to your CodeBuild projects.

Topics

Use CodeBuild with AWS Config

The procedures in this topic show you how to set up AWS Config and look up CodeBuild projects.

Topics

Prerequisites

Create your AWS CodeBuild project. For instructions, see Create a build project.

Step 1: Set up AWS Config

- Setting up AWS Config (console)
- Setting up AWS Config (AWS CLI)

> **Note**
>
> After you complete setup, it might take up to 10 minutes before you can see AWS CodeBuild projects in the AWS Config console.

Step 2: Look up AWS CodeBuild projects

1. Sign in to the AWS Management Console and open the AWS Config console at https://console.aws.amazon.com/config.
2. On the **Resource inventory** page, select **AWS CodeBuild Project** under **Resource type**. Scroll down and select the **CodeBuild project** check box.
3. Choose **Look up**.
4. After the list of CodeBuild projects is added, choose the CodeBuild project name link in the **Config timeline** column.

Step 3: View AWS CodeBuild data in the AWS Config console

When you look up resources on the **Resource inventory** page, you can choose the AWS Config timeline to view details about your CodeBuild project. The details page for a resource provides information about the configuration, relationships, and number of changes made to that resource.

The blocks at the top of the page are collectively called the timeline. The timeline shows the date and time that the recording was made.

For more information, see Viewing configuration details in the AWS Config console in the *AWS Config Developer Guide*.

Build notifications sample for CodeBuild

Amazon CloudWatch Events has built-in support for AWS CodeBuild. CloudWatch Events is a stream of system events describing changes in your AWS resources. With CloudWatch Events, you write declarative rules to associate events of interest with automated actions to be taken. This sample uses Amazon CloudWatch Events and Amazon Simple Notification Service (Amazon SNS) to send build notifications to subscribers whenever builds succeed, fail, go from one build phase to another, or any combination of these events.

Important

Running this sample might result in charges to your AWS account. These include possible charges for CodeBuild and for AWS resources and actions related to Amazon CloudWatch and Amazon SNS. For more information, see CodeBuild pricing, Amazon CloudWatch pricing, and Amazon SNS pricing.

Topics

Run the build notifications sample

Use the following procedure to run the build notifications sample.

To run this sample

1. If you already have a topic set up and subscribed to in Amazon SNS that you want to use for this sample, skip ahead to step 4. Otherwise, if you are using an IAM user instead of an AWS root account or an administrator user to work with Amazon SNS, add the following statement (between *### BEGIN ADDING STATEMENT HERE ###* and *### END ADDING STATEMENT HERE ###*) to the user (or IAM group the user is associated with). Using an AWS root account is not recommended. This statement enables viewing, creating, subscribing, and testing the sending of notifications to topics in Amazon SNS. Ellipses (`...`) are used for brevity and to help you locate where to add the statement. Do not remove any statements, and do not type these ellipses into the existing policy.

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "sns:CreateTopic",
        "sns:GetTopicAttributes",
        "sns:List*",
        "sns:Publish",
        "sns:SetTopicAttributes",
        "sns:Subscribe"
```

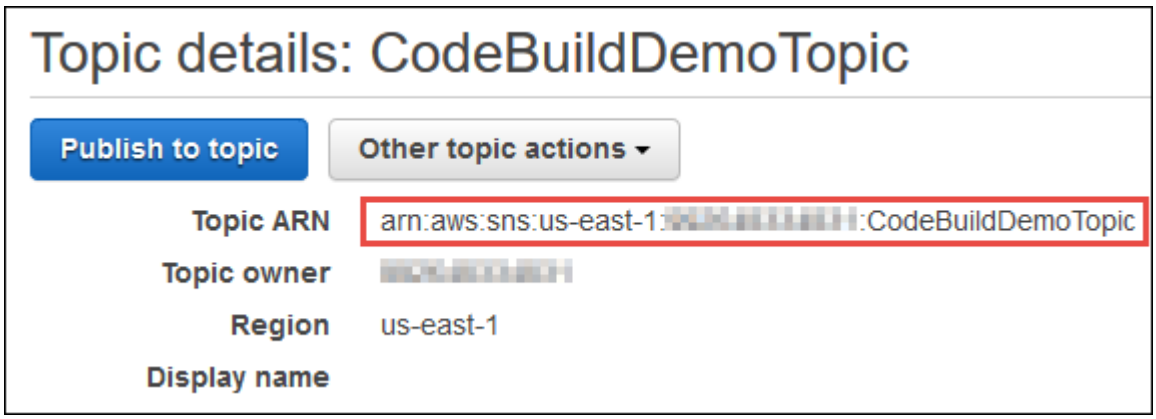
```
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> The IAM entity that modifies this policy must have permission in IAM to modify policies.
> For more information, see Editing customer managed policies or the "To edit or delete an inline policy for a group, user, or role" section in Working with inline policies (console) in the *IAM User Guide*.

2. Create or identify a topic in Amazon SNS. AWS CodeBuild uses CloudWatch Events to send build notifications to this topic through Amazon SNS.

   To create a topic:

   1. Open the Amazon SNS console at https://console.aws.amazon.com/sns.
   2. Choose **Create topic**.
   3. In **Create new topic**, for **Topic name**, enter a name for the topic (for example, **`CodeBuildDemoTopic`**). (If you choose a different name, substitute it throughout this sample.)
   4. Choose **Create topic**.
   5. On the **Topic details: CodeBuildDemoTopic** page, copy the **Topic ARN** value. You need this value for the next step.

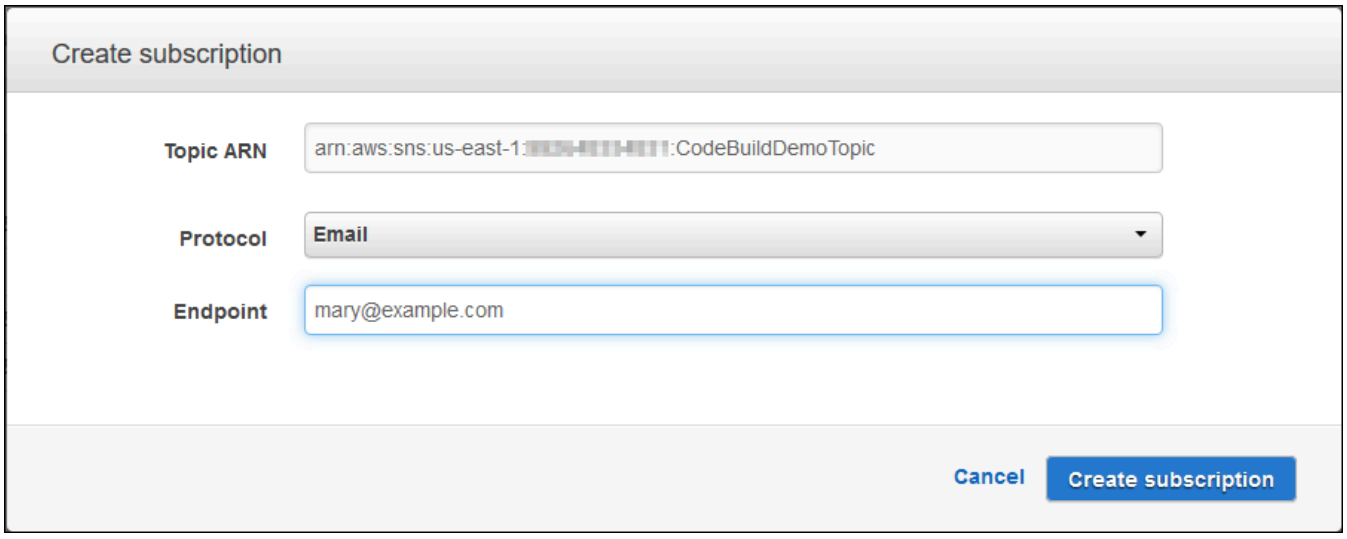


For more information, see Create a topic in the *Amazon SNS Developer Guide*.

3. Subscribe one or more recipients to the topic to receive email notifications.

   To subscribe a recipient to a topic:

   1. With the Amazon SNS console open from the previous step, in the navigation pane, choose **Subscriptions**, and then choose **Create subscription**.
   2. In **Create subscription**, for **Topic ARN**, paste the topic ARN you copied from the previous step.
   3. For **Protocol**, choose **Email**.
   4. For **Endpoint**, enter the recipient's full email address.



   5. Choose **Create Subscription**.

6. Amazon SNS sends a subscription confirmation email to the recipient. To begin receiving email notifications, the recipient must choose the **Confirm subscription** link in the subscription confirmation email. After the recipient clicks the link, if successfully subscribed, Amazon SNS displays a confirmation message in the recipient's web browser.

For more information, see Subscribe to a topic in the *Amazon SNS Developer Guide*.

4. If you are using an user instead of an AWS root account or an administrator user to work with CloudWatch Events, add the following statement (between *### BEGIN ADDING STATEMENT HERE ###* and *### END ADDING STATEMENT HERE ###*) to the user (or IAM group the user is associated with). Using an AWS root account is not recommended. This statement is used to allow the user to work with CloudWatch Events. Ellipses (`...`) are used for brevity and to help you locate where to add the statement. Do not remove any statements, and do not type these ellipses into the existing policy.

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

> **Note**
>
> The IAM entity that modifies this policy must have permission in IAM to modify policies.
> For more information, see Editing customer managed policies or the "To edit or delete an inline policy for a group, user, or role" section in Working with inline policies (console) in the *IAM User Guide*.

5. Create a rule in CloudWatch Events. To do this, open the CloudWatch console, at https://console.aws.amazon.com/cloudwatch.
6. In the navigation pane, under **Events**, choose **Rules**, and then choose **Create rule**.
7. On the **Step 1: Create rule page**, **Event Pattern** and **Build event pattern to match events by service** should already be selected.
8. For **Service Name**, choose **CodeBuild**. For **Event Type**, **All Events** should already be selected.
9. The following code should be displayed in **Event Pattern Preview**:

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

10. Choose **Edit** and replace the code in **Event Pattern Preview** with one of the following two rule patterns.

    This first rule pattern triggers an event when a build starts or completes for the specified build projects in AWS CodeBuild.

```
{
  "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"
    ]
  }
}
```

In the preceding rule, make the following code changes as needed.

- To trigger an event when a build starts or completes, either leave all of the values as shown in the `build-status` array, or remove the `build-status` array altogether.
- To trigger an event only when a build completes, remove `IN_PROGRESS` from the `build-status` array.
- To trigger an event only when a build starts, remove all of the values except `IN_PROGRESS` from the `build-status` array.
- To trigger events for all build projects, remove the `project-name` array altogether.
- To trigger events only for individual build projects, specify the name of each build project in the `project-name` array.

This second rule pattern triggers an event whenever a build moves from one build phase to another for the specified build projects in AWS CodeBuild.

```
{
  "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"
    ]
  }
}
```

In the preceding rule, make the following code changes as needed.

- To trigger an event for every build phase change (which might send up to nine notifications for each build), either leave all of the values as shown in the `completed-phase` array, or remove the `completed-phase` array altogether.
- To trigger events only for individual build phase changes, remove the name of each build phase in the `completed-phase` array that you do not want to trigger an event for.
- To trigger an event for every build phase status change, either leave all of the values as shown in the `completed-phase-status` array, or remove the `completed-phase-status` array altogether.
- To trigger events only for individual build phase status changes, remove the name of each build phase status in the `completed-phase-status` array that you do not want to trigger an event for.
- To trigger events for all build projects, remove the `project-name` array.
- To trigger events for individual build projects, specify the name of each build project in the `project-name` array.

For more information about event patterns, see Event Patterns in the Amazon EventBridge User Guide.

For more information about filtering with event patterns, see Content-based Filtering with Event Patterns in the Amazon EventBridge User Guide.

> **Note**
>
> If you want to trigger events for both build state changes and build phase changes, you must create two separate rules: one for build state changes and another for build

phase changes. If you try to combine both rules into a single rule, the combined rule might produce unexpected results or stop working altogether.

When you have finished replacing the code, choose **Save**.

11. For **Targets**, choose **Add target**.
12. In the list of targets, choose **SNS topic**.
13. For **Topic**, choose the topic you identified or created earlier.
14. Expand **Configure input**, and then choose **Input Transformer**.
15. In the **Input Path** box, enter one of the following input paths.

    For a rule with a `detail-type` value of `CodeBuild Build State Change`, enter the following.

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

    For a rule with a `detail-type` value of `CodeBuild Build Phase Change`, enter the following.

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

    To get other types of information, see the Build notifications input format reference.
16. In the **Input Template** box, enter one of the following input templates.

    For a rule with a `detail-type` value of `CodeBuild Build State Change`, enter the following.

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

    For a rule with a `detail-type` value of `CodeBuild Build Phase Change`, enter the following.

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

17. Choose **Configure details**.
18. On the **Step 2: Configure rule details** page, enter a name and an optional description. For **State**, leave **Enabled** selected.
19. Choose **Create rule**.
20. Create build projects, run the builds, and view build information.
21. Confirm that CodeBuild is now successfully sending build notifications. For example, check to see if the build notification emails are now in your inbox.

To change a rule's behavior, in the CloudWatch console, choose the rule you want to change, choose **Actions**, and then choose **Edit**. Make changes to the rule, choose **Configure details**, and then choose **Update rule**.

To stop using a rule to send build notifications, in the CloudWatch console, choose the rule you want to stop using, choose **Actions**, and then choose **Disable**.

To delete a rule altogether, in the CloudWatch console, choose the rule you want to delete, choose **Actions**, and then choose **Delete**.

Build notifications input format reference

CloudWatch delivers notifications in JSON format.

Build state change notifications use the following format:

```
{
  "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"
  }
}
```

Build phase change notifications use the following format:

```
{
  "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"
  }
}
```

Build badges sample with CodeBuild

AWS CodeBuild now supports the use of build badges, which provide an embeddable, dynamically generated image (*badge*) that displays the status of the latest build for a project. This image is

accessible through a publicly available URL generated for your CodeBuild project. This allows anyone to view the status of a CodeBuild project. Build badges do not contain any security information, so they do not require authentication.

Topics

Create a build project with build badges

Use the following one of the following procedures to create a build project with build badges enabled. You can use AWS CLI or the AWS Management Console.

To create a build project with build badges enabled (AWS CLI)

- For information about creating a build project, see Create a build project (AWS CLI). To include build badges with your AWS CodeBuild project, you must specify `badgeEnabled` with a value of `true`.

To create a build project with build badges enabled (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.
3. In **Project name**, enter a name for this build project. Build project names must be unique across each AWS account. You can also include an optional description of the build project to help other users understand what this project is used for.
4. In **Source**, for **Source provider**, choose the source code provider type, and then do one of the following:

> **Note**
>
> CodeBuild does not support build badges with the Amazon S3 source provider. Because AWS CodePipeline uses Amazon S3 for artifact transfers, build badges are not supported for build projects that are part of a pipeline created in CodePipeline.

- If you chose **CodeCommit**, then for **Repository**, choose the name of the repository. Select **Enable build badge** to make your project's build status visible and embeddable.
- If you chose **GitHub**, follow the instructions to connect (or reconnect) with GitHub. On the GitHub **Authorize application** page, for **Organization access**, choose **Request access** next to each repository you want AWS CodeBuild to be able to access. After you choose **Authorize application**, back in the AWS CodeBuild console, for **Repository**, choose the name of the repository that contains the source code. Select **Enable build badge** to make your project's build status visible and embeddable.
- If you chose **Bitbucket**, follow the instructions to connect (or reconnect) with Bitbucket. On the Bitbucket **Confirm access to your account** page, for **Organization access**, choose **Grant access**. After you choose **Grant access**, back in the AWS CodeBuild console, for **Repository**, choose the name of the repository that contains the source code. Select **Enable build badge** to make your project's build status visible and embeddable.

> **Important**
>
> Updating your project source might affect the accuracy of the project's build badges.

5. In **Environment**:

   For **Environment image**, do one of the following:

   - To use a Docker image managed by AWS CodeBuild, choose **Managed image**, and then make selections from **Operating system**, **Runtime(s)**, **Image**, and **Image version**. Make a selection from **Environment type** if it is available.
   - To use another Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. If you choose **Other registry**, for **External registry URL**, enter the name and tag of the Docker image in Docker Hub, using the format *`docker`*

repository/*docker image name*. If you choose **Amazon ECR**, use **Amazon ECR repository** and **Amazon ECR image** to choose the Docker image in your AWS account.

- To use a private Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. For **Image registry**, choose **Other registry**, and then enter the ARN of the credentials for your private Docker image. The credentials must be created by Secrets Manager. For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.

6. In **Service role**, do one of the following:

   - If you do not have a CodeBuild service role, choose **New service role**. In **Role name**, enter a name for the new role.
   - If you have a CodeBuild service role, choose **Existing service role**. In **Role ARN**, choose the service role.

   > **Note**
   >
   > When you use the console to create or update a build project, you can create a CodeBuild service role at the same time. By default, the role works with that build project only. If you use the console to associate this service role with another build project, the role is updated to work with the other build project. A service role can work with up to 10 build projects.

7. In **Buildspec**, do one of the following:

   - Choose **Use a buildspec file** to use the buildspec.yml file in the source code root directory.
   - Choose **Insert build commands** to use the console to insert build commands.

   For more information, see the Buildspec reference.

8. In **Artifacts**, for **Type**, do one of the following:

   - If you do not want to create build output artifacts, choose **No artifacts**.
   - To store the build output in an S3 bucket, choose **Amazon S3**, and then do the following:
     - If you want to use your project name for the build output ZIP file or folder, leave **Name** blank. Otherwise, enter the name. By default, the artifact name is the project name. If you want to use a different name, enter it in the artifacts name box. If you want to output a ZIP file, include the zip extension.

- For **Bucket name**, choose the name of the output bucket.
- If you chose **Insert build commands** earlier in this procedure, for **Output files**, enter the locations of the files from the build that you want to put into the build output ZIP file or folder. For multiple locations, separate each location with a comma (for example, `appspec.yml, target/my-app.jar`). For more information, see the description of `files` in Buildspec syntax.

9. Expand **Additional configuration** and choose options as appropriate.
10. Choose **Create build project**. On the **Review** page, choose **Start build** to run the build.

Access AWS CodeBuild build badges

You can use AWS CodeBuild console or the AWS CLI to access build badges.

- In the CodeBuild console, in the list of build projects, in the **Name** column, choose the link that corresponds to the build project. On the **Build project: *`project-name`*** page, in **Configuration**, choose **Copy badge URL**. For more information, see View a build project's details (console).
- In the AWS CLI, run the `batch-get-projects` command. The build badge URL is included in the project environment details section of the output. For more information, see View a build project's details (AWS CLI).

The build badge request URL is generated with a common default branch, but you can specify any branch in your source repository that you have used to run a build. For example:

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

You can also specify a tag from your source repository by substituting the `branch` parameter with the `tag` parameter in the badge URL. For example:

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

Publish CodeBuild build badges

You can display the status of the latest build in a markdown file using your build badge URL in a markdown image. This is useful to display the status of the most recent build in the readme.md file in your source repository (for example, GitHub or CodeCommit). For example:

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

CodeBuild badge statuses

The CodeBuild build badge can have one of the following statuses.

- **PASSING** The most recent build on the given branch passed.
- **FAILING** The most recent build on the given branch timed out, failed, faulted, or was stopped.
- **IN_PROGRESS** The most recent build on the given branch is in progress.
- **UNKNOWN** The project has not yet run a build for the given branch or at all. Also, the build badges feature might have been disabled.

'Test report using the AWS CLI' sample

Tests that you specify in your buildspec file are run during your build. This sample shows you how to use the AWS CLI to incorporate tests into builds in CodeBuild. You can use JUnit to create unit tests, or you can use another tool to create configuration tests. You can then evaluate the test results to fix issues or optimize your application.

You can use the CodeBuild API or the AWS CodeBuild console to access the test results. This sample shows you how to configure your report so its test results are exported to an S3 bucket.

Topics

Run the test report sample

Use the following steps to run the trest report sample.

Topics

Prerequisites

- Create your test cases. This sample is written with the assumption that you have test cases to include in your sample test report. You specify the location of your test files in the buildspec file.

  The following test report file formats are supported:

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

  Create your test cases with any test framework that can create report files in one of these formats (for example, Surefire JUnit plugin, TestNG, or Cucumber).
- Create an S3 bucket and make a note of its name. For more information, see How do I create an S3 bucket? in the *Amazon S3 User Guide*.
- Create an IAM role and make a note of its ARN. You need the ARN when you create your build project.
- If your role does not have the following permissions, add them.

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

For more information, see Permissions for test reporting operations.

Step 1: Create a report group

1. Create a file named `CreateReportGroupInput.json`.
2. Create a folder in your S3 bucket where your test results are exported.
3. Copy the following into `CreateReportGroupInput.json`. For *<bucket-name>*, use the name of the S3 bucket. For *<path-to-folder>*, enter the path to the folder in your S3 bucket.

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

4. Run the following command in the directory that contains `CreateReportGroupInput.json`.

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

The output looks like the following. Make a note of the ARN for the `reportGroup`. You use it when you create a project that uses this report group.

```
{
  "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
  }
}
```

Step 2: Configure a project with a report group

To run a report, you first create a CodeBuild build project that is configured with your report group. Test cases specified for your report group are run when you run a build.

1. Create a buildspec file named `buildspec.yml`.
2. Use the following YAML as a template for your `buildspec.yml` file. Be sure to include the commands that run your tests. In the `reports` section, specify the files that contain the results of your test cases. These files store the test results you can access with CodeBuild. They expire 30 days after they are created. These files are different from the raw test case result files you export to an S3 bucket.

```
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**
>
> Instead of the ARN of an existing report group, you can also specify a name for a report group that has not been created. If you specify a name instead of an ARN,

CodeBuild creates a report group when it runs a build. Its name contains your project name and the name you specify in the buildspec file, in this format: `project-name-report-group-name`. For more information, see Create test reports and Report group naming.

3. Create a file named `project.json`. This file contains input for the **create-project** command.
4. Copy the following JSON into `project.json`. For `source`, enter the type and location of the repository that contains your source files. For `serviceRole`, specify the ARN of the role you are using.

```
{
  "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. Run the following command in the directory that contains `project.json`. This creates a project named `test-project`.

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

Step 3: Run and view results of a report

In this section, you run a build of the project you created earlier. During the build process, CodeBuild creates a report with the results of the test cases. The report is contained in the report group you specified.

1. To start a build, run the following command. `test-report-project` is the name of the build project created above. Make a note of the build ID that appears in the output.

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

2. Run the following command to get information about your build, including the ARN of your report. For *<build-id>*, specify your build ID. Make a note of the report ARN in the `reportArns` property of the output.

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

3. Run the following command to get details about your report. For *<report-arn>*, specify your report ARN.

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

The output looks like the following. This sample output shows how many of the tests were successful, failed, skipped, resulted in an error, or return an unknown status.

```
{
  "reports": [
    {
      "status": "FAILED",
      "reportGroupArn": "<report-group-arn>",
      "name": "<report-group-name>",
      "created": 1573324770.154,
      "exportConfig": {
        "exportConfigType": "S3",
        "s3Destination": {
          "bucket": "<amzn-s3-demo-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. Run the following command to list information about test cases for your report. For *<report-arn>*, specify the ARN of your report. For the optional `--filter` parameter, you can specify one status result (SUCCEEDED, FAILED, SKIPPED, ERROR, or UNKNOWN).

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

The output looks like the following.

```
{
  "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>"
    }
  ]
}
```

Docker samples for CodeBuild

This section describes sample integrations between Docker and AWS CodeBuild.

Sample	Description
Docker in custom image sample for CodeBuild	This sample builds and runs a Docker image by using CodeBuild and a custom Docker build image (`docker:dind` in Docker Hub).
Windows Docker builds sample for CodeBuild	This sample builds and runs a Windows Docker image by using CodeBuild.
'Publish Docker image to an Amazon ECR image repository' sample for CodeBuild	This sample produces as build output a Docker image and then pushes the Docker image to an Amazon Elastic Container Registry (Amazon ECR) image repository.
Private registry with AWS Secrets Manager sample for CodeBuild	This sample shows you how to use a Docker image that is stored in a private registry as your CodeBuild runtime environment.

Docker in custom image sample for CodeBuild

The following sample builds and runs a Docker image by using AWS CodeBuild and a custom Docker build image (`docker:dind` in Docker Hub).

To learn how to build a Docker image by using a build image provided by CodeBuild with Docker support instead, see our 'Publish Docker image to Amazon ECR' sample.

> **Important**
>
> Running this sample might result in charges to your AWS account. These include possible charges for CodeBuild and for AWS resources and actions related to Amazon S3, AWS KMS, and CloudWatch Logs. For more information, see CodeBuild pricing, Amazon S3 pricing, AWS Key Management Service pricing, and Amazon CloudWatch pricing.

Topics

Run the Docker in custom image sample

Use the following procedure to run the Docker in custom image sample. For more information about this sample, see Docker in custom image sample for CodeBuild.

To run the Docker in custom image sample

1. Create the files as described in the Directory structure and Files sections of this topic, and then upload them to an S3 input bucket or an AWS CodeCommit, GitHub, or Bitbucket repository.

   > **Important**
   >
   > Do not upload *(root directory name)*, just the files inside of *(root directory name)*.
   >
   > If you are using an S3 input bucket, be sure to create a ZIP file that contains the files, and then upload it to the input bucket. Do not add *(root directory name)* to the ZIP file, just the files inside of *(root directory name)*.

2. Create a build project, run the build, and view related build information.

If you use the AWS CLI to create the build project, the JSON-formatted input to the `create-project` command might look similar to this. (Replace the placeholders with your own values.)

```
{
  "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**
>
> By default, Docker daemon is enabled for non-VPC builds. If you would like to use Docker containers for VPC builds, see Runtime Privilege and Linux Capabilities on the Docker Docs website and enable privileged mode. Also, Windows does not support privileged mode.

3. To see the build results, look in the build's log for the string `Hello, World!`. For more information, see View build details.

Directory structure

This sample assumes this directory structure.

```
(root directory name)
```

```
### buildspec.yml
### Dockerfile
```

Files

The base image of the operating system used in this sample is Ubuntu. The sample uses these files.

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

```
FROM maven:3.3.9-jdk-8

RUN echo "Hello World"
```

Windows Docker builds sample for CodeBuild

The following sample builds and runs a Windows Docker image by using CodeBuild.

Topics

- Run Windows Docker builds sample

Run Windows Docker builds sample

Use the following procedure to run the Windows Docker builds.

To run Windows Docker builds sample

1. Create the files as described in the Directory structure and Files sections of this topic, and then upload them to an S3 input bucket or an AWS CodeCommit, GitHub, or Bitbucket repository.

> **Important**
>
> Do not upload *(root directory name)*, just the files inside of *(root directory name)*.
>
> If you are using an S3 input bucket, be sure to create a ZIP file that contains the files, and then upload it to the input bucket. Do not add *(root directory name)* to the ZIP file, just the files inside of *(root directory name)*.

2. Create a `WINDOWS_EC2` fleet.

   If you use the AWS CLI to create the fleet, the JSON-formatted input to the `create-fleet` command might look similar to this. (Replace the placeholders with your own values.)

```
{
  "name": "fleet-name",
  "baseCapacity": 1,
  "environmentType": "WINDOWS_EC2",
  "computeType": "BUILD_GENERAL1_MEDIUM"
}
```

3. Create a build project, run the build, and view related build information.

   If you use the AWS CLI to create the build project, the JSON-formatted input to the `create-project` command might look similar to this. (Replace the placeholders with your own values.)

```
{
  "name": "project-name",
  "source": {
    "type": "S3",
    "location": "bucket-name/DockerImageSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "WINDOWS_EC2",
    "image": "Windows",
    "computeType": "BUILD_GENERAL1_MEDIUM",
    "fleet": {
```

```
        "fleetArn": "fleet-arn"
      }
    },
    "serviceRole": "arn:aws:iam::account-ID:role/role-name"
}
```

4. To see the build results, look in the build's log for the string `Hello, World!`. For more information, see View build details.

Directory structure

This sample assumes this directory structure.

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

Files

The base image of the operating system used in this sample is `mcr.microsoft.com/windows/servercore:ltsc2022`. The sample uses these files.

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

```
version: 0.2

phases:
  pre_build:
    commands:
      - docker build -t helloworld .
  build:
    commands:
      - docker images
      - docker run helloworld powershell -Command "Write-Host 'Hello World!'"
```

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

```
FROM mcr.microsoft.com/windows/servercore:ltsc2022

RUN powershell -Command "Write-Host 'Hello World'"
```

'Publish Docker image to an Amazon ECR image repository' sample for CodeBuild

This sample produces as build output a Docker image and then pushes the Docker image to an Amazon Elastic Container Registry (Amazon ECR) image repository. You can adapt this sample to push the Docker image to Docker Hub. For more information, see Adapt the 'Publish Docker image to Amazon ECR' sample to push to Docker Hub.

To learn how to build a Docker image by using a custom Docker build image (`docker:dind` in Docker Hub), see our Docker in custom image sample.

This sample was tested referencing `golang:1.12`.

This sample uses the new multi-stage Docker builds feature, which produces a Docker image as build output. It then pushes the Docker image to an Amazon ECR image repository. Multi-stage Docker image builds help to reduce the size of the final Docker image. For more information, see Use multi-stage builds with Docker.

> **Important**
>
> Running this sample might result in charges to your AWS account. These include possible charges for AWS CodeBuild and for AWS resources and actions related to Amazon S3, AWS KMS, CloudWatch Logs, and Amazon ECR. For more information, see CodeBuild pricing, Amazon S3 pricing, AWS Key Management Service pricing, Amazon CloudWatch pricing, and Amazon Elastic Container Registry pricing.

Topics

Run the 'Publish Docker image to Amazon ECR' sample

Use the following procedure to run the sample that publishes a Docker image to Amazon ECR. For more infomration about this sample, see 'Publish Docker image to an Amazon ECR image repository' sample for CodeBuild.

To run this sample

1. If you already have an image repository in Amazon ECR you want to use, skip to step 3. Otherwise, if you are using an user instead of an AWS root account or an administrator user to work with Amazon ECR, add this statement (between *### BEGIN ADDING STATEMENT HERE ###* and *### END ADDING STATEMENT HERE ###*) to the user (or IAM group the user is associated with). Using an AWS root account is not recommended.This statement allows the creation of Amazon ECR repositories for storing Docker images. Ellipses (`...`) are used for brevity and to help you locate where to add the statement. Do not remove any statements, and do not type these ellipses into the policy. For more information, see Working with inline policies using the AWS Management Console in the *user Guide*.

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

> **Note**
>
> The IAM entity that modifies this policy must have permission in IAM to modify policies.

2. Create an image repository in Amazon ECR. Be sure to create the repository in the same AWS Region where you create your build environment and run your build. For more information, see Creating a repository in the *Amazon ECR User Guide*. This repository's name must match the repository name you specify later in this procedure, represented by the `IMAGE_REPO_NAME` environment variable. Ensure that the Amazon ECR repository policy grants image push access for your CodeBuild service IAM role.

3. Add this statement (between *### BEGIN ADDING STATEMENT HERE ###* and *### END ADDING STATEMENT HERE ###*) to the policy you attached to your AWS CodeBuild service role. This statement allows CodeBuild to upload Docker images to Amazon ECR repositories. Ellipses (`...`) are used for brevity and to help you locate where to add the statement. Do not remove any statements, and do not type these ellipses into the policy.

```
{
  "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**
>
> The IAM entity that modifies this policy must have permission in IAM to modify policies.

4. Create the files as described in the Directory structure and Files sections of this topic, and then upload them to an S3 input bucket or an AWS CodeCommit, GitHub, or Bitbucket repository. For more information, see Image definitions file reference in the *AWS CodePipeline User Guide*.

> **Important**
>
> Do not upload *(root directory name)*, just the files inside of *(root directory name)*.

If you are using an S3 input bucket, be sure to create a ZIP file that contains the files, and then upload it to the input bucket. Do not add *(root directory name)* to the ZIP file, just the files inside of *(root directory name)*.

5. Create a build project, run the build, and view build information.

   If you use the console to create your project:

   a. For **Operating system**, choose **Ubuntu**.

   b. For **Runtime**, choose **Standard**.

   c. For **Image**, choose **aws/codebuild/standard:5.0**.

   d. Add the following environment variables:

      - AWS_DEFAULT_REGION with a value of *region-ID*
      - AWS_ACCOUNT_ID with a value of *account-ID*
      - IMAGE_TAG with a value of Latest
      - IMAGE_REPO_NAME with a value of *Amazon-ECR-repo-name*

   If you use the AWS CLI to create the build project, the JSON-formatted input to the `create-project` command might look similar to this. (Replace the placeholders with your own values.)

```
{
  "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. Confirm that CodeBuild successfully pushed the Docker image to the repository:

   1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/.
   2. Choose the repository name. The image should be listed in the **Image tag** column.

Directory structure

This sample assumes this directory structure.

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

Files

This sample uses these files.

`buildspec.yml` (in *`(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` (in *`(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 overrides the `ENTRYPOINT` for custom Docker images.

Adapt the 'Publish Docker image to Amazon ECR' sample to push to Docker Hub

To adapt the 'Publish Docker image to Amazon ECR' sample so that the Docker image is pushed to Docker Hub instead of Amazon ECR, edit the sample's code. For more information about the

sample, see 'Publish Docker image to an Amazon ECR image repository' sample for CodeBuild and Run the 'Publish Docker image to Amazon ECR' sample.

> **Note**
>
> If you are using a version of Docker earlier than 17.06, remove the `--no-include-email` option.

1. Replace these Amazon ECR-specific lines of code in the `buildspec.yml` file:

```
...
  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
...
```

   With these Docker Hub-specific lines of code:

```
...
  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. Upload the edited code to an S3 input bucket or an AWS CodeCommit, GitHub, or Bitbucket repository.

> **Important**
>
> Do not upload *(root directory name)*, just the files inside of *(root directory name)*.
>
> If you are using an S3 input bucket, be sure to create a ZIP file that contains the files, and then upload it to the input bucket. Do not add *(root directory name)* to the ZIP file, just the files inside of *(root directory name)*.

3. Replace these lines of code from the JSON-formatted input to the `create-project` command:

```
...
    "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"
      }
    ]
...
```

With these lines of code:

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

4. Create a build environment, run the build, and view related build information.
5. Confirm that AWS CodeBuild successfully pushed the Docker image to the repository. Sign in to Docker Hub, go to the repository, and choose the **Tags** tab. The `latest` tag should contain a very recent **Last Updated** value.

Private registry with AWS Secrets Manager sample for CodeBuild

This sample shows you how to use a Docker image that is stored in a private registry as your AWS CodeBuild runtime environment. The credentials for the private registry are stored in AWS Secrets Manager. Any private registry works with CodeBuild. This sample uses Docker Hub.

> **Note**
>
> Secrets are visible to actions and are not masked when written to a file.

Topics

- Private registry sample requirements
- Create a CodeBuild project with a private registry

- Configure a private registry credential for self-hosted runners

Private registry sample requirements

To use a private registry with AWS CodeBuild, you must have the following:

- A Secrets Manager secret that stores your Docker Hub credentials. The credentials are used to access your private repository.

> **Note**
>
> You will be charged for secrets that you create.

- A private repository or account.
- A CodeBuild service role IAM policy that grants access to your Secrets Manager secret.

Follow these steps to create these resources and then create a CodeBuild build project using the Docker images stored in your private registry.

Create a CodeBuild project with a private registry

1. For information about how to create a free private repository, see Repositories on Docker Hub. You can also run the following commands in a terminal to pull an image, get its ID, and push it to a new repository.

```
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. Follow the steps in Create an AWS Secrets Manager secret in the *AWS Secrets Manager User Guide*.

   a. In step 3, in **Choose secret type**, choose **Other type of secret**.

   b. In **Key/value pairs**, create one key-value pair for your Docker Hub user name and one key-value pair for your Docker Hub password.

   c. Continue following the steps in Create an AWS Secrets Manager secret.

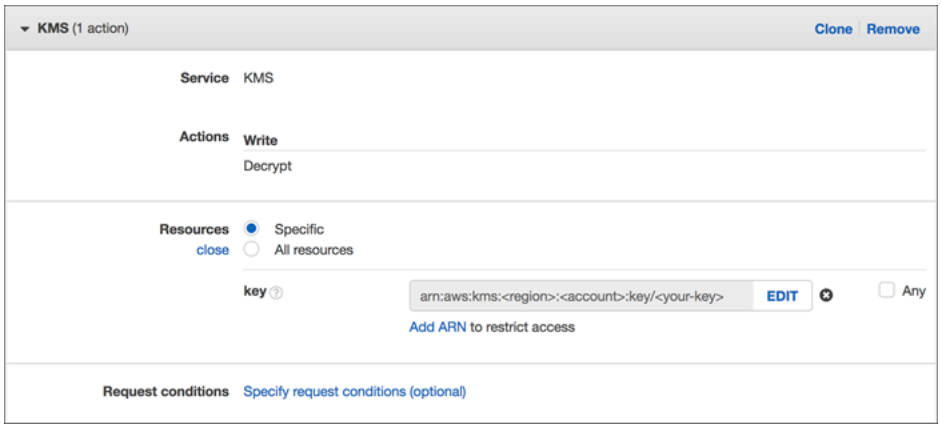
d. In step 5, on the **Configure automatic rotation** page, turn it off because the keys correspond to your Docker Hub credentials.

e. Finish following the steps in Create an AWS Secrets Manager secret.

For more information, see What is AWS Secrets Manager?

3. When you create an AWS CodeBuild project in the console, CodeBuild attaches the required permission for you. If you use an AWS KMS key other than `DefaultEncryptionKey`, you must add it to the service role. For more information, see Modifying a role (console) in the *IAM User Guide*.

   For your service role to work with Secrets Manager, it must have, at a minimum, the `secretsmanager:GetSecretValue` permission.



4. To use the console to create a project with an environment stored in a private registry, do the following while you create a project. For information, see Create a build project (console).

> **Note**
>
> If your private registry is in your VPC, it must have public internet access. CodeBuild cannot pull an image from a private IP address in a VPC.

a. In **Environment image**, choose **Custom image**.

b. For **Environment type**, choose **Linux** or **Windows**.

c. For **Image registry**, choose **Other registry**.

d. In **External registry URL**, enter the image location and in **Registry credential - optional** enter the ARN or name of your Secrets Manager credentials.

> **Note**
>
> If your credentials do not exist in your current Region, then you must use the ARN. You cannot use the credential name if the credentials exist in a different Region.

Configure a private registry credential for self-hosted runners

Use the following instructions to configure a registry credential for a self-hosted runner.

> **Note**
>
> Note that these credentials will only be used if the images are overridden with those from private registries.

AWS Management Console

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a build project or select an existing project. For information, see Create a build project (console) and Change a build project's settings (console).
3. In **Environment**, choose **Additional configuration**.
4. In **Additional configuration**, enter the name or ARN of the secret from AWS Secrets Manager for **Registry credential - optional**.

AWS CLI

1. If you'd like to create a new project, run the **create-project** command.

```
aws codebuild create-project \
    --name project-name \
    --source type=source-type,location=source-location \
```

```
    --environment "type=environment-type,image=image,computeType=compute-
type,registryCredential={credentialProvider=SECRETS_MANAGER,credential=secret-
name-or-arn},imagePullCredentialsType=CODEBUILD|SERVICE_ROLE" \
    --artifacts type=artifacts-type \
    --service-role arn:aws:iam::account-ID:role/service-role/service-role-name
```

2. If you'd like to update an existing project, run the **update-project** command.

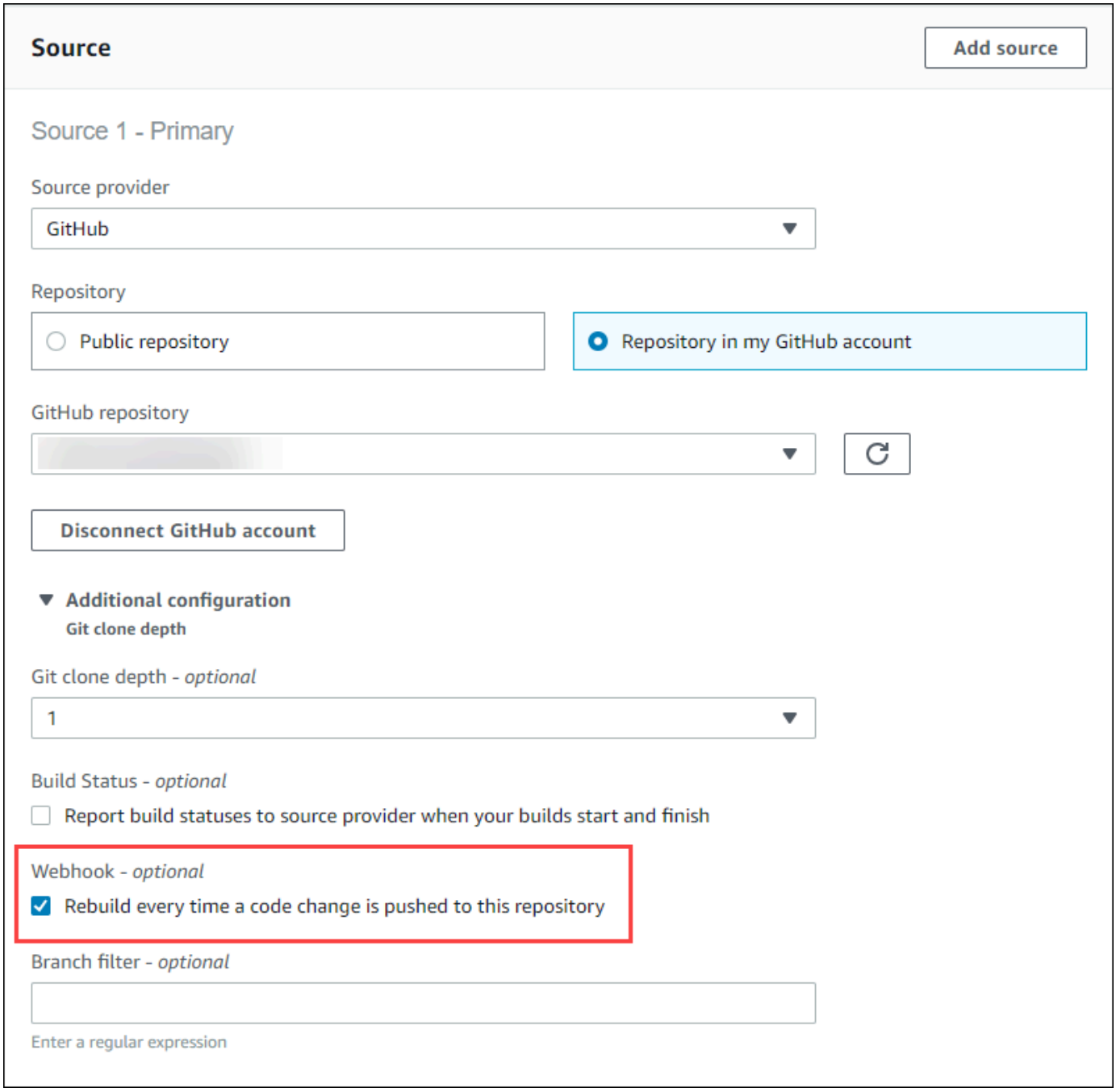
```
aws codebuild update-project \
    --name project-name \
    --environment "type=environment-type,image=image,computeType=compute-
type,registryCredential={credentialProvider=SECRETS_MANAGER,credential=secret-
name-or-arn}"
```

Create a static website with build output hosted in an S3 bucket

You can disable the encryption of artifacts in a build. You might want to do this so that you can publish artifacts to a location that is configured to host a website. (You cannot publish encrypted artifacts.) This sample shows how you can use webhooks to trigger a build and publish its artifacts to an S3 bucket that is configured to be a website.

1. Follow the instructions in Setting up a static website to configure an S3 bucket to function like a website.
2. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
3. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.
4. In **Project name**, enter a name for this build project. Build project names must be unique across each AWS account. You can also include an optional description of the build project to help other users understand what this project is used for.
5. In **Source**, for **Source provider**, choose **GitHub**. Follow the instructions to connect (or reconnect) with GitHub, and then choose **Authorize**.

   For **Webhook**, select **Rebuild every time a code change is pushed to this repository**. You can select this check box only if you chose **Use a repository in my account**.



6. In **Environment**:

   For **Environment image**, do one of the following:

   - To use a Docker image managed by AWS CodeBuild, choose **Managed image**, and then make selections from **Operating system**, **Runtime(s)**, **Image**, and **Image version**. Make a selection from **Environment type** if it is available.

- To use another Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. If you choose **Other registry**, for **External registry URL**, enter the name and tag of the Docker image in Docker Hub, using the format *`docker repository/docker image name`*. If you choose **Amazon ECR**, use **Amazon ECR repository** and **Amazon ECR image** to choose the Docker image in your AWS account.
- To use a private Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. For **Image registry**, choose **Other registry**, and then enter the ARN of the credentials for your private Docker image. The credentials must be created by Secrets Manager. For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.

7. In **Service role**, do one of the following:

   - If you do not have a CodeBuild service role, choose **New service role**. In **Role name**, enter a name for the new role.
   - If you have a CodeBuild service role, choose **Existing service role**. In **Role ARN**, choose the service role.

   > **Note**
   >
   > When you use the console to create or update a build project, you can create a CodeBuild service role at the same time. By default, the role works with that build project only. If you use the console to associate this service role with another build project, the role is updated to work with the other build project. A service role can work with up to 10 build projects.

8. In **Buildspec**, do one of the following:

   - Choose **Use a buildspec file** to use the buildspec.yml file in the source code root directory.
   - Choose **Insert build commands** to use the console to insert build commands.

   For more information, see the Buildspec reference.

9. In **Artifacts**, for **Type**, choose **Amazon S3** to store the build output in an S3 bucket.
10. For **Bucket name**, choose the name of the S3 bucket you configured to function as a website in step 1.

11. If you chose **Insert build commands** in **Environment**, then for **Output files**, enter the locations of the files from the build that you want to put into the output bucket. If you have more than one location, use a comma to separate each location (for example, **`appspec.yml, target/my-app.jar`**). For more information, see Artifacts reference-key in the buildspec file.
12. Select **Disable artifacts encryption**.
13. Expand **Additional configuration** and choose options as appropriate.
14. Choose **Create build project**. On the build project page, in **Build history**, choose **Start build** to run the build.
15. (Optional) Follow the instructions in Example: Speed up your website with Amazon CloudFront in the *Amazon S3 Developer Guide*.

Multiple input sources and output artifacts sample

You can create an AWS CodeBuild build project with more than one input source and more than one set of output artifacts. This sample shows you how to set up a build project that:

- Uses multiple sources and repositories of varying types.
- Publishes build artifacts to multiple S3 buckets in a single build.

In the following sample, you create a build project and use it to run a build. The sample uses the build project's buildspec file to show you how to incorporate more than one source and create more than one set of artifacts.

To learn how to to create a pipeline that uses multiple source inputs to CodeBuild to create multiple output artifacts, see Sample of a CodePipeline/CodeBuild integration with multiple input sources and output artifacts.

Topics

- Create a build project with multiple inputs and outputs
- Create a build project without a source

Create a build project with multiple inputs and outputs

Use the following procedure to create a build project with multiple inputs and outputs.

To create a build project with multiple inputs and outputs

1. Upload your sources to one or more S3 buckets, CodeCommit, GitHub, GitHub Enterprise Server, or Bitbucket repositories.
2. Choose which source is the primary source. This is the source in which CodeBuild looks for and runs your buildspec file.
3. Create a build project. For more information, see Create a build project in AWS CodeBuild.
4. Create your build project, run the build, and get information about the build.
5. If you use the AWS CLI to create the build project, the JSON-formatted input to the `create-project` command might look similar to the following:

```
{
  "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"
}
```

Your primary source is defined under the `source` attribute. All other sources are called secondary sources and appear under `secondarySources`. All secondary sources are installed in their own directory. This directory is stored in the built-in environment variable `CODEBUILD_SRC_DIR_`*`sourceIdentifer`*. For more information, see Environment variables in build environments.

The `secondaryArtifacts` attribute contains a list of artifact definitions. These artifacts use the `secondary-artifacts` block of the buildspec file that is nested inside the `artifacts` block.

Secondary artifacts in the buildspec file have the same structure as artifacts and are separated by their artifact identifier.

> **Note**
>
> In the CodeBuild API, the `artifactIdentifier` on a secondary artifact is a required attribute in `CreateProject` and `UpdateProject`. It must be used to reference a secondary artifact.

Using the preceding JSON-formatted input, the buildspec file for the project might look like:

```
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
```

You can override the version of the primary source using the API with the `sourceVersion` attribute in `StartBuild`. To override one or more secondary source versions, use the `secondarySourceVersionOverride` attribute.

The JSON-formatted input to the the `start-build` command in the AWS CLI might look like:

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

Create a build project without a source

You can configure a CodeBuild project by choosing the **NO_SOURCE** source type when you configure your source. When your source type is **NO_SOURCE**, you cannot specify a buildspec file because your project does not have a source. Instead, you must specify a YAML-formatted buildspec string in

the `buildspec` attribute of the JSON-formatted input to the `create-project` CLI command. It might look like this:

```
{
  "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"
}
```

For more information, see Create a build project (AWS CLI).

Runtime versions in buildspec file sample for CodeBuild

If you use the Amazon Linux 2 (AL2) standard image version 1.0 or later, or the Ubuntu standard image version 2.0 or later, you can specify one or more runtimes in the `runtime-versions` section of your buildspec file. The following samples show how you can change your project runtime, specify more than one runtime, and specify a runtime that is dependent on another runtime. For information about supported runtimes, see Docker images provided by CodeBuild.

> **Note**
>
> If you use Docker in your build container, your build must run in privileged mode. For more information, see Run AWS CodeBuild builds manually and Create a build project in AWS CodeBuild.

Topics

- Update the runtime version in the buildspec file
- Specify two runtimes

Update the runtime version in the buildspec file

You can modify the runtime used by your project to a new version by updating the `runtime-versions` section of your buildspec file. The following examples show how to specify java versions 8 and 11.

- A `runtime-versions` section that specifies version 8 of Java:

```
phases:
  install:
    runtime-versions:
      java: corretto8
```

- A `runtime-versions` section that specifies version 11 of Java:

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

The following examples show how to specify different versions of Python using the Ubuntu standard image 5.0 or the Amazon Linux 2 standard image 3.0:

- A `runtime-versions` section that specifies Python version 3.7:

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

- A `runtime-versions` section that specifies Python version 3.8:

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

This sample demonstrates a project that starts with the Java version 8 runtime, and then is updated to the Java version 10 runtime.

1. Download and install Maven. For information, see Downloading Apache Maven and Installing Apache Maven on the Apache Maven website.
2. Switch to an empty directory on your local computer or instance, and then run this Maven command.

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

If successful, this directory structure and files are created.

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

3. Create a file named `buildspec.yml` with the following contents. Store the file in the *(root directory name)*`/my-web-app` directory.

```
version: 0.2

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

In the buildspec file:

- The `runtime-versions` section specifies that the project uses version 8 of the Java runtime.
- The `- java -version` command displays the version of Java used by your project when it builds.

Your file structure should now look like this.

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

4. Upload the contents of the `my-web-app` directory to an S3 input bucket or a CodeCommit, GitHub, or Bitbucket repository.

> **Important**
>
> Do not upload *(root directory name)* or *(root directory name)*`/my-web-app`, just the directories and files in *(root directory name)*`/my-web-app`.
> If you are using an S3 input bucket, be sure to create a ZIP file that contains the directory structure and files, and then upload it to the input bucket. Do not add *(root directory name)* or *(root directory name)*`/my-web-app` to the ZIP file, just the directories and files in *(root directory name)*`/my-web-app`.

5. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
6. Create a build project. For more information, see Create a build project (console) and Run a build (console). Leave all settings at their default values, except for these settings.

- For **Environment**:
  - For **Environment image**, choose **Managed image**.

- For **Operating system**, choose **Amazon Linux 2**.
- For **Runtime(s)**, choose **Standard**.
- For **Image**, choose **aws/codebuild/amazonlinux-x86_64-standard:4.0**.

7. Choose **Start build**.
8. On **Build configuration**, accept the defaults, and then choose **Start build**.
9. After the build is complete, view the build output on the **Build logs** tab. You should see output similar to the following:

```
[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. Update the `runtime-versions` section with Java version 11:

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

11. After you save the change, run your build again and view the build output. You should see that the installed version of Java is 11. You should see output similar to the following:

```
[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/*;
```

Specify two runtimes

You can specify more than one runtime in the same CodeBuild build project. This sample project uses two source files: one that uses the Go runtime and one that uses the Node.js runtime.

1. Create a directory named `my-source`.
2. Inside the `my-source` directory, create a directory named `golang-app`.
3. Create a file named `hello.go` with the following contents. Store the file in the `golang-app` directory.

```
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. Inside the `my-source` directory, create a directory named `nodejs-app`. It should be at the same level as the `golang-app` directory.

5. Create a file named `index.js` with the following contents. Store the file in the `nodejs-app` directory.

```
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. Create a file named `package.json` with the following contents. Store the file in the `nodejs-app` directory.

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

7. Create a file named `buildspec.yml` with the following contents. Store the file in the `my-source` directory, at the same level as the `nodejs-app` and `golang-app` directories. The `runtime-versions` section specifies the Node.js version 12 and Go version 1.13 runtimes.

```
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. Your file structure should now look like this.

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

9. Upload the contents of the `my-source` directory to an S3 input bucket or a CodeCommit, GitHub, or Bitbucket repository.

> **Important**
>
> If you are using an S3 input bucket, be sure to create a ZIP file that contains the directory structure and files, and then upload it to the input bucket. Do not add `my-source` to the ZIP file, just the directories and files in `my-source`.

10. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
11. Create a build project. For more information, see Create a build project (console) and Run a build (console). Leave all settings at their default values, except for these settings.
    - For **Environment**:
      - For **Environment image**, choose **Managed image**.

- For **Operating system**, choose **Amazon Linux 2**.
- For **Runtime(s)**, choose **Standard**.
- For **Image**, choose **aws/codebuild/amazonlinux-x86_64-standard:4.0**.

12. Choose **Create build project**.
13. Choose **Start build**.
14. On **Build configuration**, accept the defaults, and then choose **Start build**.
15. After the build is complete, view the build output on the **Build logs** tab. You should see output similar to the following. It shows output from the Go and Node.js runtimes. It also shows output from the Go and Node.js applications.

```
[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
```

Source version sample with AWS CodeBuild

This sample demonstrates how to specify a version of your source using a format other than a commit ID (also known as a commit SHA). You can specify the version of your source in the following ways:

- For an Amazon S3 source provider, use the version ID of the object that represents the build input ZIP file.
- For CodeCommit, Bitbucket, GitHub, and GitHub Enterprise Server, use one of the following:
  - Pull request as a pull request reference (for example, `refs/pull/1/head`).
  - Branch as a branch name.
  - Commit ID.
  - Tag.
  - Reference and a commit ID. The reference can be one of the following:
    - A tag (for example, `refs/tags/mytagv1.0^{full-commit-SHA}`).
    - A branch (for example, `refs/heads/mydevbranch^{full-commit-SHA}`).
    - A pull request (for example, `refs/pull/1/head^{full-commit-SHA}`).
- For GitLab and GitLab Self Managed, use one of the following:
  - Branch as a branch name.

- Commit ID.
- Tag.

> **Note**
>
> You can specify the version of a pull request source only if your repository is GitHub or GitHub Enterprise Server.

If you use a reference and a commit ID to specify a version, the `DOWNLOAD_SOURCE` phase of your build is faster than if you provide the version only. This is because when you add a reference, CodeBuild does not need to download the entire repository to find the commit.

- You can specify a source version with only a commit ID, such as `12345678901234567890123467890123456789`. If you do this, CodeBuild must download the entire repository to find the version.
- You can specify a source version with a reference and a commit ID in this format: *`refs`*`/`*`heads`*`/`*`branchname`*`^{`*`full-commit-SHA`*`}` (for example, `refs/heads/main^{12345678901234567890123467890123456789}`). If you do this, CodeBuild downloads only the specified branch to find the version. .

> **Note**
>
> To speed up the `DOWNLOAD_SOURCE` phase of your build, you can also to set **Git clone depth** to a low number. CodeBuild downloads fewer versions of your repository.

Topics

- Specify a GitHub repository version with a commit ID
- Specify a GitHub repository version with a reference and commit ID

Specify a GitHub repository version with a commit ID

You can specify a source version with only a commit ID, such as 12345678901234567890123467890123456789. If you do this, CodeBuild must download the entire repository to find the version.

To specify a GitHub repository version with a commit ID

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a build project. For information, see Create a build project (console) and Run a build (console). Leave all settings at their default values, except for these settings:
   - In **Source**:
     - For **Source provider**, choose **GitHub**. If you are not connected to GitHub, follow the instructions to connect.
     - For **Repository**, choose **Public repository**.
     - For **Repository URL**, enter **`https://github.com/aws/aws-sdk-ruby.git`**.
   - In **Environment**:
     - For **Environment image**, choose **Managed image**.
     - For **Operating system**, choose **Amazon Linux 2**.
     - For **Runtime(s)**, choose **Standard**.
     - For **Image**, choose **aws/codebuild/amazonlinux-x86_64-standard:4.0**.
3. For **Build specifications**, choose **Insert build commands**, and then choose **Switch to editor**.
4. In **Build commands**, replace the placeholder text with the following:

```
version: 0.2

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

The `runtime-versions` section is required when you use the Ubuntu standard image 2.0. Here, the Ruby version 2.6 runtime is specified, but you can use any runtime. The `echo` command displays the version of the source code stored in the `CODEBUILD_RESOLVED_SOURCE_VERSION` environment variable.

5. On **Build configuration**, accept the defaults, and then choose **Start build**.
6. For **Source version**, enter **`046e8b67481d53bdc86c3f6affdd5d1afae6d369`**. This is the SHA of a commit in the `https://github.com/aws/aws-sdk-ruby.git` repository.
7. Choose **Start build**.
8. When the build is complete, you should see the following:

   - On the **Build logs** tab, which version of the project source was used. Here is an example.

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

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

   - On the **Environment variables** tab, the **Resolved source version** matches the commit ID used to create the build.
   - On the **Phase details** tab, the duration of the `DOWNLOAD_SOURCE` phase.

Specify a GitHub repository version with a reference and commit ID

You can specify a source version with a reference and a commit ID in this format: *`refs`*`/`*`heads`*`/`*`branchname`*`^{`*`full-commit-SHA`*`}` (for example, `refs/heads/main^{12345678901234567890123467890123456789}`). If you do this, CodeBuild downloads only the specified branch to find the version.

To specify a GitHub repository version with a reference and commit ID.

1. Complete the steps in Specify a GitHub repository version with a commit ID.
2. From the left navigation pane, choose **Build projects**, and then choose the project you created earlier.
3. Choose **Start build**.

4. In **Source version**, enter **`refs/heads/main^{046e8b67481d53bdc86c3f6affdd5d1afae6d369}`**. This is the same commit ID and a reference to a branch in the format *`refs`*/*`heads`*/*`branchname`*^{*`full-commit-SHA`*}.
5. Choose **Start build**.
6. When the build is complete, you should see the following:

   - On the **Build logs** tab, which version of the project source was used. Here is an example.

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

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

   - On the **Environment variables** tab, the **Resolved source version** matches the commit ID used to create the build.
   - On the **Phase details** tab, the duration of the `DOWNLOAD_SOURCE` phase should be shorter than the duration when you used only the commit ID to specify the version of your source.

Third-party source repository samples for CodeBuild

This section describes sample integrations between third-party source repositories and CodeBuild.

Sample	Description
BitBucket pull request and webhook filter sample – see Run the 'Bitbucket pull request and webhook filter' sample for CodeBuild	This sample shows you how to create a pull request using a Bitbucket repository. It also shows you how to use a Bitbucket webhook to trigger CodeBuild to create a build of a project.
GitHub Enterprise Server sample – see Run the GitHub Enterprise Server sample for CodeBuild	This sample shows you how to set up your CodeBuild projects when your GitHub Enterprise Server repository has a certifica te installed. It also shows how to enable webhooks so that CodeBuild rebuilds the source code every time a code change is

Sample	Description
	pushed to your GitHub Enterprise Server repository.
GitHub pull request and webhook filter sample – see Run the GitHub pull request and webhook filter sample for CodeBuild	This sample shows you how to create a pull request using a GitHub Enterprise Server repository. It also shows how to enable webhooks so that CodeBuild rebuilds the source code every time a code change is pushed to your GitHub Enterprise Server repository.

Run the 'Bitbucket pull request and webhook filter' sample for CodeBuild

AWS CodeBuild supports webhooks when the source repository is Bitbucket. This means that for a CodeBuild build project that has its source code stored in a Bitbucket repository, webhooks can be used to rebuild the source code every time a code change is pushed to the repository. For more information, see Bitbucket webhook events.

This sample shows you how to create a pull request using a Bitbucket repository. It also shows you how to use a Bitbucket webhook to trigger CodeBuild to create a build of a project.

> **Note**
>
> When using webhooks, it is possible for a user to trigger an unexpected build. To mitigate this risk, see Best practices for using webhooks.

Topics

Prerequisites

To run this sample you must connect your AWS CodeBuild project with your Bitbucket account.

> **Note**
>
> CodeBuild has updated its permissions with Bitbucket. If you previously connected your project to Bitbucket and now receive a Bitbucket connection error, you must reconnect to grant CodeBuild permission to manage your webhooks.

Step 1: Create a build project with Bitbucket and enable webhooks

The following steps describe how to create an AWS CodeBuild project with Bitbucket as a source repository and enable webhooks.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.
3. Choose **Create build project**.
4. In **Project configuration**:

   Project name

   Enter a name for this build project. Build project names must be unique across each AWS account. You can also include an optional description of the build project to help other users understand what this project is used for.

5. In **Source**:

   Source provider

   Choose **Bitbucket**. Follow the instructions to connect (or reconnect) with Bitbucket and then choose **Authorize**.

   Repository

   Choose **Repository in my Bitbucket account**.

If you have not previously connected to your Bitbucket account, enter your Bitbucket username and app password, and select **Save Bitbucket credentials**.

Bitbucket repository

Enter the URL for your Bitbucket repository.

6. In **Primary source webhook events**, select the following.

> **Note**
>
> The **Primary source webhook events** section is only visible if you chose **Repository in my Bitbucket account** in the previous step.

1. Select **Rebuild every time a code change is pushed to this repository** when you create your project.
2. From **Event type**, choose one or more events.
3. To filter when an event triggers a build, under **Start a build under these conditions**, add one or more optional filters.
4. To filter when an event is not triggered, under **Don't start a build under these conditions**, add one or more optional filters.
5. Choose **Add filter group** to add another filter group, if needed.

For more information about Bitbucket webhook event types and filters, see Bitbucket webhook events.

7. In **Environment**:

Environment image

Choose one of the following:

To use a Docker image managed by AWS CodeBuild:

Choose **Managed image**, and then make selections from **Operating system**, **Runtime(s)**, **Image**, and **Image version**. Make a selection from **Environment type** if it is available.

To use another Docker image:

Choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. If you choose **Other registry**, for **External registry URL**, enter the name and tag of the Docker image in Docker Hub, using the format *`docker repository`*/*`docker image name`*. If you choose **Amazon ECR**, use **Amazon ECR repository** and **Amazon ECR image** to choose the Docker image in your AWS account.

To use a private Docker image:

Choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. For **Image registry**, choose **Other registry**, and then enter the ARN of the credentials for your private Docker image. The credentials must be created by Secrets Manager. For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.

Service role

Choose one of the following:

- If you do not have a CodeBuild service role, choose **New service role**. In **Role name**, enter a name for the new role.
- If you have a CodeBuild service role, choose **Existing service role**. In **Role ARN**, choose the service role.

> **Note**
>
> When you use the console to create or update a build project, you can create a CodeBuild service role at the same time. By default, the role works with that build project only. If you use the console to associate this service role with another build project, the role is updated to work with the other build project. A service role can work with up to 10 build projects.

8. In **Buildspec**, do one of the following:

   - Choose **Use a buildspec file** to use the buildspec.yml file in the source code root directory.
   - Choose **Insert build commands** to use the console to insert build commands.

   For more information, see the Buildspec reference.

9. In **Artifacts**:

Type

Choose one of the following:

- If you do not want to create build output artifacts, choose **No artifacts**.
- To store the build output in an S3 bucket, choose **Amazon S3**, and then do the following:
  - If you want to use your project name for the build output ZIP file or folder, leave **Name** blank. Otherwise, enter the name. By default, the artifact name is the project name. If you want to use a different name, enter it in the artifacts name box. If you want to output a ZIP file, include the zip extension.
  - For **Bucket name**, choose the name of the output bucket.
  - If you chose **Insert build commands** earlier in this procedure, for **Output files**, enter the locations of the files from the build that you want to put into the build output ZIP file or folder. For multiple locations, separate each location with a comma (for example, `appspec.yml, target/my-app.jar`). For more information, see the description of `files` in Buildspec syntax.

Additional configuration

Expand **Additional configuration** and set options as appropriate.

10. Choose **Create build project**. On the **Review** page, choose **Start build** to run the build.

Step 2: Trigger a build with a Bitbucket webhook

For a project that uses Bitbucket webhooks, AWS CodeBuild creates a build when the Bitbucket repository detects a change in your source code.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. On the navigation pane, choose **Build projects**, and then choose a project associated with a Bitbucket repository with webhooks. For information about creating a Bitbucket webhook project, see the section called "Step 1: Create a build project with Bitbucket and enable webhooks".
3. Make some changes in the code in your project's Bitbucket repository.
4. Create a pull request on your Bitbucket repository. For more information, see Making a pull request.
5. On the Bitbucket webhooks page, choose **View request** to see a list of recent events.

6. Choose **View details** to see details about the response returned by CodeBuild. It might look something like this:

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

7. Navigate to the Bitbucket pull request page to see the status of the build.

Run the GitHub Enterprise Server sample for CodeBuild

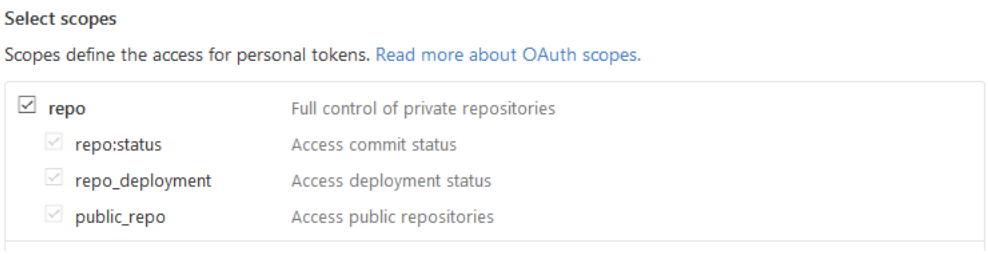
AWS CodeBuild supports GitHub Enterprise Server as a source repository. This sample shows how to set up your CodeBuild projects when your GitHub Enterprise Server repository has a certificate installed. It also shows how to enable webhooks so that CodeBuild rebuilds the source code every time a code change is pushed to your GitHub Enterprise Server repository.

Topics

Prerequisites

1. Generate a personal access token for your CodeBuild project. We recommend that you create a GitHub Enterprise user and generate a personal access token for this user. Copy it to your clipboard so that it can be used when you create your CodeBuild project. For more information, see Creating a personal access token for the command line on the GitHub Help website.

   When you create the personal access token, include the **repo** scope in the definition.



2. Download your certificate from GitHub Enterprise Server. CodeBuild uses the certificate to make a trusted SSL connection to the repository.

   Linux/macOS clients:

From a terminal window, run the following command:

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

Replace the placeholders in the command with the following values:

HOST. The IP address of your GitHub Enterprise Server repository.

PORTNUMBER. The port number you are using to connect (for example, 443).

folder. The folder where you downloaded your certificate.

filename. The file name of your certificate file.

> **Important**
>
> Save the certificate as a .pem file.

Windows clients:

Use your browser to download your certificate from GitHub Enterprise Server. To see the site's certificate details, choose the padlock icon. For information about how to export the certificate, see your browser documentation.

> **Important**
>
> Save the certificate as a .pem file.

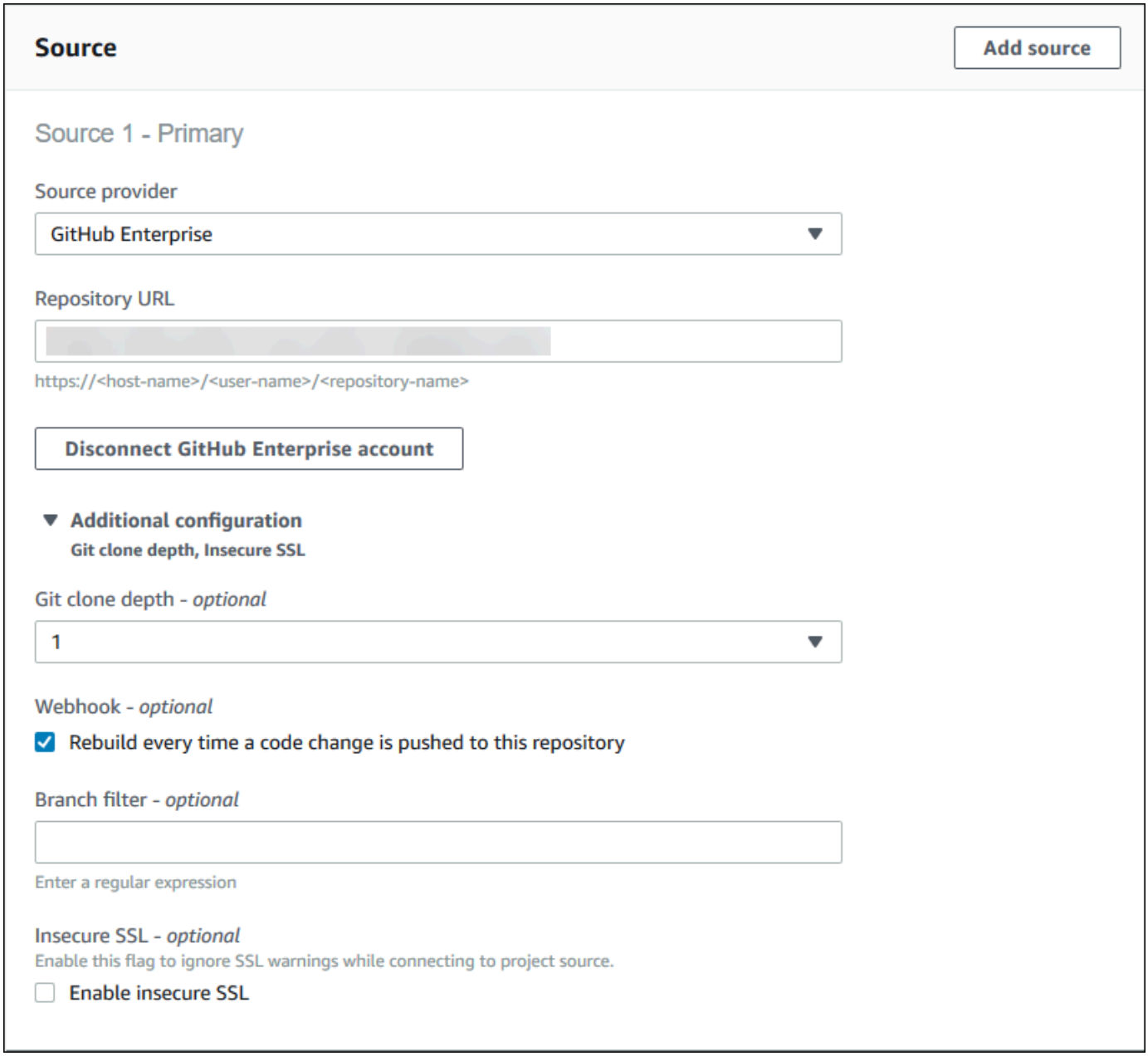
3. Upload your certificate file to an S3 bucket. For information about how to create an S3 bucket, see How do I create an S3 Bucket? For information about how to upload objects to an S3 bucket, see How do I upload files and folders to a bucket?

> **Note**
>
> This bucket must be in the same AWS region as your builds. For example, if you instruct CodeBuild to run a build in the US East (Ohio) Region, the bucket must be in the US East (Ohio) Region.

Step 1: Create a build project with GitHub Enterprise Server and enable webhooks

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.
3. In **Project name**, enter a name for this build project. Build project names must be unique across each AWS account. You can also include an optional description of the build project to help other users understand what this project is used for.
4. In **Source**, in **Source provider**, choose **GitHub Enterprise Server**.
   - Choose **Manage account credentials**, and then choose **Personal access token**. For **Service**, choose **Secrets Manager (recommended)**, and configure your secret. Then in **,GitHub Enterprise personal access token**, enter your personal access token and choose **Save**.
   - In **Repository URL**, enter the path to your repository, including the name of the repository.
   - Expand **Additional configuration**.
   - Select **Rebuild every time a code change is pushed to this repository** to rebuild every time a code change is pushed to this repository.
   - Select **Enable insecure SSL** to ignore SSL warnings while you connect to your GitHub Enterprise Server project repository.

     > **Note**
     >
     > We recommend that you use **Enable insecure SSL** for testing only. It should not be used in a production environment.



5. In **Environment**:

   For **Environment image**, do one of the following:

   - To use a Docker image managed by AWS CodeBuild, choose **Managed image**, and then make selections from **Operating system**, **Runtime(s)**, **Image**, and **Image version**. Make a selection from **Environment type** if it is available.
   - To use another Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. If you choose **Other registry**, for **External registry URL**,

enter the name and tag of the Docker image in Docker Hub, using the format *`docker repository`*/*`docker image name`*. If you choose **Amazon ECR**, use **Amazon ECR repository** and **Amazon ECR image** to choose the Docker image in your AWS account.

- To use a private Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. For **Image registry**, choose **Other registry**, and then enter the ARN of the credentials for your private Docker image. The credentials must be created by Secrets Manager. For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.

6. In **Service role**, do one of the following:

- If you do not have a CodeBuild service role, choose **New service role**. In **Role name**, enter a name for the new role.
- If you have a CodeBuild service role, choose **Existing service role**. In **Role ARN**, choose the service role.

> **Note**
>
> When you use the console to create or update a build project, you can create a CodeBuild service role at the same time. By default, the role works with that build project only. If you use the console to associate this service role with another build project, the role is updated to work with the other build project. A service role can work with up to 10 build projects.

7. Expand **Additional configuration**.

   If you want CodeBuild to work with your VPC:

- For **VPC**, choose the VPC ID that CodeBuild uses.
- For **VPC Subnets**, choose the subnets that include resources that CodeBuild uses.
- For **VPC Security groups**, choose the security groups that CodeBuild uses to allow access to resources in the VPCs.

   For more information, see Use AWS CodeBuild with Amazon Virtual Private Cloud.

8. In **Buildspec**, do one of the following:

- Choose **Use a buildspec file** to use the buildspec.yml file in the source code root directory.

- Choose **Insert build commands** to use the console to insert build commands.

For more information, see the Buildspec reference.

9. In **Artifacts**, for **Type**, do one of the following:

   - If you do not want to create build output artifacts, choose **No artifacts**.
   - To store the build output in an S3 bucket, choose **Amazon S3**, and then do the following:
     - If you want to use your project name for the build output ZIP file or folder, leave **Name** blank. Otherwise, enter the name. By default, the artifact name is the project name. If you want to use a different name, enter it in the artifacts name box. If you want to output a ZIP file, include the zip extension.
     - For **Bucket name**, choose the name of the output bucket.
     - If you chose **Insert build commands** earlier in this procedure, for **Output files**, enter the locations of the files from the build that you want to put into the build output ZIP file or folder. For multiple locations, separate each location with a comma (for example, `appspec.yml, target/my-app.jar`). For more information, see the description of `files` in Buildspec syntax.

10. For **Cache type**, choose one of the following:

    - If you do not want to use a cache, choose **No cache**.
    - If you want to use an Amazon S3 cache, choose **Amazon S3**, and then do the following:
      - For **Bucket**, choose the name of the S3 bucket where the cache is stored.
      - (Optional) For **Cache path prefix**, enter an Amazon S3 path prefix. The **Cache path prefix** value is similar to a directory name. It makes it possible for you to store the cache under the same directory in a bucket.

        > **Important**
        >
        > Do not append a trailing slash (/) to the end of the path prefix.

    - If you want to use a local cache, choose **Local**, and then choose one or more local cache modes.

> **Note**
>
> Docker layer cache mode is available for Linux only. If you choose it, your project must run in privileged mode.

Using a cache saves considerable build time because reusable pieces of the build environment are stored in the cache and used across builds. For information about specifying a cache in the buildspec file, see Buildspec syntax. For more information about caching, see Cache builds to improve performance.

11. Choose **Create build project**. On the build project page, choose **Start build**.

Run the GitHub pull request and webhook filter sample for CodeBuild

AWS CodeBuild supports webhooks when the source repository is GitHub. This means that for a CodeBuild build project that has its source code stored in a GitHub repository, webhooks can be used to rebuild the source code every time a code change is pushed to the repository. For CodeBuild samples, see AWS CodeBuild Samples.

> **Note**
>
> When using webhooks, it is possible for a user to trigger an unexpected build. To mitigate this risk, see Best practices for using webhooks.

Topics

Step 1: Create a build project with GitHub and enable webhooks

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.

3. Choose **Create build project**.
4. In **Project configuration**:

   Project name

   Enter a name for this build project. Build project names must be unique across each AWS account. You can also include an optional description of the build project to help other users understand what this project is used for.

5. In **Source**:

   Source provider

   Choose **GitHub**. Follow the instructions to connect (or reconnect) with GitHub and then choose **Authorize**.

   Repository

   Choose **Repository in my GitHub account**.

   GitHub repository

   Enter the URL for your GitHub repository.

6. In **Primary source webhook events**, select the following.

   > **Note**
   >
   > The **Primary source webhook events** section is only visible if you chose **Repository in my GitHub account** in the previous step.

   1. Select **Rebuild every time a code change is pushed to this repository** when you create your project.
   2. From **Event type**, choose one or more events.
   3. To filter when an event triggers a build, under **Start a build under these conditions**, add one or more optional filters.
   4. To filter when an event is not triggered, under **Don't start a build under these conditions**, add one or more optional filters.
   5. Choose **Add filter group** to add another filter group, if needed.

For more information about GitHub webhook event types and filters, see GitHub webhook events.

7. In **Environment**:

Environment image

Choose one of the following:

To use a Docker image managed by AWS CodeBuild:

Choose **Managed image**, and then make selections from **Operating system**, **Runtime(s)**, **Image**, and **Image version**. Make a selection from **Environment type** if it is available.

To use another Docker image:

Choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. If you choose **Other registry**, for **External registry URL**, enter the name and tag of the Docker image in Docker Hub, using the format *`docker repository`*/*`docker image name`*. If you choose **Amazon ECR**, use **Amazon ECR repository** and **Amazon ECR image** to choose the Docker image in your AWS account.

To use a private Docker image:

Choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. For **Image registry**, choose **Other registry**, and then enter the ARN of the credentials for your private Docker image. The credentials must be created by Secrets Manager. For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.

Service role

Choose one of the following:

- If you do not have a CodeBuild service role, choose **New service role**. In **Role name**, enter a name for the new role.
- If you have a CodeBuild service role, choose **Existing service role**. In **Role ARN**, choose the service role.

> **Note**
>
> When you use the console to create or update a build project, you can create a CodeBuild service role at the same time. By default, the role works with that build project only. If you use the console to associate this service role with another build project, the role is updated to work with the other build project. A service role can work with up to 10 build projects.

8. In **Buildspec**, do one of the following:

   - Choose **Use a buildspec file** to use the buildspec.yml file in the source code root directory.
   - Choose **Insert build commands** to use the console to insert build commands.

   For more information, see the Buildspec reference.

9. In **Artifacts**:

   Type

   Choose one of the following:

   - If you do not want to create build output artifacts, choose **No artifacts**.
   - To store the build output in an S3 bucket, choose **Amazon S3**, and then do the following:
     - If you want to use your project name for the build output ZIP file or folder, leave **Name** blank. Otherwise, enter the name. By default, the artifact name is the project name. If you want to use a different name, enter it in the artifacts name box. If you want to output a ZIP file, include the zip extension.
     - For **Bucket name**, choose the name of the output bucket.
     - If you chose **Insert build commands** earlier in this procedure, for **Output files**, enter the locations of the files from the build that you want to put into the build output ZIP file or folder. For multiple locations, separate each location with a comma (for example, `appspec.yml, target/my-app.jar`). For more information, see the description of `files` in Buildspec syntax.

   Additional configuration

   Expand **Additional configuration** and set options as appropriate.

10. Choose **Create build project**. On the **Review** page, choose **Start build** to run the build.

Step 2: Verify that webhooks are enabled

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. Do one of the following:
   - Choose the link for the build project with webhooks you want to verify, and then choose **Build details**.
   - Choose the button next to the build project with webhooks you want to verify, choose **View details**, and then choose the **Build details** tab.
4. In **Primary source webhook events**, choose the **Webhook** URL link.
5. In your GitHub repository, on the **Settings** page, under **Webhooks**, verify that **Pull Requests** and **Pushes** are selected.
6. In your GitHub profile settings, under **Personal settings**, **Applications**, **Authorized OAuth Apps**, you should see that your application has been authorized to access the AWS Region you selected.

Tutorial: Apple code signing with Fastlane in CodeBuild using S3 for certificate storage

fastlane is a popular open source automation tool to automate beta deployments and releases for your iOS and Android apps. It handles all tedious tasks, like generating screenshots, dealing with code signing, and releasing your application.

Prerequisites

To complete this tutorial, you must first have set up the following:

- An AWS account
- An Apple Developer account
- An S3 bucket for storing certificates
- fastlane installed in your project - Guide to install fastlane

Step 1: Set up Fastlane Match with S3 on your local machine

Fastlane Match is one of the Fastlane tools, and it allows for seamless configuration for code signing in both your local development environment and on CodeBuild. Fastlane Match stores all of your code signing certificates and provisioning profiles in a Git repository/S3 Bucket/Google Cloud Storage, and downloads and installs the necessary certificates and profiles when required.

In this example configuration, you will set up and use an Amazon S3 bucket for storage.

1. Initialize match in your project:

```
fastlane match init
```

2. When prompted, choose S3 as the storage mode.
3. Update your `*Matchfile*` to use S3:

```
storage_mode("s3")
   s3_bucket("your-s3-bucket-name")
   s3_region("your-aws-region")
   type("appstore") # The default type, can be: appstore, adhoc, enterprise or
 development
```

Step 2: Set up your Fastfile

Create or update your `Fastfile` with the following lane.

On CodeBuild, Fastlane Match will need to be run every time you build and sign your app. The easiest way to do this is to add the `match` action to the lane which builds your app.

```
default_platform(:ios)

platform :ios do
  before_all do
    setup_ci
  end

  desc "Build and sign the app"
  lane :build do
    match(type: "appstore", readonly: true)
```

```
    gym(
      scheme: "YourScheme",
      export_method: "app-store"
    )
  end
end
```

> **Note**
>
> Make sure to add `setup_ci` to the `before_all` section in `Fastfile` for the match action to work correctly. This ensures that a temporary Fastlane keychain with the appropriate permissions is used. Without using this you may see build failures or inconsistent results.

Step 3: Run the `fastlane match` command to generate respective certificates and profiles

The fastlane match command for the given type (i.e., development, appstore, adhoc, enterprise) will generate the certificate and profile if not available in remote store. The certificates and profiles will be stored in S3 by fastlane.

```
bundle exec fastlane match appstore
```

The command execution will be interactive and fastlane will ask to set pass phrase for decrypting the certificates.

Step 4: Create the application file for your project

Create or add the application file as appropriate for your project.

1. Create or add the Gymfile, Appfile, Snapfile, Deliverfile based on your project build requirements.
2. Commit the changes to your remote repository

Step 5: Create environment variables in Secrets Manager

Create two secrets for storing the fastlane session cookie and matching pass phrase. For more information about creating secrets in Secrets Manager, see Create an AWS Secrets Manager secret.

1. Access your fastlane session cookie as follows.

   a. Secret key - `FASTLANE_SESSION`

   b. Secret value - session cookie generated from running the following command on your local machine.

   > **Note**
   >
   > This value is available after authentication in a local file: `~/.fastlane/spaceship/my_appleid_username/cookie`.

   ```
   fastlane spaceauth -u <apple account>
   ```

2. Fastlane Match pass phrase - To enable Fastlane Match to decrypt the certificates and profiles stored in the S3 bucket, it is necessary to add the encryption passphrase that you configured in the Match setup step to the CodeBuild project's environment variables.

   a. Secret key - `MATCH_PASSWORD`

   b. Secret value - *`<match passphrase to decrypt certificates>`*. The passphrase is set while generating the certificates in Step 3.

> **Note**
>
> While creating the above secrets in Secrets Manager, remember to give a secret name with the following prefix: `/CodeBuild/`

Step 6: Create a compute fleet

Create the compute fleet for your project.

1. In the console, go to CodeBuild and create a new compute fleet.
2. Choose "macOS" as the operating system and select an appropriate compute type and image.

Step 7: Create a project in CodeBuild

Create your project in CodeBuild.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a build project. For information, see Create a build project (console) and Run a build (console).
3. Set up your source provider (such as GitHub, CodeCommit). This is iOS project source repository and not certificates repository.
4. In **Environment**:
   - Choose **Reserved Capacity**.
   - For **Fleet**, select the fleet created above.
   - Provide the name of the service role that CodeBuild will create for you.
   - Provide the below environment variables.
     - Name: `MATCH_PASSWORD`, Value: *`<secrets arn>`*, Type: Secrets Manager (Secrets ARN created in step 5 for MATCH_PASSWORD)
     - Name: `FASTLANE_SESSION`, Value: *`<secrets arn>`*, Type: Secrets Manager (Secrets ARN created in Step 5 for FASTLANE_SESSION)
5. In **Buildspec**, add the following:

```
version: 0.2

phases:
  install:
    commands:
      - gem install bundler
      - bundle install
  build:
    commands:
```

```
      - echo "Building and signing the app..."
      - bundle exec fastlane build
  post_build:
    commands:
      - echo "Build completed on date"

artifacts:
  files:
    - '*/.ipa'
  name: app-$(date +%Y-%m-%d)
```

Step 8: Configure IAM role

Once the project is created, ensure your CodeBuild project's service role has permissions to access the S3 bucket containing the certificates. Add the following policy to the role:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::your-s3-bucket-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::your-s3-bucket-name/*"
        }
    ]
}
```

Step 9: Run the build

Run the build. You can review the build status and logs in CodeBuild.

Once the job is completed, you will be able to view the log of the job.

Troubleshooting

- If you encounter issues with certificate fetching, ensure your IAM permissions are set up correctly for S3 access.
- If you encounter issues with certificate decrypting, ensure you set correct passphrase in MATCH_PASSWORD environment variable.
- For code signing issues, verify that your Apple Developer account has the necessary certificates and profiles, and that the bundle identifier in your Xcode project matches the one in your provisioning profile.

Security considerations

The following are security considerations for this tutorial.

- Ensure your S3 bucket has appropriate security settings, including encryption at rest. In particular, make sure the bucket has no public access and restrict access to only CodeBuild and the system that needs to have an access.
- Consider using AWS Secrets Manager for storing sensitive information like the MATCH_PASSWORD and FASTLANE_SESSION.

This sample provides a setup for iOS code signing with Fastlane in CodeBuild using Amazon S3 for certificate storage. You may need to adjust some steps based on your specific project requirements and CodeBuild environment. This approach leverages AWS services for enhanced security and integration within the AWS ecosystem.

Tutorial: Apple code signing with Fastlane in CodeBuild using GitHub for certificate storage

fastlane is a popular open source automation tool to automate beta deployments and releases for your iOS and Android apps. It handles all tedious tasks, like generating screenshots, dealing with code signing, and releasing your application.

This sample demonstrates how to set up Apple code signing using Fastlane in a CodeBuild project running on Mac fleet, with GitHub as the storage for certificates and provisioning profiles.

Prerequisites

To complete this tutorial, you must first have set up the following:

- An AWS account
- An Apple Developer account
- A private GitHub repository for storing certificates
- fastlane installed in your project - Guide to install fastlane

Step 1: Set up Fastlane Match with GitHub on your local machine

Fastlane Match is one of the Fastlane tools, and it allows for seamless configuration for code signing in both your local development environment and on CodeBuild. Fastlane Match stores all of your code signing certificates and provisioning profiles in a Git repository/S3 Bucket/Google Cloud Storage, and downloads and installs the necessary certificates and profiles when required.

In this example configuration, we will set up and use a Git repository for storage.

1. Initialize match in your project:

```
fastlane match init
```

2. When prompted, choose GitHub as the storage mode.
3. Update your `*Matchfile*` to use GitHub:

```
git_url("https://github.com/your-username/your-certificate-repo.git")
storage_mode("git")
type("development") # The default type, can be: appstore, adhoc, enterprise or
 development
```

> **Note**
>
> Make sure you enter HTTPS URL for your Git repository for fastlane to successfully authenticate and clone. Otherwise, you may see an authentication error when you attempt to use match.

Step 2: Set up your Fastfile

Create or update your `Fastfile` with the following lane.

On CodeBuild, Fastlane Match will need to be run every time you build and sign your app. The easiest way to do this is to add the `match` action to the lane which builds your app.

```
default_platform(:ios)

platform :ios do
  before_all do
    setup_ci
  end

  desc "Build and sign the app"
  lane :build do
    match(type: "appstore", readonly: true)
    gym(
      scheme: "YourScheme",
      export_method: "app-store"
    )
  end
end
```

> **Note**
>
> Make sure to add `setup_ci` to the `before_all` section in `Fastfile` for the match action to work correctly. This ensures that a temporary Fastlane keychain with the appropriate permissions is used. Without using this you may see build failures or inconsistent results.

Step 3: Run the `fastlane match` command to generate respective certificates and profiles

The fastlane match command for the given type (i.e. development, appstore, adhoc, enterprise) will generate the certificate and profile if not available in remote store. The certificates and profiles will be stored in GitHub by fastlane.

```
bundle exec fastlane match appstore
```

The command execution will be interactive and fastlane will ask to set pass phrase for decrypting the certificates.

Step 4: Create the application file for your project

Create or add the application file as appropriate for your project.

1. Create or add the Gymfile, Appfile, Snapfile, Deliverfile based on your project build requirements.
2. Commit the changes to your remote repository.

Step 5: Create environment variables in Secrets Manager

Create three secrets for storing the fastlane session cookie and matching pass phrase. For more information about creating secrets in Secrets Manager, see Create an AWS Secrets Manager secret.

1. Access your fastlane session cookie as follows.

   a. Secret key - `FASTLANE_SESSION`

   b. Secret value - session cookie generated from running the following command on your local machine.

   > **Note**
   >
   > This value is available after authentication in a local file: `~/.fastlane/spaceship/my_appleid_username/cookie`.

   ```
   fastlane spaceauth -u <Apple_account>
   ```

2. Fastlane Match pass phrase - To enable Fastlane Match to decrypt the certificates and profiles stored in the Git repository, it is necessary to add the encryption passphrase that you configured in the Match setup step to the CodeBuild project's environment variables.

   a. Secret key - `MATCH_PASSWORD`

   b. Secret value - `<match passphrase to decrypt certificates>`. The passphrase is set while generating the certificates in Step 3.

3. Fastlane `MATCH_GIT_BASIC_AUTHORIZATION` - set a basic authorization for *match*:

   a. Secret key:

      `MATCH_GIT_BASIC_AUTHORIZATION`

   b. Secret value - The value should be a base64 encoded string of your username and personal access token (PAT) in the format `username:password`. You can generate it using the following command:

      ```
      echo -n your_github_username:your_personal_access_token | base64
      ```

      You can generate your PAT on the GitHub console in **Your Profile > Settings > Developers Settings > Personal Access Token**. For more information, see the following guide: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens.

> **Note**
>
> While creating the above secrets in Secrets Manager, remember to give a secret name with the following prefix: `/CodeBuild/`

Step 6: Create a compute fleet

Create the compute fleet for your project.

1. In the console, go to CodeBuild and create a new compute fleet.
2. Choose `macOS` as the operating system and select an appropriate compute type and image.

Step 7: Create a project in CodeBuild

Create your project in CodeBuild.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.

2. Create a build project. For information, see Create a build project (console) and Run a build (console).
3. Set up your source provider (such as GitHub, CodeCommit). This is iOS project source repository and not certificates repository.
4. In **Environment**:
   - Choose **Reserved Capacity**.
   - For **Fleet**, select the fleet created above.
   - Provide the name of the service role that CodeBuild will create for you.
   - Provide the below environment variables.
     - Name: `MATCH_PASSWORD`, Value: *`<secrets arn>`*, Type: Secrets Manager (Secrets ARN created in step 5 for MATCH_PASSWORD)
     - Name: `FASTLANE_SESSION`, Value: *`<secrets arn>`*, Type: Secrets Manager (Secrets ARN created in step 5 for FASTLANE_SESSION)
     - Name: `MATCH_GIT_BASIC_AUTHORIZATION`, Value: *`<secrets ARN>`*, Type: Secrets Manager Secrets ARN (created in step 5 for `MATCH_GIT_BASIC_AUTHORIZATION`)
5. In **Buildspec**, add the following:

```
version: 0.2

phases:
  install:
    commands:
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo "Building and signing the app..."
      - bundle exec fastlane build
  post_build:
    commands:
      - echo "Build completed on date"

artifacts:
  files:
    - '*/.ipa'
  name: app-$(date +%Y-%m-%d)
```

Step 8: Run the build

Run the build. You can review the build status and logs in CodeBuild.

Once the job is completed, you will be able to view the log of the job.

Troubleshooting

- If you encounter issues accessing the GitHub repository, double-check your personal access token and the MATCH_GIT_BASIC_AUTHORIZATION environment variable.
- If you encounter issues with certificate decrypting, ensure you set correct passphrase in MATCH_PASSWORD environment variable.
- For code signing issues, verify that your Apple Developer account has the necessary certificates and profiles, and that the bundle identifier in your Xcode project matches the one in your provisioning profile.

Security considerations

The following are security considerations for this tutorial.

- Keep your GitHub repository for certificates private and regularly audit access.
- Consider using AWS Secrets Manager for storing sensitive information like the MATCH_PASSWORD and FASTLANE_SESSION.

This sample provides a setup for iOS code signing with Fastlane in CodeBuild using GitHub for certificate storage. You may need to adjust some steps based on your specific project requirements and CodeBuild environment. This approach leverages AWS services for enhanced security and integration within the AWS ecosystem.

Set artifact names at build time using semantic versioning

This sample contains example buildspec files that demonstrate how to specify an artifact name that is created at build time. A name specified in a buildspec file can incorporate Shell commands and environment variables to make it unique. A name you specify in a buildspec file overrides a name you enter in the console when you create your project.

If you build multiple times, using an artifact name specified in the buildspec file can ensure your output artifact file names are unique. For example, you can use a date and timestamp that is inserted into an artifact name at build time.

If you want to override the artifact name you entered in the console with a name in the buildspec file, do the following:

1. Set your build project to override the artifact name with a name in the buildspec file.
   - If you use the console to create your build project, select **Enable semantic versioning**. For more information, see Create a build project (console).
   - If you use the AWS CLI, set the `overrideArtifactName` to true in the JSON-formatted file passed to `create-project`. For more information, see Create a build project (AWS CLI).
   - If you use the AWS CodeBuild API, set the `overrideArtifactName` flag on the `ProjectArtifacts` object when a project is created or updated or a build is started.
2. Specify a name in the buildspec file. Use the following sample buildspec files as a guide.

This Linux example shows you how to specify an artifact name that includes the date the build is created:

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

This Linux example shows you how to specify an artifact name that uses a CodeBuild environment variable. For more information, see Environment variables in build environments.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
```

```
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

This Windows example shows you how to specify an artifact name that includes the date and time the build is created:

```
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")
```

This Windows example shows you how to specify an artifact name that uses a variable declared in the buildspec file and a CodeBuild environment variable. For more information, see Environment variables in build environments.

```
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
```

For more information, see Build specification reference for CodeBuild.

Run Microsoft Windows samples for CodeBuild

These samples use an AWS CodeBuild build environment running Microsoft Windows Server 2019, the .NET Framework, and the .NET Core SDK to build runtime files out of code written in F# and Visual Basic.

> **Important**
>
> Running these samples might result in charges to your AWS account. These include possible charges for CodeBuild and for AWS resources and actions related to Amazon S3, AWS KMS, and CloudWatch Logs. For more information, see CodeBuild pricing, Amazon S3 pricing, AWS Key Management Service pricing, and Amazon CloudWatch pricing.

Run the Windows samples

Use the following procedure to run the Windows samples.

To run the Windows samples

1. Create the files as described in the Directory structure and Files sections of this topic, and then upload them to an S3 input bucket or a CodeCommit or GitHub repository.

   > **Important**
   >
   > Do not upload *(root directory name)*, just the files inside of *(root directory name)*.
   > If you are using an S3 input bucket, be sure to create a ZIP file that contains the files, and then upload it to the input bucket. Do not add *(root directory name)* to the ZIP file, just the files inside of *(root directory name)*.

2. Create a build project. The build project must use the `mcr.microsoft.com/dotnet/framework/sdk:4.8` image to build .NET Framework projects.

   If you use the AWS CLI to create the build project, the JSON-formatted input to the `create-project` command might look similar to this. (Replace the placeholders with your own values.)

```
{
  "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. Run the build, and follow the steps in Run builds manually.
4. To get the build output artifact, in your S3 output bucket, download the *windows-build-output-artifact*`.zip` file to your local computer or instance. Extract the contents to get to the runtime and other files.

   - The runtime file for the F# sample using the .NET Framework, `FSharpHelloWorld.exe`, can be found in the `FSharpHelloWorld\bin\Debug` directory.
   - The runtime file for the Visual Basic sample using the .NET Framework, `VBHelloWorld.exe`, can be found in the `VBHelloWorld\bin\Debug` directory.

Directory structure

These samples assume the following directory structures.

F# and the .NET Framework

```
(root directory name)
```

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

Visual Basic and the .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
```

Files

These samples use the following files.

F# and the .NET Framework

`buildspec.yml` (in *`(root directory name)`*):

```
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 (in *(root directory name)*):

```
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 (in *(root directory name)*\FSharpHelloWorld):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
```

```
</configuration>
```

`AssemblyInfo.fs` (in *(root directory name)*\FSharpHelloWorld):

```
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 (in *(root directory name)*\FSharpHelloWorld):

```
<?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` (in *(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 and the .NET Framework

`buildspec.yml` (in *(root directory name)*):

```
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` (in *(root directory name)*):

```
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 (in *(root directory name)*\VBHelloWorld):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

HelloWorld.vb (in *(root directory name)*\VBHelloWorld):

```
Module HelloWorld

  Sub Main()
    MsgBox("Hello World")
  End Sub

End Module
```

VBHelloWorld.vbproj (in *(root directory name)*\VBHelloWorld):

```
<?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` (in *`(root directory name)`*`\VBHelloWorld\My Project`):

```
'------------------------------------------------------------------------------
' <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 (in *(root directory name)*\VBHelloWorld\My Project):

```
<?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 (in *(root directory name)*\VBHelloWorld\My Project):

```
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 (in *(root directory name)*\VBHelloWorld\My Project):

```
'------------------------------------------------------------------------------
' <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 (in *(root directory name)*\VBHelloWorld\My Project):

```
<?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 (in *(root directory name)*\VBHelloWorld\My Project):

```
'------------------------------------------------------------------------------
' <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` (in *`(root directory name)`*`\VBHelloWorld\My Project`):

```
<?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>
```

Plan a build in AWS CodeBuild

Before you use AWS CodeBuild, you must answer these questions:

1. **Where is the source code stored?** CodeBuild currently supports building from the following source code repository providers. The source code must contain a build specification (buildspec) file. A *buildspec* is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build. You can declare a buildspec in a build project definition.

Repository provider	Required	Documentation
CodeCommit	Repository name. (Optional) Commit ID associated with the source code.	See these topics in the *AWS CodeCommit User Guide*: Create a CodeCommit repository Create a commit in CodeCommit
Amazon S3	Input bucket name. Object name corresponding to the build input ZIP file that contains the source code. (Optional) Version ID associated with the build input ZIP file.	See these topics in the *Amazon S3 Getting Started Guide*: Create a bucket Add an object to a bucket

Repository provider	Required	Documentation
GitHub	Repository name. (Optional) Commit ID associated with the source code.	See this topic on the GitHub Help website: Create a repo
Bitbucket	Repository name. (Optional) Commit ID associated with the source code.	See this topic on the Bitbucket Cloud documentation website: Create a repository

2. **Which build commands do you need to run and in what order?** By default, CodeBuild downloads the build input from the provider you specify and uploads the build output to the bucket you specify. You use the buildspec to instruct how to turn the downloaded build input into the expected build output. For more information, see the Buildspec reference.
3. **Which runtimes and tools do you need to run the build?** For example, are you building for Java, Ruby, Python, or Node.js? Does the build need Maven or Ant or a compiler for Java, Ruby, or Python? Does the build need Git, the AWS CLI, or other tools?

   CodeBuild runs builds in build environments that use Docker images. These Docker images must be stored in a repository type supported by CodeBuild. These include the CodeBuild Docker image repository, Docker Hub, and Amazon Elastic Container Registry (Amazon ECR). For more information about the CodeBuild Docker image repository, see Docker images provided by CodeBuild.
4. **Do you need AWS resources that aren't provided automatically by CodeBuild? If so, which security policies do those resources need?** For example, you might need to modify the CodeBuild service role to allow CodeBuild to work with those resources.
5. **Do you want CodeBuild to work with your VPC?** If so, you need the VPC ID, the subnet IDs, and security group IDs for your VPC configuration. For more information, see Use AWS CodeBuild with Amazon Virtual Private Cloud.

After you have answered these questions, you should have the settings and resources you need to run a build successfully. To run your build, you can:

- Use the AWS CodeBuild console, AWS CLI, or AWS SDKs. For more information, see Run builds manually.
- Create or identify a pipeline in AWS CodePipeline, and then add a build or test action that instructs CodeBuild to automatically test your code, run your build, or both. For more information, see Use CodeBuild with CodePipeline.

Build specification reference for CodeBuild

This topic provides important reference information about build specification (buildspec) files. A *buildspec* is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build. You can include a buildspec as part of the source code or you can define a buildspec when you create a build project. For information about how a build spec works, see How CodeBuild works.

Topics

Buildspec file name and storage location

If you include a buildspec as part of the source code, by default, the buildspec file must be named `buildspec.yml` and placed in the root of your source directory.

You can override the default buildspec file name and location. For example, you can:

- Use a different buildspec file for different builds in the same repository, such as `buildspec_debug.yml` and `buildspec_release.yml`.
- Store a buildspec file somewhere other than the root of your source directory, such as `config/buildspec.yml` or in an S3 bucket. The S3 bucket must be in the same AWS Region as your build project. Specify the buildspec file using its ARN (for example, `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`).

You can specify only one buildspec for a build project, regardless of the buildspec file's name.

To override the default buildspec file name, location, or both, do one of the following:

- Run the AWS CLI `create-project` or `update-project` command, setting the `buildspec` value to the path to the alternate buildspec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. You can also do the equivalent with the `create project`

operation in the AWS SDKs. For more information, see Create a build project or Change build project settings.

- Run the AWS CLI `start-build` command, setting the `buildspecOverride` value to the path to the alternate buildspec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. You can also do the equivalent with the `start build` operation in the AWS SDKs. For more information, see Run builds manually.
- In an AWS CloudFormation template, set the `BuildSpec` property of `Source` in a resource of type `AWS::CodeBuild::Project` to the path to the alternate buildspec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. For more information, see the BuildSpec property in AWS CodeBuild project source in the *AWS CloudFormation User Guide*.

Buildspec syntax

Buildspec files must be expressed in YAML format.

If a command contains a character, or a string of characters, that is not supported by YAML, you must enclose the command in quotation marks (""). The following command is enclosed in quotation marks because a colon (:) followed by a space is not allowed in YAML. The quotation mark in the command is escaped (\").

```
"export PACKAGE_NAME=$(cat package.json | grep name | head -1 | awk -F: '{ print $2 }'
 | sed 's/[\",]//g')"
```

The buildspec has the following syntax:

```
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:
  # build-fanout:

phases:
  install:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    runtime-versions:
      runtime: version
      runtime: version
    commands:
      - command
      - command
    finally:
      - command
      - command

  pre_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
      - command
    finally:
      - command
      - command

  build:
    run-as: Linux-user-name
```

```
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
      - command
    finally:
      - command
      - command

  post_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
      - command
    finally:
      - command
      - command

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:
  key: key
  fallback-keys:
    - fallback-key
    - fallback-key
  action: restore | save
  paths:
    - path
    - path
```

The buildspec contains the following:

version

Required mapping. Represents the buildspec version. We recommend that you use `0.2`.

> **Note**
>
> Although version 0.1 is still supported, we recommend that you use version 0.2 whenever possible. For more information, see Buildspec versions.

run-as

Optional sequence. Available to Linux users only. Specifies a Linux user that runs commands in this buildspec file. `run-as` grants the specified user read and run permissions. When you specify `run-as` at the top of the buildspec file, it applies globally to all commands. If you don't want to specify a user for all buildspec file commands, you can specify one for commands in a phase by using `run-as` in one of the `phases` blocks. If `run-as` is not specified, then all commands run as the root user.

env

Optional sequence. Represents information for one or more custom environment variables.

Note

To protect sensitive information, the following are hidden in CodeBuild logs:

- AWS access key IDs. For more information, see Managing Access Keys for IAM Users in the *AWS Identity and Access Management User Guide*.
- Strings specified using the Parameter Store. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the *Amazon EC2 Systems Manager User Guide*.
- Strings specified using AWS Secrets Manager. For more information, see Key management.

env/**shell**

Optional sequence. Specifies the supported shell for Linux or Windows operating systems.

For Linux operating systems, supported shell tags are:

- `bash`
- `/bin/sh`

For Windows operating systems, supported shell tags are:

- `powershell.exe`
- `cmd.exe`

env/**variables**

Required if `env` is specified, and you want to define custom environment variables in plain text. Contains a mapping of *`key`*/*`value`* scalars, where each mapping represents a single custom environment variable in plain text. *`key`* is the name of the custom environment variable, and *`value`* is that variable's value.

Important

We strongly discourage the storing of sensitive values in environment variables. Environment variables can be displayed in plain text using tools such as the CodeBuild console and the AWS CLI. For sensitive values, we recommend that you use `parameter-store` or `secrets-manager` mapping instead, as described later in this section.

Any environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named `MY_VAR` with a value of `my_value`, and you set an environment variable named `MY_VAR` with a value of `other_value`, then `my_value` is replaced by `other_value`. Similarly, if the Docker image already contains an environment variable named `PATH` with a value of `/usr/local/sbin:/usr/local/bin`, and you set an environment variable named `PATH` with a value of `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`.
Do not set any environment variable with a name that starts with `CODEBUILD_`. This prefix is reserved for internal use.
If an environment variable with the same name is defined in multiple places, the value is determined as follows:

- The value in the start build operation call takes highest precedence. You can add or override environment variables when you create a build. For more information, see Run AWS CodeBuild builds manually.
- The value in the build project definition takes next precedence. You can add environment variables at the project level when you create or edit a project. For more information, see Create a build project in AWS CodeBuild and Change build project settings in AWS CodeBuild.
- The value in the buildspec declaration takes lowest precedence.

env/**parameter-store**

Required if `env` is specified, and you want to retrieve custom environment variables stored in Amazon EC2 Systems Manager Parameter Store. Contains a mapping of *`key`*/*`value`* scalars, where each mapping represents a single custom environment variable stored in Amazon EC2 Systems Manager Parameter Store. *`key`* is the name you use later in your build commands to refer to this custom environment variable, and *`value`* is the name of the custom environment variable stored in Amazon EC2 Systems Manager Parameter Store. To store sensitive values, see Systems Manager Parameter Store and Walkthrough: Create and test a String parameter (console) in the *Amazon EC2 Systems Manager User Guide*.

Important

To allow CodeBuild to retrieve custom environment variables stored in Amazon EC2 Systems Manager Parameter Store, you must add the `ssm:GetParameters` action to

your CodeBuild service role. For more information, see Allow CodeBuild to interact with other AWS services.
Any environment variables you retrieve from Amazon EC2 Systems Manager Parameter Store replace existing environment variables. For example, if the Docker image already contains an environment variable named `MY_VAR` with a value of `my_value`, and you retrieve an environment variable named `MY_VAR` with a value of `other_value`, then `my_value` is replaced by `other_value`. Similarly, if the Docker image already contains an environment variable named `PATH` with a value of `/usr/local/sbin:/usr/local/bin`, and you retrieve an environment variable named `PATH` with a value of `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`.
Do not store any environment variable with a name that starts with `CODEBUILD_`. This prefix is reserved for internal use.
If an environment variable with the same name is defined in multiple places, the value is determined as follows:

- The value in the start build operation call takes highest precedence. You can add or override environment variables when you create a build. For more information, see Run AWS CodeBuild builds manually.
- The value in the build project definition takes next precedence. You can add environment variables at the project level when you create or edit a project. For more information, see Create a build project in AWS CodeBuild and Change build project settings in AWS CodeBuild.
- The value in the buildspec declaration takes lowest precedence.

env/**secrets-manager**

Required if you want to retrieve custom environment variables stored in AWS Secrets Manager. Specify a Secrets Manager `reference-key` using the following pattern:

`<key>``:`*`<secret-id>`*`:`*`<json-key>`*`:`*`<version-stage>`*`:`*`<version-id>`*

`<key>`

(Required) The local environment variable name. Use this name to access the variable during the build.

<secret-id>

(Required) The name or Amazon Resource Name (ARN) that serves as a unique identifier for the secret. To access a secret in your AWS account, simply specify the secret name. To access a secret in a different AWS account, specify the secret ARN.

<json-key>

(Optional) Specifies the key name of the Secrets Manager key-value pair whose value you want to retrieve. If you do not specify a `json-key`, CodeBuild retrieves the entire secret text.

<version-stage>

(Optional) Specifies the secret version that you want to retrieve by the staging label attached to the version. Staging labels are used to keep track of different versions during the rotation process. If you use `version-stage`, don't specify `version-id`. If you don't specify a version stage or version ID, the default is to retrieve the version with the version stage value of `AWSCURRENT`.

<version-id>

(Optional) Specifies the unique identifier of the version of the secret that you want to use. If you specify `version-id`, don't specify `version-stage`. If you don't specify a version stage or version ID, the default is to retrieve the version with the version stage value of `AWSCURRENT`.

In the following example, `TestSecret` is the name of the key-value pair stored in Secrets Manager. The key for `TestSecret` is `MY_SECRET_VAR`. You access the variable during the build using the `LOCAL_SECRET_VAR` name.

```
env:
  secrets-manager:
    LOCAL_SECRET_VAR: "TestSecret:MY_SECRET_VAR"
```

For more information, see What is AWS Secrets Manager in the *AWS Secrets Manager User Guide*.

env/**exported-variables**

Optional mapping. Used to list environment variables you want to export. Specify the name of each variable you want to export on a separate line under `exported-variables`. The

variable you want to export must be available in your container during the build. The variable you export can be an environment variable.

Exported environment variables are used in conjunction with AWS CodePipeline to export environment variables from the current build stage to subsequent stages in the pipeline. For more information, see Working with variables in the *AWS CodePipeline User Guide*.

During a build, the value of a variable is available starting with the `install` phase. It can be updated between the start of the `install` phase and the end of the `post_build` phase. After the `post_build` phase ends, the value of exported variables cannot change.

> **Note**
>
> The following cannot be exported:
>
> - Amazon EC2 Systems Manager Parameter Store secrets specified in the build project.
> - Secrets Manager secrets specified in the build project
> - Environment variables that start with `AWS_`.

env/**git-credential-helper**

Optional mapping. Used to indicate if CodeBuild uses its Git credential helper to provide Git credentials. `yes` if it is used. Otherwise, no or not specified. For more information, see gitcredentials on the Git website.

> **Note**
>
> `git-credential-helper` is not supported for builds that are triggered by a webhook for a public Git repository.

proxy

Optional sequence. Used to represent settings if you run your build in an explicit proxy server. For more information, see Run CodeBuild in an explicit proxy server.

proxy/**upload-artifacts**

Optional mapping. Set to `yes` if you want your build in an explicit proxy server to upload artifacts. The default is no.

proxy/**logs**

Optional mapping. Set to `yes` for your build in a explicit proxy server to create CloudWatch logs. The default is no.

phases

Required sequence. Represents the commands CodeBuild runs during each phase of the build.

> **Note**
>
> In buildspec version 0.1, CodeBuild runs each command in a separate instance of the default shell in the build environment. This means that each command runs in isolation from all other commands. Therefore, by default, you cannot run a single command that relies on the state of any previous commands (for example, changing directories or setting environment variables). To get around this limitation, we recommend that you use version 0.2, which solves this issue. If you must use buildspec version 0.1, we recommend the approaches in Shells and commands in build environments.

phases/*/**run-as**

Optional sequence. Use in a build phase to specify a Linux user that runs its commands. If `run-as` is also specified globally for all commands at the top of the buildspec file, then the phase-level user takes precedence. For example, if globally `run-as` specifies User-1, and for the `install` phase only a `run-as` statement specifies User-2, then all commands in then buildspec file are run as User-1 *except* commands in the `install` phase, which are run as User-2.

phases/*/**on-failure**

Optional sequence. Specifies the action to take if a failure occurs during the phase. This can be one of the following values:

- `ABORT` - Abort the build.
- `CONTINUE` - Continue to the next phase.

- RETRY - Retry the build up to 3 times with an error message that matches the regular expression `.*`.
- RETRY-*`count`* - Retry the build for a specified number of times, as represented by *`count`* with an error message that matches the regular expression `.*`. Note that *`count`* must be between 0 and 100. For example, valid values include `RETRY-4` and `RETRY-8`.
- RETRY-*`regex`* - Retry the build up to 3 times, and use *`regex`* to include a regular expression to match a specifed error message. For example, valid values include `Retry-.*Error: Unable to connect to database.*` and `RETRY-invalid+`.
- RETRY-*`count`*-*`regex`* - Retry the build for a specified number of times, as represented by *`count`*. Note that *`count`* must be between 0 and 100. You can also use *`regex`* to include a regular expression to match the error message. For example, valid values include `Retry-3-.*connection timed out.*` and `RETRY-8-invalid+`.

If this property is not specified, the failure process follows the transition phases as shown in Build phase transitions.

phases/*/**finally**

Optional block. Commands specified in a `finally` block are run after commands in the `commands` block. The commands in a `finally` block are run even if a command in the `commands` block fails. For example, if the `commands` block contains three commands and the first fails, CodeBuild skips the remaining two commands and runs any commands in the `finally` block. The phase is successful when all commands in the `commands` and the `finally` blocks run successfully. If any command in a phase fails, the phase fails.

The allowed build phase names are:

phases/**install**

Optional sequence. Represents the commands, if any, that CodeBuild runs during installation. We recommend that you use the `install` phase only for installing packages in the build environment. For example, you might use this phase to install a code testing framework such as Mocha or RSpec.

phases/install/**runtime-versions**

Optional sequence. A runtime version is supported with the Ubuntu standard image 5.0 or later and the Amazon Linux 2 standard image 4.0 or later. If specified, at least one runtime must be included in this section. Specify a runtime using a specific version, a major version

followed by `.x` to specify that CodeBuild uses that major version with its latest minor version, or `latest` to use the most recent major and minor version (for example, `ruby: 3.2`, `nodejs: 18.x`, or `java: latest`). You can specify the runtime using a number or an environment variable. For example, if you use the Amazon Linux 2 standard image 4.0, then the following specifies that version 17 of Java, the latest minor version of python version 3, and a version contained in an environment variable of Ruby is installed. For more information, see Docker images provided by CodeBuild.

```
phases:
  install:
    runtime-versions:
      java: corretto8
      python: 3.x
      ruby: "$MY_RUBY_VAR"
```

You can specify one or more runtimes in the `runtime-versions` section of your buildspec file. If your runtime is dependent upon another runtime, you can also specify its dependent runtime in the buildspec file. If you do not specify any runtimes in the buildspec file, CodeBuild chooses the default runtimes that are available in the image you use. If you specify one or more runtimes, CodeBuild uses only those runtimes. If a dependent runtime is not specified, CodeBuild attempts to choose the dependent runtime for you.

If two specified runtimes conflict, the build fails. For example, `android: 29` and `java: openjdk11` conflict, so if both are specified, the build fails.

For more information about the available runtimes, see Available runtimes.

> **Note**
>
> If you specify a `runtime-versions` section and use an image other than Ubuntu Standard Image 2.0 or later, or the Amazon Linux 2 (AL2) standard image 1.0 or later, the build issues the warning, "`Skipping install of runtimes. Runtime version selection is not supported by this build image.`"

phases/install/**commands**

Optional sequence. Contains a sequence of scalars, where each scalar represents a single command that CodeBuild runs during installation. CodeBuild runs each command, one at a time, in the order listed, from beginning to end.

phases/**pre_build**

Optional sequence. Represents the commands, if any, that CodeBuild runs before the build. For example, you might use this phase to sign in to Amazon ECR, or you might install npm dependencies.

phases/pre_build/**commands**

Required sequence if `pre_build` is specified. Contains a sequence of scalars, where each scalar represents a single command that CodeBuild runs before the build. CodeBuild runs each command, one at a time, in the order listed, from beginning to end.

phases/**build**

Optional sequence. Represents the commands, if any, that CodeBuild runs during the build. For example, you might use this phase to run Mocha, RSpec, or sbt.

phases/build/**commands**

Required if `build` is specified. Contains a sequence of scalars, where each scalar represents a single command that CodeBuild runs during the build. CodeBuild runs each command, one at a time, in the order listed, from beginning to end.

phases/**post_build**

Optional sequence. Represents the commands, if any, that CodeBuild runs after the build. For example, you might use Maven to package the build artifacts into a JAR or WAR file, or you might push a Docker image into Amazon ECR. Then you might send a build notification through Amazon SNS.

phases/post_build/**commands**

Required if `post_build` is specified. Contains a sequence of scalars, where each scalar represents a single command that CodeBuild runs after the build. CodeBuild runs each command, one at a time, in the order listed, from beginning to end.

reports

report-group-name-or-arn

Optional sequence. Specifies the report group that the reports are sent to. A project can have a maximum of five report groups. Specify the ARN of an existing report group, or the name of a new report group. If you specify a name, CodeBuild creates a report group using your project name and the name you specify in the format `<project-name>-<report-group-name>`. The report group name can also be set using an environment variable in the buildspec such as `$REPORT_GROUP_NAME`. For more information, see Report group naming.

reports/<report-group>/**files**

Required sequence. Represents the locations that contain the raw data of test results generated by the report. Contains a sequence of scalars, with each scalar representing a separate location where CodeBuild can find test files, relative to the original build location or, if set, the `base-directory`. Locations can include the following:

- A single file (for example, `my-test-report-file.json`).
- A single file in a subdirectory (for example, *`my-subdirectory`*`/my-test-report-file.json` or *`my-parent-subdirectory`*`/`*`my-subdirectory`*`/my-test-report-file.json`).
- `'**/*'` represents all files recursively.
- *`my-subdirectory`*`/*` represents all files in a subdirectory named *`my-subdirectory`*.
- *`my-subdirectory`*`/**/*` represents all files recursively starting from a subdirectory named *`my-subdirectory`*.

reports/<report-group>/**file-format**

Optional mapping. Represents the report file format. If not specified, `JUNITXML` is used. This value is not case sensitive. Possible values are:

Test reports

`CUCUMBERJSON`

Cucumber JSON

`JUNITXML`

JUnit XML

`NUNITXML`

NUnit XML

`NUNIT3XML`

NUnit 3 XML

`TESTNGXML`

TestNG XML

`VISUALSTUDIOTRX`

Visual Studio TRX

Code coverage reports

`CLOVERXML`

Clover XML

`COBERTURAXML`

Cobertura XML

`JACOCOXML`

JaCoCo XML

`SIMPLECOV`

SimpleCov JSON

> **Note**
>
> CodeBuild accepts JSON code coverage reports generated by simplecov, not simplecov-json.

reports/<report-group>/**base-directory**

Optional mapping. Represents one or more top-level directories, relative to the original build location, that CodeBuild uses to determine where to find the raw test files.

reports/<report-group>/**discard-paths**

Optional. Specifies if the report file directories are flattened in the output. If this is not specified, or contains no, report files are output with their directory structure intact. If this contains `yes`, all of the test files are placed in the same output directory. For example, if a path to a test result is `com/myapp/mytests/TestResult.xml`, specifying `yes` will place this file in `/TestResult.xml`.

artifacts

Optional sequence. Represents information about where CodeBuild can find the build output and how CodeBuild prepares it for uploading to the S3 output bucket. This sequence is not required if, for example, you are building and pushing a Docker image to Amazon ECR, or you are running unit tests on your source code, but not building it.

> **Note**
>
> Amazon S3 metadata has a CodeBuild header named `x-amz-meta-codebuild-buildarn` which contains the `buildArn` of the CodeBuild build that publishes artifacts to Amazon S3. The `buildArn` is added to allow source tracking for notifications and to reference which build the artifact is generated from.

artifacts/**files**

Required sequence. Represents the locations that contain the build output artifacts in the build environment. Contains a sequence of scalars, with each scalar representing a separate location where CodeBuild can find build output artifacts, relative to the original build location or, if set, the base directory. Locations can include the following:

- A single file (for example, `my-file.jar`).
- A single file in a subdirectory (for example, *`my-subdirectory`*`/my-file.jar` or *`my-parent-subdirectory`*`/`*`my-subdirectory`*`/my-file.jar`).
- `'**/*'` represents all files recursively.
- *`my-subdirectory`*`/*` represents all files in a subdirectory named *`my-subdirectory`*.
- *`my-subdirectory`*`/**/*` represents all files recursively starting from a subdirectory named *`my-subdirectory`*.

When you specify build output artifact locations, CodeBuild can locate the original build location in the build environment. You do not have to prepend your build artifact output locations with the path to the original build location or specify `./` or similar. If you want to know the path to this location, you can run a command such as `echo $CODEBUILD_SRC_DIR` during a build. The location for each build environment might be slightly different.

artifacts/**name**

Optional name. Specifies a name for your build artifact. This name is used when one of the following is true.

- You use the CodeBuild API to create your builds and the `overrideArtifactName` flag is set on the `ProjectArtifacts` object when a project is updated, a project is created, or a build is started.
- You use the CodeBuild console to create your builds, a name is specified in the buildspec file, and you select **Enable semantic versioning** when you create or update a project. For more information, see Create a build project (console).

You can specify a name in the buildspec file that is calculated at build time. The name specified in a buildspec file uses the Shell command language. For example, you can append a date and time to your artifact name so that it is always unique. Unique artifact names prevent artifacts from being overwritten. For more information, see Shell command language.

- This is an example of an artifact name appended with the date the artifact is created.

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

- This is an example of an artifact name that uses a CodeBuild environment variable. For more information, see Environment variables in build environments.

```
version: 0.2
phases:
  build:
    commands:
```

```
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

- This is an example of an artifact name that uses a CodeBuild environment variable with the artifact's creation date appended to it.

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

You can add path information to the name so that the named artifacts are placed in directories based on the path in the name. In this example, build artifacts are placed in the output under `builds/<build number>/my-artifacts`.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: builds/$CODEBUILD_BUILD_NUMBER/my-artifacts
```

artifacts/**discard-paths**

Optional. Specifies if the build artifact directories are flattened in the output. If this is not specified, or contains no, build artifacts are output with their directory structure intact. If this contains `yes`, all of the build artifacts are placed in the same output directory. For example, if a path to a file in the build output artifact is `com/mycompany/app/HelloWorld.java`, specifying `yes` will place this file in `/HelloWorld.java`.

artifacts/**base-directory**

Optional mapping. Represents one or more top-level directories, relative to the original build location, that CodeBuild uses to determine which files and subdirectories to include in the build output artifact. Valid values include:

- A single top-level directory (for example, `my-directory`).
- `'my-directory*'` represents all top-level directories with names starting with `my-directory`.

Matching top-level directories are not included in the build output artifact, only their files and subdirectories.

You can use `files` and `discard-paths` to further restrict which files and subdirectories are included. For example, for the following directory structure:

```
.
### my-build-1
#   ### my-file-1.txt
### my-build-2
    ### my-file-2.txt
    ### my-subdirectory
        ### my-file-3.txt
```

And for the following `artifacts` sequence:

```
artifacts:
  files:
    - '*/my-file-3.txt'
  base-directory: my-build-2
```

The following subdirectory and file would be included in the build output artifact:

```
.
### my-subdirectory
    ### my-file-3.txt
```

While for the following `artifacts` sequence:

```
artifacts:
```

```
  files:
    - '**/*'
  base-directory: 'my-build*'
  discard-paths: yes
```

The following files would be included in the build output artifact:

```
.
### my-file-1.txt
### my-file-2.txt
### my-file-3.txt
```

artifacts/**exclude-paths**

Optional mapping. Represents one or more paths, relative to `base-directory`, that CodeBuild will exclude from the build artifacts. The asterisk (*) character matches zero or more characters of a name component without crossing folder boundaries. A double asterisk (**) matches zero or more characters of a name component across all directories.

Examples of exclude-paths include the following:

- To exclude a file from all directories: "**/*file-name*/**/*"
- To exclude all dot folders: "**/.*/**/*"
- To exclude all dot files: "**/.*"

artifacts/**enable-symlinks**

Optional. If the output type is `ZIP`, specifies if internal symbolic links are preserved in the ZIP file. If this contains `yes`, all internal symbolic links in the source will be preserved in the artifacts ZIP file.

artifacts/**s3-prefix**

Optional. Specifies a prefix used when the artifacts are output to an Amazon S3 bucket and the namespace type is `BUILD_ID`. When used, the output path in the bucket is `<s3-prefix>/<build-id>/<name>.zip`.

artifacts/**secondary-artifacts**

Optional sequence. Represents one or more artifact definitions as a mapping between an artifact identifier and an artifact definition. Each artifact identifiers in this block must match

an artifact defined in the `secondaryArtifacts` attribute of your project. Each separate definition has the same syntax as the `artifacts` block above.

> **Note**
>
> The `artifacts/files` sequence is always required, even when there are only secondary artifacts defined.

For example, if your project has the following structure:

```
{
  "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"
    }
  ]
}
```

Then your buildspec looks like the following:

```
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

Optional sequence. Represents information about where CodeBuild can prepare the files for uploading cache to an S3 cache bucket. This sequence is not required if the cache type of the project is `No Cache`.

cache/**key**

Optional sequence. Represents the primary key used when search or restore a cache. CodeBuild does an exact match for the primary key.

Here is an example for the key:

```
key: npm-key-$(codebuild-hash-files package-lock.json) }
```

cache/**fallback-keys**

Optional sequence. Represents a list of fallback keys used sequentially when a cache cannot be found using the primary key. Up to five fallback keys are supported, and each is matched using a prefix search. This sequence will be ignored if **key** is not provided.

Here is an example for the fallback-keys:

```
fallback-keys:
    - npm-key-$(codebuild-hash-files package-lock.json) }
    - npm-key-
    - npm-
```

cache/**action**

Optional sequence. Specifies the action to perform on the cache. Valid values include:

- `restore` which only restores the cache without saving updates.
- `save` which only saves the cache without restoring a previous version.

If no value is provided, CodeBuild defaults to performing both restore and save.

cache/**paths**

Required sequence. Represents the locations of the cache. Contains a sequence of scalars, with each scalar representing a separate location where CodeBuild can find build output artifacts, relative to the original build location or, if set, the base directory. Locations can include the following:

- A single file (for example, `my-file.jar`).
- A single file in a subdirectory (for example, *`my-subdirectory`*`/my-file.jar` or *`my-parent-subdirectory`*`/`*`my-subdirectory`*`/my-file.jar`).
- `'**/*'` represents all files recursively.
- *`my-subdirectory`*`/*` represents all files in a subdirectory named *`my-subdirectory`*.
- *`my-subdirectory`*`/**/*` represents all files recursively starting from a subdirectory named *`my-subdirectory`*.

Important

Because a buildspec declaration must be valid YAML, the spacing in a buildspec declaration is important. If the number of spaces in your buildspec declaration is invalid, builds might fail immediately. You can use a YAML validator to test whether your buildspec declarations are valid YAML.

If you use the AWS CLI, or the AWS SDKs to declare a buildspec when you create or update a build project, the buildspec must be a single string expressed in YAML format, along with required whitespace and newline escape characters. There is an example in the next section.

If you use the CodeBuild or AWS CodePipeline consoles instead of a buildspec.yml file, you can insert commands for the `build` phase only. Instead of using the preceding syntax, you list, in a single line, all of the commands that you want to run during the build phase. For multiple commands, separate each command by && (for example, `mvn test && mvn package`).

You can use the CodeBuild or CodePipeline consoles instead of a buildspec.yml file to specify the locations of the build output artifacts in the build environment. Instead of using the preceding syntax, you list, in a single line, all of the locations. For multiple locations,

separate each location with a comma (for example, `buildspec.yml, target/my-app.jar`).

Buildspec example

Here is an example of a buildspec.yml file.

```
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/**/*'
```

Here is an example of the preceding buildspec, expressed as a single string, for use with the AWS CLI, or the AWS SDKs.

```
"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/**/*'"
```

Here is an example of the commands in the `build` phase, for use with the CodeBuild or CodePipeline consoles.

```
echo Build started on `date` && mvn install
```

In these examples:

- A custom environment variable, in plain text, with the key of `JAVA_HOME` and the value of `/usr/lib/jvm/java-8-openjdk-amd64`, is set.
- A custom environment variable named `dockerLoginPassword` you stored in Amazon EC2 Systems Manager Parameter Store is referenced later in build commands by using the key `LOGIN_PASSWORD`.
- You cannot change these build phase names. The commands that are run in this example are `apt-get update -y` and `apt-get install -y maven` (to install Apache Maven), `mvn install` (to compile, test, and package the source code into a build output artifact and to install the build output artifact in its internal repository), `docker login` (to sign in to Docker with the password that corresponds to the value of the custom environment variable `dockerLoginPassword` you set in Amazon EC2 Systems Manager Parameter Store), and several `echo` commands. The `echo` commands are included here to show how CodeBuild runs commands and the order in which it runs them.
- `files` represents the files to upload to the build output location. In this example, CodeBuild uploads the single file `messageUtil-1.0.jar`. The `messageUtil-1.0.jar` file can be found in the relative directory named `target` in the build environment. Because `discard-paths: yes` is specified, `messageUtil-1.0.jar` is uploaded directly (and not to an intermediate `target` directory). The file name `messageUtil-1.0.jar` and the relative directory name of `target` is based on the way Apache Maven creates and stores build output artifacts for this example only. In your own scenarios, these file names and directories will be different.
- `reports` represents two report groups that generate reports during the build:
  - `arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1` specifies the ARN of a report group. Test results generated by the test

framework are in the `target/tests/reports` directory. The file format is `JunitXml` and the path is not removed from the files that contain test results.

- `reportGroupCucumberJson` specifies a new report group. If the name of the project is `my-project`, a report group with the name `my-project-reportGroupCucumberJson` is created when a build is run. Test results generated by the test framework are in `cucumber/target/cucumber-tests.xml`. The test file format is `CucumberJson` and the path is removed from the files that contain test results.

Buildspec versions

The following table lists the buildspec versions and the changes between versions.

Version	Changes
0.2	• `environment_variables` has been renamed to `env`. • `plaintext` has been renamed to `variables`. • The `type` property for `artifacts` has been deprecated. • In version 0.1, AWS CodeBuild runs each build command in a separate instance of the default shell in the build environment. In version 0.2, CodeBuild runs all build commands in the same instance of the default shell in the build environment.
0.1	This is the initial definition of the build specification format.

Batch build buildspec reference

This topic contains the buildspec reference for batch build properties.

batch

Optional mapping. The batch build settings for the project.

batch/**fast-fail**

Optional. Specifies the behavior of the batch build when one or more build tasks fail.

`false`

The default value. All running builds will complete.

`true`

All running builds will be stopped when one of the build tasks fails.

By default, all batch build tasks run with the build settings such as `env` and `phases`, specified in the buildspec file. You can override the default build settings by specifying different `env` values or a different buildspec file in the `batch/<batch-type>/buildspec` parameter.

The contents of the `batch` property varies based on the type of batch build being specified. The possible batch build types are:

- `batch/build-graph`
- `batch/build-list`
- `batch/build-matrix`
- `batch/build-fanout`

batch/build-graph

Defines a *build graph*. A build graph defines a set of tasks that have dependencies on other tasks in the batch. For more information, see Build graph.

This element contains an array of build tasks. Each build task contains the following properties.

identifier

Required. The identifier of the task.

buildspec

Optional. The path and file name of the buildspec file to use for this task. If this parameter is not specified, the current buildspec file is used.

debug-session

Optional. A Boolean value that indicates whether session debugging is enabled for this batch build. For more information about session debugging, see Debug builds with Session Manager.

`false`

Session debugging is disabled.

`true`

Session debugging is enabled.

depend-on

Optional. An array of task identifiers that this task depends on. This task will not run until these tasks are completed.

env

Optional. The build environment overrides for the task. This can contain the following properties:

compute-type

The identifier of the compute type to use for the task. See **computeType** in the section called "Build environment compute modes and types" for possible values.

fleet

The identifier of the fleet to use for the task. See the section called "Run builds on reserved capacity fleets" for more information.

image

The identifier of the image to use for the task. See **Image identifier** in the section called "Docker images provided by CodeBuild" for possible values.

privileged-mode

A Boolean value that indicates whether to run the Docker daemon inside a Docker container. Set to `true` only if the build project is used to build Docker images. Otherwise, a build that attempts to interact with the Docker daemon fails. The default setting is `false`.

type

The identifier of the environment type to use for the task. See **Environment type** in the section called "Build environment compute modes and types" for possible values.

variables

The environment variables that will be present in the build environment. See env/**variables** for more information.

> **Note**
>
> Note that **compute-type** and **fleet** cannot be provided in the same identifer of a single build.

ignore-failure

Optional. A Boolean value that indicates if a failure of this build task can be ignored.

`false`

The default value. If this build task fails, the batch build will fail.

`true`

If this build task fails, the batch build can still succeed.

The following is an example of a build graph buildspec entry:

```
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
  - identifier: build4
    env:
      compute-type: ARM_LAMBDA_1GB
  - identifier: build5
    env:
      fleet: fleet_name
```

batch/build-list

Defines a *build list*. A build list is used to define a number of tasks that run in parallel. For more information, see Build list.

This element contains an array of build tasks. Each build task contains the following properties.

identifier

Required. The identifier of the task.

buildspec

Optional. The path and file name of the buildspec file to use for this task. If this parameter is not specified, the current buildspec file is used.

debug-session

Optional. A Boolean value that indicates whether session debugging is enabled for this batch build. For more information about session debugging, see Debug builds with Session Manager.

`false`

Session debugging is disabled.

`true`

Session debugging is enabled.

env

Optional. The build environment overrides for the task. This can contain the following properties:

compute-type

The identifier of the compute type to use for the task. See **computeType** in the section called "Build environment compute modes and types" for possible values.

fleet

The identifier of the fleet to use for the task. See the section called "Run builds on reserved capacity fleets" for more information.

image

The identifier of the image to use for the task. See **Image identifier** in the section called "Docker images provided by CodeBuild" for possible values.

privileged-mode

A Boolean value that indicates whether to run the Docker daemon inside a Docker container. Set to `true` only if the build project is used to build Docker images. Otherwise, a build that attempts to interact with the Docker daemon fails. The default setting is `false`.

type

The identifier of the environment type to use for the task. See **Environment type** in the section called "Build environment compute modes and types" for possible values.

variables

The environment variables that will be present in the build environment. See env/**variables** for more information.

> **Note**
>
> Note that **compute-type** and **fleet** cannot be provided in the same identifer of a single build.

ignore-failure

Optional. A Boolean value that indicates if a failure of this build task can be ignored.

`false`

The default value. If this build task fails, the batch build will fail.

`true`

If this build task fails, the batch build can still succeed.

The following is an example of a build list buildspec entry:

```
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
    - identifier: build3
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build4
      env:
        fleet: fleet_name
    - identifier: build5
      env:
        compute-type: GENERAL_LINUX_XLAGRE
```

batch/build-matrix

Defines a *build matrix*. A build matrix defines tasks with different configurations that run in parallel. CodeBuild creates a separate build for each possible configuration combination. For more information, see Build matrix.

static

The static properties apply to all build tasks.

ignore-failure

Optional. A Boolean value that indicates if a failure of this build task can be ignored.

`false`

The default value. If this build task fails, the batch build will fail.

`true`

If this build task fails, the batch build can still succeed.

env

Optional. The build environment overrides for all tasks.

privileged-mode

A Boolean value that indicates whether to run the Docker daemon inside a Docker container. Set to `true` only if the build project is used to build Docker images. Otherwise, a build that attempts to interact with the Docker daemon fails. The default setting is `false`.

type

The identifier of the environment type to use for the task. See **Environment type** in the section called “Build environment compute modes and types” for possible values.

dynamic

The dynamic properties define the build matrix.

buildspec

Optional. An array that contains the path and file names of the buildspec files to use for these tasks. If this parameter is not specified, the current buildspec file is used.

env

Optional. The build environment overrides for these tasks.

compute-type

An array that contains the identifiers of the compute types to use for these tasks. See **computeType** in the section called “Build environment compute modes and types” for possible values.

image

An array that contains the identifiers of the images to use for these tasks. See **Image identifier** in the section called "Docker images provided by CodeBuild" for possible values.

variables

An array that contains the environment variables that will be present in the build environments for these tasks. See env/**variables** for more information.

The following is an example of a build matrix buildspec entry:

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
        variables:
          MY_VAR:
            - VALUE1
            - VALUE2
            - VALUE3
```

For more information, see Build matrix.

`batch/build-fanout`

Defines a *build fanout*. A build fanout is used to define a task that is split into multiple builds that runs in parallel. For more information, see Execute parallel tests in batch builds.

This element contains an build task that can be split into multiple builds. The `build-fanout` section contains the following properties.

parallelism

Required. The number of builds that will run tests in parallel.

ignore-failure

Optional. A boolean value that indicates if failure in any of the fanout build tasks can be ignored. This value of **ignore-failure** will be applied to all the fanout builds.

false

The default value. If any fanout build task fails, the batch build will fail.

true

If any fanout build task fails, the batch build can still succeed.

The following is an example of a build fanout buildspec entry:

```
version: 0.2

batch:
   fast-fail: false
   build-fanout:
     parallelism: 5
     ignore-failure: false

phases:
  install:
    commands:
      - npm install
   build:
    commands:
      - mkdir -p test-results
      - cd test-results
      - |
        codebuild-tests-run \
         --test-command 'npx jest --runInBand --coverage' \
         --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
         --sharding-strategy 'equal-distribution'
```

For more information, see Build fanout and Use the `codebuild-tests-run` CLI command.

Build environment reference for AWS CodeBuild

When you call AWS CodeBuild to run a build, you must provide information about the build environment. A *build environment* represents a combination of operating system, programming language runtime, and tools that CodeBuild uses to run a build. For information about how a build environment works, see How CodeBuild works.

A build environment contains a Docker image. For information, see the Docker glossary on the Docker Docs website.

When you provide information to CodeBuild about the build environment, you specify the identifier of a Docker image in a supported repository type. These include the CodeBuild Docker image repository, publicly available images in Docker Hub, and Amazon Elastic Container Registry (Amazon ECR) repositories that your AWS account has permissions to access.

- We recommend that you use Docker images stored in the CodeBuild Docker image repository, because they are optimized for use with the service. For more information, see Docker images provided by CodeBuild.
- To get the identifier of a publicly available Docker image stored in Docker Hub, see Searching for Repositories on the Docker Docs website.
- To learn how to work with Docker images stored in Amazon ECR repositories in your AWS account, see Amazon ECR sample.

In addition to a Docker image identifier, you also specify a set of computing resources that the build environment uses. For more information, see Build environment compute modes and types.

Topics

Docker images provided by CodeBuild

A *supported image* is the latest major version of an image available in CodeBuild and is updated with minor and patch version updates. CodeBuild optimizes the provisioning duration of builds with supported images by caching them in the machine's Amazon Machine Images (AMI). If you want to benefit from caching and minimize the provisioning duration of your build, select **Always use the latest image for this runtime version** in the **Image version** section of the CodeBuild console instead of a more granular version, such as `aws/codebuild/amazonlinux-x86_64-standard:4.0-1.0.0`.

Topics

Obtain the list of current Docker images

CodeBuild frequently updates the list of Docker images to add the latest images and deprecate old images. To get the most current list, do one of the following:

- In the CodeBuild console, in the **Create build project** wizard or **Edit Build Project** page, for **Environment image**, choose **Managed image**. Choose from the **Operating system**, **Runtime**, and **Runtime version** drop-down lists. For more information, see Create a build project (console) or Change a build project's settings (console).
- For the AWS CLI, run the `list-curated-environment-images` command:

```
aws codebuild list-curated-environment-images
```

- For the AWS SDKs, call the `ListCuratedEnvironmentImages` operation for your target programming language. For more information, see the AWS SDKs and tools reference.

EC2 compute images

AWS CodeBuild supports the following Docker images that are available for EC2 compute in CodeBuild.

> **Note**
>
> The base image of the Windows Server Core 2019 platform is only available in the following regions:
>
> - US East (N. Virginia)
> - US East (Ohio)
> - US West (Oregon)
> - Europe (Ireland)

Platform	Image identifier	Definition
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:4.0`	al/standard/4.0
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-standard:5.0`	al/standard/5.0
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:corretto8`	al/standard/corretto8
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:corretto11`	al/standard/corretto11
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-standard:2.0`	al/aarch64/standard/2.0

Platform	Image identifier	Definition
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-standard:3.0`	al/aarch64/standard/3.0
Ubuntu 20.04	`aws/codebuild/standard:5.0`	ubuntu/standard/5.0
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`	N/A
Windows Server Core 2019	`aws/codebuild/windows-base:2019-2.0`	N/A
Windows Server Core 2019	`aws/codebuild/windows-base:2019-3.0`	N/A
Windows Server Core 2022	`aws/codebuild/windows-base:2022-1.0`	N/A
macOS	`aws/codebuild/macos-arm-base:14`	N/A

Note

On November 22nd, 2024, the aliases for Linux-based standard runtime images were updated from `amazonlinux2` to `amazonlinux`. No manual update is required as the previous aliases are still valid.

Lambda compute images

AWS CodeBuild supports the following Docker images that are available for AWS Lambda compute in CodeBuild.

`aarch64` Architecture

Platform	Image identifier	Definition
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
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.21`	al-lambda/aarch64/go1.21
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.24`	al-lambda/aarch64/go1.24
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

Platform	Image identifier	Definition
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 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs22`	al-lambda/aarch64/nodejs22
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.11`	al-lambda/aarch64/python3.11
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.12`	al-lambda/aarch64/python3.12
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.13`	al-lambda/aarch64/python3.13

Platform	Image identifier	Definition
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.2	al-lambda/aarch64/ruby3.2
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.4	al-lambda/aarch64/ruby3.4

x86_64 Architecture

Platform	Image identifier	Definition
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 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.24	al-lambda/x86_64/go1.24
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lam	al-lambda/x86_64/corretto11

Platform	Image identifier	Definition
	bda-standard:corretto11	
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto17	al-lambda/x86_64/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 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs22	al-lambda/x86_64/nodejs22
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.11	al-lambda/x86_64/python3.11

Platform	Image identifier	Definition
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12`	al-lambda/x86_64/python3.12
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.13`	al-lambda/x86_64/python3.13
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.2`	al-lambda/x86_64/ruby3.2
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.4`	al-lambda/x86_64/ruby3.4

Deprecated CodeBuild images

A *deprecated image* is an image that is no longer cached or updated by CodeBuild. A deprecated image no longer receives minor version updates or patch version updates, and because they are no longer updated, using them may not be secure. If your CodeBuild project is configured to use an older image version, the provisioning process will download this docker image and use it to create the containerized runtime environment, which can increase the provisioning duration and overall build duration.

CodeBuild has deprecated the following Docker images. You can still use these images, but they won't be cached on the build host and will result in higher provisioning times.

Platform	Image identifier	Definition	Deprecation date
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:3.0`	al2/standard/3.0	May 9, 2023
Ubuntu 18.04	`aws/codebuild/standard:4.0`	ubuntu/standard/4.0	March 31, 2023
Amazon Linux 2	`aws/codebuild/amazonlinux2-aarch64-standard:1.0`	al2/aarch64/standard/1.0	March 31, 2023
Ubuntu 18.04	`aws/codebuild/standard:3.0`	ubuntu/standard/3.0	June 30, 2022
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:2.0`	al2/standard/2.0	June 30, 2022

Topics

Available runtimes

You can specify one or more runtimes in the `runtime-versions` section of your buildspec file. If your runtime is dependent upon another runtime, you can also specify its dependent runtime in the buildspec file. If you do not specify any runtimes in the buildspec file, CodeBuild chooses the default runtimes that are available in the image you use. If you specify one or more runtimes, CodeBuild uses only those runtimes. If a dependent runtime is not specified, CodeBuild attempts to choose the dependent runtime for you. For more information, see Specify runtime versions in the buildspec file.

Topics

Linux image runtimes

The following table contains the available runtimes and the standard Linux images that support them.

Ubuntu and Amazon Linux platform runtimes

Runtime name	Version	Images
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 stand dotnet6 Amazon Linux 2 AArch64 Lambda stan dotnet6 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

Runtime name	Version	Images
	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
	1.20	Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:7.0

Runtime name	Version	Images
	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 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:7.0
	1.23	Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:7.0
	1.24	Amazon Linux 2023 x86_64 Lambda st go1.24 Amazon Linux 2023 AArch64 Lambda go1.24

Runtime name	Version	Images
java	corretto8	Amazon Linux 2 x86_64 standard: corretto8 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2 AArch64 standard: 2.0 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:5.0 Ubuntu standard:7.0
	corretto11	Amazon Linux 2 x86_64 standard: corretto11 Amazon Linux 2 x86_64 Lambda stand corretto11 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2 AArch64 Lambda stan corretto11 Amazon Linux 2 AArch64 standard: 2.0 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:5.0 Ubuntu standard:7.0

Runtime name	Version	Images
	corretto17	Amazon Linux 2 x86_64 Lambda stand 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
	corretto21	Amazon Linux 2 x86_64 Lambda stand 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

Runtime name	Version	Images
	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
	18	Amazon Linux 2 x86_64 Lambda stand 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 stand 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

Runtime name	Version	Images
	22	Amazon Linux 2023 x86_64 Lambda st nodejs22 Amazon Linux 2023 AArch64 Lambda nodejs22 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:7.0
php	7.3	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
	8.1	Amazon Linux 2 x86_64 standard: 4.0 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:6.0

Runtime name	Version	Images
	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

Runtime name	Version	Images
	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 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:5.0 Ubuntu standard:7.0
	3.10	Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:6.0 Ubuntu standard:7.0
	3.11	Amazon Linux 2 x86_64 Lambda stand 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

Runtime name	Version	Images
	3.12	Amazon Linux 2 x86_64 Lambda stand 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
	3.13	Amazon Linux 2023 x86_64 Lambda st python3.13 Amazon Linux 2023 AArch64 Lambda python3.13 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

Runtime name	Version	Images
	3.1	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
	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 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:7.0

Runtime name	Version	Images
	3.4	Amazon Linux 2023 x86_64 Lambda st ruby3.4 Amazon Linux 2023 AArch64 Lambda ruby3.4 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard 3.0 Ubuntu standard:7.0

macOS image runtimes

Important

The CodeBuild curated images for Mac builds contain macOS and Xcode pre-installed. By using the Xcode software, you acknowledge, understand, and consent to the Xcode and Apple SDKs Agreement. If you do not accept the terms and conditions of the agreement, do not use the Xcode software. Instead, provide your own Amazon Machine Images (AMI). For more information, see How do I configure a reserved capacity macOS fleet?

The following table contains the available runtimes supported by macOS.

macOS platform runtimes

Runtime name	Version	Images	Additional notes
bash	3.2.57	macos-arm-base:14 macos-arm-base:15	
clang	15.0.0	macos-arm-base:14	

Runtime name	Version	Images	Additional notes
	16.0.0	macos-arm-base:15	
dotnet sdk	8.0.406	macos-arm-base:14 macos-arm-base:15	
gcc	11.5.0	macos-arm-base:14 macos-arm-base:15	Available by using the `gcc-11` alias
	12.4.0	macos-arm-base:14 macos-arm-base:15	Available by using the `gcc-12` alias
	13.3.0	macos-arm-base:14 macos-arm-base:15	Available by using the `gcc-13` alias
	14.2.0	macos-arm-base:14 macos-arm-base:15	Available by using the `gcc-14` alias
gnu	11.5.0	macos-arm-base:14 macos-arm-base:15	Available by using the `gfortran-11` alias
	12.4.0	macos-arm-base:14 macos-arm-base:15	Available by using the `gfortran-12` alias
	13.3.0	macos-arm-base:14 macos-arm-base:15	Available by using the `gfortran-13` alias
	14.2.0	macos-arm-base:14 macos-arm-base:15	Available by using the `gfortran-14` alias

Runtime name	Version	Images	Additional notes
golang	1.22.12	macos-arm-base:14 macos-arm-base:15	
	1.23.6	macos-arm-base:14 macos-arm-base:15	
	1.24.0	macos-arm-base:14 macos-arm-base:15	
java	Corretto8	macos-arm-base:14 macos-arm-base:15	
	Corretto11	macos-arm-base:14 macos-arm-base:15	
	Corretto17	macos-arm-base:14 macos-arm-base:15	
	Corretto21	macos-arm-base:14 macos-arm-base:15	
kotlin	2.1.10	macos-arm-base:14 macos-arm-base:15	
mono	6.12.0	macos-arm-base:14 macos-arm-base:15	
nodejs	18.20.7	macos-arm-base:14	

Runtime name	Version	Images	Additional notes
	20.18.3	macos-arm-base:14 macos-arm-base:15	
	22.14.0	macos-arm-base:14 macos-arm-base:15	
perl	5.34.1	macos-arm-base:14 macos-arm-base:15	
php	8.1.31	macos-arm-base:14	
	8.2.27	macos-arm-base:14 macos-arm-base:15	
	8.3.17	macos-arm-base:14 macos-arm-base:15	
	8.4.4	macos-arm-base:14 macos-arm-base:15	
python	3.9.21	macos-arm-base:14	
	3.10.16	macos-arm-base:14 macos-arm-base:15	
	3.11.11	macos-arm-base:14 macos-arm-base:15	
	3.12.9	macos-arm-base:14 macos-arm-base:15	

Runtime name	Version	Images	Additional notes
	3.13.2	macos-arm-base:14 macos-arm-base:15	
ruby	3.1.6	macos-arm-base:14	
	3.2.7	macos-arm-base:14 macos-arm-base:15	
	3.3.7	macos-arm-base:14 macos-arm-base:15	
	3.4.2	macos-arm-base:14 macos-arm-base:15	
rust	1.85.0	macos-arm-base:14 macos-arm-base:15	
swift	5.10.0.13	macos-arm-base:14	
	6.0.3.1.10	macos-arm-base:14	
Xcode	15.4	macos-arm-base:14	
	16.2	macos-arm-base:15	

Windows image runtimes

The base image of the Windows Server Core 2019 contains the following runtimes.

Windows platform runtimes

Runtime name	Windows Server Core 20 1.0 versions	Windows Server Core 20 2.0 versions	Windows Server Core 2019 sta 3.0 versions
dotnet	3.1	3.1	8.0

Runtime name	Windows Server Core 20 1.0 versions	Windows Server Core 20 2.0 versions	Windows Server Core 2019 sta 3.0 versions
	5.0	6.0 7.0	
dotnet sdk	3.1 5.0	3.1 6.0 7.0	8.0
golang	1.14	1.18	1.21 1.22 1.23
gradle	6.7	7.6	8.12
java	Corretto11	Corretto11 Corretto17	Corretto8 Corretto11 Corretto17 Corretto21
maven	3.6	3.8	3.9
nodejs	14.15	16.19	20.18 22.13
php	7.4	8.1	8.3 8.4
powershell	7.1	7.2	7.4

Runtime name	Windows Server Core 2C 1.0 versions	Windows Server Core 2C 2.0 versions	Windows Server Core 2019 sta 3.0 versions
python	3.8	3.10	3.10 3.11 3.12 3.13
ruby	2.7	3.1	3.2 3.3 3.4

Runtime versions

When you specify a runtime in the `runtime-versions` section of your buildspec file, you can specify a specific version, a specific major version and the latest minor version, or the latest version. The following table lists the available runtimes and how to specify them. Not all runtime versions are available on all images. Runtime version selection is also not supported for the custom images. For more information, see Available runtimes. If you'd like to install and use a custom runtime version instead of the pre-installed runtime versions, see Custom runtime versions.

Ubuntu and Amazon Linux 2 platform runtime versions

Runtime name	Version	Specific version	Specific major and latest minor version	Latest version
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`	

Runtime name	Version	Specific version	Specific major and latest minor version	Latest version
	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		
	1.23	golang: 1.23		
	1.24	golang: 1.24		
java	corretto8	java: corretto	java: corretto.x	java: latest
	corretto11	java: corretto1	java: corretto1.x	
	corretto17	java: corretto7	java: corretto7.x	
	corretto21	java: corretto1	java: corretto1.x	

Runtime name	Version	Specific version	Specific major and latest minor version	Latest version
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	
	22	nodejs: 22	nodejs: 22.x	
php	7.3	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	3.7	python: 3.7	python: 3.x	python: latest
	3.8	python: 3.8		
	3.9	python: 3.9		
	3.10	python: 3.10		
	3.11	python: 3.11		
	3.12	python: 3.12		

Runtime name	Version	Specific version	Specific major and latest minor version	Latest version
	3.13	`python: 3.13`		
ruby	2.6	`ruby: 2.6`	`ruby: 2.x`	`ruby: latest`
	2.7	`ruby: 2.7`		
	3.1	`ruby: 3.1`	`ruby: 3.x`	
	3.2	`ruby: 3.2`		
	3.3	`ruby: 3.3`		
	3.4	`ruby: 3.4`		

You can use a build specification to install other components (for example, the AWS CLI, Apache Maven, Apache Ant, Mocha, RSpec, or similar) during the `install` build phase. For more information, see Buildspec example.

Custom runtime versions

Instead of using the pre-installed runtime versions in CodeBuild-managed images, you can install and use custom versions of your choice. The following table lists the available custom runtimes and how to specify them.

> **Note**
>
> Custom runtime version selection is only supported for Ubuntu and Amazon Linux images.

Custom runtime versions

Runtime name	Syntax	Example
dotnet	*<major>*.*<minor>*.*<patch>*	5.0.408
golang	*<major>*.*<minor>*	1.19

Runtime name	Syntax	Example
	<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

Custom runtime buildspec example

Here is an example of a buildspec that specifies custom runtime versions.

```
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
```

Build environment compute modes and types

In CodeBuild, you can specify the compute and runtime environment image that CodeBuild uses to run your builds. *Compute* refers to the computing engine (the CPU, memory, and operating system) that is managed and maintained by CodeBuild. A *runtime environment image* is a container image that runs on top of your chosen compute platform, and includes extra tools that your build might need, such as the AWS CLI.

Topics

About compute

CodeBuild offers EC2 and AWS Lambda compute modes. EC2 offers optimized flexibility during your build and AWS Lambda offers optimized start-up speeds. AWS Lambda supports faster builds due to a lower start-up latency. AWS Lambda also automatically scales, so builds aren't waiting in queue to run. For more information, see Run builds on AWS Lambda compute.

With the EC2 compute mode, you can run your builds with on-demand or reserved capacity fleets. For on-demand fleets, you can select pre-defined compute types such as`BUILD_GENERAL1_SMALL` or `BUILD_GENERAL1_LARGE`. For more information, see About on-demand environment types. For reserved capacity fleets, you can select your compute configurations including vCPU, memory and disk space. After specifying the configurations, CodeBuild will choose a supported compute type that matches your requirements. For more information, see About reserved capacity environment types.

About reserved capacity environment types

AWS CodeBuild provides Linux x86, Arm, GPU, Windows and macOS environment types for reserved capacity fleets. The following table shows the available machine type, memory, vCPUs, and disk space sorted by region:

US East (N. Virginia)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM EC2	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`
ARM EC2	8	16 GiB	128 GB	GENERAL	`reserved.arm.8cpu.16gib`
Linux	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Linux	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Linux	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Linux	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Linux	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`
Linux	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	`reserved.gpu.4cpu.16gib.nvme`
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	`reserved.gpu.8cpu.32gib.nvme`
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	`reserved.gpu.16cpu.64gib.nvme`
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	`reserved.gpu.32cpu.128gib.nvme`
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	`reserved.gpu.48cpu.192gib.nvme`
Linux GPU	64	256 GiB	1885 GB (SSD)	NVME	`reserved.gpu.64cpu.256gib.nvme`
Linux GPU	96	384 GiB	3785 GB (SSD)	NVME	`reserved.gpu.96cpu.384gib.nvme`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
macOS	8	24 GiB	128 GB	GENERAL	`reserved.arm.m2.8cpu.24gib`
macOS	12	32 GiB	256 GB	GENERAL	`reserved.arm.m2.12cpu.32gib`
Windows	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Windows	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Windows	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Windows	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`
Windows	72	144 GiB	824 GB	GENERAL	`reserved.x86-64.72cpu.144gib`
Windows	96	192 GiB	824 GB	GENERAL	`reserved.x86-64.96cpu.192gib`
Windows EC2	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

US East (Ohio)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM EC2	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`
ARM EC2	8	16 GiB	128 GB	GENERAL	`reserved.arm.8cpu.16gib`
Linux	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Linux	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Linux	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Linux	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Linux	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`
Linux	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	`reserved.gpu.16cpu.64gib.nvme`
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	`reserved.gpu.32cpu.128gib.nvme`
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	`reserved.gpu.48cpu.192gib.nvme`
macOS	8	24 GiB	128 GB	GENERAL	`reserved.arm.m2.8cpu.24gib`
macOS	12	32 GiB	256 GB	GENERAL	`reserved.arm.m2.12cpu.32gib`
Windows	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Windows	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Windows	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Windows	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`
Windows	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`
Windows EC2	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

US West (Oregon)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM EC2	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`
ARM EC2	8	16 GiB	128 GB	GENERAL	`reserved.arm.8cpu.16gib`
Linux	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Linux	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Linux	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Linux	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Linux	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`
Linux	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	`reserved.gpu.4cpu.16gib.nvme`
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	`reserved.gpu.8cpu.32gib.nvme`
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	`reserved.gpu.16cpu.64gib.nvme`
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	`reserved.gpu.32cpu.128gib.nvme`
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	`reserved.gpu.48cpu.192gib.nvme`
Linux GPU	64	256 GiB	1885 GB (SSD)	NVME	`reserved.gpu.64cpu.256gib.nvme`
macOS	8	24 GiB	128 GB	GENERAL	`reserved.arm.m2.8cpu.24gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
macOS	12	32 GiB	256 GB	GENERAL	`reserved.arm.m2.12cpu.32gib`
Windows	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Windows	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Windows	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Windows	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`
Windows	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	72	144 GiB	824 GB	GENERAL	`reserved.x86-64.72cpu.144gib`
Windows	96	192 GiB	824 GB	GENERAL	`reserved.x86-64.96cpu.192gib`
Windows EC2	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

Asia Pacific (Tokyo)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`
ARM	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	`reserved.gpu.48cpu.192gib.nvme`
Windows	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Windows	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Windows	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Windows	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`
Windows	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	72	144 GiB	824 GB	GENERAL	`reserved.x86-64.72cpu.144gib`
Windows	96	192 GiB	824 GB	GENERAL	`reserved.x86-64.96cpu.192gib`
Windows EC2	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

Asia Pacific (Mumbai)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`
ARM	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	`reserved.gpu.16cpu.64gib.nvme`
Windows	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Windows	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Windows	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Windows	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`
Windows	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	72	144 GiB	824 GB	GENERAL	`reserved.x86-64.72cpu.144gib`
Windows	96	192 GiB	824 GB	GENERAL	`reserved.x86-64.96cpu.192gib`
Windows EC2	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

Asia Pacific (Singapore)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`
ARM	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

Asia Pacific (Sydney)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`
ARM	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`
ARM	8	16 GiB	128 GB	GENERAL	`reserved.arm.8cpu.16gib`
ARM	16	32 GiB	256 GB	GENERAL	`reserved.arm.16cpu.32gib`
ARM	32	64 GiB	256 GB	GENERAL	`reserved.arm.32cpu.64gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	48	96 GiB	512 GB	GENERAL	`reserved.arm.48cpu.96gib`
ARM	64	128 GiB	824 GB	GENERAL	`reserved.arm.64cpu.128gib`
ARM EC2	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`
ARM EC2	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`
ARM EC2	8	16 GiB	128 GB	GENERAL	`reserved.arm.8cpu.16gib`
Linux	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Linux	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Linux	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
macOS	8	24 GiB	128 GB	GENERAL	reserved.arm.m2.8cpu.24gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
macOS	12	32 GiB	256 GB	GENERAL	reserved.arm.m2.12cpu.32gib
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	72	144 GiB	824 GB	GENERAL	`reserved.x86-64.72cpu.144gib`
Windows	96	192 GiB	824 GB	GENERAL	`reserved.x86-64.96cpu.192gib`
Windows EC2	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

Europe (Frankfurt)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`
ARM	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32 GiB	256 GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64 GiB	256 GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128 GiB	824 GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	`reserved.gpu.16cpu.64gib.nvme`
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	`reserved.gpu.32cpu.128gib.nvme`
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	`reserved.gpu.48cpu.192gib.nvme`
macOS	8	24 GiB	128 GB	GENERAL	`reserved.arm.m2.8cpu.24gib`
Windows	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Windows	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

Europe (Ireland)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`
ARM	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`
ARM	8	16 GiB	128 GB	GENERAL	`reserved.arm.8cpu.16gib`
ARM	16	32 GiB	256 GB	GENERAL	`reserved.arm.16cpu.32gib`
ARM	32	64 GiB	256 GB	GENERAL	`reserved.arm.32cpu.64gib`
ARM	48	96 GiB	512 GB	GENERAL	`reserved.arm.48cpu.96gib`
ARM	64	128 GiB	824 GB	GENERAL	`reserved.arm.64cpu.128gib`
ARM EC2	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM EC2	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`
ARM EC2	8	16 GiB	128 GB	GENERAL	`reserved.arm.8cpu.16gib`
Linux	2	4 GiB	64 GB	GENERAL	`reserved.x86-64.2cpu.4gib`
Linux	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Linux	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`
Linux	16	32 GiB	256 GB	GENERAL	`reserved.x86-64.16cpu.32gib`
Linux	36	72 GiB	256 GB	GENERAL	`reserved.x86-64.36cpu.72gib`
Linux	48	96 GiB	512 GB	GENERAL	`reserved.x86-64.48cpu.96gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96 GiB	824 GB (SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux GPU	4	16 GiB	235 GB (SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32 GiB	435 GB (SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64 GiB	585 GB (SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128 GiB	885 GB (SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192 GiB	3785 GB (SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192 GiB	824 GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

South America (São Paulo)

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	2	4 GiB	64 GB	GENERAL	`reserved.arm.2cpu.4gib`
ARM	4	8 GiB	128 GB	GENERAL	`reserved.arm.4cpu.8gib`
ARM	8	16 GiB	128 GB	GENERAL	`reserved.arm.8cpu.16gib`
ARM	16	32 GiB	256 GB	GENERAL	`reserved.arm.16cpu.32gib`
ARM	32	64 GiB	256 GB	GENERAL	`reserved.arm.32cpu.64gib`

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
ARM	48	96 GiB	512 GB	GENERAL	reserved.arm.48cpu.96gib
ARM EC2	2	4 GiB	64 GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8 GiB	128 GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16 GiB	128 GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Linux	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	72	144 GiB	824 GB (SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows	2	4 GiB	64 GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8 GiB	128 GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16 GiB	128 GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32 GiB	256 GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	36	72 GiB	256 GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96 GiB	512 GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	72	144 GiB	824 GB	GENERAL	reserved.x86-64.72cpu.144gib

Environment type	vCPUs	Memory	Disk space	Machine type	Compute instance type
Windows EC2	4	8 GiB	128 GB	GENERAL	`reserved.x86-64.4cpu.8gib`
Windows EC2	8	16 GiB	128 GB	GENERAL	`reserved.x86-64.8cpu.16gib`

For more information on the pricing identifier, see https://aws.amazon.com/codebuild/pricing/.

To choose a compute type:

- In the CodeBuild console, in the **Compute fleet configuration** page, choose one of the options from **vCPUs**, **Memory**, and **Disk**. For more information, see Create a reserved capacity fleet.
- For the AWS CLI, run the `create-fleet` or `update-fleet` command, specifying the value of `computeType` to `ATTRIBUTE_BASED_COMPUTE`. For more information, see create-fleet or update-fleet.
- For the AWS SDKs, call the equivalent of the `CreateFleet` or `UpdateFleet` operation for your target programming language, specifying the value of `computeType` to `ATTRIBUTE_BASED_COMPUTE`. For more information, see the AWS SDKs and tools reference.

> **Note**
>
> For the AWS CLI and AWS SDKs, you can still use `computeType` inputs such as `BUILD_GENERAL1_SMALL`, to choose the compute types instead of `ATTRIBUTE_BASED_COMPUTE`. For more information, see About on-demand environment types.

Supported instance families

AWS CodeBuild supports the following instances for reserved capacity fleets.:

- **General purpose:** M5 | M5a | M5ad | M5d | M5dn | M5n | M5zn | M6a | M6g | M6gd | M6i | M6id | M6idn | M6in | M7a | M7g | M7gd | M7i | M7i-flex | M8g | T3 | T3a | T4g
- **Compute optimized:** C5 | C5a | C5ad | C5d | C5n | C6a | C6g | C6gd | C6gn | C6i | C6id | C6in | C7a | C7g | C7gd | C7gn | C7i | C7i-flex | C8g
- **Memory optimized:** R5 | R5a | R5ad | R5b | R5d | R5dn | R5n | R6a | R6g | R6gd | R6i | R6idn | R6in | R6id | R7a | R7g | R7gd | R7i | R7iz | R8g | U-3tb1 | U-6tb1 | U-9tb1 | U-12tb1 | U-18tb1 | U-24tb1 | U7i-6tb | U7i-8tb | U7i-12tb | U7in-16tb | U7in-24tb | U7in-32tb | X1 | X1e | X2gd | X2idn | X2iedn | X2iezn | X8g | z1d
- **Storage optimized:** D3 | D3en | I3 | I3en | I4g | I4i | I7ie | I8g | Im4gn | Is4gen
- **Accelerated computing:** DL1 | DL2q | F1 | F2 | G4ad | G4dn | G5 | G5g | G6 | G6e | Gr6 | Inf1 | Inf2 | P3 | P3dn | P4d | P5 | P5e | P5en | Trn1 | Trn1n | Trn2 | VT1
- **High-performance computing:** Hpc6a | Hpc6id | Hpc7a | Hpc7g
- **Previous generation:** A1

To create a reserved capacity fleet with a specific instance type:

- In the CodeBuild console, in the **Compute fleet configuration** page, navigate to the **Capacity configuration** section. In **Compute selection mode**, select **Manual input** and in **Compute instance type** choose one of the instance types from the drop-down menu. For more information, see Create a reserved capacity fleet.
- For the AWS CLI, run the `create-fleet` or `update-fleet` command, specifying the value of `computeType` to `CUSTOM_INSTANCE_TYPE` and the `ComputeConfiguration` `instanceType` to the specified instance type. For more information, see create-fleet or update-fleet.
- For the AWS SDKs, call the equivalent of the `CreateFleet` or `UpdateFleet` operation for your target programming language, specifying the value of `computeType` to `CUSTOM_INSTANCE_TYPE` and the **ComputeConfiguration** `instanceType` to the specified instance type. For more information, see the AWS SDKs and tools reference.

About on-demand environment types

AWS CodeBuild provides build environments with the following available memory, vCPUs, and disk space for EC2 compute mode:

Compute type	Environment computeType value	Environment type value	Memory	vCPUs	Disk space
ARM Small [1]	BUILD_GENERAL1_SMALL	ARM_CONTAINER ARM_EC2	4 GiB	2	64 GB
ARM Medium [1]	BUILD_GENERAL1_MEDIUM	ARM_CONTAINER ARM_EC2	8 GiB	4	128 GB
ARM Large [1]	BUILD_GENERAL1_LARGE	ARM_CONTAINER ARM_EC2	16 GiB	8	128 GB
ARM XLarge [1]	BUILD_GENERAL1_XLARGE	ARM_CONTAINER	64 GiB	32	256 GB
ARM 2XLarge [1]	BUILD_GENERAL1_2XLARGE	ARM_CONTAINER	96 GiB	48	824 GB
Linux Small [1]	BUILD_GENERAL1_SMALL	LINUX_CONTAINER LINUX_EC2	4 GiB	2	64 GB
Linux Medium [1]	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER LINUX_EC2	8 GiB	4	128 GB
Linux Large [1]	BUILD_GENERAL1_LARGE	LINUX_CONTAINER	16 GiB	8	128 GB

Compute type	Environment computeType value	Environment type value	Memory	vCPUs	Disk space
		LINUX_EC2			
Linux XLarge [1]	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	72 GiB	36	256 GB
Linux 2XLarge	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	144 GiB	72	824 GB (SSD)
Linux GPU Sma	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16 GiB	4	235 GB (SSD)
Linux GPU Larg	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255 GiB	32	50 GB
Windows Mediu [1]	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER WINDOWS_SERVER_2022_CONTAINER WINDOWS_EC2	8 GiB	4	128 GB

Compute type	Environment computeType value	Environment type value	Memory	vCPUs	Disk space
Windows Large [1]	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER WINDOWS_SERVER_2022_CONTAINER WINDOWS_EC2	16 GiB	8	128 GB
Windows XLarg [1]	BUILD_GENERAL1_XLARGE	WINDOWS_SERVER_2022_CONTAINER	72 GiB	36	256 GB
Windows 2XLar [1]	BUILD_GENERAL1_2XLARGE	WINDOWS_SERVER_2022_CONTAINER	144 GiB	72	824 GB

[1] The latest version of this image type is cached. If you specify a more specific version, then CodeBuild provisions that version instead of the cached version. This can result in longer build times. For example, to benefit from caching, specify `aws/codebuild/amazonlinux-x86_64-standard:5.0` instead of a more granular version, such as `aws/codebuild/amazonlinux-x86_64-standard:5.0-1.0.0`.

AWS CodeBuild provides build environments with the following available memory and disk space for AWS Lambda compute mode:

Compute type	Environment computeType value	Environment type value	Memory	Disk space
ARM Lambda 1GB	BUILD_LAMBDA_1GB	ARM_LAMBDA_CONTAINER	1 GiB	10 GB
ARM Lambda 2GB	BUILD_LAMBDA_2GB	ARM_LAMBDA_CONTAINER	2 GiB	10 GB
ARM Lambda 4GB	BUILD_LAMBDA_4GB	ARM_LAMBDA_CONTAINER	4 GiB	10 GB
ARM Lambda 8GB	BUILD_LAMBDA_8GB	ARM_LAMBDA_CONTAINER	8 GiB	10 GB
ARM Lambda 10GI	BUILD_LAMBDA_10GB	ARM_LAMBDA_CONTAINER	10 GiB	10 GB
Linux Lambda 1GB	BUILD_LAMBDA_1GB	LINUX_LAMBDA_CONTAINER	1 GiB	10 GB
Linux Lambda 2GB	BUILD_LAMBDA_2GB	LINUX_LAMBDA_CONTAINER	2 GiB	10 GB
Linux Lambda 4GB	BUILD_LAMBDA_4GB	LINUX_LAMBDA_CONTAINER	4 GiB	10 GB

Compute type	Environment computeType value	Environment type value	Memory	Disk space
Linux Lambda 8GB	BUILD_LAMBDA_8GB	LINUX_LAMBDA_CONTAINER	8 GiB	10 GB
Linux Lambda 10G	BUILD_LAMBDA_10GB	LINUX_LAMBDA_CONTAINER	10 GiB	10 GB

When using other environment types, it is recommended that you use a cached image to reduce build times.

The disk space listed for each build environment is available only in the directory specified by the CODEBUILD_SRC_DIR environment variable.

To choose a compute type:

- In the CodeBuild console, in the **Create build project** wizard or **Edit Build Project** page, in **Environment** expand **Additional configuration**, and then choose one of the options from **Compute type**. For more information, see Create a build project (console) or Change a build project's settings (console).
- For the AWS CLI, run the create-project or update-project command, specifying the computeType value of the environment object. For more information, see Create a build project (AWS CLI) or Change a build project's settings (AWS CLI).
- For the AWS SDKs, call the equivalent of the CreateProject or UpdateProject operation for your target programming language, specifying the equivalent of computeType value of the environment object. For more information, see the AWS SDKs and tools reference.

Some environment and compute types have Region availability limitations:

- The compute type Linux GPU Small (LINUX_GPU_CONTAINER) is only available in these Regions:
  - US East (N. Virginia)
  - US West (Oregon)

- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)

- The compute type Linux GPU Large (`LINUX_GPU_CONTAINER`) is only available in these Regions:
  - US East (Ohio)
  - US East (N. Virginia)
  - US West (Oregon)
  - Asia Pacific (Seoul)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Canada (Central)
  - China (Beijing)
  - China (Ningxia)
  - Europe (Frankfurt)
  - Europe (Ireland)
  - Europe (London)
- The compute type `BUILD_GENERAL1_2XLARGE` is only available in these Regions:
  - US East (Ohio)
  - US East (N. Virginia)
  - US West (N. California)
  - US West (Oregon)
  - Asia Pacific (Hyderabad)
  - Asia Pacific (Hong Kong)
  - Asia Pacific (Jakarta)
  - Asia Pacific (Melbourne)
  - Asia Pacific (Mumbai)
  - Asia Pacific (Seoul)
  - Asia Pacific (Singapore)

- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- China (Beijing)
- China (Ningxia)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Spain)
- Europe (Stockholm)
- Europe (Zurich)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- Middle East (UAE)
- South America (São Paulo)

- The environment type `ARM_CONTAINER` is only available in these Regions:
  - US East (Ohio)
  - US East (N. Virginia)
  - US West (N. California)
  - US West (Oregon)
  - Asia Pacific (Hong Kong)
  - Asia Pacific (Jakarta)
  - Asia Pacific (Hyderabad)
  - Asia Pacific (Mumbai)
  - Asia Pacific (Osaka)
  - Asia Pacific (Seoul)
  - Asia Pacific (Singapore)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)

- Canada (Central)
- China (Beijing)
- China (Ningxia)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Milan)
- Europe (Paris)
- Europe (Spain)
- Europe (Stockholm)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- Middle East (UAE)
- South America (São Paulo)

- The environment type `WINDOWS_SERVER_2022_CONTAINER` is only available in these Regions:
  - US East (Ohio)
  - US East (N. Virginia)
  - US West (Oregon)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Europe (Frankfurt)
  - Europe (Ireland)
  - South America (São Paulo)
- The environment type `LINUX_EC2` (`BUILD_GENERAL1_SMALL`, `BUILD_GENERAL1_MEDIUM`, `BUILD_GENERAL1_LARGE`) is only available in these Regions:
  - US East (Ohio)
  - US East (N. Virginia)
  - US West (N. California)
  - US West (Oregon)

- Africa (Cape Town)

- Asia Pacific (Hong Kong)
- Asia Pacific (Jakarta)
- Asia Pacific (Melbourne)
- Europe (Zurich)
- Asia Pacific (Hyderabad)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- China (Beijing)
- China (Ningxia)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Milan)
- Europe (Paris)
- Europe (Spain)
- Europe (Stockholm)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- Middle East (UAE)
- South America (São Paulo)
- AWS GovCloud (US-West)
- AWS GovCloud (US-East)

- The environment type `ARM_EC2` (`BUILD_GENERAL1_SMALL`, `BUILD_GENERAL1_MEDIUM`, `BUILD_GENERAL1_LARGE`) is only available in these Regions:
  - US East (Ohio)

- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Hong Kong)
- Asia Pacific (Jakarta)
- Europe (Zurich)
- Asia Pacific (Hyderabad)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- China (Beijing)
- China (Ningxia)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Milan)
- Europe (Paris)
- Europe (Spain)
- Europe (Stockholm)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- South America (São Paulo)
- AWS GovCloud (US-West)
- AWS GovCloud (US-East)

- The environment type `WINDOWS_EC2` (`BUILD_GENERAL1_MEDIUM`, `BUILD_GENERAL1_LARGE`) is only available in these Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (Oregon)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Europe (Frankfurt)
- Europe (Ireland)
- South America (São Paulo)

- The compute mode AWS Lambda (`ARM_LAMBDA_CONTAINER` and `LINUX_LAMBDA_CONTAINER`) is only available in these Regions:
  - US East (N. Virginia)
  - US East (Ohio)
  - US West (Oregon)
  - Asia Pacific (Mumbai)
  - Asia Pacific (Singapore)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Europe (Frankfurt)
  - Europe (Ireland)
  - South America (São Paulo)
- The compute mode `MAC_ARM` is only available in these Regions:
  - US East (N. Virginia)
  - US East (Ohio)
  - US West (Oregon)
  - Asia Pacific (Sydney)
  - Europe (Frankfurt)

For the compute type `BUILD_GENERAL1_2XLARGE`, Docker images up to 100 GB uncompressed are supported.

> **Note**
>
> For custom build environment images, CodeBuild supports Docker images up to 50 GB uncompressed in Linux and Windows, regardless of the compute type. To check your build image's size, use Docker to run the `docker images` *`REPOSITORY`*`:`*`TAG`* command.

You can use Amazon EFS to access more space in your build container. For more information, see Amazon Elastic File System sample for AWS CodeBuild. If you want to manipulate container disk space during a build, then the build must run in privileged mode.

> **Note**
>
> By default, Docker daemon is enabled for non-VPC builds. If you would like to use Docker containers for VPC builds, see Runtime Privilege and Linux Capabilities on the Docker Docs website and enable privileged mode. Also, Windows does not support privileged mode.

Shells and commands in build environments

You provide a set of commands for AWS CodeBuild to run in a build environment during the lifecycle of a build (for example, installing build dependencies and testing and compiling your source code). There are several ways to specify these commands:

- Create a build specification file and include it with your source code. In this file, specify the commands you want to run in each phase of the build lifecycle. For more information, see the Build specification reference for CodeBuild.
- Use the CodeBuild console to create a build project. In **Insert build commands**, for **Build commands**, enter the commands you want to run in the `build` phase. For more information, see Create a build project (console).
- Use the CodeBuild console to change the settings of a build project. In **Insert build commands**, for **Build commands**, enter the commands you want to run in the `build` phase. For more information, see Change a build project's settings (console).
- Use the AWS CLI or AWS SDKs to create a build project or change the settings of a build project. Reference the source code that contains a buildspec file with your commands, or specify a single string that includes the contents of an equivalent buildspec file. For more information, see Create a build project or Change build project settings.

- Use the AWS CLI or AWS SDKs to start a build, specifying a buildspec file or a single string that includes the contents of an equivalent buildspec file. For more information, see the description for the `buildspecOverride` value in Run builds manually.

You can specify any Shell Command Language (sh) command. In buildspec version 0.1, CodeBuild runs each Shell command in a separate instance in the build environment. This means that each command runs in isolation from all other commands. Therefore, by default, you cannot run a single command that relies on the state of any previous commands (for example, changing directories or setting environment variables). To get around this limitation, we recommend that you use version 0.2, which solves this issue. If you must use version 0.1, we recommend the following approaches:

- Include a shell script in your source code that contains the commands you want to run in a single instance of the default shell. For example, you could include a file named `my-script.sh` in your source code that contains commands such as `cd MyDir; mkdir -p mySubDir; cd mySubDir; pwd;`. Then, in your buildspec file, specify the command `./my-script.sh`.
- In your buildspec file or on the **Build commands** setting for the `build` phase only, enter a single command that includes all of the commands you want to run in a single instance of the default shell (for example, `cd MyDir && mkdir -p mySubDir && cd mySubDir && pwd`).

If CodeBuild encounters an error, the error might be more difficult to troubleshoot compared to running a single command in its own instance of the default shell.

Commands that are run in a Windows Server Core image use the PowerShell shell.

Environment variables in build environments

AWS CodeBuild provides several environment variables that you can use in your build commands:

AWS_DEFAULT_REGION

The AWS Region where the build is running (for example, `us-east-1`). This environment variable is used primarily by the AWS CLI.

AWS_REGION

The AWS Region where the build is running (for example, `us-east-1`). This environment variable is used primarily by the AWS SDKs.

CODEBUILD_BATCH_BUILD_IDENTIFIER

The identifier of the build in a batch build. This is specified in the batch buildspec. For more information, see the section called “Batch buildspec reference”.

CODEBUILD_BUILD_ARN

The Amazon Resource Name (ARN) of the build (for example, `arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`).

CODEBUILD_BUILD_ID

The CodeBuild ID of the build (for example, `codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`).

CODEBUILD_BUILD_IMAGE

The CodeBuild build image identifier (for example, `aws/codebuild/standard:2.0`).

CODEBUILD_BUILD_NUMBER

The current build number for the project.

CODEBUILD_BUILD_SUCCEEDING

Whether the current build is succeeding. Set to `0` if the build is failing, or `1` if the build is succeeding.

CODEBUILD_INITIATOR

The entity that started the build. If CodePipeline started the build, this is the pipeline's name (for example, `codepipeline/my-demo-pipeline`). If an user started the build, this is the user's name (for example, `MyUserName`). If the Jenkins plugin for CodeBuild started the build, this is the string `CodeBuild-Jenkins-Plugin`.

CODEBUILD_KMS_KEY_ID

The identifier of the AWS KMS key that CodeBuild is using to encrypt the build output artifact (for example, `arn:aws:kms:region-ID:account-ID:key/key-ID` or `alias/key-alias`).

CODEBUILD_PROJECT_ARN

The Amazon Resource Name (ARN) of the project (for example, `arn:aws:codebuild:region-ID:account-ID:project/project-name`).

CODEBUILD_PUBLIC_BUILD_URL

The URL of the build results for this build on the public builds website. This variable is only set if the build project has public builds enabled. For more information, see Get public build project URLs.

CODEBUILD_RESOLVED_SOURCE_VERSION

The version identifier of a build's source code. The contents depends on the source code repository:

CodeCommit, GitHub, GitHub Enterprise Server, and Bitbucket

This variable contains the commit ID.

CodePipeline

This variable contains the source revision provided by CodePipeline.

If CodePipeline is not able to resolve the source revision, such as when the source is an Amazon S3 bucket that does not have versioning enabled, this environment variable is not set.

Amazon S3

This variable is not set.

When applicable, the `CODEBUILD_RESOLVED_SOURCE_VERSION` variable is only available after the `DOWNLOAD_SOURCE` phase.

CODEBUILD_SOURCE_REPO_URL

The URL to the input artifact or source code repository. For Amazon S3, this is `s3://` followed by the bucket name and path to the input artifact. For CodeCommit and GitHub, this is the repository's clone URL. If a build originates from CodePipeline, this environment variable may be empty.

For secondary sources, the environment variable for the secondary source repository URL is `CODEBUILD_SOURCE_REPO_URL_`*`<sourceIdentifier>`*, where *`<sourceIdentifier>`* is the source identifier you create.

CODEBUILD_SOURCE_VERSION

The value's format depends on the source repository.

- For Amazon S3, it is the version ID associated with the input artifact.
- For CodeCommit, it is the commit ID or branch name associated with the version of the source code to be built.
- For GitHub, GitHub Enterprise Server, and Bitbucket it is the commit ID, branch name, or tag name associated with the version of the source code to be built.

> **Note**
>
> For a GitHub or GitHub Enterprise Server build that is triggered by a webhook pull request event, it is `pr/pull-request-number`.

For secondary sources, the environment variable for the secondary source version is `CODEBUILD_SOURCE_VERSION_<sourceIdentifier>`, where `<sourceIdentifier>` is the source identifier you create. For more information, see Multiple input sources and output artifacts sample.

CODEBUILD_SRC_DIR

The directory path that CodeBuild uses for the build (for example, `/tmp/src123456789/src`).

For secondary sources, the environment variable for the secondary source directory path is `CODEBUILD_SRC_DIR_<sourceIdentifier>`, where `<sourceIdentifier>` is the source identifier you create. For more information, see Multiple input sources and output artifacts sample.

CODEBUILD_START_TIME

The start time of the build specified as a Unix timestamp in milliseconds.

CODEBUILD_WEBHOOK_ACTOR_ACCOUNT_ID

The account ID of the user that triggered the webhook event.

CODEBUILD_WEBHOOK_BASE_REF

The base reference name of the webhook event that triggers the current build. For a pull request, this is the branch reference.

CODEBUILD_WEBHOOK_EVENT

The webhook event that triggers the current build.

CODEBUILD_WEBHOOK_MERGE_COMMIT

The identifier of the merge commit used for the build. This variable is set when a Bitbucket pull request is merged with the squash strategy and the pull request branch is closed. In this case, the original pull request commit no longer exists, so this environment variable contains the identifier of the squashed merge commit.

CODEBUILD_WEBHOOK_PREV_COMMIT

The ID of the most recent commit before the webhook push event that triggers the current build.

CODEBUILD_WEBHOOK_HEAD_REF

The head reference name of the webhook event that triggers the current build. It can be a branch reference or a tag reference.

CODEBUILD_WEBHOOK_TRIGGER

Shows the webhook event that triggered the build. This variable is available only for builds triggered by a webhook. The value is parsed from the payload sent to CodeBuild by GitHub, GitHub Enterprise Server, or Bitbucket. The value's format depends on what type of event triggered the build.

- For builds triggered by a pull request, it is `pr/`*`pull-request-number`*.
- For builds triggered by creating a new branch or pushing a commit to a branch, it is `branch/`*`branch-name`*.
- For builds triggered by a pushing a tag to a repository, it is `tag/`*`tag-name`*.

HOME

This environment variable is always set to `/root`.

AWS CodeBuild also supports a set of environment variables for self-hosted runner builds. To learn more about CodeBuild self-hosted runner, see Tutorial: Configure a CodeBuild-hosted GitHub Actions runner.

CODEBUILD_RUNNER_OWNER

The owner of the repository that triggers the self-hosted runner build.

CODEBUILD_RUNNER_REPO

The name of the repository that triggers the self-hosted runner build.

CODEBUILD_RUNNER_REPO_DOMAIN

The domain of the repository that triggers the self-hosted runner build. Only specified GitHub Enterprise builds.

CODEBUILD_WEBHOOK_LABEL

The label used to configure build overrides and the self-hosted runner during the build.

CODEBUILD_WEBHOOK_RUN_ID

The run ID of the workflow associated with the build.

CODEBUILD_WEBHOOK_JOB_ID

The job ID of the job associated with the build.

CODEBUILD_WEBHOOK_WORKFLOW_NAME

The name of the workflow associated with the build if it exists in the webhook request payload.

CODEBUILD_RUNNER_WITH_BUILDSPEC

If a buildspec override is configured in the self-hosted runner request labels, this is set to `true`.

You can also provide build environments with your own environment variables. For more information, see the following topics:

- Use CodeBuild with CodePipeline
- Create a build project
- Change build project settings
- Run builds manually
- Buildspec reference

To list all of the available environment variables in a build environment, you can run the `printenv` command (for Linux-based build environment) or `"Get-ChildItem Env:"` (for Windows-based build environments) during a build. Except for those previously listed, environment variables that start with `CODEBUILD_` are for CodeBuild internal use. They should not be used in your build commands.

Important

We strongly discourage the use of environment variables to store sensitive values, especially AWS access key IDs. Environment variables can be displayed in plain text using tools such as the CodeBuild console and the AWS CLI.
We recommend you store sensitive values in the Amazon EC2 Systems Manager Parameter Store and then retrieve them from your buildspec. To store sensitive values, see Systems Manager Parameter Store and Walkthrough: Create and test a String parameter (console) in the *Amazon EC2 Systems Manager User Guide*. To retrieve them, see the `parameter-store` mapping in Buildspec syntax.

Background tasks in build environments

You can run background tasks in build environments. To do this, in your buildspec, use the nohup command to run a command as a task in the background, even if the build process exits the shell. Use the **disown** command to forcibly stop a running background task.

Examples:

- Start a background process and wait for it to complete later:

```
|
nohup sleep 30 & echo $! > pidfile
…
wait $(cat pidfile)
```

- Start a background process and do not wait for it to ever complete:

```
|
nohup sleep 30 & disown $!
```

- Start a background process and kill it later:

```
|
nohup sleep 30 & echo $! > pidfile
…
kill $(cat pidfile)
```

Build projects

A *build project* includes information about how to run a build, including where to get the source code, which build environment to use, which build commands to run, and where to store the build output.

You can perform these tasks when working with build projects:

Topics

Create a build project in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to create a build project.

Topics

- Create a build project (AWS CloudFormation)

Prerequisites

Before creating a build project, answer the questions in Plan a build.

Create a build project (console)

Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.

If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.

Choose **Create build project**.

Fill in the following sections. Once complete, choose **Create build project** at the bottom of the page.

Sections:

- Project configuration
- Source
- Environment
- Buildspec
- Batch configuration
- Artifacts
- Logs

Project configuration

Project name

Enter a name for this build project. Build project names must be unique across each AWS account.

Description

Enter an optional description of the build project to help other users understand what this project is used for.

Build badge

(Optional) Select **Enable build badge** to make your project's build status visible and embeddable. For more information, see Build badges sample.

> **Note**
>
> Build badge does not apply if your source provider is Amazon S3.

Enable concurrent build limit

(Optional) If you want to limit the number of concurrent builds for this project, perform the following steps:

1. Select **Restrict number of concurrent builds this project can start**.
2. In **Concurrent build limit**, enter the maximum number of concurrent builds that are allowed for this project. This limit cannot be greater than the concurrent build limit set for the account. If you try to enter a number greater than the account limit, an error message is displayed.

New builds are only started if the current number of builds is less than or equal to this limit. If the current build count meets this limit, new builds are throttled and are not run.

Additional information

(Optional) For **Tags**, enter the name and value of any tags that you want supporting AWS services to use. Use **Add row** to add a tag. You can add up to 50 tags.

Source

Source provider

Choose the source code provider type. Use the following lists to make selections appropriate for your source provider:

> **Note**
>
> CodeBuild does not support Bitbucket Server.

Amazon S3

Bucket

Choose the name of the input bucket that contains the source code.

S3 object key or S3 folder

Enter the name of the ZIP file or the path to the folder that contains the source code. Enter a forward slash (/) to download everything in the S3 bucket.

Source version

Enter the version ID of the object that represents the build of your input file. For more information, seeSource version sample with AWS CodeBuild.

CodeCommit

Repository

Choose the repository you want to use.

Reference type

Choose **Branch**, **Git tag**, or **Commit ID** to specify the version of your source code. For more information, see Source version sample with AWS CodeBuild.

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Git submodules

Select **Use Git submodules** if you want to include Git submodules in your repository.

Bitbucket

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

Choose **CodeConnections**, **OAuth**, **App password**, or **Personal access token** to connect to CodeBuild.

Connection

Select a Bitbucket connection or a Secrets Manager secret to connect through your specified connection type.

Repository

Choose **Repository in my Bitbucket account** or **Public repository** and enter the repository URL.

Source version

Enter a branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Git submodules

Select **Use Git submodules** if you want to include Git submodules in your repository.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

For **Status context**, enter the value to be used for the `name` parameter in the Bitbucket commit status. For more information, see build in the Bitbucket API documentation.

For **Target URL**, enter the value to be used for the `url` parameter in the Bitbucket commit status. For more information, see build in the Bitbucket API documentation.

The status of a build triggered by a webhook is always reported to the source provider. To have the status of a build that is started from the console or an API call reported to the source provider, you must select this setting.

If your project's builds are triggered by a webhook, you must push a new commit to the repo for a change to this setting to take effect.

In **Primary source webhook events**, select **Rebuild every time a code change is pushed to this repository** if you want CodeBuild to build the source code every time a code change is pushed to this repository. For more information about webhooks and filter groups, see Bitbucket webhook events.

GitHub

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

Choose **GitHub App**, **OAuth**, or **Personal access token** to connect to CodeBuild.

Connection

Select a GitHub connection or a Secrets Manager secret to connect through your specified connection type.

Repository

Choose **Repository in my GitHub account**, **Public repository**, or **GitHub scoped webhook** and enter the repository URL.

Source version

Enter a branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Git submodules

Select **Use Git submodules** if you want to include Git submodules in your repository.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

For **Status context**, enter the value to be used for the `context` parameter in the GitHub commit status. For more information, see Create a commit status in the GitHub developer guide.

For **Target URL**, enter the value to be used for the `target_url` parameter in the GitHub commit status. For more information, see Create a commit status in the GitHub developer guide.

The status of a build triggered by a webhook is always reported to the source provider. To have the status of a build that is started from the console or an API call reported to the source provider, you must select this setting.

If your project's builds are triggered by a webhook, you must push a new commit to the repo for a change to this setting to take effect.

In **Primary source webhook events**, select **Rebuild every time a code change is pushed to this repository** if you want CodeBuild to build the source code every time a code change is pushed to this repository. For more information about webhooks and filter groups, see GitHub webhook events.

GitHub Enterprise Server

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

Choose **CodeConnections** or **Personal access token** to connect to CodeBuild.

Connection

Select a GitHub Enterprise connection or a Secrets Manager secret to connect through your specified connection type.

Repository

Choose **Repository in my GitHub Enterprise account** or **GitHub Enterprise scoped webhook** and enter the repository URL.

Source version

Enter a pull request, branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild.

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Git submodules

Select **Use Git submodules** if you want to include Git submodules in your repository.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

For **Status context**, enter the value to be used for the `context` parameter in the GitHub commit status. For more information, see Create a commit status in the GitHub developer guide.

For **Target URL**, enter the value to be used for the `target_url` parameter in the GitHub commit status. For more information, see Create a commit status in the GitHub developer guide.

The status of a build triggered by a webhook is always reported to the source provider. To have the status of a build that is started from the console or an API call reported to the source provider, you must select this setting.

If your project's builds are triggered by a webhook, you must push a new commit to the repo for a change to this setting to take effect.

Insecure SSL

Select **Enable insecure SSL** to ignore SSL warnings while connecting to your GitHub Enterprise project repository.

In **Primary source webhook events**, select **Rebuild every time a code change is pushed to this repository** if you want CodeBuild to build the source code every time a code change is pushed to this repository. For more information about webhooks and filter groups, see GitHub webhook events.

GitLab

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

CodeConnections is used to connect GitLab to CodeBuild.

Connection

Select a GitLab connection to connect through CodeConnections.

Repository

Choose the repository you want to use.

Source version

Enter a pull request ID, branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild.

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

GitLab Self Managed

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

CodeConnections is used to connect GitLab Self Managed to CodeBuild.

Connection

Select a GitLab Self Managed connection to connect through CodeConnections.

Repository

Choose the repository you want to use.

Source version

Enter a pull request ID, branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild.

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

Environment

Provisioning model

Do one of the following:

- To use on-demand fleets managed by AWS CodeBuild, choose **On-demand**. With on-demand fleets, CodeBuild provides compute for your builds. The machines are destroyed when the build finishes. On-demand fleets are fully managed, and includes automatic scaling capabilities to handle spikes in demand.
- To use reserved capacity fleets managed by AWS CodeBuild, choose **Reserved capacity**, and then select a **Fleet name**. With reserved capacity fleets, you configure a set of dedicated instances for your build environment. These machines remain idle, ready to process builds or tests immediately and reduces build durations. With reserved capacity fleets, your machines are always running and will continue to incur costs as long they're provisioned.

For information, see Run builds on reserved capacity fleets.

Environment image

Do one of the following:

- To use a Docker image managed by AWS CodeBuild, choose **Managed image**, and then make selections from **Operating system**, **Runtime(s)**, **Image**, and **Image version**. Make a selection from **Environment type** if it is available.
- To use another Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. If you choose **Other registry**, for **External registry URL**, enter the name and tag of the Docker image in Docker Hub, using the format *`docker repository`*/*`docker image name`*. If you choose **Amazon ECR**, use **Amazon ECR repository** and **Amazon ECR image** to choose the Docker image in your AWS account.
- To use a private Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. For **Image registry**, choose **Other registry**, and then enter the ARN of the credentials for your private Docker image. The credentials must be created by Secrets Manager. For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.

> **Note**
>
> CodeBuild overrides the `ENTRYPOINT` for custom Docker images.

Compute

Do one of the following:

- To use EC2 compute, choose **EC2**. EC2 compute offers optimized flexibility during action runs.
- To use Lambda compute, choose **Lambda**. Lambda compute offers optimized start-up speeds for your builds. Lambda supports faster builds due to a lower start-up latency. Lambda also automatically scales, so builds aren't waiting in queue to run. For information, see Run builds on AWS Lambda compute.

Service role

Do one of the following:

- If you do not have a CodeBuild service role, choose **New service role**. In **Role name**, enter a name for the new role.
- If you have a CodeBuild service role, choose **Existing service role**. In **Role ARN**, choose the service role.

> **Note**
>
> When you use the console to create a build project, you can create a CodeBuild service role at the same time. By default, the role works with that build project only. If you use the console to associate this service role with another build project, the role is updated to work with the other build project. A service role can work with up to 10 build projects.

Additional configuration

Auto-retry limit

Specify the number of additional automatic retries after a failed build. For example, if the auto-retry limit is set to 2, CodeBuild will call the `RetryBuild` API to automatically retry your build for up to 2 additional times.

Timeout

Specify a value, between 5 minutes and 36 hours, after which CodeBuild stops the build if it is not complete. If **hours** and **minutes** are left blank, the default value of 60 minutes is used.

Privileged

(Optional) Select **Enable this flag if you want to build Docker images or want your builds to get elevated privileges** only if you plan to use this build project to build Docker images. Otherwise, all associated builds that attempt to interact with the Docker daemon fail. You must also start the Docker daemon so that your builds can interact with it. One way to do this is to initialize the Docker daemon in the `install` phase of your build spec by running the following build commands. Do not run these commands if you chose a build environment image provided by CodeBuild with Docker support.

> **Note**
>
> By default, Docker daemon is enabled for non-VPC builds. If you would like to use Docker containers for VPC builds, see Runtime Privilege and Linux Capabilities on the Docker Docs website and enable privileged mode. Also, Windows does not support privileged mode.

```
- 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

If you want CodeBuild to work with your VPC:

- For **VPC**, choose the VPC ID that CodeBuild uses.
- For **VPC Subnets**, choose the subnets that include resources that CodeBuild uses.
- For **VPC Security groups**, choose the security groups that CodeBuild uses to allow access to resources in the VPCs.

For more information, see Use AWS CodeBuild with Amazon Virtual Private Cloud.

Compute

Choose one of the available options.

Registry credential

Specify a registry credential when the project is configured with a non-private registry image.

> **Note**
>
> This credential will only be utilized if the images are overridden with those from private registries.

Environment variables

Enter the name and value, and then choose the type of each environment variable for builds to use.

> **Note**
>
> CodeBuild sets the environment variable for your AWS Region automatically. You must set the following environment variables if you haven't added them to your buildspec.yml:
>
> - AWS_ACCOUNT_ID
> - IMAGE_REPO_NAME
> - IMAGE_TAG

Console and AWS CLI users can see environment variables. If you have no concerns about the visibility of your environment variable, set the **Name** and **Value** fields, and then set **Type** to **Plaintext**.

We recommend that you store an environment variable with a sensitive value, such as an AWS access key ID, an AWS secret access key, or a password as a parameter in Amazon EC2 Systems Manager Parameter Store or AWS Secrets Manager.

If you use Amazon EC2 Systems Manager Parameter Store, then for **Type**, choose **Parameter**. For **Name**, enter an identifier for CodeBuild to reference. For **Value**, enter the parameter's name as stored in Amazon EC2 Systems Manager Parameter Store. Using a parameter named `/CodeBuild/dockerLoginPassword` as an example, for **Type**, choose **Parameter**. For **Name**, enter `LOGIN_PASSWORD`. For **Value**, enter `/CodeBuild/dockerLoginPassword`.

Important

If you use Amazon EC2 Systems Manager Parameter Store, we recommend that you store parameters with parameter names that start with `/CodeBuild/` (for example, `/CodeBuild/dockerLoginPassword`). You can use the CodeBuild console to create a parameter in Amazon EC2 Systems Manager. Choose **Create parameter**, and then follow the instructions in the dialog box. (In that dialog box, for **KMS key**, you can specify the ARN of an AWS KMS key in your account. Amazon EC2 Systems Manager uses this key to encrypt the parameter's value during storage and decrypt it during retrieval.) If you use the CodeBuild console to create a parameter, the console starts the parameter name with `/CodeBuild/` as it is being stored. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the *Amazon EC2 Systems Manager User Guide*.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the `ssm:GetParameters` action. If you chose **New service role** earlier, CodeBuild includes this action in the default service role for your build project. However, if you chose **Existing service role**, you must include this action to your service role separately.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store with parameter names that do not start with `/CodeBuild/`, and you chose **New service role**, you must update that service role to allow access to parameter names that do not start with `/CodeBuild/`. This is because that service role allows access only to parameter names that start with `/CodeBuild/`.

If you choose **New service role**, the service role includes permission to decrypt all parameters under the `/CodeBuild/` namespace in the Amazon EC2 Systems Manager Parameter Store.

Environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named `MY_VAR` with a value of `my_value`, and you set an environment variable named `MY_VAR` with a value of `other_value`, then `my_value` is replaced by `other_value`. Similarly, if the Docker image already contains an environment variable named `PATH` with a value of `/usr/local/sbin:/usr/local/bin`, and you set an environment variable named `PATH` with a value of `$PATH:/usr/share/ant/bin`, then `/usr/`

`local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`.
Do not set any environment variable with a name that begins with `CODEBUILD_`. This prefix is reserved for internal use.
If an environment variable with the same name is defined in multiple places, the value is determined as follows:

- The value in the start build operation call takes highest precedence.
- The value in the build project definition takes next precedence.
- The value in the buildspec declaration takes lowest precedence.

If you use Secrets Manager, for **Type**, choose **Secrets Manager**. For **Name**, enter an identifier for CodeBuild to reference. For **Value**, enter a `reference-key` using the pattern *`secret-id`*`:`*`json-key`*`:`*`version-stage`*`:`*`version-id`*. For information, see Secrets Manager reference-key in the buildspec file.

Important

If you use Secrets Manager, we recommend that you store secrets with names that start with `/CodeBuild/` (for example, `/CodeBuild/dockerLoginPassword`). For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.
If your build project refers to secrets stored in Secrets Manager, the build project's service role must allow the `secretsmanager:GetSecretValue` action. If you chose **New service role** earlier, CodeBuild includes this action in the default service role for your build project. However, if you chose **Existing service role**, you must include this action to your service role separately.
If your build project refers to secrets stored in Secrets Manager with secret names that do not start with `/CodeBuild/`, and you chose **New service role**, you must update the service role to allow access to secret names that do not start with `/CodeBuild/`. This is because the service role allows access only to secret names that start with `/CodeBuild/`.
If you choose **New service role**, the service role includes permission to decrypt all secrets under the `/CodeBuild/` namespace in the Secrets Manager.

Buildspec

Build specifications

Do one of the following:

- If your source code includes a buildspec file, choose **Use a buildspec file**. By default, CodeBuild looks for a file named `buildspec.yml` in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root in **Buildspec name** (for example, `buildspec-two.yml` or `configuration/buildspec.yml`. If the buildspec file is in an S3 bucket, it must be in the same AWS Region as your build project. Specify the buildspec file using its ARN (for example, `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`).
- If your source code does not include a buildspec file, or if you want to run build commands different from the ones specified for the `build` phase in the `buildspec.yml` file in the source code's root directory, choose **Insert build commands**. For **Build commands**, enter the commands you want to run in the `build` phase. For multiple commands, separate each command by && (for example, `mvn test && mvn package`). To run commands in other phases, or if you have a long list of commands for the `build` phase, add a `buildspec.yml` file to the source code root directory, add the commands to the file, and then choose **Use the buildspec.yml in the source code root directory**.

For more information, see the Buildspec reference.

Batch configuration

You can run a group of builds as a single operation. For more information, see Run builds in batches.

Define batch configuration

Select to allow batch builds in this project.

Batch service role

Provides the service role for batch builds.

Choose one of the following:

- If you do not have a batch service role, choose **New service role**. In **Service role**, enter a name for the new role.

- If you have a batch service role, choose **Existing service role**. In **Service role**, choose the service role.

Batch builds introduce a new security role in the batch configuration. This new role is required as CodeBuild must be able to call the `StartBuild`, `StopBuild`, and `RetryBuild` actions on your behalf to run builds as part of a batch. Customers should use a new role, and not the same role they use in their build, for two reasons:

- Giving the build role `StartBuild`, `StopBuild`, and `RetryBuild` permissions would allow a single build to start more builds via the buildspec.
- CodeBuild batch builds provide restrictions that restrict the number of builds and compute types that can be used for the builds in the batch. If the build role has these permissions, it is possible the builds themselves could bypass these restrictions.

Allowed compute types for batch

Select the compute types allowed for the batch. Select all that apply.

Allowed fleets for batch

Select the fleets allowed for the batch. Select all that apply.

Maximum builds allowed in batch

Enter the maximum number of builds allowed in the batch. If a batch exceeds this limit, the batch will fail.

Batch timeout

Enter the maximum amount of time for the batch build to complete.

Combine artifacts

Select **Combine all artifacts from batch into a single location** to have all of the artifacts from the batch combined into a single location.

Batch report mode

Select the desired build status report mode for batch builds.

> **Note**
>
> This field is only available when the project source is Bitbucket, GitHub, or GitHub Enterprise, and **Report build statuses to source provider when your builds start and finish** is selected under **Source**.

Aggregated builds

Select to have the statuses for all builds in the batch combined into a single status report.

Individual builds

Select to have the build statuses for all builds in the batch reported separately.

Artifacts

Type

Do one of the following:

- If you do not want to create any build output artifacts, choose **No artifacts**. You might want to do this if you're only running build tests or you want to push a Docker image to an Amazon ECR repository.
- To store the build output in an S3 bucket, choose **Amazon S3**, and then do the following:
  - If you want to use your project name for the build output ZIP file or folder, leave **Name** blank. Otherwise, enter the name. (If you want to output a ZIP file, and you want the ZIP file to have a file extension, be sure to include it after the ZIP file name.)
  - Select **Enable semantic versioning** if you want a name specified in the buildspec file to override any name that is specified in the console. The name in a buildspec file is calculated at build time and uses the Shell command language. For example, you can append a date and time to your artifact name so that it is always unique. Unique artifact names prevent artifacts from being overwritten. For more information, see Buildspec syntax.
  - For **Bucket name**, choose the name of the output bucket.
  - If you chose **Insert build commands** earlier in this procedure, then for **Output files**, enter the locations of the files from the build that you want to put into the build output ZIP file or folder. For multiple locations, separate each location with a comma (for example, `appspec.yml, target/my-app.jar`). For more information, see the description of `files` in Buildspec syntax.
  - If you do not want your build artifacts encrypted, select **Remove artifacts encryption**.

For each secondary set of artifacts you want:

1. For **Artifact identifier**, enter a value that is fewer than 128 characters and contains only alphanumeric characters and underscores.
2. Choose **Add artifact**.

3. Follow the previous steps to configure your secondary artifacts.
4. Choose **Save artifact**.

Additional configuration

Encryption key

(Optional) Do one of the following:

- To use the AWS managed key for Amazon S3 in your account to encrypt the build output artifacts, leave **Encryption key** blank. This is the default.
- To use a customer managed key to encrypt the build output artifacts, in **Encryption key**, enter the ARN of the KMS key. Use the format `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`*.

Cache type

For **Cache type**, choose one of the following:

- If you do not want to use a cache, choose **No cache**.
- If you want to use an Amazon S3 cache, choose **Amazon S3**, and then do the following:
  - For **Bucket**, choose the name of the S3 bucket where the cache is stored.
  - (Optional) For **Cache path prefix**, enter an Amazon S3 path prefix. The **Cache path prefix** value is similar to a directory name. It makes it possible for you to store the cache under the same directory in a bucket.

    > **Important**
    >
    > Do not append a trailing slash (/) to the end of the path prefix.

- If you want to use a local cache, choose **Local**, and then choose one or more local cache modes.

  > **Note**
  >
  > Docker layer cache mode is available for Linux only. If you choose it, your project must run in privileged mode.

Using a cache saves considerable build time because reusable pieces of the build environment are stored in the cache and used across builds. For information about

specifying a cache in the buildspec file, see Buildspec syntax. For more information about caching, see Cache builds to improve performance.

Logs

Choose the logs you want to create. You can create Amazon CloudWatch Logs, Amazon S3 logs, or both.

CloudWatch

If you want Amazon CloudWatch Logs logs:

CloudWatch logs

Select **CloudWatch logs**.

Group name

Enter the name of your Amazon CloudWatch Logs log group.

Stream name

Enter your Amazon CloudWatch Logs log stream name.

S3

If you want Amazon S3 logs:

S3 logs

Select **S3 logs**.

Bucket

Choose the name of the S3 bucket for your logs.

Path prefix

Enter the prefix for your logs.

Disable S3 log encryption

Select if you do not want your S3 logs encrypted.

Create a build project (AWS CLI)

For more information about using the AWS CLI with CodeBuild, see the Command line reference.

To create a CodeBuild build project using the AWS CLI, you create a JSON-formatted Project structure, fill in the structure, and call the `create-project` command to create the project.

Create the JSON file

Create a skeleton JSON file with the `create-project` command, using the `--generate-cli-skeleton` option:

```
aws codebuild create-project --generate-cli-skeleton > <json-file>
```

This creates a JSON file with the path and file name specified by *<json-file>*.

Fill in the JSON file

Modify the JSON data as follows and save your results.

```
{
  "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>",
  "autoRetryLimit": <auto-retry-limit>,
  "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>"
      ],
      "fleetsAllowed": [
        "<fleet-name>"
      ]
    },
    "timeoutInMins": <batch-timeout>,
    "batchReportMode": "REPORT_AGGREGATED_BATCH" | "REPORT_INDIVIDUAL_BUILDS"
  },
  "concurrentBuildLimit": <concurrent-build-limit>
}
```

Replace the following:

name

Required. The name for this build project. This name must be unique across all of the build projects in your AWS account.

description

Optional. The description for this build project.

source

Required. A ProjectSource object that contains information about this build project's source code settings. After you add a `source` object, you can add up to 12 more sources using the . These settings include the following:

source/**type**

Required. The type of repository that contains the source code to build. Valid values include:

- `CODECOMMIT`
- `CODEPIPELINE`
- `GITHUB`
- `GITHUB_ENTERPRISE`
- `GITLAB`
- `GITLAB_SELF_MANAGED`
- `BITBUCKET`
- `S3`
- `NO_SOURCE`

If you use `NO_SOURCE`, the buildspec cannot be a file because the project does not have a source. Instead, you must use the `buildspec` attribute to specify a YAML-formatted string for your buildspec. For more information, see Create a build project without a source.

source/**location**

Required unless you set *`<source-type>`* to `CODEPIPELINE`. The location of the source code for the specified repository type.

- For CodeCommit, the HTTPS clone URL to the repository that contains the source code and the buildspec file (for example, `https://git-codecommit.<region-id>.amazonaws.com/v1/repos/<repo-name>`).
- For Amazon S3, the build input bucket name, followed by the path and name of the ZIP file that contains the source code and the buildspec. For example:
  - For a ZIP file located at the root of the input bucket: `<bucket-name>/<object-name>.zip`.
  - For a ZIP file located in a subfolder in the input bucket: `<bucket-name>/<subfoler-path>/<object-name>.zip`.
- For GitHub, the HTTPS clone URL to the repository that contains the source code and the buildspec file. The URL must contain github.com. You must connect your AWS account to your GitHub account. To do this, use the CodeBuild console to create a build project.
  - Choose **Authorize application**. (After you have connected to your GitHub account, you do not need to finish creating the build project. You can close the CodeBuild console.)
- For GitHub Enterprise Server, the HTTP or HTTPS clone URL to the repository that contains the source code and the buildspec file. You must also connect your AWS account to your GitHub Enterprise Server account. To do this, use the CodeBuild console to create a build project.
  1. Create a personal access token in GitHub Enterprise Server.
  2. Copy this token to your clipboard so you can use it when you create your CodeBuild project. For more information, see Creating a personal access token for the command line on the GitHub Help website.
  3. When you use the console to create your CodeBuild project, in **Source**, for **Source provider**, choose **GitHub Enterprise**.
  4. For **Personal Access Token**, paste the token that was copied to your clipboard. Choose **Save Token**. Your CodeBuild account is now connected to your GitHub Enterprise Server account.
- For GitLab and GitLab self-managed, the HTTPS clone URL to the repository that contains the source code and the buildspec file. Note that if you use GitLab, the URL must contain gitlab.com. If you use GitLab self-managed, the URL does not need to contain gitlab.com. You must connect your AWS account to your GitLab or GitLab self-managed account. To do this, use the CodeBuild console to create a build project.

  - In the Developer Tools navigation pane, choose **Settings**, **Connections**, and then **Create connection**. On this page, create either a GitLab or GitLab self-managed connection, and then choose **Connect to GitLab**.
- For Bitbucket, the HTTPS clone URL to the repository that contains the source code and the buildspec file. The URL must contain bitbucket.org. You must also connect your AWS account to your Bitbucket account. To do this, use the CodeBuild console to create a build project.

  1. When you use the console to connect (or reconnect) with Bitbucket, on the Bitbucket **Confirm access to your account** page, choose **Grant access**. (After you have connected to your Bitbucket account, you do not need to finish creating the build project. You can close the CodeBuild console.)
- For AWS CodePipeline, do not specify a `location` value for `source`. CodePipeline ignores this value because when you create a pipeline in CodePipeline, you specify the source code location in the Source stage of the pipeline.

source/**gitCloneDepth**

Optional. The depth of history to download. Minimum value is 0. If this value is 0, greater than 25, or not provided, then the full history is downloaded with each build project. If your source type is Amazon S3, this value is not supported.

source/**buildspec**

Optional. The build specification definition or file to use. If this value is not provided or is set to an empty string, the source code must contain a `buildspec.yml` file in its root directory. If this value is set, it can be either an inline buildspec definition, the path to an alternate buildspec file relative to the root directory of your primary source, or the path to an S3 bucket. The bucket must be in the same AWS Region as the build project. Specify the buildspec file using its ARN (for example, `arn:aws:s3:::`*`<my-codebuild-sample2>`*`/buildspec.yml`). For more information, see Buildspec file name and storage location.

source/**auth**

Contains information about the authorization settings for CodeBuild to access the source code to be built.

source/auth/**type**

Required. The authorization type to use. Valid values are:

- OAUTH

- CODECONNECTIONS
- SECRETS_MANAGER

source/auth/**resource**

Optional. The resource value that applies to the specified authorization type. This can be the Secrets Manager ARN or the CodeConnections ARN.

source/**reportBuildStatus**

Specifies whether to send your source provider the status of a build's start and completion. If you set this with a source provider other than GitHub, GitHub Enterprise Server, or Bitbucket, an `invalidInputException` is thrown.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

source/**buildStatusConfig**

Contains information that defines how the CodeBuild build project reports the build status to the source provider. This option is only used when the source type is `GITHUB`, `GITHUB_ENTERPRISE`, or `BITBUCKET`.

source/buildStatusConfig/**context**

For Bitbucket sources, this parameter is used for the `name` parameter in the Bitbucket commit status. For GitHub sources, this parameter is used for the `context` parameter in the GitHub commit status.

For example, you can have the `context` contain the build number and the webhook trigger using the CodeBuild environment variables:

```
AWS CodeBuild sample-project Build #$CODEBUILD_BUILD_NUMBER -
 $CODEBUILD_WEBHOOK_TRIGGER
```

This results in the context appearing like this for build #24 triggered by a webhook pull request event:

```
AWS CodeBuild sample-project Build #24 - pr/8
```

source/buildStatusConfig/**targetUrl**

For Bitbucket sources, this parameter is used for the `url` parameter in the Bitbucket commit status. For GitHub sources, this parameter is used for the `target_url` parameter in the GitHub commit status.

For example, you can set the `targetUrl` to `https://aws.amazon.com/codebuild/`*`<path to build>`* and the commit status will link to this URL.

You can also include CodeBuild environment variables in the `targetUrl` to add additional information to the URL. For example, to add the build region to the URL, set the `targetUrl` to:

```
"targetUrl": "https://aws.amazon.com/codebuild/<path to build>?region=
$AWS_REGION"
```

If the build region is `us-east-2`, this will expand to:

```
https://aws.amazon.com/codebuild/<path to build>?region=us-east-2
```

source/**gitSubmodulesConfig**

Optional. Information about the Git submodules configuration. Used with CodeCommit, GitHub, GitHub Enterprise Server, and Bitbucket only.

source/gitSubmodulesConfig/**fetchSubmodules**

Set `fetchSubmodules` to `true` if you want to include the Git submodules in your repository. Git submodules that are included must be configured as HTTPS.

source/**InsecureSsl**

Optional. Used with GitHub Enterprise Server only. Set this value to `true` to ignore TLS warnings while connecting to your GitHub Enterprise Server project repository. The default value is `false`. `InsecureSsl` should be used for testing purposes only. It should not be used in a production environment.

source/**sourceIdentifier**

A user-defined identifier for the project source. Optional for the primary source. Required for secondary sources.

secondarySources

Optional. An array of ProjectSource objects that contain information about the secondary sources for a build project. You can add up to 12 secondary sources. The `secondarySources` objects use the same properties used by the object. In a secondary source object, the `sourceIdentifier` is required.

secondarySourceVersions

Optional. An array of ProjectSourceVersion objects. If `secondarySourceVersions` is specified at the build level, then they take precedence over this.

sourceVersion

Optional. The version of the build input to be built for this project. If not specified, the latest version is used. If specified, it must be one of:

- For CodeCommit, the commit ID, branch, or Git tag to use.
- For GitHub, the commit ID, pull request ID, branch name, or tag name that corresponds to the version of the source code you want to build. If a pull request ID is specified, it must use the format `pr/pull-request-ID` (for example `pr/25`). If a branch name is specified, the branch's HEAD commit ID is used. If not specified, the default branch's HEAD commit ID is used.
- For GitLab, the commit ID, pull request ID, branch name, tag name, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild.
- For Bitbucket, the commit ID, branch name, or tag name that corresponds to the version of the source code you want to build. If a branch name is specified, the branch's HEAD commit ID is used. If not specified, the default branch's HEAD commit ID is used.
- For Amazon S3, the version ID of the object that represents the build input ZIP file to use.

If `sourceVersion` is specified at the build level, then that version takes precedence over this `sourceVersion` (at the project level). For more information, see Source version sample with AWS CodeBuild.

artifacts

Required. A ProjectArtifacts object that contains information about this build project's output artifact settings. After you add an `artifacts` object, you can add up to 12 more artifacts using the . These settings include the following:

artifacts/**type**

Required. The type of build output artifact. Valid values are:

- `CODEPIPELINE`
- `NO_ARTIFACTS`
- `S3`

artifacts/**location**

Only used with the `S3` artifact type. Not used for other artifact types.

The name of the output bucket you created or identified in the prerequisites.

artifacts/**path**

Only used with the `S3` artifact type. Not used for other artifact types.

The path in of the output bucket to place ZIP file or folder. If you do not specify a value for `path`, CodeBuild uses `namespaceType` (if specified) and `name` to determine the path and name of the build output ZIP file or folder. For example, if you specify `MyPath` for `path` and `MyArtifact.zip` for `name`, the path and name would be `MyPath/MyArtifact.zip`.

artifacts/**namespaceType**

Only used with the `S3` artifact type. Not used for other artifact types.

The namespace of the build output ZIP file or folder. Valid values include `BUILD_ID` and `NONE`. Use `BUILD_ID` to insert the build ID into the path of the build output ZIP file or folder. Otherwise, use `NONE`. If you do not specify a value for `namespaceType`, CodeBuild uses `path` (if specified) and `name` to determine the path and name of the build output ZIP file or folder. For example, if you specify `MyPath` for `path`, `BUILD_ID` for `namespaceType`, and `MyArtifact.zip` for `name`, the path and name would be `MyPath/`*`build-ID`*`/MyArtifact.zip`.

artifacts/**name**

Only used with the `S3` artifact type. Not used for other artifact types.

The name of the build output ZIP file or folder inside of `location`. For example, if you specify `MyPath` for `path` and `MyArtifact.zip` for `name`, the path and name would be `MyPath/MyArtifact.zip`.

artifacts/**overrideArtifactName**

Only used with the S3 artifact type. Not used for other artifact types.

Optional. If set to `true`, the name specified in the `artifacts` block of the buildspec file overrides `name`. For more information, see Build specification reference for CodeBuild.

artifacts/**packaging**

Only used with the S3 artifact type. Not used for other artifact types.

Optional. Specifies how to package the artifacts. Allowed values are:

NONE

Create a folder that contains the build artifacts. This is the default value.

ZIP

Create a ZIP file that contains the build artifacts.

secondaryArtifacts

Optional. An array of ProjectArtifacts objects that contain information about the secondary artifacts settings for a build project. You can add up to 12 secondary artifacts. The `secondaryArtifacts` uses many of the same settings used by the object.

cache

Required. A ProjectCache object that contains information about this build project's cache settings. For more information, see Cache builds.

environment

Required. A ProjectEnvironment object that contains information about this project's build environment settings. These settings include:

environment/**type**

Required. The type of build environment. For more information, see type in the *CodeBuild API Reference*.

environment/**image**

Required. The Docker image identifier used by this build environment. Typically, this identifier is expressed as *`image-name`*:*`tag`*. For example, in the Docker repository that CodeBuild

uses to manage its Docker images, this could be `aws/codebuild/standard:5.0`. In Docker Hub, `maven:3.3.9-jdk-8`. In Amazon ECR, *`account-id`*`.dkr.ecr.`*`region-id`*`.amazonaws.com/`*`your-Amazon-ECR-repo-name`*`:`*`tag`*. For more information, see Docker images provided by CodeBuild.

environment/**computeType**

Required. Specifies the compute resources used by this build environment. For more information, see computeType in the *CodeBuild API Reference*.

environment/**certificate**

Optional. The ARN of the Amazon S3 bucket, path prefix, and object key that contains the PEM-encoded certificate. The object key can be either just the .pem file or a .zip file containing the PEM-encoded certificate. For example, if your Amazon S3 bucket name is *`<my-bucket>`*, your path prefix is *`<cert>`*, and your object key name is *`<certificate.pem>`*, then acceptable formats for `certificate` are *`<my-bucket/cert/certificate.pem>`* or `arn:aws:s3:::`*`<my-bucket/cert/certificate.pem>`*.

environment/**environmentVariables**

Optional. An array of EnvironmentVariable objects that contains the environment variables you want to specify for this build environment. Each environment variable is expressed as an object that contains a `name`, `value`, and `type` of `name`, `value`, and `type`.

Console and AWS CLI users can see all environment variables. If you have no concerns about the visibility of your environment variable, set `name` and `value`, and set `type` to `PLAINTEXT`.

We recommend you store environment variables with sensitive values, such as an AWS access key ID, an AWS secret access key, or a password, as a parameter in Amazon EC2 Systems Manager Parameter Store or AWS Secrets Manager. For `name`, for that stored parameter, set an identifier for CodeBuild to reference.

If you use Amazon EC2 Systems Manager Parameter Store, for `value`, set the parameter's name as stored in the Parameter Store. Set `type` to `PARAMETER_STORE`. Using a parameter named `/CodeBuild/dockerLoginPassword` as an example, set `name` to `LOGIN_PASSWORD`. Set `value` to `/CodeBuild/dockerLoginPassword`. Set `type` to `PARAMETER_STORE`.

> **Important**
>
> If you use Amazon EC2 Systems Manager Parameter Store, we recommend that you store parameters with parameter names that start with `/CodeBuild/` (for example, `/`

`CodeBuild/dockerLoginPassword`). You can use the CodeBuild console to create a parameter in Amazon EC2 Systems Manager. Choose **Create parameter**, and then follow the instructions in the dialog box. (In that dialog box, for **KMS key**, you can specify the ARN of an AWS KMS key in your account. Amazon EC2 Systems Manager uses this key to encrypt the parameter's value during storage and decrypt it during retrieval.) If you use the CodeBuild console to create a parameter, the console starts the parameter name with `/CodeBuild/` as it is being stored. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the *Amazon EC2 Systems Manager User Guide*.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the `ssm:GetParameters` action. If you chose **New service role** earlier, CodeBuild includes this action in the default service role for your build project. However, if you chose **Existing service role**, you must include this action to your service role separately.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store with parameter names that do not start with `/CodeBuild/`, and you chose **New service role**, you must update that service role to allow access to parameter names that do not start with `/CodeBuild/`. This is because that service role allows access only to parameter names that start with `/CodeBuild/`.

If you choose **New service role**, the service role includes permission to decrypt all parameters under the `/CodeBuild/` namespace in the Amazon EC2 Systems Manager Parameter Store.

Environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named `MY_VAR` with a value of `my_value`, and you set an environment variable named `MY_VAR` with a value of `other_value`, then `my_value` is replaced by `other_value`. Similarly, if the Docker image already contains an environment variable named `PATH` with a value of `/usr/local/sbin:/usr/local/bin`, and you set an environment variable named `PATH` with a value of `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`.

Do not set any environment variable with a name that begins with `CODEBUILD_`. This prefix is reserved for internal use.

If an environment variable with the same name is defined in multiple places, the value is determined as follows:

- The value in the start build operation call takes highest precedence.
- The value in the build project definition takes next precedence.

- The value in the buildspec declaration takes lowest precedence.

If you use Secrets Manager, for `value`, set the parameter's name as stored in Secrets Manager. Set `type` to `SECRETS_MANAGER`. Using a secret named `/CodeBuild/dockerLoginPassword` as an example, set `name` to `LOGIN_PASSWORD`. Set `value` to `/CodeBuild/dockerLoginPassword`. Set `type` to `SECRETS_MANAGER`.

Important

If you use Secrets Manager, we recommend that you store secrets with names that start with `/CodeBuild/` (for example, `/CodeBuild/dockerLoginPassword`). For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.
If your build project refers to secrets stored in Secrets Manager, the build project's service role must allow the `secretsmanager:GetSecretValue` action. If you chose **New service role** earlier, CodeBuild includes this action in the default service role for your build project. However, if you chose **Existing service role**, you must include this action to your service role separately.
If your build project refers to secrets stored in Secrets Manager with secret names that do not start with `/CodeBuild/`, and you chose **New service role**, you must update the service role to allow access to secret names that do not start with `/CodeBuild/`. This is because the service role allows access only to secret names that start with `/CodeBuild/`.
If you choose **New service role**, the service role includes permission to decrypt all secrets under the `/CodeBuild/` namespace in the Secrets Manager.

environment/**registryCredential**

Optional. A RegistryCredential object that specifies the credentials that provide access to a private Docker registry.

environment/registryCredential/**credential**

Specifies the ARN or name of credentials created using AWS Managed Services. You can use the name of the credentials only if they exist in your current Region.

environment/registryCredential/**credentialProvider**

The only valid value is `SECRETS_MANAGER`.

When this is set:

- `imagePullCredentials` must be set to `SERVICE_ROLE`.
- The image cannot be a curated image or an Amazon ECR image.

environment/**imagePullCredentialsType**

Optional. The type of credentials CodeBuild uses to pull images in your build. There are two valid values:

CODEBUILD

`CODEBUILD` specifies that CodeBuild uses its own credentials. You must edit your Amazon ECR repository policy to trust the CodeBuild service principal.

SERVICE_ROLE

Specifies that CodeBuild uses your build project's service role.

When you use a cross-account or private registry image, you must use `SERVICE_ROLE` credentials. When you use a CodeBuild curated image, you must use `CODEBUILD` credentials.

environment/**privilegedMode**

Set to `true` only if you plan to use this build project to build Docker images. Otherwise, all associated builds that attempt to interact with the Docker daemon fail. You must also start the Docker daemon so that your builds can interact with it. One way to do this is to initialize the Docker daemon in the `install` phase of your buildspec file by running the following build commands. Do not run these commands if you specified a build environment image provided by CodeBuild with Docker support.

> **Note**
>
> By default, Docker daemon is enabled for non-VPC builds. If you would like to use Docker containers for VPC builds, see Runtime Privilege and Linux Capabilities on the Docker Docs website and enable privileged mode. Also, Windows does not support privileged mode.

```
- 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

Required. The ARN of the service role CodeBuild uses to interact with services on behalf of the user (for example, `arn:aws:iam::account-id:role/role-name`).

autoRetryLimit

Optional. The number of additional automatic retries after a failed build. For example, if the auto-retry limit is set to 2, CodeBuild will call the `RetryBuild` API to automatically retry your build for up to 2 additional times.

timeoutInMinutes

Optional. The number of minutes, between 5 to 2160 (36 hours), after which CodeBuild stops the build if it is not complete. If not specified, the default of 60 is used. To determine if and when CodeBuild stopped a build due to a timeout, run the `batch-get-builds` command. To determine if the build has stopped, look in the output for a `buildStatus` value of `FAILED`. To determine when the build timed out, look in the output for the `endTime` value associated with a `phaseStatus` value of `TIMED_OUT`.

queuedTimeoutInMinutes

Optional. The number of minutes, between 5 to 480 (8 hours), after which CodeBuild stops the build if it is is still queued. If not specified, the default of 60 is used.

encryptionKey

Optional. The alias or ARN of the AWS KMS key used by CodeBuild to encrypt the build output. If you specify an alias, use the format `arn:aws:kms:region-ID:account-ID:key/key-ID` or, if an alias exists, use the format `alias/key-alias`. If not specified, the AWS-managed KMS key for Amazon S3 is used.

tags

Optional. An array of Tag objects that provide the tags you want to associate with this build project. You can specify up to 50 tags. These tags can be used by any AWS service that supports CodeBuild build project tags. Each tag is expressed as an object with a `key` and a `value`.

vpcConfig

Optional. A VpcConfig object that contains information information about the VPC configuration for your project. For more information, see Use AWS CodeBuild with Amazon Virtual Private Cloud.

These properties include:

vpcId

Required. The VPC ID that CodeBuild uses. Run this command to get a list of all VPC IDs in your Region:

```
aws ec2 describe-vpcs --region <region-ID>
```

subnets

Required. An array of subnet IDs that include resources used by CodeBuild. Run this command to get these IDs:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region <region-ID>
```

securityGroupIds

Required. An array of security group IDs used by CodeBuild to allow access to resources in the VPC. Run this command to get these IDs:

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --<region-ID>
```

badgeEnabled

Optional. Specifies whether to include build badges with your CodeBuild project. Set to `true` to enable build badges, or `false` otherwise. For more information, see Build badges sample with CodeBuild.

logsConfig

A LogsConfig object that contains information about where this build's logs are located.

logsConfig/**cloudWatchLogs**

A CloudWatchLogsConfig object that contains information about pushing logs to CloudWatch Logs.

logsConfig/**s3Logs**

An S3LogsConfig object that contains information about pushing logs to Amazon S3.

fileSystemLocations

Optional. An array of ProjectFileSystemsLocation objects that contains informationabout your Amazon EFS configuration.

buildBatchConfig

Optional. The `buildBatchConfig` object is a ProjectBuildBatchConfig structure that contains the batch build configuration information for the project.

buildBatchConfig/**serviceRole**

The service role ARN for the batch build project.

buildBatchConfig/**combineArtifacts**

A Boolean value that specifies whether to combine the build artifacts for the batch build into a single artifact location.

buildBatchConfig/restrictions/**maximumBuildsAllowed**

The maximum number of builds allowed.

buildBatchConfig/restrictions/**computeTypesAllowed**

An array of strings that specify the compute types that are allowed for the batch build. See Build environment compute types for these values.

buildBatchConfig/restrictions/**fleetsAllowed**

An array of strings that specify the fleets that are allowed for the batch build. See Run builds on reserved capacity fleets for more information.

buildBatchConfig/**timeoutInMinutes**

The maximum amount of time, in minutes, that the batch build must be completed in.

buildBatchConfig/**batchReportMode**

Specifies how build status reports are sent to the source provider for the batch build. Valid values include:

`REPORT_AGGREGATED_BATCH`

(Default) Aggregate all of the build statuses into a single status report.

`REPORT_INDIVIDUAL_BUILDS`

Send a separate status report for each individual build.

concurrentBuildLimit

The maximum number of concurrent builds that are allowed for this project.

New builds are only started if the current number of builds is less than or equal to this limit. If the current build count meets this limit, new builds are throttled and are not run.

Create the project

To create the project, run the **`create-project`** command again, passing your JSON file:

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

If successful, the JSON representation of a Project object appears in the console output. See the CreateProject Response Syntax for an example of this data.

Except for the build project name, you can change any of the build project's settings later. For more information, see Change a build project's settings (AWS CLI).

To start running a build, see Run a build (AWS CLI).

If your source code is stored in a GitHub repository, and you want CodeBuild to rebuild the source code every time a code change is pushed to the repository, see Start running builds automatically (AWS CLI).

Create a build project (AWS SDKs)

For information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Create a build project (AWS CloudFormation)

For information about using AWS CodeBuild with AWS CloudFormation, see the AWS CloudFormation template for CodeBuild in the *AWS CloudFormation User Guide*.

Create a notification rule

You can use notification rules to notify users when important changes, such as build successes and failures, occur. Notification rules specify both the events and the Amazon SNS topic that is used to send notifications. For more information, see What are notifications?

You can use the console or the AWS CLI to create notification rules for AWS CodeBuild.

To create a notification rule (console)

1. Sign in to the AWS Management Console and open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. Choose **Build**, choose **Build projects**, and then choose a build project where you want to add notifications.
3. On the build project page, choose **Notify**, and then choose **Create notification rule**. You can also go to the **Settings** page for the build project and choose **Create notification rule**.
4. In **Notification name**, enter a name for the rule.
5. In **Detail type**, choose **Basic** if you want only the information provided to Amazon EventBridge included in the notification. Choose **Full** if you want to include information provided to Amazon EventBridge and information that might be supplied by the CodeBuild or the notification manager.

   For more information, see Understanding Notification Contents and Security.
6. In **Events that trigger notifications**, select the events for which you want to send notifications. For more information, see Events for Notification Rules on Build Projects.
7. In **Targets**, do one of the following:

   - If you have already configured a resource to use with notifications, in **Choose target type**, choose either **Amazon Q Developer in chat applications (Slack)** or **SNS topic**. In **Choose target**, choose the name of the client (for a Slack client configured in Amazon Q Developer in chat applications) or the Amazon Resource Name (ARN) of the Amazon SNS topic (for Amazon SNS topics already configured with the policy required for notifications).
   - If you have not configured a resource to use with notifications, choose **Create target**, and then choose **SNS topic**. Provide a name for the topic after **codestar-notifications-**, and then choose **Create**.

> **Note**
>
> - If you create the Amazon SNS topic as part of creating the notification rule, the policy that allows the notifications feature to publish events to the topic is applied for you. Using a topic created for notification rules helps ensure that you subscribe only those users that you want to receive notifications about this resource.
> - You cannot create an Amazon Q Developer in chat applications client as part of creating a notification rule. If you choose Amazon Q Developer in chat applications (Slack), you will see a button directing you to configure a client in Amazon Q Developer in chat applications. Choosing that option opens the Amazon Q Developer in chat applications console. For more information, see Configure Integrations Between Notifications and Amazon Q Developer in chat applications.
> - If you want to use an existing Amazon SNS topic as a target, you must add the required policy for AWS CodeStar Notifications in addition to any other policies that might exist for that topic. For more information, see Configure Amazon SNS Topics for Notifications and Understanding Notification Contents and Security.

8. To finish creating the rule, choose **Submit**.
9. You must subscribe users to the Amazon SNS topic for the rule before they can receive notifications. For more information, see Subscribe Users to Amazon SNS Topics That Are Targets. You can also set up integration between notifications and Amazon Q Developer in chat applications to send notifications to Amazon Chime chatrooms. For more information, see Configure Integration Between Notifications and Amazon Q Developer in chat applications.

To create a notification rule (AWS CLI)

1. At a terminal or command prompt, run the **create-notification rule** command to generate the JSON skeleton:

```
aws codestarnotifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

You can name the file anything you want. In this example, the file is named *rule.json*.

2. Open the JSON file in a plain-text editor and edit it to include the resource, event types, and target you want for the rule. The following example shows a notification rule named **MyNotificationRule** for a build project named *MyBuildProject* in an AWS acccount with the ID *123456789012*. Notifications are sent with the full detail type to an Amazon SNS topic named *codestar-notifications-MyNotificationTopic* when builds are successful:

```
{
    "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"
}
```

   Save the file.

3. Using the file you just edited, at the terminal or command line, run the **create-notification-rule** command again to create the notification rule:

```
aws codestarnotifications create-notification-rule --cli-input-json
 file://rule.json
```

4. If successful, the command returns the ARN of the notification rule, similar to the following:

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

Change build project settings in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to change a build project's settings.

If you add test reporting to a build project, make sure your IAM role has the permissions described in Test report permissions.

Topics

Change a build project's settings (console)

To change the settings for a build project, perform the following procedure:

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. Do one of the following:
   - Choose the link for the build project you want to change, and then choose **Build details**.
   - Choose the button next to the build project you want to change, choose **View details**, and then choose **Build details**.

You can modify the following sections:

Sections

- Project configuration
- Source
- Environment
- Buildspec
- Batch configuration
- Artifacts
- Logs

Project configuration

In the **Project configuration** section, choose **Edit**. When your changes are complete, choose **Update configuration** to save the new configuration.

You can modify the following properties.

Description

Enter an optional description of the build project to help other users understand what this project is used for.

Build badge

Select **Enable build badge** to make your project's build status visible and embeddable. For more information, see Build badges sample.

> **Note**
>
> Build badge does not apply if your source provider is Amazon S3.

Enable concurrent build limit

If you want to limit the number of concurrent builds for this project, perform the following steps:

1. Select **Restrict number of concurrent builds this project can start**.
2. In **Concurrent build limit**, enter the maximum number of concurrent builds that are allowed for this project. This limit cannot be greater than the concurrent build limit set for the account. If you try to enter a number greater than the account limit, an error message is displayed.

New builds are only started if the current number of builds is less than or equal to this limit. If the current build count meets this limit, new builds are throttled and are not run.

Enable public build access

To make your project's build results available to the public, including users without access to an AWS account, select **Enable public build access** and confirm that you want to make the build results public. The following properties are used for public build projects:

Public build service role

Select **New service role** if you want to have CodeBuild create a new service role for you, or **Existing service role** if you want to use an existing service role.

The public build service role enables CodeBuild to read the CloudWatch Logs and download the Amazon S3 artifacts for the project's builds. This is required to make the project's build logs and artifacts available to the public.

Service role

Enter the name of the new service role or an existing service role.

To make your project's build results private, clear **Enable public build access**.

For more information, see Get public build project URLs.

Warning

The following should be kept in mind when making your project's build results public:

- All of a project's build results, logs, and artifacts, including builds that were run when the project was private, are available to the public.
- All build logs and artifacts are available to the public. Environment variables, source code, and other sensitive information may have been output to the build logs and artifacts. You must be careful about what information is output to the build logs. Some best practices are:
  - Do not store sensitive values, especially AWS access key IDs and secret access keys, in environment variables. We recommend that you use an Amazon EC2 Systems Manager Parameter Store or AWS Secrets Manager to store sensitive values.
  - Follow Best practices for using webhooks to limit which entities can trigger a build, and do not store the buildspec in the project itself, to ensure that your webhooks are as secure as possible.
- A malicious user can use public builds to distribute malicious artifacts. We recommend that project administrators review all pull requests to verify that the pull request is a legitimate change. We also recommend that you validate any artifacts with their checksums to make sure that the correct artifacts are being downloaded.

Additional information

For **Tags**, enter the name and value of any tags that you want supporting AWS services to use. Use **Add row** to add a tag. You can add up to 50 tags.

Source

In the **Source** section, choose **Edit**. When your changes are complete, choose **Update configuration** to save the new configuration.

You can modify the following properties:

Source provider

Choose the source code provider type. Use the following lists to make selections appropriate for your source provider:

> **Note**
>
> CodeBuild does not support Bitbucket Server.

Amazon S3

Bucket

Choose the name of the input bucket that contains the source code.

S3 object key or S3 folder

Enter the name of the ZIP file or the path to the folder that contains the source code. Enter a forward slash (/) to download everything in the S3 bucket.

Source version

Enter the version ID of the object that represents the build of your input file. For more information, see Source version sample with AWS CodeBuild.

CodeCommit

Repository

Choose the repository you want to use.

Reference type

Choose **Branch**, **Git tag**, or **Commit ID** to specify the version of your source code. For more information, see Source version sample with AWS CodeBuild.

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Git submodules

Select **Use Git submodules** if you want to include Git submodules in your repository.

Bitbucket

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

Choose **CodeConnections**, **OAuth**, **App password**, or **Personal access token** to connect to CodeBuild.

Connection

Select a Bitbucket connection or a Secrets Manager secret to connect through your specified connection type.

Repository

Choose **Repository in my Bitbucket account** or **Public repository** and enter the repository URL.

Source version

Enter a branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Git submodules

Select **Use Git submodules** if you want to include Git submodules in your repository.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

For **Status context**, enter the value to be used for the `name` parameter in the Bitbucket commit status. For more information, see build in the Bitbucket API documentation.

For **Target URL**, enter the value to be used for the `url` parameter in the Bitbucket commit status. For more information, see build in the Bitbucket API documentation.

The status of a build triggered by a webhook is always reported to the source provider. To have the status of a build that is started from the console or an API call reported to the source provider, you must select this setting.

If your project's builds are triggered by a webhook, you must push a new commit to the repo for a change to this setting to take effect.

In **Primary source webhook events**, select **Rebuild every time a code change is pushed to this repository** if you want CodeBuild to build the source code every time a code change is pushed to this repository. For more information about webhooks and filter groups, see Bitbucket webhook events.

GitHub

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

Choose **GitHub App**, **OAuth**, or **Personal access token** to connect to CodeBuild.

Connection

Select a GitHub connection or a Secrets Manager secret to connect through your specified connection type.

Repository

Choose **Repository in my GitHub account**, **Public repository**, or **GitHub scoped webhook** and enter the repository URL.

Source version

Enter a branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Git submodules

Select **Use Git submodules** if you want to include Git submodules in your repository.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

For **Status context**, enter the value to be used for the `context` parameter in the GitHub commit status. For more information, see Create a commit status in the GitHub developer guide.

For **Target URL**, enter the value to be used for the `target_url` parameter in the GitHub commit status. For more information, see Create a commit status in the GitHub developer guide.

The status of a build triggered by a webhook is always reported to the source provider. To have the status of a build that is started from the console or an API call reported to the source provider, you must select this setting.

If your project's builds are triggered by a webhook, you must push a new commit to the repo for a change to this setting to take effect.

In **Primary source webhook events**, select **Rebuild every time a code change is pushed to this repository** if you want CodeBuild to build the source code every time a code change is pushed to this repository. For more information about webhooks and filter groups, see GitHub webhook events.

GitHub Enterprise Server

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

Choose **CodeConnections** or **Personal access token** to connect to CodeBuild.

Connection

Select a GitHub Enterprise connection or a Secrets Manager secret to connect through your specified connection type.

Repository

Choose **Repository in my GitHub Enterprise account** or **GitHub Enterprise scoped webhook** and enter the repository URL.

Source version

Enter a pull request, branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild.

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Git submodules

Select **Use Git submodules** if you want to include Git submodules in your repository.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

For **Status context**, enter the value to be used for the `context` parameter in the GitHub commit status. For more information, see Create a commit status in the GitHub developer guide.

For **Target URL**, enter the value to be used for the `target_url` parameter in the GitHub commit status. For more information, see Create a commit status in the GitHub developer guide.

The status of a build triggered by a webhook is always reported to the source provider. To have the status of a build that is started from the console or an API call reported to the source provider, you must select this setting.

If your project's builds are triggered by a webhook, you must push a new commit to the repo for a change to this setting to take effect.

Insecure SSL

Select **Enable insecure SSL** to ignore SSL warnings while connecting to your GitHub Enterprise project repository.

In **Primary source webhook events**, select **Rebuild every time a code change is pushed to this repository** if you want CodeBuild to build the source code every time a code change is pushed to this repository. For more information about webhooks and filter groups, see GitHub webhook events.

GitLab

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

CodeConnections is used to connect GitLab to CodeBuild.

Connection

Select a GitLab connection to connect through CodeConnections.

Repository

Choose the repository you want to use.

Source version

Enter a pull request ID, branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild.

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

GitLab Self Managed

Credential

Choose **Default source credential** or **Custom source credential** and follow the instructions to manage the default source credential or customize the source credential.

Connection type

CodeConnections is used to connect GitLab Self Managed to CodeBuild.

Connection

Select a GitLab Self Managed connection to connect through CodeConnections.

Repository

Choose the repository you want to use.

Source version

Enter a pull request ID, branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild.

> **Note**
>
> We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

Git clone depth

Choose **Git clone depth** to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

Build status

Select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

Environment

In the **Environment** section, choose **Edit**. When your changes are complete, choose **Update configuration** to save the new configuration.

You can modify the following properties:

Provisioning model

To change the provisioning model, choose **Change provisioning model** and do one of the following:

- To use on-demand fleets managed by AWS CodeBuild, choose **On-demand**. With on-demand fleets, CodeBuild provides compute for your builds. The machines are destroyed when the build finishes. On-demand fleets are fully managed, and includes automatic scaling capabilities to handle spikes in demand.
- To use reserved capacity fleets managed by AWS CodeBuild, choose **Reserved capacity**, and then select a **Fleet name**. With reserved capacity fleets, you configure a set of dedicated instances for your build environment. These machines remain idle, ready to process builds or tests immediately and reduces build durations. With reserved capacity fleets, your machines are always running and will continue to incur costs as long they're provisioned.

For information, see Run builds on reserved capacity fleets.

Environment image

To change the build image, choose **Override image** and do one of the following:

- To use a Docker image managed by AWS CodeBuild, choose **Managed image**, and then make selections from **Operating system**, **Runtime(s)**, **Image**, and **Image version**. Make a selection from **Environment type** if it is available.
- To use another Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. If you choose **Other registry**, for **External registry URL**, enter the name and tag of the Docker image in Docker Hub, using the format *`docker repository`*/*`docker image name`*. If you choose **Amazon ECR**, use **Amazon ECR repository** and **Amazon ECR image** to choose the Docker image in your AWS account.
- To use a private Docker image, choose **Custom image**. For **Environment type**, choose **ARM**, **Linux**, **Linux GPU**, or **Windows**. For **Image registry**, choose **Other registry**, and then enter the ARN of the credentials for your private Docker image. The credentials must be created by Secrets Manager. For more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.

> **Note**
>
> CodeBuild overrides the `ENTRYPOINT` for custom Docker images.

Service role

Do one of the following:

- If you do not have a CodeBuild service role, choose **New service role**. In **Role name**, enter a name for the new role.
- If you have a CodeBuild service role, choose **Existing service role**. In **Role ARN**, choose the service role.

> **Note**
>
> When you use the console to create a build project, you can create a CodeBuild service role at the same time. By default, the role works with that build project only. If you use the console to associate this service role with another build project, the role is updated to work with the other build project. A service role can work with up to 10 build projects.

Additional configuration

Timeout

Specify a value, between 5 minutes and 36 hours, after which CodeBuild stops the build if it is not complete. If **hours** and **minutes** are left blank, the default value of 60 minutes is used.

Privileged

Select **Enable this flag if you want to build Docker images or want your builds to get elevated privileges.** only if you plan to use this build project to build Docker images. Otherwise, all associated builds that attempt to interact with the Docker daemon fail. You must also start the Docker daemon so that your builds can interact with it. One way to do this is to initialize the Docker daemon in the `install` phase of your build spec by running the following build commands. Do not run these commands if you chose a build environment image provided by CodeBuild with Docker support.

> **Note**
>
> By default, Docker daemon is enabled for non-VPC builds. If you would like to use Docker containers for VPC builds, see Runtime Privilege and Linux Capabilities on the Docker Docs website and enable privileged mode. Also, Windows does not support privileged mode.

```
- 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

If you want CodeBuild to work with your VPC:

- For **VPC**, choose the VPC ID that CodeBuild uses.
- For **VPC Subnets**, choose the subnets that include resources that CodeBuild uses.
- For **VPC Security groups**, choose the security groups that CodeBuild uses to allow access to resources in the VPCs.

For more information, see Use AWS CodeBuild with Amazon Virtual Private Cloud.

Compute

Choose one of the available options.

Registry credential

Specify a registry credential when the project is configured with a non-private registry image.

> **Note**
>
> This credential will only be utilized if the images are overridden with those from private registries.

Environment variables

Enter the name and value, and then choose the type of each environment variable for builds to use.

> **Note**
>
> CodeBuild sets the environment variable for your AWS Region automatically. You must set the following environment variables if you haven't added them to your buildspec.yml:
>
> - AWS_ACCOUNT_ID
> - IMAGE_REPO_NAME

- IMAGE_TAG

Console and AWS CLI users can see environment variables. If you have no concerns about the visibility of your environment variable, set the **Name** and **Value** fields, and then set **Type** to **Plaintext**.

We recommend that you store an environment variable with a sensitive value, such as an AWS access key ID, an AWS secret access key, or a password as a parameter in Amazon EC2 Systems Manager Parameter Store or AWS Secrets Manager.

If you use Amazon EC2 Systems Manager Parameter Store, then for **Type**, choose **Parameter**. For **Name**, enter an identifier for CodeBuild to reference. For **Value**, enter the parameter's name as stored in Amazon EC2 Systems Manager Parameter Store. Using a parameter named `/CodeBuild/dockerLoginPassword` as an example, for **Type**, choose **Parameter**. For **Name**, enter `LOGIN_PASSWORD`. For **Value**, enter `/CodeBuild/dockerLoginPassword`.

Important

If you use Amazon EC2 Systems Manager Parameter Store, we recommend that you store parameters with parameter names that start with `/CodeBuild/` (for example, `/CodeBuild/dockerLoginPassword`). You can use the CodeBuild console to create a parameter in Amazon EC2 Systems Manager. Choose **Create parameter**, and then follow the instructions in the dialog box. (In that dialog box, for **KMS key**, you can specify the ARN of an AWS KMS key in your account. Amazon EC2 Systems Manager uses this key to encrypt the parameter's value during storage and decrypt it during retrieval.) If you use the CodeBuild console to create a parameter, the console starts the parameter name with `/CodeBuild/` as it is being stored. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the *Amazon EC2 Systems Manager User Guide*.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the `ssm:GetParameters` action. If you chose **New service role** earlier, CodeBuild includes this action in the default service role for your build project. However, if you chose **Existing service role**, you must include this action to your service role separately.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store with parameter names that do not start with `/CodeBuild/`, and you chose **New service role**, you must update that service role to allow access to parameter names that do not start with `/CodeBuild/`. This is because that service role allows access only to parameter names that start with `/CodeBuild/`.
If you choose **New service role**, the service role includes permission to decrypt all parameters under the `/CodeBuild/` namespace in the Amazon EC2 Systems Manager Parameter Store.
Environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named `MY_VAR` with a value of `my_value`, and you set an environment variable named `MY_VAR` with a value of `other_value`, then `my_value` is replaced by `other_value`. Similarly, if the Docker image already contains an environment variable named `PATH` with a value of `/usr/local/sbin:/usr/local/bin`, and you set an environment variable named `PATH` with a value of `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`.
Do not set any environment variable with a name that begins with `CODEBUILD_`. This prefix is reserved for internal use.
If an environment variable with the same name is defined in multiple places, the value is determined as follows:

- The value in the start build operation call takes highest precedence.
- The value in the build project definition takes next precedence.
- The value in the buildspec declaration takes lowest precedence.

If you use Secrets Manager, for **Type**, choose **Secrets Manager**. For **Name**, enter an identifier for CodeBuild to reference. For **Value**, enter a `reference-key` using the pattern *`secret-id`*`:`*`json-key`*`:`*`version-stage`*`:`*`version-id`*. For information, see Secrets Manager reference-key in the buildspec file.

Important

If you use Secrets Manager, we recommend that you store secrets with names that start with `/CodeBuild/` (for example, `/CodeBuild/dockerLoginPassword`). For

more information, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.
If your build project refers to secrets stored in Secrets Manager, the build project's service role must allow the `secretsmanager:GetSecretValue` action. If you chose **New service role** earlier, CodeBuild includes this action in the default service role for your build project. However, if you chose **Existing service role**, you must include this action to your service role separately.
If your build project refers to secrets stored in Secrets Manager with secret names that do not start with `/CodeBuild/`, and you chose **New service role**, you must update the service role to allow access to secret names that do not start with `/CodeBuild/`. This is because the service role allows access only to secret names that start with `/CodeBuild/`.
If you choose **New service role**, the service role includes permission to decrypt all secrets under the `/CodeBuild/` namespace in the Secrets Manager.

Buildspec

In the **Buildspec** section, choose **Edit**. When your changes are complete, choose **Update configuration** to save the new configuration.

You can modify the following properties:

Build specifications

Do one of the following:

- If your source code includes a buildspec file, choose **Use a buildspec file**. By default, CodeBuild looks for a file named `buildspec.yml` in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root in **Buildspec name** (for example, `buildspec-two.yml` or `configuration/buildspec.yml`. If the buildspec file is in an S3 bucket, it must be in the same AWS Region as your build project. Specify the buildspec file using its ARN (for example, `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`).
- If your source code does not include a buildspec file, or if you want to run build commands different from the ones specified for the `build` phase in the `buildspec.yml` file in the source code's root directory, choose **Insert build commands**. For **Build commands**, enter the commands you want to run in the `build` phase. For multiple commands, separate each

command by && (for example, `mvn test && mvn package`). To run commands in other phases, or if you have a long list of commands for the `build` phase, add a `buildspec.yml` file to the source code root directory, add the commands to the file, and then choose **Use the buildspec.yml in the source code root directory**.

For more information, see the Buildspec reference.

Batch configuration

In the **Batch configuration** section, choose **Edit**. When your changes are complete, choose **Update configuration** to save the new configuration. For more information, see Run builds in batches.

You can modify the following properties:

Batch service role

Provides the service role for batch builds.

Choose one of the following:

- If you do not have a batch service role, choose **New service role**. In **Service role**, enter a name for the new role.
- If you have a batch service role, choose **Existing service role**. In **Service role**, choose the service role.

Batch builds introduce a new security role in the batch configuration. This new role is required as CodeBuild must be able to call the `StartBuild`, `StopBuild`, and `RetryBuild` actions on your behalf to run builds as part of a batch. Customers should use a new role, and not the same role they use in their build, for two reasons:

- Giving the build role `StartBuild`, `StopBuild`, and `RetryBuild` permissions would allow a single build to start more builds via the buildspec.
- CodeBuild batch builds provide restrictions that restrict the number of builds and compute types that can be used for the builds in the batch. If the build role has these permissions, it is possible the builds themselves could bypass these restrictions.

Allowed compute types for batch

Select the compute types allowed for the batch. Select all that apply.

Allowed fleets for batch

Select the fleets allowed for the batch. Select all that apply.

Maximum builds allowed in batch

Enter the maximum number of builds allowed in the batch. If a batch exceeds this limit, the batch will fail.

Batch timeout

Enter the maximum amount of time for the batch build to complete.

Combine artifacts

Select **Combine all artifacts from batch into a single location** to have all of the artifacts from the batch combined into a single location.

Batch report mode

Select the desired build status report mode for batch builds.

> **Note**
>
> This field is only available when the project source is Bitbucket, GitHub, or GitHub Enterprise, and **Report build statuses to source provider when your builds start and finish** is selected under **Source**.

Aggregated builds

Select to have the statuses for all builds in the batch combined into a single status report.

Individual builds

Select to have the build statuses for all builds in the batch reported separately.

Artifacts

In the **Artifacts** section, choose **Edit**. When your changes are complete, choose **Update configuration** to save the new configuration.

You can modify the following properties:

Type

Do one of the following:

- If you do not want to create any build output artifacts, choose **No artifacts**. You might want to do this if you're only running build tests or you want to push a Docker image to an Amazon ECR repository.
- To store the build output in an S3 bucket, choose **Amazon S3**, and then do the following:
  - If you want to use your project name for the build output ZIP file or folder, leave **Name** blank. Otherwise, enter the name. (If you want to output a ZIP file, and you want the ZIP file to have a file extension, be sure to include it after the ZIP file name.)
  - Select **Enable semantic versioning** if you want a name specified in the buildspec file to override any name that is specified in the console. The name in a buildspec file is calculated at build time and uses the Shell command language. For example, you can append a date and time to your artifact name so that it is always unique. Unique artifact names prevent artifacts from being overwritten. For more information, see Buildspec syntax.
  - For **Bucket name**, choose the name of the output bucket.
  - If you chose **Insert build commands** earlier in this procedure, then for **Output files**, enter the locations of the files from the build that you want to put into the build output ZIP file or folder. For multiple locations, separate each location with a comma (for example, `appspec.yml, target/my-app.jar`). For more information, see the description of `files` in Buildspec syntax.
  - If you do not want your build artifacts encrypted, select **Remove artifacts encryption**.

For each secondary set of artifacts you want:

1. For **Artifact identifier**, enter a value that is fewer than 128 characters and contains only alphanumeric characters and underscores.
2. Choose **Add artifact**.
3. Follow the previous steps to configure your secondary artifacts.
4. Choose **Save artifact**.

Additional configuration

Encryption key

Do one of the following:

- To use the AWS managed key Amazon S3 in your account to encrypt the build output artifacts, leave **Encryption key** blank. This is the default.

- To use a customer managed key to encrypt the build output artifacts, in **Encryption key**, enter the ARN of the customer managed key. Use the format `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`*.

Cache type

For **Cache type**, choose one of the following:

- If you do not want to use a cache, choose **No cache**.
- If you want to use an Amazon S3 cache, choose **Amazon S3**, and then do the following:
  - For **Bucket**, choose the name of the S3 bucket where the cache is stored.
  - (Optional) For **Cache path prefix**, enter an Amazon S3 path prefix. The **Cache path prefix** value is similar to a directory name. It makes it possible for you to store the cache under the same directory in a bucket.

    Important

    Do not append a trailing slash (/) to the end of the path prefix.
- If you want to use a local cache, choose **Local**, and then choose one or more local cache modes.

  Note

  Docker layer cache mode is available for Linux only. If you choose it, your project must run in privileged mode.

Using a cache saves considerable build time because reusable pieces of the build environment are stored in the cache and used across builds. For information about specifying a cache in the buildspec file, see Buildspec syntax. For more information about caching, see Cache builds to improve performance.

Logs

In the **Logs** section, choose **Edit**. When your changes are complete, choose **Update configuration** to save the new configuration.

You can modify the following properties:

Choose the logs you want to create. You can create Amazon CloudWatch Logs, Amazon S3 logs, or both.

CloudWatch

If you want Amazon CloudWatch Logs logs:

CloudWatch logs

Select **CloudWatch logs**.

Group name

Enter the name of your Amazon CloudWatch Logs log group.

Stream name

Enter your Amazon CloudWatch Logs log stream name.

S3

If you want Amazon S3 logs:

S3 logs

Select **S3 logs**.

Bucket

Choose the name of the S3 bucket for your logs.

Path prefix

Enter the prefix for your logs.

Disable S3 log encryption

Select if you do not want your S3 logs encrypted.

Change a build project's settings (AWS CLI)

For information about using the AWS CLI with AWS CodeBuild, see the Command line reference.

To update a CodeBuild project with the AWS CLI, you create a JSON file with the updated properties and pass that file to the `update-project` command. Any properties not contained in the update file remain unchanged.

In the update JSON file, only the `name` property and the modified properties are required. The `name` property identifies the project to modify. For any modified structures, the required parameters for those structures must also be included. For example, to modify the environment for the project, the `environment/type` and `environment/computeType` properties are required. Here is an example that updates the environment image:

```
{
  "name": "<project-name>",
  "environment": {
    "type": "LINUX_CONTAINER",
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/amazonlinux-x86_64-standard:4.0"
  }
}
```

If you need to obtain the current property values for a project, use the **batch-get-projects** command to obtain the current properties of the project you are modifying, and write the output to a file.

```
aws codebuild batch-get-projects --names "<project-name>" > project-info.json
```

The *project-info.json* file contains an array of projects, so it cannot be used directly to update a project. You can, however, copy the properties that you want to modify from the *project-info.json* file and paste them into your update file as a baseline for the properties you want to modify. For more information, see View a build project's details (AWS CLI).

Modify the update JSON file as described in Create a build project (AWS CLI), and save your results. When you are finished modifying the update JSON file, run the `update-project` command, passing the update JSON file.

```
aws codebuild update-project --cli-input-json file://<update-project-file>
```

If successful, the updated project JSON appears in the output. If any required parameters are missing, an error message is displayed in the output that identifies the missing parameters. For example, this is the error message displayed if the `environment/type` parameter is missing:

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

```
Parameter validation failed:
Missing required parameter in environment: "type"
```

Change a build project's settings (AWS SDKs)

For information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Multiple access tokens in CodeBuild

CodeBuild supports sourcing access tokens to third party providers from your secrets in AWS Secrets Manager or through AWS CodeConnections connections. You can set your secret or connection as the default credential for interactions with a specified third party provider such as GitHub, GitHub Enterprise, or Bitbucket.

You can set your source credentials at three different levels:

1. **Account level credentials for all projects:** These are default credentials for all projects in an AWS account. They will be used on a project when no project or source level credentials are specified.
2. **Source level credentials for a specific repository:** This is when a Secrets Manager secret or CodeConnections connection is defined on a project source. These credentials will only be used for operations on the specified source repository. This allows you to set up multiple access tokens with different permission scopes in the same project, and not use the default account level credentials.
3. **Project level fallback credentials:** You can set a project level fallback credential by using `NO_SOURCE` as primary source type and define a secret or connection on it. This is can be used when you have multiple sources on a project, but want to use the same credentials for them, or when you don't want to use the default account level credentials for your project.

Topics

Step 1: Create a Secrets Manager secret or a CodeConnections connection

Use the following instructions to create a Secrets Manager secret or a CodeConnections connection:

- Create and store a token in a Secrets Manager secret.
- Create a connection to GitHub
- Create a connection to to GitHub Enterprise Server
- Create a connection to Bitbucket

Step 2: Grant CodeBuild project IAM role access to Secrets Manager secrets

> **Note**
>
> Before you continue, you must have access to the token created in Secrets Manager or CodeConnections.

To grant CodeBuild project IAM role access to Secrets Manager or CodeConnections, you must add the following IAM policy.

To grant CodeBuild project IAM role access

1. Create an IAM role for your CodeBuild project by following the instructions to Allow CodeBuild to interact with other AWS services for your CodeBuild project.
2. Do one of the following:
   - Add the following IAM policy to your CodeBuild project role to grant access to your secret.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
```

```
            ],
            "Resource": [
                "<secret-arn>"
            ]
        }
    ]
}
```

(Optional) If you're using AWS KMS customer managed keys to encrypt a Secrets Manager secret, you can add the following policy statement to grant access.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "<kms-key-arn>",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:SecretARN": "<secret-arn>"
                }
            }
        }
    ]
}
```

- Add the following IAM policy to your CodeBuild project role to grant access to your connection.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "codeconnections:GetConnectionToken",
              "codeconnections:GetConnection"
            ],
            "Resource": [
```

```
                <connection-arn>
            ]
        }
    ]
}
```

Step 3: Configure Secrets Manager or CodeConnections tokens

You can set your source credentials at three different levels with either Secrets Manager or CodeConnections tokens.

Configure Secrets Manager or CodeConnections tokens as account level credentials

You can configure a Secrets Manager secret or CodeConnections connection as an account level credential and use it in a project.

AWS Management Console

To configure a connection as an account level credential in the AWS Management Console

1. For **Source provider**, choose **Bitbucket**, **GitHub**, or **GitHub Enterprise**.
2. For **Credential**, do one of the following:
   - Choose **Default source credential** to use your account's default source credential to apply to all projects.

     a. If you aren't connected to your source provider, choose **Manage default source credential**.

     b. For **Credential type**, choose a credential type.

     c. If you chose **CodeConnections**, choose to use an existing connection or create a new connection.

     If you chose a different credential type, for **Service** choose which service you'd like to use to store your token and do the following:

     - If you chose to use **Secrets Manager**, you can choose to use an existing secret connection or create a new secret and choose **Save**. For more information how to create a new secret, see Create and store a token in a Secrets Manager secret.

- If you chose to use **CodeBuild**, enter your token or your username and app password, and choose **Save**.

- Choose **Custom source credential** to use a custom source credential to override your account's default settings.

  a. For **Credential type**, choose a credential type.

  b. In **Connection**, choose to use an existing connection or create a new connection.

AWS CLI

To configure a connection as an account level credential in the AWS CLI

- Open a terminal (Linux, macOS, or Unix) or command prompt (Windows). Use the AWS CLI to run the **import-source-credentials** command.

  Use the following command to configure a Secrets Manager secret:

```
aws codebuild import-source-credentials \
    --token "<secret-arn>" \
    --server-type <source-provider> \
    --auth-type SECRETS_MANAGER \
    --region <aws-region>
```

  Use the following command to configure a CodeConnections connection:

```
aws codebuild import-source-credentials \
    --token "<connection-arn>" \
    --server-type <source-provider> \
    --auth-type CODECONNECTIONS \
    --region <aws-region>
```

  This command allows you to import a token as the account level default source credentials. When you import a credential using the ImportSourceCredentials API, CodeBuild will use the token for all interactions with the source provider, such as webhooks, build status reporting and git clone operations unless a more specific set of credentials has been configured in the project.

You can now use the token in your build project and run it. For more information, see Create a build project in AWS CodeBuild and Run AWS CodeBuild builds manually.

Configure multiple tokens as source level credentials

To use Secrets Manager secrets or CodeConnections connections as source level credentials, directly reference the token in CodeBuild project, and start a build.

AWS Management Console

To configure multiple tokens as source level credentials in the AWS Management Console

1. For **Source provider**, choose **GitHub**.
2. For **Credential**, do one of the following:
   - Choose **Default source credential** to use your account's default source credential to apply to all projects.
     a. If you aren't connected to GitHub, choose **Manage default source credential**.
     b. For **Credential type**, choose **GitHub App**.
     c. In **Connection**, choose to use an existing connection or create a new connection.
   - Choose **Custom source credential** to use a custom source credential to override your account's default settings.
     a. For **Credential type**, choose **GitHub App**.
     b. In **Connection**, choose to use an existing connection or create a new connection.
3. Choose **Add source** and repeat the process of choosing your source provider and credentials.

AWS CLI

To configure multiple tokens as source level credentials in the AWS CLI

- Open a terminal (Linux, macOS, or Unix) or command prompt (Windows). Use the AWS CLI to run the **create-project** command.

  Use the following command:

```
aws codebuild create-project --region <aws-region> \
```

```
    --name <project-name> \
    --artifacts type=NO_ARTIFACTS \
    --environment "type=LINUX_CONTAINER,
                   computeType=BUILD_GENERAL1_SMALL,
                   image=aws/codebuild/amazonlinux-x86_64-standard:5.0" \
    --service-role <service-role-name> \
    --source "type=GITHUB,
              location=<github-repository-1>,
              auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn-1>}"
 \
    --secondary-sources "type=GITHUB,
              location=<github-repository-2>,
              auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn-2>},
              sourceIdentifier=secondary"

aws codebuild start-build --region <aws-region> --project-name <project-name>
```

Set a project level source credential fallback

To set up project level source credential fallback, use NO_SOURCE for your project's primary source and reference the token.

```
aws codebuild create-project \
    --name <project-name> \
    --service-role <service-role-name> \
    --artifacts type=NO_ARTIFACTS \
    --environment "type=LINUX_CONTAINER,
                   computeType=BUILD_GENERAL1_SMALL,
                   image=aws/codebuild/amazonlinux-x86_64-standard:5.0" \
    --service-role <service-role-name> \
    --source "type=NO_SOURCE,
              auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn>},
              buildspec=<buildspec>"
    --secondary-sources "type=GITHUB,
                         location=<github-repository>,
                         sourceIdentifier=secondary"

aws codebuild start-build --region <aws-region> --project-name <project_name>
```

When using NO_SOURCE, a buildspec typically is provided within the source model as it is not directly configured to use an external source to fetch the buildspec. Commonly, a NO_SOURCE

source will handle cloning all relevant repositories from within the buildspec. To ensure the configured credentials are available for those operations, you can enable the `git-credential-helper` option in the buildspec.

```
env:
  git-credential-helper: yes
```

During the build, CodeBuild will then read the `AuthServer` field from the configured token and use the token credentials for all git requests to that particular third party source provider.

Additional setup options

You can configure Secrets Manager account level credentials by using AWS CloudFormation templates. You can use the following AWS CloudFormation template to set an account level credential:

```
Parameters:
  GitHubToken:
    Type: String
    NoEcho: true
    Default: placeholder
Resources:
  CodeBuildAuthTokenSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: CodeBuild auth token
      Name: codebuild-auth-token
      SecretString:
        !Join
          - ''
          - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'
            - !Ref GitHubToken
            - '"}'
      Tags:
        - Key: codebuild:source:provider
          Value: github
        - Key: codebuild:source:type
          Value: personal_access_token
  CodeBuildSecretsManagerAccountCredential:
    Type: AWS::CodeBuild::SourceCredential
    Properties:
      ServerType: GITHUB
```

```
      AuthType: SECRETS_MANAGER
      Token: !Ref CodeBuildAuthTokenSecret
```

> **Note**
>
> If you're also creating a project in the same stack, use the AWS CloudFormation attribute DependsOn to ensure the `AccountCredential` is created before the project.

You can also configure Secrets Manager multiple source level credentials by using AWS CloudFormation templates. You can use the following AWS CloudFormation template to use multiple tokens to pull in multiple sources:

```
Parameters:
  GitHubTokenOne:
    Type: String
    NoEcho: true
    Default: placeholder
  GitHubTokenTwo:
    Type: String
    NoEcho: true
    Default: placeholder

Resources:
  CodeBuildSecretsManagerProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Name: codebuild-multitoken-example
      ServiceRole: <service-role>
      Environment:
        Type: LINUX_CONTAINER
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux-x86_64-standard:5.0
      Source:
        Type: GITHUB
        Location: <github-repository-one>
        Auth:
          Type: SECRETS_MANAGER
          Resource: !Ref CodeBuildAuthTokenSecretOne
      SecondarySources:
        - Type: GITHUB
          Location: <github-repository-two>
```

```
        Auth:
          Type: SECRETS_MANAGER
          Resource: !Ref CodeBuildAuthTokenSecretTwo
        SourceIdentifier: secondary
    Artifacts:
      Type: NO_ARTIFACTS
    LogsConfig:
      CloudWatchLogs:
        Status: ENABLED
CodeBuildProjectIAMRoleSecretAccess:
  Type: AWS::IAM::RolePolicy
  Properties:
    RoleName: <role-name>
    PolicyName: CodeBuildProjectIAMRoleSecretAccessPolicy
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - secretsmanager:GetSecretValue
          Resource:
            - !Ref CodeBuildAuthTokenSecretOne
            - !Ref CodeBuildAuthTokenSecretTwo
CodeBuildAuthTokenSecretOne:
  Type: AWS::SecretsManager::Secret
  Properties:
    Description: CodeBuild auth token one
    Name: codebuild-auth-token-one
    SecretString:
      !Join
        - ''
        - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'
          - !Ref GitHubTokenOne
          - '"}'
    Tags:
      - Key: codebuild:source:provider
        Value: github
      - Key: codebuild:source:type
        Value: personal_access_token
CodeBuildAuthTokenSecretTwo:
  Type: AWS::SecretsManager::Secret
  Properties:
    Description: CodeBuild auth token two
    Name: codebuild-auth-token-two
```

```
      SecretString:
        !Join
          - ''
          - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'
            - !Ref GitHubTokenTwo
            - '"}'
      Tags:
        - Key: codebuild:source:provider
          Value: github
        - Key: codebuild:source:type
          Value: personal_access_token
```

Delete build projects in AWS CodeBuild

You can use the CodeBuild console, AWS CLI, or AWS SDKs to delete a build project in CodeBuild. If you delete a project, its builds are not deleted.

> **Warning**
>
> You cannot delete a project that has builds and a resource policy. To delete a project with a resource policy and builds, you must first remove the resource policy and delete its builds.

Topics

Delete a build project (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. Do one of the following:

   - Choose the radio button next to the build project you want to delete, and then choose **Delete**.

- Choose the link for the build project you want to delete, and then choose **Delete**.

> **Note**
>
> By default, only the most recent 10 build projects are displayed. To view more build projects, choose a different value for **Projects per page** or use the back and forward arrows for viewing projects.

Delete a build project (AWS CLI)

1. Run the `delete-project` command:

   ```
   aws codebuild delete-project --name name
   ```

   Replace the following placeholder:

   - *name*: Required string. The name of the build project to delete. To get a list of available build projects, run the `list-projects` command. For more information, see View a list of build project names (AWS CLI).
2. If successful, no data and no errors appear in the output.

For more information about using the AWS CLI with AWS CodeBuild, see the Command line reference.

Delete a build project (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Get public build project URLs

AWS CodeBuild allows you to make the build results, logs, and artifacts for your build projects available to the general public. This allows contributors to your source repositories to view the results and download the artifacts of a build, without requiring them to have access to an AWS account.

When you make your project's builds available to the public, all of a project's build results, logs, and artifacts, including builds that were run when the project was private, are made available to the public. Likewise, when you make a public build project private, the build results for that project are no longer available to the public.

For information about how to change the public visibility of your project's build results, see **Enable public build access**.

CodeBuild provides a URL for the public builds for your project that is unique to your project.

To obtain the public URL for your build project, use the following procedure.

To obtain the URL of a public build project

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. Choose the link for the build project you want to obtain the public URL for.
4. The public URL is displayed in the **Public project URL** field in the **Configuration** section. You can choose the link to open the URL, or copy the URL with the copy button.

Warning

The following should be kept in mind when making your project's build results public:

- All of a project's build results, logs, and artifacts, including builds that were run when the project was private, are available to the public.
- All build logs and artifacts are available to the public. Environment variables, source code, and other sensitive information may have been output to the build logs and artifacts. You must be careful about what information is output to the build logs. Some best practices are:
  - Do not store sensitive values, especially AWS access key IDs and secret access keys, in environment variables. We recommend that you use an Amazon EC2 Systems Manager Parameter Store or AWS Secrets Manager to store sensitive values.
  - Follow Best practices for using webhooks to limit which entities can trigger a build, and do not store the buildspec in the project itself, to ensure that your webhooks are as secure as possible.

- A malicious user can use public builds to distribute malicious artifacts. We recommend that project administrators review all pull requests to verify that the pull request is a legitimate change. We also recommend that you validate any artifacts with their checksums to make sure that the correct artifacts are being downloaded.

Share build projects

Project sharing allows project owners to share their AWS CodeBuild projects with other AWS accounts or users. In this model, the account that owns the project (owner) shares a project with other accounts (consumers). A consumer cannot edit or run a project.

Topics

Share a project

The consumer can use both the AWS CLI and AWS CodeBuild console to view the project and builds you've shared. The consumer cannot edit or run the project.

You can add a project to an existing resource share or you can create one in the AWS RAM console.

> **Note**
>
> You cannot delete a project with builds that has been added to a resource share.

To share a project with organizational units or an entire organization, you must enable sharing with AWS Organizations. For more information, see Enable sharing with AWS Organizations in the *AWS RAM User Guide*.

You can use the AWS CodeBuild console, AWS RAM console, or the AWS CLI to share a project that you own.

Prerequisites for sharing projects

Before you start sharing a project, make sure your AWS account owns it. You cannot share a project that has been shared with you.

To share a project that you own (CodeBuild console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.

   > **Note**
   >
   > By default, only the 10 most recent build projects are displayed. To view more build projects, choose the gear icon, and then choose a different value for **Projects per page** or use the back and forward arrows.

3. Choose the project you want to share, and then choose **Share**. For more information, see Create a resource share in the *AWS RAM User Guide*.

To share a project that you own (AWS RAM console)

See Creating a resource share in the *AWS RAM User Guide*.

To share a project that you own (AWS RAM command)

Use the create-resource-share command.

To share a project that you own (CodeBuild command)

Use the put-resource-policy command:

1. Create a file named `policy.json` and copy the following into it.

```
{
  "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. Update `policy.json` with the project ARN and identifiers to share it with. The following example grants read-only access to the root user for the AWS account identified by 123456789012.

```
{
  "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. Run the put-resource-policy command.

```
aws codebuild put-resource-policy --resource-arn <project-arn> --policy file://
policy.json
```

4. Get the AWS RAM resource share ARN.

```
aws ram list-resources --resource-owner SELF --resource-arns <project-arn>
```

This will return a response similar to this:

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
}
```

From the response, copy the *<resource-share-arn>* value to use in the next step.

5. Run the AWS RAM promote-resource-share-created-from-policy command.

```
aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-share-arn>
```

Related services

Project sharing integrates with AWS Resource Access Manager (AWS RAM), a service that makes it possible for you to share your AWS resources with any AWS account or through AWS Organizations. With AWS RAM, you share resources by creating a *resource share* that specifies the resources and the consumers to share them with. Consumers can be individual AWS accounts, organizational units in AWS Organizations, or an entire organization in AWS Organizations.

For more information, see the *AWS RAM User Guide*.

Access CodeBuild projects shared with you

To access a shared project, a consumer's IAM role requires the `BatchGetProjects` permission. You can attach the following policy to their IAM role:

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
```

```
        "codebuild:BatchGetProjects"
    ]
}
```

For more information, see Using identity-based policies for AWS CodeBuild.

Unshare a shared project

An unshared project, including its builds, can be accessed only by its owner. If you unshare a project, any AWS account or user you previously shared it with cannot access the project or its builds.

To unshare a shared project that you own, you must remove it from the resource share. You can use the AWS CodeBuild console, AWS RAM console, or AWS CLI to do this.

To unshare a shared project that you own (AWS RAM console)

See Updating a resource share in the *AWS RAM User Guide*.

To unshare a shared project that you own (AWS CLI)

Use the disassociate-resource-share command.

To unshare project that you own (CodeBuild command)

Run the delete-resource-policy command and specify the ARN of the project you want to unshare:

```
aws codebuild delete-resource-policy --resource-arn project-arn
```

Identify a shared project

Owners and consumers can use the AWS CLI to identify shared projects.

To identify projects shared with your AWS account or user (AWS CLI)

Use the list-shared-projects command to return the projects that are shared with you.

Shared project permissions

Permissions for owners

A project owner can edit the project and use it to run builds.

Permissions for consumers

A project consumer can view a project and its builds, but cannot edit a project or use it to run builds.

Tag build projects

A *tag* is a custom attribute label that you or AWS assigns to an AWS resource. Each AWS tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, `Project`, or `Secret`). Tag keys are case sensitive.
- An optional field known as a *tag value* (for example, `111122223333`, `Production`, or a team name). Omitting the tag value is the same as using an empty string. Like tag keys, tag values are case sensitive.

Together these are known as key-value pairs. For information about the number of tags you can have on a project and restrictions on tag keys and values, see Tags.

Tags help you identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related. For example, you can assign the same tag to a CodeBuild project that you assign to an S3 bucket. For more information about using tags, see Tagging Best Practices.

In CodeBuild, the primary resources are the project and the report group. You can use the CodeBuild console, the AWS CLI, CodeBuild APIs, or AWS SDKs to add, manage, and remove tags for a project. In addition to identifying, organizing, and tracking your project with tags, you can use tags in IAM policies to help control who can view and interact with your project. For examples of tag-based access policies, see Using tags to control access to AWS CodeBuild resources.

Important

When using the reserved capacity feature, data cached on fleet instances, including source files, Docker layers, and cached directories specified in the buildspec, can be accessible to other projects within the same account. This is by design and allows projects within the same account to share fleet instances.

Topics

Add a tag to a project

Adding tags to a project can help you identify and organize your AWS resources and manage access to them. First, you add one or more tags (key-value pairs) to a project. Keep in mind that there are limits on the number of tags you can have on a project. There are restrictions on the characters you can use in the key and value fields. For more information, see Tags. After you have tags, you can create IAM policies to manage access to the project based on these tags. You can use the CodeBuild console or the AWS CLI to add tags to a project.

> **Important**
>
> When using the reserved capacity feature, data cached on fleet instances, including source files, Docker layers, and cached directories specified in the buildspec, can be accessible to other projects within the same account. This is by design and allows projects within the same account to share fleet instances.

For more information about adding tags to a project when you create it, see Add a tag to a project (console).

> **Important**
>
> Before you add a tag to a project, make sure to review any IAM policies that might use tags to control access to resources such as build projects. For examples of tag-based access policies, see Using tags to control access to AWS CodeBuild resources.

Topics

Add a tag to a project (console)

You can use the CodeBuild console to add one or more tags to a CodeBuild project.

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Build projects**, choose the name of the project where you want to add tags.
3. In the navigation pane, choose **Settings**. Choose **Build project tags**.
4. If no tags have been added to the project, choose **Add tag**. Otherwise, choose **Edit**, and then choose **Add tag**.
5. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
6. (Optional) To add another tag, choose **Add tag** again.
7. When you have finished adding tags, choose **Submit**.

Add a tag to a project (AWS CLI)

To add a tag to a project when you create it, see Create a build project (AWS CLI). In `create-project.json`, add your tags.

In these steps, we assume that you have already installed a recent version of the AWS CLI or updated to the current version. For more information, see Installing the AWS Command Line Interface.

If successful, this command returns nothing.

View tags for a project

Tags can help you identify and organize your AWS resources and manage access to them. For more information about using tags, see the Tagging best practices whitepaper. For examples of tag-based access policies, see Using tags to control access to AWS CodeBuild resources.

View tags for a project (console)

You can use the CodeBuild console to view the tags associated with a CodeBuild project.

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Build projects**, choose the name of the project where you want to view tags.
3. In the navigation pane, choose **Settings**. Choose **Build project tags**.

View tags for a project (AWS CLI)

To view tags for a build project, run the following command. Use the name of your project for the `--names` parameter.

```
aws codebuild batch-get-projects --names your-project-name
```

If successful, this command returns JSON-formatted information about your build project that includes something like the following:

```
{
    "tags": {
        "Status": "Secret",
        "Team": "JanesProject"
    }
}
```

If the project does not have tags, the `tags` section is empty:

```
"tags": []
```

Edit tags for a project

You can change the value for a tag associated with a project. You can also change the name of the key, which is equivalent to removing the current tag and adding a different one with the new name and the same value as the other key. Keep in mind that there are limits on the characters you can use in the key and value fields. For more information, see Tags.

> **Important**
>
> Editing tags for a project can impact access to that project. Before you edit the name (key) or value of a tag for a project, make sure to review any IAM policies that might use the key or value for a tag to control access to resources such as build projects. For examples of tag-based access policies, see Using tags to control access to AWS CodeBuild resources.

Edit a tag for a project (console)

You can use the CodeBuild console to edit the tags associated with a CodeBuild project.

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Build projects**, choose the name of the project where you want to edit tags.
3. In the navigation pane, choose **Settings**. Choose **Build project tags**.
4. Choose **Edit**.
5. Do one of the following:

   - To change the tag, enter a new name in **Key**. Changing the name of the tag is the equivalent of removing a tag and adding a new tag with the new key name.
   - To change the value of a tag, enter a new value. If you want to change the value to nothing, delete the current value and leave the field blank.
6. When you have finished editing tags, choose **Submit**.

Edit tags for a project (AWS CLI)

To add, change, or delete tags from a build project, see Change a build project's settings (AWS CLI). Update the `tags` section in the JSON-formatted data you use to update the project.

Remove a tag from a project

You can remove one or more tags associated with a project. Removing a tag does not delete the tag from other AWS resources that are associated with that tag.

> **Important**
>
> Removing tags for a project can impact access to that project. Before you remove a tag from a project, make sure to review any IAM policies that might use the key or value for a tag to control access to resources such as build projects. For examples of tag-based access policies, see Using tags to control access to AWS CodeBuild resources.

Remove a tag from a project (console)

You can use the CodeBuild console to remove the association between a tag and a CodeBuild project.

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Build projects**, choose the name of the project where you want to remove tags.

3. In the navigation pane, choose **Settings**. Choose **Build project tags**.
4. Choose **Edit**.
5. Find the tag you want to remove, and then choose **Remove tag**.
6. When you have finished removing tags, choose **Submit**.

Remove a tag from a project (AWS CLI)

To delete one or more tags from a build project, see Change a build project's settings (AWS CLI). Update the `tags` section in the JSON-formatted data with an updated list of tags that does not contain the ones you want to delete. If you want to delete all tags, update the `tags` section to:

```
"tags: []"
```

> **Note**
>
> If you delete a CodeBuild build project, all tag associations are removed from the deleted build project. You do not have to remove tags before you delete a build project.

Use runners with AWS CodeBuild

AWS CodeBuild supports integration with GitHub Actions runners, self-managed GitLab runners, and the Buildkite runner.

Topics

- Self-hosted GitHub Actions runners in AWS CodeBuild
- Self-managed GitLab runners in AWS CodeBuild
- Self-managed Buildkite runner in AWS CodeBuild

Self-hosted GitHub Actions runners in AWS CodeBuild

You can configure your project to set up self-hosted GitHub Actions runners in CodeBuild containers to process your GitHub Actions workflow jobs. This can be done by setting up a webhook using your CodeBuild project, and updating your GitHub Actions workflow YAML to use self-hosted runners hosted on CodeBuild machines.

The high-level steps to configure a CodeBuild project to run GitHub Actions jobs are as follows:

1. If you haven't done so already, create a personal access token or connect with an OAuth app to connect your project to GitHub.
2. Navigate to the CodeBuild console and create a CodeBuild project with a webhook and set up your webhook filters.
3. Update your GitHub Actions workflow YAML in GitHub to configure your build environment.

For a more detailed procedure, see Tutorial: Configure a CodeBuild-hosted GitHub Actions runner.

This feature allows your GitHub Actions workflow jobs to get native integration with AWS, which provides security and convenience through features like IAM, AWS Secrets Manager integration, AWS CloudTrail, and Amazon VPC. You can access latest instance types, including ARM-based instances.

Topics

About the CodeBuild-hosted GitHub Actions runner

The following are some common questions about the CodeBuild-hosted GitHub Actions runner.

When should I include the image and instance overrides in the label?

You can include the image and instance overrides in the label in order to specify different build environment for each of your GitHub Actions workflow jobs. This can be done without the need to create multiple CodeBuild projects or webhooks. For example, this is useful when you need to use a matrix for your workflow jobs.

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
```

```
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:${{ matrix.os }}
      - instance-size:${{ matrix.size }}
    strategy:
      matrix:
        include:
          - os: arm-3.0
            size: small
          - os: linux-5.0
            size: large
    steps:
      - run: echo "Hello World!"
```

Note

Quotation marks might be required if `runs-on` has multiple labels containing GitHub Actions context.

Can I use AWS CloudFormation for this feature?

Yes, you can include a filter group in your AWS CloudFormation template that specifies a GitHub Actions workflow job event filter in your project webhook.

```
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
```

For more information, see Filter GitHub webhook events (AWS CloudFormation).

If you need help setting up project credentials in your AWS CloudFormation template, see AWS::CodeBuild::SourceCredential in the *AWS CloudFormation User Guide* for more information.

How can I mask secrets when using this feature?

By default, secrets that are printed in the log is not masked. If you would like to mask your secrets, you can use the following syntax: `::add-mask::`*`value`*. The following is an example of how you can use this syntax in your YAML:

```
name: Secret Job
on: [push]
jobs:
  Secret-Job:
    runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    env:
      SECRET_NAME: "secret-name"
    steps:
      - run: echo "::add-mask::$SECRET_NAME"
```

For more information, see Masking a value in a log on GitHub.

Can I receive GitHub Actions webhook events from multiple repositories within a single project?

CodeBuild supports organization and global level webhooks, which receive events from a specified organization or enterprise. For more information, see GitHub global and organization webhooks.

Which regions support using a CodeBuild-hosted GitHub Actions runner?

CodeBuild-hosted GitHub Actions runners are supported in all CodeBuild regions. For more information about AWS Regions where CodeBuild is available, see AWS Services by Region.

Which platforms support using a CodeBuild-hosted GitHub Actions runner?

CodeBuild-hosted GitHub Actions runners are supported on both Amazon EC2 and AWS Lambda compute. You can use the following platforms: Amazon Linux 2, Amazon Linux 2023, Ubuntu, and Windows Server Core 2019. For more information, see EC2 compute images and Lambda compute images.

Tutorial: Configure a CodeBuild-hosted GitHub Actions runner

This tutorial shows you how to configure your CodeBuild projects to run GitHub Actions jobs. For more information about using GitHub Actions with CodeBuild see Tutorial: Configure a CodeBuild-hosted GitHub Actions runner.

To complete this tutorial, you must first:

- Connect with a personal access token, a Secrets Manager secret, OAuth app, or GitHub App. If you'd like to connect with an OAuth app, you must use the CodeBuild console to do so. If

you'd like to create a personal access token, you can either use the CodeBuild console or use the ImportSourceCredentials API. For more instructions, see GitHub and GitHub Enterprise Server access in CodeBuild.

- Connect CodeBuild to your GitHub account. To do so, you can do one of the following:
  - You can add GitHub as a source provider in the console. You can connect with either a personal access token, a Secrets Manager secret, OAuth app, or GitHub App. For instructions, see GitHub and GitHub Enterprise Server access in CodeBuild.
  - You can import your GitHub credentials via the ImportSourceCredentials API. This can only be done with a personal access token. If you connect using an OAuth app, you must connect using the console instead. For instructions, see Connect GitHub with an access token (CLI) .

> **Note**
>
> This only needs to be done if you haven't connected to GitHub for your account.

Step 1: Create a CodeBuild project with a webhook

In this step, you will create a CodeBuild project with a webhook and review it in the GitHub console. You can also choose GitHub Enterprise as your source provider. To learn more about creating a webhook within GitHub Enterprise, see GitHub manual webhooks.

To create a CodeBuild project with a webhook

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a build project. For information, see Create a build project (console) and Run a build (console).
3. In **Project type**, choose **Runner project**.

   In **Runner**:

   a. For **Runner provider**, choose **GitHub**.

   b. For **Runner location**, choose **Repository**.

   c. For Repository URL under **Repository**, choose **https://github.com/user-name/repository-name**.

> **Note**
>
> By default, your project will only receive `WORKFLOW_JOB_QUEUED` events for a single repository. If you would like to receive events for all repositories within an organization or enterprise, see GitHub global and organization webhooks.

4.
   - In **Environment**:
     - Choose a supported **Environment image** and **Compute**. Note that you have the option to override the image and instance settings by using a label in your GitHub Actions workflow YAML. For more information, see Step 2: Update your GitHub Actions workflow YAML
     - In **Buildspec**:
       - Note that your buildspec will be ignored unless `buildspec-override:true` is added as a label. Instead, CodeBuild will override it to use commands that will setup the self-hosted runner.
5. Continue with the default values and then choose **Create build project**.
6. Open the GitHub console at `https://github.com/`*`user-name`*`/`*`repository-name`*`/settings/hooks` to verify that a webhook has been created and is enabled to deliver **Workflow jobs** events.

Step 2: Update your GitHub Actions workflow YAML

In this step, you will update your GitHub Actions workflow YAML file in `GitHub` to configure your build environment and use GitHub Actions self-hosted runners in CodeBuild. For more information, see Using labels with self-hosted runners and Label overrides supported with the CodeBuild-hosted GitHub Actions runner.

Update your GitHub Actions workflow YAML

Navigate to `GitHub` and update the `runs-on` setting in your GitHub Actions workflow YAML to configure your build environment. To do so, you can do one of the following:

- You can specify the project name and run ID, in which case the build will use your existing project configuration for the compute, image, image version, and instance size. The project name is needed to link the AWS-related settings of your GitHub Actions job to a specific CodeBuild project. By including the project name in the YAML, CodeBuild is allowed to invoke jobs with

the correct project settings. By providing the run ID, CodeBuild will map your build to specific workflow runs and stop the build when the workflow run is cancelled. For more information, see github context.

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
```

> **Note**
>
> Make sure that your *<project-name>* matches the name of the project that you created in the previous step. If it doesn't match, CodeBuild will not process the webhook and the GitHub Actions workflow might hang.

The following is an example of a GitHub Actions workflow YAML:

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    steps:
      - run: echo "Hello World!"
```

- You can also override your image and compute type in the label. See Compute images supported with the CodeBuild-hosted GitHub Actions runner for a list of curated images. For using custom images, see Label overrides supported with the CodeBuild-hosted GitHub Actions runner. The compute type and image in the label will override the environment settings on your project. To override your environment settings for an CodeBuild EC2 or Lambda compute build, use the following syntax:

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
  - image:<environment-type>-<image-identifier>
  - instance-size:<instance-size>
```

The following is an example of a GitHub Actions workflow YAML:

```
name: Hello World
```

```
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:arm-3.0
      - instance-size:small
    steps:
      - run: echo "Hello World!"
```

- You can override the fleet used for your build in the label. This will override the fleet settings configured on your project to use the specified fleet. For more information, see Run builds on reserved capacity fleets. To override your fleet settings for an Amazon EC2 compute build, use the following syntax:

  ```
  runs-on:
    - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
    - fleet:<fleet-name>
  ```

  To override both the fleet and the image used for the build, use the following syntax:

  ```
  runs-on:
    - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
    - fleet:<fleet-name>
    - image:<environment-type>-<image-identifier>
  ```

  The following is an example of a GitHub Actions workflow YAML:

  ```
  name: Hello World
  on: [push]
  jobs:
    Hello-World-Job:
      runs-on:
        - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        - fleet:myFleet
        - image:arm-3.0
      steps:
        - run: echo "Hello World!"
  ```

- In order to run your GitHub Actions jobs on a custom image, you can configure a custom image in your CodeBuild project and avoid providing an image override label. CodeBuild will use the image configured in the project if no image override label is provided.
- Optionally, you can provide labels outside of those that CodeBuild supports. These labels will be ignored for the purpose of overriding attributes of the build, but will not fail the webhook request. For example, adding `testLabel` as a label will not prevent the build from running.

> **Note**
>
> If a dependency provided by GitHub-hosted runners is unavailable in the CodeBuild environment, you can install the dependency using GitHub Actions in your workflow run. For example, you can use the `setup-python` action to install Python for your build environment.

Run buildspec commands the INSTALL, PRE_BUILD, and POST_BUILD phases

By default, CodeBuild ignores any buildspec commands when running a self-hosted GitHub Actions build. To run buildspec commands during the build, `buildspec-override:true` can be added as a suffix to the label:

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
  - buildspec-override:true
```

By using this command, CodeBuild will create a folder called `actions-runner` in the container's primary source folder. When the GitHub Actions runner starts during the `BUILD` phase, the runner will run in the `actions-runner` directory.

There are several limitations when using a buildspec override in a self-hosted GitHub Actions build:

- CodeBuild will not run buildspec commands during the `BUILD` phase, as the self-hosted runner runs in the `BUILD` phase.
- CodeBuild will not download any primary or secondary sources during the `DOWNLOAD_SOURCE` phase. If you have a buildspec file configured, only that file will be downloaded from the project's primary source.

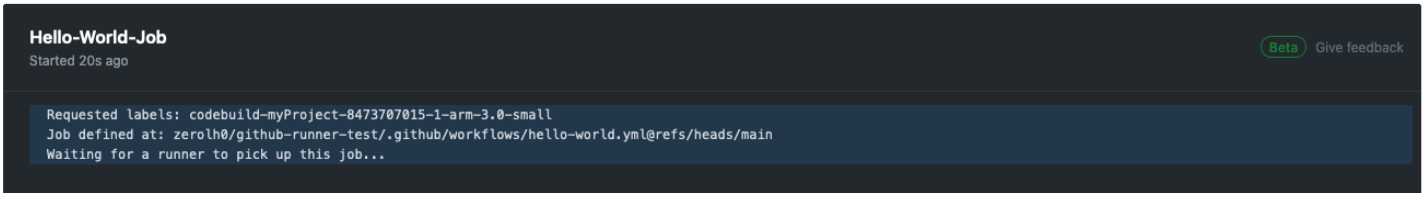
- If a build command fails in the `PRE_BUILD` or `INSTALL` phase, CodeBuild will not start the self-hosted runner and the GitHub Actions workflow job will need to be cancelled manually.
- CodeBuild fetches the runner token during the `DOWNLOAD_SOURCE` phase, which has an expiration time of one hour. If your `PRE_BUILD` or `INSTALL` phases exceed an hour, the runner token may expire before the GitHub self-hosted runner starts.

Step 3: Review your results

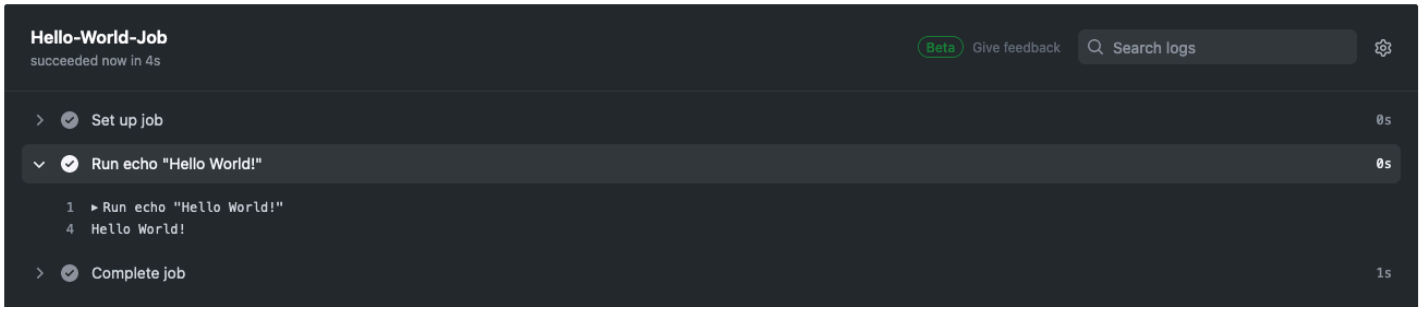
Whenever a GitHub Actions workflow run occurs, CodeBuild would receive the workflow job events through the webhook. For each job in the workflow, CodeBuild starts a build to run an ephemeral GitHub Actions runner. The runner is responsible for executing a single workflow job. Once the job is completed, the runner and the associated build process will be immediately terminated.

To view your workflow job logs, navigate to your repository in GitHub, choose **Actions**, choose your desired workflow, and then choose the specific **Job** that you'd like to review the logs for.

You can review the requested labels in the log while the job is waiting to be picked up by a self-hosted runner in CodeBuild.



Once the job is completed, you will be able to view the log of the job.



GitHub Actions runner configuration options

You can specify the following environment variables in your project configuration to modify the setup configuration of your self-hosted runners.

`CODEBUILD_CONFIG_GITHUB_ACTIONS_ORG_REGISTRATION_NAME`

CodeBuild will register self-hosted runners to the organization name specified as the value of this environment variable. For more information about registering runners at the organization level and the necessary permissions, see Create configuration for a just-in-time runner for an organization.

`CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATION_NAME`

CodeBuild will register self-hosted runners to the enterprise name specified as the value of this environment variable. For more information about registering runners at the enterprise level and the necessary permissions, see Create configuration for a just-in-time runner for an Enterprise.

> **Note**
>
> Enterprise runners are not available to organization repositories by default. For self-hosted runners to pick up workflow jobs, you might need to configure your runner group access settings. For more information, see Making enterprise runners available to repositories.

`CODEBUILD_CONFIG_GITHUB_ACTIONS_RUNNER_GROUP_ID`

CodeBuild will register self-hosted runners to the integer runner group ID stored as the value of this environment variable. By default, this value is 1. For more information about self-hosted runner groups, see Managing access to self-hosted runners using groups.

Filter GitHub Actions webhook events (AWS CloudFormation)

The following YAML-formatted portion of an AWS CloudFormation template creates a filter group that triggers a build when it evaluates to true. The following filter group specifies a GitHub Actions workflow job request with a workflow name matching the regular expression `\[CI-CodeBuild\]`.

```
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: CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: organization-name
        Scope: GITHUB_ORGANIZATION
      FilterGroups:
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

Filter GitHub Actions webhook events (AWS CDK)

The following AWS CDK template creates a filter group that triggers a build when it evaluates to true. The following filter group specifies a GitHub Actions workflow job request.

```
import { aws_codebuild as codebuild } from 'aws-cdk-lib';
import {EventAction, FilterGroup} from "aws-cdk-lib/aws-codebuild";

const source = codebuild.Source.gitHub({
      owner: 'owner',
      repo: 'repo',
      webhook: true,
      webhookFilters: [FilterGroup.inEventOf(EventAction.WORKFLOW_JOB_QUEUED)],
    })
```

Filter GitHub Actions webhook events (Terraform)

The following Terraform template creates a filter group that triggers a build when it evaluates to true. The following filter group specifies a GitHub Actions workflow job request.

```
resource "aws_codebuild_webhook" "example" {
  project_name = aws_codebuild_project.example.name
  build_type   = "BUILD"
  filter_group {
    filter {
```

```
      type    = "EVENT"
      pattern = "WORKFLOW_JOB_QUEUED"
    }
  }
}
```

Troubleshoot the webhook

Issue: The webhook you set up in Tutorial: Configure a CodeBuild-hosted GitHub Actions runner isn't working or your workflow job is hanging on GitHub.

Possible causes:

- Your webhook **Workflow jobs** event might be failing to trigger a build. Review the **Response** logs to view the response or error message.
- Your jobs are being assigned to the incorrect runner agent due to their label configuration. This issue can occur when one of your jobs within a single workflow run has fewer labels than another job. For example, if you have two jobs with the following labels in the same workflow run:
  - **Job 1**: `codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}`
  - **Job 2**: `codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}, instance-size:medium`

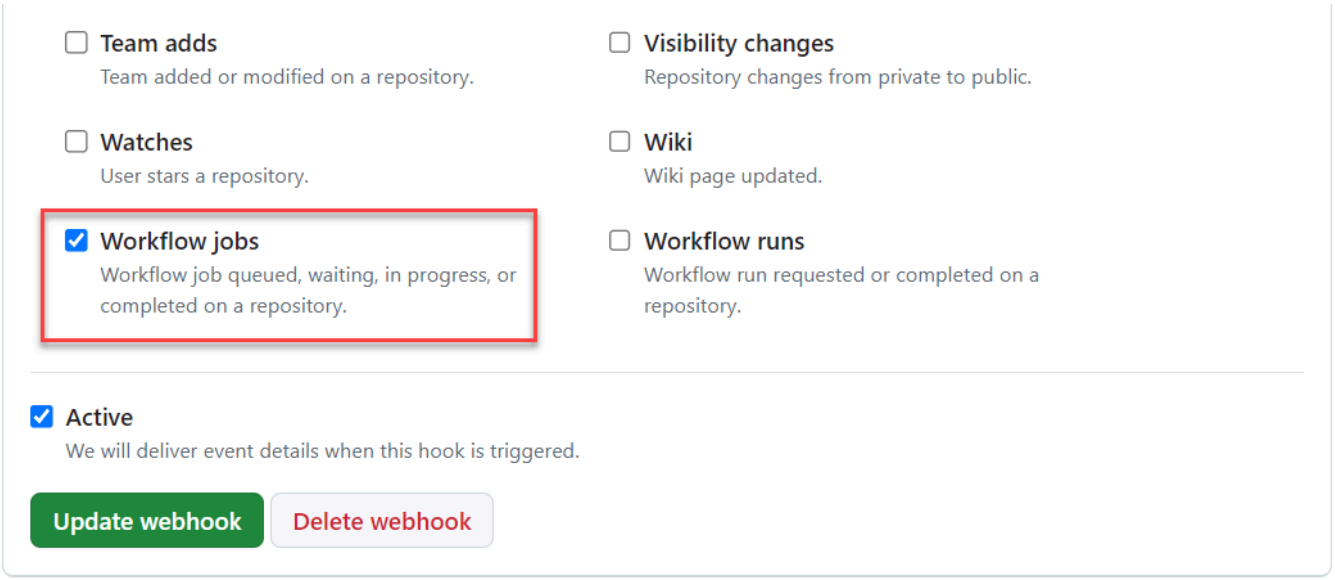
  When routing a self-hosted GitHub Actions job, GitHub will route the job to any runner with all the job's specified labels. This behavior means that **Job 1** can be picked up by either the runner created for **Job 1** or **Job 2**, but **Job 2** can only be picked up by the runner created for **Job 2** since it has an additional label. If **Job 1** is picked up by the runner created for **Job 2**, then **Job 2** will become stuck since the **Job 1** runner doesn't have the `instance-size:medium` label.

Recommended solutions:

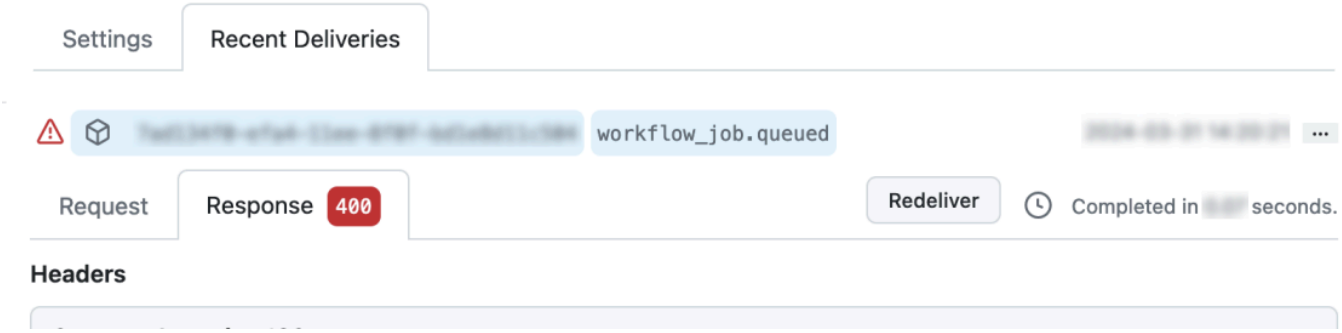
When creating multiple jobs within the same workflow run, use the same number of label overrides for each job or assign each job a custom label, such as `job1` or `job2`.

If the error persists, use the following instructions to debug the issue.

1. Open the GitHub console at `https://github.com/`*`user-name`*`/`*`repository-name`*`/settings/hooks` to view your repository's webhook settings. On this page, you'll see a webhook that was created for your repository.
2. Choose **Edit** and confirm that the webhook is enabled to deliver **Workflow jobs** events.



3. Navigate to the **Recent Deliveries** tab, find the corresponding `workflow_job.queued` event, and expand the event.
4. Review the **labels** field in the **Payload** and make sure it's as expected.
5. Finally, review the **Response** tab, as this contains the response or error message returned from CodeBuild.



6. Alternatively, you can debug webhook failures using GitHub's APIs. You can view recent deliveries for a webhook using the List deliveries for a repository webhook API:

```
gh api \
  -H "Accept: application/vnd.github+json" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  /repos/owner/repo/hooks/hook-id/deliveries
```

After finding the webhook delivery you're looking to debug and noting the delivery ID, you can use the Get a delivery for a repository webhook API. CodeBuild's response to the webhook's delivery payload can be found in the `response` section:

```
gh api \
  -H "Accept: application/vnd.github+json" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  /repos/owner/repo/hooks/hook-id/deliveries/delivery-id
```

Label overrides supported with the CodeBuild-hosted GitHub Actions runner

In your GitHub Actions workflow YAML, you can provide a variety of label overrides that modify your self-hosted runner build. Any builds not recognized by CodeBuild will be ignored but will not fail your webhook request. For example, the following workflow YAML includes overrides for image, instance size, fleet, and the buildspec:

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:${{ matrix.os }}
      - instance-size:${{ matrix.size }}
      - fleet:myFleet
      - buildspec-override:true
    strategy:
      matrix:
        include:
          - os: arm-3.0
            size: small
          - os: linux-5.0
            size: large
    steps:
      - run: echo "Hello World!"
```

Note

If your workflow job is hanging on GitHub, see Troubleshoot the webhook and Using custom labels to route jobs.

`codebuild-<project-name>-${{github.run_id}}-${{github.run_attempt}}` (required)

- Example: `codebuild-fake-project-${{ github.run_id }}-${{ github.run_attempt }}`
- Required for all GitHub Actions workflow YAMLs. *<project name>* should be equal to the name of the project for which the self-hosted runner webhook is configured.

`image:`*<environment-type>*`-`*<image-identifier>*

- Example: `image:arm-3.0`
- Overrides the image and environment type used when starting the self-hosted runner build with a curated image. To learn about supported values, see Compute images supported with the CodeBuild-hosted GitHub Actions runner.
  - To override the image and environment type used with a custom image, use `image:custom-`*<environment-type>*`-`*<custom-image-identifier>*
  - Example: `image:custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0`

> **Note**
>
> If the custom image resides in a private registry, see Configure a private registry credential for self-hosted runners.

`instance-size:`*<instance-size>*

- Example: `instance-size:medium`
- Overrides the instance type used when starting the self-hosted runner build. To learn about supported values, see Compute images supported with the CodeBuild-hosted GitHub Actions runner.

`fleet:`*<fleet-name>*

- Example: `fleet:myFleet`
- Overrides the fleet settings configured on your project to use the specified fleet. For more information, see Run builds on reserved capacity fleets.

`buildspec-override:<boolean>`

- Example: `buildspec-override:true`
- Allows the build to run buildspec commands in the `INSTALL`, `PRE_BUILD`, and `POST_BUILD` phases if set to `true`.

Single label override (legacy)

CodeBuild allows you to provide multiple overrides in a single label using the following:

- To override your environment settings for an Amazon EC2/Lambda compute build, use the following syntax:

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<environment-type>-<image-identifier>-<instance-size>
```

- To override your fleet settings for Amazon EC2 compute build, use the following syntax:

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}-
fleet-<fleet-name>
```

- To override both the fleet and the image used for the build, use the following syntax:

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-image-<image-version>-fleet-<fleet-name>
```

- To run buildspec commands during the build, `-with-buildspec` can be added as a suffix to the label:

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<image>-<image-version>-<instance-size>-with-buildspec
```

- Optionally, you can provide an instance size override without overriding the image. For Amazon EC2 builds, you can exclude both environment type and image identifier. For Lambda builds, you can exclude the image identifier.

Compute images supported with the CodeBuild-hosted GitHub Actions runner

In the label you configured in Tutorial: Configure a CodeBuild-hosted GitHub Actions runner, you can override your Amazon EC2 environment settings by using the values in the first three columns. CodeBuild provides the following Amazon EC2 compute images. For more information about

Environment type	Image identifier	Instance size	Platform	Resolved image	Definition
linux	4.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-standard:4.0	al/standard/4.0
linux	5.0	2xlarge gpu_small gpu_large	Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-standard:5.0	al/standard/5.0
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0	2xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0

Environment type	Image identifier	Instance size	Platform	Resolved image	Definition
ubuntu	5.0	small medium large xlarge 2xlarge gpu_small gpu_large	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0		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	N/A
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	N/A
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	N/A
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	N/A

In addition, you can override your Lambda environment settings by using the following values. For more information about CodeBuild Lambda compute, see Run builds on AWS Lambda compute. CodeBuild supports the following Lambda compute images:

Environment type	Image identifier	Instance size

`linux-lambda`
`arm-lambda`
	`dotnet6`
	`go1.21`
	`corretto11`
	`corretto17`
	`corretto21`
	`nodejs18`
	`nodejs20`
	`python3.11`
	`python3.12`
	`ruby3.2`
		`1GB`
		`2GB`
		`4GB`
		`8GB`
		`10GB`

For more information, see Build environment compute modes and types and Docker images provided by CodeBuild.

Self-managed GitLab runners in AWS CodeBuild

GitLab provides two execution modes to run GitLab jobs in your CI/CD pipeline. One mode is GitLab-hosted runners, which are managed by GitLab and fully integrated with GitLab. The other

mode is self-managed runners, which allows you to bring your own customized environment to run jobs in the GitLab CI/CD pipeline.

The high-level steps to configure a CodeBuild project to run GitLab CI/CD pipeline jobs are as follows:

1. If you haven't done so already, connect with an OAuth app to connect your project to GitLab.
2. Navigate to the CodeBuild console and create a CodeBuild project with a webhook and set up your webhook filters.
3. Update your GitLab CI/CD pipeline YAML in GitLab to configure your build environment.

For a more detailed procedure, see Tutorial: Configure a CodeBuild-hosted GitLab runner.

This feature allows your GitLab CI/CD pipeline jobs to get native integration with AWS, which provides security and convenience through features like IAM, AWS CloudTrail, and Amazon VPC. You can access latest instance types, including ARM-based instances.

Topics

About the CodeBuild-hosted GitLab runner

The following are some common questions about the CodeBuild-hosted GitLab runner.

What source types are supported for CodeBuild-hosted GitLab runners?

CodeBuild-hosted GitLab runners are supported for the `GITLAB` and `GITLAB_SELF_MANAGED` source type.

When should I include the image and instance overrides in the label?

You can include the image and instance overrides in the label in order to specify different build environment for each of your GitLab CI/CD pipeline jobs. This can be done without the need to create multiple CodeBuild projects or webhooks.

Can I use AWS CloudFormation for this feature?

Yes, you can include a filter group in your AWS CloudFormation template that specifies a GitLab workflow job event filter in your project webhook.

```
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
```

For more information, see Filter GitLab webhook events (AWS CloudFormation).

If you need help setting up project credentials in your AWS CloudFormation template, see AWS::CodeBuild::SourceCredential in the *AWS CloudFormation User Guide* for more information.

How can I mask secrets when using this feature?

By default, secrets that are printed in the log is not masked. If you would like to mask your secrets, you can do so by updating your CI/CD environment variable settings:

To mask secrets in GitLab

1. In your **GitLab Settings**, choose **CI/CD**.
2. In **Variables**, choose **Edit** for the secret you'd like to mask.
3. In **Visibility**, select **Mask variable**, and then choose **Update variable** to save your changes.

Can I receive GitLab webhook events from multiple projects within a single group?

CodeBuild supports group webhooks, which receive events from a specified GitLab group. For more information, see GitLab group webhooks.

Can I execute a job in docker executor for the self-managed runner? For example, I want to run a pipeline job on a specific image to maintain the same build environment in a separate and isolated container.

You can run the GitLab self-managed runner in CodeBuild with a specific image by creating the project with a custom image or overriding the image in your `.gitlab-ci.yml` file.

What executor does the self-managed runner in CodeBuild run with?

The self-managed runner in CodeBuild runs with the shell executor, where the build runs locally along with the GitLab runner that is running inside the docker container.

Can I provide buildspec commands along with the self-managed runner?

Yes, it is possible to add buildspec commands along with self-managed runner. You can provide the buildspec.yml file in your GitLab repository and use the `buildspec-override:true` tag in the **Tags** section of the job. For more information, see Buildspec file name and storage location.

Which regions support using a CodeBuild-hosted GitLab runner?

CodeBuild-hosted GitLab runners are supported in all CodeBuild regions. For more information about AWS Regions where CodeBuild is available, see AWS Services by Region.

Which platforms support using a CodeBuild-hosted GitLab runner?

CodeBuild-hosted GitLab runners are supported on both Amazon EC2 and AWS Lambda compute. You can use the following platforms: Amazon Linux 2, Amazon Linux 2023, Ubuntu, and Windows Server Core 2019. For more information, see EC2 compute images and Lambda compute images.

Tutorial: Configure a CodeBuild-hosted GitLab runner

This tutorial shows you how to configure your CodeBuild projects to run GitLab CI/CD pipeline jobs. For more information about using GitLab or GitLab Self Managed with CodeBuild, see Self-managed GitLab runners in AWS CodeBuild.

To complete this tutorial, you must first:

- Connect with a OAuth app by using CodeConnections. Note that when connecting with an OAuth app, you must use the CodeBuild console to do so. For more instructions, see GitLab access in CodeBuild.
- Connect CodeBuild to your GitLab account. To do so, you can add GitLab as a source provider in the console. For instructions, see GitLab access in CodeBuild.

> **Note**
>
> This only needs to be done if you haven't connected to GitLab for your account.
> With this feature, CodeBuild needs additional permissions. such as `create_runner` and `manage_runner` from the GitLab OAuth app. If there are existing CodeConnections for a

particular GitLab account, then it doesn't automatically request for permission updates. To do so, you can go to the CodeConnections console and create a dummy connection to the same GitLab account to trigger the reauthorization to get the additional prmissions. With this, all the existing connections can use the runner feature. Once complete, you can delete the dummy connection.

Step 1: Create a CodeBuild project with a webhook

In this step, you will create a CodeBuild project with a webhook and review it in the GitLab console.

To create a CodeBuild project with a webhook

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a build project. For information, see Create a build project (console) and Run a build (console).

   In **Project type**, choose **Runner project**.

   - In **Runner**:
     - For **Runner provider**, choose **GitLab**.
     - For **Credential**, choose one of the following:
       - Choose **Default source credential**. Default connection applies a default GitLab connection across all projects.
       - Choose **Custom source credential**. Custom connection applies a custom GitLab connection that overrides your account's default settings.

     > **Note**
     >
     > If you have not already created a connection to your provider, you'll have to create a new GitLab connection. For instructions, see Connect CodeBuild to GitLab.

     - For **Runner location**, choose **Repository**.
     - For **Repository**, choose the name of your project in GitLab by specifying the project path with the namespace.
   - In **Environment**:

- Choose a supported **Environment image** and **Compute**. Note that you have the option to override the image and instance settings by using a label in your GitLab CI/CD pipeline YAML. For more information, see Step 2: Create a .gitlab-ci.yml file in your repository.
- In **Buildspec**:
  - Note that your buildspec will be ignored unless `buildspec-override:true` is added as a label. Instead, CodeBuild will override it to use commands that will setup the self-managed runner.
-

3. Continue with the default values and then choose **Create build project**.
4. Open the GitLab console at `https://gitlab.com/`*`user-name`*`/`*`repository-name`*`/-/hooks` to verify that a webhook has been created and is enabled to deliver **Workflow jobs** events.

Step 2: Create a .gitlab-ci.yml file in your repository

In this step, you will create a `.gitlab-ci.yml` file in GitLab to configure your build environment and use GitLab self-managed runners in CodeBuild. For more information, see Use self-managed runners.

Update your GitLab CI/CD pipeline YAML

Navigate to `https://gitlab.com/`*`user-name`*`/`*`project-name`*`/-/tree/`*`branch-name`* and create a `.gitlab-ci.yml` file in your repository. You can configure your build environment by doing one of the following:

- You can specify the CodeBuild project name, in which case the build will use your existing project configuration for the compute, image, image version, and instance size. The project name is needed to link the AWS-related settings of your GitLab job to a specific CodeBuild project. By including the project name in the YAML, CodeBuild is allowed to invoke jobs with the correct project settings.

  ```
  tags:
      - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
  ```

  `$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME` is required to map the build to specific pipeline job runs and stop the build when the pipeline run is cancelled.

> **Note**
>
> Make sure that your *<project-name>* matches the name of the project that you created in CodeBuild. If it doesn't match, CodeBuild will not process the webhook and the GitLab CI/CD pipeline might hang.

The following is an example of a GitLab CI/CD pipeline YAML:

```
workflow:
  name: HelloWorld
stages:          # List of stages for jobs, and their order of execution
  - build

build-job:       # This job runs in the build stage, which runs first.
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
```

- You can also override your image and compute type in the tag. See Compute images supported with the CodeBuild-hosted GitLab runner for a list of curated images. For using custom images, see Label overrides supported with the CodeBuild-hosted GitLab runner. The compute type and image in the tag will override the environment settings on your project. To override your environment settings for an Amazon EC2 compute build, use the following syntax:

```
tags:
    - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - image:<environment-type>-<image-identifier>
    - instance-size:<instance-size>
```

The following is an example of a GitLab CI/CD pipeline YAML:

```
stages:
  - build

build-job:
  stage: build
```

```
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - image:arm-3.0
    - instance-size:small
```

- You can override the fleet used for your build in the tag. This will override the fleet settings configured on your project to use the specified fleet. For more information, see Run builds on reserved capacity fleets. To override your fleet settings for an Amazon EC2 compute build, use the following syntax:

```
tags:
    - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - fleet:<fleet-name>
```

  To override both the fleet and the image used for the build, use the following syntax:

```
tags:
    - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - fleet:<fleet-name>
    - image:<environment-type>-<image-identifier>
```

  The following is an example of a GitLab CI/CD pipeline YAML:

```
stages:
  - build

build-job:
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - fleet:myFleet
    - image:arm-3.0
```

- In order to run your GitLab CI/CD pipeline jobs on a custom image, you can configure a custom image in your CodeBuild project and avoid providing an image override label. CodeBuild will use the image configured in the project if no image override label is provided.

After you commit your changes to `.gitlab-ci.yml`, a GitLab pipeline will be triggered and the `build-job` will send a webhook notification that will start your build in CodeBuild.

Run buildspec commands the INSTALL, PRE_BUILD, and POST_BUILD phases

By default, CodeBuild ignores any buildspec commands when running a self-managed GitLab build. To run buildspec commands during the build, `buildspec-override:true` can be added as a suffix to `tags`:

```
tags:
    - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - buildspec-override:true
```

By using this command, CodeBuild will create a folder called `gitlab-runner` in the container's primary source folder. When the GitLab runner starts during the `BUILD` phase, the runner will run in the `gitlab-runner` directory.

There are several limitations when using a buildspec override in a self-managed GitLab build:

- CodeBuild will not run buildspec commands during the `BUILD` phase, as the self-managed runner runs in the `BUILD` phase.
- CodeBuild will not download any primary or secondary sources during the `DOWNLOAD_SOURCE` phase. If you have a buildspec file configured, only that file will be downloaded from the project's primary source.
- If a build command fails in the `PRE_BUILD` or `INSTALL` phase, CodeBuild will not start the self-managed runner and the GitLab CI/CD pipeline job will need to be cancelled manually.
- CodeBuild fetches the runner token during the `DOWNLOAD_SOURCE` phase, which has an expiration time of one hour. If your `PRE_BUILD` or `INSTALL` phases exceed an hour, the runner token may expire before the GitLab self-managed runner starts.

Step 3: Review your results

Whenever a GitLab CI/CD pipeline run occurs, CodeBuild would receive the CI/CD pipeline job events through the webhook. For each job in the CI/CD pipeline, CodeBuild starts a build to run an ephemeral GitLab runner. The runner is responsible for executing a single CI/CD pipeline job. Once the job is completed, the runner and the associated build process will be immediately terminated.

To view your CI/CD pipeline job logs, navigate to your repository in GitLab, choose **Build**, **Jobs**, and then choose the specific **Job** that you'd like to review the logs for.

You can review the requested labels in the log while the job is waiting to be picked up by a self-managed runner in CodeBuild.

Filter GitLab webhook events (AWS CloudFormation)

The following YAML-formatted portion of an AWS CloudFormation template creates a filter group that triggers a build when it evaluates to true. The following filter group specifies a GitLab CI/CD pipeline job request with a CI/CD pipeline name matching the regular expression `\[CI-CodeBuild\]`.

```
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: GITLAB
      Location: CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: group-name
        Scope: GITLAB_GROUP
      FilterGroups:
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

Label overrides supported with the CodeBuild-hosted GitLab runner

In your GitLab CI/CD pipeline YAML, you can provide a variety of label overrides that modify your self-managed runner build. Any builds not recognized by CodeBuild will be ignored but will not fail your webhook request. For example, the following YAML includes overrides for image, instance size, fleet, and the buildspec:

```
workflow:
  name: HelloWorld
stages:
  - build

build-job:
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - image:arm-3.0
    - instance-size:small
    - fleet:myFleet
    - buildspec-override:true
```

`codebuild-`*`<project-name>`*`-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME` (required)

- Example: `codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME`
- Required for all GitLab CI/CD pipeline YAMLs. *`<project name>`* should be equal to the name of the project for which the self-managed runner webhook is configured.

`image:`*`<environment-type>`*`-`*`<image-identifier>`*

- Example: `image:arm-3.0`
- Overrides the image and environment type used when starting the self-managed runner build. To learn about supported values, see Compute images supported with the CodeBuild-hosted GitLab runner.
  - To override the image and environment type used with a custom image, use `image:custom-`*`<environment-type>`*`-`*`<custom-image-identifier>`*
  - Example: `image:custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0`

> **Note**
>
> If the custom image resides in a private registry, see Configure a private registry credential for self-hosted runners.

`instance-size:<instance-size>`

- Example: `instance-size:small`
- Overrides the instance type used when starting the self-managed runner build. To learn about supported values, see Compute images supported with the CodeBuild-hosted GitLab runner.

`fleet:<fleet-name>`

- Example: `fleet:myFleet`
- Overrides the fleet settings configured on your project to use the specified fleet. For more information, see Run builds on reserved capacity fleets.

`buildspec-override:<boolean>`

- Example: `buildspec-override:true`
- Allows the build to run buildspec commands in the `INSTALL`, `PRE_BUILD`, and `POST_BUILD` phases if set to `true`.

Compute images supported with the CodeBuild-hosted GitLab runner

In the label you configured in Tutorial: Configure a CodeBuild-hosted GitLab runner, you can override your Amazon EC2 environment settings by using the values in the first three columns. CodeBuild provides the following Amazon EC2 compute images. For more information about

Environment type	Image identifier	Instance size	Platform	Image	Definition

`linux`	`4.0`	`small`
		`medium`
		`large`
		`xlarge`
			Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:4.0`	al/standard/4.0

`linux`	`5.0`	`2xlarge` `gpu_small`	Amazon Linux 2	`aws/codebuild/amaz`	al/standard/5.0

Environment type	Image identifier	Instance size	Platform	Image	Definition
		gpu_large		onlinux-x86_64-standard:5.0	
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0	2xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0
ubuntu	5.0	small medium	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0	large xlarge 2xlarge	Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0	gpu_small gpu_large	Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0

Environment type	Image identifier	Instance size	Platform	Image	Definition
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	N/A
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	N/A
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	N/A
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	N/A

In addition, you can override your Lambda environment settings by using the following values. For more information about CodeBuild Lambda compute, see Run builds on AWS Lambda compute. CodeBuild supports the following Lambda compute images:

Environment type	Runtime version	Instance size			
linux-lambda	dotnet6 go1.21 corretto11	1GB 2GB 4GB 8GB			
arm-lambda					

Environment type	Runtime version	Instance size			
	corretto17 corretto21 nodejs18 nodejs20 python3.11 python3.12 ruby3.2	10GB			

For more information, see Build environment compute modes and types and Docker images provided by CodeBuild.

Self-managed Buildkite runner in AWS CodeBuild

You can configure your project to set up self-hosted Buildkite runners in CodeBuild containers to process your Buildkite jobs. This can be done by setting up a webhook using your CodeBuild project, and updating your Buildkite pipeline YAML steps to use self-hosted runners hosted on CodeBuild machines.

The high-level steps to configure a CodeBuild project to run Buildkite jobs are as follows:

- Navigate to the CodeBuild console and create a CodeBuild project with the Buildkite runner project runner type configuration
- Add a `job.scheduled` webhook to your Buildkite organization.
- Update your Buildkite pipeline YAML steps in Buildkite to configure your build environment.

For a more detailed procedure, see Tutorial: Configure a CodeBuild-hosted Buildkite runner. This feature allows your Buildkite jobs to get native integration with AWS, which provides security and convenience through features like IAM, AWS Secrets Manager, AWS CloudTrail, and Amazon VPC. You can access the latest instance types, including ARM-based instances.

About the CodeBuild-hosted Buildkite runner

The following are some common questions about the CodeBuild-hosted Buildkite runner.

When should I include the image and instance overrides in the label?

You can include the image and instance overrides in the label in order to specify different build environment for each of your Buildkite jobs. This can be done without the need to create multiple CodeBuild projects or webhooks. For example, this is useful when you need to use a matrix for Buildkite jobs.

```
agents:
  queue: "myQueue"
steps:
  - command: "echo \"Hello World\""
    agents:
      project: "codebuild-myProject"
      image: "{{matrix.os}}"
      instance-size: "{{matrix.size}}"
    matrix:
      setup:
        os:
          - "arm-3.0"
          - "al2-5.0"
        size:
          - "small"
          - "large"
```

Can CodeBuild create webhooks within Buildkite automatically?

Currently, Buildkite requires that all webhooks are created manually using their console. You can follow the tutorial at Tutorial: Configure a CodeBuild-hosted Buildkite runner to create a Buildkite webhook manually in the Buildkite console.

Can I use AWS CloudFormation to create Buildkite webhooks?

AWS CloudFormation is not currently supported for Buildkite runner webhooks, as Buildkite requires webhooks to be created manually using their console.

Which regions support using a CodeBuild-hosted Buildkite runner?

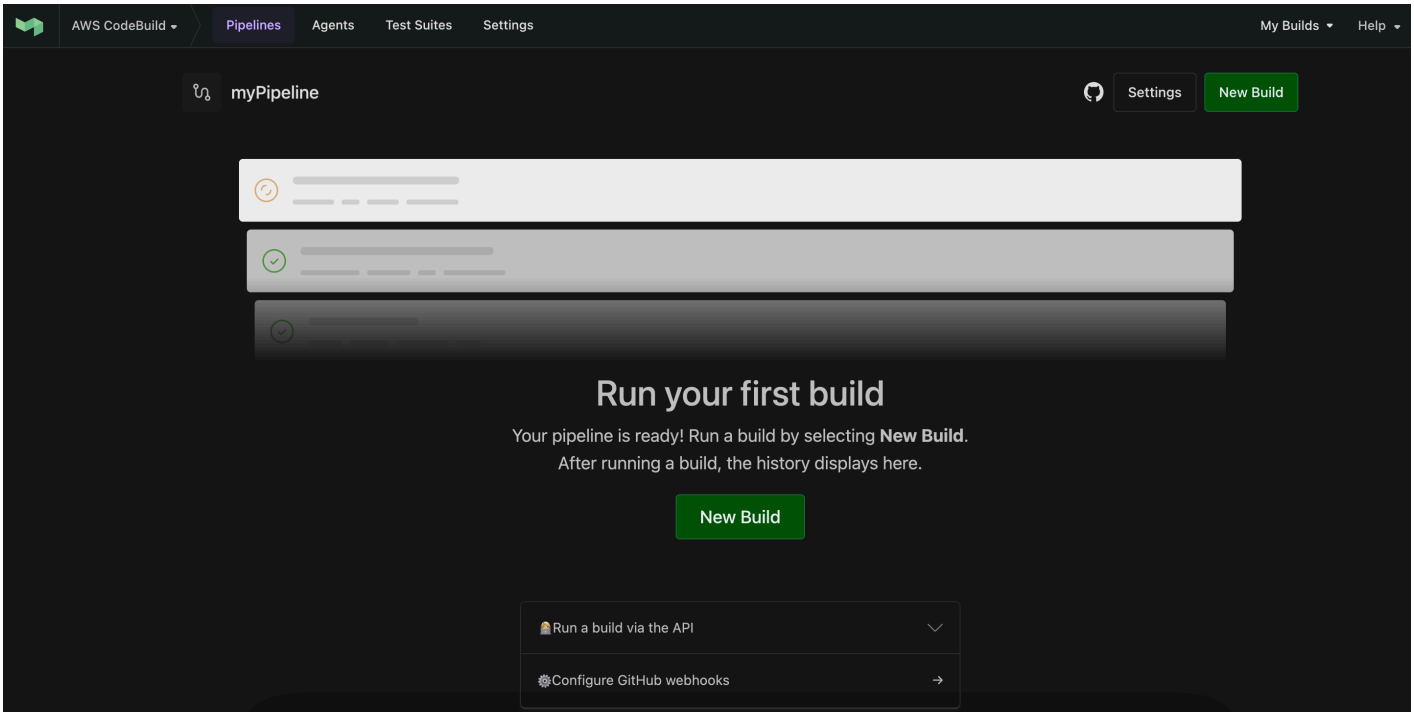
CodeBuild-hosted Buildkite runners are supported in all CodeBuild regions. For more information about AWS Regions where CodeBuild is available, see AWS Services by Region.

Tutorial: Configure a CodeBuild-hosted Buildkite runner

This tutorial shows you how to configure your CodeBuild projects to run Buildkite jobs. For more information about using Buildkite with CodeBuild see Self-managed Buildkite runner in AWS CodeBuild.

To complete this tutorial, you must first:

- Have access to a Buildkite organization. For more information about setting up a Buildkite account and organization, you can follow this Getting Started Tutorial.
- Create a Buildkite pipeline, cluster, and queue configured to use self-hosted runners. For more information about setting up these resources, you can reference the Buildkite Pipeline Setup Tutorial.

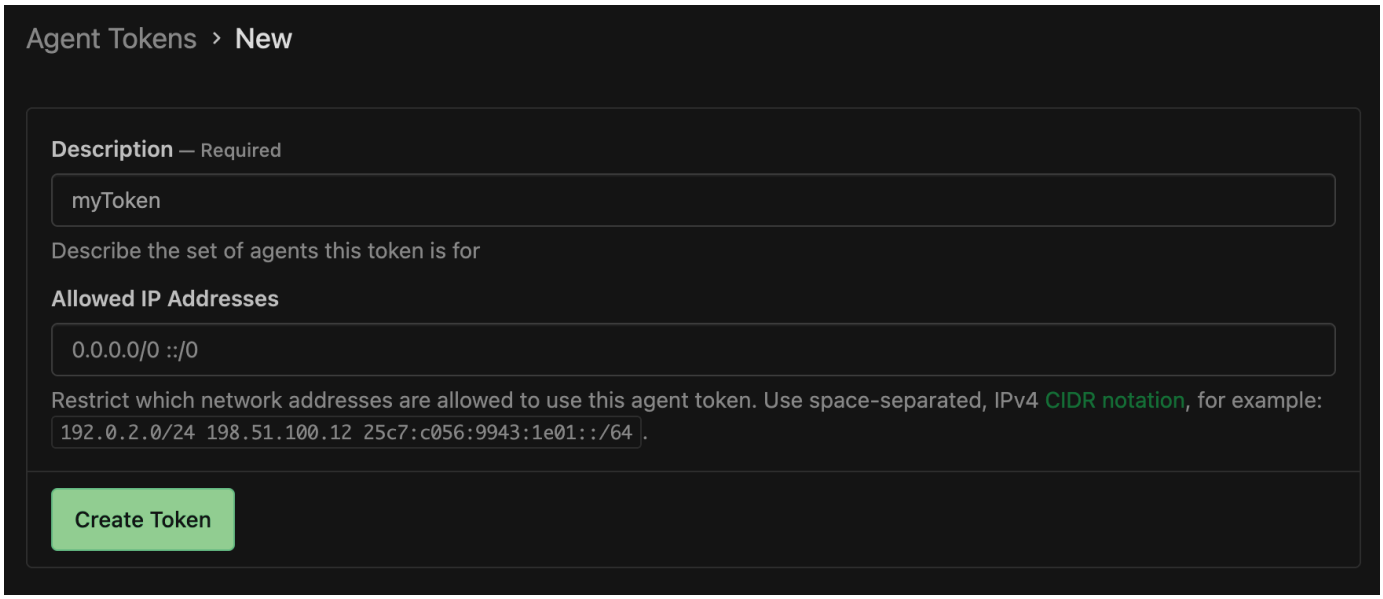


Step 1: Generate a Buildkite agent token

In this step, you will generate an agent token within Buildkite that will be used to authenticate the CodeBuild self-hosted runners. For more information about this resource, see Buildkite Agent Tokens.

To generate a Buildkite agent token

1. In your Buildkite cluster, choose **Agent Tokens**, and then choose **New Token**.
2. Add a description to the token and click **Create Token**.
3. Save the agent token value, as it will be used later during the CodeBuild project setup.



Step 2: Create a CodeBuild project with a webhook

To create a CodeBuild project with a webhook

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a self-hosted build project. For information, see Create a build project (console) and Run a build (console).
   - In **Project configuration**, select **Runner project**. In **Runner**:
     - For **Runner provider**, choose **Buildkite**.

- For **Buildkite agent token**, choose **Create a new agent token by using the create secret page**. You will be prompted to create a new secret in AWS Secrets Manager with a secret value equal to the Buildkite agent token you generated above.
- (Optional) If you would like to use CodeBuild managed credentials for your job, select your job's source repository provider under **Buildkite source credential options** and verify that credentials are configured for your account. Additionally, verify that your Buildkite pipeline uses **Checkout using HTTPS**.

> **Note**
>
> Buildkite requires source credentials within the build environment in order to pull your job's source. See Authenticating Buildkite to a Private Repository for available source credential options.

- (Optional) In **Environment**:
  - Choose a supported **Environment image** and **Compute**.

    Note that you have the option to override the image and instance settings by using a label in your Buildkite YAML steps. For more information, see Step 4: Update your Buildkite pipeline steps.
- (Optional) In **Buildspec**:
  - Your buildspec will be ignored by default unless `buildspec-override: "true"` is added as a label. Instead, CodeBuild will override it to use commands that will set up the self-hosted runner.

> **Note**
>
> CodeBuild does not support buildspec files for Buildkite self-hosted runner builds. For inline buildspecs, you will need to enable git-credential-helper in your buildspec if you have configured CodeBuild managed source credentials

3. Continue with the default values and then choose **Create build project**.
4. Save the **Payload URL** and **Secret** values from the **Create Webhook** popup. Either follow the instructions in the popup to create a new Buildkite organization webhook or continue to the next section.

Step 3: Create a CodeBuild webhook within Buildkite

In this step, you will use the **Payload URL** and **Secret** values from the CodeBuild webhook to create a new webhook within Buildkite. This webhook will be used to trigger builds within CodeBuild when a valid Buildkite job starts.

To create a new webhook in Buildkite

1. Navigate to your Buildkite organization's **Settings** page.
2. Under **Integrations**, select **Notification Services**.
3. Choose **Add** next to the **Webhook** box. In the **Add Webhook Notification** page, use the following configuration:

    a. Under **Webhook URL**, add the saved **Payload URL** value.

    b. Under **Token**, verify that **Send the token as X-Buildkite-Token** is selected. Add your webhook **Secret** value to the **Token** field.

    c. Under , verify that **Send the token as X-Buildkite-Token** is selected. Add your webhook **Secret** value to the **Token** field.

    d. Under **Events**, select the `job.scheduled` webhook event.

    e. (Optional) Under **Pipelines**, you can optionally choose to only trigger builds for a specific pipeline.
4. Choose **Add Webhook Notification**.

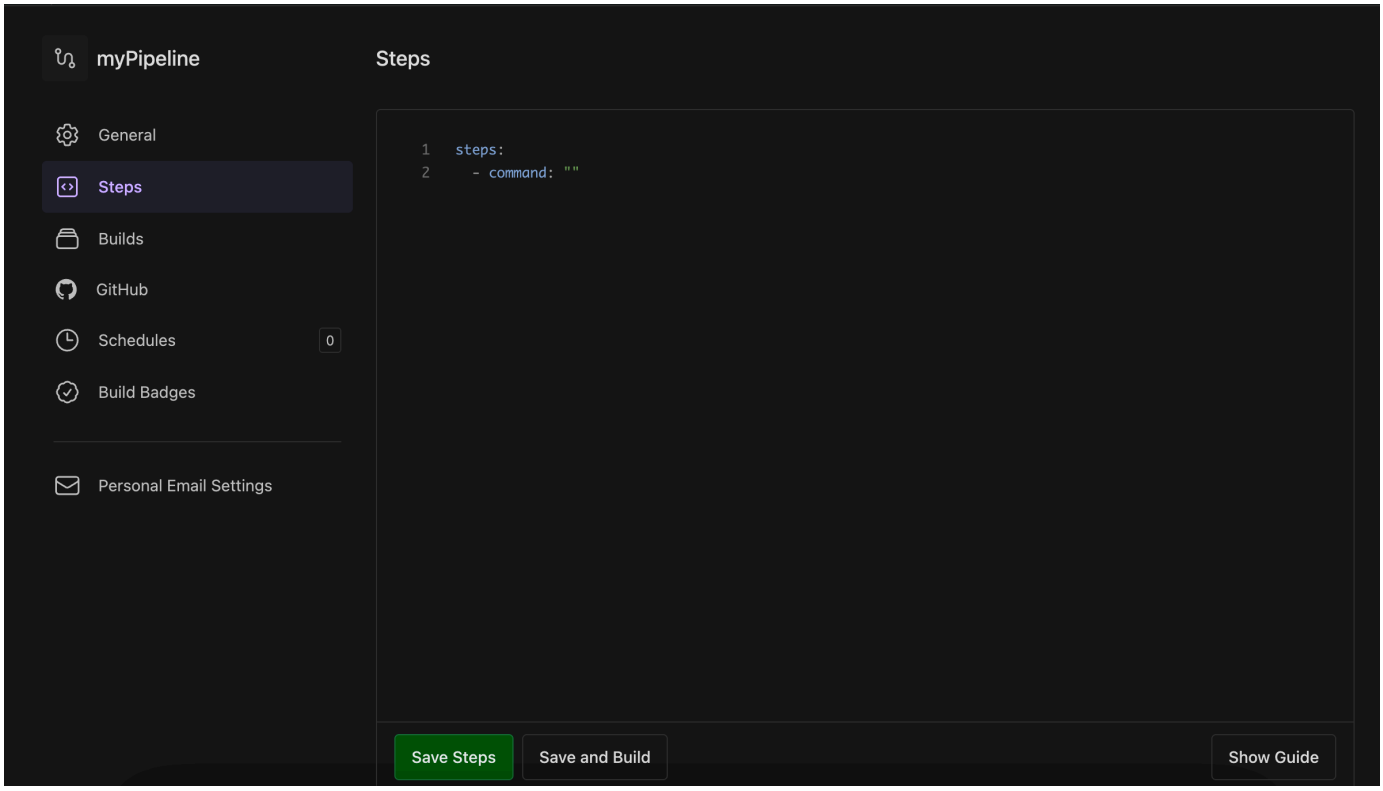
Step 4: Update your Buildkite pipeline steps

In this step, you will update your Buildkite pipeline's steps in order to add necessary labels and optional overrides. For the full list of supported label overrides, see Label overrides supported with the CodeBuild-hosted Buildkite runner.

Update your pipeline steps

1. Navigate to your Buildkite pipeline steps page by selecting your Buildkite pipeline, choosing **Settings**, and then choosing **Steps**.

    If you haven't already, choose **Convert to YAML steps**.



2. At a minimum, you will need to specify a Buildkite agent tag referencing the name of your CodeBuild pipeline. The project name is needed to link the AWS-related settings of your Buildkite job to a specific CodeBuild project. By including the project name in the YAML, CodeBuild is allowed to invoke jobs with the correct project settings.

```
agents:
  project: "codebuild-<project name>"
```

The following is an example of Buildkite pipeline steps with just the project label tag:

```
agents:
  project: "codebuild-myProject"
steps:
  - command: "echo \"Hello World\""
```

You can also override your image and compute type in the label. See Compute images supported with the CodeBuild-hosted Buildkite runner for a list of available images. The compute type and image in the label will override the environment settings on your project. To override your environment settings for a CodeBuild EC2 or Lambda compute build, use the following syntax:

```
agents:
  project: "codebuild-<project name>"
  image: "<environment-type>-<image-identifier>"
  instance-size: "<instance-size>"
```

The following is an example of Buildkite pipeline steps with image and instance size overrides:

```
agents:
  project: "codebuild-myProject"
  image: "arm-3.0"
  instance-size: "small"
steps:
  - command: "echo \"Hello World\""
```

You can override the fleet used for your build in the label. This will override the fleet settings configured on your project to use the specified fleet. For more information, see Run builds on reserved capacity fleets .

To override your fleet settings for an Amazon EC2 compute build, use the following syntax:

```
agents:
  project: "codebuild-<project name>"
  fleet: "<fleet-name>"
```

To override both the fleet and the image used for the build, use the following syntax:

```
agents:
  project: "codebuild-<project name>"
  fleet: "<fleet-name>"
  image: "<environment-type>-<image-identifier>"
```

The following is an example of Buildkite pipeline steps with fleet and image overrides:

```
agents:
  project: "codebuild-myProject"
  fleet: "myFleet"
  image: "arm-3.0"
steps:
  - command: "echo \"Hello World\""
```

3. You can choose to run inline buildspec commands during the self-hosted Buildkite runner build (see Run buildspec commands for the INSTALL, PRE_BUILD, and POST_BUILD phases for more details). To specify that the CodeBuild build should run buildspec commands during your Buildkite self-hosted runner build, use the following syntax:

```
agents:
  project: "codebuild-<project name>"
  buildspec-override: "true"
```

   The following is an example of a Buildkite pipeline with a buildspec override:

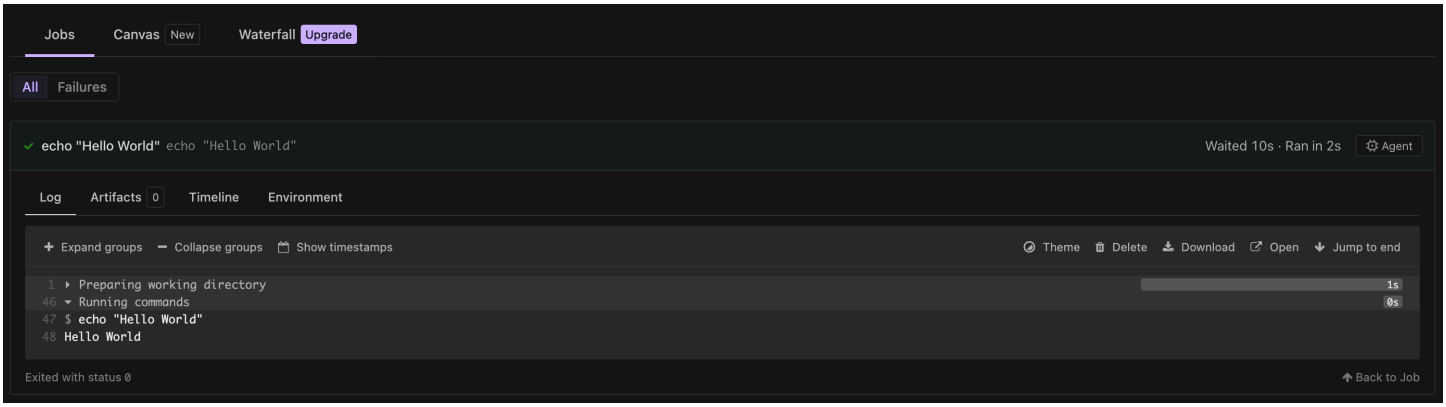
```
agents:
  project: "codebuild-myProject"
  buildspec-override: "true"
steps:
  - command: "echo \"Hello World\""
```

4. Optionally, you can provide labels outside of those that CodeBuild supports. These labels will be ignored for the purpose of overriding attributes of the build, but will not fail the webhook request. For example, adding `myLabel: “testLabel"` as a label will not prevent the build from running.

Step 5: Review your results

Whenever a Buildkite job is started in your pipeline, CodeBuild will receive a `job.scheduled` webhook event through the Buildkite webhook. For each job in your Buildkite build, CodeBuild will start a build to run an ephemeral Buildkite runner. The runner is responsible for executing a single Buildkite job. Once the job is completed, the runner and the associated build process will be immediately terminated.

To view your workflow job logs, navigate to your Buildkite pipeline and select the most recent build (you can trigger a new build by choosing **New Build**). Once the associated CodeBuild build for each of your jobs starts and picks up the job, you should see logs for the job within the Buildkite console



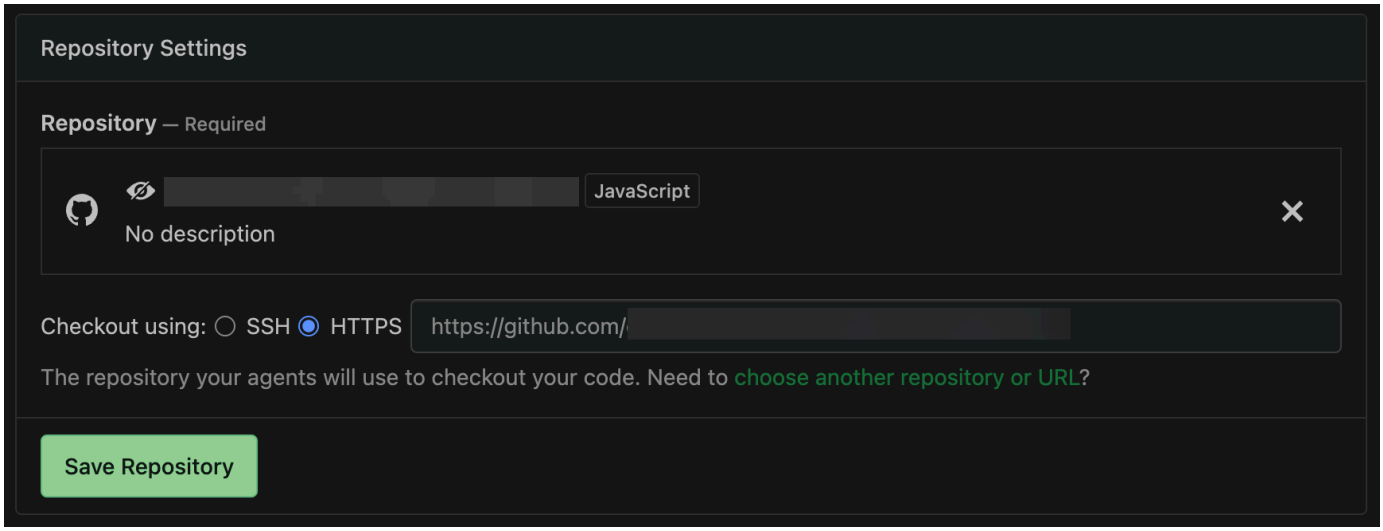
Authenticating Buildkite to a Private Repository

If you have a private repository configured within your Buildkite pipeline, Buildkite requires additional permissions within the build environment to pull the repository, as Buildkite does not vend credentials to self-hosted runners to pull from private repositories. In order to authenticate the Buildkite self-hosted runner agent to your external private source repository, you can use one of the following options.

To authenticate with CodeBuild

CodeBuild offers managed credentials handling for Supported source types. In order to use CodeBuild source credentials to pull your job's source repository, you can use the following steps:

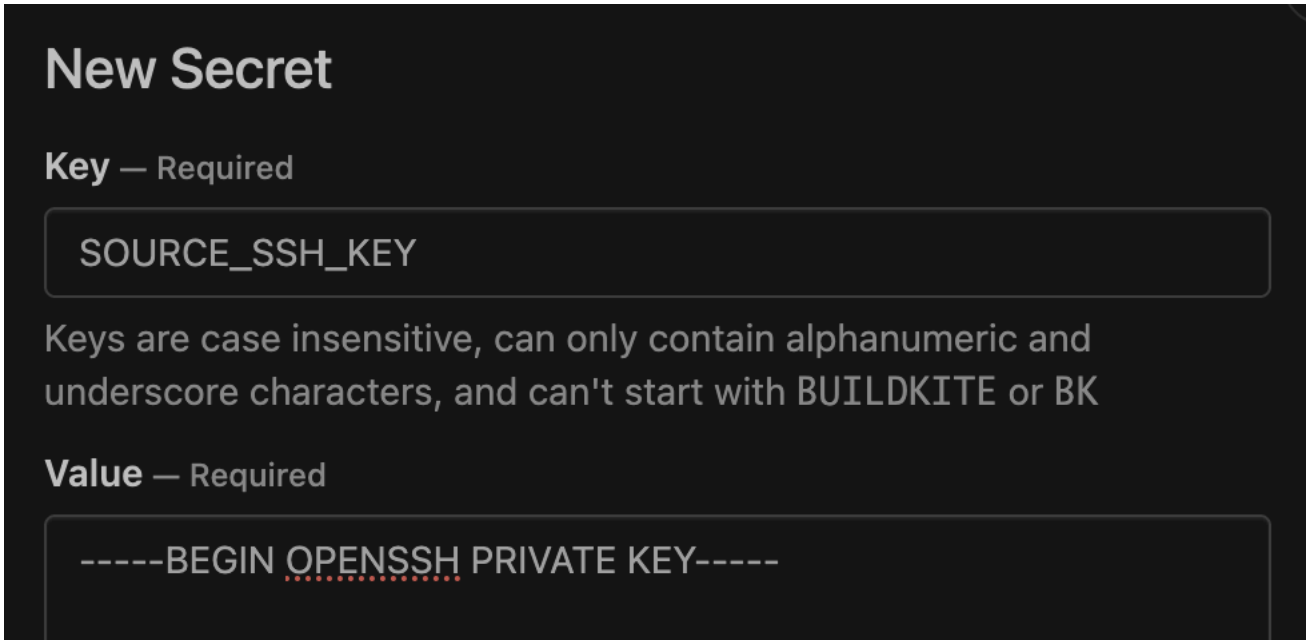
1. In the CodeBuild console, navigate to **Edit project** or create a new CodeBuild project using the steps in Step 2: Create a CodeBuild project with a webhook.
2. Under **Buildkite source credential options**, select your job's source repository provider.
    1. If you would like to use account-level CodeBuild credentials, verify that they are configured correctly. Additionally, if your project has an inline buildspec configured, verify that git-credential-helper is enabled.
    2. If you would like to use project level CodeBuild credentials, select **Use override credentials for this project only** and set up credentials for your project.
3. In your Buildkite pipeline settings, navigate to **Repository Settings**. Set your source repository checkout settings to **Checkout using HTTPS**



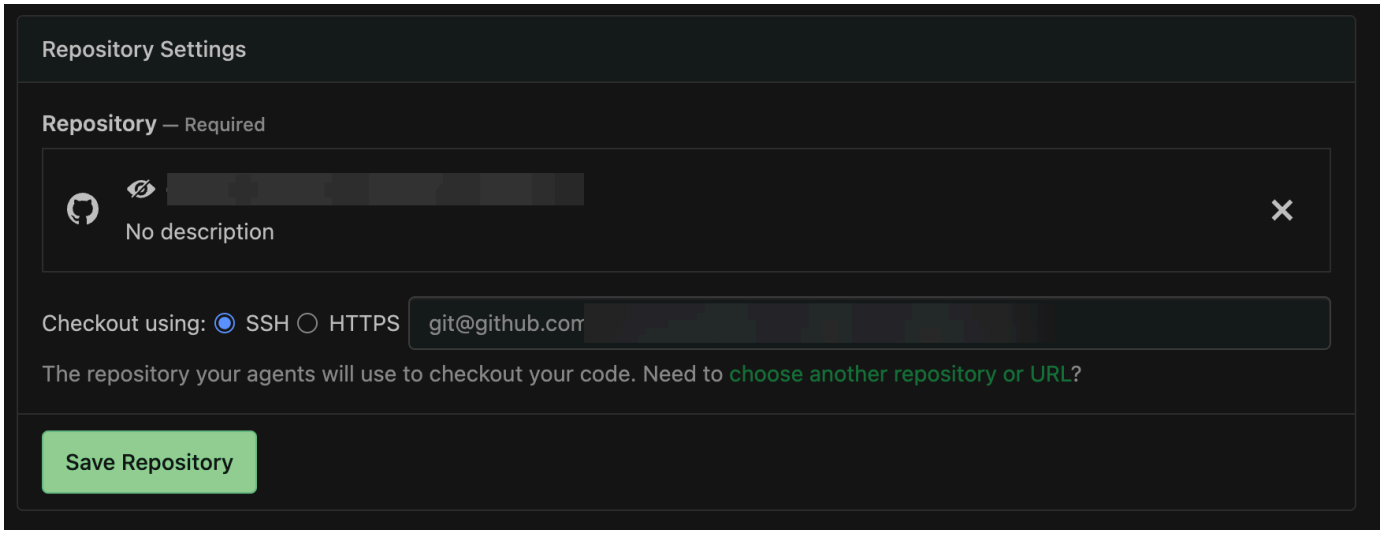
To authenticate with Buildkite secrets

Buildkite maintains an ssh-checkout plugin which can be used to authenticate the self-hosted runner to an external source repository using an ssh key. The key value is stored as a Buildkite secret and fetched automatically by the Buildkite self-hosted runner agent when attempting to pull a private repository. In order to configure the ssh-checkout plugin for your Buildkite pipeline, you can use the following steps:

1. Generate a private and public ssh key using your email address e.g. `ssh-keygen -t rsa -b 4096 -C "myEmail@address.com"`
2. Add the public key to your private source repository. For example, you can follow this guide to add a key to a GitHub account.
3. Add a new SSH key secret to your Buildkite cluster. Within your Buildkite cluster, select **Secrets** → **New Secret**. Add a name for you secret in the **Key** field and add your private SSH key into the **Value** field:



4. Within your Buildkite pipeline, navigate to your repository settings and set checkout to use **SSH**.



5. Update your pipeline YAML steps to use the `git-ssh-checkout` plugin. For example, the following pipeline YAML file uses the checkout action with the above Buildkite secret key:

```
agents:
  project: "codebuild-myProject"
steps:
  - command: "npm run build"
    plugins:
      - git-ssh-checkout#v0.4.1:
          ssh-secret-key-name: 'SOURCE_SSH_KEY'
```

6. When running a Buildkite self-hosted runner job within CodeBuild, Buildkite will now automatically use your configured secret value when pulling your private repository

Runner configuration options

You can specify the following environment variables in your project configuration to modify the setup configuration of your self-hosted runners:

- `CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN`: CodeBuild will fetch the secret value configured as the value of this environment variable from AWS Secrets Manager in order to register the Buildkite self-hosted runner agent. This environment variable must be of type `SECRETS_MANAGER`, and the value should be the name of your secret in Secrets Manager. A Buildkite agent token environment variable is required for all Buildkite runner projects.
- `CODEBUILD_CONFIG_BUILDKITE_CREDENTIAL_DISABLE`: By default, CodeBuild will load account or project level source credentials into the build environment, as these credentials are used by the Buildkite agent to pull the job's source repository. To disable this behavior, you can add this environment variable to your project with the value set to `true`, which will prevent source credentials from being loaded into the build environment.

Run buildspec commands for the INSTALL, PRE_BUILD, and POST_BUILD phases

By default, CodeBuild ignores any buildspec commands when running a self-hosted Buildkite runner build. To run buildspec commands during the build,

```
buildspec-override: "true"
```

can be added as a suffix to the label:

```
agents:
  project: "codebuild-<project name>"
  buildspec-override: "true"
```

By using this command, CodeBuild will create a folder called `buildkite-runner` in the container's primary source folder. When the Buildkite runner starts during the `BUILD` phase, the runner will run in the `buildkite-runner` directory.

There are several limitations when using a buildspec override in a self-hosted Buildkite build:

- The Buildkite agent requires that source credentials exist within the build environment to pull the job's source repository. If you use CodeBuild source credentials for authentication, you will need to enable `git-credential-helper` in your buildspec. For example, you can use the following buildspec to enable `git-credential-helper` for your Buildkite builds:

```
version: 0.2
env:
  git-credential-helper: yes
phases:
  pre_build:
    commands:
       - echo "Hello World"
```

- CodeBuild will not run buildspec commands during the `BUILD` phase, as the self-hosted runner runs in the `BUILD` phase.
- CodeBuild does not support buildspec files for Buildkite runner builds. Only inline buildspecs are supported for Buildlkite self-hosted runners
- If a build command fails in the `PRE_BUILD` or `INSTALL` phase, CodeBuild will not start the self-hosted runner and the Buildkite job will need to be cancelled manually.

Setting up a Buildkite runner programmatically

In order to configure a Buildkite runner project programatically, you will need to configure the following resources:

To create a Buildkite runner programmatically

1. Create a Buildkite agent token and save the token in plaintext within AWS Secrets Manager.
2. Set up a CodeBuild project with your preferred configuration. You will need to configure the following additional attributes:

    1. An environment value with name `CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN`, type `SECRETS_MANAGER`, and a value equal to the Buildkite agent token associated with your Buildkite cluster.
    2. Source type equal to `NO_SOURCE`
    3. Permissions to access the secret created in step 1 in your project's service role

For example, you can use the following command to create a valid Buildkite runner project through the CLI:

```
aws codebuild create-project \
--name buildkite-runner-project \
--source "{\"type\": \"NO_SOURCE\",\"buildspec\":\"\"}" \
--environment "{\"image\":\"aws/codebuild/amazonlinux-x86_64-standard:5.0\",
\"type\":\"LINUX_CONTAINER\",\"computeType\":\"BUILD_GENERAL1_MEDIUM\",
\"environmentVariables\":[{\"name\":\"CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN\",
\"type\":\"SECRETS_MANAGER\",\"value\":\"<buildkite-secret-name>\"}]}" \
--artifacts "{\"type\": \"NO_ARTIFACTS\"}" \
--service-role <service-role>
```

3. Create a Buildkite runner webhook on the project created in step 2. You will need to use the following configuration options when creating the webhook:

   1. **build-type** should be equal to `RUNNER_BUILDKITE_BUILD`
   2. A filter with type `EVENT` and a pattern equal to `WORKFLOW_JOB_QUEUED`

   For example, you can use the following command to create a valid Buildkite runner webhook through the CLI:

   ```
   aws codebuild create-webhook \
   --project-name buildkite-runner-project \
   --filter-groups "[[{\"type\":\"EVENT\",\"pattern\":\"WORKFLOW_JOB_QUEUED\"}]]" \
   --build-type RUNNER_BUILDKITE_BUILD
   ```

4. Save the **Payload URL** and **Secret** values returned by the `create-webhook` call and use the credentials to create a webhook within the Buildkite console. You can reference Step 3: Create a CodeBuild webhook within Buildkite in Tutorial: Configure a CodeBuild-hosted Buildkite runner for a guide on how to set up this resource.

Troubleshoot the webhook for failed builds or a hanging job

Issue:

The webhook you set up in Tutorial: Configure a CodeBuild-hosted Buildkite runner isn't working or your workflow job is hanging in Buildkite.

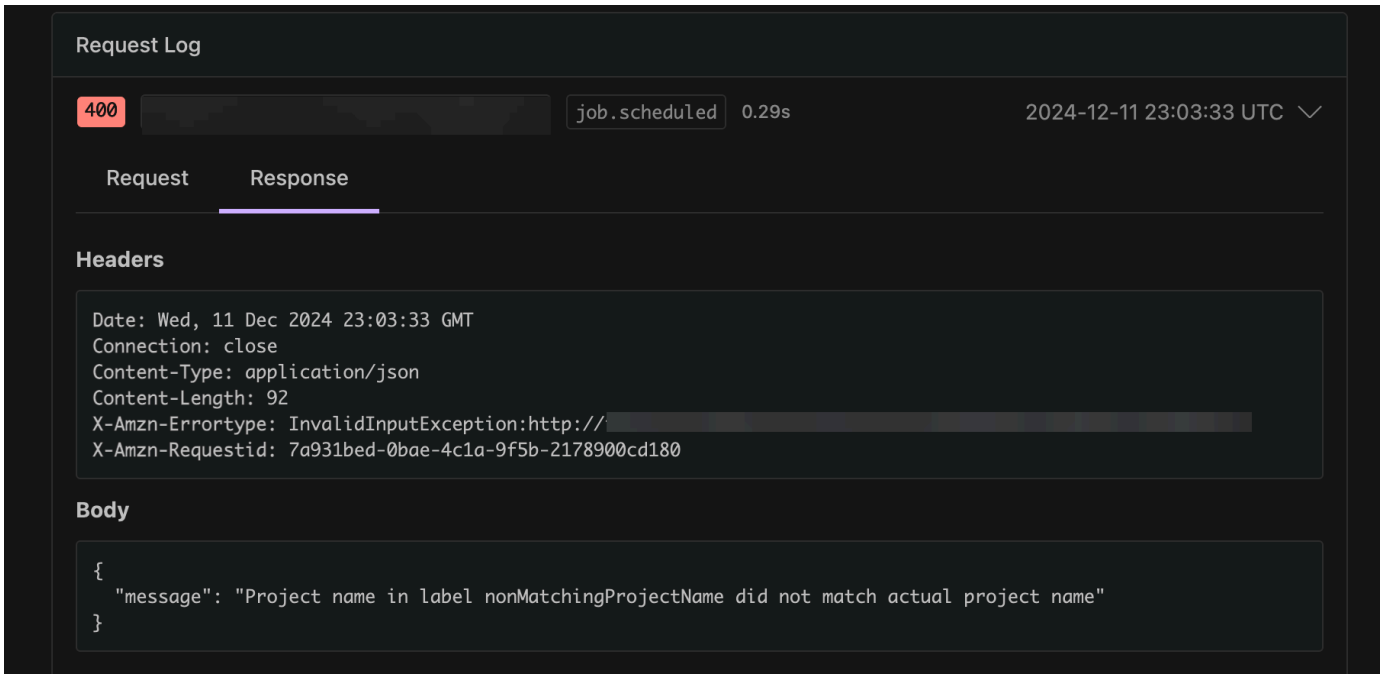
Possible causes:

- Your webhook **job.scheduled** event might be failing to trigger a build. Review the **Response** logs to view the response or error message.
- Your CodeBuild build fails before starting the Buildkite self-hosted runner agent to handle your job.

Recommended solutions:

To debug failed Buildkite webhook events:

1. In your Buildkite organization settings, navigate to **Notification Services**, select your CodeBuild webhook, and then find the **Request Log**.
2. Find the `job.scheduled` webhook event associated with your stuck Buildkite job. You can use the job ID field within the webhook payload to correlate the webhook event to your Buildkite job.
3. Select the **Response** tab and check the response body. Verify that the **Response** status code is `200` and the **Response** body doesn't contain any unexpected messages.



Troubleshoot the webhook permission issues

Issue:

The Buildkite job fails to checkout the job's source repository due to permission issues.

Possible causes:

- CodeBuild does not have sufficient permissions to checkout the job's source repository.
- The pipeline's repository settings are set to check out using SSH for CodeBuild managed credentials.

Recommended solutions:

- Verify that CodeBuild has sufficient permissions configured to check out the job's source repository. Additionally, verify that your CodeBuild project's service role has sufficient permissions to access the configured source permission option.
- Verify that your Buildkite pipeline is configured to use checkout using HTTPS if you are using CodeBuild managed source repository credentials.

Label overrides supported with the CodeBuild-hosted Buildkite runner

In your Buildkite pipeline steps agent tag labels, you can provide a variety of label overrides that modify your self-hosted runner build. Any builds not recognized by CodeBuild will be ignored but will not fail your webhook request. For example, the following workflow YAML includes overrides for image, instance size, fleet, and the buildspec:

```
agents:
  queue: "myQueue"
steps:
  - command: "echo \"Hello World\""
    agents:
      project: "codebuild-myProject"
      image: "{{matrix.os}}"
      instance-size: "{{matrix.size}}"
      buildspec-override: "true"
    matrix:
      setup:
        os:
          - "arm-3.0"
          - "al2-5.0"
        size:
          - "small"
```

```
          - "large"
```

`project:codebuild-<project-name>` (required)

- Example: `project: "codebuild-myProject"`
- Required for all Buildkite pipeline step configurations. *<project name>* should be equal to the name of the project for which the self-hosted runner webhook is configured.

`queue: "<queue-name>"`

- Example: `queue: "<queue-name>"`
- Used to route Buildkite jobs to a specific queue. See the Buildkite Agent Queue Tag for more information.

`image: "<environment-type>-<image-identifier>"`

- Example: `image: "arm-3.0"`
- Overrides the image and environment type used when starting the self-hosted runner build with a curated image. To learn about supported values, see Compute images supported with the CodeBuild-hosted Buildkite runner.
  1. To override the image and environment type used with a custom image, use `image: "custom-<environment-type>-<custom-image-identifier>"`
  2. Example:

```
image:
      "custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0"
```

> **Note**
>
> If the custom image resides in a private registry, you must configure the appropriate registry credentials in your CodeBuild project.

`instance-size: "<instance-size>"`

- Example: `instance-size: "medium"`

- Overrides the instance type used when starting the self-hosted runner build. To learn about supported values, see Compute images supported with the CodeBuild-hosted Buildkite runner.

`fleet: "<fleet-name>"`

- Example: `fleet: "myFleet"`
- Overrides the fleet settings configured on your project to use the specified fleet. For more information, see Run builds on reserved capacity fleets .

`buildspec-override: "<boolean>"`

- Example: `buildspec-override: "true"`
- Allows the build to run buildspec commands in the `INSTALL`, `PRE_BUILD`, and `POST_BUILD` phases if set to `true`.

Compute images supported with the CodeBuild-hosted Buildkite runner

In the label you configured in Self-managed Buildkite runner in AWS CodeBuild, you can override your Amazon EC2 environment settings by using the values in the first three columns. CodeBuild provides the following Amazon EC2 compute images. For more information about

Environment type	Image identifier	Instance size	Platform	Resolved image	Definition

`linux`	`4.0`	`small`
		`medium`
		`large`
		`xlarge`
			Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:4.0`	al/standard/4.0

`linux`	`5.0`	`2xlarge`
		`gpu_small`
		`gpu_large`
			Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-sta`	al/standard/5.0

Environment type	Image identifier	Instance size	Platform	Resolved image	Definition
				ndard:5.0	
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0	2xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0
ubuntu	5.0	small medium large xlarge 2xlarge gpu_small gpu_large	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0		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	N/A

Environment type	Image identifier	Instance size	Platform	Resolved image	Definition
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	N/A
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	N/A
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	N/A

In addition, you can override your Lambda environment settings by using the following values. For more information about CodeBuild Lambda compute, see Run builds on AWS Lambda compute. CodeBuild supports the following Lambda compute images:

Environment type	Image identifier	Instance size			
linux-lambda	dotnet6 go1.21 corretto11 corretto17 corretto21	1GB 2GB 4GB 8GB 10GB			
arm-lambda					

Environment type	Image identifier	Instance size			

	`nodejs18`
	
	`nodejs20`
	
	`python3.11`
	
	`python3.12`
	
	`ruby3.2`				

For more information, see Build environment compute modes and types and Docker images provided by CodeBuild.

Use webhooks with AWS CodeBuild

AWS CodeBuild supports webhook integration with GitHub, GitHub Enterprise Server, GitLab, GitLab Self Managed, and Bitbucket.

Topics

- Best practices for using webhooks with AWS CodeBuild
- Bitbucket webhook events
- GitHub global and organization webhooks
- GitHub manual webhooks
- GitHub webhook events
- GitLab group webhooks
- GitLab manual webhooks
- GitLab webhook events
- Buildkite manual webhooks

Best practices for using webhooks with AWS CodeBuild

For projects that use public repositories to setup webhooks, we recommend the following options:

Setup `ACTOR_ACCOUNT_ID` filters

Add `ACTOR_ACCOUNT_ID` filters to your project's webhook filter groups to specify which users can trigger a build. Every webhook event delivered to CodeBuild comes with sender information that specifies the actor's identifier. CodeBuild will filter the webhooks based on the regular expression pattern provided in the filters. You can specify the specific users that are allowed to trigger builds with this filter. For more information, see GitHub webhook events and Bitbucket webhook events.

Setup `FILE_PATH` filters

Add `FILE_PATH` filters to your project's webhook filter groups to include or exclude the files that can trigger a build when changed. For example, you can deny build requests for changes to the `buildspec.yml` file using a regular expression pattern such as `^buildspec.yml$`, along with the `excludeMatchedPattern` property. For more information, see GitHub webhook events and Bitbucket webhook events.

Scope down the permissions for your build IAM role

Builds triggered by a webhook use the IAM service role specified in the project. We recommend setting the permissions in the service role to the minimum set of permissions required to run the build. For example, in a test and deploy scenario, create one project for testing and another project for deployment. The testing project accepts webhook builds from the repository, but provides no write permissions to your resources. The deployment project provides write permissions to your resources, and the webhook filter is configured to only allow trusted users to trigger builds.

Use an inline or an Amazon S3 stored buildspec

If you define your buildspec inline within the project itself, or store the buildspec file in an Amazon S3 bucket, the buildspec file is only visible to the project owner. This prevents pull requests from making code changes to the buildspec file and triggering unwanted builds. For more information, see ProjectSource.buildspec in the *CodeBuild API Reference*.

Bitbucket webhook events

You can use webhook filter groups to specify which Bitbucket webhook events trigger a build. For example, you can specify that a build is only triggered for changes to specific branches.

You can create one or more webhook filter groups to specify which webhook events trigger a build. A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true. When you create a filter group, you specify:

An event

For Bitbucket, you can choose one or more of the following events:

- PUSH
- PULL_REQUEST_CREATED
- PULL_REQUEST_UPDATED
- PULL_REQUEST_MERGED
- PULL_REQUEST_CLOSED

The webhook's event type is in its header in the `X-Event-Key` field. The following table shows how `X-Event-Key` header values map to the event types.

> **Note**
>
> You must enable the `merged` event in your Bitbucket webhook setting if you create a webhook filter group that uses the `PULL_REQUEST_MERGED` event type. You must also enable the `declined` event in your Bitbucket webhook setting if you create a webhook filter group that uses the `PULL_REQUEST_CLOSED` event type.

X-Event-Key Header value	Event type
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED

X-Event-Key Header value	Event type
pullrequest:rejected	PULL_REQUEST_CLOSED

For PULL_REQUEST_MERGED, if a pull request is merged with the squash strategy and the pull request branch is closed, the original pull request commit no longer exists. In this case, the CODEBUILD_WEBHOOK_MERGE_COMMIT environment variable contains the identifier of the squashed merge commit.

One or more optional filters

Use a regular expression to specify a filter. For an event to trigger a build, every filter within the group associated with it must evaluate to true.

ACTOR_ACCOUNT_ID (ACTOR_ID in the console)

A webhook event triggers a build when a Bitbucket account ID matches the regular expression pattern. This value appears in the account_id property of the actor object in the webhook filter payload.

HEAD_REF

A webhook event triggers a build when the head reference matches the regular expression pattern (for example, refs/heads/branch-name and refs/tags/tag-name). A HEAD_REF filter evaluates the Git reference name for the branch or tag. The branch or tag name appears in the name field of the new object in the push object of the webhook payload. For pull request events, the branch name appears in the name field in the branch object of the source object in the webhook payload.

BASE_REF

A webhook event triggers a build when the base reference matches the regular expression pattern. A BASE_REF filter works with pull request events only (for example, refs/heads/branch-name). A BASE_REF filter evaluates the Git reference name for the branch. The branch name appears in the name field of the branch object in the destination object in the webhook payload.

FILE_PATH

A webhook triggers a build when the path of a changed file matches the regular expression pattern.

`COMMIT_MESSAGE`

A webhook triggers a build when the head commit message matches the regular expression pattern.

`WORKFLOW_NAME`

A webhook triggers a build when the workflow name matches the regular expression pattern.

> **Note**
>
> You can find the webhook payload in the webhook settings of your Bitbucket repository.

Topics

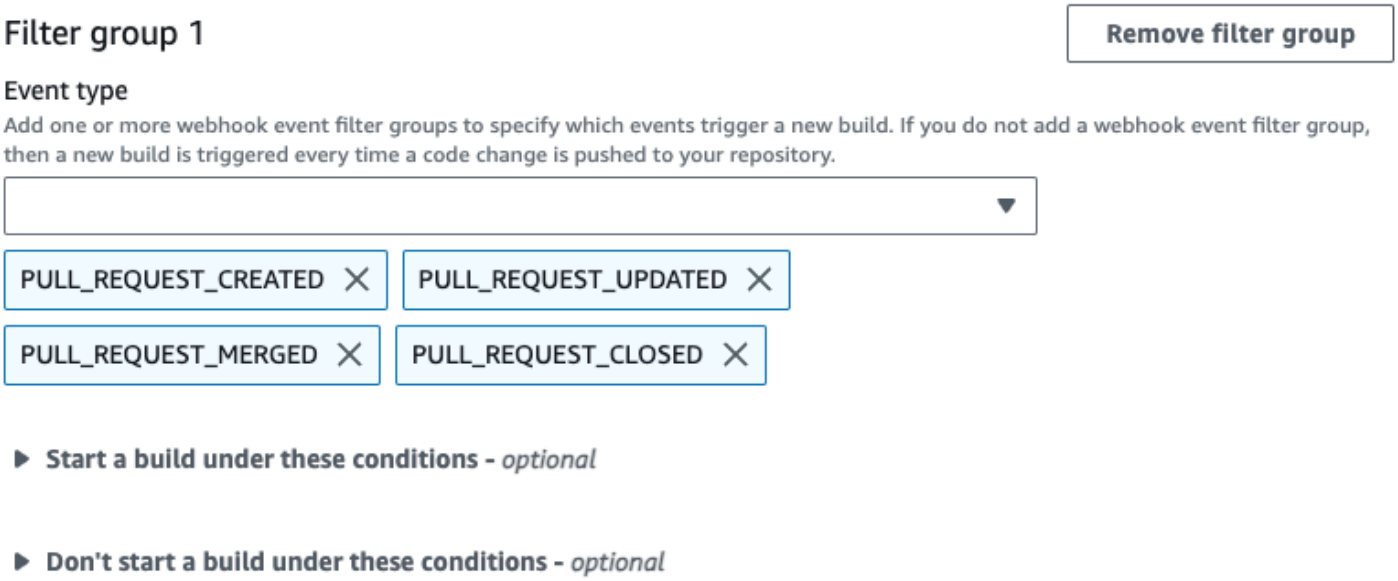
Filter Bitbucket webhook events (console)

To use the AWS Management Console to filter webhook events:

1. Select **Rebuild every time a code change is pushed to this repository** when you create your project.
2. From **Event type**, choose one or more events.
3. To filter when an event triggers a build, under **Start a build under these conditions**, add one or more optional filters.
4. To filter when an event is not triggered, under **Don't start a build under these conditions**, add one or more optional filters.
5. Choose **Add filter group** to add another filter group.

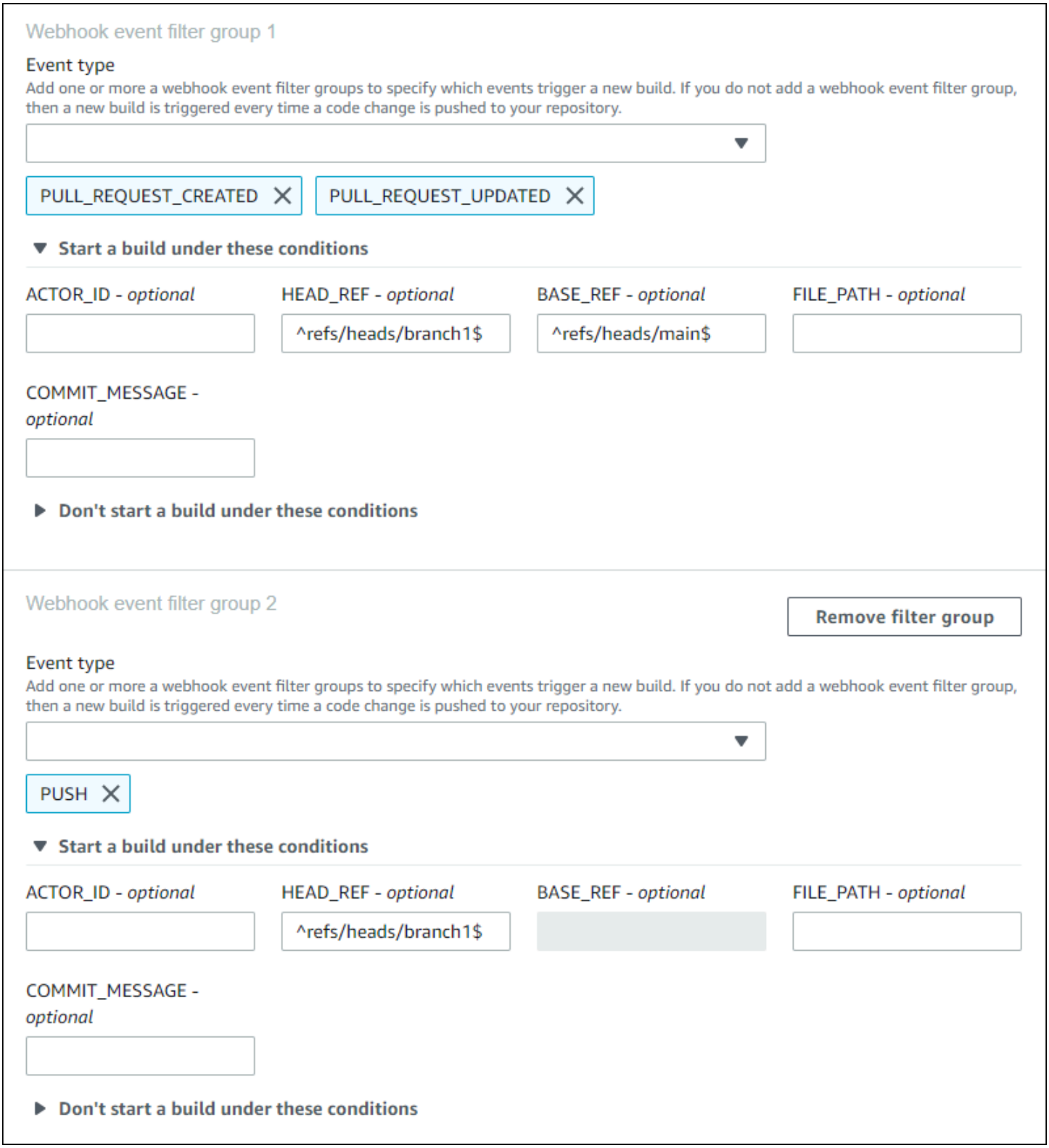
For more information, see Create a build project (console) and WebhookFilter in the *AWS CodeBuild API Reference*.

In this example, a webhook filter group triggers a build for pull requests only:

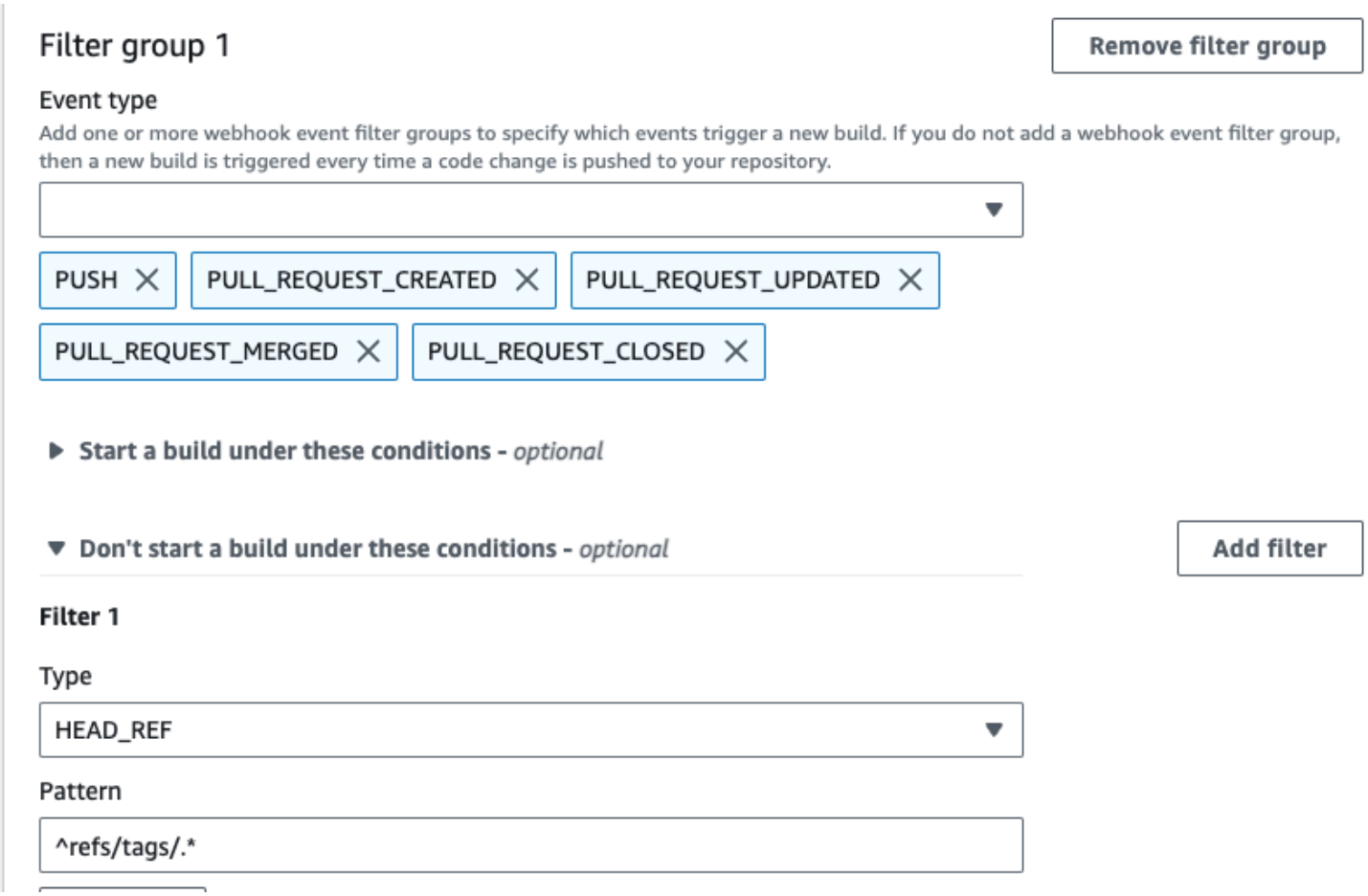


Using an example of two filter groups, a build is triggered when one or both evaluate to true:

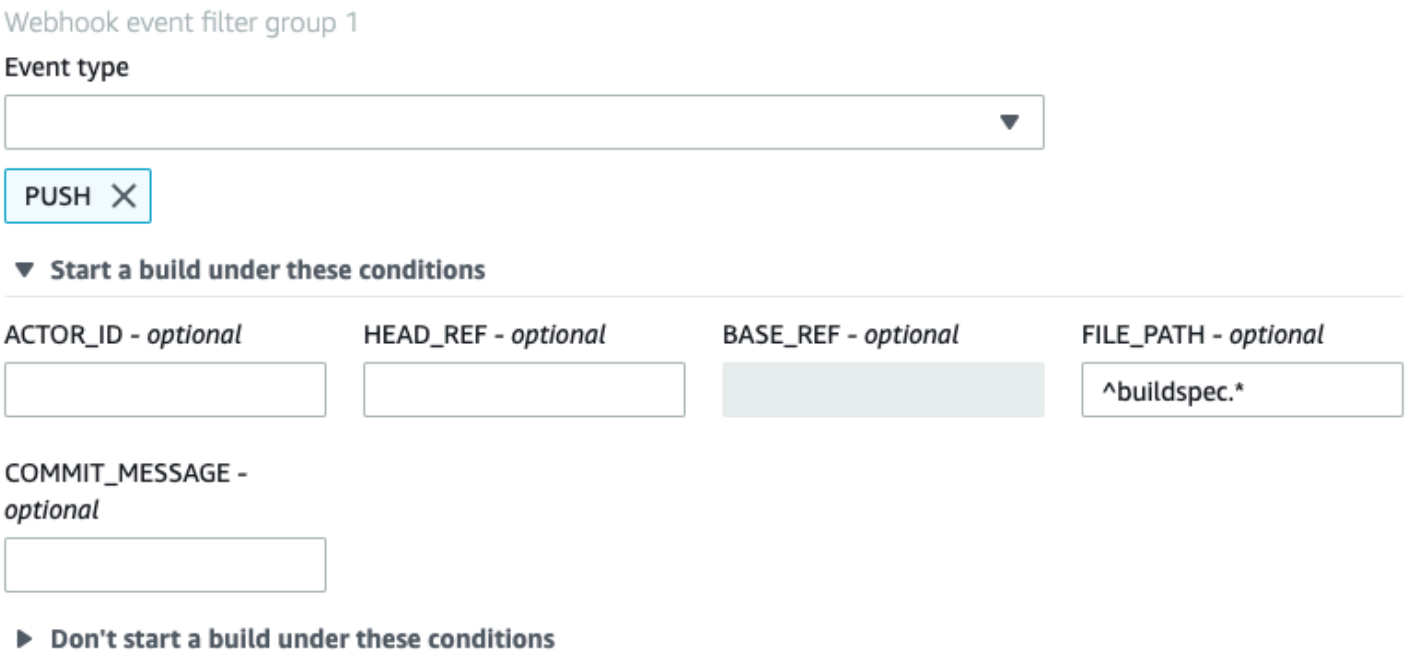
- The first filter group specifies pull requests that are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` and head references that match `^refs/heads/branch1!`.
- The second filter group specifies push requests on branches with Git reference names that match the regular expression `^refs/heads/branch1$`.



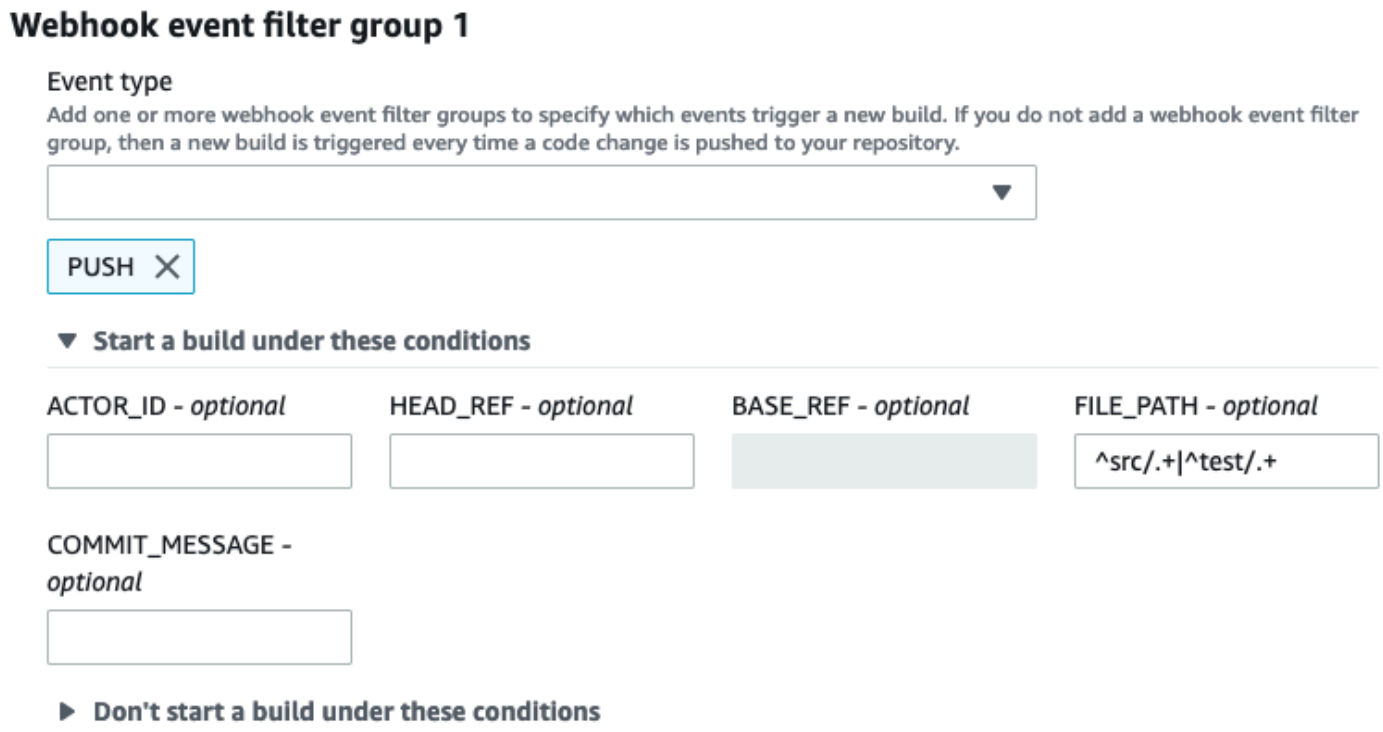
In this example, a webhook filter group triggers a build for all requests except tag events.



In this example, a webhook filter group triggers a build only when files with names that match the regular expression `^buildspec.*` change.

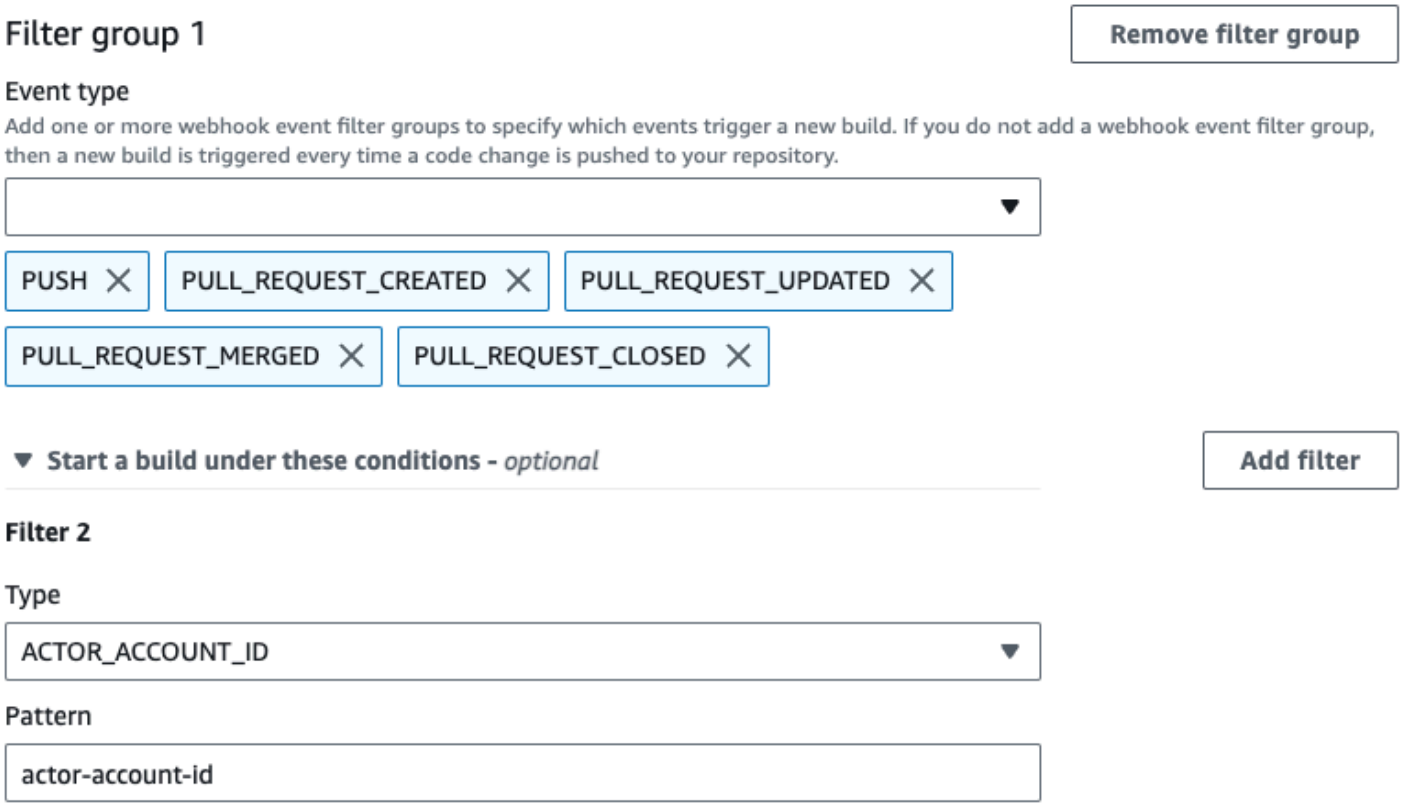


In this example, a webhook filter group triggers a build only when files are changed in `src` or `test` folders.

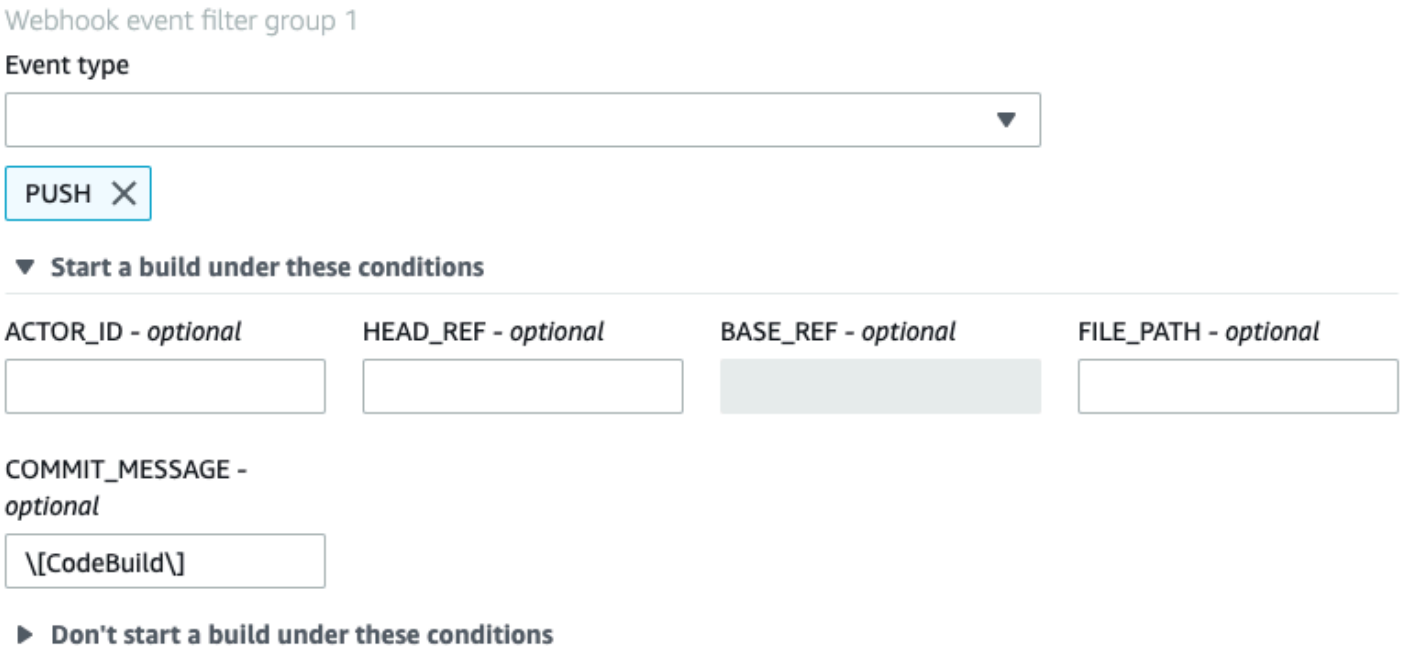


In this example, a webhook filter group triggers a build only when a change is made by a Bitbucket user who does not have an account ID that matches the regular expression `actor-account-id`.

> **Note**
>
> For information about how to find your Bitbucket account ID, see https://api.bitbucket.org/2.0/users/*user-name*, where *user-name* is your Bitbucket user name.



In this example, a webhook filter group triggers a build for a push event when the head commit message matches the regular expression `\[CodeBuild\]`.



Filter Bitbucket webhook events (SDK)

To use the AWS CodeBuild SDK to filter webhook events, use the `filterGroups` field in the request syntax of the `CreateWebhook` or `UpdateWebhook` API methods. For more information, see WebhookFilter in the *CodeBuild API Reference*.

To create a webhook filter that triggers a build for pull requests only, insert the following into the request syntax:

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED,
 PULL_REQUEST_CLOSED"
    }
  ]
]
```

To create a webhook filter that triggers a build for specified branches only, use the `pattern` parameter to specify a regular expression to filter branch names. Using an example of two filter groups, a build is triggered when one or both evaluate to true:

- The first filter group specifies pull requests that are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` and head references that match `^refs/heads/myBranch$`.
- The second filter group specifies push requests on branches with Git reference names that match the regular expression `^refs/heads/myBranch$`.

```
"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$"
    }
  ]
]
```

You can use the `excludeMatchedPattern` parameter to specify which events do not trigger a build. In this example, a build is triggered for all requests except tag events.

```
"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
    }
  ]
]
```

You can create a filter that triggers a build only when a change is made by a Bitbucket user with account ID `actor-account-id`.

> **Note**
>
> For information about how to find your Bitbucket account ID, see https://api.bitbucket.org/2.0/users/*user-name*, where *user-name* is your Bitbucket user name.

```
"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"
    }
  ]
]
```

You can create a filter that triggers a build only when files with names that match the regular expression in the `pattern` argument change. In this example, the filter group specifies that a build is triggered only when files with a name that matches the regular expression `^buildspec.*` change.

```
"filterGroups": [
```

```
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^buildspec.*"
    }
  ]
]
```

In this example, the filter group specifies that a build is triggered only when files are changed in `src` or `test` folders.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

You can create a filter that triggers a build only when the head commit message matches the regular expression in the pattern argument. In this example, the filter group specifies that a build is triggered only when the head commit message of the push event matches the regular expression `\[CodeBuild\]`.

```
  "filterGroups": [
    [
      {
        "type": "EVENT",
        "pattern": "PUSH"
      },
      {
        "type": "COMMIT_MESSAGE",
        "pattern": "\[CodeBuild\]"
```

```
      }
    ]
  ]
```

Filter Bitbucket webhook events (AWS CloudFormation)

To use an AWS CloudFormation template to filter webhook events, use the AWS CodeBuild project's `FilterGroups` property. The following YAML-formatted portion of an AWS CloudFormation template creates two filter groups. Together, they trigger a build when one or both evaluate to true:

- The first filter group specifies pull requests are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` by a Bitbucket user who does not have account ID `12345`.
- The second filter group specifies push requests are created on branches with Git reference names that match the regular expression `^refs/heads/.*`.
- The third filter group specifies a push request with a head commit message matching the regular expression `\[CodeBuild\]`.

```
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: 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 global and organization webhooks

You can use CodeBuild GitHub global or organization webhooks to start builds on webhook events from any repository within a GitHub organization or enterprise. Global and organization webhooks work with any of the existing GitHub webhook event types, and can be configured by adding a scope configuration when creating a CodeBuild webhook. You can also use global and organization webhooks to set up self-hosted GitHub Action runners within CodeBuild in order to receive `WORKFLOW_JOB_QUEUED` events from multiple repositories within a single project.

Topics

- Set up a global or organization GitHub webhook
- Filter GitHub global or organization webhook events (console)
- Filter GitHub organization webhook events (AWS CloudFormation)

Set up a global or organization GitHub webhook

The high-level steps to set up a global or organization GitHub webhook are as follows. For more information about global and organization GitHub webhooks, see GitHub global and organization webhooks.

1. Set your project's source location to `CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION`.

2. In the webhook's scope configuration, set the scope to either `GITHUB_ORGANIZATION` or `GITHUB_GLOBAL` depending on whether it should be an organization or global webhook. For more information, see Types of webhooks.
3. Specify a name as part of the webhook's scope configuration. For organization webhooks, this is the organization name, and for global webhooks this is the enterprise name.

> **Note**
>
> If the project's source type is `GITHUB_ENTERPRISE`, you will also need to specify a domain as part of the webhook scope configuration.

4. (Optional) If you would only like to receive webhook events for specific repositories within your organization or enterprise, you can specify `REPOSITORY_NAME` as a filter when creating the webhook.
5. If you are creating an organization webhook, ensure that CodeBuild has permissions to create organization level webhooks within GitHub. You can create a GitHub personal access token with organization webhook permissions, or use CodeBuild OAuth. For more information, see GitHub and GitHub Enterprise Server access token.

   Note that organization webhooks work with any of the existing GitHub webhook event types.
6. If you are creating a global webhook, the webhook will need to be created manually. For more information about how to manually create a webhook within GitHub, see GitHub manual webhooks.

   Note that global webhooks only support the `WORKFLOW_JOB_QUEUED` event type. For more information, see Tutorial: Configure a CodeBuild-hosted GitHub Actions runner.

Filter GitHub global or organization webhook events (console)

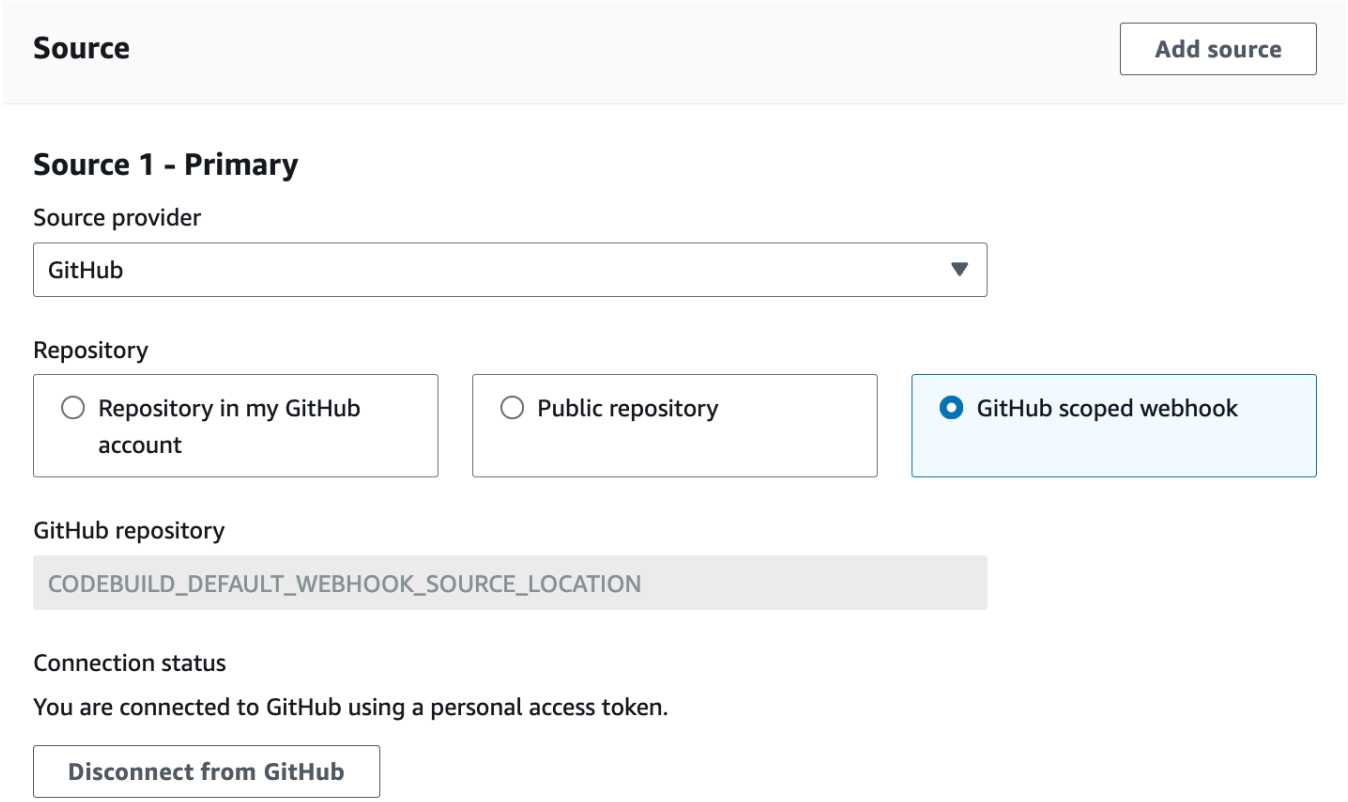
When creating a GitHub project through the console, select the following options to create a GitHub global or organization webhook within the project. For more information about global and organization GitHub webhooks, see GitHub global and organization webhooks.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a build project. For information, see Create a build project (console) and Run a build (console).

- In **Source**:
  - For **Source provider**, choose **GitHub** or **GitHub Enterprise**.
  - For **Repository**, choose **GitHub scoped webhook**.

    The GitHub repository will automatically be set to `CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION`, which is the required source location for global and organization webhooks.

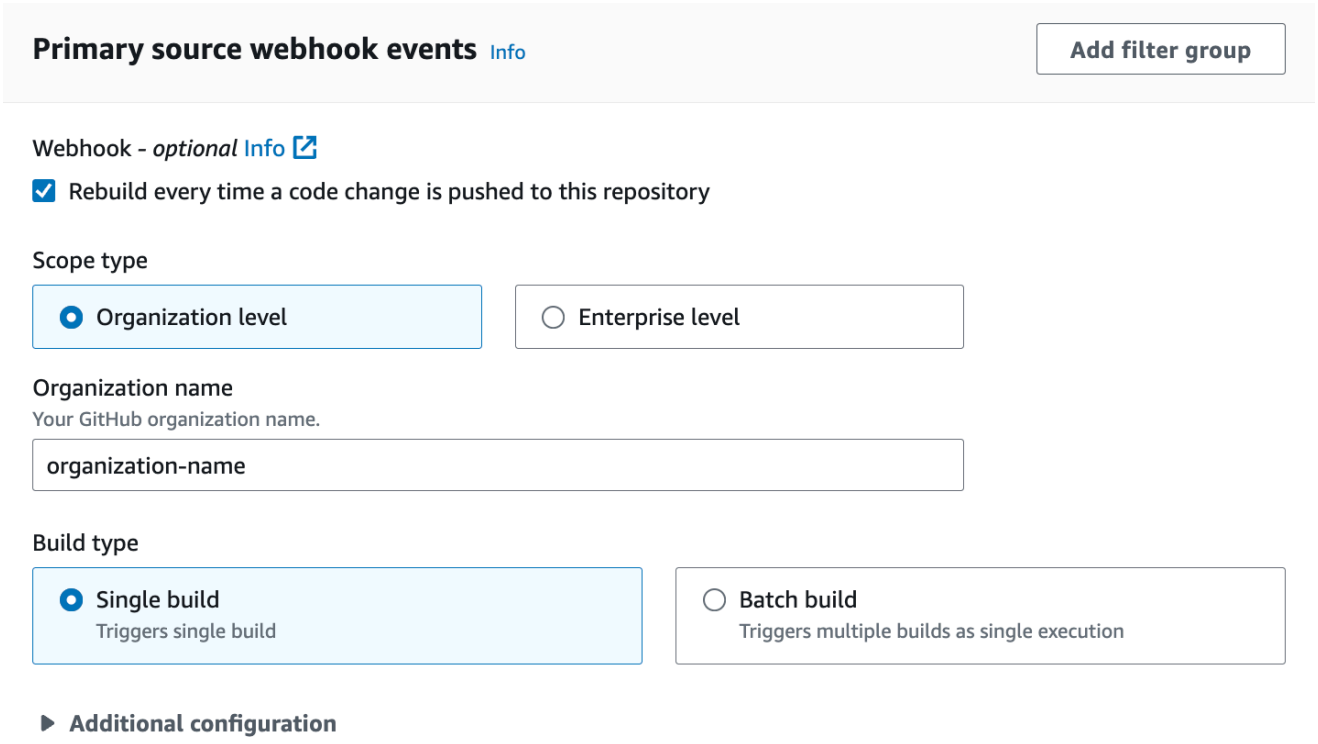
    > **Note**
    >
    > If you are using organization webhooks, make sure that CodeBuild has permissions to create organization level webhooks within GitHub. If you're using an existing OAuth connection, you may need to regenerate the connection in order to grant CodeBuild this permission. Alternatively, you can create the webhook manually using the CodeBuild manual webhooks feature. Note that if you have an existing GitHub OAuth token and would like to add additional organization permissions, you can revoke the OAuth token's permission and reconnect the token through the CodeBuild console.



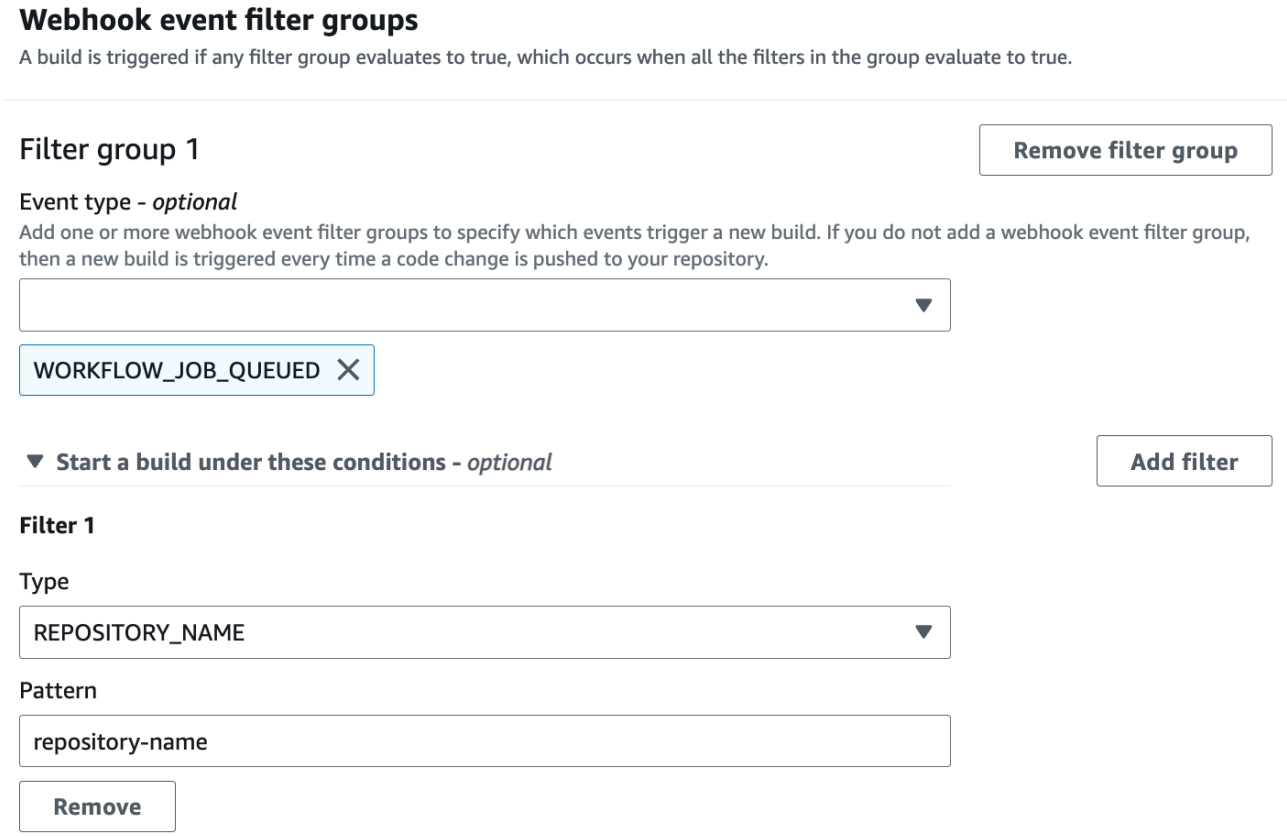
- In **Primary source webhook events**:
  - For **Scope type**, choose **Organization level** if you're creating an organization webhook or **Enterprise level** if you're creating a global webhook.
  - For **Name**, enter either the enterprise or organization name, depending on if the webhook is a global or organization webhook.

    If the project's source type is `GITHUB_ENTERPRISE`, you also need to specify a domain as part of the webhook organization configuration. For example, if the URL of your organization is **`https://domain.com/orgs/org-name`**, then the domain is **`https://domain.com`**.

    > **Note**
    >
    > This name cannot be changed after the webhook has been created. To change the name, you can delete and re-create the webhook. If you want to remove the webhook entirely, you can also update the project source location to a GitHub repository.



- (Optional) In **Webhook event filter groups**, you can specify which events you would like to trigger a new build. You can also specify `REPOSITORY_NAME` as a filter to only trigger builds on webhook events from specific repositories.



  You can also set the event type to `WORKFLOW_JOB_QUEUED` to set up self-hosted GitHub Actions runners. For more information, see Tutorial: Configure a CodeBuild-hosted GitHub Actions runner.

3. Continue with the default values and then choose **Create build project**.

Filter GitHub organization webhook events (AWS CloudFormation)

To use an AWS CloudFormation template to filter organization webhook events, use the AWS CodeBuild project's `ScopeConfiguration` property. For more information about global and organization GitHub webhooks, see GitHub global and organization webhooks.

> **Note**
>
> Global webhooks and GitHub Enterprise webhooks are not supported by AWS CloudFormation.

The following YAML-formatted portion of an AWS CloudFormation template creates four filter groups. Together, they trigger a build when one or all evaluate to true:

- The first filter group specifies pull requests are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` by a GitHub user who does not have account ID `12345`.
- The second filter group specifies push requests are created on files with names that match the regular expression `READ_ME` in branches with Git reference names that match the regular expression `^refs/heads/.*`.
- The third filter group specifies a push request with a head commit message matching the regular expression `\[CodeBuild\]`.
- The fourth filter group specifies a GitHub Actions workflow job request with a workflow name matching the regular expression `\[CI-CodeBuild\]`.

```
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
      ScopeConfiguration:
        Name: organization-name
        Scope: GITHUB_ORGANIZATION
```

```
      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: \[CI-CodeBuild\]
```

GitHub manual webhooks

You can configure manual GitHub webhooks to prevent CodeBuild from automatically attempting to create a webhook within GitHub. CodeBuild returns a payload URL in as part of the call to create the webhook and can be used to manually create the webhook within GitHub. Even if CodeBuild is not allowlisted to create a webhook in your GitHub account, you can still manually create a webhook for your build project.

Use the following procedure to create a GitHub manual webhook.

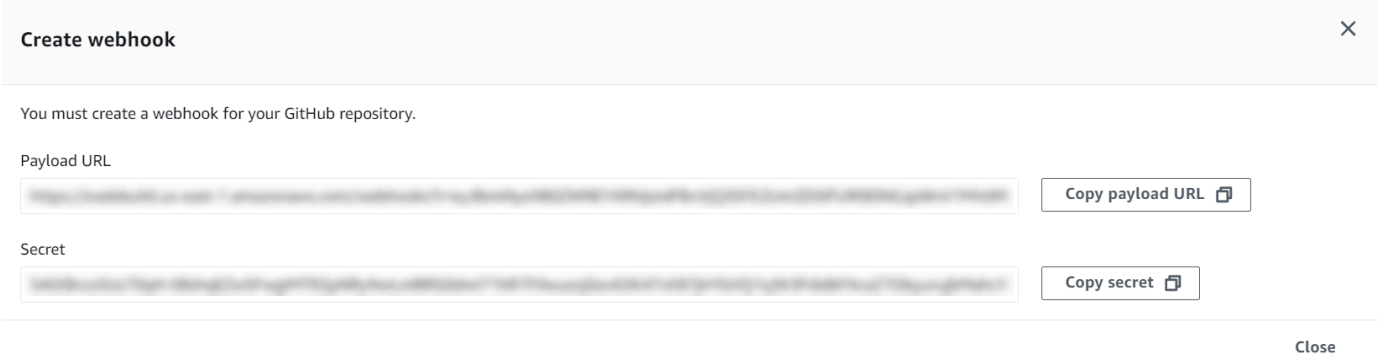
To create a GitHub manual webhook

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.

2. Create a build project. For information, see Create a build project (console) and Run a build (console).

   - In **Source**:
     - For **Source provider**, choose **GitHub**.
     - For **Repository**, choose **Repository in my GitHub account**.
     - For **Repository URL**, enter **`https://github.com/user-name/repository-name`**.
   - In **Primary source webhook events**:
     - For **Webhook - optional**, choose **Rebuild every time a code change is pushed to this repository**.
     - Choose **Additional configuration** and for **Manual creation - optional**, choose **Manually create a webhook for this repository in GitHub console.**.

3. Continue with the default values and then choose **Create build project**. Take note of the **Payload URL** and **Secret** values as you will use these later.



4. Open the GitHub console at `https://github.com/user-name/repository-name/settings/hooks` and choose **Add webhook**.

   - For **Payload URL**, enter the Payload URL value you took note of earlier.
   - For **Content type**, choose **application/json**.
   - For **Secret**, enter the Secret value you took note of earlier.
   - Configure the individual events that will send a webhook payload to CodeBuild. For **Which events would you like to trigger this webhook?**, choose **Let me select individual events**, and then choose from the following events: **Pushes**, **Pull requests**, and **Releases**. If you want to start builds for `WORKFLOW_JOB_QUEUED` events, choose **Workflow jobs**. To learn more about GitHub Actions runners, see Tutorial: Configure a CodeBuild-hosted GitHub Actions runner. To learn more about event types supported by CodeBuild, see GitHub webhook events.

5. Choose **Add webhook**.

GitHub webhook events

You can use webhook filter groups to specify which GitHub webhook events trigger a build. For example, you can specify that a build is only triggered for changes to specific branches.

You can create one or more webhook filter groups to specify which webhook events trigger a build. A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true. When you create a filter group, you specify:

An event

For GitHub, you can choose one or more of the following events: `PUSH`, `PULL_REQUEST_CREATED`, `PULL_REQUEST_UPDATED`, `PULL_REQUEST_REOPENED`, `PULL_REQUEST_MERGED`, `PULL_REQUEST_CLOSED`, `RELEASED`, `PRERELEASED`, and `WORKFLOW_JOB_QUEUED`. The webhook event type is in the `X-GitHub-Event` header in the webhook payload. In the `X-GitHub-Event` header, you might see `pull_request` or `push`. For a pull request event, the type is in the `action` field of the webhook event payload. The following table shows how `X-GitHub-Event` header values and webhook pull request payload `action` field values map to the available event types.

`X-GitHub-Event` Header value	**Webhook event payload `action` value**	**Event type**
`pull_request`	`opened`	`PULL_REQUEST_CREATED`
`pull_request`	`reopened`	`PULL_REQUEST_REOPENED`
`pull_request`	`synchronize`	`PULL_REQUEST_UPDATED`
`pull_request`	`closed`, and the `merged` field is `true`	`PULL_REQUEST_MERGED`
`pull_request`	`closed`, and the `merged` field is `false`	`PULL_REQUEST_CLOSED`
`push`	n/a	`PUSH`

X-GitHub-Event Header value	Webhook event payload action value	Event type
release	released	RELEASED
release	prereleased	PRERELEASED
workflow_job	queued	WORKFLOW_JOB_QUEUED

> **Note**
>
> The PULL_REQUEST_REOPENED event type can be used with GitHub and GitHub Enterprise Server only. The RELEASED and PRERELEASED event type can be used with GitHub only. For more information on WORKFLOW_JOB_QUEUED, see Tutorial: Configure a CodeBuild-hosted GitHub Actions runner.

One or more optional filters

Use a regular expression to specify a filter. For an event to trigger a build, every filter within the group associated with it must evaluate to true.

ACTOR_ACCOUNT_ID (ACTOR_ID in the console)

A webhook event triggers a build when a GitHub or GitHub Enterprise Server account ID matches the regular expression pattern. This value is found in the id property of the sender object in the webhook payload.

HEAD_REF

A webhook event triggers a build when the head reference matches the regular expression pattern (for example, refs/heads/branch-name or refs/tags/tag-name). For a push event, the reference name is found in the ref property in the webhook payload. For pull requests events, the branch name is found in the ref property of the head object in the webhook payload.

BASE_REF

A webhook event triggers a build when the base reference matches the regular expression pattern (for example, refs/heads/branch-name). A BASE_REF filter can be used with pull

request events only. The branch name is found in the `ref` property of the `base` object in the webhook payload.

`FILE_PATH`

A webhook triggers a build when the path of a changed file matches the regular expressions pattern. A `FILE_PATH` filter can be used with GitHub push and pull request events and GitHub Enterprise Server push events. It cannot be used with GitHub Enterprise Server pull request events.

`COMMIT_MESSAGE`

A webhook triggers a build when the head commit message matches the regular expression pattern. A `COMMIT_MESSAGE` filter can be used with GitHub push and pull request events and GitHub Enterprise Server push events. It cannot be used with GitHub Enterprise Server pull request events.

`TAG_NAME`

A webhook triggers a build when the tag name of the release matches the regular expression pattern. A `TAG_NAME` filter can be used with GitHub released and prereleased request events.

`RELEASE_NAME`

A webhook triggers a build when the release name matches the regular expression pattern. A `RELEASE_NAME` filter can be used with GitHub released and prereleased request events.

`REPOSITORY_NAME`

A webhook triggers a build when the repository name matches the regular expression pattern. A `REPOSITORY_NAME` filter can only be used with GitHub global or organization webhooks.

`ORGANIZATION_NAME`

A webhook triggers a build when the organization name matches the regular expression pattern. A `ORGANIZATION_NAME` filter can only be used with GitHub global webhooks.

`WORKFLOW_NAME`

A webhook triggers a build when the workflow name matches the regular expression pattern. A `WORKFLOW_NAME` filter can be used with GitHub Actions workflow job queued request events.

> **Note**
>
> You can find the webhook payload in the webhook settings of your GitHub repository.

Topics

Filter GitHub webhook events (console)

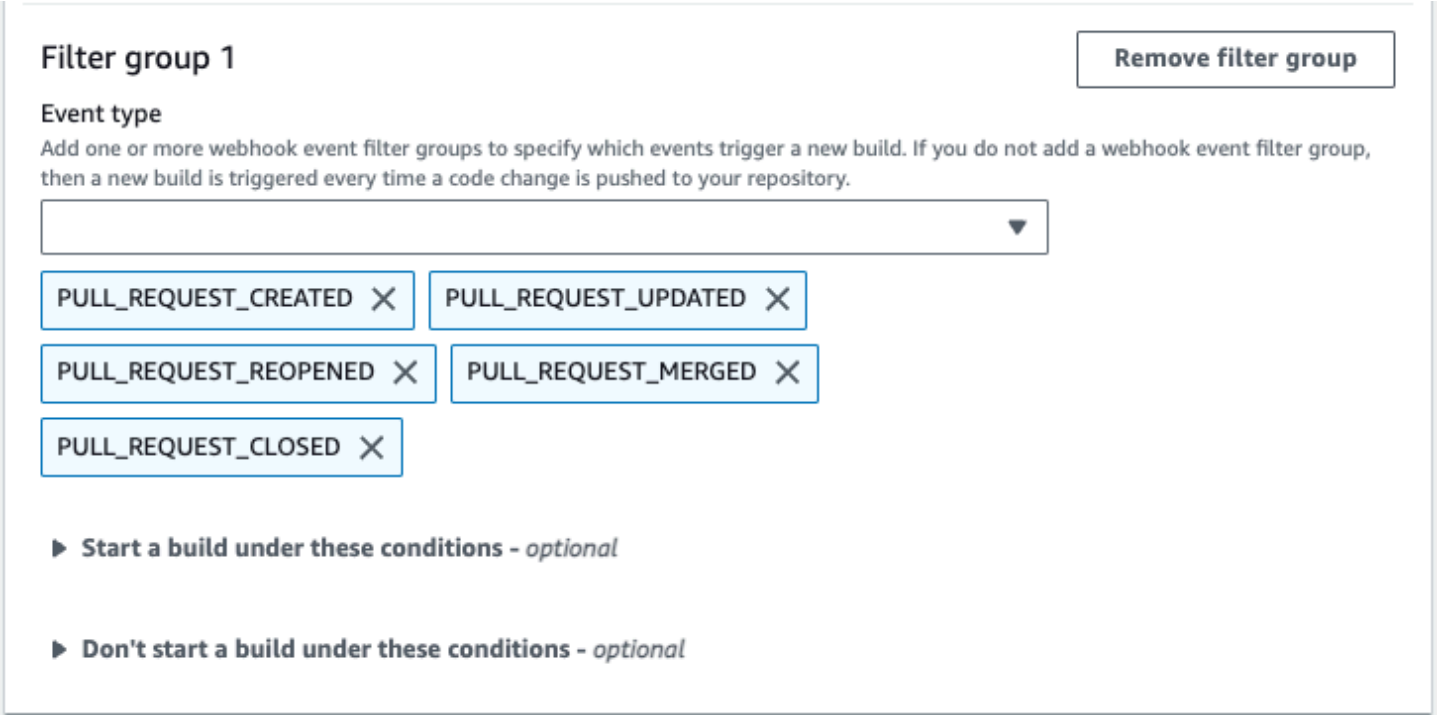
Use the following instructions to filter GitHub webhook events using the AWS Management Console. For more information about GitHub webhook events, see GitHub webhook events.

In **Primary source webhook events**, select the following. This section is only available when you chose **Repository in my GitHub account** for the source repository.

1. Select **Rebuild every time a code change is pushed to this repository** when you create your project.
2. From **Event type**, choose one or more events.
3. To filter when an event triggers a build, under **Start a build under these conditions**, add one or more optional filters.
4. To filter when an event is not triggered, under **Don't start a build under these conditions**, add one or more optional filters.
5. Choose **Add filter group** to add another filter group, if needed.

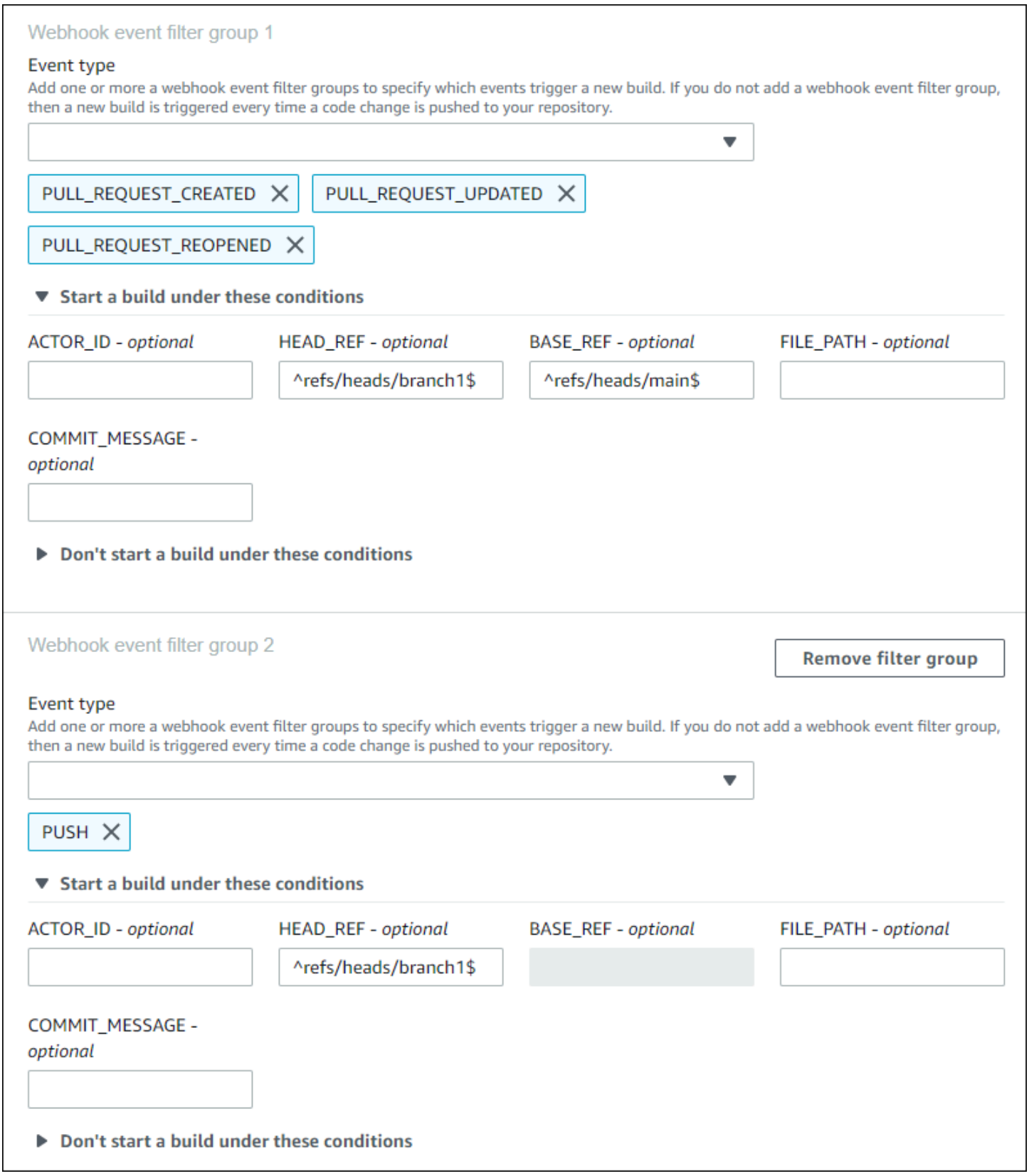
For more information, see Create a build project (console) and WebhookFilter in the *AWS CodeBuild API Reference*.

In this example, a webhook filter group triggers a build for pull requests only:

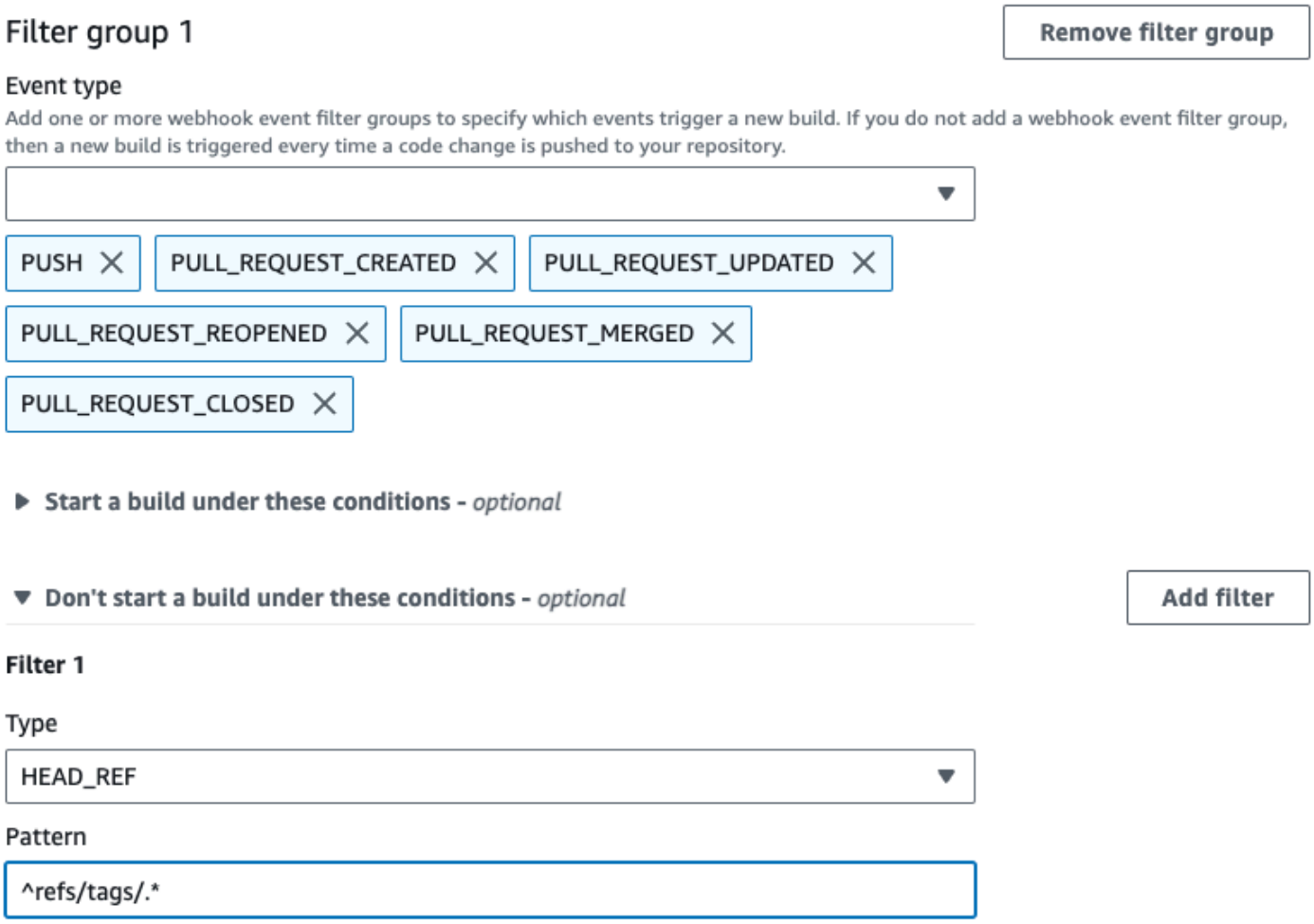


Using an example of two webhook filter groups, a build is triggered when one or both evaluate to true:

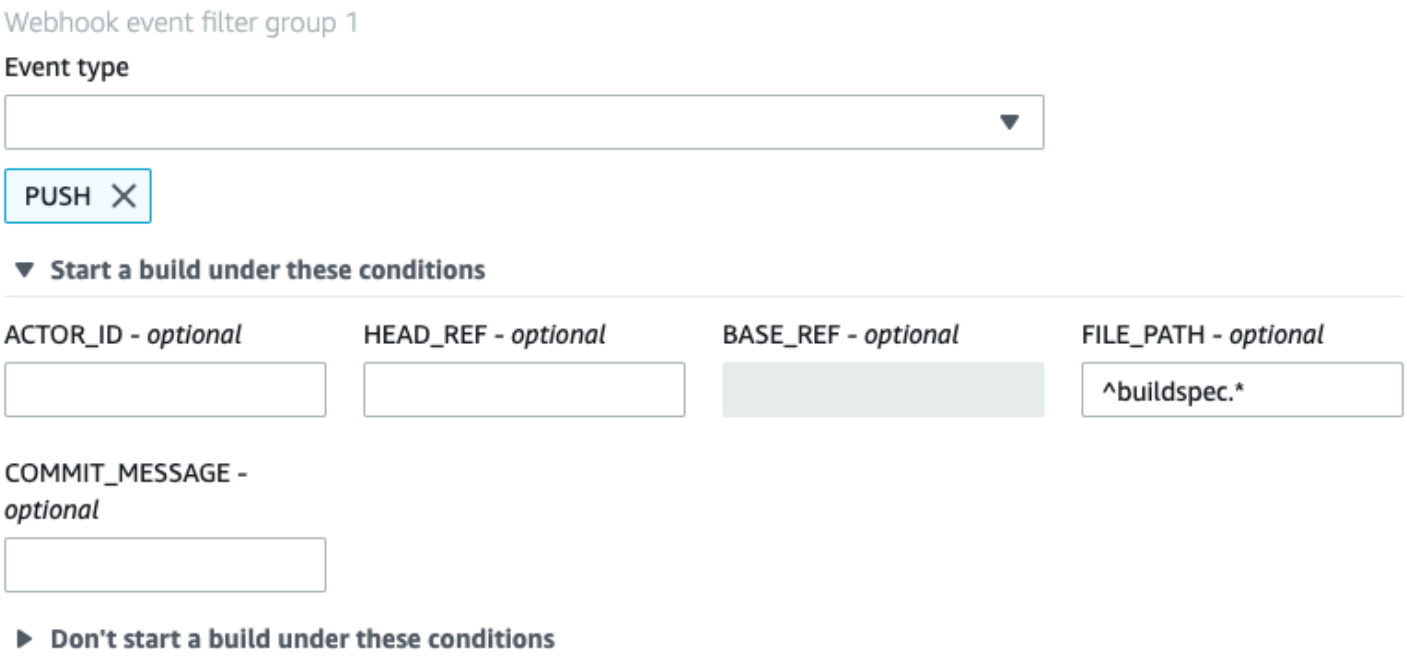
- The first filter group specifies pull requests that are created, updated, or reopened on branches with Git reference names that match the regular expression `^refs/heads/main$` and head references that match `^refs/heads/branch1$`.
- The second filter group specifies push requests on branches with Git reference names that match the regular expression `^refs/heads/branch1$`.



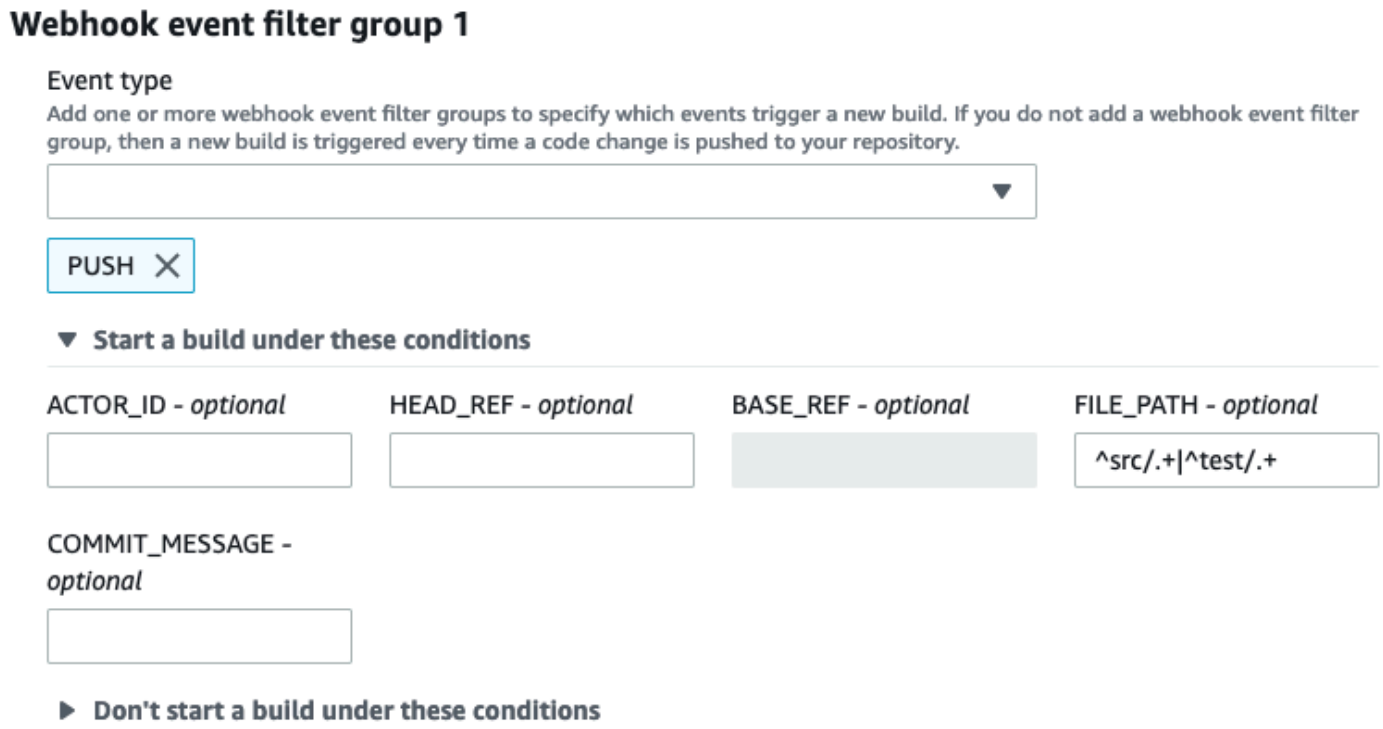
In this example, a webhook filter group triggers a build for all requests except tag events.



In this example, a webhook filter group triggers a build only when files with names that match the regular expression `^buildspec.*` change.

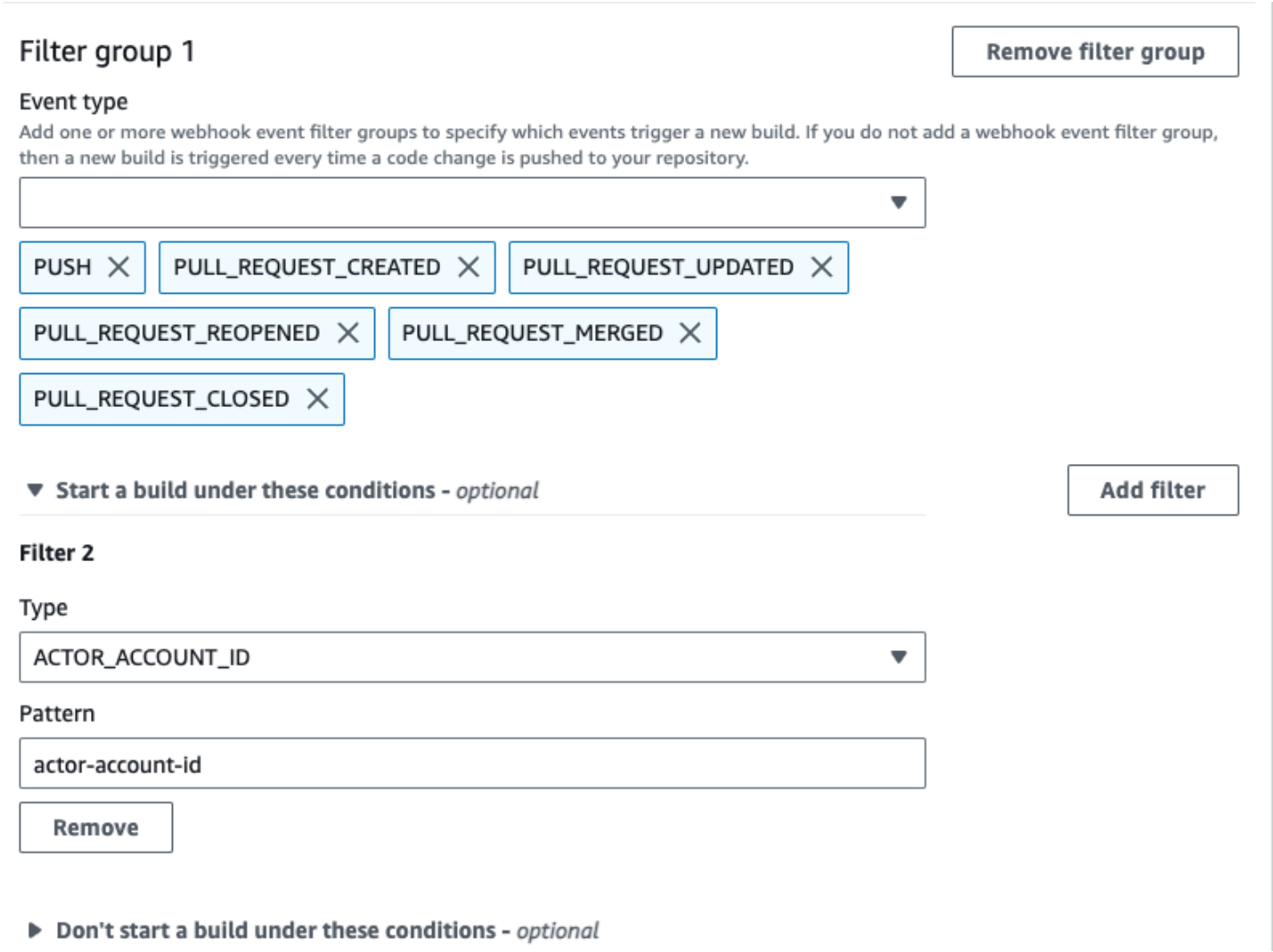


In this example, a webhook filter group triggers a build only when files are changed in `src` or `test` folders.

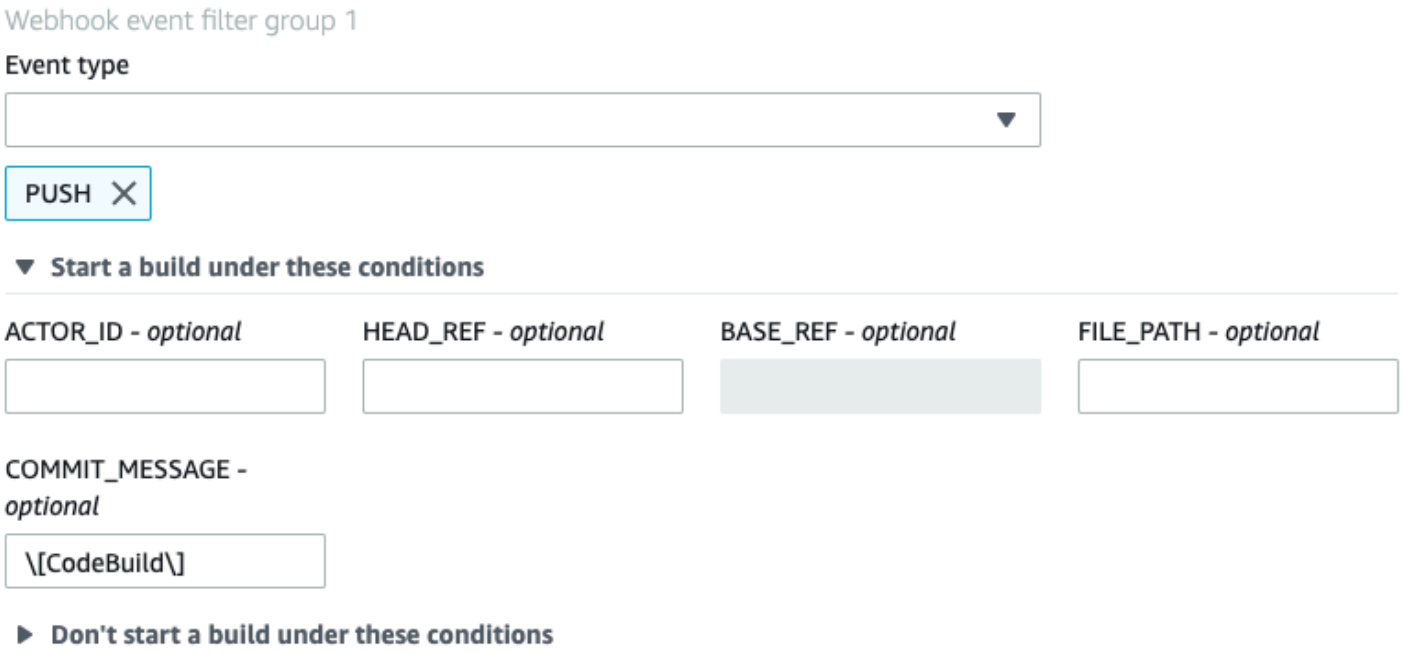


In this example, a webhook filter group triggers a build only when a change is made by a specified GitHub or GitHub Enterprise Server user with an account ID that matches the regular expression `actor-account-id`.

> **Note**
>
> For information about how to find your GitHub account ID, see https://api.github.com/users/*user-name*, where *user-name* is your GitHub user name.

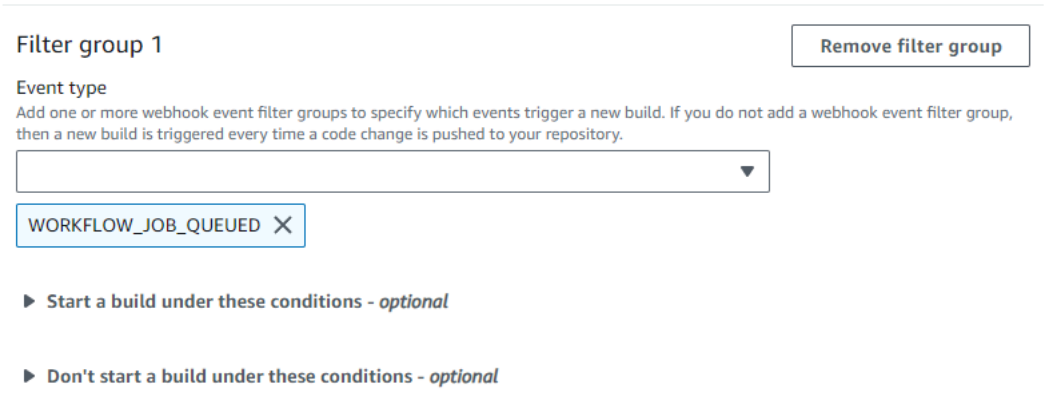


In this example, a webhook filter group triggers a build for a push event when the head commit message matches the regular expression `\[CodeBuild\]`.

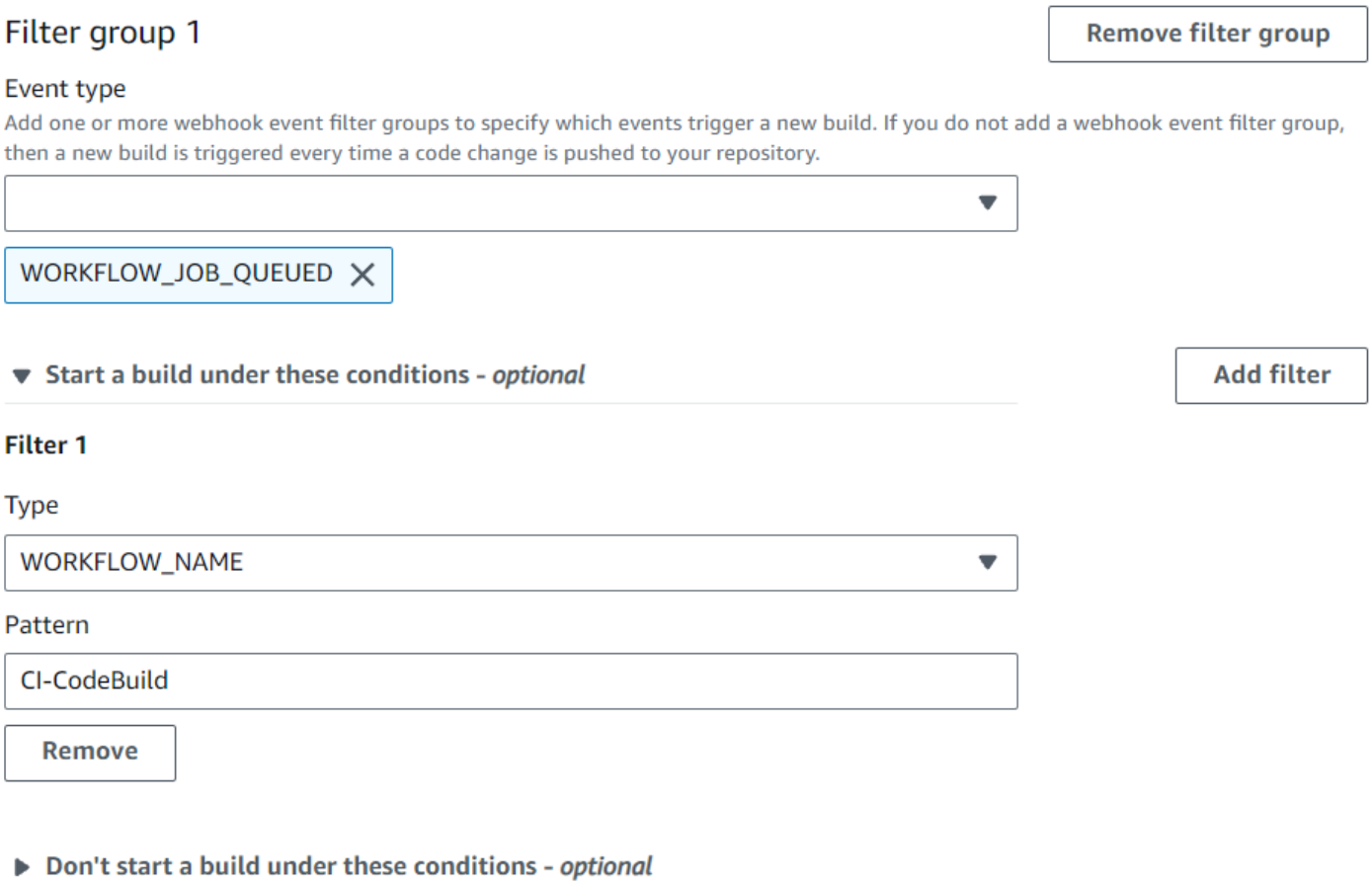


In this example, a webhook filter group triggers a build for GitHub Actions workflow job events only.

> **Note**
>
> CodeBuild will only process GitHub Actions workflow jobs if a webhook has filter groups containing the **WORKFLOW_JOB_QUEUED** event filter.



In this example, a webhook filter group triggers a build for a workflow name that matches the regular expression `CI-CodeBuild`.



Filter GitHub webhook events (SDK)

To use the AWS CodeBuild SDK to filter webhook events, use the `filterGroups` field in the request syntax of the `CreateWebhook` or `UpdateWebhook` API methods. For more information, see WebhookFilter in the *CodeBuild API Reference*.

For more information about GitHub webhook events, see GitHub webhook events.

To create a webhook filter that triggers a build for pull requests only, insert the following into the request syntax:

```
"filterGroups": [
   [
        {
            "type": "EVENT",
            "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
        }
```

```
    ]
]
```

To create a webhook filter that triggers a build for specified branches only, use the `pattern` parameter to specify a regular expression to filter branch names. Using an example of two filter groups, a build is triggered when one or both evaluate to true:

- The first filter group specifies pull requests that are created, updated, or reopened on branches with Git reference names that match the regular expression `^refs/heads/main$` and head references that match `^refs/heads/myBranch$`.
- The second filter group specifies push requests on branches with Git reference names that match the regular expression `^refs/heads/myBranch$`.

```
"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$"
        }
    ]
]
```

You can use the `excludeMatchedPattern` parameter to specify which events do not trigger a build. For example, in this example a build is triggered for all requests except tag events.

```
"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
        }
    ]
]
```

You can create a filter that triggers a build only when files with names that match the regular expression in the `pattern` argument change. In this example, the filter group specifies that a build is triggered only when files with a name that matches the regular expression `^buildspec.*` change.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^buildspec.*"
        }
    ]
]
```

In this example, the filter group specifies that a build is triggered only when files are changed in `src` or `test` folders.

```
"filterGroups": [
    [
```

```
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

You can create a filter that triggers a build only when a change is made by a specified GitHub or GitHub Enterprise Server user with account ID `actor-account-id`.

> **Note**
>
> For information about how to find your GitHub account ID, see https://api.github.com/users/*user-name*, where *user-name* is your GitHub user name.

```
"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"
        }
    ]
]
```

You can create a filter that triggers a build only when the head commit message matches the regular expression in the pattern argument. In this example, the filter group specifies that a build is triggered only when the head commit message of the push event matches the regular expression `\[CodeBuild\]`.

```
"filterGroups": [
```

```
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "COMMIT_MESSAGE",
            "pattern": "\[CodeBuild\]"
        }
    ]
]
```

To create a webhook filter that triggers a build for GitHub Actions workflow jobs only, insert the following into the request syntax:

```
"filterGroups": [
   [
        {
            "type": "EVENT",
            "pattern": "WORKFLOW_JOB_QUEUED"
        }
    ]
]
```

Filter GitHub webhook events (AWS CloudFormation)

To use an AWS CloudFormation template to filter webhook events, use the AWS CodeBuild project's `FilterGroups` property.

For more information about GitHub webhook events, see GitHub webhook events.

The following YAML-formatted portion of an AWS CloudFormation template creates two filter groups. Together, they trigger a build when one or both evaluate to true:

- The first filter group specifies pull requests are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` by a GitHub user who does not have account ID `12345`.
- The second filter group specifies push requests are created on files with names that match the regular expression `READ_ME` in branches with Git reference names that match the regular expression `^refs/heads/.*`.

- The third filter group specifies a push request with a head commit message matching the regular expression `\[CodeBuild\]`.
- The fourth filter group specifies a GitHub Actions workflow job request with a workflow name matching the regular expression `\[CI-CodeBuild\]`.

```
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 group webhooks

You can use CodeBuild GitLab group webhooks to start builds on webhook events from any repository within a GitLab group. Group webhooks work with any of the existing GitLab webhook event types, and can be configured by adding a scope configuration when creating a CodeBuild webhook. You can also use group webhooks to set up self-hosted GitLab runners within CodeBuild in order to receive `WORKFLOW_JOB_QUEUED` events from multiple repositories within a single project.

Topics

- Set up a group GitLab webhook
- Filter GitLab group webhook events (console)
- Filter GitLab group webhook events (AWS CloudFormation)

Set up a group GitLab webhook

The high-level steps to set up a group GitLab webhook are as follows. For more information about group GitLab webhooks, see GitLab group webhooks.

1. Set your project's source location to `CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION`.
2. In the webhook's scope configuration, set the scope to `GITLAB_GROUP`.
3. Specify a name as part of the webhook's scope configuration. For group webhooks, this is the group name.

> **Note**
>
> If the project's source type is `GITLAB_SELF_MANAGED`, you will also need to specify a domain as part of the webhook scope configuration.

4. (Optional) If you would only like to receive webhook events for specific repositories within your organization or enterprise, you can specify `REPOSITORY_NAME` as a filter when creating the webhook.
5. When creating a group webhook, ensure that CodeBuild has permissions to create group level webhooks within GitLab. To do so, you can use CodeBuild OAuth though CodeConnections. For more information, see GitLab access in CodeBuild.

   Note that group webhooks work with any of the existing GitLab webhook event types.

Filter GitLab group webhook events (console)

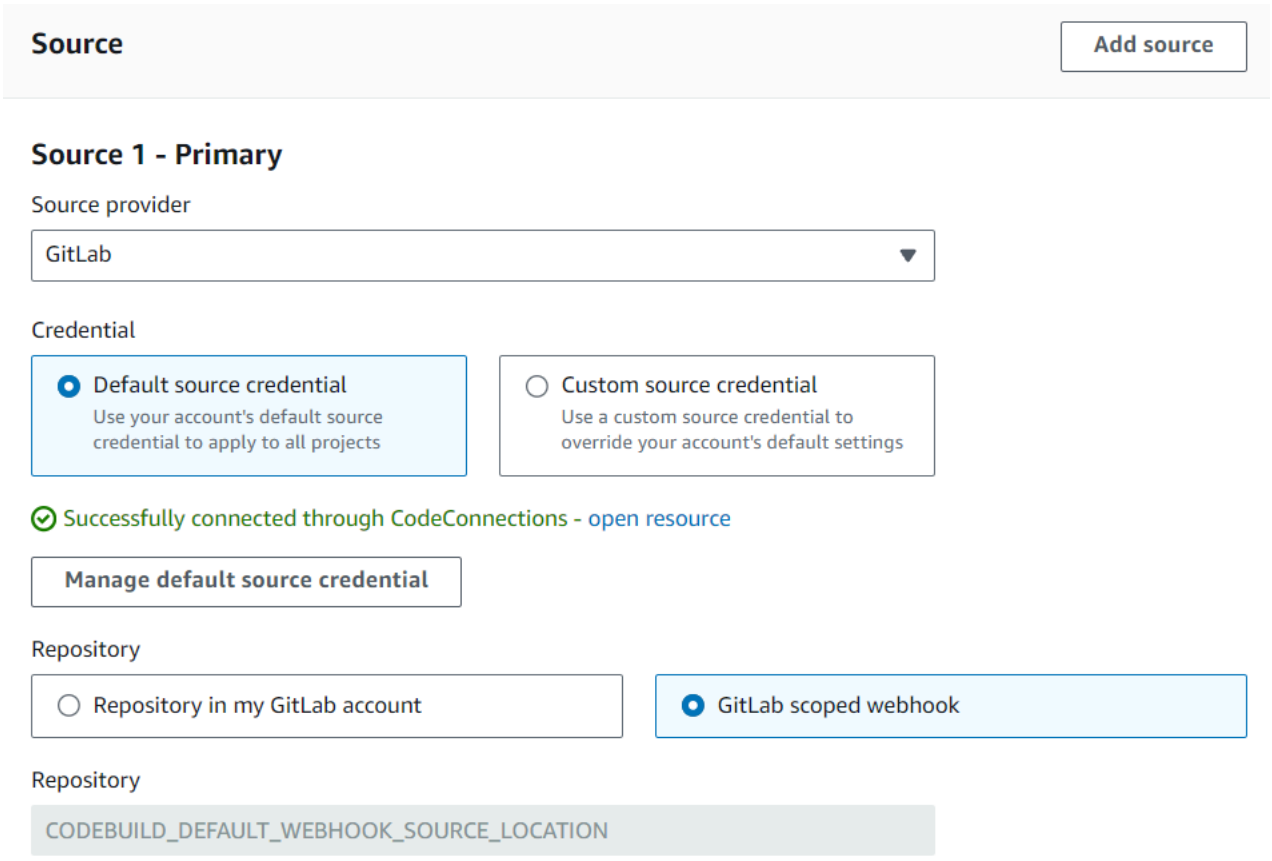
When creating a GitLab project through the console, select the following options to create a GitLab group webhook within the project. For more information about group GitLab webhooks, see GitLab group webhooks.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a build project. For information, see Create a build project (console) and Run a build (console).

   - In **Source**:
     - For **Source provider**, choose **GitLab** or **GitLab Self Managed**.
     - For **Repository**, choose **GitLab scoped webhook**.

       The GitLab repository will automatically be set to `CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION`, which is the required source location for group webhooks.

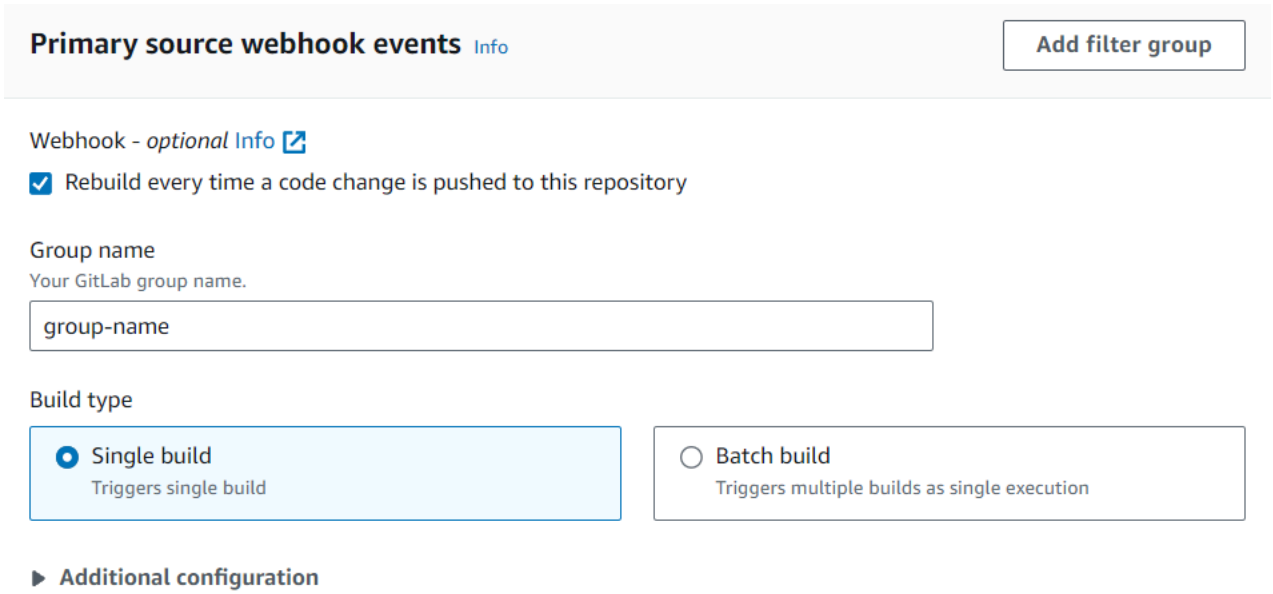
       > **Note**
       >
       > When using group webhooks, make sure that CodeBuild has permissions to create group level webhooks within GitLab. If you're using an existing OAuth connection, you may need to regenerate the connection in order to grant CodeBuild this permission.



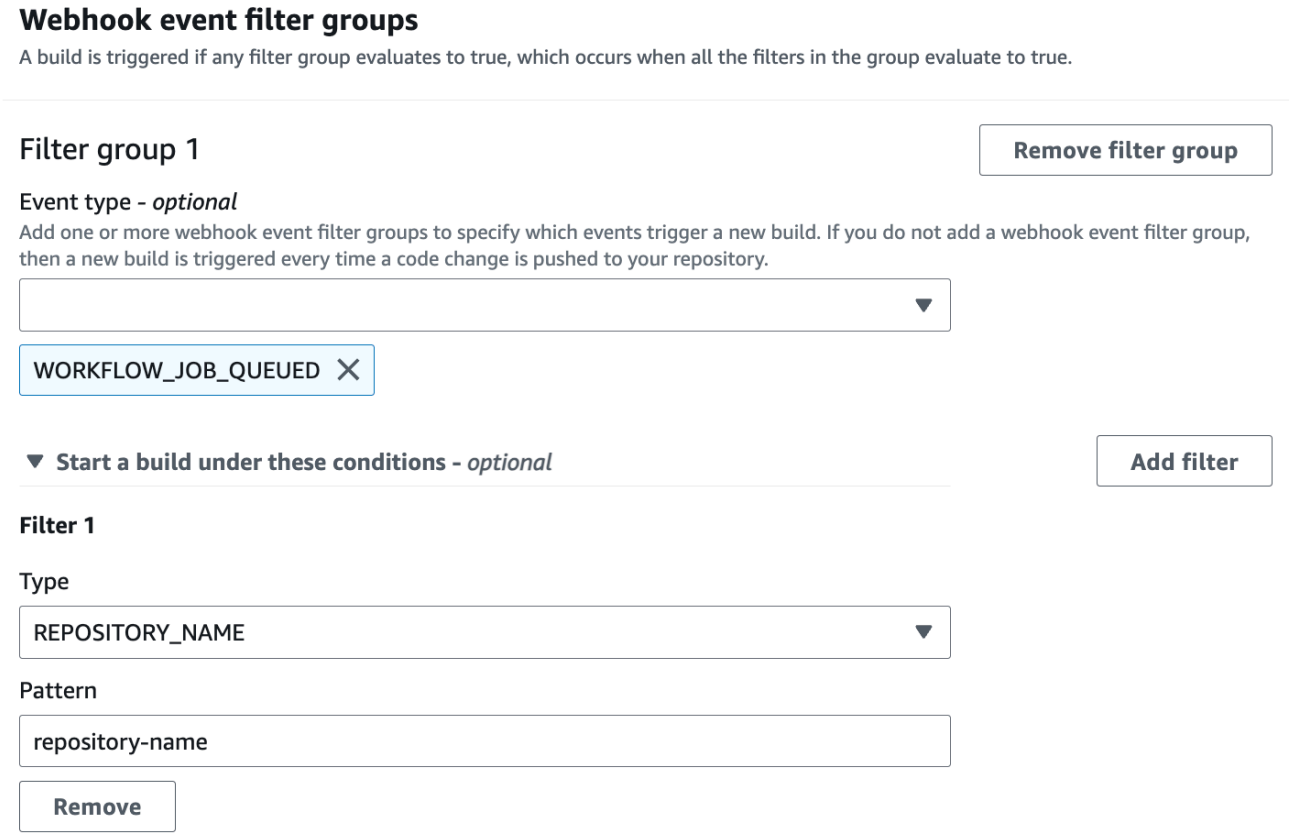
- In **Primary source webhook events**:
  - For **Group name**, enter the group name.

    If the project's source type is `GITLAB_SELF_MANAGED`, you also need to specify a domain as part of the webhook group configuration. For example, if the URL of your group is **`https://domain.com/group/group-name`**, then the domain is **`https://domain.com`**.

    > **Note**
    >
    > This name cannot be changed after the webhook has been created. To change the name, you can delete and re-create the webhook. If you want to remove the webhook entirely, you can also update the project source location to a GitLab repository.



- (Optional) In **Webhook event filter groups**, you can specify which events you would like to trigger a new build. You can also specify `REPOSITORY_NAME` as a filter to only trigger builds on webhook events from specific repositories.



You can also set the event type to `WORKFLOW_JOB_QUEUED` to set up self-hosted GitLab runners. For more information, see Self-managed GitLab runners in AWS CodeBuild.

3. Continue with the default values and then choose **Create build project**.

Filter GitLab group webhook events (AWS CloudFormation)

To use an AWS CloudFormation template to filter group webhook events, use the AWS CodeBuild project's `ScopeConfiguration` property. For more information about group GitLab webhooks, see GitLab group webhooks.

The following YAML-formatted portion of an AWS CloudFormation template creates four filter groups. Together, they trigger a build when one or all evaluate to true:

- The first filter group specifies pull requests are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` by a GitLab user who does not have account ID `12345`.
- The second filter group specifies push requests are created on files with names that match the regular expression `README_ME` in branches with Git reference names that match the regular expression `^refs/heads/.*`.
- The third filter group specifies a push request with a head commit message matching the regular expression `\[CodeBuild\]`.
- The fourth filter group specifies a GitLab CI/CD pipeline job request with a CI/CD pipeline name matching the regular expression `\[CI-CodeBuild\]`.

```
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: GITLAB
```

```
      Location: source-location
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: group-name
        Scope: GITLAB_GROUP
      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 manual webhooks

You can configure manual GitLab webhooks to prevent CodeBuild from automatically attempting to create a webhook within GitLab. CodeBuild returns a payload URL in as part of the call to create the webhook and can be used to manually create the webhook within GitLab. Even if CodeBuild is not allowlisted to create a webhook in your GitLab account, you can still manually create a webhook for your build project.

Use the following procedure to create a GitLab manual webhook.

To create a GitLab manual webhook

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Create a build project. For information, see Create a build project (console) and Run a build (console).
   - In **Source**:
     - For **Source provider**, choose **GitLab**.
     - For **Repository**, choose **Repository in my GitLab account**.
     - For **Repository URL**, enter **`https://gitlab.com/`*`user-name`*`/`*`repository-name`***.
   - In **Primary source webhook events**:
     - For **Webhook - optional**, choose **Rebuild every time a code change is pushed to this repository**.
     - Choose **Additional configuration** and for **Manual creation - optional**, choose **Manually create a webhook for this repository in GitLab console.**.
3. Continue with the default values and then choose **Create build project**. Take note of the **Payload URL** and **Secret** values as you will use these later.
4. Open the GitLab console at `https://gitlab.com/`*`user-name`*`/`*`repository-name`*`/-/hooks` and choose **Add new webhook**.
   - For **URL**, enter the Payload URL value you took note of earlier.
   - For **Secret token**, enter the Secret value you took note of earlier.
   - Configure the individual events that will send a webhook payload to CodeBuild. For **Trigger**, choose from the following events: **Push events**, **Merge request events**, **Releases events**, and **Job events**. To learn more about event types supported by CodeBuild, see GitLab webhook events.
5. Choose **Add webhook**.

GitLab webhook events

You can use webhook filter groups to specify which GitLab webhook events trigger a build. For example, you can specify that a build is only triggered for changes to specific branches.

You can create one or more webhook filter groups to specify which webhook events trigger a build. A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true. When you create a filter group, you specify:

An event

For GitLab, you can choose one or more of the following events: PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED, PULL_REQUEST_REOPENED, PULL_REQUEST_CLOSED, RELEASED, and WORKFLOW_JOB_QUEUED.

The webhook's event type is in its header in the `X-GitLab-Event` field. The following table shows how `X-GitLab-Event` header values map to the event types. For the `Merge Request Hook` webhook event, the payload's `object_atttributes.action` will contain additional information on merge request type.

X-GitLab-Event Header value	object_atttributes .action	Event type
Push Hook	N/A	PUSH
Merge Request Hook	open	PULL_REQUEST_CREATED
Merge Request Hook	update	PULL_REQUEST_UPDATED
Merge Request Hook	merge	PULL_REQUEST_MERGED
Merge Request Hook	reopen	PULL_REQUEST_REOPENED
Merge Request Hook	close	PULL_REQUEST_CLOSED
Release Hook	create, update	RELEASED
Job Hook	N/A	WORKFLOW_JOB_QUEUED

For PULL_REQUEST_MERGED, if a pull request is merged with the squash strategy and the pull request branch is closed, the original pull request commit no longer exists. In this case, the

CODEBUILD_WEBHOOK_MERGE_COMMIT environment variable contains the identifier of the squashed merge commit.

One or more optional filters

Use a regular expression to specify a filter. For an event to trigger a build, every filter within the group associated with it must evaluate to true.

ACTOR_ACCOUNT_ID (ACTOR_ID in the console)

A webhook event triggers a build when a GitLab account ID matches the regular expression pattern. This value appears in the account_id property of the actor object in the webhook filter payload.

HEAD_REF

A webhook event triggers a build when the head reference matches the regular expression pattern (for example, refs/heads/branch-name and refs/tags/tag-name). A HEAD_REF filter evaluates the Git reference name for the branch or tag. The branch or tag name appears in the name field of the new object in the push object of the webhook payload. For pull request events, the branch name appears in the name field in the branch object of the source object in the webhook payload.

BASE_REF

A webhook event triggers a build when the base reference matches the regular expression pattern. A BASE_REF filter works with pull request events only (for example, refs/heads/branch-name). A BASE_REF filter evaluates the Git reference name for the branch. The branch name appears in the name field of the branch object in the destination object in the webhook payload.

FILE_PATH

A webhook triggers a build when the path of a changed file matches the regular expression pattern.

COMMIT_MESSAGE

A webhook triggers a build when the head commit message matches the regular expression pattern.

WORKFLOW_NAME

A webhook triggers a build when the workflow name matches the regular expression pattern.

> **Note**
>
> You can find the webhook payload in the webhook settings of your GitLab repository.

Topics

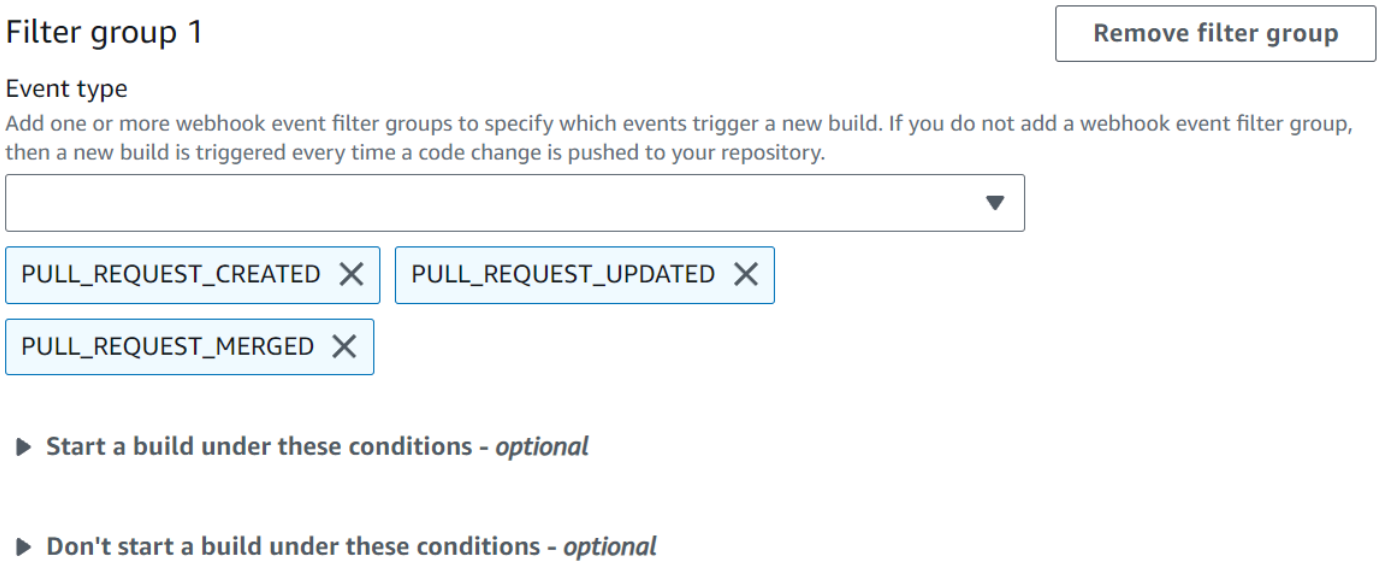
Filter GitLab webhook events (console)

Use the following instructions to use the AWS Management Console to filter webhook events. For more information about GitLab webhook events, see GitLab webhook events.

1. Select **Rebuild every time a code change is pushed to this repository** when you create your project.
2. From **Event type**, choose one or more events.
3. To filter when an event triggers a build, under **Start a build under these conditions**, add one or more optional filters.
4. To filter when an event is not triggered, under **Don't start a build under these conditions**, add one or more optional filters.
5. Choose **Add filter group** to add another filter group.

For more information, see Create a build project (console) and WebhookFilter in the *AWS CodeBuild API Reference*.

In this example, a webhook filter group triggers a build for pull requests only:



Using an example of two filter groups, a build is triggered when one or both evaluate to true:

- The first filter group specifies pull requests that are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` and head references that match `^refs/heads/branch1!`.
- The second filter group specifies push requests on branches with Git reference names that match the regular expression `^refs/heads/branch1$`.

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.

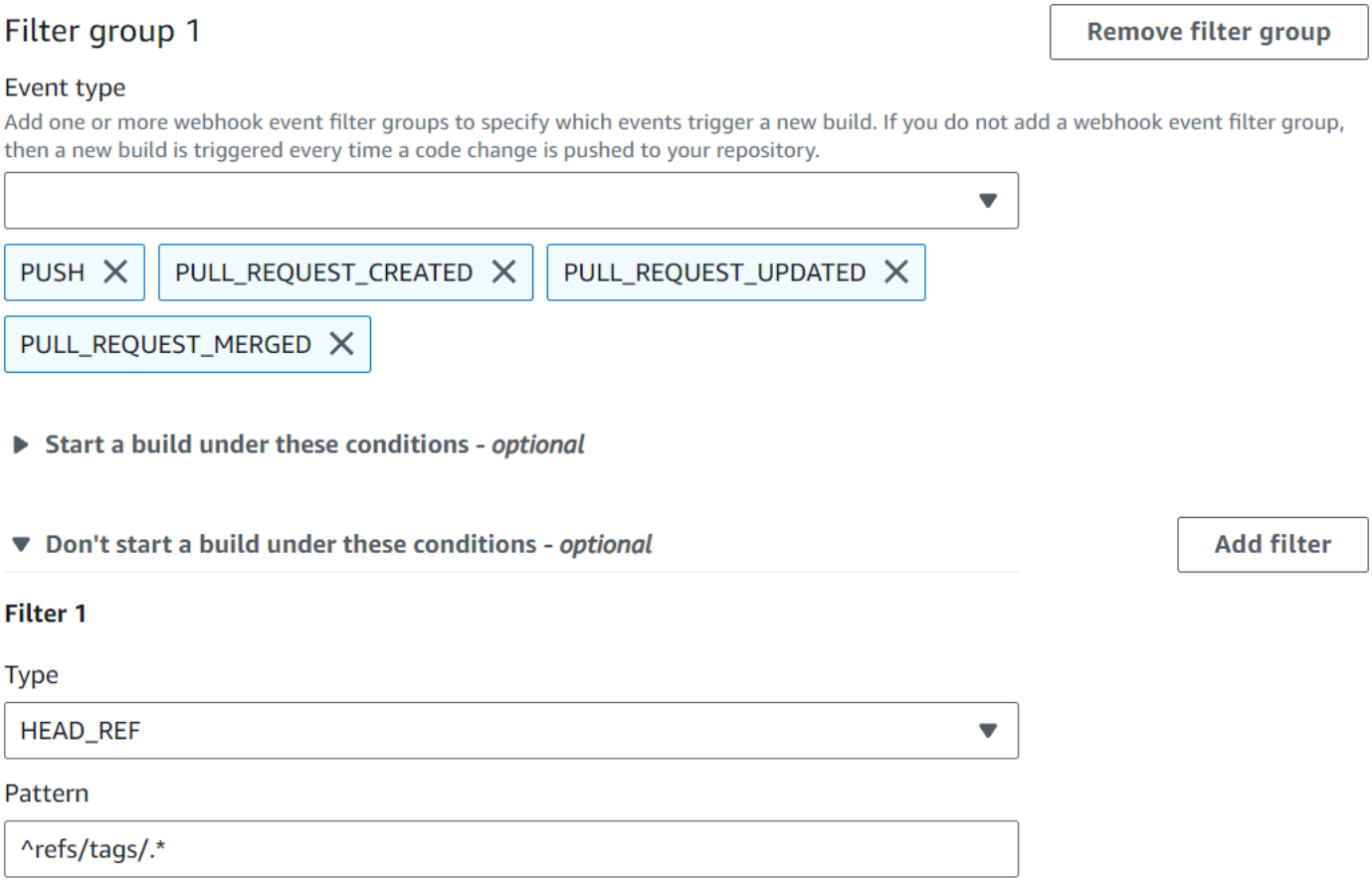
PUSH ✕

▼ **Start a build under these conditions** - *optional*

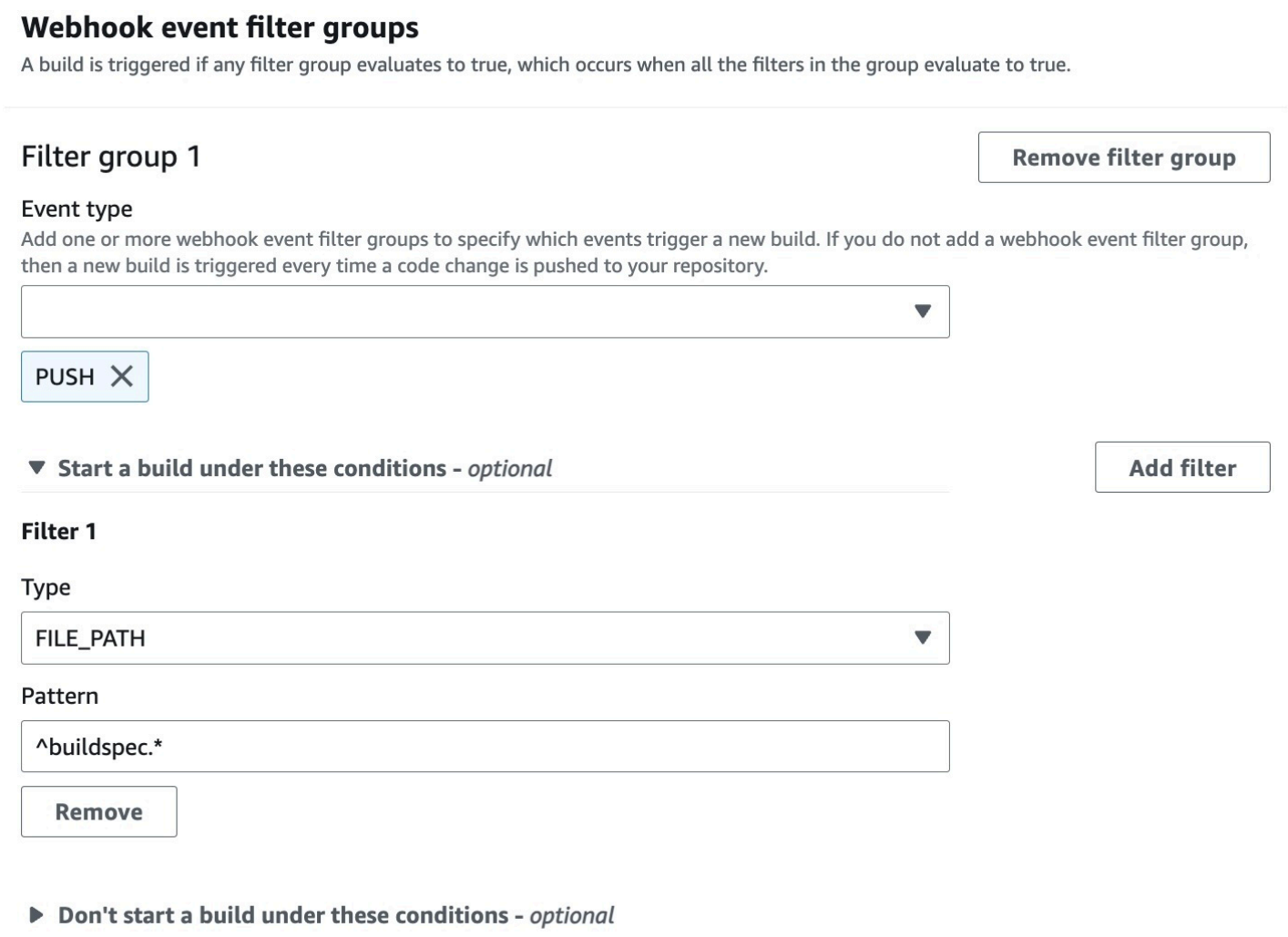
Add filter

Filter 1

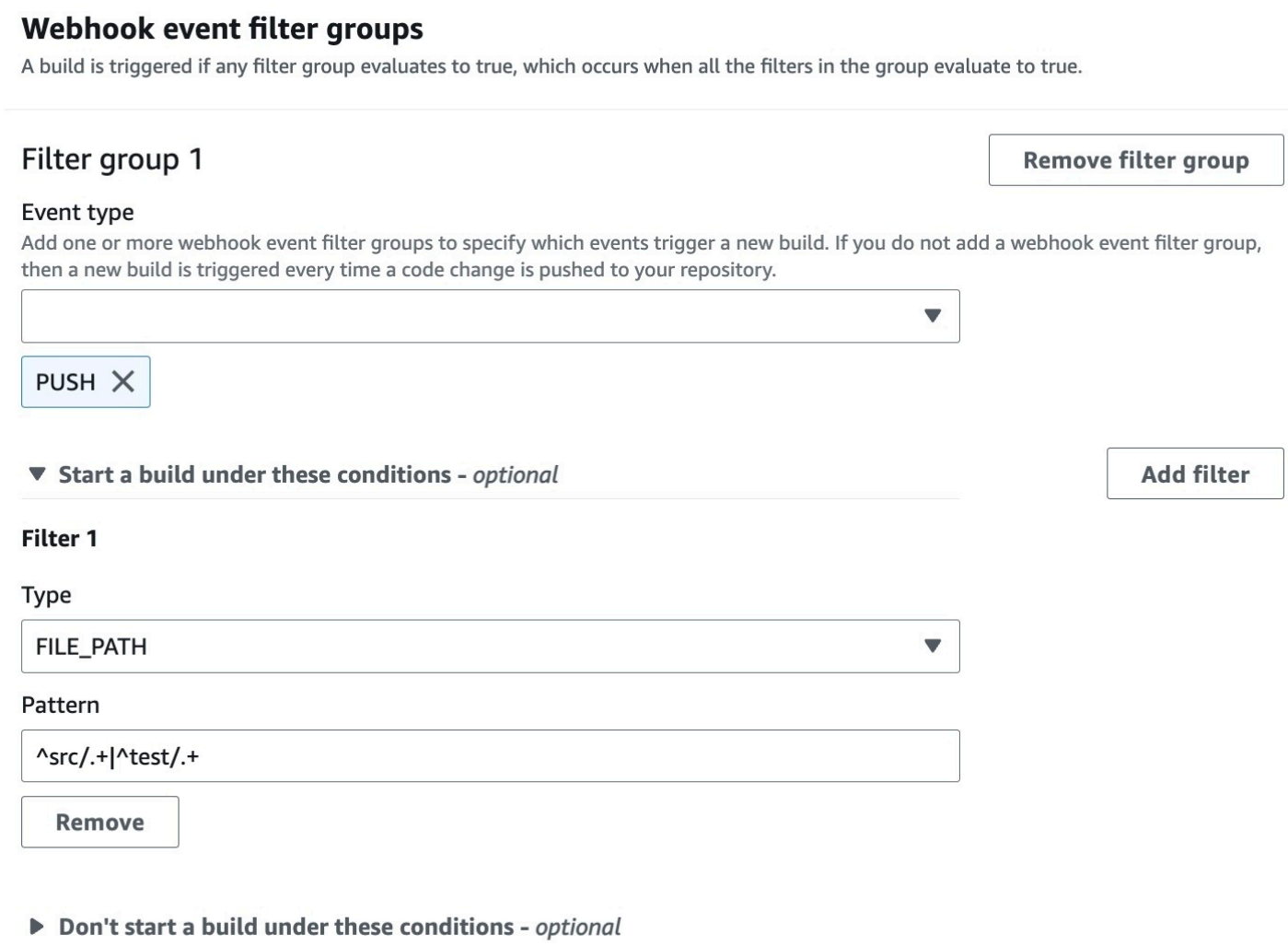
In this example, a webhook filter group triggers a build for all requests except tag events.



In this example, a webhook filter group triggers a build only when files with names that match the regular expression `^buildspec.*` change.

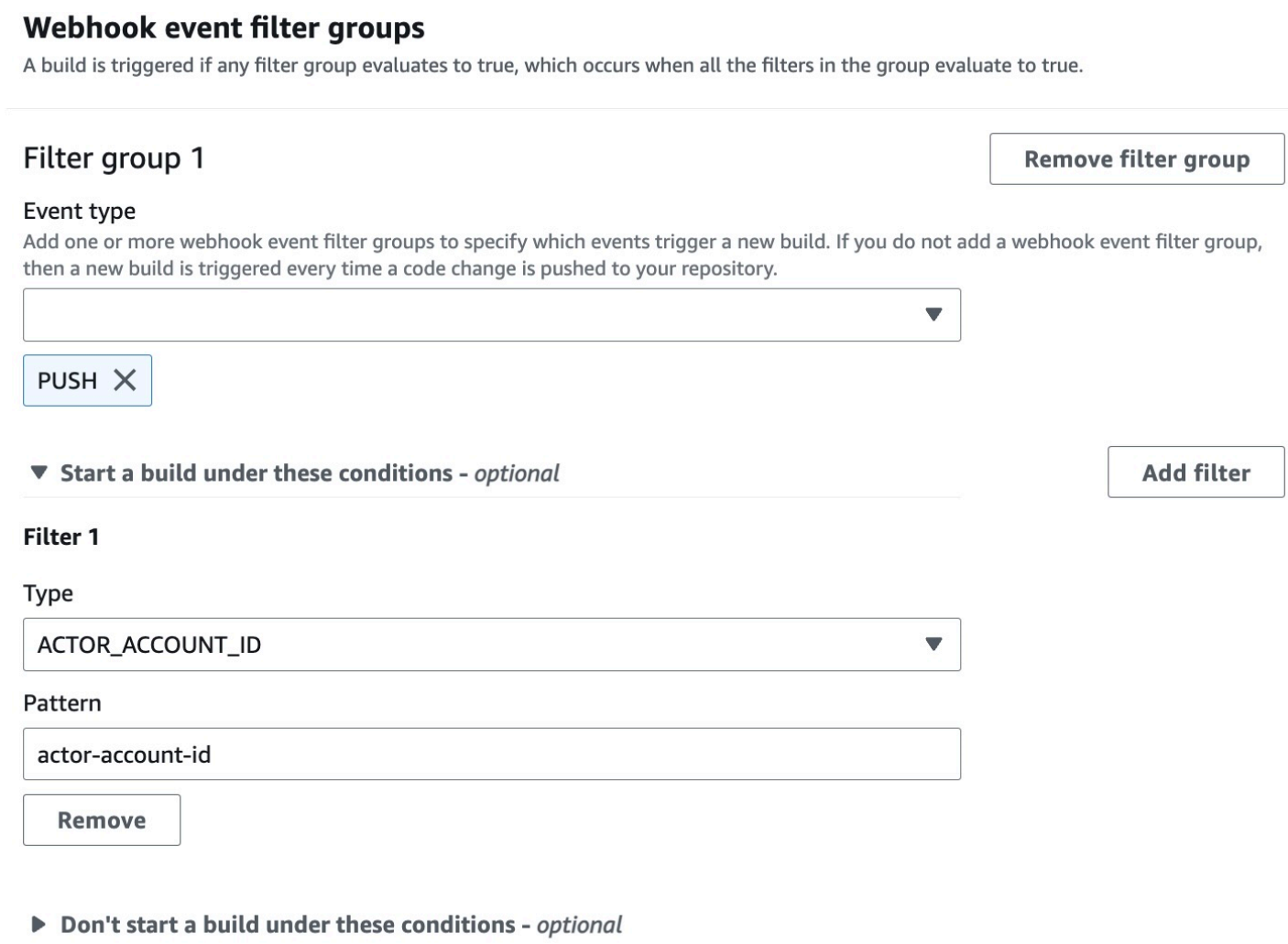


In this example, a webhook filter group triggers a build only when files are changed in `src` or `test` folders.

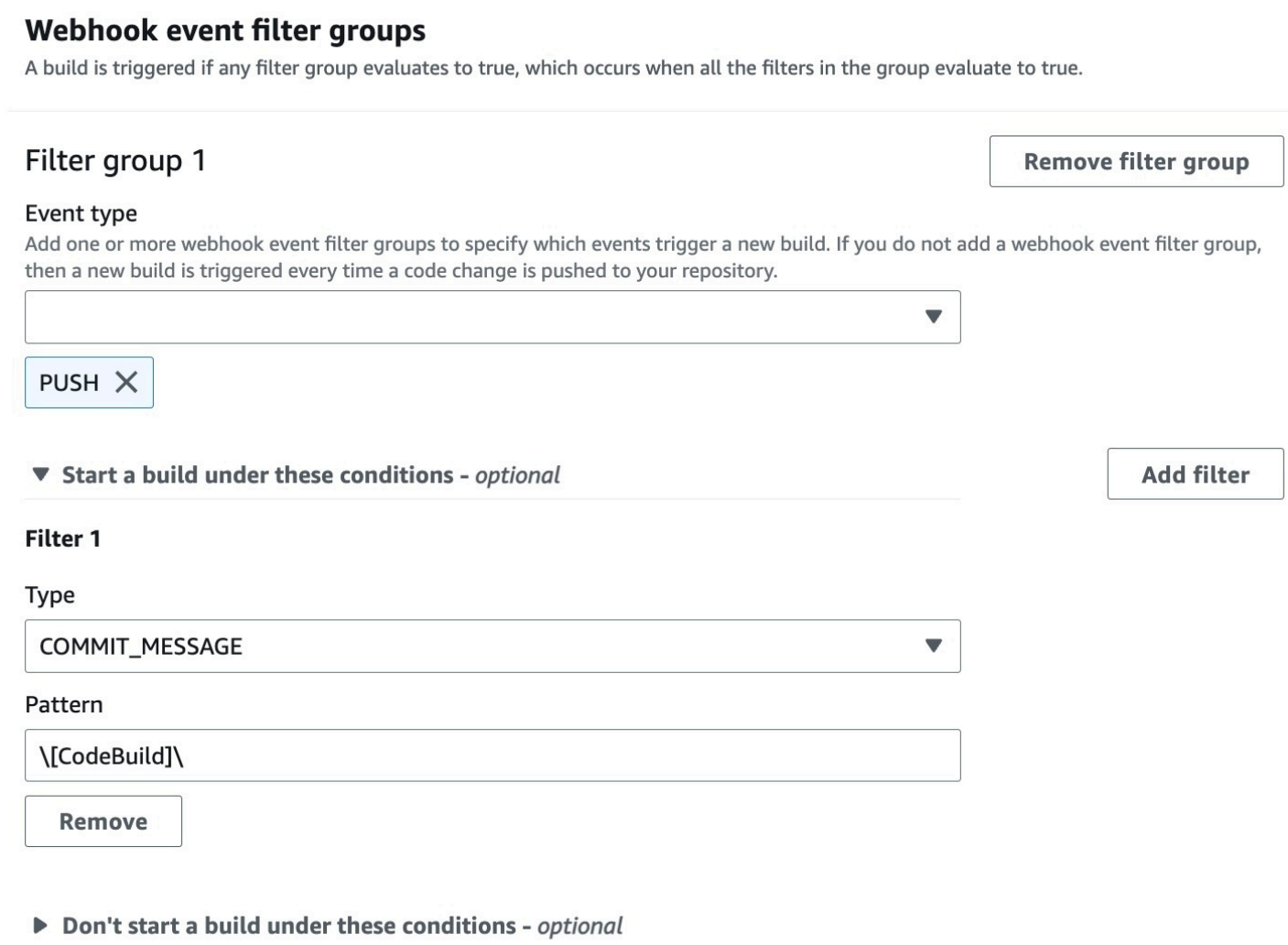


In this example, a webhook filter group triggers a build only when a change is made by a GitLab user who does not have an account ID that matches the regular expression `actor-account-id`.

> **Note**
>
> For information about how to find your GitLab account ID, see https://api.github.com/users/*`user-name`*, where *`user-name`* is your GitLab user name.



In this example, a webhook filter group triggers a build for a push event when the head commit message matches the regular expression `\[CodeBuild\]`.



Filter GitLab webhook events (SDK)

To use the AWS CodeBuild SDK to filter webhook events, use the `filterGroups` field in the request syntax of the `CreateWebhook` or `UpdateWebhook` API methods. For more information, see WebhookFilter in the *CodeBuild API Reference*.

For more information about GitLab webhook events, see GitLab webhook events.

To create a webhook filter that triggers a build for pull requests only, insert the following into the request syntax:

```
"filterGroups": [
  [
    {
      "type": "EVENT",
```

```
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"
    }
  ]
]
```

To create a webhook filter that triggers a build for specified branches only, use the `pattern` parameter to specify a regular expression to filter branch names. Using an example of two filter groups, a build is triggered when one or both evaluate to true:

- The first filter group specifies pull requests that are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` and head references that match `^refs/heads/myBranch$`.
- The second filter group specifies push requests on branches with Git reference names that match the regular expression `^refs/heads/myBranch$`.

```
"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$"
    }
  ]
```

```
]
```

You can use the `excludeMatchedPattern` parameter to specify which events do not trigger a build. In this example, a build is triggered for all requests except tag events.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/tags/.*",
      "excludeMatchedPattern": true
    }
  ]
]
```

You can create a filter that triggers a build only when a change is made by a GitLab user with account ID `actor-account-id`.

> **Note**
>
> For information about how to find your GitLab account ID, see https://api.github.com/users/*user-name*, where *user-name* is your GitLab user name.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
  ]
```

```
]
```

You can create a filter that triggers a build only when files with names that match the regular expression in the `pattern` argument change. In this example, the filter group specifies that a build is triggered only when files with a name that matches the regular expression `^buildspec.*` change.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^buildspec.*"
    }
  ]
]
```

In this example, the filter group specifies that a build is triggered only when files are changed in `src` or `test` folders.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

You can create a filter that triggers a build only when the head commit message matches the regular expression in the pattern argument. In this example, the filter group specifies that a build is triggered only when the head commit message of the push event matches the regular expression `\[CodeBuild\]`.

```
  "filterGroups": [
    [
      {
        "type": "EVENT",
        "pattern": "PUSH"
      },
      {
        "type": "COMMIT_MESSAGE",
        "pattern": "\[CodeBuild\]"
      }
    ]
  ]
```

Filter GitLab webhook events (AWS CloudFormation)

To use an AWS CloudFormation template to filter webhook events, use the AWS CodeBuild project's `FilterGroups` property. For more information about GitLab webhook events, see GitLab webhook events.

The following YAML-formatted portion of an AWS CloudFormation template creates two filter groups. Together, they trigger a build when one or both evaluate to true:

- The first filter group specifies pull requests are created or updated on branches with Git reference names that match the regular expression `^refs/heads/main$` by a GitLab user who does not have account ID `12345`.
- The second filter group specifies push requests are created on branches with Git reference names that match the regular expression `^refs/heads/.*`.
- The third filter group specifies a push request with a head commit message matching the regular expression `\[CodeBuild\]`.
- The fourth filter group specifies a GitHub Actions workflow job request with a workflow name matching the regular expression `\[CI-CodeBuild\]`.

```
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: 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: \[CodeBuild\]
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

Buildkite manual webhooks

Currently, CodeBuild requires all Buildkite webhooks to be created manually. CodeBuild returns a payload URL as part of the call to create the webhook, which can be used to manually create the webhook within Buildkite.

Use the following procedure to create a Buildkite manual webhook.

To create a CodeBuild project with a webhook

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.

2. Create a build project. For information, see Create a build project (console) and Run a build (console).
3. In **Project configuration**, choose **Runner project**.

   In **Runner**:

   - For **Runner provider**, choose **Buildkite**.
   - For **Buildkite agent token**, choose **Create a new agent token by using the create secret page**. You will be prompted to create a new secret in AWS Secrets Manager with a secret value equal to the Buildkite agent token you generated above.
   - (Optional) If you would like to use CodeBuild managed credentials for your job, select your job's source repository provider under **Buildkite source credential options** and verify that credentials are configured for your account. Additionally, verify that your Buildkite pipeline uses **Checkout using HTTPS**.
4. - In **Environment**:
     - Choose a supported **Environment image** and **Compute**. Note that you have the option to override the image and instance settings by using a label in your GitHub Actions workflow YAML. For more information, see Step 2: Update your GitHub Actions workflow YAML
   - In **Buildspec**:
     - Note that your buildspec will be ignored unless `buildspec-override:true` is added as a label. Instead, CodeBuild will override it to use commands that will setup the self-hosted runner.
5. Continue with the default values and then choose **Create build project**.
6. Save the **Payload URL** and **Secret** values from the **Create Webhook** popup. Follow the instructions in the popup to create a new Buildkite organization webhook.

View a build project's details in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to view the details of a build project in CodeBuild.

Topics

- View a build project's details (console)
- View a build project's details (AWS CLI)
- View a build project's details (AWS SDKs)

View a build project's details (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.

> **Note**
>
> By default, only the 10 most recent build projects are displayed. To view more build projects, choose the gear icon, and then choose a different value for **Projects per page** or use the back and forward arrows.

3. In the list of build projects, in the **Name** column, choose the link for the build project.
4. On the **Build project: *project-name*** page, choose **Build details**.

View a build project's details (AWS CLI)

Run the **batch-get-projects** command:

```
aws codebuild batch-get-projects --names names
```

In the preceding command, replace the following placeholder:

- *names*: Required string used to indicate one or more build project names to view details about. To specify more than one build project, separate each build project's name with a space. You can specify up to 100 build project names. To get a list of build projects, see View a list of build project names (AWS CLI).

For example, if you run this command:

```
aws codebuild batch-get-projects --names codebuild-demo-project codebuild-demo-project2
 my-other-demo-project
```

A result similar to the following might appear in the output. Ellipses (`. . .`) are used to represent data omitted for brevity.

```
{
```

```
  "projectsNotFound": [
    "my-other-demo-project"
  ],
  "projects": [
    {
      ...
      "name": codebuild-demo-project,
      ...
    },
    {
      ...
      "name": codebuild-demo-project2",
      ...
    }
  ]
}
```

In the preceding output, the `projectsNotFound` array lists any build project names that were specified, but not found. The `projects` array lists details for each build project where information was found. Build project details have been omitted from the preceding output for brevity. For more information, see the output of Create a build project (AWS CLI).

The **batch-get-projects** command does not support filtering for certain property values, but you can write a script that enumerates the properties for a project. For example, the following Linux shell script enumerates the projects in the current region for the current account, and prints the image used by each project.

```
#!/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
```

For more information about using the AWS CLI with AWS CodeBuild, see the Command line reference.

View a build project's details (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

View build project names in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to view a list of build projects in CodeBuild.

Topics

- View a list of build project names (console)
- View a list of build project names (AWS CLI)

- View a list of build project names (AWS SDKs)

View a list of build project names (console)

You can view a list of build projects in an AWS Region in the console. Information includes the name, source provider, repository, latest build status, and description, if any.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.

> **Note**
>
> By default, only the 10 most recent build projects are displayed. To view more build projects, choose the gear icon, and then choose a different value for **Projects per page** or use the back and forward arrows.

View a list of build project names (AWS CLI)

Run the **list-projects** command:

```
aws codebuild list-projects --sort-by sort-by --sort-order sort-order --next-
token next-token
```

In the preceding command, replace the following placeholders:

- *sort-by*: Optional string used to indicate the criterion to be used to list build project names. Valid values include:
  - CREATED_TIME: List the build project names based on when each build project was created.
  - LAST_MODIFIED_TIME: List the build project names based on when information about each build project was last changed.
  - NAME: List the build project names based on each build project's name.
- *sort-order*: Optional string used to indicate the order in which to list build projects, based on *sort-by*. Valid values include ASCENDING and DESCENDING.
- *next-token*: Optional string. During a previous run, if there were more than 100 items in the list, only the first 100 items are returned, along with a unique string called *next token*. To get the

next batch of items in the list, run this command again, adding the next token to the call. To get all of the items in the list, keep running this command with each subsequent next token, until no more next tokens are returned.

For example, if you run this command:

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

A result similar to the following might appear in the output:

```
{
  "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"
  ]
}
```

If you run this command again:

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-token
 Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=
```

A result similar to the following might appear in the output:

```
{
  "projects": [
    "codebuild-demo-project100",
    "codebuild-demo-project101",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project122"
  ]
}
```

View a list of build project names (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Builds in AWS CodeBuild

A *build* represents a set of actions performed by AWS CodeBuild to create output artifacts (for example, a JAR file) based on a set of input artifacts (for example, a collection of Java class files).

The following rules apply when you run multiple builds:

- When possible, builds run concurrently. The maximum number of concurrently running builds can vary. For more information, see Quotas for AWS CodeBuild.
- If the build project has a concurrent build limit set, builds return an error if the number of running builds reaches the concurrent build limit for the project. For more information, see Enable concurrent build limit.
- If the build project does not have a concurrent build limit set, builds are queued if the number of running builds reaches the concurrent build limit for the platform and compute type. The maximum number of builds in a queue is five times the concurrent build limit. For more information, see Quotas for AWS CodeBuild.

  A build in a queue that does not start after the number of minutes specified in its time out value is removed from the queue. The default timeout value is eight hours. You can override the build queue timeout with a value between five minutes and eight hours when you run your build. For more information, see Run AWS CodeBuild builds manually.

  It is not possible to predict the order in which queued builds start.

> **Note**
>
> You can access the history of a build for one year.

You can perform these tasks when working with builds:

Topics

- Run AWS CodeBuild builds manually
- Run builds on AWS Lambda compute
- Run builds on reserved capacity fleets
- Run builds in batches

- Execute parallel tests in batch builds
- Cache builds to improve performance
- Debug builds in AWS CodeBuild
- Delete builds in AWS CodeBuild
- Retry builds manually in AWS CodeBuild
- Retry builds automatically in AWS CodeBuild
- Stop builds in AWS CodeBuild
- Stop batch builds in AWS CodeBuild
- Trigger AWS CodeBuild builds automatically
- View build details in AWS CodeBuild
- View a list of build IDs in AWS CodeBuild
- View a list of build IDs for a build project in AWS CodeBuild

Run AWS CodeBuild builds manually

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to run a build in CodeBuild.

Topics

- Run builds locally with the AWS CodeBuild agent
- Run a build (console)
- Run a build (AWS CLI)
- Run a batch build (AWS CLI)
- Start running builds automatically (AWS CLI)
- Stop running builds automatically (AWS CLI)
- Run a build (AWS SDKs)

Run builds locally with the AWS CodeBuild agent

You can use the AWS CodeBuild agent to run CodeBuild builds on a local machine. There are agents available for x86_64 and ARM platforms.

You can also subscribe to receive notifications when new versions of the agent are published.

Prerequisites

Before you begin, you need to do the following:

- Install Git on your local machine.
- Install and set up Docker on your local machine.

Set up the build image

You only need to set up the build image the first time you run the agent, or when the image has changed.

To set up the build image

1. If you want to use a curated Amazon Linux 2 image, you can pull it from the CodeBuild public Amazon ECR repository at https://gallery.ecr.aws/codebuild/amazonlinux-x86_64-standard with the following command:

   ```
   $ docker pull public.ecr.aws/codebuild/amazonlinux-x86_64-standard:4.0
   ```

   Alternatively, if you want to use another Linux image, perform the following steps:

   a. Clone the CodeBuild image repo:

      ```
      $ git clone https://github.com/aws/aws-codebuild-docker-images.git
      ```

   b. Change to the image directory. For this example, use the `aws/codebuild/standard:5.0` image:

      ```
      $ cd aws-codebuild-docker-images/ubuntu/standard/5.0
      ```

   c. Build the image. This will take several minutes.

      ```
      $ docker build -t aws/codebuild/standard:5.0 .
      ```

2. Download the CodeBuild agent.

   To download the x86_64 version of the agent, run the following command:

```
$ docker pull public.ecr.aws/codebuild/local-builds:latest
```

To download the ARM version of the agent, run the following command:

```
$ docker pull public.ecr.aws/codebuild/local-builds:aarch64
```

3. The CodeBuild agent is available from https://gallery.ecr.aws/codebuild/local-builds.

   The Secure Hash Algorithm (SHA) signature for the x86_64 version of the agent is:

   ```
   sha256:ccb19bdd7af94e4dc761e4c58c267e9455c28ec68d938086b4dc1cf8fe6b0940
   ```

   The SHA signature for the ARM version of the agent is:

   ```
   sha256:7d7b5d35d2ac4e062ae7ba8c662ffed15229a52d09bd0d664a7816c439679192
   ```

   You can use the SHA to identify the version of the agent. To see the agent's SHA signature, run the following command and look for the SHA under `RepoDigests`:

   ```
   $ docker inspect public.ecr.aws/codebuild/local-builds:latest
   ```

Run the CodeBuild agent

To run the CodeBuild agent

1. Change to the directory that contains your build project source.
2. Download the codebuild_build.sh script:

   ```
   $ curl -O  https://raw.githubusercontent.com/aws/aws-codebuild-docker-images/
   master/local_builds/codebuild_build.sh
   $ chmod +x codebuild_build.sh
   ```

3. Run the `codebuild_build.sh` script and specify your container image and the output directory.

   To run an x86_64 build, run the following command:

```
$ ./codebuild_build.sh -i <container-image> -a <output directory>
```

To run an ARM build, run the following command:

```
$ ./codebuild_build.sh -i <container-image> -a <output directory> -l
 public.ecr.aws/codebuild/local-builds:aarch64
```

Replace *<container-image>* with the name of the container image, such as `aws/codebuild/standard:5.0` or `public.ecr.aws/codebuild/amazonlinux-x86_64-standard:4.0`.

The script launches the build image and runs the build on the project in the current directory. To specify the location of the build project, add the `-s` *<build project directory>* option to the script command.

Receive notifications for new CodeBuild agent versions

You can subscribe to Amazon SNS notifications so you will be notified when new versions of the AWS CodeBuild agent are released.

To subscribe to CodeBuild agent notifications

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. In the navigation bar, if it's not already selected, change the AWS Region to **US East (N. Virginia)**. You must select this AWS Region because the Amazon SNS notifications that you are subscribing to are created in this Region.
3. In the navigation pane, choose **Subscriptions**.
4. Choose **Create subscription**.
5. In **Create subscription**, do the following:

   a. For **Topic ARN**, use the following Amazon Resource Name (ARN):

   ```
   arn:aws:sns:us-east-1:850632864840:AWS-CodeBuild-Local-Agent-Updates
   ```

   b. For **Protocol**, choose **Email** or **SMS**.

c. For **Endpoint**, choose where (email or SMS) to receive the notifications. Enter an email or address or phone number, including area code.

d. Choose **Create subscription**.

e. Choose **Email** to receive an email asking you to confirm your subscription. Follow the directions in the email to complete your subscription.

If you no longer want to receive these notifications, use the following procedure to unsubscribe.

To unsubscribe from CodeBuild agent notifications

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. In the navigation pane, choose **Subscriptions**.
3. Select the subscription and from **Actions**, choose **Delete subscriptions**. When you are prompted to confirm, choose **Delete**.

Run a build (console)

To use AWS CodePipeline to run a build with CodeBuild, skip these steps and follow the instructions in Use CodeBuild with CodePipeline.

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. In the list of build projects, choose the build project.
4. You can run the build with the default build project settings, or override build settings for this build only.

   a. If you want to run the build with the default build project settings, choose **Start build**. The build starts immediately.

   b. If you want to override the default build project settings, choose **Start build with overrides**. In the **Start build** page, you can override the following:

      - **Build configuration**
      - **Source**

- **Environment variable overrides**

If you need to select more advanced overrides, choose **Advanced build overrides**. In this page, you can override the following:

- **Build configuration**
- **Source**
- **Environment**
- **Buildspec**
- **Artifacts**
- **Logs**

When you have made your override selections, choose **Start build**.

For detailed information about this build, see View build details (console).

Run a build (AWS CLI)

> **Note**
>
> To use CodePipeline to run a build with AWS CodeBuild, skip these steps and follow the instructions in Create a pipeline that uses CodeBuild (AWS CLI).
> For more information about using the AWS CLI with CodeBuild, see the Command line reference.

1. Run the `start-build` command in one of the following ways:

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

Use this if you want to run a build that uses the latest version of the build input artifact and the build project's existing settings.

```
aws codebuild start-build --generate-cli-skeleton
```

Use this if you want to run a build with an earlier version of the build input artifact or if you want to override the settings for the build output artifacts, environment variables, buildspec, or default build timeout period.

2. If you run the **start-build** command with the `--project-name` option, replace *<project-name>* with the name of the build project, and then skip to step 6 of this procedure. To get a list of build projects, see View build project names.

3. If you run the **start-build** command with the `--idempotency-token` option, a unique case-sensitive identifier or token, is included with the `start-build` request. The token is valid for 5 minutes after the request. If you repeat the `start-build` request with the same token, but change a parameter, CodeBuild returns a parameter mismatch error.

4. If you run the **start-build** command with the `--generate-cli-skeleton` option, JSON-formatted data appears in the output. Copy the data to a file (for example, *start-build.json*) in a location on the local computer or instance where the AWS CLI is installed. Modify the copied data to match the following format, and save your results:

```
{
  "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"
}
```

Replace the following placeholders:

- *projectName*: Required string. The name of the build project to use for this build.
- *sourceVersion*: Optional string. A version of the source code to be built, as follows:
  - For Amazon S3, the version ID that corresponds to the version of the input ZIP file you want to build. If *sourceVersion* is not specified, then the latest version is used.
  - For CodeCommit, the commit ID that corresponds to the version of the source code you want to build. If *sourceVersion* is not specified, the default branch's HEAD commit ID is used. (You cannot specify a tag name for *sourceVersion*, but you can specify the tag's commit ID.)
  - For GitHub, the commit ID, pull request ID, branch name, or tag name that corresponds to the version of the source code you want to build. If a pull request ID is specified, it must use the format `pr/`*pull-request-ID* (for example, `pr/25`). If a branch name is specified, the branch's HEAD commit ID is used. If *sourceVersion* is not specified, the default branch's HEAD commit ID is used.
  - For Bitbucket, the commit ID, branch name, or tag name that corresponds to the version of the source code you want to build. If a branch name is specified, the branch's HEAD commit ID is used. If *sourceVersion* is not specified, the default branch's HEAD commit ID is used.
- The following placeholders are for `artifactsOverride`.
  - *type*: Optional. The build output artifact type that overrides for this build the one defined in the build project.

  - *location*: Optional. The build output artifact location that overrides for this build the one defined in the build project.
  - *path*: Optional. The build output artifact path that overrides for this build the one defined in the build project.
  - *namespaceType*: Optional. The build output artifact path type that overrides for this build the one defined in the build project.
  - *name*: Optional. The build output artifact name that overrides for this build the one defined in the build project.
  - *packaging*: Optional. The build output artifact packaging type that overrides for this build the one defined in the build project.
- *buildspecOverride*: Optional. A buildspec declaration that overrides for this build the one defined in the build project. If this value is set, it can be either an inline buildspec definition, the path to an alternate buildspec file relative to the value of the built-in `CODEBUILD_SRC_DIR` environment variable, or the path to an S3 bucket. The S3 bucket must be in the same AWS Region as the build project. Specify the buildspec file using its ARN (for example, `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`). If this value is not provided or is set to an empty string, the source code must contain a `buildspec.yml` file in its root directory. For more information, see Buildspec file name and storage location.
- The following placeholders are for `cacheOverride`.
  - *cacheOverride-location*: Optional. The location of a `ProjectCache` object for this build that overrides the `ProjectCache` object specified in the build project. `cacheOverride` is optional and takes a `ProjectCache` object. `location` is required in a `ProjectCache` object.
  - *cacheOverride-type*: Optional. The type of a `ProjectCache` object for this build that overrides the `ProjectCache` object specified in the build project. `cacheOverride` is optional and takes a `ProjectCache` object. `type` is required in a `ProjectCache` object.
- *certificateOverride*: Optional. The name of a certificate for this build that overrides the one specified in the build project.
- *environmentTypeOverride*: Optional. A container type for this build that overrides the one specified in the build project. The current valid string is `LINUX_CONTAINER`.
- The following placeholders are for `environmentVariablesOverride`.
  - *environmentVariablesOverride-name*: Optional. The name of an environment variable in the build project whose value you want to override for this build.

  - *environmentVariablesOverride-type*: Optional. The type of environment variable in the build project whose value you want to override for this build.
  - *environmentVariablesValue*: Optional. The value of the environment variable defined in the build project that you want to override for this build.
- *gitCloneDepthOverride*: Optional. The value of the **Git clone depth** in the build project whose value you want to override for this build. If your source type is Amazon S3, this value is not supported.
- *imageOverride*: Optional. The name of an image for this build that overrides the one specified in the build project.
- *idempotencyToken*: Optional. A string that serves as a token to specify that the build request is idempotent. You can choose any string that is 64 characters or less. The token is valid for 5 minutes after the start-build request. If you repeat the start-build request with the same token, but change a parameter, CodeBuild returns a parameter mismatch error.
- *insecureSslOverride*: Optional boolean that specifies whether to override the insecure TLS setting specified in the build project. The insecure TLS setting determines whether to ignore TLS warnings while connecting to the project source code. This override applies only if the build's source is GitHub Enterprise Server.
- *privilegedModeOverride*: Optional boolean. If set to true, the build overrides privileged mode in the build project.
- *queuedTimeoutInMinutesOverride*: Optional integer that specifies the number of minutes a build is allowed to be queued before it times out. Its minimum value is five minutes and its maximum value is 480 minutes (eight hours).
- *reportBuildStatusOverride*: Optional boolean that specifies whether to send your source provider the status of a build's start and completion. If you set this with a source provider other than GitHub, GitHub Enterprise Server, or Bitbucket, an invalidInputException is thrown.
- *sourceAuthOverride*: Optional string. An authorization type for this build that overrides the one defined in the build project. This override applies only if the build project's source is Bitbucket or GitHub.
- *sourceLocationOverride*: Optional string. A location that overrides for this build the source location for the one defined in the build project.
- *serviceRoleOverride*: Optional string. The name of a service role for this build that overrides the one specified in the build project.

- *sourceTypeOverride*: Optional string. A source input type for this build that overrides the source input defined in the build project. Valid strings are NO_SOURCE, CODECOMMIT, CODEPIPELINE, GITHUB, S3, BITBUCKET, and GITHUB_ENTERPRISE.
- *timeoutInMinutesOverride*: Optional number. The number of build timeout minutes that overrides for this build the one defined in the build project.

We recommend that you store an environment variable with a sensitive value, such as an AWS access key ID, an AWS secret access key, or a password as a parameter in Amazon EC2 Systems Manager Parameter Store. CodeBuild can use a parameter stored in Amazon EC2 Systems Manager Parameter Store only if that parameter's name starts with /CodeBuild/ (for example, /CodeBuild/dockerLoginPassword). You can use the CodeBuild console to create a parameter in Amazon EC2 Systems Manager. Choose **Create a parameter**, and then follow the instructions. (In that dialog box, for **KMS key**, you can optionally specify the ARN of an AWS KMS key in your account. Amazon EC2 Systems Manager uses this key to encrypt the parameter's value during storage and decrypt during retrieval.) If you use the CodeBuild console to create a parameter, the console starts the parameter with /CodeBuild/ as it is being stored. However, if you use the Amazon EC2 Systems Manager Parameter Store console to create a parameter, you must start the parameter's name with /CodeBuild/, and you must set **Type** to **Secure String**. For more information, see AWS Systems Manager parameter store and Walkthrough: Create and test a String parameter (console) in the *Amazon EC2 Systems Manager User Guide*.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the ssm:GetParameters action. If you chose **Create a new service role in your account** earlier, then CodeBuild includes this action in the default service role for your build project automatically. However, if you chose **Choose an existing service role from your account**, then you must include this action in your service role separately.

Environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named MY_VAR with a value of my_value, and you set an environment variable named MY_VAR with a value of other_value, then my_value is replaced by other_value. Similarly, if the Docker image already contains an environment variable named PATH with a value of /usr/local/sbin:/usr/local/bin, and you set an environment variable named PATH with a value of $PATH:/

`usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`.

Do not set any environment variable with a name that begins with `CODEBUILD_`. This prefix is reserved for internal use.

If an environment variable with the same name is defined in multiple places, the environment variable's value is determined as follows:

- The value in the start build operation call takes highest precedence.
- The value in the build project definition takes next precedence.
- The value in the buildspec file declaration takes lowest precedence.

For information about valid values for these placeholders, see Create a build project (AWS CLI). For a list of the latest settings for a build project, see View build project details.

5. Switch to the directory that contains the file you just saved, and run the `start-build` command again.

```
aws codebuild start-build --cli-input-json file://start-build.json
```

6. If successful, data similar to that described in the To run the build procedure appears in the output.

To work with detailed information about this build, make a note of the `id` value in the output, and then see View build details (AWS CLI).

Run a batch build (AWS CLI)

1. Run the `start-build-batch` command in one of the following ways:

```
aws codebuild start-build-batch --project-name <project-name>
```

Use this if you want to run a build that uses the latest version of the build input artifact and the build project's existing settings.

```
aws codebuild start-build-batch --generate-cli-skeleton > <json-file>
```

Use this if you want to run a build with an earlier version of the build input artifact or if you want to override the settings for the build output artifacts, environment variables, buildspec, or default build timeout period.

2. If you run the **start-build-batch** command with the `--project-name` option, replace *<project-name>* with the name of the build project, and then skip to step 6 of this procedure. To get a list of build projects, see View build project names.

3. If you run the **start-build-batch** command with the `--idempotency-token` option, a unique case-sensitive identifier, or token, is included with the `start-build-batch` request. The token is valid for 5 minutes after the request. If you repeat the `start-build-batch` request with the same token, but change a parameter, CodeBuild returns a parameter mismatch error.

4. If you run the **start-build-batch** command with the `--generate-cli-skeleton` option, JSON-formatted data is output to the *<json-file>* file. This file is similar to the skelton produced by the **start-build** command, with the addition of the following object. For more information about the common objects, see Run a build (AWS CLI).

   Modify this file to add any build overrides, and save your results.

```
"buildBatchConfigOverride": {
  "combineArtifacts": combineArtifacts,
  "restrictions": {
    "computeTypesAllowed": [
      allowedComputeTypes
    ],
    "maximumBuildsAllowed": maximumBuildsAllowed
  },
  "serviceRole": "batchServiceRole",
  "timeoutInMins": batchTimeout
}
```

   The `buildBatchConfigOverride` object is a ProjectBuildBatchConfig structure that contains the batch build configuration overides for this build.

   combineArtifacts

   A boolean that specifies if the build artifacts for the batch build should be combined into a single artifact location.

allowedComputeTypes

An array of strings that specify the compute types that are allowed for the batch build. See Build environment compute types for these values.

maximumBuildsAllowed

Specifies the maximum number of builds allowed.

batchServiceRole

Specifies the service role ARN for the batch build project.

batchTimeout

Specifies the maximum amount of time, in minutes, that the batch build must be completed in.

5. Switch to the directory that contains the file you just saved, and run the `start-build-batch` command again.

```
aws codebuild start-build-batch --cli-input-json file://start-build.json
```

6. If successful, the JSON representation of a BuildBatch object appears in the console output. See the StartBuildBatch Response Syntax for an example of this data.

Start running builds automatically (AWS CLI)

If your source code is stored in a GitHub or a GitHub Enterprise Server repository, you can use GitHub webhooks to have AWS CodeBuild rebuild your source code whenever a code change is pushed to the repository.

Run the **create-webhook** command as follows:

```
aws codebuild create-webhook --project-name <project-name>
```

<project-name> is the name of the build project that contains the source code to be rebuilt.

For GitHub, information similar to the following appears in the output:

```
{
  "webhook": {
    "url": "<url>"
```

```
  }
}
```

<url> is the URL to the GitHub webhook.

For GitHub Enterprise Server, information similar to the following appears in the output:

1. Copy the secret key and payload URL from the output. You need them to add a webhook in GitHub Enterprise Server.
2. In GitHub Enterprise Server, choose the repository where your CodeBuild project is stored. Choose **Settings**, choose **Hooks & services**, and then choose **Add webhook**.
3. Enter the payload URL and secret key, accept the defaults for the other fields, and then choose **Add webhook**.

Stop running builds automatically (AWS CLI)

If your source code is stored in a GitHub or a GitHub Enterprise Server repository, you can set up GitHub webhooks to have AWS CodeBuild rebuild your source code whenever a code change is pushed to the repository. For more information, see Start running builds automatically (AWS CLI).

If you have enabled this behavior, you can turn it off by running the `delete-webhook` command as follows:

```
aws codebuild delete-webhook --project-name <project-name>
```

- where *<project-name>* is the name of the build project that contains the source code to be rebuilt.

If this command is successful, no information and no errors appear in the output.

> **Note**
>
> This deletes the webhook from your CodeBuild project only. You should also delete the webhook from your GitHub or GitHub Enterprise Server repository.

Run a build (AWS SDKs)

To use CodePipeline to run a build with AWS CodeBuild, skip these steps and follow the instructions in Use AWS CodeBuild with AWS CodePipeline to test code and run builds instead.

For information about using CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Run builds on AWS Lambda compute

AWS Lambda compute offers optimized start-up speeds for your builds. AWS Lambda supports faster builds due to a lower start-up latency. AWS Lambda also automatically scales, so builds aren't waiting in queue to run. However, there are some use-cases which AWS Lambda does not support, and if they impact you, use the EC2 compute. For more information, see Limitations of AWS Lambda compute.

Topics

Which tools and runtimes will be included in the curated runtime environment docker images which run on AWS Lambda?

AWS Lambda supports the following tools: AWS CLI v2, AWS SAM CLI, git, go, Java, Node.js, Python, pip, Ruby, and .NET.

What if the curated image doesn't include the tools I need?

If the curated image doesn't include the tools you need, you can provide a custom environment Docker image that includes the necessary tools.

> **Note**
>
> Lambda does not support functions that use multi-architecture container images. For more information, see Create a Lambda function using a container image in the *AWS Lambda Developer Guide*.

Note that you require the following Amazon ECR permissions to use custom images for Lambda compute:

```
{
  "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"
    }
  ]
}
```

Also note that `curl` or `wget` must be installed in order to use custom images.

Which regions support AWS Lambda compute in CodeBuild?

In CodeBuild, AWS Lambda compute is supported in the following AWS Regions: US East (N. Virginia), US East (Ohio), US West (Oregon), Asia Pacific (Mumbai), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Europe (Frankfurt), Europe (Ireland), and South America (São Paulo). For more information about AWS Regions where CodeBuild is available, see AWS Services by Region.

Limitations of AWS Lambda compute

There are some use-cases which AWS Lambda does not support, and if they impact you, use the EC2 compute:

- AWS Lambda doesn't support tools that require root permissions. For tools such as `yum` or `rpm`, use the EC2 compute type or other tools that don't require root permissions.
- AWS Lambda doesn't support Docker builds or runs.
- AWS Lambda doesn't support writing to files outside `/tmp`. The included package managers are configured to use the `/tmp` directory by default for downloading and referencing packages.
- AWS Lambda doesn't support the environment type `LINUX_GPU_CONTAINER` and isn't supported on Windows Server Core 2019.
- AWS Lambda doesn't support caching, custom build timeouts, queue timeout, build badges, privileged mode, custom runtime environments, or runtimes longer than 15 minutes.
- AWS Lambda doesn't support VPC connectivity, a fixed range of CodeBuild source IP addresses, EFS, installing certificates, or SSH access with Session Manager.

Deploy a Lambda function using AWS SAM with CodeBuild Lambda Java

The AWS Serverless Application Model (AWS SAM) is an open-source framework for building serverless applications. For more information, see the AWS Serverless Application Model repository on GitHub. The following Java sample uses Gradle to build and test a AWS Lambda function. After which, the AWS SAM CLI is used to deploy the AWS CloudFormation template and deployment bundle. By using CodeBuild Lambda, the build, test, and deployment steps are all handled automatically, allowing for infrastructure to be quickly updated without manual intervention in a single build.

Set up your AWS SAM repository

Create an AWS SAM `Hello World` project using the AWS SAM CLI.

To create your AWS SAM Project

1. Follow the instructions in the *AWS Serverless Application Model Developer Guide* for Installing the AWS SAM CLI on your local machine.
2. Run `sam init` and select the following project configuration.

```
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. Upload the AWS SAM project folder to a supported source repository. For a list of supported source types, see ProjectSource.

Create a CodeBuild Lambda Java project

Create an AWS CodeBuild Lambda Java project and set up the IAM permissions needed for the build.

To create your CodeBuild Lambda Java project

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.
3. In **Project name**, enter a name for this build project. Build project names must be unique across each AWS account. You can also include an optional description of the build project to help other users understand what this project is used for.

4. In **Source**, select the source repository where your AWS SAM project is located.
5. In **Environment**:
   - For **Compute**, select **Lambda**.
   - For **Runtime(s)**, select **Java**.
   - For **Image**, select **aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto21**.
   - For **Service role**, leave **New service role** selected. Make a note of the **Role name**. This will be required when you update the project's IAM permissions later in this sample.
6. Choose **Create build project**.
7. Open the IAM console at https://console.aws.amazon.com/iam/.
8. In the navigation pane, choose **Roles** and select the service role associated with your project. You can find your project role in CodeBuild by selecting your build project, choosing **Edit**, **Environment**, and then **Service role**.
9. Choose the **Trust relationships** tab, and then choose **Edit trust policy**.
10. Add the following inline policy to your IAM role. This will be used to deploy your AWS SAM infrastructure later on. For more information, see Adding and removing IAM identity permissions in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "lambda:*",
                "iam:*",
                "apigateway:*",
                "s3:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Set up the project buildspec

In order to build, test, and deploy your Lambda function, CodeBuild reads and executes build commands from a buildspec.

To set up your project buildspec

1. In the CodeBuild console, select your build project, then choose **Edit** and **Buildspec**.
2. In **Buildspec**, choose **Insert build commands** and then **Switch to editor**.
3. Delete the pre-filled build commands and paste in the following buildspec.

```
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. Choose **Update buildspec**.

Deploy your AWS SAM Lambda infrastructure

Use CodeBuild Lambda to automatically deploy your Lambda infrastructure

To deploy your Lambda infrastructure

1. Choose **Start build**. This will automatically build, test, and deploy your AWS SAM application to AWS Lambda using AWS CloudFormation.
2. Once the build has finished, navigate to the AWS Lambda console and search for your new Lambda function under the AWS SAM project name.

3. Test your Lambda function by selecting **API Gateway** under the **Function** overview, then clicking the **API endpoint** URL. You should see a page open with the message `"message": "hello world"`.

Clean up your infrastructure

To avoid further charges for resources you used during this tutorial, delete the resources created by your AWS SAM template and CodeBuild.

To clean up your infrastructure

1. Navigate to the AWS CloudFormation console and select the `aws-sam-cli-managed-default`.
2. In **Resources**, empty the deployment bucket `SamCliSourceBucket`.
3. Delete the `aws-sam-cli-managed-default` stack.
4. Delete the AWS CloudFormation stack associated with your AWS SAM project. This stack should have the same name as your AWS SAM project.
5. Navigate to the CloudWatch console and delete the CloudWatch log groups associated with your CodeBuild project.
6. Navigate to the CodeBuild console and delete your CodeBuild project by choosing **Delete build project**.

Create a single page React app with CodeBuild Lambda Node.js

Create React App is a way to create single-page React applications. The following Node.js sample uses Node.js to build the source artifacts from Create React App and returns the build artifacts.

Set up your source repository and artifacts bucket

Create a source repository for your project using yarn and Create React App.

To set up the source repository and artifacts bucket

1. On your local machine, run `yarn create react-app` *`<app-name>`* to create a simple React app.
2. Upload the React app project folder to a supported source repository. For a list of supported source types, see ProjectSource.

Create a CodeBuild Lambda Node.js project

Create an AWS CodeBuild Lambda Node.js project.

To create your CodeBuild Lambda Node.js project

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.
3. In **Project name**, enter a name for this build project. Build project names must be unique across each AWS account. You can also include an optional description of the build project to help other users understand what this project is used for.
4. In **Source**, select the source repository where your AWS SAM project is located.
5. In **Environment**:
   - For **Compute**, select **Lambda**.
   - For **Runtime(s)**, select **Node.js**.
   - For **Image**, select **aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs20**.
6. In **Artifacts**:
   - For **Type**, select **Amazon S3**.
   - For **Bucket name**, select the project artifacts bucket you created earlier.
   - For **Artifacts packaging**, select **Zip**.
7. Choose **Create build project**.

Set up the project buildspec

In order to build your React app, CodeBuild reads and executes build commands from a buildspec file.

To set up your project buildspec

1. In the CodeBuild console, select your build project, then choose **Edit** and **Buildspec**.
2. In **Buildspec**, choose **Insert build commands** and then **Switch to editor**.
3. Delete the pre-filled build commands and paste in the following buildspec.

```
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. Choose **Update buildspec**.

Build and run your React app

Build the React app on CodeBuild Lambda, download the build artifacts, and run the React app locally.

To build and run your React app

1. Choose **Start build**.
2. Once the build has finished, navigate to your Amazon S3 project artifacts bucket and download the React app artifact.
3. Unzip the React build artifact and `run npm install -g serve && serve -s build` in the project folder.

4. The `serve` command will serve the static site on a local port and print output to your terminal. You can visit the localhost URL under `Local:` in the terminal output to view your React app.

To learn more about how to handle deployment for a React based server, see Create React App Deployment.

Clean up your infrastructure

To avoid further charges for resources you used during this tutorial, delete the resources created for your CodeBuild project.

To clean up your infrastructure

1. Delete your project artifacts Amazon S3 bucket
2. Navigate to the CloudWatch console and delete the CloudWatch log groups associated with your CodeBuild project.
3. Navigate to the CodeBuild console and delete your CodeBuild project by choosing **Delete build project**.

Update a Lambda function configuration with CodeBuild Lambda Python

The following Python sample uses Boto3 and CodeBuild Lambda Python to update a Lambda function's configuration. This sample can be extended to manage other AWS resources programmatically. For more information, see Boto3 documentation.

Prerequisites

Create or find a Lambda function in your account.

This sample assumes that you have already created a Lambda function in your account and will use CodeBuild to update the Lambda function's environment variables. For more information on setting up a Lambda function through CodeBuild, see the Deploy a Lambda function using AWS SAM with CodeBuild Lambda Java sample or visit AWS Lambda.

Set up your source repository

Create a source repository to store your Boto3 python script.

To set up the source repository

1. Copy the following python script to a new file called `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. Upload the python file to a supported source repository. For a list of supported source types, see ProjectSource.

Create a CodeBuild Lambda Python project

Create a CodeBuild Lambda Python project.

To create your CodeBuild Lambda Java project

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.

2. If a CodeBuild information page is displayed, choose **Create build project**. Otherwise, on the navigation pane, expand **Build**, choose **Build projects**, and then choose **Create build project**.
3. In **Project name**, enter a name for this build project. Build project names must be unique across each AWS account. You can also include an optional description of the build project to help other users understand what this project is used for.
4. In **Source**, select the source repository where your AWS SAM project is located.
5. In **Environment**:
   - For **Compute**, select **Lambda**.
   - For **Runtime(s)**, select **Python**.
   - For **Image**, select **aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12**.
   - For **Service role**, leave **New service role** selected. Make a note of the **Role name**. This will be required when you update the project's IAM permissions later in this sample.
6. Choose **Create build project**.
7. Open the IAM console at https://console.aws.amazon.com/iam/.
8. In the navigation pane, choose **Roles** and select the service role associated with your project. You can find your project role in CodeBuild by selecting your build project, choosing **Edit**, **Environment**, and then **Service role**.
9. Choose the **Trust relationships** tab, and then choose **Edit trust policy**.
10. Add the following inline policy to your IAM role. This will be used to deploy your AWS SAM infrastructure later on. For more information, see Adding and removing IAM identity permissions in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "UpdateLambdaPermissions",
            "Effect": "Allow",
            "Action": [
                "lambda:UpdateFunctionConfiguration"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

```
}
```

Set up the project buildspec

In order to update the Lambda function, the script reads environment variables from the buildspec to find the Lambda function's name, environment variable name, and environment variable value.

To set up your project buildspec

1. In the CodeBuild console, select your build project, then choose **Edit** and **Buildspec**.
2. In **Buildspec**, choose **Insert build commands** and then **Switch to editor**.
3. Delete the pre-filled build commands and paste in the following buildspec.

```
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. Choose **Update buildspec**.

Update your Lambda configuration

Use CodeBuild Lambda Python to automatically update your Lambda function's configuration.

To update your Lambda function's configuration

1. Choose **Start build**.
2. Once the build has finished, navigate to your Lambda function.
3. Select **Configuration** and then **Environment** variables. You should see a new environment variable with key FEATURE_ENABLED and value `true`.

Clean up your infrastructure

To avoid further charges for resources you used during this tutorial, delete the resources created for your CodeBuild project.

To clean up your infrastructure

1. Navigate to the CloudWatch console and delete the CloudWatch log groups associated with your CodeBuild project.
2. Navigate to the CodeBuild console and delete your CodeBuild project by choosing **Delete build project**.
3. If you created a Lambda function for the purpose of this sample, choose **Actions** and **Delete function** to clean up your Lambda function.

Extensions

If you want to extend this sample to manage other AWS resources using AWS CodeBuild Lambda Python:

- Update the Python script to modify the new resources using Boto3.
- Update the IAM role associated with your CodeBuild project to have permissions for the new resources.
- Add any new environment variables associated with the new resources to your buildspec.

Run builds on reserved capacity fleets

CodeBuild offers the following compute fleets:

- On-demand fleets
- Reserved capacity fleets

With on-demand fleets, CodeBuild provides compute for your builds. The machines are destroyed when the build finishes. On-demand fleets are fully managed, and includes automatic scaling capabilities to handle spikes in demand.

> **Note**
>
> On-demand fleets do not support macOS.

CodeBuild also offers reserved capacity fleets which contain instances powered by Amazon EC2 that are maintained by CodeBuild. With reserved capacity fleets, you configure a set of dedicated instances for your build environment. These machines remain idle, ready to process builds or tests immediately and reduces build durations. With reserved capacity fleets, your machines are always running and will continue to incur costs as long they're provisioned.

> **Important**
>
> Regardless of how long you run an instance for, reserved capacity fleets incur an initial charge per instance, after which there may be additional associated costs. For more information, see https://aws.amazon.com/codebuild/pricing/.

Topics

Create a reserved capacity fleet

Use the following instructions to create a reserved capacity fleet.

To create a reserved capacity fleet

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Compute fleets**, and then choose **Create Fleet**.
3. In the **Compute fleet name** text field, enter a name for your fleet.
4. From the **Operating system** drop-down menu, choose the operating system.
5. From the **Architecture** drop-down menu, choose the architecture.
6. (Optional) Select **Use instance running mode - optional** to run on an Amazon EC2 instance directly instead of a Docker container. Then choose a **Major version** and **Minor version**.
7. (Optional) In **Additional configuration** do the following:

   - Select **Configure VPC - optional** to connect your fleet to a VPC to access private resources during usage.
     - From the **VPC** drop-down menu, select a VPC that your CodeBuild fleet will access.
     - From the **Subnets** drop-down menu, select the subnets that CodeBuild should use to set up your VPC configuration.
     - From the **Security groups** drop-down menu, select the security groups that CodeBuild should use to work with your VPC.
     - In the **Fleet Service Role** field, choose an existing service role.

     > **Note**
     >
     > Make sure that your fleet role has the necessary permissions. For more information, see Allow a user to add a permission policy for a fleet service role.

     - If you chose the Amazon Linux operating system, select **Define proxy configurations - optional** to apply network access control for your reserved capacity instances.
     - For **Default behavior**, choose to allow or deny outgoing traffic to all destinations by default.
     - For **Proxy rules**, choose **Add proxy rule** to specify destination domains or IPs to allow or deny network access control to.
   - Select **Configure custom AMI - optional** to use a custom Amazon Machine Image (AMI).
     - From the **AMI** drop-down menu, select a an Amazon Machine Image (AMI) for your fleet.
     - In the **Fleet Service Role** field, choose an existing service role.

> **Note**
>
> Make sure that your fleet role has the necessary permissions. For more information, see Allow a user to add a permission policy for a fleet service role.

8. In **Capacity configuration**, from **Compute selection mode**, choose one of the following:

   - If you choose **Guided selection**, do the following:
     - For **Compute**, choose the type of instances included in this fleet.
     - In the **Capacity** text field, enter the minimum number of instances in the fleet.
     - (Optional) In **Additional configuration** do the following:
       - Select **Configure scaling - optional** to automatically scale your fleet based on this configuration. From the **Scaling mode - optional** drop-down menu, choose the behavior when demand exceeds the fleet capacity.
   - If you choose **Custom instance**, do the following:
     - From the **Compute instance type** drop-down menu, select the type of instances included in this fleet.
     - In the **Additional EBS volume size - optional** text field, enter the volume additional to the 64GB of disk space provided.
     - In the **Capacity** text field, enter the minimum number of instances in the fleet.
     - (Optional) In **Additional configuration** do the following:
       - Select **Configure scaling - optional** to automatically scale your fleet based on this configuration. From the **Scaling mode - optional** drop-down menu, choose the behavior when demand exceeds the fleet capacity.
9. Choose **Create compute fleet**.
10. After the compute fleet is created, create a new CodeBuild project or edit an existing one. From **Environment**, choose **Reserved capacity** under **Provisioning model**, and then choose the specified fleet under **Fleet name**.

Best practices

When using reserved capacity fleets, we recommend that you follow these best practices.

- We recommend using source cache mode to help improve the build performance by caching the source.
- We recommend using Docker layer caching to help improve the build performance by caching existing Docker layers.

Can I share a reserved capacity fleet across multiple CodeBuild projects?

Yes, you can maximize the utilization of a fleet's capacity by using it across multiple projects.

> **Important**
>
> When using the reserved capacity feature, data cached on fleet instances, including source files, Docker layers, and cached directories specified in the buildspec, can be accessible to other projects within the same account. This is by design and allows projects within the same account to share fleet instances.

How does attribute-based compute work?

If you choose `ATTRIBUTE_BASED_COMPUTE` as your fleet's `computeType`, you can specify the attributes in a new field called `computeConfiguration`. These attributes include vCPUs, memory, disk space, and the `machineType`. This `machineType` is either `GENERAL` or `NVME`. After specifying one or some of the available attributes, CodeBuild will choose a compute type from the available supported instance types as the finalized `computeConfiguration`.

> **Note**
>
> CodeBuild will choose the cheapest instance that match all input requirements. The chosen instances' memory, vCPUs, and disk space will all be greater than or equal to the input requirements. You can check the resolved `computeConfiguration` in the created or updated fleet.

If you input a `computeConfiguration` that is not possible to satisfy in CodeBuild, you'll receive a validation exception. Also note that on-demand fleet overflow behavior will be overridden to queue behavior if the `computeConfiguration` is not available for on-demand.

Can I manually specify an Amazon EC2 instance for my fleet?

Yes, you can directly input your desired Amazon EC2 instance in the console by selecting **Custom instance** or by configuring the API parameter, `InstanceType`. This field is used in the following APIs: CreateFleet, UpdateFleet, CreateProject, UpdateProject and StartBuild. For more information, see Compute instance type.

Which regions support reserved capacity fleets?

Reserved capacity Amazon Linux and Windows fleets are supported in the following AWS Regions: US East (N. Virginia), US East (Ohio), US West (Oregon), Asia Pacific (Mumbai), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Europe (Frankfurt), Europe (Ireland), and South America (São Paulo). For more information about AWS Regions where CodeBuild is available, see AWS Services by Region.

Reserved capacity macOS Medium fleets are supported in the following AWS Regions: US East (N. Virginia), US East (Ohio), US West (Oregon), Asia Pacific (Sydney), and Europe (Frankfurt). Reserved capacity macOS Large fleets are supported in the following AWS Regions: US East (N. Virginia), US East (Ohio), US West (Oregon), and Asia Pacific (Sydney).

How do I configure a reserved capacity macOS fleet?

To configure a reserved capacity macOS fleet

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Compute fleets**, and then choose **Create Fleet**.
3. In the **Compute fleet name** text field, enter a name for your fleet.
4. From the **Operating system** drop-down menu, choose **macOS**.
5. In the **Compute** field, choose one of the following compute machine types: **Apple M2, 24 GB memory, 8 vCPUs** or **Apple M2, 32 GB memory, 12 vCPUs**.
6. In the **Capacity** text field, enter the minimum number of instances in the fleet.
7. (Optional) To use a custom image for your fleet, see How do I configure a custom Amazon Machine Image (AMI) for a reserved capacity fleet? to ensure that your Amazon Machine Image (AMI) has the required prerequisites.
8. (Optional) To configure a VPC with your fleet, in **Additional configuration** do the following:

- From the **VPC - optional** drop-down menu, select a VPC that your CodeBuild fleet will access.
- From the **Subnets** drop-down menu, select the subnets that CodeBuild should use to set up your VPC configuration.
- From the **Security groups** drop-down menu, select the security groups that CodeBuild should use to work with your VPC.
- In the **Fleet service role** field, choose an existing service role.

> **Note**
>
> Make sure that your fleet role has the necessary permissions. For more information, see Allow a user to add a permission policy for a fleet service role.

9. Choose **Create compute fleet** and wait for the fleet instance to launch. Once launched the capacity will be *n*/*n*, where *n* is the capacity provided.
10. After the compute fleet has launched, create a new CodeBuild project or edit an existing one. From **Environment**, choose **Reserved capacity** under **Provisioning model**, and then choose the specified fleet under **Fleet name**.

How do I configure a custom Amazon Machine Image (AMI) for a reserved capacity fleet?

To configure a custom Amazon Machine Image (AMI) for a reserved capacity fleet

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Compute fleets**, and then choose **Create Fleet**.
3. In the **Compute fleet name** text field, enter a name for your fleet.
4. Choose **Custom image** for your fleet and ensure that your Amazon Machine Image (AMI) has the following prerequisites:

   - If your environment type is `MAC_ARM`, make sure that your AMI **Architecture** is 64-bit `Mac-Arm`.
   - If your environment type is `LINUX_EC2`, make sure that your AMI **Architecture** is 64-bit `x86`.

- If your environment type is `ARM_EC2`, make sure that your AMI **Architecture** is 64-bit `Arm`.
- If your environment type is `WINDOWS_EC2`, make sure that your AMI **Architecture** is 64-bit `x86`.
- The AMI allows the CodeBuild service **Organization ARN**. For a list of Organization ARNs, see Amazon Machine Images (AMI).
- If the AMI is encrypted with a AWS KMS key, the AWS KMS key must also allow the CodeBuild service **Organization ID**. For a list of Organization IDs, see Amazon Machine Images (AMI). For more information on AWS KMS keys, see Allow organizations and OUs to use a KMS key in the *Amazon EC2 User Guide*. To give CodeBuild organization permission to use a KMS key, add the following statement to the key policy:

```
{
    "Sid": "Allow access for organization root",
    "Effect": "Allow",
    "Principal": "*",
    "Action": [
        "kms:Describe*",
        "kms:List*",
        "kms:Get*",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:PrincipalOrgID": "o-123example"
        }
    }
}
```

- In the **Fleet service role** field, grant the following Amazon EC2 permissions:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
               "ec2:DescribeImages",
               "ec2:DescribeSnapshots"
            ],
            "Resource": "*"
         }
    ]
}
```

Limitations of reserved capacity fleets

There are some use-cases which reserved capacity fleets do not support, and if they impact you, use on-demand fleets instead:

- Reserved capacity fleets don't support build utilization metrics.
- Reserved capacity macOS fleets don't support debug session.

For more information on limits and quotas, see Compute fleets.

Reserved capacity fleet properties

A reserved capacity fleet contains the following properties. For more information about reserved capacity fleets, see Run builds on reserved capacity fleets.

Operating system

The operating system. The following operating systems are available:

- Amazon Linux
- macOS
- Windows Server 2019
- Windows Server 2022

Architecture

The processor architecture. The following architectures are available:

- x86_64
- Arm64

Environment type

The environment types available when **Amazon Linux** is selected. The following environment types are available:

- Linux EC2
- Linux GPU

Compute instance type

The compute configurations for fleet instances.

Guided selection

Specify different compute types by selecting vCPU, memory and disk space settings. For information about compute type availability by region, see About reserved capacity environment types.

Custom instance

Manually specify the desired instance type.

Capacity

The initial number of machines allocated to the fleet, which defines the number of builds that can run in parallel.

Overflow behavior

Defines the behavior when the number of builds exceeds the fleet capacity.

On-demand

Overflow builds run on CodeBuild on-demand.

> **Note**
>
> If you choose to set your overflow behavior to on-demand while creating a VPC-connected fleet, make sure that you add the required VPC permissions to your project service role. For more information, see Example policy statement to allow CodeBuild access to AWS services required to create a VPC network interface.

> **Important**
>
> If you choose to set your overflow behavior to on-demand, note that overflow builds will be billed separately, similar to on-demand Amazon EC2. For more information, see https://aws.amazon.com/codebuild/pricing/.

Queue

Build runs are placed in a queue until a machine is available. This limits additional costs because no additional machines are allocated.

Amazon Machine Images (AMI)

The Amazon Machine Image (AMI) properties for your fleet. The following properties are supported by CodeBuild:

AWS Regions	Organization ARN	Organization ID
us-east-1	arn:aws:organizations::851725618577:organization/o-c6wcu152r1	o-c6wcu152r1
us-east-2	arn:aws:organizations::992382780434:organization/o-seufr2suvq	o-seufr2suvq
us-west-2	arn:aws:organizations::381491982620:organization/o-0412o99a4r	o-0412o99a4r
ap-northeast-1	arn:aws:organizations::891376993293:organization/o-b6k3sjqavm	o-b6k3sjqavm

AWS Regions	Organization ARN	Organization ID
ap-south-1	arn:aws:organizations::891376924779:organization/o-krtah1lkeg	o-krtah1lkeg
ap-southeast-1	arn:aws:organizations::654654522137:organization/o-mcn8uvc3tp	o-mcn8uvc3tp
ap-southeast-2	arn:aws:organizations::767398067170:organization/o-6crt0f6bu4	o-6crt0f6bu4
eu-central-1	arn:aws:organizations::590183817084:organization/o-lb2lne3te6	o-lb2lne3te6
eu-west-1	arn:aws:organizations::891376938588:organization/o-ullrrg5qf0	o-ullrrg5qf0
sa-east-1	arn:aws:organizations::533267309133:organization/o-db63c45ozw	o-db63c45ozw

Additional configuration

VPC - optional

The VPC that your CodeBuild fleet will access. For more information, see Use AWS CodeBuild with Amazon Virtual Private Cloud.

Subnets

The VPC subnets that CodeBuild uses to set up your VPC configuration. Note that reserved capacity fleets support only one subnet in a single Availablity Zone. Also, ensure that your subnets include a NAT gateway.

Security groups

The VPC security groups that CodeBuild uses with your VPC. Ensure that your security groups allow outbound connections.

Fleet Service Role

Defines the service role for your fleet from an existing service role in your account.

Define proxy configurations - optional

Proxy configurations that apply network access control to your reserved capacity instances. For more information, see Use AWS CodeBuild with a managed proxy server.

> **Note**
>
> Proxy configurations don't support VPC, Windows, or MacOS.

Default behavior

Defines the behavior of outgoing traffic.

Allow

Allows outgoing traffic to all destinations by default.

Deny

Denies outgoing traffic to all destinations by default.

Proxy rules

Specifies destination domains or IPs to allow or deny network access control to.

Reserved capacity samples with AWS CodeBuild

These samples can be used to experiment with reserved capacity fleets in CodeBuild.

Topics

Caching with reserved capacity sample

A cache can store reusable pieces of your build environment and use them across multiple builds. This sample demonstrated how to enable caching within your build project using reserved capacity. For more information, see Cache builds to improve performance.

You can start by specifying one or more cache modes in your project settings:

```
Cache:
        Type: LOCAL
        Modes:
          - LOCAL_CUSTOM_CACHE
          - LOCAL_DOCKER_LAYER_CACHE
          - LOCAL_SOURCE_CACHE
```

> **Note**
>
> Make sure to enable privileged mode in order to use Docker layer cache.

Your project buildspec settings should look like the following:

```
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'
```

You can start by running a build with the new project to seed the cache. Once that's complete, you should start another build with an overriding buildspec, similar to the following:

```
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'
```

Run builds in batches

You can use AWS CodeBuild to run concurrent and coordinated builds of a project with batch builds.

Topics

- Batch report mode
- More information

Security role

Batch builds introduce a new security role in the batch configuration. This new role is required as CodeBuild must be able to call the `StartBuild`, `StopBuild`, and `RetryBuild` actions on your behalf to run builds as part of a batch. Customers should use a new role, and not the same role they use in their build, for two reasons:

- Giving the build role `StartBuild`, `StopBuild`, and `RetryBuild` permissions would allow a single build to start more builds via the buildspec.
- CodeBuild batch builds provide restrictions that restrict the number of builds and compute types that can be used for the builds in the batch. If the build role has these permissions, it is possible the builds themselves could bypass these restrictions.

Batch build types

CodeBuild supports the following batch build types:

Batch build types

- Build graph
- Build list
- Build matrix
- Build fanout

Build graph

A build graph defines a set of tasks that have dependencies on other tasks in the batch.

The following example defines a build graph that creates a dependency chain.

```
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
    - identifier: build4
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build5
      env:
        fleet: fleet_name
```

In this example:

- `build1` runs first because it has no dependencies.
- `build2` has a dependency on `build1`, so `build2` runs after `build1` completes.
- `build3` has a dependency on `build2`, so `build3` runs after `build2` completes.

For more information about the build graph buildspec syntax, see `batch/build-graph`.

Build list

A build list defines a number of tasks that run in parallel.

The following example defines a build list. The `build1` and `build2` builds will run in parallel.

```
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
    - identifier: build3
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build4
      env:
        fleet: fleet_name
    - identifier: build5
      env:
        compute-type: GENERAL_LINUX_XLAGRE
```

For more information about the build list buildspec syntax, see batch/build-list.

Build matrix

A build matrix defines tasks with different configurations that run in parallel. CodeBuild creates a separate build for each possible configuration combination.

The following example shows a build matrix with two buildspec files and three values for an environment variable.

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
        variables:
          MY_VAR:
            - VALUE1
            - VALUE2
```

```
            - VALUE3
```

In this example, CodeBuild creates six builds:

- `matrix1.yml` with `$MY_VAR=VALUE1`
- `matrix1.yml` with `$MY_VAR=VALUE2`
- `matrix1.yml` with `$MY_VAR=VALUE3`
- `matrix2.yml` with `$MY_VAR=VALUE1`
- `matrix2.yml` with `$MY_VAR=VALUE2`
- `matrix2.yml` with `$MY_VAR=VALUE3`

Each build will have the following settings:

- `ignore-failure` set to `false`
- `env/type` set to `LINUX_CONTAINER`
- `env/image` set to `aws/codebuild/amazonlinux-x86_64-standard:4.0`
- `env/privileged-mode` set to `true`

These builds run in parallel.

For more information about the build matrix buildspec syntax, see `batch/build-matrix`.

Build fanout

A build fanout defines a task that will be split into multiple builds in the batch. This can be used for running tests in parallel. CodeBuild creates a separate build for each shard of test cases based on the value set in `parallelism` field.

The following example defines a build fanout that creates five builds that run in parallel.

```
version: 0.2

batch:
   fast-fail: false
   build-fanout:
     parallelism: 5
     ignore-failure: false
```

```
phases:
  install:
    commands:
      - npm install
   build:
    commands:
      - mkdir -p test-results
      - cd test-results
      - |
        codebuild-tests-run \
         --test-command 'npx jest --runInBand --coverage' \
         --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
         --sharding-strategy 'equal-distribution'
```

In this example, assuming that there are 100 tests that needs to be run, CodeBuild creates five builds that each runs 20 tests in parallel.

For more information about the build graph buildspec syntax, see `batch/build-fanout`.

Batch report mode

If the source provider for your project is Bitbucket, GitHub, or GitHub Enterprise, and your project is configured to report build statuses to the source provider, you can select how you want your batch build statuses sent to the source provider. You can select to have the statuses sent as a single aggregate status report for the batch, or have the status of each build in the batch reported individually.

For more information, see the following topics:

- Batch configuration (create)
- Batch configuration (update)

More information

For more information, see the following topics:

- Batch build buildspec reference
- Batch configuration
- Run a batch build (AWS CLI)
- Stop batch builds in AWS CodeBuild

Execute parallel tests in batch builds

You can use AWS CodeBuild to execute parallel tests in batch builds. Parallel test execution is a testing approach where multiple test cases run simultaneously across different environments, machines, or browsers, rather than executing sequentially. This approach can significantly reduce overall test execution time and improve testing efficiency. In CodeBuild, you can split your tests across multiple environments and run them concurrently.

The key advantages of parallel test execution include:

1. **Reduced execution time** - Tests that would take hours sequentially can complete in minutes.
2. **Better resource utilization** - Makes efficient use of available computing resources.
3. **Earlier feedback** - Faster test completion means quicker feedback to developers.
4. **Cost-effective** - Saves both time and computing costs in the long run.

When implementing parallel test execution, two main approaches are commonly considered: separate environments and multithreading. While both methods aim to achieve concurrent test execution, they differ significantly in their implementation and effectiveness. Separate environments create isolated instances where each test suite runs independently, while multithreading executes multiple tests simultaneously within the same process space using different threads.

The key advantages of separate environments over multithreading include:

1. **Isolation** - Each test runs in a completely isolated environment, preventing interference between tests.
2. **Resource conflicts** - No competition for shared resources that often occurs in multithreading.
3. **Stability** - Less prone to race conditions and synchronization issues.
4. **Easier debugging** - When tests fail, it's simpler to identify the cause as each environment is independent.
5. **State management** - Easily manage shared state issues that plague multithreaded tests.
6. **Better scalability** - Can easily add more environments without complexity.

Topics

- Enable parallel test execution in batch builds
- Use the codebuild-tests-run CLI command
- Use the codebuild-glob-search CLI command
- About test splitting
- Automatically merge individual build reports
- Parallel test execution for various test frameworks sample

Support in AWS CodeBuild

AWS CodeBuild provides robust support for parallel test execution through its batch build feature, specifically designed to leverage separate environment execution. This implementation aligns perfectly with the benefits of isolated testing environments.

Batch build with test distribution

CodeBuild's batch build functionality enables the creation of multiple build environments that run simultaneously. Each environment operates as a completely isolated unit, with its own compute resources, runtime environment, and dependencies. Through the batch build configuration, you can specify how many parallel environments they need and how tests should be distributed across them.

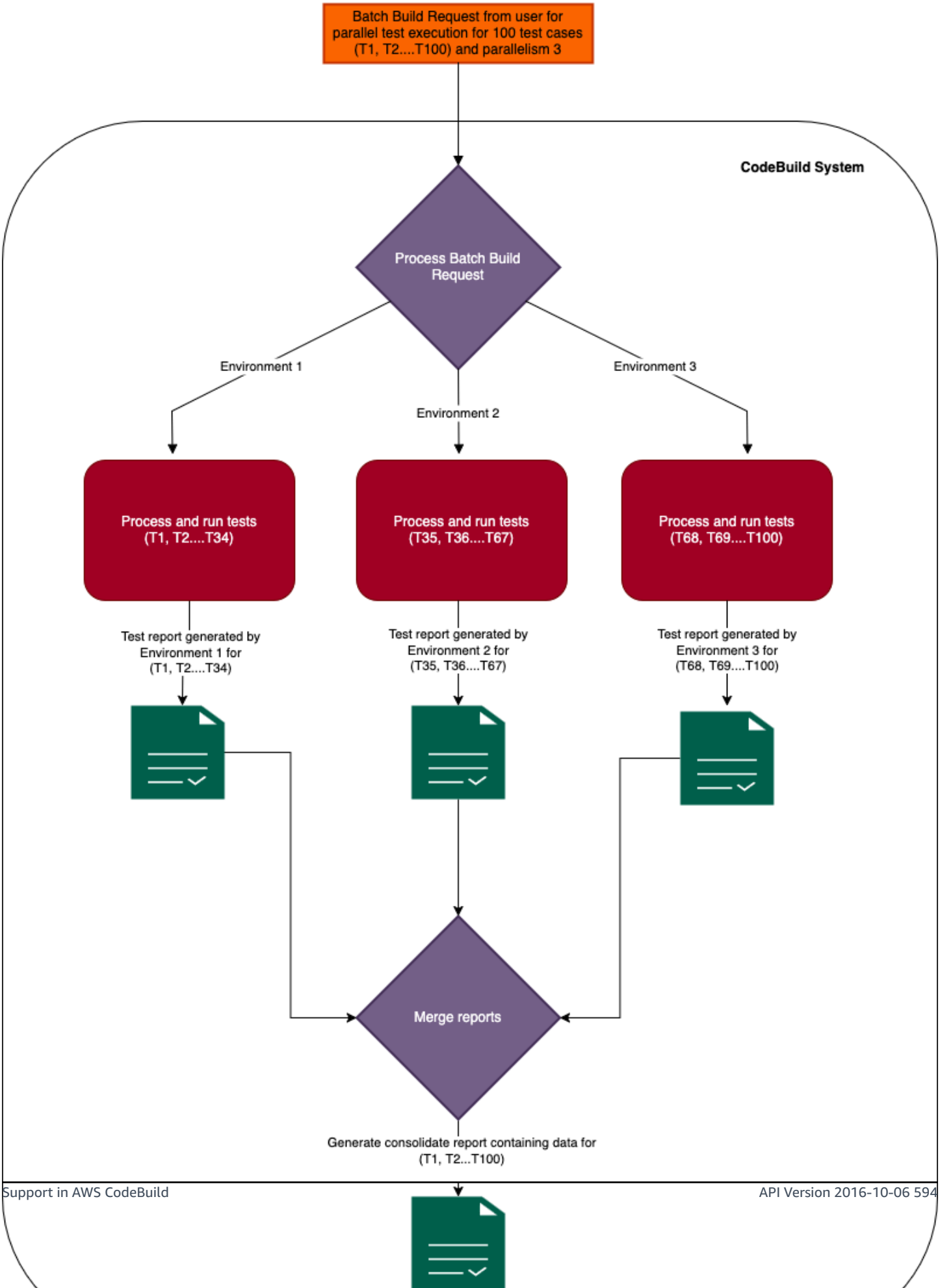
Test sharding CLI

CodeBuild includes a built-in test distribution mechanism through its CLI tool, `codebuild-tests-run`, which automatically divides tests into different environments.

Report aggregation

One of the key strengths of CodeBuild's implementation is its ability to handle test result aggregation seamlessly. While tests execute in separate environments, CodeBuild automatically collects and combines the test reports from each environment into a unified test report at the batch build level. This consolidation provides a comprehensive view of test results while maintaining the efficiency benefits of parallel execution.

The following is the diagram explains the complete concept of parallel test execution in AWS CodeBuild.


Batch Build Request from user for parallel test execution for 100 test cases (T1, T2....T100) and parallelism 3
CodeBuild System
Process Batch Build Request
Environment 1
Environment 2
Environment 3
Process and run tests (T1, T2....T34)
Process and run tests (T35, T36....T67)
Process and run tests (T68, T69....T100)
Test report generated by Environment 1 for (T1, T2....T34)
Test report generated by Environment 2 for (T35, T36....T67)
Test report generated by Environment 3 for (T68, T69....T100)
Merge reports
Generate consolidate report containing data for (T1, T2...T100)

Enable parallel test execution in batch builds

To run tests in parallel, update the batch build buildspec file to include the build-fanout field and the number of parallel builds to split the test suite in the `parallelism` field as shown below. The `parallelism` field specifies how many independent executors are setup to execute the test suite.

To run the tests in multiple parallel execution environments, set the `parallelism` field to a value greater than zero. In example below, `parallelism` is set to five, meaning CodeBuild starts five identical builds that executes a portion of the test suite in parallel.

You can use the codebuild-tests-run CLI command to split and run your tests. Your test files will be split up, and a portion of your tests run in each build. This reduces the overall time taken to run the full test suite. In the following example, tests will be split up into five and the split points are calculated based on name of the tests.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - npm install jest-junit --save-dev
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - |
        codebuild-tests-run \
         --test-command 'npx jest --runInBand --coverage' \
         --files-search "codebuild-glob-search '**/_tests_/**/*.test.js'" \
         --sharding-strategy 'equal-distribution'

  post_build:
    commands:
      - codebuild-glob-search '**/*.xml'
      - echo "Running post-build steps..."
```

```
      - echo "Build completed on `date`"

reports:
  test-reports:
    files:
      - '**/junit.xml'
    base-directory: .
    discard-paths: yes
    file-format: JUNITXML
```

If reports are configured for build-fanout build, then the test reports are generated for each build separately, which can be viewed under the **Reports** tab of the corresponding builds in the AWS CodeBuild console.

For more information on how to execute parallel tests in batch, see Parallel test execution for various test frameworks sample.

Use the `codebuild-tests-run` CLI command

AWS CodeBuild provides CLI that will take test command and test file location as input. The CLI with these input will split the tests into number of shards as specified in the `parallelism` field based on test file names. The assignment of test files to shard is decided by the sharding strategy.

```
codebuild-tests-run \
    --files-search "codebuild-glob-search '**/__tests__/*.js'" \
    --test-command 'npx jest --runInBand --coverage' \
    --sharding-strategy 'equal-distribution'
```

The following table describes the fields for the `codebuild-tests-run` CLI command.

Field name	Type	Required or optional	Definition
`test-command`	String	Required	This command is used for running the tests.
`files-search`	String	Required	This command gives a list of test files. You can use the AWS CodeBuild provided

Field name	Type	Required or optional	Definition
			codebuild-glob-search CLI command or any other file search tool of your choice. **Note** Ensure that the `files-search` command outputs file names, each separated by a new line.

Field name	Type	Required or optional	Definition
`sharding-strategy`	Enum	Optional	Valid values: `equal-distribution` (default), `stability` • `equal-distribution` : Shard test files evenly based on test file names. • `stability` : Shard test files using consistent hashing of the file names. For more information, see About test splitting.

The `codebuild-tests-run` CLI works first to identify the list of test files using the command provided in the `files-search` parameter. It then determines a subset of test files designated for the current shard (environment) using the specified sharding strategy. Finally, this subset of test files is formatted into a space-separated list and appended to the end of the command provided in the `test-command` parameter before being executed.

For test frameworks that don't accept space-separated lists, the `codebuild-tests-run` CLI provides a flexible alternative through the `CODEBUILD_CURRENT_SHARD_FILES` environment variable. This variable contains a newline-separated list of test file paths designated for the current build shard. By leveraging this environment variable, you can easily adapt to various test framework requirements, accommodating those that expect input formats different from space-separated lists. Moreover, you can also format the test file names as per need of test framework. The following is an example of the use of `CODEBUILD_CURRENT_SHARD_FILES` on Linux with

the Django framework. Here CODEBUILD_CURRENT_SHARD_FILES is used to get *dot notation* file paths supported by Django:

```
codebuild-tests-run \
    –files-search "codebuild-glob-search '/tests/test_.py'" \
    –test-command 'python3 manage.py test $(echo "$CODEBUILD_CURRENT_SHARD_FILES" | sed
 -E "s/\//__/g; s/\.py$//; s/__/./g")' \
    –sharding-strategy 'equal-distribution'
```

Note

Note that the CODEBUILD_CURRENT_SHARD_FILES environment variable can be used only inside the scope of the codebuild-tests-run CLI.
Also, if you are using CODEBUILD_CURRENT_SHARD_FILES inside test-command, put CODEBUILD_CURRENT_SHARD_FILES inside double quotes as shown in above example.

Use the codebuild-glob-search CLI command

AWS CodeBuild provides a built-in CLI tool called codebuild-glob-search that allows you to search for files in your working directory based on one or more glob patterns. This tool can be particularly useful when you want to run tests on specific files or directories within your project.

Usage

The codebuild-glob-search CLI has the following usage syntax:

```
codebuild-glob-search <glob_pattern1> [<glob_pattern2> ...]
```

- *<glob_pattern1>*, *<glob_pattern2>*, etc.: One or more glob patterns to match against the files in your working directory.
- *: Matches any sequence of characters (excluding path separators).
- **: Matches any sequence of characters (including path separators).

Note

Ensure that the glob string has quotes. To check the results of pattern-matching, use the echo command.

```
version: 0.2

phases:
  build:
    commands:
      - echo $(codebuild-glob-search '**/__tests__/*.js')
      - codebuild-glob-search '**/__tests__/*.js' | xargs -n 1 echo
```

Output

The CLI will output a newline-separated list of file paths that match the provided glob patterns. The file paths returned will be relative to the working directory.

If no files are found matching the provided patterns, the CLI will output a message indicating that no files were found.

Note that directories found due to any given pattern will be excluded from the search results.

Example

If you want to search only for files inside the tests directory and its subdirectories with a `.js` extension, you can use the following command with the `codebuild-glob-search` CLI:

```
codebuild-glob-search '**/__tests__/*.js'
```

This command will search for all files with a `.js` extension inside the `__tests__`directory and its subdirectories, as denoted by the pattern.

About test splitting

AWS CodeBuild's test splitting feature allows you to parallelize your test suite execution across multiple compute instances, reducing the overall test run time. This feature is enabled through the batch configuration in your CodeBuild project settings and the `codebuild-tests-run`utility in your buildspec file.

The tests are split based on the sharding strategy specified. CodeBuild provides two sharding strategies as specified below:

Equal-distribution

The `equal-distribution` sharding strategy divides the tests across parallel builds based on the alphabetical order of the test file names. This approach first sorts the test files and then employs a chunk-based method to distribute them, ensuring that similar files are grouped together for testing. It is recommended when dealing with a relatively small set of test files. While this method aims to allocate an approximately equal number of files to each shard, with a maximum difference of one, it does not guarantee stability. When test files are added or removed in subsequent builds, the distribution of existing files may change, potentially causing reassignment across shards.

Stability

The `stability` sharding strategy employs a consistent hashing algorithm to split tests among shards, ensuring that file distribution remains stable. When new files are added or removed, this approach ensures that the existing file-to-shard assignments remain largely unchanged. For large test suites, it is recommended to use the stability option to evenly distribute the tests across shards. This mechanism aims to provide a near-equal distribution, ensuring that each shard receives a similar number of files, with only minimal variance. While the stability strategy does not guarantee an ideal equal distribution, it offers a near-equal distribution that maintains consistency in file assignments across builds, even as files are added or removed.

To enable test splitting, you need to configure the batch section in your CodeBuild project settings, specifying the desired `parallelism` level and other relevant parameters. Additionally, you'll need to include the `codebuild-tests-run` utility in your buildspec file, along with the appropriate test commands and splitting method.

Automatically merge individual build reports

In fanout batch builds, AWS CodeBuild supports automatic merging of individual build reports into a consolidated batch-level report. This feature provides a comprehensive view of test results and code coverage across all builds within a batch.

How it works

When executing `fanout` batch builds, each individual build generates test reports. CodeBuild then automatically consolidates identical reports from different builds into a unified report, which is attached to the batch build. These consolidated reports are readily accessible through the

BatchGetBuildBatches API's `reportArns` field, and can also be viewed in the **Reports** tab of the console. This merging capability extends to auto-discovered reports as well.

Consolidated reports are created under report groups that are either specified in the buildspec or auto-discovered by CodeBuild. You can analyze trends of the merged reports directly under these report groups, providing valuable insights into the overall build performance and quality metrics across historical builds of the same build-batch project.

For each individual build within the batch, CodeBuild automatically creates separate report groups. These follow a specific naming convention, combining the batch build report group name with a suffix of `BuildFanoutShard<shard_number>`, where the `shard_number` represents the number of the shard in which the report group is created. This organization allows you to track and analyze trends at both the consolidated and individual build levels, providing flexibility in how you monitor and evaluate their build processes.

The batch-build report follows the same structure as individual build reports. The following key fields in the **Report** tab are specific to batch-build reports:

Batch build report status

The status of batch build reports follows specific rules depending on the report type:

- Test reports:
  - Succeeded: Status is set to succeeded when all individual build reports have succeeded.
  - Failed: Status is set to failed if any individual build report has failed.
  - Incomplete: Status is marked as incomplete if any individual build report is missing or has an incomplete status.
- Code coverage reports:
  - Complete: Status is set to complete when all individual build reports are complete.
  - Failed: Status is set to failed if any individual build report has failed.
  - Incomplete: Status is marked as incomplete if any individual build report is missing or has an incomplete status.

Test summary

The merged test report consolidates the following fields from all individual build reports:

- duration-in-nano-seconds: Maximum test duration time in nanoseconds among all individual build reports.

- total: The combined count of all test cases, summing the total number of tests from each build.
- status-counts: Provides a consolidated view of test statuses such as passed, failed, or skipped, calculated by aggregating the count of each status type across all individual builds.

Code coverage summary

The merged code coverage report combines fields from all individual builds using the following calculations:

- branches-covered: Sum of all covered branches from individual reports.
- branches-missed: Sum of all missed branches from individual reports.
- branch-coverage-percentage: `(Total covered branches / Total branches) * 100`
- lines-covered: Sum of all covered lines from individual reports.
- lines-missed: Sum of all missed lines from individual reports.
- lines-coverage-percentage: `(Total covered lines / Total lines) * 100`

Execution ID

The batch build ARN.

Test cases

The merged report contains a consolidated list of all test cases from individual builds, accessible through both the DescribeTestCases API and the batch build report in the console.

Code coverages

The merged code coverage report provides consolidated line and branch coverage information for each file across all individual builds, accessible through both the DescribeCodeCoverages API and the batch build report in the console. Note: For files covered by multiple test files distributed across different shards, the merged report uses the following selection criteria:

1. Primary selection is based on the highest line coverage among shards.
2. If line coverage is equal across multiple shards, the shard with the highest branch coverage is selected.

Parallel test execution for various test frameworks sample

You can use the `codebuild-tests-run` CLI command to split and run your tests across parallel execution environments. The following section provides `buildspec.yml` samples for various frameworks, illustrating the usage of the `codebuild-tests-run` command.

- Each example below includes a `parallelism` level of five, meaning that five identical execution environments will be created to split your tests across. You can choose a `parallelism` level to suit your project by modifying the `parallelism` value in the `build-fanout` section.
- Each example below shows configuring your tests to be split by the test file name, which is by default. This distributes the tests evenly across the parallel execution environments.

Before you get started, see Execute parallel tests in batch builds for more information.

For a full list of options when using the `codebuild-tests-run` CLI command, see Use the `codebuild-tests-run` CLI command.

Topics

Configure parallel tests with Django

The following is sample of a `buildspec.yml` that shows parallel test execution with Django on an Ubuntu platform:

```
version: 0.2
```

```
batch:
  fast-fail: false
  build-fanout:
    parallelism: 5

phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - sudo yum install -y python3 python3-pip
      - python3 -m ensurepip --upgrade
      - python3 -m pip install django
  pre_build:
    commands:
      - echo 'Prebuild'
  build:
    commands:
      - echo 'Running Django Tests'
      - |
        codebuild-tests-run \
         --test-command 'python3 manage.py test $(echo "$CODEBUILD_CURRENT_SHARD_FILES"
 | sed -E "s/\//__/g; s/\.py$//; s/__/./g")' \
         --files-search "codebuild-glob-search '**/tests/*test_*.py'" \
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo 'Test execution completed'
```

The above example shows the usage of the environment variable `CODEBUILD_CURRENT_SHARD_FILES`. Here `CODEBUILD_CURRENT_SHARD_FILES` is used to fetch dot notation file paths supported by Django. Use `CODEBUILD_CURRENT_SHARD_FILES` inside double quotes as shown above.

Configure parallel tests with Elixir

The following is sample of a `buildspec.yml` that shows parallel test execution with Elixir on an Ubuntu platform:

```
version: 0.2

batch:
```

```
  fast-fail: false
  build-fanout:
    parallelism: 5

phases:
  install:
    commands:
      - echo 'Installing Elixir dependencies'
      - sudo apt update
      - sudo DEBIAN_FRONTEND=noninteractive apt install -y elixir
      - elixir --version
      - mix --version
  pre_build:
    commands:
      - echo 'Prebuild'
  build:
    commands:
      - echo 'Running Elixir Tests'
      - |
        codebuild-tests-run \
         --test-command 'mix test' \
         --files-search "codebuild-glob-search '**/test/**/*_test.exs'" \
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

Configure parallel tests with Go

The following is sample of a `buildspec.yml` that shows parallel test execution with Go on an Linux platform:

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
```

```
      - echo 'Fetching Go version'
      - go version
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running go Tests'
      - go mod init calculator
      - cd calc
      - |
        codebuild-tests-run \
         --test-command "go test -v calculator.go" \
         --files-search "codebuild-glob-search '**/*test.go'"
  post_build:
    commands:
      - echo "Test execution completed"
```

In above example, `calculator.go` function contains simple mathematical functions to test and all test files and `calculator.go` file is inside `calc` folder.

Configure parallel tests with Java (Maven)

The following is sample of a `buildspec.yml` that shows parallel test execution with Java on an Linux platform:

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo "Running mvn test"
      - |
        codebuild-tests-run \
```

```
          --test-command 'mvn test -Dtest=$(echo "$CODEBUILD_CURRENT_SHARD_FILES" | sed
 "s|src/test/java/||g; s/\.java//g; s|/|.|g; s/ /,/g" | tr "\n" "," | sed "s/,$//")' \
          --files-search "codebuild-glob-search '**/test/**/*.java'"

  post_build:
    commands:
      - echo "Running post-build steps..."
      - echo "Test execution completed"
```

In the given example, the environment variable `CODEBUILD_CURRENT_SHARD_FILES` contains test files in the current shard, separated by newlines. These files are converted into a comma-separated list of class names in the format accepted by the `-Dtest` parameter for Maven.

Configure parallel tests with Javascript (Jest)

The following is sample of a `buildspec.yml` that shows parallel test execution with Javascript on an Ubuntu platform:

```
version: 0.2

batch:
  fast-fail: true
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Node.js dependencies'
      - apt-get update
      - apt-get install -y nodejs
      - npm install
      - npm install --save-dev jest-junit
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running JavaScript Tests'
      - |
         codebuild-tests-run \
          --test-command "npx jest" \
```

```
          --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
          --sharding-strategy 'stability'
    post_build:
      commands:
        - echo 'Test execution completed'
```

Configure parallel tests with Kotlin

The following is sample of a `buildspec.yml` that shows parallel test execution with Kotlin on an Linux platform:

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 2
    ignore-failure: false

phases:
  install:
    runtime-versions:
      java: corretto11
    commands:
      - echo 'Installing dependencies'
      - KOTLIN_VERSION="1.8.20" # Replace with your desired version
      - curl -o kotlin-compiler.zip -L "https://github.com/JetBrains/kotlin/releases/
download/v${KOTLIN_VERSION}/kotlin-compiler-${KOTLIN_VERSION}.zip"
      - unzip kotlin-compiler.zip -d /usr/local
      - export PATH=$PATH:/usr/local/kotlinc/bin
      - kotlin -version
      - curl -O https://repo1.maven.org/maven2/org/junit/platform/junit-platform-
console-standalone/1.8.2/junit-platform-console-standalone-1.8.2.jar
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running Kotlin Tests'
      - |
        codebuild-tests-run \
```

```
          --test-command 'kotlinc src/main/kotlin/*.kt $(echo
 "$CODEBUILD_CURRENT_SHARD_FILES" | tr "\n" " ") -d classes -cp junit-platform-console-
standalone-1.8.2.jar' \
          --files-search "codebuild-glob-search 'src/test/kotlin/*.kt'"
      - |
        codebuild-tests-run \
          --test-command '
            java -jar junit-platform-console-standalone-1.8.2.jar --class-path classes
 \
              $(for file in $CODEBUILD_CURRENT_SHARD_FILES; do
                 class_name=$(basename "$file" .kt)
                 echo "--select-class $class_name"
                done)
          ' \
          --files-search "codebuild-glob-search 'src/test/kotlin/*.kt'"
  post_build:
    commands:
      - echo "Test execution completed"
```

In the above example, the `codebuild-tests-run` CLI is used twice. During the first run, kotlinc compiles the files. The `CODEBUILD_CURRENT_SHARD_FILES` variable retrieves the test files assigned to the current shard, which are then converted into a space-separated list. In the second run, JUnit executes the tests. Again, `CODEBUILD_CURRENT_SHARD_FILES` fetches the test files assigned to the current shard, but this time they are converted into class names.

Configure parallel tests with PHPUnit

The following is sample of a `buildspec.yml` that shows parallel test execution with PHPUnit on an Linux platform:

```
version: 0.2

batch:
   fast-fail: false
   build-fanout:
     parallelism: 5
     ignore-failure: false

phases:
   install:
     commands:
       - echo 'Install dependencies'
```

```
      - composer require --dev phpunit/phpunit
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running phpunit Tests'
      - composer dump-autoload
      - |
        codebuild-tests-run \
         --test-command "./vendor/bin/phpunit --debug" \
         --files-search "codebuild-glob-search '**/tests/*Test.php'"
  post_build:
      commands:
        - echo 'Test execution completed'
```

Configure parallel tests with Pytest

The following is sample of a `buildspec.yml` that shows parallel test execution with Pytest on an Ubuntu platform:

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - apt-get update
      - apt-get install -y python3 python3-pip
      - pip3 install --upgrade pip
      - pip3 install pytest
  build:
    commands:
      - echo 'Running Python Tests'
      - |
         codebuild-tests-run \
          --test-command 'python -m pytest' \
```

```
          --files-search "codebuild-glob-search 'tests/test_*.py'" \
          --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

The following is sample of a `buildspec.yml` that shows parallel test execution with Pytest on an Windows platform:

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - pip install pytest
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running pytest'
      - |
        & codebuild-tests-run `
         --test-command 'pytest @("$env:CODEBUILD_CURRENT_SHARD_FILES" -split \"`r?`n
\")'  `
         --files-search "codebuild-glob-search '**/test_*.py' '**/*_test.py'" `
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

In above example, `CODEBUILD_CURRENT_SHARD_FILES` environment variable is used to fetch test files assigned to current shard and passed as array to pytest command.

Configure parallel tests with Ruby (Cucumber)

The following is sample of a `buildspec.yml` that shows parallel test execution with Cucumber on an Linux platform:

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Ruby dependencies'
      - gem install bundler
      - bundle install
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running Cucumber Tests'
      - cucumber --init
      - |
        codebuild-tests-run \
         --test-command "cucumber" \
         --files-search "codebuild-glob-search '**/*.feature'"
  post_build:
    commands:
      - echo "Test execution completed"
```

Configure parallel tests with Ruby (RSpec)

The following is sample of a `buildspec.yml` that shows parallel test execution with RSpec on an Ubuntu platform:

```
version: 0.2

batch:
```

```
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Ruby dependencies'
      - apt-get update
      - apt-get install -y ruby ruby-dev build-essential
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo 'Running Ruby Tests'
      - |
         codebuild-tests-run \
          --test-command 'bundle exec rspec' \
          --files-search "codebuild-glob-search 'spec/**/*_spec.rb'" \
          --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

Cache builds to improve performance

You can save time when your project builds by using a cache. A cache can store reusable pieces of your build environment and use them across multiple builds. Your build project can use one of two types of caching: Amazon S3 or local. If you use a local cache, you must choose one or more of three cache modes: source cache, Docker layer cache, and custom cache.

> **Note**
>
> Docker layer cache mode is available for the Linux environment only. If you choose this mode, you must run your build in privileged mode. CodeBuild projects granted privileged mode grants its container access to all devices. For more information, see Runtime privilege and Linux capabilities on the Docker Docs website.

Topics

Amazon S3 caching

Amazon S3 caching stores the cache in an Amazon S3 bucket that is available across multiple build hosts. This is a good option for small to intermediate sized build artifacts that are more expensive to build than to download.

To use Amazon S3 in a build, you can specify the paths for the files you want to cache in `buildspec.yml`. CodeBuild will automatically store and update the cache to the Amazon S3 location configured on the project. If you don't specify the file paths, CodeBuild will best-effort cache common language dependencies to help you speed up the builds. You can view the cache details in the build logs.

Additionally, if you want to have multiple versions of cache, you can define a cache key in the `buildspec.yml`. CodeBuild stores the cache under the context of this cache key, and create a unique cache copy that will not be updated once created. The cache keys can be shared across projects as well. Features such as dynamic keys, cache versioning, and cache sharing across builds are only available when a key is specified.

To learn more about the cache syntax in buildspec file, see cache in the buildspec reference.

Topics

Generate dynamic keys

A cache key can include shell commands and environment variables to make it unique, enabling automatic cache updates when key changes. For example, you can define a key using the hash of the `package-lock.json` file. When the dependencies in that file change, the hash—and therefore the cache key—changes, triggering the automatic creation of a new cache.

```
cache:
    key: npm-key-$(codebuild-hash-files package-lock.json)
```

CodeBuild will evaluate the expression `$(codebuild-hash-files package-lock.json)` to get the final key:

```
npm-key-abc123
```

You can also define a cache key using environment variables, such as `CODEBUILD_RESOLVED_SOURCE_VERSION`. This ensures that whenever your source changes, a new key is generated, resulting in a new cache being saved automatically:

```
cache:
   key: npm-key-$CODEBUILD_RESOLVED_SOURCE_VERSION
```

CodeBuild will evaluate the expression and get the final key:

```
npm-key-046e8b67481d53bdc86c3f6affdd5d1afae6d369
```

codebuild-hash-files

`codebuild-hash-files` is a CLI tool that calculates a SHA-256 hash for a set of files in the CodeBuild source directory using glob patterns:

```
codebuild-hash-files <glob-pattern-1> <glob-pattern-2> ...
```

Here are some examples using `codebuild-hash-files`:

```
codebuild-hash-files package-lock.json
codebuild-hash-files '**/*.md'
```

Cache version

The cache version is a hash generated from the paths of the directories being cached. If two caches have different versions, they are treated as distinct caches during the matching process. For example, the following two caches are considered different because they reference different paths:

```
version: 0.2
```

```
phases:
  build:
    commands:
      - pip install pandas==2.2.3 --target pip-dependencies
cache:
  key: pip-dependencies
  paths:
    - "pip-dependencies/**/*"
```

```
version: 0.2

phases:
  build:
    commands:
      - pip install pandas==2.2.3 --target tmp/pip-dependencies
cache:
  key: pip-dependencies
  paths:
    - "tmp/pip-dependencies/**/*"
```

Cache sharing between projects

You can use the `cacheNamespace` API field under the `cache` section to share a cache across multiple projects. This field defines the scope of the cache. To share a cache, must do the following:

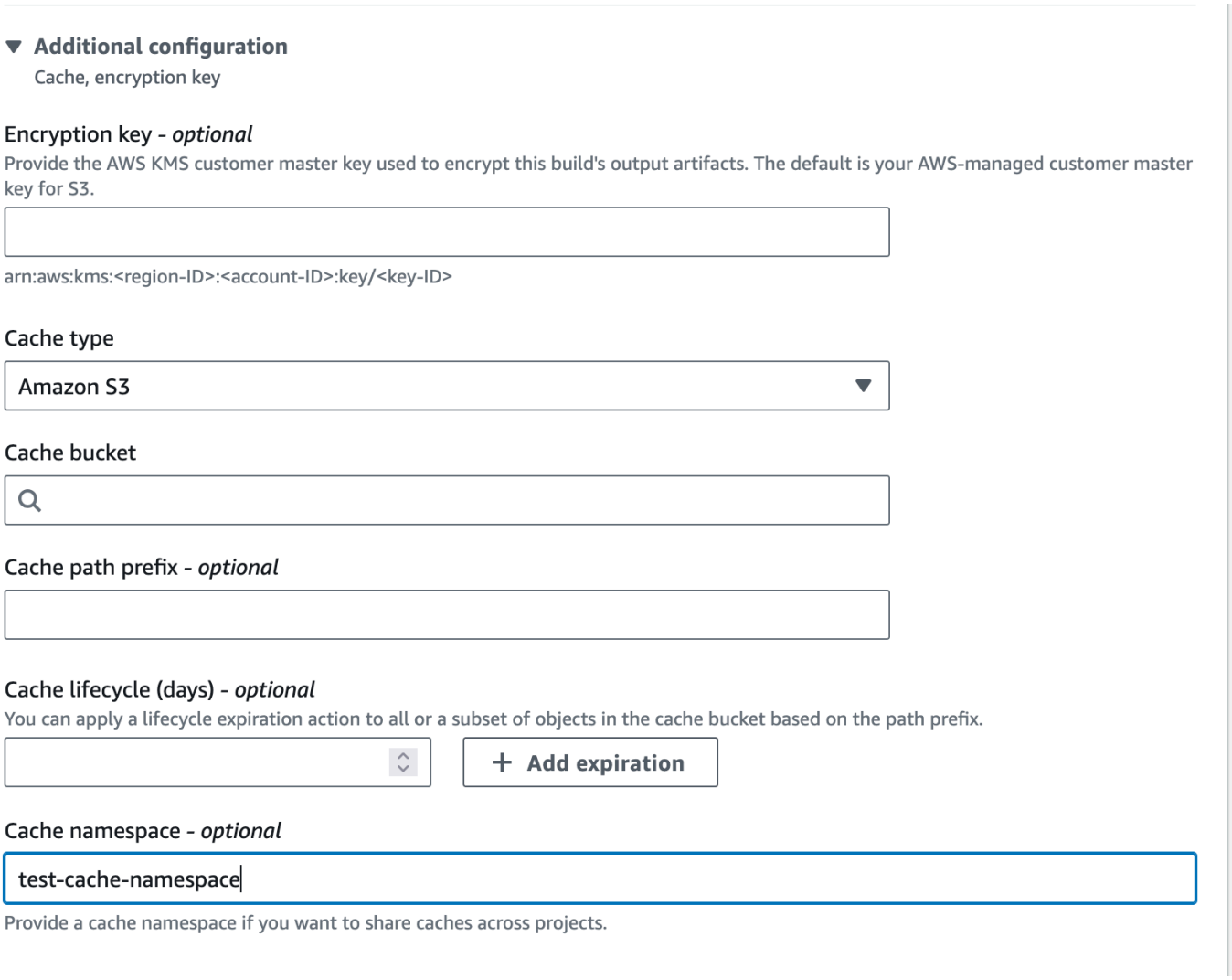
- Use the same `cacheNamespace`.
- Specify the same cache `key`.
- Define identical cache paths.
- Use the same Amazon S3 buckets and `pathPrefix` if set.

This ensures consistency and enables cache sharing across projects.

Specify a cache namespace (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Choose **Create project**. For information, see Create a build project (console) and Run a build (console).
3. In **Artifacts**, choose **Additional configuration**.

4. For **Cache type**, choose **Amazon S3**.
5. For **Cache namespace - optional**, enter a namespace value.



6. Continue with the default values and then choose **Create build project**.

Specify a cache namespace (AWS CLI)

You can use the the `--cache` parameter in the AWS CLI to specify a cache namespace.

```
--cache '{"type": "S3", "location": "your-s3-bucket", "cacheNamespace": "test-cache-namespace"}'
```

Buildspec examples

Here are several buildspec examples for common languages:

Topics

Cache Node.js dependencies

If your project includes a `package-lock.json` file and uses `npm` to manage Node.js dependencies, the following example shows how to set up caching. By default, `npm` installs dependencies into the `node_modules` directory.

```
version: 0.2

phases:
  build:
    commands:
      - npm install
cache:
  key: npm-$(codebuild-hash-files package-lock.json)
  paths:
    - "node_modules/**/*"
```

Cache Python dependencies

If your project includes a `requirements.txt` file and uses pip to manage Python dependencies, the following example demonstrates how to configure caching. By default, pip installs packages into the system's `site-packages` directory.

```
version: 0.2

phases:
  build:
    commands:
      - pip install -r requirements.txt
cache:
  key: python-$(codebuild-hash-files requirements.txt)
  paths:
```

```
    - "/root/.pyenv/versions/${python_version}/lib/python${python_major_version}/site-
packages/**/*"
```

Additionally, you can install dependencies into a specific directory and configure caching for that directory.

```
version: 0.2

phases:
  build:
    commands:
      - pip install -r requirements.txt --target python-dependencies
cache:
  key: python-$(codebuild-hash-files requirements.txt)
  paths:
    - "python-dependencies/**/*"
```

Cache Ruby dependencies

If your project includes a `Gemfile.lock` file and uses `Bundler` to manage gem dependencies, the following example demonstrates how to configure caching effectively.

```
version: 0.2

phases:
  build:
    commands:
      - bundle install --path vendor/bundle
cache:
  key: ruby-$(codebuild-hash-files Gemfile.lock)
  paths:
    - "vendor/bundle/**/*"
```

Cache Go dependencies

If your project includes a `go.sum` file and uses Go modules to manage dependencies, the following example demonstrates how to configure caching. By default, Go modules are downloaded and stored in the `${GOPATH}/pkg/mod` directory.

```
version: 0.2
```

```
phases:
  build:
    commands:
      - go mod download
cache:
  key: go-$(codebuild-hash-files go.sum)
  paths:
    - "/go/pkg/mod/**/*"
```

Local caching

Local caching stores a cache locally on a build host that is available to that build host only. This is a good option for intermediate to large build artifacts because the cache is immediately available on the build host. This is not the best option if your builds are infrequent. This means that build performance is not impacted by network transfer time.

If you choose local caching, you must choose one or more of the following cache modes:

- Source cache mode caches Git metadata for primary and secondary sources. After the cache is created, subsequent builds pull only the change between commits. This mode is a good choice for projects with a clean working directory and a source that is a large Git repository. If you choose this option and your project does not use a Git repository (AWS CodeCommit, GitHub, GitHub Enterprise Server, or Bitbucket), the option is ignored.
- Docker layer cache mode caches existing Docker layers. This mode is a good choice for projects that build or pull large Docker images. It can prevent the performance issues caused by pulling large Docker images down from the network.

> **Note**
>
> - You can use a Docker layer cache in the Linux environment only.
> - The `privileged` flag must be set so that your project has the required Docker permissions.
>
>   By default, Docker daemon is enabled for non-VPC builds. If you would like to use Docker containers for VPC builds, see Runtime Privilege and Linux Capabilities on the Docker Docs website and enable privileged mode. Also, Windows does not support privileged mode.
> - You should consider the security implication before you use a Docker layer cache.

- Custom cache mode caches directories you specify in the buildspec file. This mode is a good choice if your build scenario is not suited to one of the other two local cache modes. If you use a custom cache:
  - Only directories can be specified for caching. You cannot specify individual files.
  - Symlinks are used to reference cached directories.
  - Cached directories are linked to your build before it downloads its project sources. Cached items overrides source items if they have the same name. Directories are specified using cache paths in the buildspec file. For more information, see Buildspec syntax.
  - Avoid directory names that are the same in the source and in the cache. Locally-cached directories may override, or delete the contents of, directories in the source repository that have the same name.

> **Note**
>
> Local caching is not supported with the `LINUX_GPU_CONTAINER` environment type and the `BUILD_GENERAL1_2XLARGE` compute type. For more information, see Build environment compute modes and types.

> **Note**
>
> Local caching is not supported when you configure CodeBuild to work with a VPC. For more information on using VPCs with CodeBuild, see Use AWS CodeBuild with Amazon Virtual Private Cloud.

Specify a local cache

You can use the AWS CLI, console, SDK, or AWS CloudFormation to specify a local cache. For more information about local caching, see Local caching.

Topics

- Specify local caching (CLI)
- Specify local caching (console)
- Specify local caching (AWS CloudFormation)

Specify local caching (CLI)

You can use the the `--cache` parameter in the AWS CLI to specify each of the three local cache types.

- To specify a source cache:

```
--cache type=LOCAL,mode=[LOCAL_SOURCE_CACHE]
```

- To specify a Docker layer cache:

```
--cache type=LOCAL,mode=[LOCAL_DOCKER_LAYER_CACHE]
```

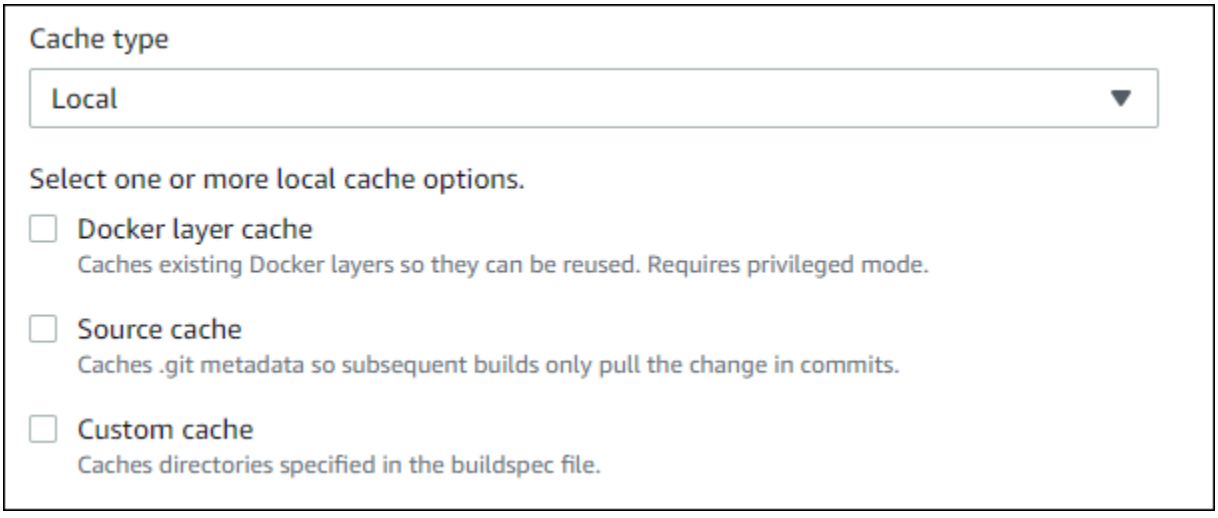
- To specify a custom cache:

```
--cache type=LOCAL,mode=[LOCAL_CUSTOM_CACHE]
```

For more information, see Create a build project (AWS CLI).

Specify local caching (console)

You specify a cache in the **Artifacts** section of the console. For **Cache type**, choose **Amazon S3** or **Local**. If you choose **Local**, choose one or more of the three local cache options.



For more information, see Create a build project (console).

Specify local caching (AWS CloudFormation)

If you use AWS CloudFormation to specify a local cache, on the `Cache` property, for `Type`, specify `LOCAL`. The following sample YAML-formatted AWS CloudFormation code specifies all three local cache types. You can specify any combination of the types. If you use a Docker layer cache, under `Environment`, you must set `PrivilegedMode` to `true` and `Type` to `LINUX_CONTAINER`.

```
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**
>
> By default, Docker daemon is enabled for non-VPC builds. If you would like to use Docker containers for VPC builds, see Runtime Privilege and Linux Capabilities on the Docker Docs website and enable privileged mode. Also, Windows does not support privileged mode.

For more information, see Create a build project (AWS CloudFormation).

Debug builds in AWS CodeBuild

AWS CodeBuild provides two methods for debugging builds during development and troubleshooting. You can use the CodeBuild Sandbox environment to investigate issues and validate fixes in real-time, or you can use AWS Systems Manager Session Manager to connect to the build container and view the container state.

Debug builds with CodeBuild sandbox

The CodeBuild sandbox environment provides an interactive debug session in a secure and isolated environment. You can interact with the environment directly through the AWS Management Console or AWS CLI, execute commands, and validate your build process step by step. It uses a cost-effective per-second billing model and supports the same native integration with source providers and AWS services as your build environment. You can also connect to a sandbox environment using SSH clients or from your integrated development environments (IDEs).

To learn more about the CodeBuild sandbox pricing, visit the CodeBuild pricing documentation. For detailed instructions, visit the Debug builds with CodeBuild sandbox documentation.

Debug builds with Session Manager

AWS Systems Manager Session Manager enables direct access to running builds in their actual execution environment. This approach allows you to connect to active build containers and inspect the build process in real-time. You can examine the file system, monitor running processes, and troubleshoot issues as they occur.

For detailed instructions, visit the Debug builds with Session Manager documentation.

Debug builds with CodeBuild sandbox

In AWS CodeBuild, you can debug a build by using CodeBuild sandbox to run custom commands and troubleshoot your build.

Topics

Prerequisites

Before using a CodeBuild sandbox, make sure that your CodeBuild service role has the following SSM policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
          "ssmmessages:CreateControlChannel",
          "ssmmessages:CreateDataChannel",
          "ssmmessages:OpenControlChannel",
          "ssmmessages:OpenDataChannel"
       ],
       "Resource": "*"
     },
     {
       "Effect": "Allow",
       "Action": [
          "ssm:StartSession"
       ],
       "Resource": [
          "arn:aws:codebuild:<region>:<account-id>:build/*",
          "arn:aws:ssm:<region>::document/AWS-StartSSHSession"
       ]
```

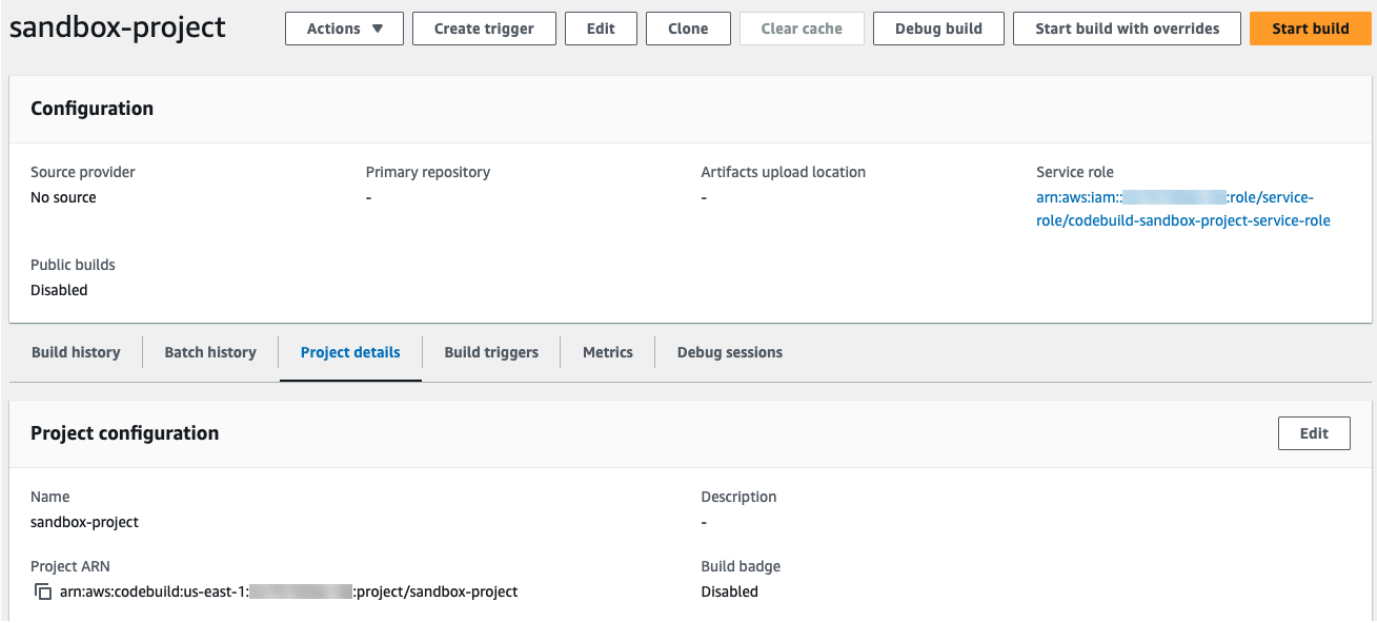
```
      }
    ]
}
```

Debug builds with CodeBuild sandbox (console)

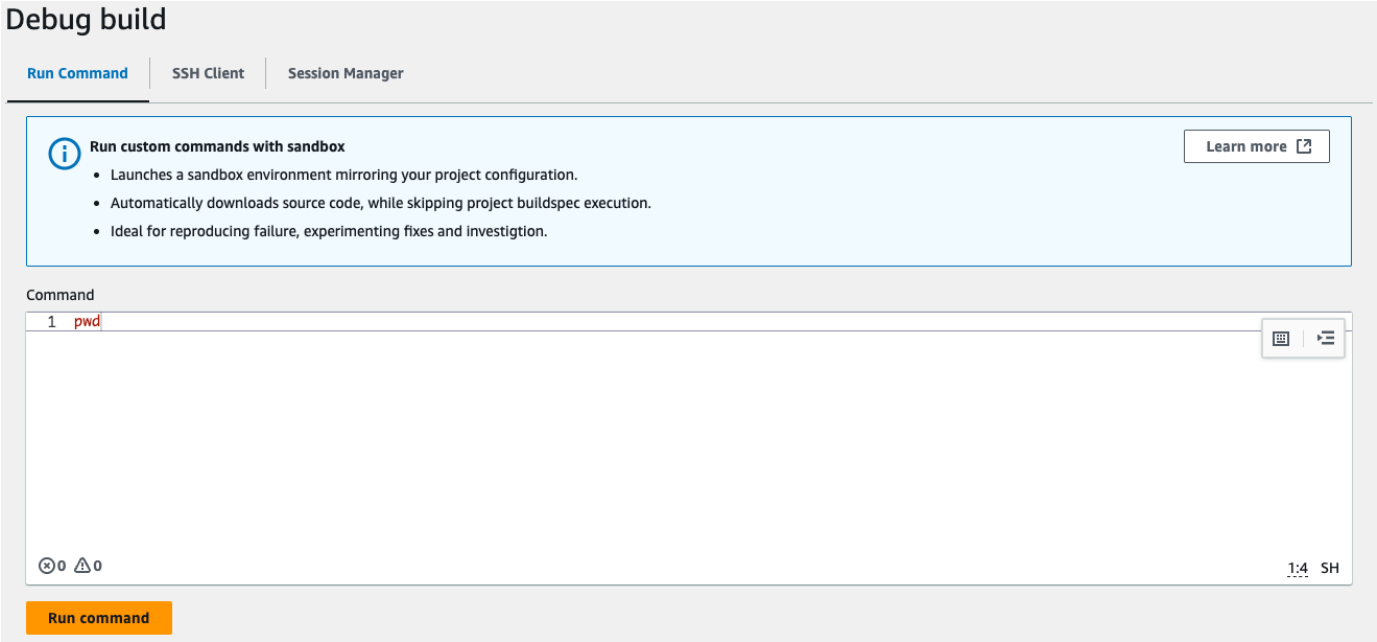
Use the following instructions to run commands and connect your SSH client with CodeBuild sandbox in the console.

Run commands with CodeBuild sandbox (console)

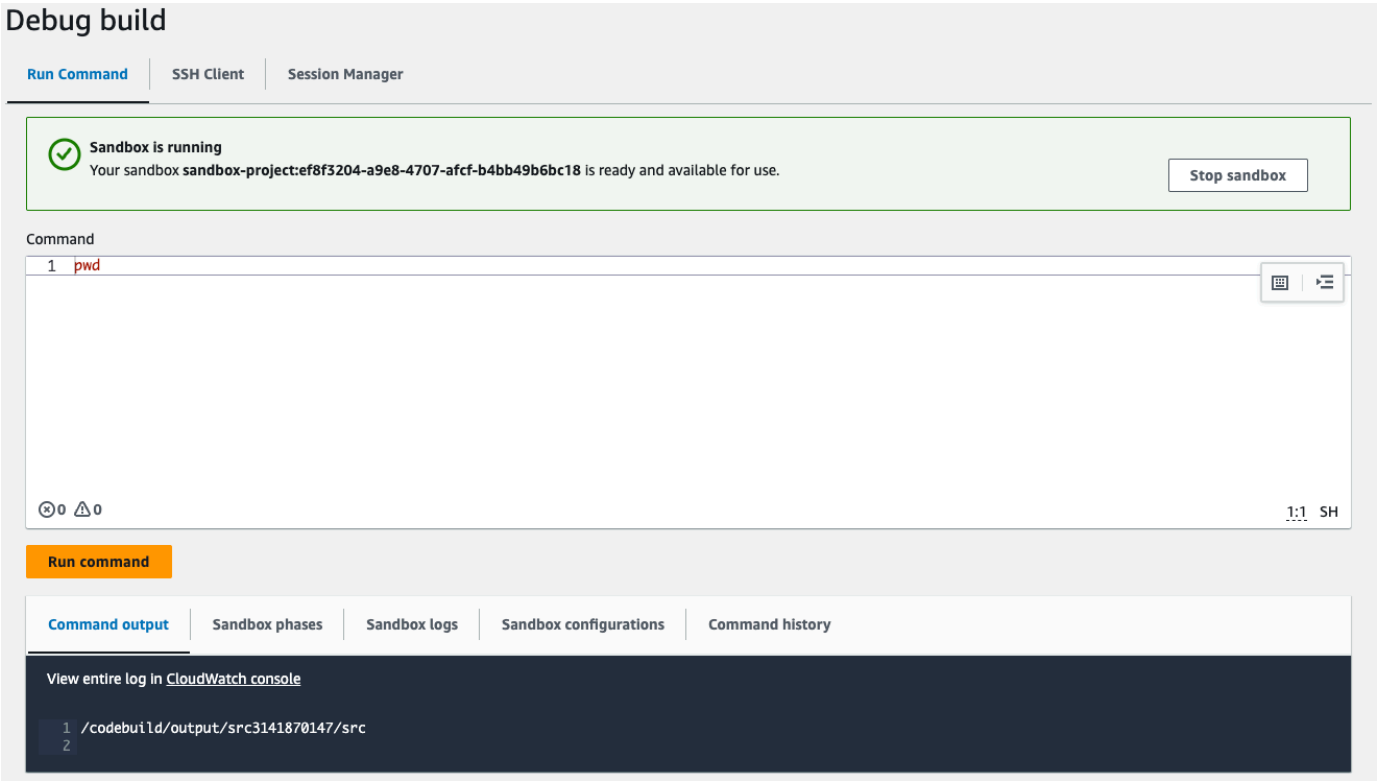
1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**. Choose the build project, and then choose **Debug build**.



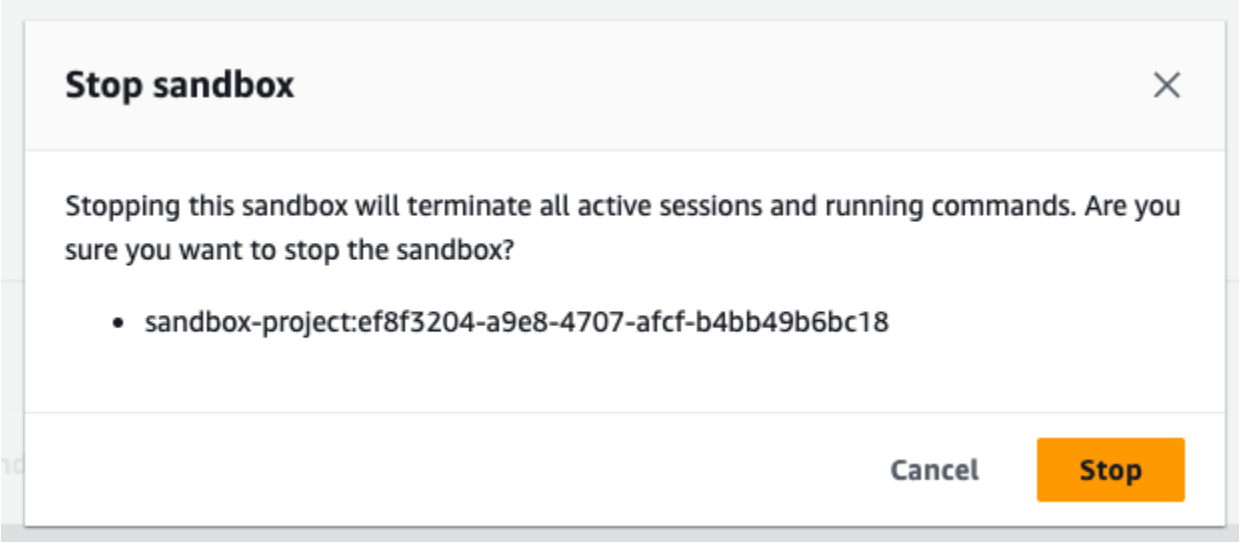
3. In the **Run command** tab, enter your custom commands, and then choose **Run command**.

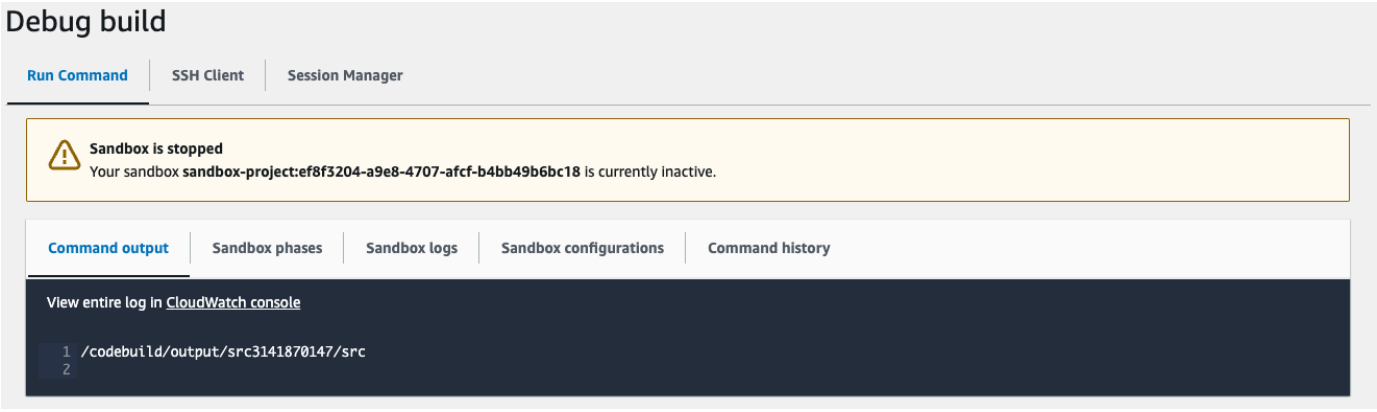


4. Your CodeBuild sandbox will then be initialized and start running your custom commands. The output will be shown in the **Output** tab when it's completed.



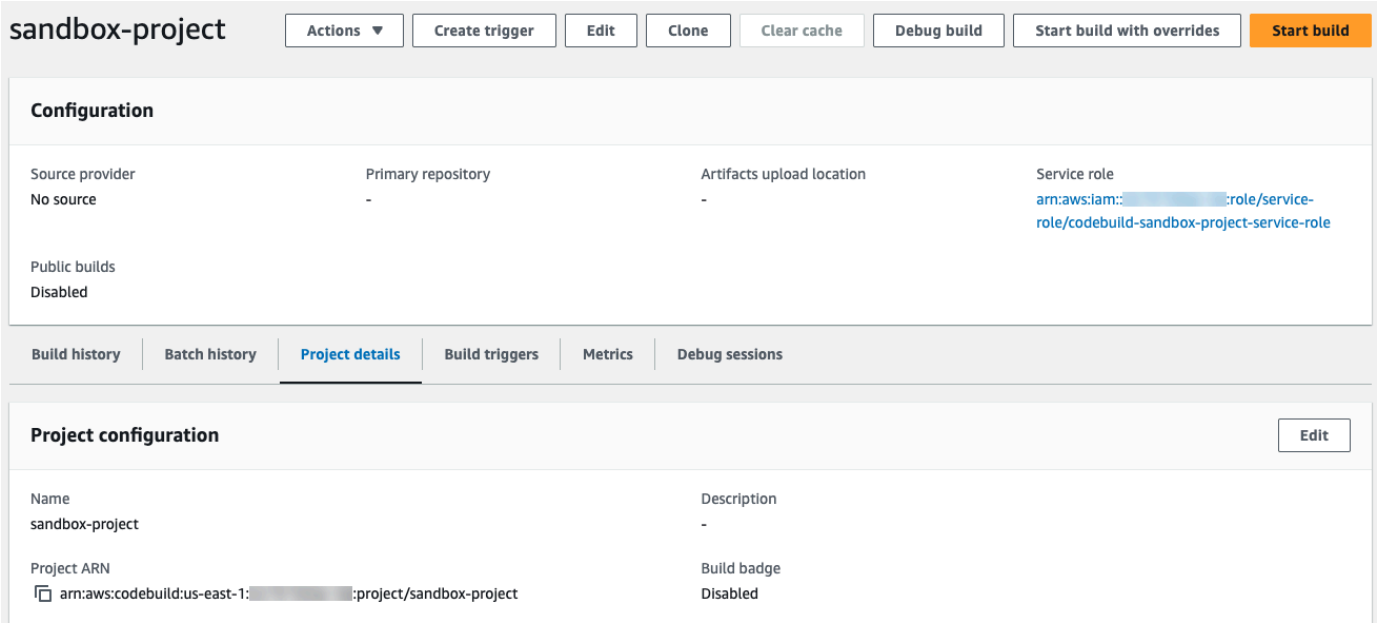
5. When troubleshooting is completed, you can stop the sandbox by choosing **Stop sandbox**. Then choose **Stop** to confirm that your sandbox will be stopped.



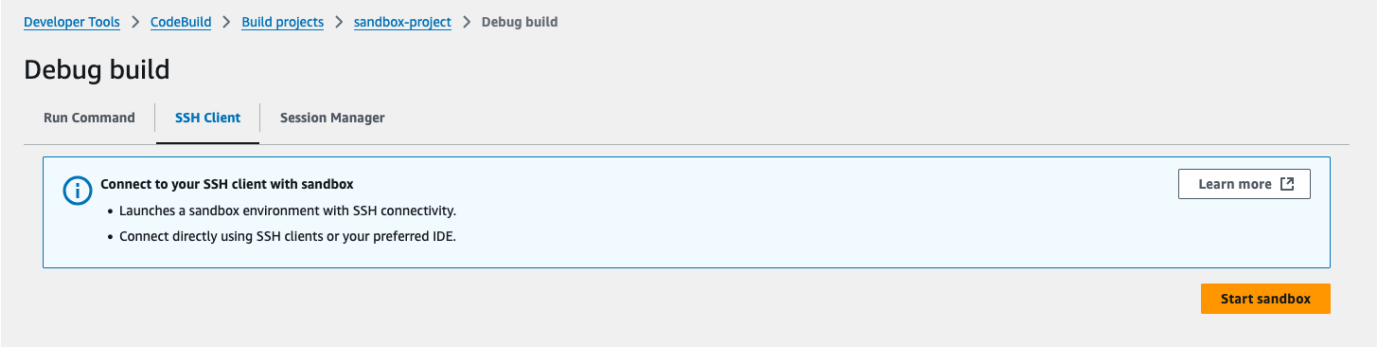


Connect to your SSH client with CodeBuild sandbox (console)

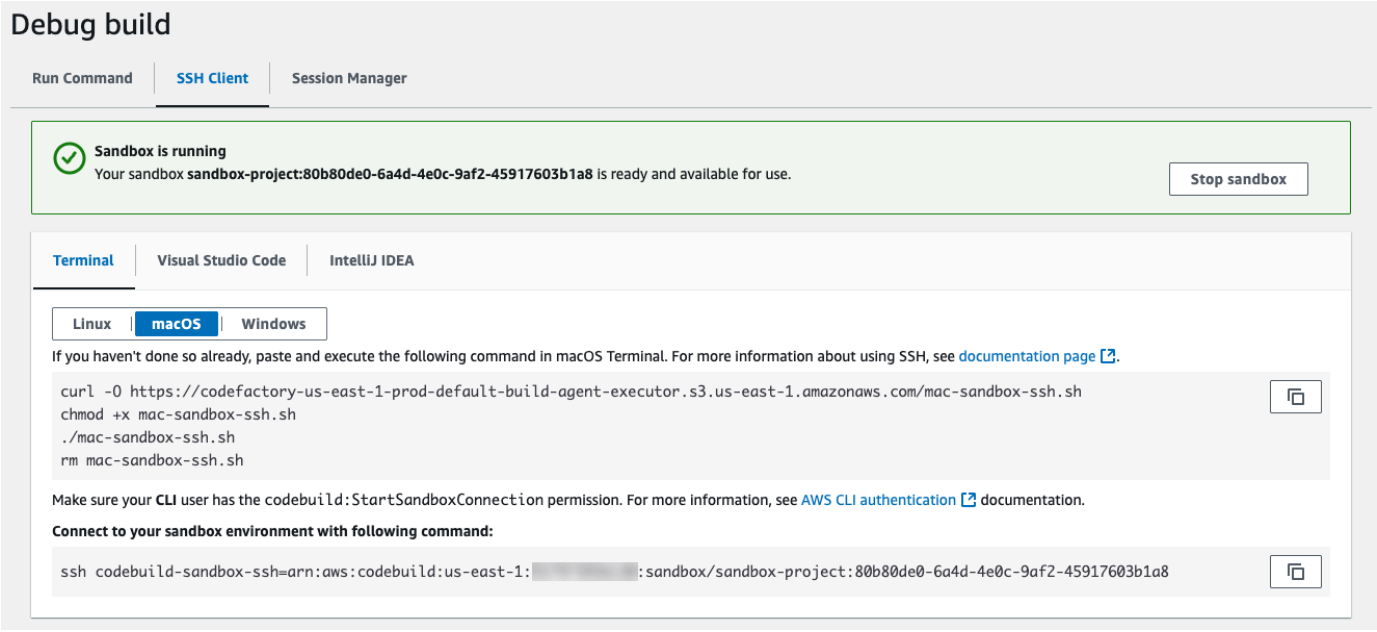
1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**. Choose the build project, and then choose **Debug build**.



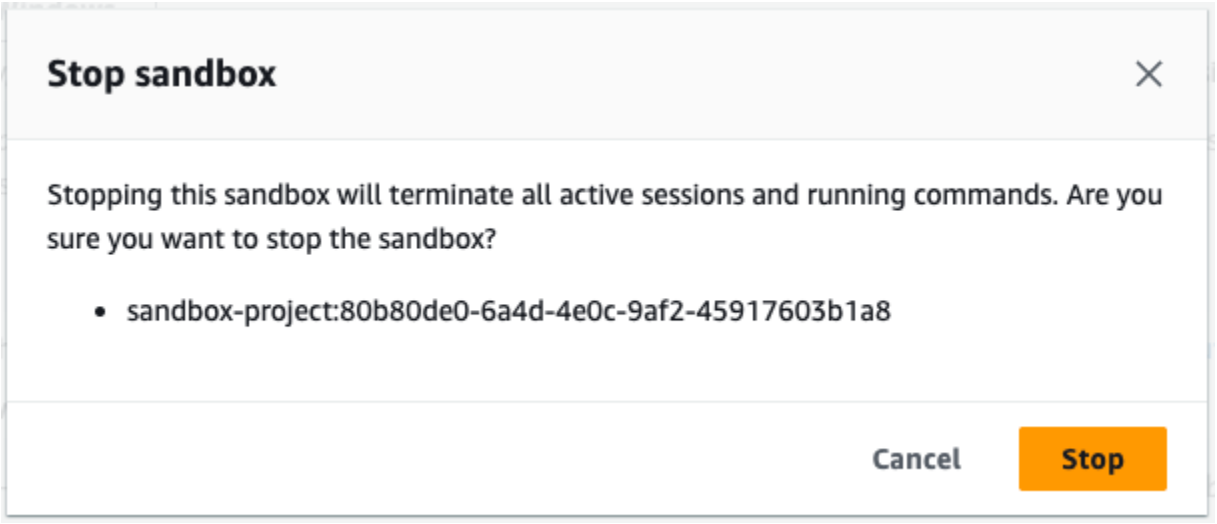
3. In the **SSH Client** tab and choose **Start sandbox**.

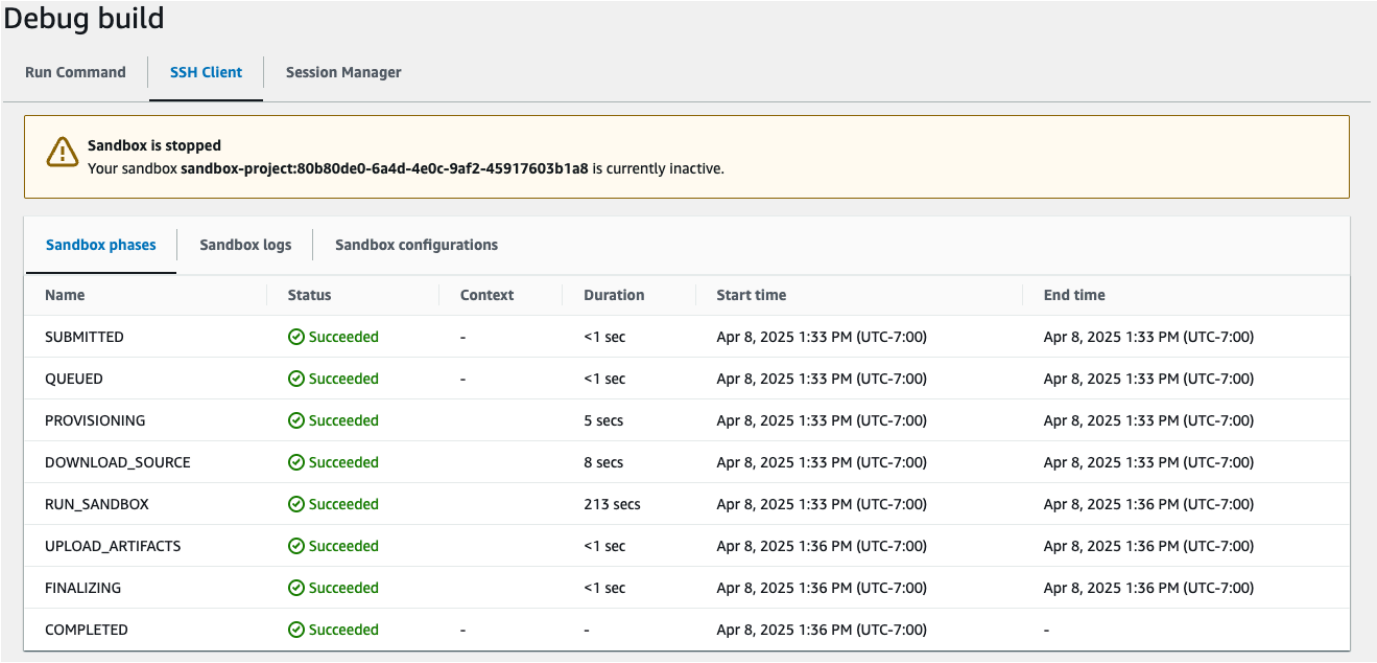


4. After the CodeBuild sandbox starts running, follow the console instructions to connect your SSH client with the sandbox.



5. When troubleshooting is completed, you can stop the sandbox by choosing **Stop sandbox**. Then choose **Stop** to confirm that your sandbox will be stopped.





Debug builds with CodeBuild sandbox (AWS CLI)

Use the following instructions to run commands and connect your SSH client with CodeBuild sandbox.

Start a CodeBuild sandbox (AWS CLI)

CLI command

```
aws codebuild start-sandbox --project-name $PROJECT_NAME
```

- `--project-name` : CodeBuild project name

Sample request

```
aws codebuild start-sandbox --project-name "project-name"
```

Sample response

```
{
    "id": "project-name",
    "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
    "projectName": "project-name",
```

```
    "requestTime": "2025-02-06T11:24:15.560000-08:00",
    "status": "QUEUED",
    "source": {
        "type": "S3",
        "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
        "insecureSsl": false
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:6.0",
        "computeType": "BUILD_GENERAL1_SMALL",
        "environmentVariables": [{
                "name": "foo",
                "value": "bar",
                "type": "PLAINTEXT"
            },
            {
                "name": "bar",
                "value": "baz",
                "type": "PLAINTEXT"
            }
        ],
        "privilegedMode": false,
        "imagePullCredentialsType": "CODEBUILD"
    },
    "timeoutInMinutes": 10,
    "queuedTimeoutInMinutes": 480,
    "logConfig": {
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
            "encryptionDisabled": false
        }
    },
    "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
    "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
    "currentSession": {
        "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
```

```
        "currentPhase": "QUEUED",
        "status": "QUEUED",
        "startTime": "2025-02-06T11:24:15.626000-08:00",
        "logs": {
            "groupName": "group",
            "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        }
    }
}
```

Get information about the sandbox status (AWS CLI)

CLI command

```
aws codebuild batch-get-sandboxes --ids $SANDBOX_IDs
```

Sample request

```
aws codebuild stop-sandbox --id "arn:aws:codebuild:us-west-2:962803963624:sandbox/
project-name"
```

- `--ids` : Comma separated list of `sandboxIds` or `sandboxArns`.

You can either provide a sandbox ID or a sandbox ARN:

- Sandbox ID: *<codebuild-project-name>*:*<UUID>*

  For example, `project-name:d25be134-05cb-404a-85da-ac5f85d2d72c`.

- Sandbox ARN: arn:aws:codebuild:*<region>*:*<account-id>*:sandbox/*<codebuild-project-name>*:*<UUID>*

  For example, `arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name:d25be134-05cb-404a-85da-ac5f85d2d72c`.

Sample response

```
{
    "sandboxes": [{
        "id": "project-name",
        "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
        "projectName": "project-name",
        "requestTime": "2025-02-06T11:24:15.560000-08:00",
        "endTime": "2025-02-06T11:39:21.587000-08:00",
        "status": "STOPPED",
        "source": {
            "type": "S3",
            "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
            "insecureSsl": false
        },
        "environment": {
            "type": "LINUX_CONTAINER",
            "image": "aws/codebuild/standard:6.0",
            "computeType": "BUILD_GENERAL1_SMALL",
            "environmentVariables": [{
                    "name": "foo",
                    "value": "bar",
                    "type": "PLAINTEXT"
                },
                {
                    "name": "bar",
                    "value": "baz",
```

```
                    "type": "PLAINTEXT"
                }
            ],
            "privilegedMode": false,
            "imagePullCredentialsType": "CODEBUILD"
        },
        "timeoutInMinutes": 10,
        "queuedTimeoutInMinutes": 480,
        "logConfig": {
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        },
        "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/
SampleEncryptionKey",
        "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
        "currentSession": {
            "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "currentPhase": "COMPLETED",
            "status": "STOPPED",
            "startTime": "2025-02-06T11:24:15.626000-08:00",
            "endTime": "2025-02-06T11:39:21.600000-08:00",
            "phases": [{
                    "phaseType": "SUBMITTED",
                    "phaseStatus": "SUCCEEDED",
                    "startTime": "2025-02-06T11:24:15.577000-08:00",
                    "endTime": "2025-02-06T11:24:15.606000-08:00",
                    "durationInSeconds": 0
                },
                {
                    "phaseType": "QUEUED",
                    "phaseStatus": "SUCCEEDED",
                    "startTime": "2025-02-06T11:24:15.606000-08:00",
                    "endTime": "2025-02-06T11:24:16.067000-08:00",
                    "durationInSeconds": 0
                },
```

```
                {
                    "phaseType": "PROVISIONING",
                    "phaseStatus": "SUCCEEDED",
                    "startTime": "2025-02-06T11:24:16.067000-08:00",
                    "endTime": "2025-02-06T11:24:20.519000-08:00",
                    "durationInSeconds": 4,
                    "contexts": [{
                        "statusCode": "",
                        "message": ""
                    }]
                },
                {
                    "phaseType": "DOWNLOAD_SOURCE",
                    "phaseStatus": "SUCCEEDED",
                    "startTime": "2025-02-06T11:24:20.519000-08:00",
                    "endTime": "2025-02-06T11:24:22.238000-08:00",
                    "durationInSeconds": 1,
                    "contexts": [{
                        "statusCode": "",
                        "message": ""
                    }]
                },
                {
                    "phaseType": "RUNNING_SANDBOX",
                    "phaseStatus": "TIMED_OUT",
                    "startTime": "2025-02-06T11:24:22.238000-08:00",
                    "endTime": "2025-02-06T11:39:21.560000-08:00",
                    "durationInSeconds": 899,
                    "contexts": [{
                        "statusCode": "BUILD_TIMED_OUT",
                        "message": "Build has timed out. "
                    }]
                },
                {
                    "phaseType": "COMPLETED",
                    "startTime": "2025-02-06T11:39:21.560000-08:00"
                }
            ],
            "logs": {
                "groupName": "group",
                "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/
home?region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
```

```
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
                    "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
            }
        }
    }],
    "sandboxesNotFound": []
}
```

Stop a sandbox (AWS CLI)

CLI command

```
aws codebuild stop-sandbox --id $SANDBOX-ID
```

- `--id` : A `sandboxId` or `sandboxArn`.

Sample request

```
aws codebuild stop-sandbox --id "arn:aws:codebuild:us-west-2:962803963624:sandbox/
project-name"
```

Sample response

```
{
    "id": "project-name",
```

```
    "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
    "projectName": "project-name",
    "requestTime": "2025-02-06T11:24:15.560000-08:00",
    "status": "STOPPING",
    "source": {
        "type": "S3",
        "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
        "insecureSsl": false
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:6.0",
        "computeType": "BUILD_GENERAL1_SMALL",
        "environmentVariables": [{
                "name": "foo",
                "value": "bar",
                "type": "PLAINTEXT"
            },
            {
                "name": "bar",
                "value": "baz",
                "type": "PLAINTEXT"
            }
        ],
        "privilegedMode": false,
        "imagePullCredentialsType": "CODEBUILD"
    },
    "timeoutInMinutes": 10,
    "queuedTimeoutInMinutes": 480,
    "logConfig": {
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
            "encryptionDisabled": false
        }
    },
    "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
    "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
```

```
    "currentSession": {
        "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "currentPhase": "RUN_SANDBOX",
        "status": "STOPPING",
        "startTime": "2025-02-06T11:24:15.626000-08:00",
        "phases": [{
                "phaseType": "SUBMITTED",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:26.144000-08:00",
                "endTime": "2025-02-08T14:33:26.173000-08:00",
                "durationInSeconds": 0
            },
            {
                "phaseType": "QUEUED",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:26.173000-08:00",
                "endTime": "2025-02-08T14:33:26.702000-08:00",
                "durationInSeconds": 0
            },
            {
                "phaseType": "PROVISIONING",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:26.702000-08:00",
                "endTime": "2025-02-08T14:33:30.530000-08:00",
                "durationInSeconds": 3,
                "contexts": [{
                    "statusCode": "",
                    "message": ""
                }]
            },
            {
                "phaseType": "DOWNLOAD_SOURCE",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:30.530000-08:00",
                "endTime": "2025-02-08T14:33:33.478000-08:00",
                "durationInSeconds": 2,
                "contexts": [{
                    "statusCode": "",
                    "message": ""
                }]
            },
            {
                "phaseType": "RUN_SANDBOX",
                "startTime": "2025-02-08T14:33:33.478000-08:00"
```

```
            }
        ],
        "logs": {
            "groupName": "group",
            "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        }
    }
}
```

Start a command execution (AWS CLI)

CLI command

```
aws codebuild start-command-execution --command $COMMAND --type $TYPE --sandbox-id
 $SANDBOX-ID
```

- `--command` : The command that needs to be executed.
- `--sandbox-id` : A `sandboxId` or `sandboxArn`.
- `--type` : The command type, `SHELL`.

Sample request

```
aws codebuild start-command-execution --command "echo "Hello World"" --type SHELL --
sandbox-id "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name
```

Sample response

```
{
    "id": "e1c658c2-02bb-42a8-9abb-94835241fcd6",
    "sandboxId": "f7126a4a-b0d5-452f-814c-fea73718f805",
    "submitTime": "2025-02-06T20:12:02.683000-08:00",
    "status": "SUBMITTED",
    "command": "echo \"Hello World\"",
    "type": "SHELL",
    "logs": {
        "groupName": "group",
        "streamName": "stream",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logsV2:log-groups/log-group/group/log-events/stream",
        "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/codefactory-test-
pool-1-us-west-2-beta-default-build-logs/f7126a4a-b0d5-452f-814c-fea73718f805.gz?
region=us-west-2",
        "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
        "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-default-
build-logs/f7126a4a-b0d5-452f-814c-fea73718f805.gz",
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
            "encryptionDisabled": false
        }
    }
}
```

Get information about the command executions (AWS CLI)

CLI command

```
aws codebuild batch-get-command-executions --command-execution-ids $COMMAND-IDs --
sandbox-id $SANDBOX-IDs
```

- `--command-execution-ids` : Comma separated list of `commandExecutionIds`.
- `--sandbox-id` : A `sandboxId` or `sandboxArn`.

Sample request

```
aws codebuild batch-get-command-executions --command-execution-
ids"c3c085ed-5a8f-4531-8e95-87d547f27ffd" --sandbox-id "arn:aws:codebuild:us-
west-2:962803963624:sandbox/project-name"
```

Sample response

```
{
    "commandExecutions": [{
        "id": "c3c085ed-5a8f-4531-8e95-87d547f27ffd",
        "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
        "submitTime": "2025-02-10T20:18:17.118000-08:00",
        "startTime": "2025-02-10T20:18:17.939000-08:00",
        "endTime": "2025-02-10T20:18:17.976000-08:00",
        "status": "SUCCEEDED",
        "command": "echo \"Hello World\"",
        "type": "SHELL",
        "exitCode": "0",
        "standardOutputContent": "Hello World\n",
        "logs": {
            "groupName": "group",
            "streamName": "stream",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logsV2:log-groups/log-group/group/log-events/stream",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
```

```
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        }
    }],
    "commandExecutionsNotFound": []
}
```

List command executions for a sandbox (AWS CLI)

CLI command

```
aws codebuild list-command-executions-for-sandbox --sandbox-id $SANDBOX-ID --next-
token $NEXT_TOKEN --max-results $MAX_RESULTS --sort-order $SORT_ORDER
```

- `--next-token` : The next token, if any, to get paginated results. You will get this value from previous execution of list sandboxes.
- `--max-results` : (Optional) The maximum number of sandbox records to be retrieved.
- `--sort-order` : The order in which sandbox records should be retrieved.

Sample request

```
aws codebuild list-command-executions-for-sandbox --sandbox-id
 "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name"
```

Sample response

```
{
    "commandExecutions": [{
            "id": "aad6687e-07bc-45ab-a1fd-f5440229b528",
```

```
            "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
            "submitTime": "2025-02-10T20:18:35.304000-08:00",
            "startTime": "2025-02-10T20:18:35.615000-08:00",
            "endTime": "2025-02-10T20:18:35.651000-08:00",
            "status": "FAILED",
            "command": "fail command",
            "type": "SHELL",
            "exitCode": "127",
            "standardErrContent": "/codebuild/output/tmp/script.sh: 4: fail: not
 found\n",
            "logs": {
                "groupName": "group",
                "streamName": "stream",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream",
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
                    "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
            }
        },
        {
            "id": "c3c085ed-5a8f-4531-8e95-87d547f27ffd",
            "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
            "submitTime": "2025-02-10T20:18:17.118000-08:00",
            "startTime": "2025-02-10T20:18:17.939000-08:00",
            "endTime": "2025-02-10T20:18:17.976000-08:00",
            "status": "SUCCEEDED",
            "command": "echo \"Hello World\"",
            "type": "SHELL",
```

```
            "exitCode": "0",
            "standardOutputContent": "Hello World\n",
            "logs": {
                "groupName": "group",
                "streamName": "stream",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream",
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
                    "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
            }
        }
    ]
}
```

List sandboxes (AWS CLI)

CLI command

```
aws codebuild list-sandboxes --next-token $NEXT_TOKEN --max-results $MAX_RESULTS --
sort-order $SORT_ORDER
```

Sample request

```
aws codebuild list-sandboxes
```

Sample response


```
{
    "ids": [
        "s3-log-project-integ-test-temp1739250628149 85d64e0f-7880-41df-9a3c-
fb6597a266d2:827a5243-0841-4b69-a720-4438796f6967",
        "s3-log-project-integ-test-temp1739249999716bbd438dd-8bb8-47bd-
ba6b-0133ac65b3d3:e2fa4eab-73af-42e3-8903-92fddaf9f378",
        "s3-log-project-integ-test-
temp17392474779450fbdacc2-2d6e-4190-9ad5-28f891bb7415:cd71e456-2a4c-4db4-ada5-
da892b0bba05",
        "s3-log-project-integ-test-temp1739224628416430 1421c-5030-4fa1-b4d3-
ca15e44771c5:9e26ab3f-65e4-4896-a19c-56b1a95e630a",
        "s3-log-project-integ-test-temp17392136731949705 6d8d-6d8e-4f5a-a37c-
a62f5686731f:22d91b06-df1e-4e9c-a664-c0abb8d5920b",
        "s3-log-project-integ-test-temp1739213439503f6283f19-390c-4dc8-95a9-
c8480113384a:82cc413e-fc46-47ab-898f-ae23c83a613f",
        "s3-log-project-integ-test-temp1739054385570b1f1ddc2-0a23-4062-
bd0c-24e9e4a99b99:c02562f3-2396-42ec-98da-38e3fe5da13a",
        "s3-log-project-integ-test-temp173905400540237dab1ac-1fde-4dfb-a8f5-
c0114333dc89:d2f30493-f65e-4fa0-a7b6-08a5e77497b9",
        "s3-log-project-integ-test-
temp17390534055719c534090-7bc4-48f1-92c5-34acaec5bf1e:df5f1c8a-f017-43b7-91ba-
ad2619e2c059",
        "s3-log-project-integ-test-temp1739052719086a61813cc-
ebb9-4db4-9391-7f43cc984ee4:d61917ec-8037-4647-8d52-060349272c4a",
        "s3-log-project-integ-test-temp173898670094078b67edb-
c42f-42ed-9db2-4b5c1a5fc66a:ce33dfbc-beeb-4466-8c99-a3734a0392c7",
        "s3-log-project-integ-test-
temp17389863425584d21b7cd-32e2-4f11-9175-72c89ecaffef:046dadf0-1f3a-4d51-a2c0-
e88361924acf",
        "s3-log-project-integ-test-
temp1738985884273977ccd23-394b-46cc-90d3-7ab94cf764dc:0370dc41-9339-4b0a-91ed-51929761b244",
        "s3-log-project-integ-test-temp1738985365972241b614f-8e41-4387-
bd25-2b8351fbc9e0:076c392a-9630-47d8-85a9-116aa34edfff",
        "s3-log-project-integ-test-
temp1738985043988a51a9e2b-09d6-4d24-9c3c-1e6e21ac9fa8:6ea3949c-435b-4177-
aa4d-614d5956244c",
        "s3-log-project-integ-test-temp1738984123354c68b31ad-49d1-4f4b-981d-
b66c00565ff6:6c3fff6c-815b-48b5-ada3-737400a6dee8",
        "s3-log-project-integ-test-
temp1738977263715d4d5bf6c-370a-48bf-8ea6-905358a6cf92:968a0f54-724a-42d1-9207-6ed854b2fae8",
```

```
        "s3-log-project-integ-test-
temp173897358796816ce8d7d-2a5e-41ef-855b-4a94a8d2795d:80f9a7ce-930a-402e-934e-
d8b511d68b04",
        "s3-log-project-integ-test-temp17389730633301af5e452-0966-467c-
b684-4e36d47f568c:cabbe989-2e8a-473c-af25-32edc8c28646",
        "s3-log-project-integ-test-temp1738901503813173fd468-
b723-4d7b-9f9f-82e88d17f264:f7126a4a-b0d5-452f-814c-fea73718f805",
        "s3-log-project-integ-test-temp1738890502472c13616fb-
bd0f-4253-86cc-28b74c97a0ba:c6f197e5-3a53-45b6-863e-0e6353375437",
        "s3-log-project-integ-test-
temp1738890304468361 0daf3-8da7-43c6-8580-9978432432ce:d20aa317-8838-4966-
bbfc-85b908213df1",
        "s3-log-project-integ-test-temp173888857196780b5ab8b-e54b-44fd-a222-
c5a374fffe96:ab4b9970-ffae-47a0-b3a8-7b6790008cad",
        "s3-log-project-integ-test-temp1738888336931c11d378d-e74d-49a4-
a723-3b92e6f7daac:4922f0e8-9b7d-4119-9c9f-115cd85e703e",
        "s3-log-project-integ-test-temp17388881717651612a397-c23f-4d88-
ba87-2773cd3fc0c9:be91c3fc-418e-4feb-8a3a-ba58ff8f4e8a",
        "s3-log-project-integ-test-
temp17388879727174c3c62ed-6195-4afb-8a03-59674d0e1187:a48826a8-3c0d-43c5-
a1b5-1c98a0f978e9",
        "s3-log-project-integ-test-temp1738885948597cef305e4-b8b4-46b0-a65b-
e2d0a7b83294:c050e77d-e3f8-4829-9a60-46149628fe96",
        "s3-log-project-integ-test-temp173888561463001a7d2a8-
e4e4-4434-94db-09d3da9a9e17:8c3ac3f5-7111-4297-aec9-2470d3ead873",
        "s3-log-project-integ-test-
temp1738869855076eb19cafd-04fe-41bd-8aa0-40826d0c0d27:d25be134-05cb-404a-85da-
ac5f85d2d72c",
        "s3-project-integ-test-temp1738868157467148eacfc-d39b-49fc-a137-
e55381cd2978:4909557b-c221-4814-b4b6-7d9e93d37c35",
        "s3-project-integ-test-temp1738820926895abec0af2-
e33d-473c-9cf4-2122dd9d6876:8f5cf218-71d6-40a4-a4be-6cacebd7765f",
        "s3-project-integ-test-temp173881998877574f969a6-1c2e-4441-b463-
ab175b45ce32:04396851-c901-4986-9117-585528e3877f",
        "s3-project-integ-test-temp17388189812309abd2604-29ba-4cf6-
b6bf-073207b7db9c:540075c7-f5ec-41e8-9341-2233c09247eb",
        "s3-project-integ-test-temp1738818843474d3ea9ac1-b609-461b-
bbdb-2da245c9bc96:865d4c3c-fbfe-4ece-9c92-d0c928341404",
        "s3-project-integ-test-temp1738818542236006e9169-e6d9-4344-9b59-
f557e7aec619:1f9ffa87-da15-4290-83e2-eebdd877497b",
        "s3-project-integ-test-
temp173881809557486ad11fd-7931-48d7-81d5-499cea52a6bc:c4c2efc4-685f-4e13-8b0f-1ef85ec300b1",
        "s3-project-integ-test-temp1738817941033229410 20-3f0b-49c3-b836-
fcd818ec9484:0344cfba-de48-456d-b2a8-6566bd4a5d6e",
```

```
        "s3-project-integ-test-temp1738817680747b93d0d0b-ea16-497f-9559-
af25ee6dcfdf:654a3a55-d92a-4dc6-8da8-56fd4d40d7e1",
        "s3-project-integ-test-temp17388174027191255c3da-086c-4270-b047-
acac0b7bee0d:b7e82740-2c69-42fc-ab5a-dbf15bc016a1",
        "s3-project-integ-test-temp1738817099799016e7fa3-b9b5-46a2-
bcd5-0888c646743f:8705a6a4-79ff-427a-a1c3-85c4e8fe462e",
        "s3-project-integ-test-temp1738816479281bb0c3606-5ebf-4623-
bed5-12b60e9d3512:f23fc74b-a981-4835-8e28-375fcd4c99e4",
        "s3-project-integ-test-
temp1738816263585c939a133-4d37-482c-9238-1dbff34b7674:ca28e234-0045-4ae6-8732-938b17597f50",
        "s3-project-integ-test-
temp173881580873072d18733-8fe4-43b1-83f7-95f25bb27ccf:c6f0f55b-5736-47c7-
a3aa-1b8461a6d5ed"
    ]
}
```

Tutorial: Connecting to a sandbox using SSH

This tutorial shows you how to connect to a CodeBuild sandbox using an SSH client.

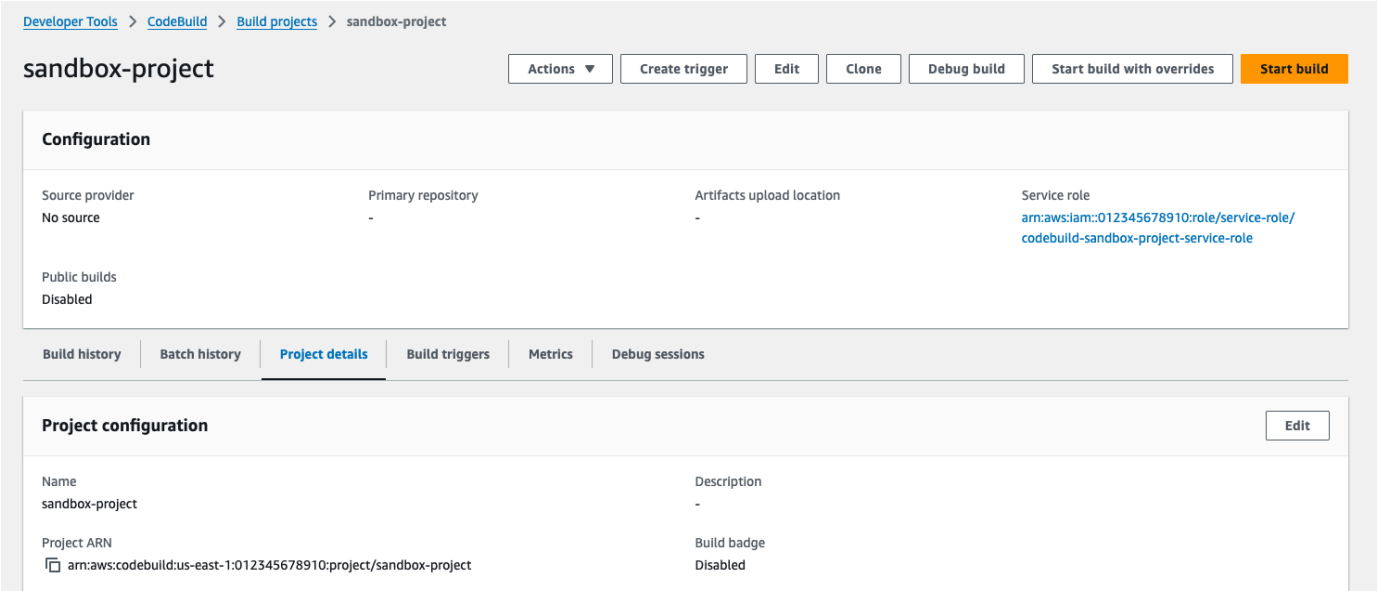
To complete this tutorial, you must first:

- Ensure you have an existing AWS CodeBuild project.
- Setup appropriate IAM permissions configured for your CodeBuild project role.
- Install and configure AWS CLI on your local machine.

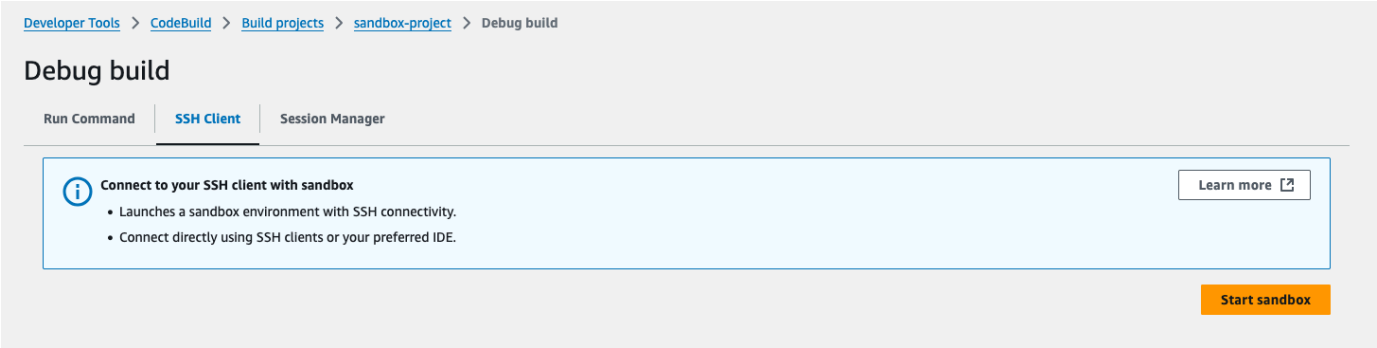
Step 1: Start a sandbox

To start a CodeBuild sandbox in the console

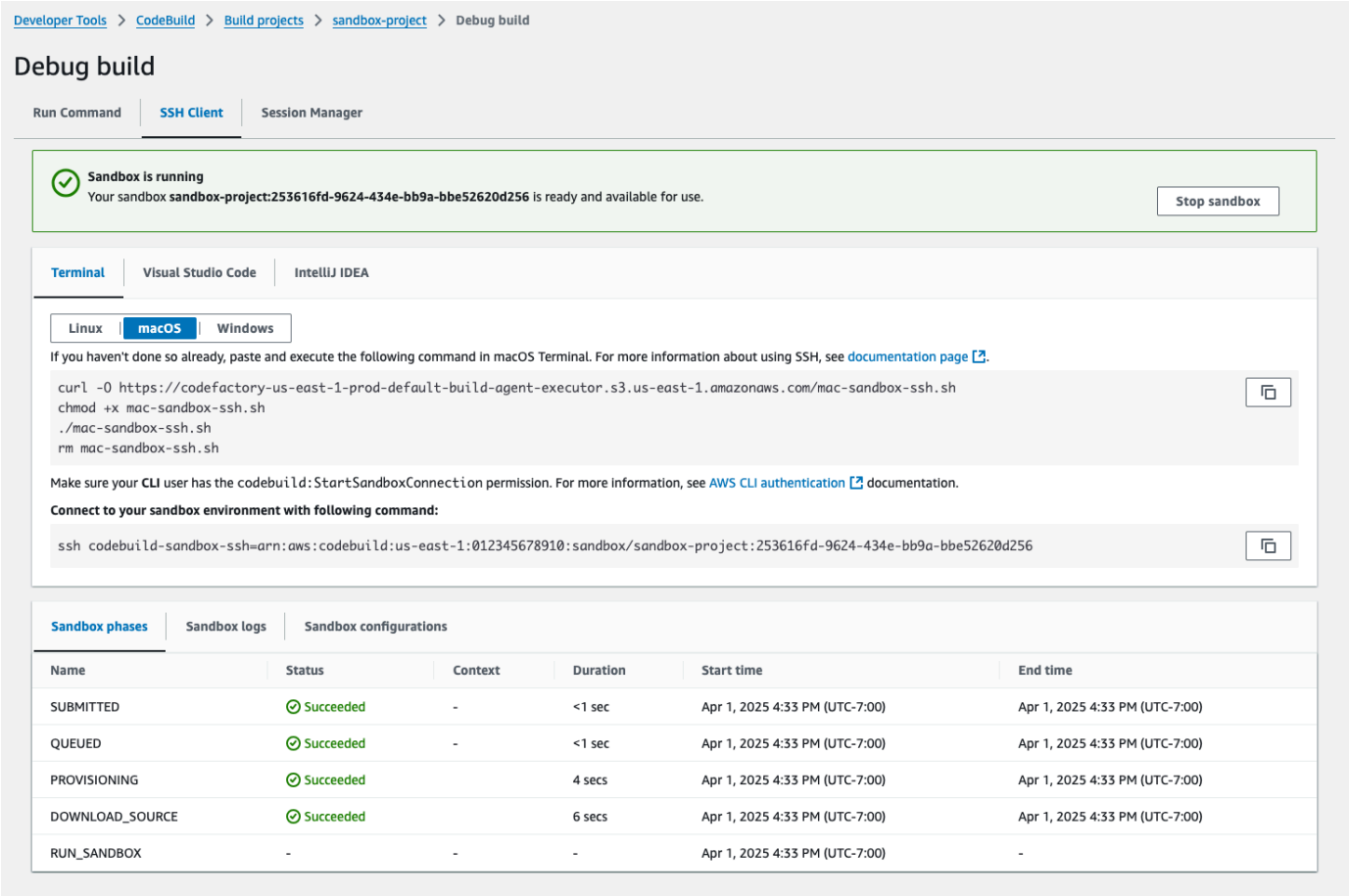
1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**. Choose the build project, and then choose **Debug build**.



3. In the **SSH Client** tab and choose **Start sandbox**.



4. The sandbox initialization process may take some time. You can connect to the sandbox when its status changes to RUN_SANDDBOX.

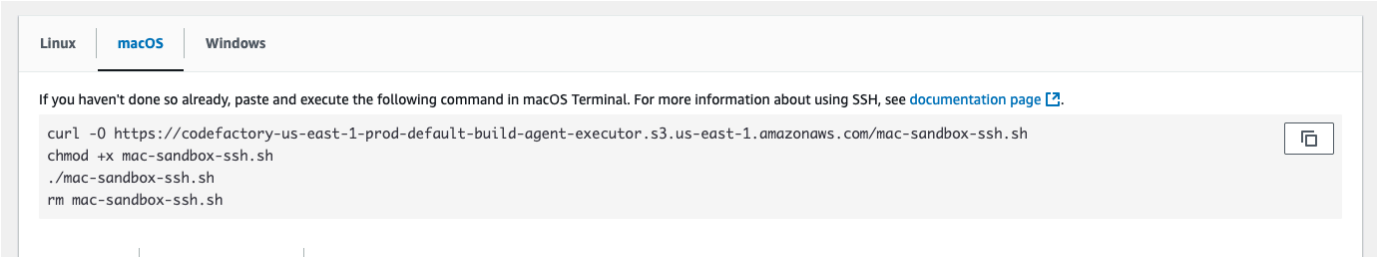


Step 2: Modify local SSH configuration

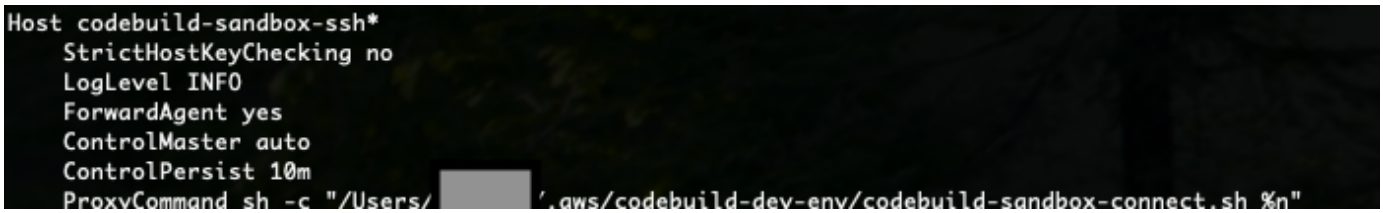
If you're connecting to sandbox for the first time, you need to perform a one-time setup process using the following steps:

To modify the local SSH configuration in the console

1. Locate the setup commands for your operating system.
2. Open your local terminal, then copy and execute the provided commands to download and run the script to set up your local SSH configuration. For example, if your operating system is macOS, use the following command:



3. The configuration script will add the required configurations for connecting to your sandboxes. You'll be prompted to accept these changes.
4. Upon successful configuration, a new SSH configuration entry for CodeBuild sandbox will be created.



Step 3: Connect to the sandbox

To modify the local SSH configuration in the console

1. Configure AWS CLI Authentication and ensure your AWS CLI user has the `codebuild:StartSandboxConnection` permission. For more information, see Authenticating using IAM user credentials for the AWS CLI in the *AWS Command Line Interface User Guide for Version 1*.
2. Connect to your sandbox with following command:

```
ssh codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:<account-id>:sandbox/<sandbox-id>
```

> **Note**
>
> To troubleshoot connection failures, use the `-v` flag to enable verbose output.
> For example, `ssh -v codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:<account-id>:sandbox/<sandbox-id>`.
> For additional troubleshooting guidance, see Troubleshooting AWS CodeBuild sandbox SSH connection issues.

Step 4: Review your results

Once connected, you can debug build failures, test build commands, experiment with configuration changes and verify environment variables and dependencies with your sandbox.

Troubleshooting AWS CodeBuild sandbox SSH connection issues

Use the information in this topic to help you identify, diagnose, and address CodeBuild sandbox SSH connection issues.

Topics

`StartSandboxConnectionInvalidInputException` error when SSH into CodeBuild sandbox environment

Issue: When attempting to connect to a CodeBuild sandbox environment using the command `ssh codebuild-sandbox-ssh=<sandbox-arn>`, you may encounter an `InvalidInputException` error such as:

```
An error occurred (InvalidInputException) when calling the StartSandboxConnection
operation: Failed to start SSM session for {sandbox-arn}
User: arn:aws:sts::<account-ID>:assumed-role/<service-role-name>/AWSCodeBuild-<UUID>
is not authorized to perform: ssm:StartSession on resource.
```

```
An error occurred (InvalidInputException) when calling the StartSandboxConnection
operation: Failed to start SSM session for
sandbox <sandbox-arn>: codebuild:<UUID> is not connected.
```

Possible cause:

- Missing Amazon EC2 Systems Manager Agent: The build image does not have the SSM agent properly installed or configured.
- Insufficient Permissions: The CodeBuild project service role lacks the required SSM permissions.

Recommended solution: If you are using a custom image for your build, do the following.

1. Install the SSM Agent. For more information, see Manually installing and uninstalling SSM Agent on Amazon EC2 instances for Linux in the . The SSM Agent version must be `3.0.1295.0` or later.
2. Copy the file, https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/7.0/amazon-ssm-agent.json to the `/etc/amazon/ssm/` directory in your image. This enables **Container Mode** in the SSM agent.
3. Ensure your CodeBuild project's service role has the following permissions, then restart the sandbox environment:

```
{
   "Effect": "Allow",
      "Action": [
         "ssmmessages:CreateControlChannel",
         "ssmmessages:CreateDataChannel",
         "ssmmessages:OpenControlChannel",
         "ssmmessages:OpenDataChannel"
      ],
      "Resource": "*"
 },
 {
    "Effect": "Allow",
    "Action": [
       "ssm:StartSession"
     ],
     "Resource": [
        "arn:aws:codebuild:region:account-id:build/*",
        "arn:aws:ssm:region::document/AWS-StartSSHSession"
     ]
 }
```

Error: "Unable to locate credentials" when SSH into CodeBuild sandbox environment

Issue: When attempting to connect to a CodeBuild sandbox environment using the command `ssh codebuild-sandbox-ssh=<sandbox-arn>`, you may encounter the following credentials error:

```
Unable to locate credentials. You can configure credentials by running
"aws configure".
```

Possible cause: AWS credentials have not been properly configured in your local environment.

Recommended solution: Configure your AWS CLI credentials by following the official documentation: Configuring settings for the AWS CLI in the *AWS Command Line Interface User Guide for Version 2*.

`StartSandboxConnectionAccessDeniedException` error when SSH into CodeBuild sandbox environment

Issue: When attempting to connect to a CodeBuild sandbox environment using the command `ssh codebuild-sandbox-ssh=<sandbox-arn>`, you may encounter the following permission error:

```
An error occurred (AccessDeniedException) when calling the StartSandboxConnection
operation:
User: arn:aws:sts::account-id:assumed-role/role-name
is not authorized to perform: codebuild:StartSandboxConnection on resource:
sandbox-arn
because no identity-based policy allows the codebuild:StartSandboxConnection action
```

Possible cause: Your AWS credentials lack the necessary CodeBuild permissions to perform this operation.

Recommended solution: Ensure that the IAM user or role associated with your AWS CLI credentials has the following permissions:

```
{
    "Effect": "Allow",
    "Action": [
       "codebuild:StartSandboxConnection"
     ],
     "Resource": [
        "arn:aws:codebuild:region:account-id:sandbox/*"
     ]
}
```

Error: "ssh: Could not resolve hostname" when SSH into CodeBuild sandbox environment

Issue: When attempting to connect to a CodeBuild sandbox environment using the command `ssh codebuild-sandbox-ssh=<sandbox-arn>`, you encounter the following hostname resolution error:

```
ssh: Could not resolve hostname
```

Possible cause: This error typically occurs when the required CodeBuild sandbox connection script has not been properly executed in your local environment.

Recommended solution:

1. Download the CodeBuild sandbox connection script.
2. Execute the script in your terminal to establish the necessary SSH configuration.
3. Retry your SSH connection to the sandbox environment.

Debug builds with Session Manager

In AWS CodeBuild, you can pause a running build and then use AWS Systems Manager Session Manager to connect to the build container and view the state of the container.

> **Note**
>
> This feature is not available in Windows environments.

Topics

Prerequisites

To allow Session Manager to be used with the build session, you must enable session connection for the build. There are two prerequisites:

- CodeBuild Linux standard curated images already have the SSM agent installed and the SSM agent ContainerMode enabled.

  If you are using a custom image for your build, do the following:

1. Install the SSM Agent. For more information, see Manually install SSM Agent on EC2 instances for Linux in the AWS Systems Manager User Guide. The SSM Agent version must be 3.0.1295.0 or later.
2. Copy the file https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/5.0/amazon-ssm-agent.json to the `/etc/amazon/ssm/` directory in your image. This enables Container Mode in the SSM agent.

> **Note**
>
> Custom images would require most updated SSM agent for this feature to work as expected.

- The CodeBuild service role must have the following SSM policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
          "ssmmessages:CreateControlChannel",
          "ssmmessages:CreateDataChannel",
          "ssmmessages:OpenControlChannel",
          "ssmmessages:OpenDataChannel"
       ],
       "Resource": "*"
     }
   ]
}
```

  You can have the CodeBuild console automatically attach this policy to your service role when you start the build. Alternatively, you can attach this policy to your service role manually.

- If you have **Auditing and logging session activity** enabled in Systems Manager preferences, the CodeBuild service role must also have additional permissions. The permissions are different, depending on where the logs are stored.

CloudWatch Logs

If using CloudWatch Logs to store your logs, add the following permission to the CodeBuild service role:

```
{
  "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

If using Amazon S3 to store your logs, add the following permission to the CodeBuild service role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetEncryptionConfiguration",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<bucket-name>",
        "arn:aws:s3:::<bucket-name>/*"
```

```
      ]
    }
  ]
}
```

For more information, see Auditing and logging session activity in the *AWS Systems Manager User Guide*.

Pause the build

To pause the build, insert the **codebuild-breakpoint** command in any of the build phases in your buildspec file. The build will be paused at this point, which allows you to connect to the build container and view the container in its current state.

For example, add the following to the build phases in your buildspec file.

```
phases:
  pre_build:
    commands:
      - echo Entered the pre_build phase...
      - echo "Hello World" > /tmp/hello-world
      - codebuild-breakpoint
```

This code creates the `/tmp/hello-world` file and then pauses the build at this point.

Start the build

To allow Session Manager to be used with the build session, you must enable session connections for the build. To do this, when starting the build, follow these steps:

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**. Choose the build project, and then choose **Start build with overrides**.
3. Choose **Advanced build overrides**.
4. In the **Environment** section, choose the **Enable session connection** option. If this option is not selected, all of the **codebuild-breakpoint** and **codebuild-resume** commands are ignored.
5. Make any other desired changes, and choose **Start build**.

6. Monitor the build status in the console. When the session is available, the **AWS Session Manager** link appears in the **Build status** section.

Connect to the build container

You can connect to the build container in one of two ways:

CodeBuild console

In a web browser, open the **AWS Session Manager** link to connect to the build container. A terminal session opens that allows you to browse and control the build container.

AWS CLI

> **Note**
>
> Your local machine must have the Session Manager plugin installed for this procedure. For more information, see Install the Session Manager Plugin for the AWS CLI in the AWS Systems Manager User Guide.

1. Call the **batch-get-builds** api with the build ID to get information about the build, including the session target identifier. The session target identifier property name varies depending on the output type of the `aws` command. This is why `--output json` is added to the command.

   ```
   aws codebuild batch-get-builds --ids <buildID> --region <region> --output json
   ```

2. Copy the `sessionTarget` property value. The `sessionTarget` property name can vary depending on the output type of the `aws` command. This is why `--output json` is added to the command in the previous step.
3. Use the following command to connect to the build container.

   ```
   aws ssm start-session --target <sessionTarget> --region <region>
   ```

For this example, verify that the `/tmp/hello-world` file exists and contains the text `Hello World`.

Resume the build

After you finish examining the build container, issue the **codebuild-resume** command from the container shell.

```
$ codebuild-resume
```

Delete builds in AWS CodeBuild

You can use the AWS CLI or the AWS SDKs to delete builds in AWS CodeBuild.

Topics

Delete builds (AWS CLI)

Run the `batch-delete-builds` command:

```
aws codebuild batch-delete-builds --ids ids
```

In the preceding command, replace the following placeholder:

- *ids*: Required string. The IDs of the builds to delete. To specify multiple builds, separate each build ID with a space. To get a list of build IDs, see the following topics:
  - View a list of build IDs (AWS CLI)
  - View a list of build IDs for a build project (AWS CLI)

If successful, a `buildsDeleted` array appears in the output, containing the Amazon Resource Name (ARN) of each build that was successfully deleted. Information about builds that were not successfully deleted appears in output within a `buildsNotDeleted` array.

For example, if you run this command:

```
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
```

Information similar to the following appears in the output:

```
{
  "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"
  ]
}
```

Delete builds (AWS SDKs)

For information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Retry builds manually in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI,or AWS SDKs to manually retry either a single build or a batch build in AWS CodeBuild.

Topics

- Retry a build manually (console)
- Retry a build manually (AWS CLI)
- Retry a build manually (AWS SDKs)

Retry a build manually (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Do one of the following:
   - If the ***build-project-name*:*build-ID*** page is displayed, choose **Retry build**.

- In the navigation pane, choose **Build history**. In the list of builds, select the box for the build, and then choose **Retry build**.
- In the navigation pane, choose **Build projects**. In the list of build projects, in the **Name** column, choose the link for the build project's name. In the list of builds, select the box for the build, and then choose **Retry build**.

> **Note**
>
> By default, only the most recent 100 builds or build projects are displayed. To view more builds or build projects, choose the gear icon, and then choose a different value for **Builds per page** or **Projects per page** or use the back and forward arrows.

Retry a build manually (AWS CLI)

- Run the **retry-build** command:

```
aws codebuild retry-build --id <build-id> --idempotency-token <idempotencyToken>
```

In the preceding command, replace the following placeholder:

- *<build-id>*: Required string. The ID of the build or batch build to retry. To get a list of build IDs, see the following topics:
  - View a list of build IDs (AWS CLI)
  - View a list of batch build IDs (AWS CLI)
  - View a list of build IDs for a build project (AWS CLI)
  - View a list of batch build IDs for a build project (AWS CLI)
- `--idempotency-token`: Optional. If you run the **retry-build** command with the option, a unique case-sensitive identifier, or token, is included with the `retry-build` request. The token is valid for 5 minutes after the request. If you repeat the `retry-build` request with the same token, but change a parameter, CodeBuild returns a parameter mismatch error.

Retry a build manually (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Retry builds automatically in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to automatically retry your builds in AWS CodeBuild. With auto-retry enabled, CodeBuild will automatically call `RetryBuild` using the project's service role after a failed build up to a specified limit. For example, if the auto-retry limit is set to two, CodeBuild will call the `RetryBuild` API to automatically retry your build for up to two additional times.

> **Note**
>
> CodeBuild does not support auto-retry for CodePipeline.

Topics

Retry a build automatically (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Choose **Create project**. For information, see Create a build project (console) and Run a build (console).
   - In **Environment**:
     - For **Auto-retry limit**, enter the maximum number of auto-retries desired after a failed build.
3. In **Environment**, choose **Additional configuration**.
4. Continue with the default values and then choose **Create build project**.

Retry a build automatically (AWS CLI)

- Run the **create-project** command:

```
aws codebuild create-project \
    --name "<project-name>" \
    --auto-retry-limit <auto-retry-limit> \
    --source "<source>" \
    --artifacts {<artifacts>} \
    --environment "{\"type\": \"environment-type>\",\"image\": \"image-type>\",
\"computeType\": \"compute-type>\"}" \
    --service-role "service-role>"
```

  In the preceding command, replace the following placeholders:

  - *<auto-retry-limit>*: Set the auto-retry limit to the maximum number of auto-retries desired after a failed build.
  - *<project-name>*, *<source>*, *<artifacts>*, *environment-type>*, *image-type>*, *compute-type>*, and *service-role>*: Set your desired project configuration settings.

Automatically retry a build (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Stop builds in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI,or AWS SDKs to stop a build in AWS CodeBuild.

Topics

- Stop a build (console)
- Stop a build (AWS CLI)
- Stop a build (AWS SDKs)

Stop a build (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Do one of the following:
   - If the ***build-project-name*:*build-ID*** page is displayed, choose **Stop build**.
   - In the navigation pane, choose **Build history**. In the list of builds, select the box for the build, and then choose **Stop build**.
   - In the navigation pane, choose **Build projects**. In the list of build projects, in the **Name** column, choose the link for the build project's name. In the list of builds, select the box for the build, and then choose **Stop build**.

> **Note**
>
> By default, only the most recent 100 builds or build projects are displayed. To view more builds or build projects, choose the gear icon, and then choose a different value for **Builds per page** or **Projects per page** or use the back and forward arrows.
> If AWS CodeBuild cannot successfully stop a build (for example, if the build process is already complete), the **Stop** button is disabled or might not appear.

Stop a build (AWS CLI)

- Run the **stop-build** command:

```
aws codebuild stop-build --id id
```

In the preceding command, replace the following placeholder:

- *id*: Required string. The ID of the build to stop. To get a list of build IDs, see the following topics:
  - View a list of build IDs (AWS CLI)
  - View a list of build IDs for a build project (AWS CLI)

If AWS CodeBuild successfully stops the build, the `buildStatus` value in the `build` object in the output is `STOPPED`.

If CodeBuild cannot successfully stop the build (for example, if the build is already complete), the `buildStatus` value in the `build` object in the output is the final build status (for example, `SUCCEEDED`).

Stop a build (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Stop batch builds in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI,or AWS SDKs to stop a batch build in AWS CodeBuild.

> **Note**
>
> If you use Lambda compute in your batch build, the in-progress Lambda build cannot be stopped.

Topics

Stop a batch build (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Do one of the following:
   - If the ***build-project-name*:*build-ID*** page is displayed, choose **Stop build**.

- In the navigation pane, choose **Build history**. In the list of builds, select the box for the build, and then choose **Stop build**.
- In the navigation pane, choose **Build projects**. In the list of build projects, in the **Name** column, choose the link for the build project's name. In the list of builds, select the box for the build, and then choose **Stop build**.

> **Note**
>
> By default, only the most recent 100 builds or build projects are displayed. To view more builds or build projects, choose the gear icon, and then choose a different value for **Builds per page** or **Projects per page** or use the back and forward arrows.

Stop a batch build (AWS CLI)

- Run the `stop-build-batch` command:

```
aws codebuild stop-build-batch --id <batch-build-id>
```

In the preceding command, replace the following placeholder:

- *<batch-build-id>*: Required string. The identifier of the batch build to stop. To get a list of batch build identifiers, see the following topics:
  - View a list of batch build IDs (AWS CLI)
  - View a list of batch build IDs for a build project (AWS CLI)

Stop a batch build (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Trigger AWS CodeBuild builds automatically

You can create a trigger on a project to schedule a build once every hour, day, or week. You can also edit a trigger to use a custom rule with an Amazon CloudWatch cron expression. For example, using a cron expression, you can schedule a build at a specific time on every weekday. For information

about creating and editing triggers, see Create AWS CodeBuild triggers and Edit AWS CodeBuild triggers.

Topics

Create AWS CodeBuild triggers

You can create a trigger on a project to schedule a build once every hour, day, or week. You can also create a trigger using a custom rule with an Amazon CloudWatch cron expression. For example, using a cron expression, you can schedule a build at a specific time every weekday.

> **Note**
>
> It is not possible to start a batch build from a build trigger, an Amazon EventBridge event, or an AWS Step Functions task.

Topics

Create AWS CodeBuild triggers (console)

Use the following procedure to create triggers using the AWS Management Console.

To create a trigger

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. Choose the link for the build project to which you want to add a trigger, and then choose the **Build triggers** tab.

> **Note**
>
> By default, the 100 most recent build projects are displayed. To view more build projects, choose the gear icon, and then choose a different value for **Projects per page** or use the back and forward arrows.

4. Choose **Create trigger**.
5. Enter a name in **Trigger name**.
6. From the **Frequency** drop-down list, choose the frequency for your trigger. If you want to create a frequency using a cron expression, choose **Custom**.
7. Specify the parameters for the frequency of your trigger. You can enter the first few characters of your selections in the text box to filter drop-down menu items.

> **Note**
>
> Start hours and minutes are zero-based. The start minute is a number between zero and 59. The start hour is a number between zero and 23. For example, a daily trigger that starts every day at 12:15 P.M. has a start hour of 12 and a start minute of 15.
> A daily trigger that starts every day at midnight has a start hour of zero and a start minute of zero. A daily trigger that starts every day at 11:59 P.M. has a start hour of 23 and a start minute of 59.

Frequency	Required Parameters	Details
Hourly	Start minute	Use the **Start minute** drop-down menu.
Daily	Start minute Start hour	Use the **Start minute** drop-down menu. Use the **Start hour** drop-down menu.
Weekly	Start minute Start hour	Use the **Start minute** drop-down menu.

Frequency	Required Parameters	Details
	Start day	Use the **Start hour** drop-down menu. Use the **Start day** drop-down menu.
Custom	Cron expression	Enter a cron expression in **Cron expression**. A cron expression has six required fields that are separated by white space. The fields specify a start value for minute, hour, day of month, month, day of week, and year. You can use wildcards to specify a range, additional values, and more. For example, the cron expression `0 9 ? * MON-FRI *` schedules a build every weekday at 9:00 A.M. For more information, see Cron Expressions in the *Amazon CloudWatch Events User Guide*.

8. Select **Enable this trigger**.
9. (Optional) Expand **Advanced section**. In **Source version**, type a version of your source.

   - For Amazon S3, enter the version ID that corresponds to the version of the input artifact you want to build. If **Source version** is left blank, the latest version is used.
   - For AWS CodeCommit, type a commit ID. If **Source version** is left blank, the default branch's HEAD commit ID is used.
   - For GitHub or GitHub Enterprise, type a commit ID, a pull request ID, a branch name, or a tag name that corresponds to the version of the source code you want to build. If you specify a pull request ID, it must use the format `pr/pull-request-ID` (for example, `pr/25`). If you

specify a branch name, the branch's HEAD commit ID is used. If **Source version** is blank, the default branch's HEAD commit ID is used.

- For Bitbucket, type a commit ID, a branch name, or a tag name that corresponds to the version of the source code you want to build. If you specify a branch name, the branch's HEAD commit ID is used. If **Source version** is blank, the default branch's HEAD commit ID is used.

10. (Optional) Specify a timeout between 5 minutes and 2160 minutes (36 hours). This value specifies how long AWS CodeBuild attempts a build before it stops. If **Hours** and **Minutes** are left blank, the default timeout value specified in the project is used.
11. Choose **Create trigger**.

Create AWS CodeBuild triggers programmatically

CodeBuild uses Amazon EventBridge rules for build triggers. You can use the EventBridge API to programmatically create build triggers for your CodeBuild projects. See Amazon EventBridge API Reference for more information.

Edit AWS CodeBuild triggers

You can edit a trigger on a project to schedule a build once every hour, day, or week. You can also edit a trigger to use a custom rule with an Amazon CloudWatch cron expression. For example, using a cron expression, you can schedule a build at a specific time on every weekday. For information about creating a trigger, see Create AWS CodeBuild triggers.

Topics

- Edit AWS CodeBuild triggers (console)
- Edit AWS CodeBuild triggers programmatically

Edit AWS CodeBuild triggers (console)

Use the following procedure to edit triggers using the AWS Management Console.

To edit a trigger

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.

2. In the navigation pane, choose **Build projects**.
3. Choose the link for the build project you want to change, and then choose the **Build triggers** tab.

> **Note**
>
> By default, the 100 most recent build projects are displayed. To view more build projects, choose the gear icon, and then choose a different value for **Projects per page** or use the back and forward arrows.

4. Choose the radio button next to the trigger you want to change, and then choose **Edit**.
5. From the **Frequency** drop-down list, choose the frequency for your trigger. If you want to create a frequency using a cron expression, choose **Custom**.
6. Specify the parameters for the frequency of your trigger. You can enter the first few characters of your selections in the text box to filter drop-down menu items.

> **Note**
>
> Start hours and minutes are zero-based. The start minute is a number between zero and 59. The start hour is a number between zero and 23. For example, a daily trigger that starts every day at 12:15 P.M. has a start hour of 12 and a start minute of 15.
> A daily trigger that starts every day at midnight has a start hour of zero and a start minute of zero. A daily trigger that starts every day at 11:59 P.M. has a start hour of 23 and a start minute of 59.

Frequency	Required Parameters	Details
Hourly	Start minute	Use the **Start minute** drop-down menu.
Daily	Start minute Start hour	Use the **Start minute** drop-down menu. Use the **Start hour** drop-down menu.

Frequency	Required Parameters	Details
Weekly	Start minute Start hour Start day	Use the **Start minute** drop-down menu. Use the **Start hour** drop-down menu. Use the **Start day** drop-down menu.
Custom	Cron expression	Enter a cron expression in **Cron expression**. A cron expression has six required fields that are separated by white space. The fields specify a start value for minute, hour, day of month, month, day of week, and year. You can use wildcards to specify a range, additiona l values, and more. For example, the cron expressio n `0 9 ? * MON-FRI *` schedules a build every weekday at 9:00 A.M. For more information, see Cron Expressions in the *Amazon CloudWatch Events User Guide*.

7. Select **Enable this trigger**.

> **Note**
>
> You can use the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/ to edit source version, timeout, and other options that are not available in AWS CodeBuild.

Edit AWS CodeBuild triggers programmatically

CodeBuild uses Amazon EventBridge rules for build triggers. You can use the EventBridge API to programmatically edit the build triggers for your CodeBuild projects. See Amazon EventBridge API Reference for more information.

View build details in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to view details about builds managed by CodeBuild.

Topics

View build details (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Do one of the following:
   - In the navigation pane, choose **Build history**. In the list of builds, in the **Build run** column, choose the link for the build.
   - In the navigation pane, choose **Build projects**. In the list of build projects, in the **Name** column, choose the link for the name of the build project. Then, in the list of builds, in the **Build run** column, choose the link for the build.

> **Note**
>
> By default, only the 10 most recent builds or build projects are displayed. To view more builds or build projects, choose the gear icon, and then choose a different value for **Builds per page** or **Projects per page** or use the back and forward arrows.

View build details (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command line reference.

Run the **batch-get-builds** command:

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

Replace the following placeholder:

- *ids*: Required string. One or more build IDs to view details about. To specify more than one build ID, separate each build ID with a space. You can specify up to 100 build IDs. To get a list of build IDs, see the following topics:
  - View a list of build IDs (AWS CLI)
  - View a list of build IDs for a build project (AWS CLI)

For example, if you run this command:

```
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
```

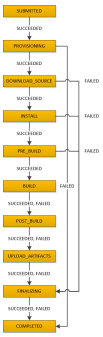
If the command is successful, data similar to that described in To view summarized build information appears in the output.

View build details (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Build phase transitions

Builds in AWS CodeBuild proceed in phases:



> **Important**
>
> The `UPLOAD_ARTIFACTS` phase is always attempted, even if the `BUILD` phase fails.

View a list of build IDs in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to view a list of build IDs for builds managed by CodeBuild.

Topics

- View a list of build IDs (console)
- View a list of build IDs (AWS CLI)
- View a list of batch build IDs (AWS CLI)
- View a list of build IDs (AWS SDKs)

View a list of build IDs (console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build history**.

> **Note**
>
> By default, only the 10 most recent builds are displayed. To view more builds, choose the gear icon, and then choose a different value for **Builds per page** or use the back and forward arrows.

View a list of build IDs (AWS CLI)

For more information about using the AWS CLI with CodeBuild, see the Command line reference.

- Run the **list-builds** command:

```
aws codebuild list-builds --sort-order sort-order --next-token next-token
```

In the preceding command, replace the following placeholders:

- *sort-order*: Optional string used to indicate how to list the build IDs. Valid values include ASCENDING and DESCENDING.
- *next-token*: Optional string. During a previous run, if there were more than 100 items in the list, only the first 100 items are returned, along with a unique string called *next token*. To get the next batch of items in the list, run this command again, adding the next token to the call. To get all of the items in the list, keep running this command with each subsequent next token, until no more next tokens are returned.

For example, if you run this command:

```
aws codebuild list-builds --sort-order ASCENDING
```

A result similar to the following might appear in the output:

```
{
  "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"
  ]
}
```

If you run this command again:

```
aws codebuild list-builds --sort-order ASCENDING --next-token 4AEA6u7J...The full
 token has been omitted for brevity...MzY2OA==
```

A result similar to the following might appear in the output:

```
{
  "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"
  ]
}
```

View a list of batch build IDs (AWS CLI)

For more information about using the AWS CLI with CodeBuild, see the Command line reference.

- Run the **list-build-batches** command:

```
aws codebuild list-build-batches --sort-order sort-order --next-token next-token
```

In the preceding command, replace the following placeholders:

- *sort-order*: Optional string used to indicate how to list the batch build IDs. Valid values include ASCENDING and DESCENDING.
- *next-token*: Optional string. During a previous run, if there were more than 100 items in the list, only the first 100 items are returned, along with a unique string called *next token*. To get the next batch of items in the list, run this command again, adding the next token to the call. To get all of the items in the list, keep running this command with each subsequent next token, until no more next tokens are returned.

For example, if you run this command:

```
aws codebuild list-build-batches --sort-order ASCENDING
```

A result similar to the following might appear in the output:

```
{
  "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"
  ]
}
```

If you run this command again:

```
aws codebuild list-build-batches --sort-order ASCENDING --next-token 4AEA6u7J...The
 full token has been omitted for brevity...MzY2OA==
```

A result similar to the following might appear in the output:

```
{
  "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"
  ]
}
```

View a list of build IDs (AWS SDKs)

For more information about using CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

View a list of build IDs for a build project in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to view a list of build IDs for a build project in CodeBuild.

Topics

View a list of build IDs for a build project (console)

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In the navigation pane, choose **Build projects**. In the list of build projects, in the **Name** column, choose the build project.

> **Note**
>
> By default, only the most recent 100 builds or build projects are displayed. To view more builds or build projects, choose the gear icon, and then choose a different value for **Builds per page** or **Projects per page** or use the back and forward arrows.

View a list of build IDs for a build project (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command line reference.

Run the **list-builds-for-project** command, as follows:

```
aws codebuild list-builds-for-project --project-name project-name --sort-order sort-order --next-token next-token
```

In the preceding command, replace the following placeholders:

- *project-name*: Required string used to indicate the name of the build project to list builds IDs for. To get a list of build projects, see View a list of build project names (AWS CLI).
- *sort-order*: Optional string used to indicate how to list the build IDs. Valid values include ASCENDING and DESCENDING.
- *next-token*: Optional string. During a previous run, if there were more than 100 items in the list, only the first 100 items are returned, along with a unique string called *next token*. To get the next batch of items in the list, run this command again, adding the next token to the call. To get all of the items in the list, keep running this command with each subsequent next token that is returned, until no more next tokens are returned.

For example, if you run this command similar to this:

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-
order ASCENDING
```

A result like the following might appear in the output:

```
{
  "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"
  ]
}
```

If you run this command again:

```
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==
```

You might see a result like the following in the output:

```
{
  "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"
  ]
}
```

View a list of batch build IDs for a build project (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command line reference.

Run the **list-build-batches-for-project** command, as follows:

```
aws codebuild list-build-batches-for-project --project-name project-name --sort-
order sort-order --next-token next-token
```

In the preceding command, replace the following placeholders:

- *project-name*: Required string used to indicate the name of the build project to list builds IDs for. To get a list of build projects, see View a list of build project names (AWS CLI).
- *sort-order*: Optional string used to indicate how to list the build IDs. Valid values include ASCENDING and DESCENDING.
- *next-token*: Optional string. During a previous run, if there were more than 100 items in the list, only the first 100 items are returned, along with a unique string called *next token*. To get the next batch of items in the list, run this command again, adding the next token to the call. To get all of the items in the list, keep running this command with each subsequent next token that is returned, until no more next tokens are returned.

For example, if you run this command similar to this:

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project --
sort-order ASCENDING
```

A result like the following might appear in the output:

```
{
  "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"
  ]
}
```

If you run this command again:

```
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==
```

You might see a result like the following in the output:

```
{
  "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"
  ]
}
```

View a list of build IDs for a build project (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference.

Test reports in AWS CodeBuild

You can create reports in CodeBuild that contain details about tests that are run during builds. You can create tests such as unit tests, configuration tests, and functional tests.

The following test report file formats are supported:

- 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**
>
> The latest supported version of `cucumber-js` is 7.3.2.

Create your test cases with any test framework that can create report files in one of these formats (for example, Surefire JUnit plugin, TestNG, or Cucumber).

To create a test report, you add a report group name to the buildspec file of a build project with information about your test cases. When you run the build project, the test cases are run and a test report is created. A new test report is created in the report group each time the test cases run. You do not need to create a report group before you run your tests. If you specify a report group name, CodeBuild creates a report group for you when you run your reports. If you want to use a report group that already exists, you specify its ARN in the buildspec file.

You can use a test report to help troubleshoot a problem during a build run. If you have many test reports from multiple builds of a build project, you can use your test reports to view trends and test and failure rates to help you optimize builds.

A report expires 30 days after it was created. You cannot view an expired test report. If you want to keep test reports for more than 30 days, you can export your test results' raw data files to an

Amazon S3 bucket. Exported test files do not expire. Information about the S3 bucket is specified when you create the report group.

> **Note**
>
> The CodeBuild service role specified in the project is used for permissions to upload to the S3 bucket.

Topics

Create test reports

To create a test report, you run a build project that is configured with one to five report groups in its buildspec file. A test report is created during the run. It contains the results of the test cases that are specified for the report groups. A new test report is generated for each subsequent build that uses the same buildspec file.

To create a test report

1. Create a build project. For information, see Create a build project in AWS CodeBuild.
2. Configure the buildspec file of your project with test report informaton:

    a. Add a `reports:` section and specify either the ARN of an existing report group, or the name of a report group.

    If you specify an ARN, CodeBuild uses that report group.

If you specify a name, CodeBuild creates a report group for you using your project name, and the name you specified, in the format *<project-name>*-*<report-group-name>*. If the named report group already exists, CodeBuild uses that report group.

b. Under the report group, specify the location of the files that contain the test results. If you use more than one report group, specify test result file locations for each one. A new test report is created each time your build project runs. For more information, see Specify test files.

c. In the `commands` section of the `build` or `post_build` sequence, specify the commands that run the tests cases you specified for your report groups. For more information, see Specify test commands .

The following is an example of a buildspec `reports` section:

```
reports:
  php-reports:
    files:
      - "reports/php/*.xml"
    file-format: "JUNITXML"
  nunit-reports:
    files:
      - "reports/nunit/*.xml"
    file-format: "NUNITXML"
```

3. Run a build of the build project. For more information, see Run AWS CodeBuild builds manually.
4. When the build is complete, choose the new build run from **Build history** on your project page. Choose **Reports** to view the test report. For more information, see View test reports for a build.

Create code coverage reports

CodeBuild allows you to generate code coverage reports for your tests. The following code coverage reports are provided:

Line coverage

Line coverage measures how many statements your tests cover. A statement is a single instruction, not including comments or conditionals.

```
line coverage = (total lines covered)/(total number of lines)
```

Branch coverage

Branch coverage measures how many branches your tests cover out of every possible branch of a control structure, such as an `if` or `case` statement.

```
branch coverage = (total branches covered)/(total number of branches)
```

The following code coverage report file formats are supported:

- JaCoCo XML
- SimpleCov JSON[1]
- Clover XML
- Cobertura XML
- LCOV INFO

[1] CodeBuild accepts JSON code coverage reports generated by simplecov, not simplecov-json.

Create a code coverage report

To create a code coverage report, you run a build project that is configured with at least one code coverage report group in its buildspec file. CodeBuild will interpret the code coverage results and provide a code coverage report for the run. A new test report is generated for each subsequent build that uses the same buildspec file.

To create a test report

1. Create a build project. For information, see Create a build project in AWS CodeBuild.
2. Configure the buildspec file of your project with test report information:

   a. Add a `reports:` section and specify the name for your report group. CodeBuild creates a report group for you using your project name and the name you specified in the format `project-name-report-group-name-in-buildspec`. If you already have a report

group you want to use, specify its ARN. If you use the name instead of the ARN, CodeBuild creates a new report group. For more information, see Reports syntax in the buildspec file.

b. Under the report group, specify the location of the files that contain the code coverage results. If you use more than one report group, specify result file locations for each report group. A new code coverage report is created each time your build project runs. For more information, see Specify test files.

   This is an example that generates a code coverage report for a JaCoCo XML results file located in test-`results/jacoco-coverage-report.xml`.

```
reports:
  jacoco-report:
    files:
      - 'test-results/jacoco-coverage-report.xml'
    file-format: 'JACOCOXML'
```

c. In the `commands` section of the `build` or `post_build` sequence, specify the commands that run the code coverage analysis. For more information, see Specify test commands .

3. Run a build of the build project. For more information, see Run AWS CodeBuild builds manually.
4. When the build is complete, choose the new build run from **Build history** on your project page. Choose **Reports** to view the code coverage report. For more information, see View test reports for a build.

Auto-discover reports in CodeBuild

With auto-discovery, CodeBuild searches through all your build files after the build phase has completed, searches for any supported report file types, and automatically creates new test and code coverage report groups and reports. For any discovered report types, CodeBuild creates new report groups with the following pattern:

```
<project-name>-<report-file-format>-AutoDiscovered
```

> **Note**
>
> If the discovered report files have the same format type, they will be placed in to the same report group or report.

Report auto-discover is configured by your project environment variables:

`CODEBUILD_CONFIG_AUTO_DISCOVER`

This variable determines whether report auto-discover is disabled during the build. By default, report auto-discover is enabled for all builds. To disable this feature, set `CODEBUILD_CONFIG_AUTO_DISCOVER` to `false`.

`CODEBUILD_CONFIG_AUTO_DISCOVER_DIR`

(Optional) This variable determines where CodeBuild searches for potential report files. Note that by default, CodeBuild searches in `**/*` by default.

These environment variables can be modified during the build phase. For example, if you only want to enable report auto-discover for builds on the `main` git branch, you can check the git branch during the build process and set `CODEBUILD_CONFIG_AUTO_DISCOVER` to false if the build is not on the `main` branch. Report auto-discover can be disabled using the console or using project environment variables.

Topics

Configure report auto-discover using the console

Use the following procedure to configure report auto-discovery using the console.

To configure report auto-discover using the console

1. Create a build project or choose a build project to edit. For information, see Create a build project in AWS CodeBuild or Change build project settings in AWS CodeBuild.
2. In **Environment**, select **Additional configuration**.
3. To disable report auto-discover, in **Report auto-discover**, select **Disable report auto-discover**.
4. (Optional) In **Auto-discover directory - optional**, enter a directory pattern for CodeBuild to search for supported report format files. Note that CodeBuild searches in `**/*` by default.

Configure report auto-discover using project environment variables

Use the following procedure to configure report auto-discovery using project environment variables.

To configure report auto-discover using project environment variables

1. Create a build project or choose a build project to edit. For information, see Create a build project in AWS CodeBuild or Change build project settings in AWS CodeBuild.
2. In **Environment variables**, do the following:
   a. To disable report auto-discover, for **Name**, enter **`CODEBUILD_CONFIG_AUTO_DISCOVER`** and for **Value**, enter **`false`**. This disables report auto-discover.
   b. (Optional) For **Name**, enter **`CODEBUILD_CONFIG_AUTO_DISCOVER_DIR`** and for **Value**, enter the directory where CodeBuild should search for supported report format files. For example, `output/*xml` searches for `.xml` files in the `output` directory

Report groups

A *report group* contains test reports and specifies shared settings. You use the buildspec file to specify the test cases to run and the commands to run them when it builds. For each report group configured in a build project, a run of the build project creates a test report. Multiple runs of a build project configured with a report group create multiple test reports in that report group, each with results of the same test cases specified for that report group.

The test cases are specified for a report group in the buildspec file of a build project. You can specify up to five report groups in one build project. When you run a build, all the test cases run. A new test report is created with the results of each test case specified for a report group. Each time you run a new build, the test cases run and a new test report is created with the new test results.

Report groups can be used in more than one build project. All test reports created with one report group share the same configuration, such as its export option and permissions, even if the test reports are created using different build projects. Test reports created with one report group in multiple build projects can contain the results from running different sets of test cases (one set of test cases for each build project). This is because you can specify different test case files for the report group in each project's buildspec file. You can also change the test case files for a report group in a build project by editing its buildspec file. Subsequent build runs create new test reports that contain the results of the test case files in the updated buildspec.

Topics

Create a report group

You can use the CodeBuild console, the AWS CLI, or a buildspec file to create a report group. Your IAM role must have the permissions required to create a report group. For more information, see Test report permissions.

Topics

Create a report group (buildspec)

A report group created using the buildspec does not export raw test result files. You can view your report group and specify export settings. For more information, see Update a report group.

To create a report group using a buildspec file

1. Choose a report group name that is not associated with a report group in your AWS account.
2. Configure the `reports` section of the buildspec file with this name. In this example, the report group name is `new-report-group` and the use test cases are created with the JUnit framework:

```
reports:
 new-report-group: #surefire junit reports
```

```
    files:
      - '**/*'
    base-directory: 'surefire/target/surefire-reports'
```

The report group name can also be specified by using environment variables in the buildspec:

```
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'
```

For more information, see Specify test files and Reports syntax in the buildspec file.

3. In the `commands` section, specify the command to run your tests. For more information, see Specify test commands .
4. Run the build. When the build is complete, a new report group is created with a name that uses the format `project-name-report-group-name`. For more information, see Report group naming.

Create a report group (console)

Use the following procedure to create a report group using the AWS Management Console.

To create a report group

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Report groups**.
3. Choose **Create report group**.

4. For **Report group name**, enter a name for your report group.
5. (Optional) For **Tags**, enter the name and value of any tags that you want supporting AWS services to use. Use **Add row** to add a tag. You can add up to 50 tags.
6. If you want to upload the raw data of your test report results to an Amazon S3 bucket:

   a. Select **Export to Amazon S3**.

   b. For **S3 bucket name**, enter the name of the S3 bucket.

   c. (Optional) For **S3 bucket owner**, enter the AWS account identifier of the account that owns the S3 bucket. This allows report data to be exported to an Amazon S3 bucket that is owned by an account other than the account running the build.

   d. For **Path prefix**, enter the path in your S3 bucket where you want to upload your test results.

   e. Select **Compress test result data in a zip file** to compress your raw test result data files.

   f. Expand **Additional configuration** to display encryption options. Choose one of the following:

   - **Default AWS managed key** to use a AWS managed key for Amazon S3. For more information, see Customer managed CMKs in the *AWS Key Management Service User Guide*. This is the default encryption option.
   - **Choose a custom key** to use a customer managed key that you create and configure. For **AWS KMS encryption key**, enter the ARN of your encryption key. Its format is `arn:aws:kms:<region-id>: <aws-account-id>:key/<key-id>` . For more information, see Creating KMS keys in the *AWS Key Management Service User Guide*.
   - **Disable artifact encryption** to disable encryption. You might choose this if you want to share your test results, or publish them to a static website. (A dynamic website can run code to decrypt test results.)

   For more information about encryption of data at rest, see Data encryption.

   > **Note**
   >
   > The CodeBuild service role specified in the project is used for permissions to upload to the S3 bucket.

7. Choose **Create report group**.

Create a report group (CLI)

Use the following procedure to create a report group using the AWS CLI.

To create a report group

1. Create a file named `CreateReportGroup.json`.
2. Depending on your requirements, copy one of the following JSON code snippets into `CreateReportGroup.json`:

   - Use the following JSON to specify that your test report group exports raw test result files to an Amazon S3 bucket.

```
{
  "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"
      }
    ]
  }
}
```

   - Replace *<bucket-name>* with your Amazon S3 bucket name and *<path>* with the path in your bucket to where you want to export the files.
   - If you want to compress the exported files, for `packaging`, specify `ZIP`. Otherwise, specify `NONE`.
   - `bucketOwner` is optional and is only required if the Amazon S3 bucket is owned by an account other than the account running the build.

- Use `encryptionDisabled` to specify whether to encrypt the exported files. If you encrypt the exported files, enter your customer managed key. For more information, see Update a report group.
- Use the following JSON to specify that your test report does not export raw test files:

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "NO_EXPORT"
  }
}
```

> **Note**
>
> The CodeBuild service role specified in the project is used for permissions to upload to the S3 bucket.

3. Run the following command:

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

Create a report group (AWS CloudFormation)

Use the following instructions to create a report group using the AWS CloudFormation template

To create a report group using the AWS CloudFormation template

You can use an AWS CloudFormation template file to create and provision a report group. For more information, see AWS CloudFormation User Guide.

The following AWS CloudFormation YAML template creates a report group that does not export raw test result files.

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
```

```
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: NO_EXPORT
```

The following AWS CloudFormation YAML template creates a report group that exports raw test result files to an Amazon S3 bucket.

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: S3
        S3Destination:
          Bucket: amzn-s3-demo-bucket
          Path: path-to-folder-for-exported-files
          Packaging: ZIP
          EncryptionKey: my-KMS-encryption-key
          EncryptionDisabled: false
```

Note

The CodeBuild service role specified in the project is used for permissions to upload to the S3 bucket.

Report group naming

When you use the AWS CLI or the AWS CodeBuild console to create a report group, you specify a name for the report group. If you use the buildspec to create a new report group, it is named using the format *project-name*-*report-group-name-specified-in-buildspec*. All reports created by running builds of that build project belong to the new report group that has the new name.

If you do not want CodeBuild to create a new report group, specify the ARN of the report group in a build project's buildspec file. You can specify a report group's ARN in multiple build projects. After each build project runs, the report group contains test reports created by each build project.

For example, if you create one report group with the name `my-report-group`, and then use its name in two different build projects named `my-project-1` and `my-project-2` and create a build of both projects, two new report groups are created. The result is three report groups with the following names:

- `my-report-group`: Does not have any test reports.
- `my-project-1-my-report-group`: Contains reports with results of tests run by the build project named `my-project-1`.
- `my-project-2-my-report-group`: Contains reports with results of tests run by the build project named `my-project-2`.

If you use the ARN of the report group named `my-report-group` in both projects, and then run builds of each project, you still have one report group (`my-report-group`). That report group contains test reports with results of tests run by both build projects.

If you choose a report group name that doesn't belong to a report group in your AWS account, and then use that name for a report group in a buildspec file and run a build of its build project, a new report group is created. The format of name of the new report group is *`project-name-new-group-name`*. For example, if there is not a report group in your AWS account with the name `new-report-group`, and specify it in a build project called `test-project`, a build run creates a new report group with the name `test-project-new-report-group`.

Share report groups

Report group sharing allows multiple AWS accounts or users to view a report group, its unexpired reports, and the test results of its reports. In this model, the account that owns the report group (owner) shares a report group with other accounts (consumers). A consumer cannot edit a report group. A report expires 30 days after it is created.

Topics

Share a report group

When you share a report group, the consumer is granted read-only access to the report group and its reports. The consumer can use the AWS CLI to view the report group, its reports, and the test case results for each report. The consumer cannot:

- View a shared report group or its reports in the CodeBuild console.
- Edit a shared report group.
- Use the ARN of the shared report group in a project to run a report. A project build that specifies a shared report group fails.

You can use the CodeBuild console to add a report group to an existing resource share. If you want to add the report group to a new resource share, you must first create it in the AWS RAM console.

To share a report group with organizational units or an entire organization, you must enable sharing with AWS Organizations. For more information, see Enable sharing with AWS Organizations in the *AWS RAM User Guide*.

You can use the CodeBuild console, AWS RAM console, or AWS CLI to share report groups that you own.

Prerequisite

To share a report group, your AWS account must own it. You cannot share a report group that has been shared with you.

To share a report group that you own (CodeBuild console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Report groups**.
3. Choose the project you want to share, and then choose **Share**. For more information, see Create a resource share in the *AWS RAM User Guide*.

To share report groups that you own (AWS RAM console)

See Creating a resource share in the *AWS RAM User Guide*.

To share report groups that you own (AWS RAM command)

Use the create-resource-share command.

To share a report group that you own (CodeBuild command)

Use the put-resource-policy command:

1. Create a file named `policy.json` and copy the following into it.

```
{
   "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. Update `policy.json` with the report group ARN and identifiers to share it with.
   The following example grants read-only access to the report group with the ARN `arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group` to Alice and the root user for the AWS account identified by 123456789012.

```
{
   "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. Run the following command.

```
aws codebuild put-resource-policy --resource-arn report-group-arn --policy file://
policy.json
```

Related services

Report group sharing integrates with AWS Resource Access Manager (AWS RAM), a service that makes it possible for you to share your AWS resources with any AWS account or through AWS Organizations. With AWS RAM, you share resources that you own by creating a *resource share* that specifies the resources and the consumers to share them with. Consumers can be individual AWS accounts, organizational units in AWS Organizations, or an entire organization in AWS Organizations.

For more information, see the *AWS RAM User Guide*.

Access report groups shared with you

To access a shared report group, a consumer's IAM role requires the `BatchGetReportGroups` permission. You can attach the following policy to their IAM role:

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetReportGroups"
    ]
}
```

For more information, see Using identity-based policies for AWS CodeBuild.

Unshare a shared report group

An unshared report group, including its reports and their test case results, can be accessed only by its owner. If you unshare a report group, any AWS account or user you previously shared it with cannot access the report group, its reports, or the results of test cases in the reports.

To unshare a shared report group that you own, you must remove it from the resource share. You can use the AWS RAM console or AWS CLI to do this.

To unshare a shared report group that you own (AWS RAM console)

See Updating a resource share in the *AWS RAM User Guide*.

To unshare a shared report group that you own (AWS RAM command)

Use the disassociate-resource-share command.

To unshare report group that you own CodeBuild command)

Run the delete-resource-policy command and specify the ARN of the report group you want to unshare:

```
aws codebuild delete-resource-policy --resource-arn report-group-arn
```

Identify a shared report group

Owners and consumers can use the AWS CLI to identify shared report groups.

To identify and get information about a shared report group and its reports, use the following commands:

- To see the ARNs of report groups shared with you, run `list-shared-report-groups`:

  ```
  aws codebuild list-shared-report-groups
  ```

- To see the ARNs of the reports in a report group, run `list-reports-for-report-group` using the report group ARN:

  ```
  aws codebuild list-reports-for-report-group --report-group-arn report-group-arn
  ```

- To see information about test cases in a report, run `describe-test-cases` using the report ARN:

```
aws codebuild describe-test-cases --report-arn report-arn
```

The output looks like the following:

```
{
    "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"
        }
    ]
}
```

Shared report group permissions

Permissions for owners

A report group owner can edit the report group and specify it in a project to run reports.

Permissions for consumers

A report group consumer can view a report group, its reports, and the test case results for its reports. A consumer cannot edit a report group or its reports, and cannot use it to create reports.

Specify test files

You specify the test result files and their location for each report group in the `reports` section of your build project's buildspec file. For more information, see Reports syntax in the buildspec file.

The following is a sample `reports` section that specifies two report groups for a build project. One is specified with its ARN, the other with a name. The `files` section specifies the files that contain the test case results. The optional `base-directory` section specifies the directory where the test case files are located. The optional `discard-paths` section specifies whether paths to test result files uploaded to an Amazon S3 bucket are discarded.

```
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
```

Specify test commands

You specify the commands that run your test cases in the `commands` section of your buildspec file. These commands run the test cases specified for your report groups in the `reports` section of your buildspec file. The following is a sample `commands` section that includes commands to run the tests in test files:

```
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
```

For more information, see Buildspec syntax.

Tag a report group in AWS CodeBuild

A *tag* is a custom attribute label that you or AWS assigns to an AWS resource. Each AWS tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, `Project`, or `Secret`). Tag keys are case sensitive.
- An optional field known as a *tag value* (for example, `111122223333`, `Production`, or a team name). Omitting the tag value is the same as using an empty string. Like tag keys, tag values are case sensitive.

Together these are known as key-value pairs. For limits on the number of tags you can have on a report group and restrictions on tag keys and values, see Tags.

Tags help you identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related. For example, you can assign the same tag to a CodeBuild report group that you assign to an Amazon S3 bucket. For more information about using tags, see the Tagging best practices whitepaper.

In CodeBuild, the primary resources are the report group and the project. You can use the CodeBuild console, the AWS CLI, CodeBuild APIs, or AWS SDKs to add, manage, and remove tags for a report group. In addition to identifying, organizing, and tracking your report group with tags, you can use tags in IAM policies to help control who can view and interact with your report group. For examples of tag-based access policies, see Using tags to control access to AWS CodeBuild resources.

Topics

Add tags to a report group

Adding tags to a report group can help you identify and organize your AWS resources and manage access to them. First, you add one or more tags (key-value pairs) to a report group. Keep in mind

that there are limits on the number of tags you can have on a report group. There are restrictions on the characters you can use in the key and value fields. For more information, see Tags. After you have tags, you can create IAM policies to manage access to the report group based on these tags. You can use the CodeBuild console or the AWS CLI to add tags to a report group.

> **Important**
>
> Adding tags to a report group can impact access to that report group. Before you add a tag to a report group, make sure to review any IAM policies that might use tags to control access to resources such as report groups. For examples of tag-based access policies, see Using tags to control access to AWS CodeBuild resources.

For more information about adding tags to a report group when you create it, see Create a report group (console).

Topics

- Add a tag to a report group (console)
- Add a tag to a report group (AWS CLI)

Add a tag to a report group (console)

You can use the CodeBuild console to add one or more tags to a CodeBuild report group.

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Report groups**, choose the name of the report group where you want to add tags.
3. In the navigation pane, choose **Settings**.
4. If no tags have been added to the report group, choose **Add tag**. You can also choose **Edit**, and then choose **Add tag**.
5. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
6. (Optional) To add another tag, choose **Add tag** again.
7. When you have finished adding tags, choose **Submit**.

Add a tag to a report group (AWS CLI)

To add a tag to a report group when you create it, see Create a report group (CLI). In `CreateReportGroup.json`, add your tags.

To add tags to an existing report group, see Update a report group (CLI) and add your tags in `UpdateReportGroupInput.json`.

In these steps, we assume that you have already installed a recent version of the AWS CLI or updated to the current version. For more information, see Installing the AWS Command Line Interface.

View tags for a report group

Tags can help you identify and organize your AWS resources and manage access to them. For more information about using tags, see the Tagging best practices whitepaper. For examples of tag-based access policies, see Deny or allow actions on report groups based on resource tags.

View tags for a report group (console)

You can use the CodeBuild console to view the tags associated with a CodeBuild report group.

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Report groups**, choose the name of the report group where you want to view tags.
3. In the navigation pane, choose **Settings**.

View tags for a report group (AWS CLI)

Follow these steps to use the AWS CLI to view the AWS tags for a report group. If no tags have been added, the returned tags list is empty.

1. Use the console or the AWS CLI to locate the ARN of your report group. Make a note of it.

   AWS CLI

   Run the following command.

   ```
   aws list-report-groups
   ```

   This command returns JSON-formatted information similar to the following:

```
{
    "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"
    ]
}
```

A report group ARN ends with its name, which you can use to identify the ARN for your report group.

Console

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Report groups**, choose the name of your report group with the tags you want to view.
3. In **Configuration** locate your report group's ARN.

2. Run the following command. Use the ARN you made a note of for the `--report-group-arns` parameter.

```
aws codebuild batch-get-report-groups --report-group-arns
 arn:aws:codebuild:region:123456789012:report-group/report-group-name
```

If successful, this command returns JSON-formatted information that contains a `tags` section similar to the following:

```
{
    ...
    "tags": {
        "Status": "Secret",
        "Project": "TestBuild"
    }
    ...
}
```

Edit tags for a report group

You can change the value for a tag associated with a report group. You can also change the name of the key, which is equivalent to removing the current tag and adding a different one with the new name and the same value as the other key. Keep in mind that there are restrictions on the characters you can use in the key and value fields. For more information, see Tags.

> **Important**
>
> Editing tags for a report group can impact access to that report group. Before you edit the name (key) or value of a tag for a report group, make sure to review any IAM policies that might use the key or value for a tag to control access to resources such as report groups. For examples of tag-based access policies, see Deny or allow actions on report groups based on resource tags.

Edit a tag for a report group (console)

You can use the CodeBuild console to edit the tags associated with a CodeBuild report group.

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Report groups**, choose the name of the report group where you want to edit tags.
3. In the navigation pane, choose **Settings**.
4. Choose **Edit**.
5. Do one of the following:
   - To change the tag, enter a new name in **Key**. Changing the name of the tag is the equivalent of removing a tag and adding a new tag with the new key name.
   - To change the value of a tag, enter a new value. If you want to change the value to nothing, delete the current value and leave the field blank.
6. When you have finished editing tags, choose **Submit**.

Edit tags for a report group (AWS CLI)

To add, change, or delete tags from a report group, see Update a report group (CLI). Update the tags in `UpdateReportGroupInput.json`.

Remove tags from a report group

You can remove one or more tags associated with a report group. Removing a tag does not delete the tag from other AWS resources that are associated with that tag.

> **Important**
>
> Removing tags for a report group can impact access to that report group. Before you remove a tag from a report group, make sure to review any IAM policies that might use the key or value for a tag to control access to resources such as report groups. For examples of tag-based access policies, see Using tags to control access to AWS CodeBuild resources.

Remove a tag from a report group (console)

You can use the CodeBuild console to remove the association between a tag and a CodeBuild report group.

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In **Report groups**, choose the name of the report group where you want to remove tags.
3. In the navigation pane, choose **Settings**.
4. Choose **Edit**.
5. Find the tag you want to remove, and then choose **Remove tag**.
6. When you have finished removing tags, choose **Submit**.

Remove a tag from a report group (AWS CLI)

Follow these steps to use the AWS CLI to remove a tag from a CodeBuild report group. Removing a tag does not delete it, but simply removes the association between the tag and the report group.

> **Note**
>
> If you delete a CodeBuild report group, all tag associations are removed from the deleted report group. You do not have to remove tags before you delete a report group.

To delete one or more tags from a report group, see Edit tags for a report group (AWS CLI). Update the `tags` section in the JSON-formatted data with an updated list of tags that does not contain the ones you want to delete. If you want to delete all tags, update the `tags` section to:

```
"tags: []"
```

Update a report group

When you update a report group, you can specify information about whether to export the raw test result data to files in an Amazon S3 bucket. If you choose to export to an S3 bucket, you can specify the following for your report group:

- Whether the raw test results files are compressed in a ZIP file.
- Whether the raw test result files are encrypted. You can specify encryption with one of the following:
  - An AWS managed key for Amazon S3.
  - A customer managed key that you create and configure.

For more information, see Data encryption.

If you use the AWS CLI to update a report group, you can also update or add tags. For more information, see Tag a report group in AWS CodeBuild.

> **Note**
>
> The CodeBuild service role specified in the project is used for permissions to upload to the S3 bucket.

Topics

- Update a report group (console)
- Update a report group (CLI)

Update a report group (console)

Use the following procedure to update a report group using the AWS Management Console.

To update a report group

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Report groups**.
3. Choose the report group you want to update.
4. Choose **Edit**.
5. Select or clear **Backup to Amazon S3**. If you selected this option, specify your export settings:
   a. For **S3 bucket name**, enter the name of the S3 bucket.
   b. For **Path prefix**, enter the path in your S3 bucket where you want to upload your test results.
   c. Select **Compress test result data in a zip file** to compress your raw test result data files.
   d. Expand **Additional configuration** to display encryption options. Choose one of the following:
      - **Default AWS managed key** to use a AWS managed key for Amazon S3. For more information, see Customer managed CMKs in the *AWS Key Management Service User Guide*. This is the default encryption option.
      - **Choose a custom key** to use a customer managed key that you create and configure. For **AWS KMS encryption key**, enter the ARN of your encryption key. Its format is `arn:aws:kms:<region-id>: <aws-account-id>:key/<key-id>` . For more information, see Creating KMS keys in the *AWS Key Management Service User Guide*.
      - **Disable artifact encryption** to disable encryption. You might choose this if you want to share your test results, or publish them to a static website. (A dynamic website can run code to decrypt test results.)

Update a report group (CLI)

Use the following procedure to update a report group using the AWS CLI.

To update a report group

1. Create a file named `UpdateReportGroupInput.json`.
2. Copy the following into `UpdateReportGroupInput.json`:

```
{
    "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. Enter the ARN of your report group in the `arn` line (for example, `"arn":"arn:aws:codebuild:region:123456789012:report-group/report-group-1"`).
4. Update `UpdateReportGroupInput.json` with the updates you want to apply to your report group.

   - If you want to update your report group to export raw test result files to an S3 bucket, update the `exportConfig` section. Replace `bucket-name` with your S3 bucket name and `path` with the path in your S3 bucket that you want to export the files to. If you want to compress the exported files, for `packaging`, specify `ZIP`. Otherwise, specify `NONE`. Use `encryptionDisabled` to specify whether to encrypt the exported files. If you encrypt the exported files, enter your customer managed key.
   - If you want to update your report group so that it does not export raw test result files to an S3 bucket, update the `exportConfig` section with the following JSON:

     ```
     {
       "exportConfig": {
           "exportConfigType": "NO_EXPORT"
       }
     ```

```
}
```

- If you want to update the report group's tags, update the `tags` section. You can change, add, or remove tags. If you want to remove all tags, update it with the following JSON:

```
"tags": []
```

5. Run the following command:

```
aws codebuild update-report-group \
--cli-input-json file://UpdateReportGroupInput.json
```

Test frameworks

The topics in this section demonstrate how to set up test reporting in AWS CodeBuild for various test frameworks.

Topics

- Set up test reporting with Jasmine
- Set up test reporting with Jest
- Set up test reporting with pytest
- Set up test reporting with RSpec

Set up test reporting with Jasmine

The following procedure demonstrates how to set up test reporting in AWS CodeBuild with the JasmineBDD testing framework.

The procedure requires the following prerequisites:

- You have an existing CodeBuild project.
- Your project is a Node.js project that is set up to use the Jasmine testing framework.

Add the `jasmine-reporters` package to the `devDependencies` section of your project's `package.json` file. This package has a collection of JavaScript reporter classes that can be used with Jasmine.

```
npm install --save-dev jasmine-reporters
```

If it's not already present, add the `test` script to your project's `package.json` file. The `test` script ensures that Jasmine is called when **npm test** is run.

```
{
  "scripts": {
    "test": "npx jasmine"
  }
}
```

CodeBuild supports the following Jasmine test reporters:

JUnitXmlReporter

Used to generate reports in the `JunitXml` format.

NUnitXmlReporter

Used to generate reports in the `NunitXml` format.

A Node.js project with Jasmine will, by default, have a `spec` sub-directory, which contains the Jasmine configuration and test scripts.

To configure Jasmine to generate reports in the `JunitXML` format, instantiate the `JUnitXmlReporter` reporter by adding the following code to your tests.

```
var reporters = require('jasmine-reporters');

var junitReporter = new reporters.JUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(junitReporter);
```

To configure Jasmine to generate reports in the `NunitXML` format, instantiate the `NUnitXmlReporter` reporter by adding the following code to your tests.

```
var reporters = require('jasmine-reporters');
```

```
var nunitReporter = new reporters.NUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(nunitReporter)
```

The test reports are exported to the file specified by *<test report directory>*/*<report filename>*.

In your `buildspec.yml` file, add/update the following sections.

```
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>
```

If you are using the the `NunitXml` report format, change the `file-format` value to the following.

```
    file-format: NUNITXML
```

Set up test reporting with Jest

The following procedure demonstrates how to set up test reporting in AWS CodeBuild with the Jest testing framework.

The procedure requires the following prerequisites:

- You have an existing CodeBuild project.
- Your project is a Node.js project that is set up to use the Jest testing framework.

Add the `jest-junit` package to the `devDependencies` section of your project's `package.json` file. CodeBuild uses this package to generate reports in the `JunitXml` format.

```
npm install --save-dev jest-junit
```

If it's not already present, add the `test` script to your project's `package.json` file. The `test` script ensures that Jest is called when **npm test** is run.

```
{
  "scripts": {
    "test": "jest"
  }
}
```

Configure Jest to use the `JunitXml` reporter by adding the following to your Jest configuration file. If your project does not have a Jest configuration file, create a file named `jest.config.js` in the root of your project and add the following. The test reports are exported to the file specified by *<test report directory>*/*<report filename>*.

```
module.exports = {
  reporters: [
    'default',
    [ 'jest-junit', {
      outputDirectory: <test report directory>,
      outputName: <report filename>,
    } ]
  ]
};
```

In your `buildspec.yml` file, add/update the following sections.

```
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>
```

Set up test reporting with pytest

The following procedure demonstrates how to set up test reporting in AWS CodeBuild with the pytest testing framework.

The procedure requires the following prerequisites:

- You have an existing CodeBuild project.
- Your project is a Python project that is set up to use the pytest testing framework.

Add the following entry to either the `build` or `post_build` phase of your `buildspec.yml` file. This code automatically discovers tests in the current directory and exports the test reports to the file specified by *<test report directory>*/*<report filename>*. The report uses the `JunitXml` format.

```
      - python -m pytest --junitxml=<test report directory>/<report filename>
```

In your `buildspec.yml` file, add/update the following sections.

```
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
```

Set up test reporting with RSpec

The following procedure demonstrates how to set up test reporting in AWS CodeBuild with the RSpec testing framework.

The procedure requires the following prerequisites:

- You have an existing CodeBuild project.
- Your project is a Ruby project that is set up to use the RSpec testing framework.

Add/update the following in your `buildspec.yml` file. This code runs the tests in the *<test source directory>* directory and exports the test reports to the file specified by *<test report directory>*/*<report filename>*. The report uses the `JunitXml` format.

```
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
```

View test reports

You can view details about a test report, such as information about its test cases, pass and fail numbers, and how long it took for it to run. You can view test reports grouped by build run, report group, or your AWS account. Choose a test report in the console to see its details and results of its test cases.

You can see view test reports that are not expired. Test reports expire 30 days after they are created. You cannot view an expired report in CodeBuild.

Topics

- View test reports for a build
- View test reports for a report group
- View test reports in your AWS account

View test reports for a build

To view test reports for a build

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. Locate the build you want to view. If you know the project that ran the build that created the test report:

    1. In the navigation pane, choose **Build projects**, and then choose the project with the build that ran the test report you want to view.
    2. Choose **Build history**, and then choose the build that ran created the reports you want to view.

    You can also locate the build in the build history for your AWS account:

    1. In the navigation pane, choose **Build history**, and then choose the build that created the test reports you want to view.

3. In the build page, choose **Reports**, and then choose a test report to see its details.

View test reports for a report group

To view test reports in a report group

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Report groups**.
3. Choose the report group that contains the test reports you want to view.
4. Choose a test report to see its details.

View test reports in your AWS account

To view test reports in your AWS account

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Report history**.
3. Choose a test report to see its details.

Test report permissions

This topic describes important information about permissions related to test reporting.

Topics

- IAM role for test reports
- Permissions for test reporting operations
- Test reporting permissions examples

IAM role for test reports

To run a test report, and to update a project to include test reports, your IAM role requires the following permissions. These permissions are included in the predefined AWS managed policies.

If you want to add test reporting to an existing build project, you must add these permissions yourself.

- `CreateReportGroup`
- `CreateReport`
- `UpdateReport`
- `BatchPutTestCases`

To run a code coverage report, your IAM role must also include the `BatchPutCodeCoverages` permission.

> **Note**
>
> `BatchPutTestCases`, `CreateReport`, `UpdateReport`, and `BatchPutCodeCoverages` are not public permissions. You cannot call a corresponding AWS CLI command or SDK method for these permissions.

To make sure you have these permissions, you can attach the following policy to your IAM role:

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

We recommend that you restrict this policy to only those report groups you must use. The following restricts permissions to only the report groups with the two ARNs in the policy:

```
{
    "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"
    ]
}
```

The following restricts permissions to only report groups created by running builds of a project named `my-project`:

```
{
    "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

The CodeBuild service role specified in the project is used for permissions to upload to the S3 bucket.

Permissions for test reporting operations

You can specify permissions for the following test reporting CodeBuild API operations:

- `BatchGetReportGroups`
- `BatchGetReports`
- `CreateReportGroup`
- `DeleteReportGroup`
- `DeleteReport`
- `DescribeTestCases`
- `ListReportGroups`
- `ListReports`
- `ListReportsForReportGroup`
- `UpdateReportGroup`

For more information, see AWS CodeBuild permissions reference.

Test reporting permissions examples

For information about sample policies related to test reporting, see the following:

- Allow a user to change a report group
- Allow a user to create a report group
- Allow a user to delete a report
- Allow a user to delete a report group
- Allow a user to get information about report groups
- Allow a user to get information about reports
- Allow a user to get a list of report groups
- Allow a user to get a list of reports
- Allow a user to get a list of reports for a report group
- Allow a user to get a list of test cases for a report

Test report statuses

The status of a test report can be one of the following:

- `GENERATING`: The run of the test cases is still in progress.

- `DELETING`: The test report is being deleted. When a test report is deleted, its test cases are also deleted. Raw test result data files exported to an S3 bucket are not deleted.
- `INCOMPLETE`: The test report was not completed. This status might be returned for one of the following reasons:
  - A problem with the configuration of the report group that specifies this report's test cases. For example, the path to the test cases under the report group in the buildspec file might be incorrect.
  - The IAM user that ran the build does not have permissions to run tests. For more information, see Test report permissions.
  - The build was not completed because of an error that is not related to the tests.
- `SUCCEEDED`: All test cases were successful.
- `FAILED`: Some of the test cases were not successful.

Each test case returns a status. The status for a test case can be one of the following:

- `SUCCEEDED`: The test case passed.
- `FAILED`: The test case failed.
- `ERROR`: The test case resulted in an unexpected error.
- `SKIPPED`: The test case did not run.
- `UNKNOWN`: The test case returned a status other than `SUCCEEDED`, `FAILED`, `ERROR`, or `SKIPPED`.

A test report can have a maximum of 500 test case results. If more than 500 test cases are run, CodeBuild prioritizes tests with the status `FAILED` and truncates the test case results.

Use AWS CodeBuild with Amazon Virtual Private Cloud

Typically, AWS CodeBuild cannot access resources in a VPC. To enable access, you must provide additional VPC-specific configuration information in your CodeBuild project configuration. This includes the VPC ID, the VPC subnet IDs, and the VPC security group IDs. VPC-enabled builds can then access resources inside your VPC. For more information about setting up a VPC in Amazon VPC, see the Amazon VPC User Guide.

Topics

Use cases

VPC connectivity from AWS CodeBuild builds makes it possible to:

- Run integration tests from your build against data in an Amazon RDS database that's isolated on a private subnet.
- Query data in an Amazon ElastiCache cluster directly from tests.
- Interact with internal web services hosted on Amazon EC2, Amazon ECS, or services that use internal Elastic Load Balancing.
- Retrieve dependencies from self-hosted, internal artifact repositories, such as PyPI for Python, Maven for Java, and npm for Node.js.
- Access objects in an S3 bucket configured to allow access through an Amazon VPC endpoint only.
- Query external web services that require fixed IP addresses through the Elastic IP address of the NAT gateway or NAT instance associated with your subnet.

Your builds can access any resource that's hosted in your VPC.

Best practices for VPCs

Use this checklist when you set up a VPC to work with CodeBuild.

- Set up your VPC with public and private subnets, and a NAT gateway. The NAT gateway must reside in a public subnet. For more information, see VPC with public and private subnets (NAT) in the *Amazon VPC User Guide*.

  > **Important**
  >
  > You need a NAT gateway or NAT instance to use CodeBuild with your VPC so that CodeBuild can reach public endpoints (for example, to run CLI commands when running builds). You cannot use the internet gateway instead of a NAT gateway or a NAT instance because CodeBuild does not support assigning Elastic IP addresses to the network interfaces that it creates, and auto-assigning a public IP address is not supported by Amazon EC2 for any network interfaces created outside of Amazon EC2 instance launches.

- Include multiple Availability Zones with your VPC.
- Make sure that your security groups have no inbound (ingress) traffic allowed to your builds. CodeBuild does not have specific requirements for outbound traffic, but you must allow access to any Internet resources required for your build, such as GitHub or Amazon S3.

  For more information, see Security groups rules in the *Amazon VPC User Guide*.
- Set up separate subnets for your builds.
- When you set up your CodeBuild projects to access your VPC, choose private subnets only.

For more information about setting up a VPC in Amazon VPC, see the Amazon VPC User Guide.

For more information about using AWS CloudFormation to configure a VPC to use the CodeBuild VPC feature, see the AWS CloudFormation VPC template.

Limitations of VPCs

- VPC connectivity from CodeBuild is not supported for shared VPCs.

Allow Amazon VPC access in your CodeBuild projects

Include these settings in your VPC configuration:

- For **VPC ID**, choose the VPC ID that CodeBuild uses.
- For **Subnets**, choose a private subnet with NAT translation that includes or has routes to the resources used by CodeBuild.
- For **Security Groups**, choose the security groups that CodeBuild uses to allow access to resources in the VPCs.

To use the console to create a build project, see Create a build project (console). When you create or change your CodeBuild project, in **VPC**, choose your VPC ID, subnets, and security groups.

To use the AWS CLI to create a build project, see Create a build project (AWS CLI). If you are using the AWS CLI with CodeBuild, the service role used by CodeBuild to interact with services on behalf of the IAM user must have a policy attached. For information, see Allow CodeBuild access to AWS services required to create a VPC network interface.

The *vpcConfig* object should include your *vpcId*, *securityGroupIds*, and *subnets*.

- *vpcId*: Required. The VPC ID that CodeBuild uses. Run this command to get a list of all Amazon VPC IDs in your Region:

```
aws ec2 describe-vpcs
```

- *subnets*: Required. The subnet IDs that include resources used by CodeBuild. Run this command obtain these IDs:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

> **Note**
>
> Replace `us-east-1` with your Region.

- *securityGroupIds*: Required. The security group IDs used by CodeBuild to allow access to resources in the VPCs. Run this command to obtain these IDs:

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

> **Note**
>
> Replace `us-east-1` with your Region.

Troubleshoot your VPC setup

Use the information that appears in the error message to help you identify, diagnose, and address issues.

The following are some guidelines to assist you when troubleshooting a common CodeBuild VPC error: `Build does not have internet connectivity. Please check subnet network configuration`.

1. Make sure that your internet gateway is attached to VPC.
2. Make sure that the route table for your public subnet points to the internet gateway.
3. Make sure that your network ACLs allow traffic to flow.
4. Make sure that your security groups allow traffic to flow.
5. Troubleshoot your NAT gateway.
6. Make sure that the route table for private subnets points to the NAT gateway.
7. Make sure that the service role used by CodeBuild to interact with services on behalf of the IAM user has the permissions in this policy. For more information, see Allow CodeBuild to interact with other AWS services.

   If CodeBuild is missing permissions, you might receive an error that says, `Unexpected EC2 error: UnauthorizedOperation`. This error can occur if CodeBuild does not have the Amazon EC2 permissions required to work with a VPC.

Use VPC endpoints

You can improve the security of your builds by configuring AWS CodeBuild to use an interface VPC endpoint. Interface endpoints are powered by PrivateLink, a technology that you can use to

privately access Amazon EC2 and CodeBuild by using private IP addresses. PrivateLink restricts all network traffic between your managed instances, CodeBuild, and Amazon EC2 to the Amazon network. (Managed instances don't have access to the internet.) Also, you don't need an internet gateway, NAT device, or virtual private gateway. You are not required to configure PrivateLink, but it's recommended. For more information about PrivateLink and VPC endpoints, see What is AWS PrivateLink?.

Before you create VPC endpoints

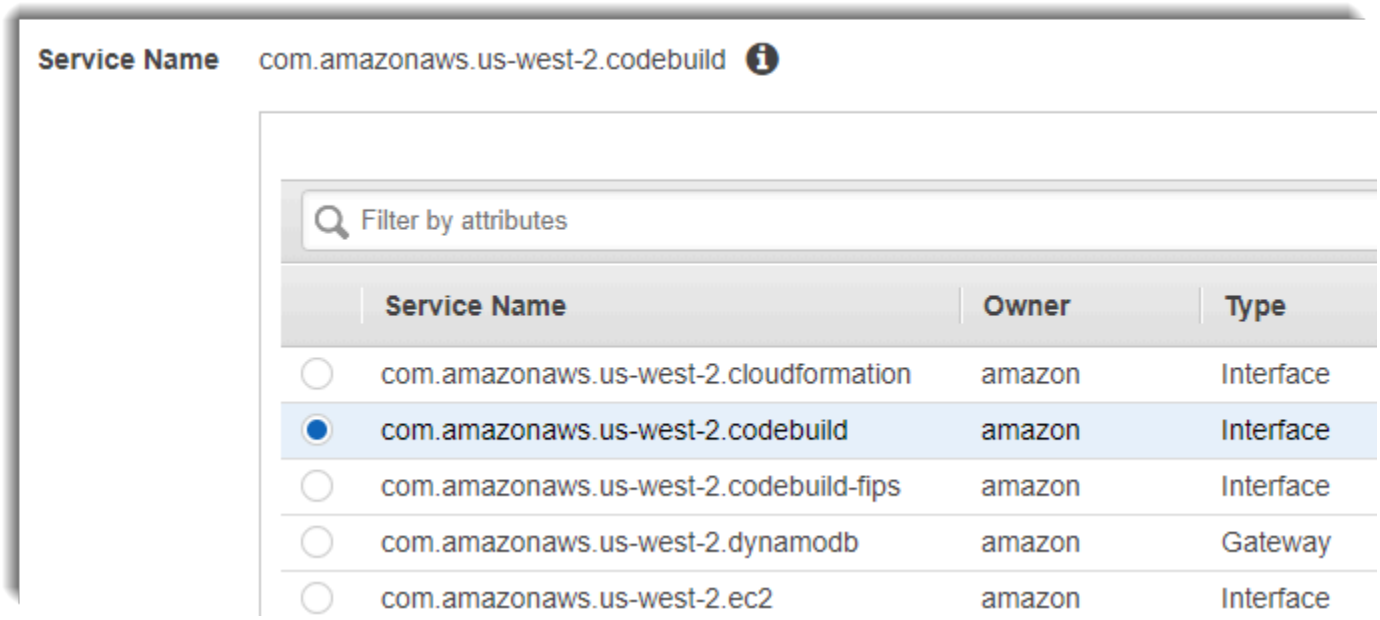
Before you configure VPC endpoints for AWS CodeBuild, be aware of the following restrictions and limitations.

> **Note**
>
> Use a NAT gateway if you want to use CodeBuild with AWS services that do not support Amazon VPC PrivateLink connections.

- VPC endpoints support Amazon-provided DNS through Amazon Route 53 only. If you want to use your own DNS, you can use conditional DNS forwarding. For more information, see DHCP option sets in the *Amazon VPC User Guide*.
- VPC endpoints currently do not support cross-Region requests. Make sure that you create your endpoint in the same AWS Region as any S3 buckets that store your build input and output. You can use the Amazon S3 console or the get-bucket-location command to find the location of your bucket. Use a Region-specific Amazon S3 endpoint to access your bucket (for example, `<bucket-name>.s3-us-west-2.amazonaws.com`). For more information about Region-specific endpoints for Amazon S3, see Amazon Simple Storage Service in the *Amazon Web Services General Reference*. If you use the AWS CLI to make requests to Amazon S3, set your default Region to the same Region where your bucket was created, or use the `--region` parameter in your requests.

Create VPC endpoints for CodeBuild

Follow the instructions in Creating an interface endpoint to create the endpoint `com.amazonaws.region.codebuild`. This is a VPC endpoint for AWS CodeBuild.



region represents the region identifier for an AWS Region supported by CodeBuild, such as `us-east-2` for the US East (Ohio) Region. For a list of supported AWS Regions, see CodeBuild in the *AWS General Reference*. The endpoint is prepopulated with the Region you specified when you signed in to AWS. If you change your Region, the VPC endpoint is updated accordingly.

Create a VPC endpoint policy for CodeBuild

You can create a policy for Amazon VPC endpoints for AWS CodeBuild in which you can specify:

- The principal that can perform actions.
- The actions that can be performed.
- The resources that can have actions performed on them.

The following example policy specifies that all principals can only start and view builds for the `project-name` project.

```
{
    "Statement": [
        {
            "Action": [
                "codebuild:ListBuildsForProject",
                "codebuild:StartBuild",
                "codebuild:BatchGetBuilds"
            ],
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:codebuild:region-ID:account-ID:project/project-name",
            "Principal": "*"
        }
    ]
}
```

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Use AWS CodeBuild with a managed proxy server

To run AWS CodeBuild reserved capacity fleets in a managed proxy server, you must configure the proxy server to allow or deny traffic to and from external sites using proxy rules. Note that running reserved capacity fleets in a managed proxy server is not supported for VPC, Windows, or MacOS.

> **Important**
>
> There are additional costs based on the duration that a proxy configuration is present in the fleet. For more information, see https://aws.amazon.com/codebuild/pricing/.

Topics

- Configure a managed proxy configuration for reserved capacity fleets
- Run a CodeBuild reserved capacity fleet

Configure a managed proxy configuration for reserved capacity fleets

To configure a managed proxy server for your reserved capacity fleet, you must enable this feature when creating your fleet in your console or using the AWS CLI. There are several properties which you need to define:

Define proxy configurations - optional

Proxy configurations that apply network access control to your reserved capacity instances.

Default behavior

Defines the behavior of outgoing traffic.

Allow

Allows outgoing traffic to all destinations by default.

Deny

Denies outgoing traffic to all destinations by default.

Proxy rules

Specifies destination domains to restrict network access control to.

To define proxy configurations in your console, see Create a reserved capacity fleet for instructions. To define proxy configurations using the AWS CLI, you can do so by modifying the following JSON syntax and saving your results:

```
"proxyConfiguration": {
    "defaultBehavior": "ALLOW_ALL" | "DENY_ALL",
    "orderedProxyRules": [
        {
            "type": "DOMAIN" | "IP",
            "effect": "ALLOW" | "DENY",
            "entities": [
                "destination"
            ]
        }
    ]
}
```

Your JSON file may look similar to the following:

```
"proxyConfiguration": {
    "defaultBehavior": "DENY_ALL",
    "orderedProxyRules": [
        {
            "type": "DOMAIN",
            "effect": "ALLOW",
            "entities": [
                "github.com"
            ]
        }
    ]
}
```

Run a CodeBuild reserved capacity fleet

When running AWS CodeBuild reserved capacity fleets with your managed proxy server, CodeBuild will automatically set its HTTP_PROXY and HTTPS_PROXY environment variables with the managed proxy addresses. If your dependency software has its own configuration and does not adhere to the environment variables, you can refer to these values and update your software configuration in your build commands to properly route your build traffic through the managed proxy. For more information, see Create a build project in AWS CodeBuild and Change build project settings in AWS CodeBuild.

Use AWS CodeBuild with a proxy server

You can use AWS CodeBuild with a proxy server to regulate HTTP and HTTPS traffic to and from the internet. To run CodeBuild with a proxy server, you install a proxy server in a public subnet and CodeBuild in a private subnet in a VPC.

There are two primary use cases for running CodeBuild in a proxy server:

- It eliminates the use of a NAT gateway or NAT instance in your VPC.
- It lets you specify the URLs that instances in the proxy server can access and the URLs to which the proxy server denies access.

You can use CodeBuild with two types of proxy servers. For both, the proxy server runs in a public subnet and CodeBuild runs in a private subnet.

- **Explicit proxy**: If you use an explicit proxy server, you must configure NO_PROXY, HTTP_PROXY, and HTTPS_PROXY environment variables in CodeBuild at the project level. For more information, see Change build project settings in AWS CodeBuild and Create a build project in AWS CodeBuild.
- **Transparent proxy**: If you use a transparent proxy server, no special configuration is required.

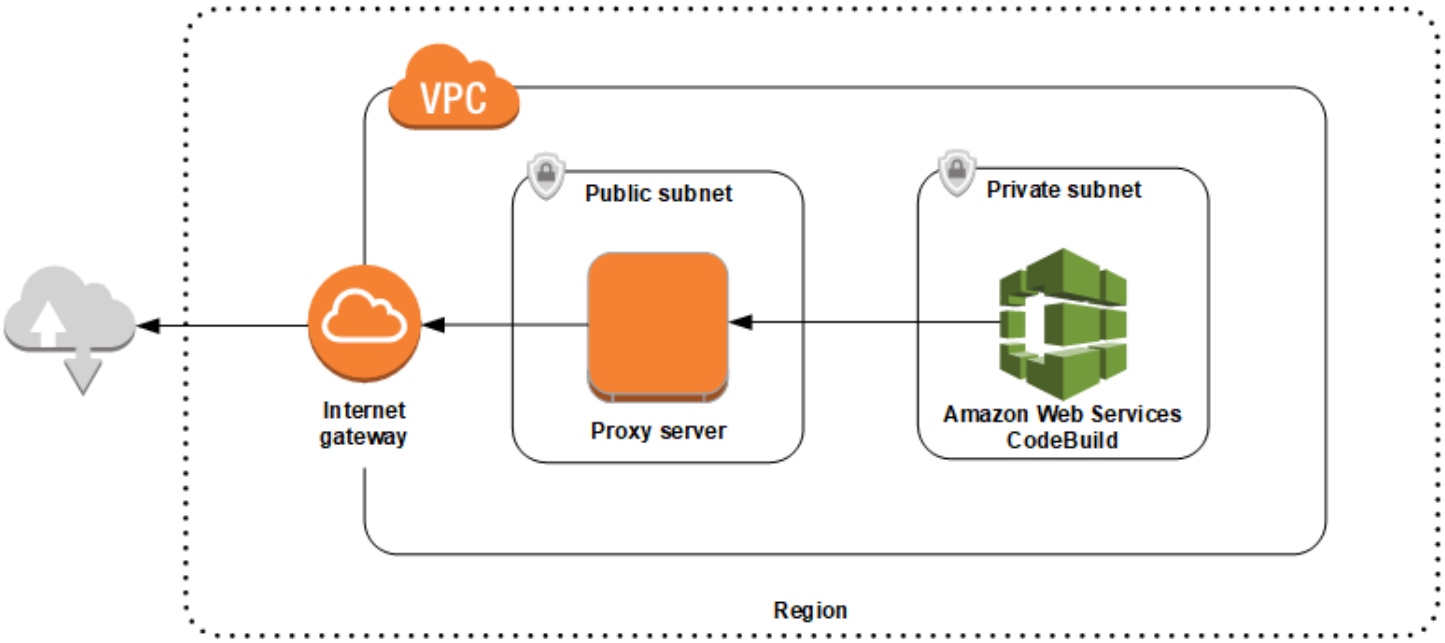
Topics

Set up components required to run CodeBuild in a proxy server

You need these components to run AWS CodeBuild in a transparent or explicit proxy server:

- A VPC.
- One public subnet in your VPC for the proxy server.
- One private subnet in your VPC for CodeBuild.
- An internet gateway that allows communcation between the VPC and the internet.

The following diagram shows how the components interact.



Set up a VPC, subnets, and a network gateway

The following steps are required to run AWS CodeBuild in a transparent or explicit proxy server.

1. Create a VPC. For information, see Creating a VPC in the *Amazon VPC User Guide*.
2. Create two subnets in your VPC. One is a public subnet named `Public Subnet` in which your proxy server runs. The other is a private subnet named `Private Subnet` in which CodeBuild runs.

   For information, see Creating a subnet in your VPC.
3. Create and attach an internet gateway to your VPC. For more information, see Creating and attaching an internet gateway.

4. Add a rule to the default route table that routes outgoing traffic from the VPC (0.0.0.0/0) to the internet gateway. For information, see Adding and removing routes from a route table.
5. Add a rule to the default security group of your VPC that allows ingress SSH traffic (TCP 22) from your VPC (0.0.0.0/0).
6. Follow the instructions in Launching an instance using the launch instance wizard in the *Amazon EC2 User Guide* to launch an Amazon Linux instance. When you run the wizard, choose the following options:
   - In **Choose an Instance Type**, choose an Amazon Linux Amazon Machine Image (AMI).
   - In **Subnet**, choose the public subnet you created earlier in this topic. If you used the suggested name, it is **Public Subnet**.
   - In **Auto-assign Public IP**, choose **Enable**.
   - On the **Configure Security Group** page, for **Assign a security group**, choose **Select an existing security group**. Next, choose the default security group.
   - After you choose **Launch**, choose an existing key pair or create one.

   Choose the default settings for all other options.
7. After your EC2 instance is running, disable source/destination checks. For information, see Disabling Source/Destination checks in the *Amazon VPC User Guide*.
8. Create a route table in your VPC. Add a rule to the route table that routes traffic destined for the internet to your proxy server. Associate this route table with your private subnet. This is required so that outbound requests from instances in your private subnet, where CodeBuild runs, are always routed through the proxy server.

Install and configure a proxy server

There are many proxy servers from which to choose. An open-source proxy server, Squid, is used here to demonstrate how AWS CodeBuild runs in a proxy server. You can apply the same concepts to other proxy servers.

To install Squid, use a yum repo by running the following commands:

```
sudo yum update -y
sudo yum install -y squid
```

After you install Squid, edit its `squid.conf` file using the instructions later in this topic.

Configure Squid for HTTPS traffic

For HTTPS, the HTTP traffic is encapsulated in a Transport Layer Security (TLS) connection. Squid uses a feature called SslPeekAndSplice to retrieve the Server Name Indication (SNI) from the TLS initiation that contains the requested internet host. This is required so Squid does not need to unencrypt HTTPS traffic. To enable SslPeekAndSplice, Squid requires a certificate. Create this certificate using OpenSSL:

```
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

For HTTP, Squid does not require configuration. From all HTTP/1.1 request messages, it can retrieve the host header field, which specifies the internet host that is being requested.

Run CodeBuild in an explicit proxy server

To run AWS CodeBuild in an explicit proxy server, you must configure the proxy server to allow or deny traffic to and from external sites, and then configure the HTTP_PROXY and HTTPS_PROXY environment variables.

Topics

- Configure Squid as an explicit proxy server
- Create a CodeBuild project
- Explicit proxy server sample squid.conf file

Configure Squid as an explicit proxy server

To configure the Squid proxy server to be explicit, you must make the following modifications to its `/etc/squid/squid.conf` file:

- Remove the following default access control list (ACL) rules.

```
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
```

  Add the following in place of the default ACL rules you removed. The first line allows requests from your VPC. The next two lines grant your proxy server access to destination URLs that might be used by AWS CodeBuild. Edit the regular expression in the last line to specify S3 buckets or a CodeCommit repository in an AWS Region. For example:

  - If your source is Amazon S3, use the command **acl download_src dstdom_regex .*s3\.us-west-1\.amazonaws\.com**to grant access to S3 buckets in the `us-west-1` Region.
  - If your source is AWS CodeCommit, use `git-codecommit.<`*`your-region`*`>.amazonaws.com` to add an AWS Region to an allow list.

```
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
```

- Replace `http_access allow localnet` with the following:

```
http_access allow localnet allowed_sites
http_access allow localnet download_src
```

- If you want your build to upload logs and artifacts, do one of the following:

  1. Before the `http_access deny all` statement, insert the following statements. They allow CodeBuild to access CloudWatch and Amazon S3. Access to CloudWatch is required so that CodeBuild can create CloudWatch logs. Access to Amazon S3 is required for uploading artifacts and Amazon S3 caching.

     - ```
       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
```

- After you save `squid.conf`, run the following command:

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service squid restart
```

2. Add `proxy` to your buildspec file. For more information, see Buildspec syntax.

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
      - command
```

> **Note**
>
> If you receive a RequestError timeout error, see RequestError timeout error when running CodeBuild in a proxy server.

For more information, see Explicit proxy server sample `squid.conf` file later in this topic.

Create a CodeBuild project

To run AWS CodeBuild with your explicit proxy server, set its HTTP_PROXY and HTTPS_PROXY environment variables with the private IP address of the EC2 instance you created for your proxy server and port 3128 at the project level. The private IP address looks like `http://your-ec2-private-ip-address:3128`. For more information, see Create a build project in AWS CodeBuild and Change build project settings in AWS CodeBuild.

Use the following command to view the Squid proxy access log:

```
sudo tail -f /var/log/squid/access.log
```

Explicit proxy server sample `squid.conf` file

The following is an example of a `squid.conf` file that is configured for an explicit proxy server.

```
 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
```

Run CodeBuild in a transparent proxy server

To run AWS CodeBuild in a transparent proxy server, you must configure the proxy server with access to the websites and domains it interacts with.

Topics

- Configure Squid as a transparent proxy server

- Create a CodeBuild project

Configure Squid as a transparent proxy server

To configure a proxy server to be transparent, you must grant it access to the domains and websites you want it to access. To run AWS CodeBuild with a transparent proxy server, you must grant it access to `amazonaws.com`. You must also grant access to other websites CodeBuild uses. These vary, depending on how you create your CodeBuild projects. Example websites are those for repositories such as GitHub, Bitbucket, Yum, and Maven. To grant Squid access to specific domains and websites, use a command similar to the following to update the `squid.conf` file. This sample command grants access to `amazonaws.com`, `github.com`, and `bitbucket.com`. You can edit this sample to grant access to other websites.

```
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
```

Incoming requests from instances in the private subnet must redirect to the Squid ports. Squid listens on port 3129 for HTTP traffic (instead of 80) and 3130 for HTTPS traffic (instead of 443). Use the **iptables** command to route traffic:

```
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
```

Create a CodeBuild project

After you configure your proxy server, you can use it with AWS CodeBuild in a private subnet without more configuration. Every HTTP and HTTPS request goes through the public proxy server. Use the following command to view the Squid proxy access log:

```
sudo tail -f /var/log/squid/access.log
```

Run a package manager and other tools in a proxy server

Use the following procedures to run a package manager and other tools in a proxy server.

To run a tool, such as a package manager, in a proxy server

1. Add the tool to the allow list in your proxy server by adding statements to your `squid.conf` file.
2. Add a line to your buildspec file that points to the private endpoint of your proxy server.

The following examples demonstrate how to do this for `apt-get`, `curl`, and `maven`. If you use a different tool, the same principles apply. Add it to an allow list in the `squid.conf` file and add a command to your buildspec file to make CodeBuild aware of your proxy server's endpoint.

To run `apt-get` in a proxy server

1. Add the following statements to your `squid.conf` file to add `apt-get` to an allow list in your proxy server. The first three lines allow `apt-get` to run in the build environment.

   ```
   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. Add the following statement in your buildspec file so that `apt-get` commands look for the proxy configuration in `/etc/apt/apt.conf.d/00proxy`.

```
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
```

To run `curl` in a proxy server

1. Add the following to your `squid.conf` file to add `curl` to an allow list in your build environment.

```
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. Add the following statement in your buildspec file so `curl` uses the private proxy server to access the website you added to the `squid.conf`. In this example, the website is `google.com`.

```
curl -x <private-ip-of-proxy-server>:3128 https://www.google.com
```

To run `maven` in a proxy server

1. Add the following to your `squid.conf` file to add `maven` to an allow list in your build environment.

```
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. Add the following statement to your buildspec file.

```
maven clean install -DproxySet=true -DproxyHost=<private-ip-of-proxy-server> -
DproxyPort=3128
```

AWS CloudFormation VPC template

AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly, by using template files to create and delete a collection of resources together as a single unit (a *stack*). For more information, see the AWS CloudFormation User Guide.

The following is an AWS CloudFormation YAML template for configuring a VPC to use AWS CodeBuild. This file is also available in samples.zip.

```
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
```

Logging and monitoring in AWS CodeBuild

Logging and monitoring is an important part of maintaining the reliability, availability, and performance of AWS CodeBuild and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure, if one occurs. AWS provides the following tools for monitoring your CodeBuild resources and builds and for responding to potential incidents.

Topics

Log AWS CodeBuild API calls with AWS CloudTrail

AWS CodeBuild is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in CodeBuild. CloudTrail captures all API calls for CodeBuild as events, including calls from the CodeBuild console and from code calls to the CodeBuild APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an S3 bucket, including events for CodeBuild. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to CodeBuild, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Topics

About AWS CodeBuild information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in CodeBuild, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail event history in the *AWS CloudTrail User Guide*.

For an ongoing record of events in your AWS account, including events for CodeBuild, create a trail. A trail enables CloudTrail to deliver log files to an S3 bucket. By default, when you create a trail in the console, the trail applies to all regions. The trail logs events from all regions in the AWS partition and delivers the log files to the S3 bucket that you specify. You can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All CodeBuild actions are logged by CloudTrail and are documented in the CodeBuild API Reference. For example, calls to the `CreateProject` (in the AWS CLI, `create-project`), `StartBuild` (in the AWS CLI, `start-project`), and `UpdateProject` (in the AWS CLI, `update-project`) actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity elementin the *AWS CloudTrail User Guide*.

About AWS CodeBuild log file entries

A trail is a configuration that enables delivery of events as log files to an S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

Note

To protect sensitive information, the following are hidden in CodeBuild logs:

- AWS access key IDs. For more information, see Managing Access Keys for IAM Users in the *AWS Identity and Access Management User Guide*.
- Strings specified using the Parameter Store. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the *Amazon EC2 Systems Manager User Guide*.
- Strings specified using AWS Secrets Manager. For more information, see Key management.

The following example shows a CloudTrail log entry that demonstrates creating a build project in CodeBuild.

```
{
  "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"
}
```

Monitor CodeBuild builds with CloudWatch

You can use Amazon CloudWatch to watch your builds, report when something is wrong, and take automatic actions when appropriate. You can monitor your builds at two levels:

Project level

These metrics are for all builds in the specified project. To see metrics for a project, specify `ProjectName` for the dimension in CloudWatch.

AWS account level

These metrics are for all builds in an account. To see metrics at the AWS account level, do not enter a dimension in CloudWatch. Build resource utilization metrics are not available at the AWS account level.

CloudWatch metrics show the behavior of your builds over time. For example, you can monitor:

- How many builds were attempted in a build project or an AWS account over time.
- How many builds were successful in a build project or an AWS account over time.
- How many builds failed in a build project or an AWS account over time.
- How much time CodeBuild spent running builds in a build project or an AWS account over time.
- Build resource utilization for a build or an entire build project. Build resource utilization metrics include metrics such as CPU, memory, and storage utilization.

For more information, see View CodeBuild metrics.

CodeBuild CloudWatch metrics

The following metrics can be tracked per AWS account or build project. For more information about using CloudWatch with CodeBuild, see Monitor CodeBuild builds with CloudWatch.

BuildDuration

Measures the duration of the build's `BUILD` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

Builds

Measures the number of builds triggered.

Units: Count

Valid CloudWatch statistics: Sum

DownloadSourceDuration

Measures the duration of the build's `DOWNLOAD_SOURCE` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

Duration

Measures the duration of all builds over time.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

FailedBuilds

Measures the number of builds that failed because of client error or a timeout.

Units: Count

Valid CloudWatch statistics: Sum

FinalizingDuration

Measures the duration of the build's `FINALIZING` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

InstallDuration

Measures the duration of the build's `INSTALL` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

PostBuildDuration

Measures the duration of the build's `POST_BUILD` phase

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

PreBuildDuration

Measures the duration of the build's `PRE_BUILD` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

ProvisioningDuration

Measures the duration of the build's `PROVISIONING` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

QueuedDuration

Measures the duration of the build's `QUEUED` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

SubmittedDuration

Measures the duration of the build's `SUBMITTED` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

SucceededBuilds

Measures the number of successful builds.

Units: Count

Valid CloudWatch statistics: Sum

UploadArtifactsDuration

Measures the duration of the build's `UPLOAD_ARTIFACTS` phase.

Units: Seconds

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

CodeBuild CloudWatch resource utilization metrics

Note

CodeBuild resource utilization metrics are only available in the following regions:

- 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
- Europe (Ireland) Region
- Europe (London) Region
- Europe (Paris) Region
- South America (São Paulo) Region
- US East (N. Virginia) Region
- US East (Ohio) Region
- US West (N. California) Region
- US West (Oregon) Region

The following resource utilization metrics can be tracked. For more information about using CloudWatch with CodeBuild, see Monitor CodeBuild builds with CloudWatch.

CPUUtilized

The number of CPU units of allocated processing used by the build container.

Units: CPU units

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

CPUUtilizedPercent

The percentage of allocated processing used by the build container.

Units: Percent

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

MemoryUtilized

The number of megabytes of memory used by the build container.

Units: Megabytes

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

MemoryUtilizedPercent

The percentage of allocated memory used by the build container.

Units: Percent

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

StorageReadBytes

The storage read speed used by the build container.

Units: Bytes/second

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

StorageWriteBytes

The storage write speed used by the build container.

Units: Bytes/second

Valid CloudWatch statistics: Average (recommended), Maximum, Minimum

CodeBuild CloudWatch dimensions

CodeBuild provides the following CloudWatch metric dimensions. If none of these are specified, the metrics are for the current AWS account.

BuildId, BuildNumber, ProjectName

Metrics are provided for a build identifier, build number, and project name.

ProjectName

Metrics are provided for a project name.

CodeBuild CloudWatch alarms

You can use the CloudWatch console to create alarms based on CodeBuild metrics so you can react if something goes wrong with your builds. The two metrics that are most useful with alarms are described in the following bullets. For more information about using CloudWatch with CodeBuild, see Monitor CodeBuild builds with CloudWatch.

- `FailedBuild`. You can create an alarm that is triggered when a certain number of failed builds are detected within a predetermined number of seconds. In CloudWatch, you specify the number of seconds and how many failed builds trigger an alarm.
- `Duration`. You can create an alarm that is triggered when a build takes longer than expected. You specify how many seconds must elapse after a build is started and before a build is completed before the alarm is triggered.

For information about how to create alarms for CodeBuild metrics, see Monitor CodeBuild builds with CloudWatch alarms. For more information about alarms, see Creating Amazon CloudWatch alarms in the *Amazon CloudWatch User Guide*.

View CodeBuild metrics

AWS CodeBuild monitors functions on your behalf and reports metrics through Amazon CloudWatch. These metrics include the number of total builds, failed builds, successful builds, and the duration of builds.

You can use the CodeBuild console or the CloudWatch console to monitor metrics for CodeBuild. The following procedures show you how to view metrics.

Topics

View build metrics (CodeBuild console)

> **Note**
>
> You can't customize the metrics or the graphs used to display them in the CodeBuild console. If you want to customize the display, use the Amazon CloudWatch console to view your build metrics.

Account-level metrics

To view AWS account-level metrics

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Account metrics**.

Project-level metrics

To view project-level metrics

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. In the list of build projects, in the **Name** column, choose the project where you want to view metrics.
4. Choose the **Metrics** tab.

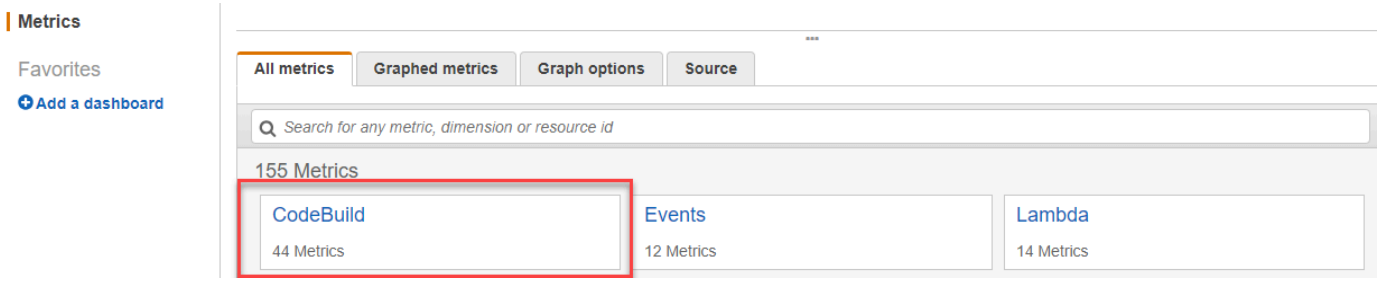
View build metrics (Amazon CloudWatch console)

You can customize the metrics and the graphs used to display them with the CloudWatch console.

Account-level metrics

To view account-level metrics

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics** tab, choose **CodeBuild**.

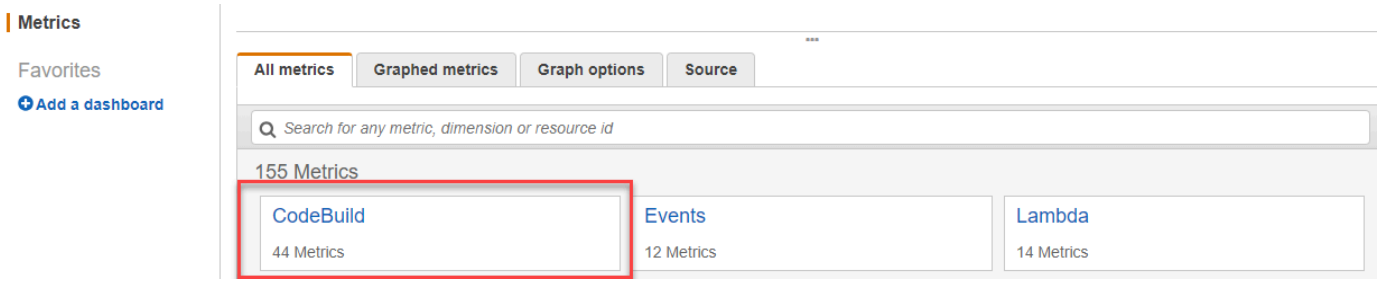


4. Choose **Account Metrics**.
5. Choose one or more projects and metrics. For each project, you can choose the **SucceededBuilds**, **FailedBuilds**, **Builds**, and **Duration** metrics. All selected project and metric combinations are displayed in the graph on the page.

Project-level metrics

To view project-level metrics

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics** tab, choose **CodeBuild**.



4. Choose **By Project**.

5. Choose one or more project and metric combinations. For each project, you can choose the **SucceededBuilds**, **FailedBuilds**, **Builds**, and **Duration** metrics. All selected project and metric combinations are displayed in the graph on the page.
6. (Optional) You can customize your metrics and graphs. For example, from the drop-down list in the **Statistic** column, you can choose a different statistic to display. Or from the drop-down menu in the **Period** column, you can choose a different time period to use to monitor the metrics.

   For more information, see Graph metrics and View available metrics in the *Amazon CloudWatch User Guide*.

View CodeBuild resource utilization metrics

AWS CodeBuild monitors build resource utilization on your behalf and reports metrics through Amazon CloudWatch. These include metrics such as CPU, memory, and storage utilization.

> **Note**
>
> CodeBuild resource utilization metrics are only recorded for builds that run for more than one minute.

You can use the CodeBuild console or the CloudWatch console to monitor resource utilization metrics for CodeBuild.

> **Note**
>
> CodeBuild resource utilization metrics are only available in the following regions:
>
> - 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

- Europe (Ireland) Region
- Europe (London) Region
- Europe (Paris) Region
- South America (São Paulo) Region
- US East (N. Virginia) Region
- US East (Ohio) Region
- US West (N. California) Region
- US West (Oregon) Region

The following procedures show you how to access your resource utilization metrics.

Topics

Access resource utilization metrics (CodeBuild console)

Note

You can't customize the metrics or the graphs used to display them in the CodeBuild console. If you want to customize the display, use the Amazon CloudWatch console to view your build metrics.

Project-level resource utilization metrics

To access project-level resource utilization metrics

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build projects**.
3. In the list of build projects, in the **Name** column, choose the project you want to view the utilization metrics for.

4. Choose the **Metrics** tab. The resource utilization metrics are displayed in the **Resource utilization metrics** section.
5. To view the project-level resource utilization metrics in the CloudWatch console, choose **View in CloudWatch** in the **Resource utilization metrics** section.

Build-level resource utilization metrics

To access build-level resource utilization metrics

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home.
2. In the navigation pane, choose **Build history**.
3. In the list of builds, in the **Build run** column, choose the build you want to view the utilization metrics for.
4. Choose the **Resource utilization** tab.
5. To view the build-level resource utilization metrics in the CloudWatch console, choose **View in CloudWatch** in the **Resource utilization metrics** section.

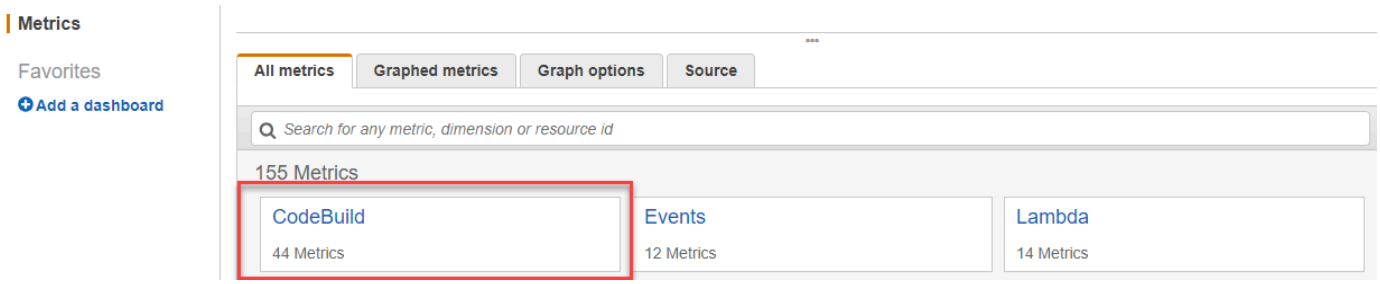
Access resource utilization metrics (Amazon CloudWatch console)

The Amazon CloudWatch console can be used to access CodeBuild resource utilization metrics.

Project-level resource utilization metrics

To access project-level resource utilization metrics

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics** tab, choose **CodeBuild**.



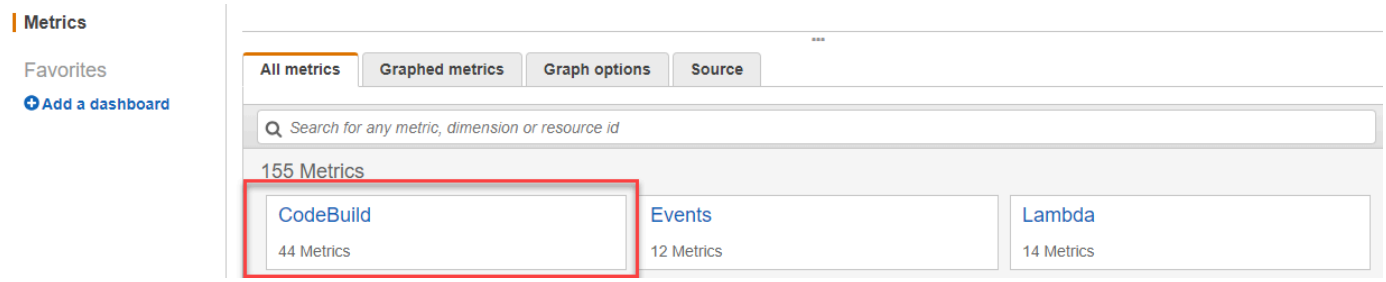
4. Choose **By Project**.
5. Choose one or more project and metric combinations to add to the graph. All selected project and metric combinations are displayed in the graph on the page.
6. (Optional) You can customize your metrics and graphs from the **Graphed metrics** tab. For example, from the drop-down list in the **Statistic** column, you can choose a different statistic to display. Or from the drop-down menu in the **Period** column, you can choose a different time period to use to monitor the metrics.

   For more information, see Graphing metrics and Viewing available metrics in the *Amazon CloudWatch User Guide*.

Build-level resource utilization metrics

To access build-level resource utilization metrics

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics** tab, choose **CodeBuild**.



4. Choose **BuildId, BuildNumber, ProjectName**.
5. Choose one or more build and metric combinations to add to the graph. All selected build and metric combinations are displayed in the graph on the page.
6. (Optional) You can customize your metrics and graphs from the **Graphed metrics** tab. For example, from the drop-down list in the **Statistic** column, you can choose a different statistic to display. Or from the drop-down menu in the **Period** column, you can choose a different time period to use to monitor the metrics.

   For more information, see Graphing metrics and Viewing available metrics in the *Amazon CloudWatch User Guide*.

Monitor CodeBuild builds with CloudWatch alarms

You can create a CloudWatch alarm for your builds. An alarm watches a single metric over a period of time that you specify and performs one or more actions based on the value of the metric relative to a specified threshold over a number of time periods. Using native CloudWatch alarm functionality, you can specify any of the actions supported by CloudWatch when a threshold is exceeded. For example, you can specify that an Amazon SNS notification is sent when more than three builds in your account fail within fifteen minutes.

To create a CloudWatch alarm for a CodeBuild metric

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**.
3. Choose **Create Alarm**.
4. Under **CloudWatch Metrics by Category**, choose **CodeBuild Metrics**. If you know you want only project-level metrics, choose **By Project**. If you know you want only account-level metrics, choose **Account Metrics**.
5. On **Create Alarm**, if it isn't already selected, choose **Select Metric**.
6. Choose a metric for which you want to create an alarm. The options are **By Project** or **Account Metrics**.
7. Choose **Next** or **Define Alarm** and then create your alarm. For more information, see Creating Amazon CloudWatch alarms in the *Amazon CloudWatch User Guide*. For more information about setting up Amazon SNS notifications when an alarm is triggered, see Set up Amazon SNS notifications in the *Amazon SNS Developer Guide*.
8. Choose **Create Alarm**.

Security in AWS CodeBuild

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security and compliance is a shared responsibility between AWS and you. This shared model can help relieve your operational burden: AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the service facilities. You assume responsibility and management of the guest operating system (including updates and security patches) and other associated application software. You're also responsible for the configuration of the AWS provided security group firewall. Your responsibilities vary with the services you use, the integration of those services into your IT environment, and applicable laws and regulations. Therefore, you should carefully consider the services that your organization uses. For more information, see Shared responsibility model.

To learn how to secure your CodeBuild resources, see the following topics.

Topics

Data protection in AWS CodeBuild

The AWS shared responsibility model applies to data protection in AWS CodeBuild. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ.

For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with CodeBuild or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

To protect sensitive information, the following are hidden in CodeBuild logs:

- Strings specified using the Parameter Store in CodeBuild project environment variables or the buildspec `env/parameter-store` section. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store console walkthrough in the *Amazon EC2 Systems Manager User Guide*.
- Strings specified using AWS Secrets Manager in CodeBuild project environment variables or the buildspec `env/secrets-manager` section. For more information, see Key management.

For more information about data protection, see the AWS shared responsibility model and GDPR blog post on the *AWS Security Blog*.

Topics

Data encryption

Encryption is an important part of CodeBuild security. Some encryption, such as for data in-transit, is provided by default and does not require you to do anything. Other encryption, such as for data at-rest, you can configure when you create your project or build.

- **Encryption of data at-rest** - Build artifacts, such as a cache, logs, exported raw test report data files, and build results, are encrypted by default using AWS managed keys. If you do not want to use these KMS keys, you must create and configure a customer managed key. For more information Creating KMS Keys and AWS Key Management Service concepts in the *AWS Key Management Service User Guide*.
  - You can store the identifier of the AWS KMS key that CodeBuild uses to encrypt the build output artifact in the `CODEBUILD_KMS_KEY_ID` environment variable. For more information, see Environment variables in build environments
  - You can specify a customer managed key when you create a build project. For more information, see Set the Encryption Key Using the Console and Set the encryption key using the CLI.

  The Amazon Elastic Block Store volumes of your build fleet are encrypted by default using AWS managed keys.
- **Encryption of data in-transit** - All communication between customers and CodeBuild and between CodeBuild and its downstream dependencies is protected using TLS connections that are signed using the Signature Version 4 signing process. All CodeBuild endpoints use SHA-256 certificates that are managed by AWS Private Certificate Authority. For more information, see Signature Version 4 signing process and What is ACM PCA.
- **Build artifact encryption** - The CodeBuild service role associated with the build project requires access to a KMS key in order to encrypt its build output artifacts. By default, CodeBuild uses an AWS managed key for Amazon S3 in your AWS account. If you do not want to use this AWS

managed key, you must create and configure a customer managed key. For more information, see Encrypt build outputs and Creating keys in the *AWS KMS Developer Guide*.

Key management

You can protect your content from unauthorized use through encryption. Store your encryption keys in AWS Secrets Manager, and then give the CodeBuild service role associated with the build project permission to obtain the encryption keys from your Secrets Manager account. For more information, see Encrypt build outputs using a customer managed key, Create a build project in AWS CodeBuild, Run AWS CodeBuild builds manually, and Tutorial: Storing and retrieving a secret.

Use the `CODEBUILD_KMS_KEY_ID` environment variable in a build command to obtain the AWS KMS key identifier. For more information, see Environment variables in build environments.

You can use Secrets Manager to protect credentials to a private registry that stores a Docker image used for your runtime environment. For more information, see Private registry with AWS Secrets Manager sample for CodeBuild.

Traffic privacy

You can improve the security of your builds by configuring CodeBuild to use an interface VPC endpoint. To do this, you do not need an internet gateway, NAT device, or virtual private gateway. It also is not required to configure PrivateLink, though it is recommended. For more information, see Use VPC endpoints. For more information about PrivateLink and VPC endpoints, see AWS PrivateLink and Accessing AWS services through PrivateLink.

Identity and access management in AWS CodeBuild

Access to AWS CodeBuild requires credentials. Those credentials must have permissions to access AWS resources, such as storing and retrieving build artifacts in S3 buckets and viewing Amazon CloudWatch Logs for builds. The following sections describe how you can use AWS Identity and Access Management (IAM) and CodeBuild to help secure access to your resources:

Overview of managing access permissions to your AWS CodeBuild resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

> **Note**
>
> An account administrator (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the *IAM User Guide*.

When you grant permissions, you decide who is getting the permissions, the resources they can access, and the actions that can be performed on those resources.

Topics

AWS CodeBuild resources and operations

In AWS CodeBuild, the primary resource is a build project. In a policy, you use an Amazon Resource Name (ARN) to identify the resource the policy applies to. Builds are also resources and have ARNs associated with them. For more information, see Amazon Resource Names (ARN) and AWS Service Namespaces in the *Amazon Web Services General Reference*.

Resource type	ARN format
Build project	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:project/` *project-name*
Build	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:build/` *build-ID*

Resource type	ARN format
Report group	arn:aws:codebuild: *region-ID* :*account-ID* :report-group/ *report-group-name*
Report	arn:aws:codebuild: *region-ID* :*account-ID* :report/*report-ID*
Fleet	arn:aws:codebuild: *region-ID* :*account-ID* :fleet/*fleet-ID*
All CodeBuild resources	arn:aws:codebuild:*
All CodeBuild resources owned by the specified account in the specified AWS Region	arn:aws:codebuild: *region-ID* :*account-ID* :*

Important

When using the reserved capacity feature, data cached on fleet instances, including source files, Docker layers, and cached directories specified in the buildspec, can be accessible to other projects within the same account. This is by design and allows projects within the same account to share fleet instances.

Note

Most AWS services treat a colon (:) or a forward slash (/) as the same character in ARNs. However, CodeBuild uses an exact match in resource patterns and rules. Be sure to use the correct characters when you create event patterns so that they match the ARN syntax in the resource.

For example, you can indicate a specific build project (*myBuildProject*) in your statement using its ARN as follows:

```
"Resource": "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject"
```

To specify all resources, or if an API action does not support ARNs, use the wildcard character (*) in the `Resource` element as follows:

```
"Resource": "*"
```

Some CodeBuild API actions accept multiple resources (for example, `BatchGetProjects`). To specify multiple resources in a single statement, separate their ARNs with commas, as follows:

```
"Resource": [
  "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject",
  "arn:aws:codebuild:us-east-2:123456789012:project/myOtherBuildProject"
]
```

CodeBuild provides a set of operations to work with the CodeBuild resources. For a list, see AWS CodeBuild permissions reference.

Understanding resource ownership

The AWS account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the AWS account of the principal entity (that is, the root account, an user, or an IAM role) that authenticates the resource creation request. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a rule, your AWS account is the owner of the CodeBuild resource.
- If you create an user in your AWS account and grant permissions to create CodeBuild resources to that user, the user can create CodeBuild resources. However, your AWS account, to which the user belongs, owns the CodeBuild resources.
- If you create an IAM role in your AWS account with permissions to create CodeBuild resources, anyone who can assume the role can create CodeBuild resources. Your AWS account, to which the role belongs, owns the CodeBuild resources.

Managing access to resources

A permissions policy describes who has access to which resources.

> **Note**
>
> This section discusses the use of IAM in AWS CodeBuild. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as identity-based policies (IAM policies). Policies attached to a resource are referred to as resource-based policies. CodeBuild supports identity-based policies, and resource-based policies for certain read only APIs for the purpose of cross-account resource sharing.

Secure access to S3 buckets

We strongly recommend that you include the following permissions in your IAM role to verify the S3 bucket associated with your CodeBuild project is owned by you or someone you trust. These permissions are not included in AWS managed policies and roles. You must add them yourself.

- `s3:GetBucketAcl`
- `s3:GetBucketLocation`

If the owner of an S3 bucket used by your project changes, you must verify you still own the bucket and update permissions in your IAM role if not. For more information, see Allow users to interact with CodeBuild and Allow CodeBuild to interact with other AWS services.

Specifying policy elements: Actions, effects, and principals

For each AWS CodeBuild resource, the service defines a set of API operations. To grant permissions for these API operations, CodeBuild defines a set of actions that you can specify in a policy. Some API operations can require permissions for more than one action in order to perform the API operation. For more information, see AWS CodeBuild resources and operations and AWS CodeBuild permissions reference.

The following are the basic policy elements:

- **Resource** – You use an Amazon Resource Name (ARN) to identify the resource that the policy applies to.

- **Action** – You use action keywords to identify resource operations you want to allow or deny. For example, the `codebuild:CreateProject` permission gives the user permissions to perform the `CreateProject` operation.
- **Effect** – You specify the effect, either allow or deny, when the user requests the action. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure a user cannot access a resource, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions.

To learn more about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

For a table showing all of the CodeBuild API actions and the resources they apply to, see the AWS CodeBuild permissions reference.

Using identity-based policies for AWS CodeBuild

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) and thereby grant permissions to perform operations on AWS CodeBuild resources.

> **Important**
>
> We recommend that you first review the introductory topics that explain the basic concepts and options available to manage access to your CodeBuild resources. For more information, see Overview of managing access permissions to your AWS CodeBuild resources.

Topics

- CodeBuild updates to AWS managed policies
- Customer-managed policy examples

The following shows an example of a permissions policy that allows a user to get information about build projects only in the `us-east-2` region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

Permissions required to use the AWS CodeBuild console

A user who uses the AWS CodeBuild console must have a minimum set of permissions that allows the user to describe other AWS resources for the AWS account. You must have permissions from the following services:

- AWS CodeBuild
- Amazon CloudWatch
- CodeCommit (if you are storing your source code in an AWS CodeCommit repository)
- Amazon Elastic Container Registry (Amazon ECR) (if you are using a build environment that relies on a Docker image in an Amazon ECR repository)

> **Note**
>
> As of July 26, 2022, the default IAM policy has been updated. For more information, see Permissions required for AWS CodeBuild to connect to Amazon Elastic Container Registry.

- Amazon Elastic Container Service (Amazon ECS) (if you are using a build environment that relies on a Docker image in an Amazon ECR repository)

- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service (Amazon S3)

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended.

Permissions required for AWS CodeBuild to connect to Amazon Elastic Container Registry

As of July 26, 2022, AWS CodeBuild has updated its default IAM policy for Amazon ECR permission. The following permissions have been removed from the default policy:

```
"ecr:PutImage",
"ecr:InitiateLayerUpload",
"ecr:UploadLayerPart",
"ecr:CompleteLayerUpload"
```

For CodeBuild projects that were created before July 26, 2022, we recommend you update your policy with the following Amazon ECR policy:

```
"Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
]
```

For more information on updating your policy, see Allow users to interact with CodeBuild.

Permissions required for the AWS CodeBuild console to connect to source providers

The AWS CodeBuild console uses the following API actions to connect to source providers (for example, GitHub repositories).

- `codebuild:ListConnectedOAuthAccounts`
- `codebuild:ListRepositories`

- `codebuild:PersistOAuthToken`
- `codebuild:ImportSourceCredentials`

You can associate source providers (such as GitHub repositories) with your build projects using the AWS CodeBuild console. To do this, you must first add the preceding API actions to IAM access policies associated with the user you use to access the AWS CodeBuild console.

The `ListConnectedOAuthAccounts`, `ListRepositories`, and `PersistOAuthToken` API actions are not intended to be called by your code. Therefore, these API actions are not included in the AWS CLI and AWS SDKs.

AWS managed (predefined) policies for AWS CodeBuild

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. The managed policies for CodeBuild also provide permissions to perform operations in other services, such as IAM, AWS CodeCommit,Amazon EC2, Amazon ECR, Amazon SNS, and Amazon CloudWatch Events, as required for the responsibilities for the users who have been granted the policy in question. For example, the `AWSCodeBuildAdminAccess` policy is an administrative-level user policy that allows users with this policy to create and manage CloudWatch Events rules for project builds and Amazon SNS topics for notifications about project-related events (topics whose names are prefixed with `arn:aws:codebuild:`), as well as administer projects and report groups in CodeBuild. For more information, see AWS Managed Policies in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to AWS CodeBuild.

AWSCodeBuildAdminAccess

Provides full access to CodeBuild including permissions to administrate CodeBuild build projects.

AWSCodeBuildDeveloperAccess

Provides access to CodeBuild but does not allow build project administration.

AWSCodeBuildReadOnlyAccess

Provides read-only access to CodeBuild.

To access build output artifacts that CodeBuild creates, you must also attach the AWS managed policy named `AmazonS3ReadOnlyAccess`.

To create and manage CodeBuild service roles, you must also attach the AWS managed policy named `IAMFullAccess`.

You can also create your own custom IAM policies to allow permissions for CodeBuild actions and resources. You can attach these custom policies to the users or groups that require those permissions.

Topics

AWSCodeBuildAdminAccess

The `AWSCodeBuildAdminAccess` policy provides full access to CodeBuild, including permissions to administer CodeBuild build projects. Apply this policy only to administrative-level users to grant them full control over CodeBuild projects, report groups, and related resources in your AWS account, including the ability to delete projects and report groups.

The `AWSCodeBuildAdminAccess` policy contains the following policy statement:

```
{
  "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": {
        "ArnLike": {
          "codestar-notifications:NotificationsForResource":
 "arn:aws:codebuild:*:*:project/*"
        }
      }
    },
    {
      "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

The `AWSCodeBuildDeveloperAccess` policy allows access to all of the functionality of CodeBuild and project and report group-related resources. This policy does not allow users to

delete CodeBuild projects or report groups, or related resources in other AWS services, such as CloudWatch Events. We recommend that you apply this policy to most users.

The `AWSCodeBuildDeveloperAccess` policy contains the following policy statement:

```
{
  "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": {
        "ArnLike": {
          "codestar-notifications:NotificationsForResource":
"arn:aws:codebuild:*:*:project/*"
        }
      }
    },
    {
      "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

The `AWSCodeBuildReadOnlyAccess` policy grants read-only access to CodeBuild and related resources in other AWS services. Apply this policy to users who can view and run builds, view projects, and view report groups, but cannot make any changes to them.

The `AWSCodeBuildReadOnlyAccess` policy contains the following policy statement:

```
{
  "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": {
        "ArnLike": {
          "codestar-notifications:NotificationsForResource":
"arn:aws:codebuild:*:*:project/*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
```

```
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets"
      ],
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

CodeBuild managed policies and notifications

CodeBuild supports notifications, which can notify users of important changes to build projects. Managed policies for CodeBuild include policy statements for notification functionality. For more information, see What are notifications?.

Permissions related to notifications in read-only managed policies

The `AWSCodeBuildReadOnlyAccess` managed policy includes the following statements to allow read-only access to notifications. Users with this managed policy applied can view notifications for resources, but cannot create, manage, or subscribe to them.

```
   {
        "Sid": "CodeStarNotificationsPowerUserAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:DescribeNotificationRule"
        ],
        "Resource": "*",
        "Condition" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codebuild:*:*:project/*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListEventTypes",
            "codestar-notifications:ListTargets"
        ],
        "Resource": "*"
```

```
    }
```

Permissions related to notifications in other managed policies

The `AWSCodeBuildDeveloperAccess` managed policy includes the following statements to allow users to create, edit, and subscribe to notifications. Users cannot delete notification rules or manage tags for resources.

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codebuild:*:*:project/*"}
        }
    },
    {
        "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": "*"
    }
```

For more information about IAM and notifications, see Identity and Access Management for AWS CodeStar Notifications.

CodeBuild updates to AWS managed policies

View details about updates to AWS managed policies for CodeBuild since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on AWS CodeBuild User Guide document history .

Change	Description	Date
`AWSCodeBuildAdminAccess` , `AWSCodeBuildDeveloperAccess` , and `AWSCodeBuildReadOnlyAccess` – Update to existing policies	CodeBuild updated a resource to these policies. The `AWSCodeBuildAdminAccess` , `AWSCodeBuildDeveloperAccess` , and `AWSCodeBuildReadOnlyAccess` policies have been changed to update an existing resource. The original resource `arn:aws:codebuild:*` has been updated to `arn:aws:codebuild:*:*:project/*` .	November 15, 2024
`AWSCodeBuildAdminAccess` , `AWSCodeBu`	CodeBuild added a resource to these policies to support	April 18, 2024

Change	Description	Date
ildDeveloperAccess , and AWSCodeBuildReadOnlyAccess – Update to existing policies	the AWS CodeConnections rebranding. The AWSCodeBuildAdminAccess , AWSCodeBuildDeveloperAccess , and AWSCodeBuildReadOnlyAccess policies have been changed to add a resource, arn:aws:codeconnections:*:*:connection/* .	
AWSCodeBuildAdminAccess and AWSCodeBuildDeveloperAccess – Update to existing policies	CodeBuild added a permission to these policies to support an additional notification type using Amazon Q Developer in chat applications. The AWSCodeBuildAdminAccess and AWSCodeBuildDeveloperAccess policies have been changed to add a permission, chatbot:ListMicrosoftTeamsChannelConfigurations .	May 16, 2023
CodeBuild started tracking changes	CodeBuild started tracking changes for its AWS managed policies.	May 16, 2021

Customer-managed policy examples

In this section, you can find example user policies that grant permissions for AWS CodeBuild actions. These policies work when you are using the CodeBuild API, AWS SDKs, or AWS CLI. When you are using the console, you must grant additional, console-specific permissions. For information, see Permissions required to use the AWS CodeBuild console.

You can use the following sample IAM policies to limit CodeBuild access for your users and roles.

Topics

- Allow a user to start running builds
- Allow a user to attempt to stop builds
- Allow a user to attempt to delete builds
- Allow a user to get information about Docker images that are managed by CodeBuild
- Allow a user to add a permission policy for a fleet service role
- Allow CodeBuild access to AWS services required to create a VPC network interface
- Use a deny statement to prevent AWS CodeBuild from disconnecting from source providers

Allow a user to get information about build projects

The following example policy statement allows a user to get information about build projects in the `us-east-2` Region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

Allow a user to get information about fleets

The following example policy statement allows a user to get information about fleets in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetFleets",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:fleet/*"
    }
  ]
```

```
}
```

Allow a user to get information about report groups

The following example policy statement allows a user to get information about report groups in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReportGroups",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

Allow a user to get information about reports

The following example policy statement allows a user to get information about reports in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReports",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

Allow a user to create build projects

The following example policy statement allows a user to create build projects with any name but only in the `us-east-2` Region for account `123456789012` and only using the specified CodeBuild service role:

```
{
  "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"
    }
  ]
}
```

The following example policy statement allows a user to create build projects with any name but only in the `us-east-2` Region for account `123456789012` and only using the specified CodeBuild service role. It also enforces that the user can only use the specified service role with AWS CodeBuild and not any other AWS services.

```
{
  "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"}
      }
    }
  ]
}}
```

Allow a user to create a fleet

The following example policy statement allows a user to create a fleet in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateFleet",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:fleet/*"
    }
  ]
}
```

Allow a user to create a report group

The following example policy statement allows a user to create a report group in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

Allow a user to delete a fleet

The following example policy statement allows a user to delete a fleet in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteFleet",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:fleet/*"
    }
  ]
}
```

Allow a user to delete a report group

The following example policy statement allows a user to delete a report group in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

Allow a user to delete a report

The following example policy statement allows a user to delete a report in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReport",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

Allow a user to delete build projects

The following example policy statement allows a user to delete build projects in the `us-east-2` Region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "codebuild:DeleteProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

Allow a user to get a list of build project names

The following example policy statement allows a user to get a list of build project names for the same account:

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

Allow a user to change information about build projects

The following example policy statement allows a user to change information about build projects with any name but only in the `us-east-2` Region for account `123456789012` and only using the specified AWS CodeBuild service role:

```
{
  "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"
    }
  ]
```

```
}
```

Allow a user to change a fleet

The following example policy statement allows a user to change a fleet in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateFleet",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:fleet/*"
    }
  ]
}
```

Allow a user to change a report group

The following example policy statement allows a user to change a report group in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

Allow a user to get information about builds

The following example policy statement allows a user to get information about builds in the `us-east-2` Region for account `123456789012` for the build projects named `my-build-project` and `my-other-build-project`:

```
{
  "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"
      ]
    }
  ]
}
```

Allow a user to get a list of build IDs for a build project

The following example policy statement allows a user to get a list of build IDs in the `us-east-2` Region for account `123456789012` for the build projects named `my-build-project` and `my-other-build-project`:

```
{
  "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"
      ]
    }
  ]
}
```

Allow a user to get a list of build IDs

The following example policy statement allows a user to get a list of all build IDs for the same account:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "codebuild:ListBuilds",
      "Resource": "*"
    }
  ]
}
```

Allow a user to get a list of fleets

The following example policy statement allows a user to get a list of fleets in the `us-east-2` Region for account `123456789012`:

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

Allow a user to get a list of report groups

The following example policy statement allows a user to get a list of report groups in the `us-east-2` Region for account `123456789012`:

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

Allow a user to get a list of reports

The following example policy statement allows a user to get a list of reports in the `us-east-2` Region for account `123456789012`:

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

Allow a user to get a list of reports for a report group

The following example policy statement allows a user to get a list of reports for a report group in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListReportsForReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

Allow a user to get a list of test cases for a report

The following example policy statement allows a user to get a list of test cases for a report in the `us-east-2` Region for account `123456789012`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DescribeTestCases",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

Allow a user to start running builds

The following example policy statement allows a user to run builds in the `us-east-2` Region for account `123456789012` for a build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

Allow a user to attempt to stop builds

The following example policy statement allows a user to attempt to stop running builds only in the `us-east-2` region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StopBuild",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

Allow a user to attempt to delete builds

The following example policy statement allows a user to attempt to delete builds only in the `us-east-2` Region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "codebuild:BatchDeleteBuilds",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

Allow a user to get information about Docker images that are managed by CodeBuild

The following example policy statement allows a user to get information about all Docker images that are managed by CodeBuild:

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

Allow a user to add a permission policy for a fleet service role

The following example resource policy statement allows a user to add a VPC permission policy for a fleet service role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildFleetVpcCreateNI",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:region:account-id:subnet/subnet-id-1",
        "arn:aws:ec2:region:account-id:security-group/security-group-id-1",
        "arn:aws:ec2:region:account-id:network-interface/*"
      ]
    },
```

```
    {
      "Sid": "CodeBuildFleetVpcPermission",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeBuildFleetVpcNIPermission",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterfacePermission"
      ],
      "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:Subnet": [
            "arn:aws:ec2:region:account-id:subnet/subnet-id-1"
          ]
        }
      }
    }
  ]
}
```

The following example resource policy statement allows a user to add a custom Amazon Managed Image (AMI) permission policy for a fleet service role:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeImages",
            "Resource": "*"
        }
```

```
    ]
}
```

The following example trust policy statement allows a user to add a permission policy for a fleet service role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildFleetVPCTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}
```

Allow CodeBuild access to AWS services required to create a VPC network interface

The following example policy statement grants AWS CodeBuild permission to create a network interface in a VPC with two subnets:

```
{
  "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"
          ]
        }
      }
    }
  ]
}
```

Use a deny statement to prevent AWS CodeBuild from disconnecting from source providers

The following example policy statement uses a deny statement to prevent AWS CodeBuild from disconnecting from source providers. It uses `codebuild:DeleteOAuthToken`, which is the inverse of `codebuild:PersistOAuthToken` and `codebuild:ImportSourceCredentials`, to connect with source providers. For more information, see Permissions required for the AWS CodeBuild console to connect to source providers.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "codebuild:DeleteOAuthToken",
      "Resource": "*"
    }
  ]
}
```

AWS CodeBuild permissions reference

You can use AWS-wide condition keys in your AWS CodeBuild policies to express conditions. For a list, see Available Keys in the *IAM User Guide*.

You specify the actions in the policy's `Action` field. To specify an action, use the `codebuild:` prefix followed by the API operation name (for example, `codebuild:CreateProject` and `codebuild:StartBuild`). To specify multiple actions in a single statement, separate them with commas (for example, `"Action": [ "codebuild:CreateProject", "codebuild:StartBuild" ]`).

Using Wildcard Characters

You specify an ARN, with or without a wildcard character (*), as the resource value in the policy's `Resource` field. You can use a wildcard to specify multiple actions or resources. For example, `codebuild:*` specifies all CodeBuild actions and `codebuild:Batch*` specifies all CodeBuild actions that begin with the word `Batch`. The following example grants access to all build project with names that begin with `my`:

```
arn:aws:codebuild:us-east-2:123456789012:project/my*
```

CodeBuild API operations and required permissions for actions

BatchDeleteBuilds

Action: `codebuild:BatchDeleteBuilds`

Required to delete builds.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

BatchGetBuilds

Action: `codebuild:BatchGetBuilds`

Required to get information about builds.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

BatchGetProjects

Action: `codebuild:BatchGetProjects`

Required to get information about build projects.

Resource: `arn:aws:codebuild:region-ID:account-ID:project/project-name`

BatchGetReportGroups

Action: `codebuild:BatchGetReportGroups`

Required to get information about report groups.

Resource: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

BatchGetReports

Action: `codebuild:BatchGetReports`

Required to get information about reports.

Resource: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

BatchPutTestCases [1]

Action: `codebuild:BatchPutTestCases`

Required to create or update a test report.

Resource: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

CreateProject

Actions: `codebuild:CreateProject`, `iam:PassRole`

Required to create build projects.

Resources:

- `arn:aws:codebuild:region-ID:account-ID:project/project-name`
- `arn:aws:iam::account-ID:role/role-name`

CreateReport [1]

Action: `codebuild:CreateReport`

Required to create a test report.

Resource: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateReportGroup

Action: codebuild:CreateReportGroup

Required to create a report group.

Resource: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateWebhook

Action: codebuild:CreateWebhook

Required to create a webhook.

Resource: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DeleteProject

Action: codebuild:DeleteProject

Required to delete a CodeBuild project.

Resource: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DeleteReport

Action: codebuild:DeleteReport

Required to delete a report.

Resource: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

DeleteReportGroup

Action: codebuild:DeleteReportGroup

Required to delete a report group.

Resource: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

DeleteSourceCredentials

Action: `codebuild:DeleteSourceCredentials`

Required to delete a set of `SourceCredentialsInfo` objects that contain information about credentials for a GitHub, GitHub Enterprise Server, or Bitbucket repository.

Resource: *

DeleteWebhook

Action: `codebuild:DeleteWebhook`

Required to create a webhook.

Resource: `arn:aws:codebuild:region-ID:account-ID:project/project-name`

DescribeTestCases

Action: `codebuild:DescribeTestCases`

Required to return a paginated list of test cases.

Resource: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

ImportSourceCredentials

Action: `codebuild:ImportSourceCredentials`

Required to import a set of `SourceCredentialsInfo` objects that contain information about credentials for a GitHub, GitHub Enterprise Server, or Bitbucket repository.

Resource: *

InvalidateProjectCache

Action: `codebuild:InvalidateProjectCache`

Required to reset the cache for a project.

Resource: `arn:aws:codebuild:region-ID:account-ID:project/project-name`

ListBuildBatches

Action: `codebuild:ListBuildBatches`

Required to get a list of build batch IDs.

Resource: *

ListBuildBatchesForProject

Action: `codebuild:ListBuildBatchesForProject`

Required to get a list of build batch IDs for a specific project.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

ListBuilds

Action: `codebuild:ListBuilds`

Required to get a list of build IDs.

Resource: *

ListBuildsForProject

Action: `codebuild:ListBuildsForProject`

Required to get a list of build IDs for a build project.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

ListCuratedEnvironmentImages

Action: `codebuild:ListCuratedEnvironmentImages`

Required to get information about all Docker images that are managed by AWS CodeBuild.

Resource: * (required, but does not refer to an addressable AWS resource)

ListProjects

Action: `codebuild:ListProjects`

Required to get a list of build project names.

Resource: *

ListReportGroups

Action: `codebuild:ListReportGroups`

Required to get a list of report groups.

Resource: *

ListReports

Action: `codebuild:ListReports`

Required to get a list of reports.

Resource: *

ListReportsForReportGroup

Action: `codebuild:ListReportsForReportGroup`

Required to get a list of reports for a report group.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

RetryBuild

Action: `codebuild:RetryBuild`

Required to retry builds.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

StartBuild

Action: `codebuild:StartBuild`

Required to start running builds.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

StopBuild

Action: `codebuild:StopBuild`

Required to attempt to stop running builds.

Resource: `arn:aws:codebuild:region-ID:account-ID:project/project-name`

UpdateProject

Actions: `codebuild:UpdateProject`, `iam:PassRole`

Required to change information about builds.

Resources:

- `arn:aws:codebuild:region-ID:account-ID:project/project-name`
- `arn:aws:iam::account-ID:role/role-name`

UpdateProjectVisibility

Actions: `codebuild:UpdateProjectVisibility`, `iam:PassRole`

Required to change the public visibility of a project's builds.

Resources:

- `arn:aws:codebuild:region-ID:account-ID:project/project-name`
- `arn:aws:iam::account-ID:role/role-name`

UpdateReport [1]

Action: `codebuild:UpdateReport`

Required to create or update a test report.

Resource: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

UpdateReportGroup

Action: `codebuild:UpdateReportGroup`

Required to update a report group.

Resource: `arn:aws:codebuild:region-ID:account-ID:report-group/report-group-name`

UpdateWebhook

Action: `codebuild:UpdateWebhook`

Required to update a webhook.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

[1] Used for permission only. There is no API for this action.

Using tags to control access to AWS CodeBuild resources

Conditions in IAM policy statements are part of the syntax that you can use to specify permissions to CodeBuild project-based actions. You can create a policy that allows or denies actions on projects based on the tags associated with those projects, and then apply those policies to the IAM groups you configure for managing users. For information about applying tags to a project using the console or AWS CLI, see Create a build project in AWS CodeBuild. For information about applying tags using the CodeBuild SDK, see CreateProject and Tags in the *CodeBuild API Reference*. For information about using tags to control access to AWS resources, see Controlling Access to AWS Resources Using Resource Tags in the *IAM User Guide*.

> **Important**
>
> When using the reserved capacity feature, data cached on fleet instances, including source files, Docker layers, and cached directories specified in the buildspec, can be accessible to other projects within the same account. This is by design and allows projects within the same account to share fleet instances.

Example Example 1: Limit CodeBuild project actions based on resource tags

The following example denies all `BatchGetProjects` actions on projects tagged with the key `Environment` with the key value of `Production`. A user's administrator must attach this IAM policy in addition to the managed user policy to unauthorized users. The `aws:ResourceTag` condition key is used to control access to resources based on their tags.

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:BatchGetProjects"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:ResourceTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Example Example 2: Limit CodeBuild project actions based on request tags

The following policy denies users permission to the `CreateProject` action if the request contains a tag with the key `Environment` and the key value `Production`. In addition, the policy prevents these unauthorized users from modifying projects by using the `aws:TagKeys` condition key to not allow `UpdateProject` if the request contains a tag with the key `Environment`. An administrator must attach this IAM policy in addition to the managed user policy to users who are not authorized to perform these actions. The `aws:RequestTag` condition key is used to control which tags can be passed in an IAM request

```
{
  "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 Example 3: Deny or allow actions on report groups based on resource tags

You can create a policy that allows or denies actions on CodeBuild resources (projects and report groups) based on the AWS tags associated with those resources, and then apply those policies to the IAM groups you configure for managing users. For example, you can create a policy that denies all CodeBuild actions on any report group with the AWS tag key `Status` and the key value of `Secret`, and then apply that policy to the IAM group you created for general developers (*`Developers`*). You then need to make sure that the developers working on those tagged report groups are not members of that general *`Developers`* group, but belong instead to a different IAM group that does not have the restrictive policy applied (`SecretDevelopers`).

The following example denies all CodeBuild actions on report groups tagged with the key `Status` and the key value of `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 Example 4: Limit CodeBuild actions to AWSCodeBuildDeveloperAccess based on resource tags

You can create policies that allow CodeBuild actions on all report groups and projects that are not tagged with specific tags. For example, the following policy allows the equivalent of AWSCodeBuildDeveloperAccess permissions for all report groups and projects except those tagged with the specified tags:

```
{
   "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"
            }
        }
    }
  ]
}
```

Viewing resources in the console

The AWS CodeBuild console requires the `ListRepositories` permission to display a list of repositories for your AWS account in the AWS Region where you are signed in. The console also includes a **Go to resource** function to quickly perform a case insensitive search for resources. This search is performed in your AWS account in the AWS Region where you are signed in. The following resources are displayed across the following services:

- AWS CodeBuild: Build projects
- AWS CodeCommit: Repositories
- AWS CodeDeploy: Applications
- AWS CodePipeline: Pipelines

To perform this search across resources in all services, you must have the following permissions:

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

Results are not returned for a service's resources if you do not have permissions for that service. Even if you have permissions for viewing resources, some resources are not returned if there is an explicit `Deny` to view those resources.

Compliance validation for AWS CodeBuild

Third-party auditors assess the security and compliance of AWS CodeBuild as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS services in scope by compliance program. For general information, see AWS compliance programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading reports in AWS Artifact.

Your compliance responsibility when using CodeBuild is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. If your use of CodeBuild is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and compliance quick start guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS compliance resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – Monitor your usage of AWS CodeBuild as it relates to security best practices by using AWS Security Hub. Security Hub uses security controls to evaluate resource configurations and security standards to help you comply with various compliance frameworks. For more information about using Security Hub to evaluate CodeBuild resources, see AWS CodeBuild controls in the AWS Security Hub User Guide.

Resilience in AWS CodeBuild

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS global infrastructure.

Infrastructure security in AWS CodeBuild

As a managed service, AWS CodeBuild is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access CodeBuild through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Access your source provider in CodeBuild

For GitHub or GitHub Enterprise Server, you use a personal access token, a Secrets Manager secret, a connection, or an OAuth app to access the source provider. For Bitbucket, you use either an access token, an app password, a Secrets Manager secret, a connection, or an OAuth app to access the source provider.

Topics

Create and store a token in a Secrets Manager secret

If you choose to use to store your access token using Secrets Manager, you can use either an existing secret connection or create a new secret. To create a new secret, do the following:

AWS Management Console

To create a Secrets Manager secret in the AWS Management Console

1. For **Source provider**, choose **Bitbucket**, **GitHub**, or **GitHub Enterprise**.
2. For **Credential**, do one of the following:
   - Choose **Default source credential** to use your account's default source credential to apply to all projects.
     a. If you aren't connected to your source provider, choose **Manage default source credential**.
     b. For **Credential type**, choose a credential type other than **CodeConnections**.
     c. For **Service**, choose **Secrets Manager** and for **Secrets** choose **New secret**.
     d. In **Secret name**, enter the name of your secret.
     e. In **Secret description - optional**, enter a description for your secret.
     f. Depending on the source provider you chose, enter your token or username and app password and choose **Save**.
   - Choose **Custom source credential** to use a custom source credential to override your account's default settings.
     a. For **Credential type**, choose a credential type other than **CodeConnections**.
     b. In **Connection**, choose **Create a secret**.
     c. In **Secret name**, enter the name of your secret.
     d. In **Secret description - optional**, enter a description for your secret.
     e. Depending on the source provider you chose, enter your token or username and app password, and choose **Create**.

AWS CLI

To create a Secrets Manager secret in the AWS CLI

- Open a terminal (Linux, macOS, or Unix) or command prompt (Windows). Use the AWS CLI to run the Secrets Manager **create-secret** command.

```
aws secretsmanager create-secret --region <aws-region> \
            --name '<secret-name>' \
```

```
            --description '<secret-description>' \
            --secret-string '{
                "ServerType":"<server-type>",
                "AuthType":"<auth-type>",
                "Token":"<token>"
                }' \
            --tags Key=codebuild:source,Value='' \
                Key=codebuild:source:type,Value=<type> \
                Key=codebuild:source:provider,Value=<provider>
```

The Secrets Manager secrets that CodeBuild accept must be in the same account and AWS Region as the CodeBuild project and must be in the following JSON format:

```
{
            "ServerType": ServerType,
            "AuthType: AuthType,
            "Token": string,
            "Username": string // Optional and is only used for Bitbucket app
 password
        }
```

Field	Valid values	Description
ServerType	GITHUB GITHUB_ENTERPRISE BITBUCKET	The third party source provider for your Secrets Manager secret.
AuthType	PERSONAL_ACCESS_TOKEN BASIC_AUTH	The type of access token used by the credentials. For GitHub, only PERSONAL_ACCESS_TOKEN is valid. BASIC_AUTH is only valid for Bitbucket app password.

Field	Valid values	Description
Token	*string*	For GitHub or GitHub Enterprise, this is the personal access token. For Bitbucket, this is either the access token or the Bitbucket app password.
Username	*string*	The Bitbucket username when the AuthType is BASIC_AUTH. This parameter is not valid for other types of source providers.

Additionally, CodeBuild uses the following resource tags on the secret to ensure the secrets are easily selectable when creating or editing projects.

Tag key	Tag value	Description
codebuild:source:provider	github github_enterprise bitbucket	Tells CodeBuild which provider this secret is intended for.
codebuild:source:type	personal_access_token basic_auth	Tells CodeBuild the type of access token in this secret.

GitHub and GitHub Enterprise Server access in CodeBuild

For GitHub, you can use a personal access token, an OAuth app, a Secrets Manager secret, or a GitHub App connection to access the source provider. For GitHub Enterprise Server, you can use a personal access token, a Secrets Manager secret, or a GitHub App connection to access the source provider.

Topics

GitHub App connections for GitHub and GitHub Enterprise Server

You can use GitHub App to connect with CodeBuild. GitHub App connections are supported through AWS CodeConnections.

The source provider access enables you to trigger a build by subscribing to GitHub webhook events using CreateWebhook, or to use Tutorial: Configure a CodeBuild-hosted GitHub Actions runner in CodeBuild.

> **Note**
>
> CodeConnections is available in fewer regions than CodeBuild. You can use cross-region connections in CodeBuild. Connections created in opt-in regions, cannot be used in other regions. For more information, see AWS CodeConnections endpoints and quotas.

Topics

Step 1: Create a connection to GitHub App (console)

Use these steps to use the CodeBuild console to add a connection for your project in GitHub.

To create a connection to GitHub

- Follow the instructions in the *Developer Tools User Guide* for Create a connection to GitHub.

> **Note**
>
> Instead of creating or using an existing connection in your account, you can use a connection shared from another AWS account. For more information, see Share connections with AWS accounts.

Step 2: Grant CodeBuild project IAM role access to use the connection

You can grant CodeBuild project IAM role access to use the GitHub tokens vended by your connection.

To grant CodeBuild project IAM role access

1. Create an IAM role for your CodeBuild project by following the instructions to Allow CodeBuild to interact with other AWS services for your CodeBuild project.
2. While following the instructions, add the following IAM policy to your CodeBuild project role to grant access to the connection.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:GetConnectionToken",
        "codeconnections:GetConnection"
      ],
      "Resource": [
        <connection-arn>
      ]
    }
  ]
}
```

Step 3: Configure CodeBuild to use the new connection

You can configure a connection as an account level credential and use it in a project.

AWS Management Console

To configure a connection as an account level credential in the AWS Management Console

1. For **Source provider**, choose **GitHub**.
2. For **Credential**, do one of the following:
   - Choose **Default source credential** to use your account's default source credential to apply to all projects.
     a. If you aren't connected to GitHub, choose **Manage default source credential**.
     b. For **Credential type**, choose **GitHub App**.
     c. In **Connection**, choose to use an existing connection or create a new connection.
   - Choose **Custom source credential** to use a custom source credential to override your account's default settings.
     a. For **Credential type**, choose **GitHub App**.
     b. In **Connection**, choose to use an existing connection or create a new connection.

AWS CLI

To configure a connection as an account level credential in the AWS CLI

- Open a terminal (Linux, macOS, or Unix) or command prompt (Windows). Use the AWS CLI to run the **import-source-credentials** command, specifying the `--auth-type`, `--server-type`, and `--token` for your connection.

  Use the following command:

```
aws codebuild import-source-credentials --auth-type CODECONNECTIONS --server-type GITHUB --token <connection-arn>
```

You can also set up multiple tokens for your CodeBuild projects. For more information, see Configure multiple tokens as source level credentials.

Troubleshooting problems with the GitHub App

The following information can help you troubleshoot common issues with the GitHub App.

Topics

Install the AWS Connector for GitHub app in an undesired region

Issue: You installed the AWS Connector for GitHub from the GitHub Marketplace, but the connection was created in an undesired region. If you attempt to reconfigure the app on the GitHub website, it won't work because the app is already installed on your GitHub account.

Possible cause: The app is already installed in your GitHub account, so you can only reconfigure the app permissions.

Recommended solution: You can create a new connection with the installation ID in the desired region.

1. Open the CodeConnections console at https://console.aws.amazon.com/codesuite/settings/connections and navigate to the desired region using the region selector in the AWS console navigation bar.
2. Follow the instructions in the *Developer Tools User Guide* for Create a connection to GitHub.

> **Note**
>
> Since you've already installed the AWS Connector for GitHub app, you can choose it instead of installing a new app.

The GitHub App connection doesn't have access to repositories

Issue: An AWS service using the connection, such as CodeBuild or CodePipeline, reports that it doesn't have access to the repository or the repository doesn't exist. Some possible error messages include:

- `Authentication required for primary source.`
- `Unable to create webhook at this time. Please try again later.`
- `Failed to create webhook. GitHub API limit reached. Please try again later.`

Possible cause: *You might have been using the GitHub app and haven't granted the webhook permission scope.*

Recommended solution: To grant the required permission scope, follow the instructions in Navigating to the GitHub App you want to review or modify to configure the installed app. Under the permissions section, you'll see the app doesn't have webhooks permission, and there is an option for you to review the newly requested permissions. Review and accept the new permissions. For more infomation, see Approving updated permissions for a GitHub App.

Possible cause: *The connection was working as expected, but suddenly doesn't have access to the repositories.*

Possible solution: Start by reviewing your authorizations and your installations, then verify the GitHub App is authorized and installed. If the GitHub App installation is suspended, then you need to unsuspended it. If the GitHub App is not authorized for a UAT (User Access Token) connection, or not installed for an IAT (Installation Access Token) connection, the existing connection is not usable any more, and you will need to create a new connection. Note that reinstalling the GitHub App will not revive the previous connection that was associated to the old installation.

Possible solution: If the connection is a UAT connection, make sure the connection is not concurrently being used, such as a being used in multiple CodeBuild concurrent runs of build. This is because GitHub immediately invalidates a previously issued UAT if an expiring token is refreshed by the connection. If you need to use UAT connection for multiple concurrent CodeBuild builds, you can create multiple connections and use each connection independently.

Possible solution: If the UAT connection hasn't been used in the past 6 months, the connection will be invalidated by GitHub. To fix this, create a new connection.

Possible cause: *You might have been using a UAT connection without installing the app.*

Recommended solution: Though creating a UAT connection doesn't require associating the connection with a GitHub App installation, an installation is required for the repository to be accessible. Follow the instructions to review installations to make sure the GitHub App is installed. If it is not installed, navigate to the GitHub App's page to install the app. For more information about UAT's access, see About user access tokens.

The AWS service's IAM role is missing necessary IAM permissions.

Issue: You see any of the following error messages:

- `Access denied to connection <connection-arn>`
- `Failed to get access token from <connection-arn>`

Recommended solution: Typically you use a connection with an AWS service, such as CodePipeline or CodeBuild. When you give the AWS service an IAM role, the AWS service can use the role's permission to act on your behalf. Make sure the IAM role has necessary permission. For more information about the necessary IAM permission, see Grant CodeBuild project IAM role access to use the connection and Identity and access management for AWS CodeStar Notifications and CodeConnections in the *Developer Tools console User Guide*.

GitHub and GitHub Enterprise Server access token

Access token prerequisites

Before you begin, you must add the proper permission scopes to your GitHub access token.

For GitHub, your personal access token must have the following scopes.

- **repo**: Grants full control of private repositories.
- **repo:status**: Grants read/write access to public and private repository commit statuses.
- **admin:repo_hook**: Grants full control of repository hooks. This scope is not required if your token has the `repo` scope.
- **admin:org_hook**: Grants full control of organization hooks. This scope is only required if you are using the organization webhook feature.

For more information, see Understanding scopes for OAuth apps on the GitHub website.

If you are using fine-grained personal access tokens, depending on your use case, your personal access token might need the following permissions:

- **Contents: Read-only**: Grants access to private repositories. This permission is required if you are using private repositories as source.
- **Commit statuses: Read and write**: Grants permission to create commit statuses. This permission is required if your project has webhook set up, or you have report build status feature enabled.
- **Webhooks: Read and write**: Grants permission to manage webhooks. This permission is required if your project has webhook set up.

- **Pull requests: Read-only**: Grants permission to access pull requests. This permission is required if your webhook has a `FILE_PATH` filter on pull request events.
- **Administration: Read and write**: This permission is required if you are using the self-hosted GitHub Actions runner feature with CodeBuild. For more details, see Create a registration token for a repository and Tutorial: Configure a CodeBuild-hosted GitHub Actions runner.

> **Note**
>
> If you want to access organization repositories, make sure you specify the organization as the resource owner of the access token.

For more information, see Permissions required for fine-grained personal access tokens on the GitHub website.

Connect GitHub with an access token (console)

To use the console to connect your project to GitHub using an access token, do the following when you create a project. For information, see Create a build project (console).

1. For **Source provider**, choose **GitHub**.
2. For **Credential**, do one of the following:
   - Choose to use account credentials to apply your account's default source credential to all projects.
     a. If you aren't connected to GitHub, choose **Manage account credential**.
     b. For **Credential type**, choose **Personal access token**.
   - If you chose to use account level credentials for **Service**, choose which service you'd like to use to store your token and do the following:
     a. If you choose to use **Secrets Manager**, you can choose to use an existing secret connection or create a new secret, and then choose **Save**. For more information how to create a new secret, see Create and store a token in a Secrets Manager secret.
     b. If you choose to use **CodeBuild**, enter your GitHub personal access token, and then choose **Save**.
   - Select **Use override credentials for this project only** to use a custom source credential to override your account's credential settings.

a. From the populated credential list, choose one of the options under **Personal access token**.

b. You can also create new personal access token by selecting **create a new personal access token connection** in the description.

Connect GitHub with an access token (CLI)

Follow these steps to use the AWS CLI to connect your project to GitHub using an access token. For information about using the AWS CLI with AWS CodeBuild, see the Command line reference.

1. Run the **import-source-credentials** command:

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

JSON-formatted data appears in the output. Copy the data to a file (for example, *import-source-credentials.json*) in a location on the local computer or instance where the AWS CLI is installed. Modify the copied data as follows, and save your results.

```
{
    "serverType": "server-type",
    "authType": "auth-type",
    "shouldOverwrite": "should-overwrite",
    "token": "token",
    "username": "username"
    }
```

Replace the following:

- *server-type*: Required value. The source provider used for this credential. Valid values are GITHUB, BITBUCKET, GITHUB_ENTERPRISE, GITLAB, and GITLAB_SELF_MANAGED.
- *auth-type*: Required value. The type of authentication used to connect to a repository. Valid values are OAUTH, BASIC_AUTH, PERSONAL_ACCESS_TOKEN, CODECONNECTIONS, and SECRETS_MANAGER. For GitHub, only PERSONAL_ACCESS_TOKEN is allowed. BASIC_AUTH is only allowed with Bitbucket app password.
- *should-overwrite*: Optional value. Set to `false` to prevent overwriting the repository source credentials. Set to `true` to overwrite the repository source credentials. The default value is `true`.

- *token*: Required value. For GitHub or GitHub Enterprise Server, this is the personal access token. For Bitbucket, this is the personal access token or app password. For the auth-type CODECONNECTIONS, this is the connection ARN. For the auth-type SECRETS_MANAGER, this is the secret ARN.
- *username*: Optional value. This parameter is ignored for GitHub and GitHub Enterprise Server source providers.

2. To connect your account with an access token, switch to the directory that contains the `import-source-credentials.json` file you saved in step 1 and run the **import-source-credentials** command again.

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON-formatted data appears in the output with an Amazon Resource Name (ARN).

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
    }
```

> **Note**
>
> If you run the **import-source-credentials** command with the same server type and auth type a second time, the stored access token is updated.

After your account is connected with an access token, you can use `create-project` to create your CodeBuild project. For more information, see Create a build project (AWS CLI).

3. To view the connected access tokens, run the **list-source-credentials** command.

```
aws codebuild list-source-credentials
```

A JSON-formatted `sourceCredentialsInfos` object appears in the output:

```
{
        "sourceCredentialsInfos": [
            {
                "authType": "auth-type",
```

```
                "serverType": "server-type",
                "arn": "arn"
            }
        ]
    }
```

The `sourceCredentialsObject` contains a list of connected source credentials information:

- The `authType` is the type of authentication used by credentials. This can be `OAUTH`, `BASIC_AUTH`, `PERSONAL_ACCESS_TOKEN`, `CODECONNECTIONS`, or `SECRETS_MANAGER`.
- The `serverType` is the type of source provider. This can be `GITHUB`, `GITHUB_ENTERPRISE`, `BITBUCKET`, `GITLAB`, or `GITLAB_SELF_MANAGED`.
- The `arn` is the ARN of the token.

4. To disconnect from a source provider and remove its access tokens, run the **delete-source-credentials** command with its ARN.

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON-formatted data is returned with an ARN of the deleted credentials.

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
    }
```

GitHub OAuth app

Connect GitHub using OAuth (console)

To use the console to connect your project to GitHub using an OAuth app, do the following when you create a project. For information, see Create a build project (console).

1. For **Source provider**, choose **GitHub**.
2. For **Credential**, do one of the following:
   - Choose to use account credentials to apply your account's default source credential to all projects.
     a. If you aren't connected to GitHub, choose **Manage account credential**.

b. For **Credential type**, choose **OAuth app**.

- If you chose to use account level credentials for **Service**, choose which service you'd like to use to store your token and do the following:

  a. If you choose to use **Secrets Manager**, you can choose to use an existing secret connection or create a new secret, and then choose **Save**. For more information how to create a new secret, see Create and store a token in a Secrets Manager secret.

  b. If you choose to use **CodeBuild** and then choose **Save**.

- Select **Use override credentials for this project only** to use a custom source credential to override your account's credential settings.

  a. From the populated credential list, choose one of the options under **OAuth app**.

  b. You can also create new OAuth app token by selecting **create a new Oauth app token connection** in the description.

To review your authorized OAuth apps, navigate to Applications on GitHub, and verify that an application named `AWS CodeBuild (region)` owned by aws-codesuite is listed.

Bitbucket access in CodeBuild

For Bitbucket, you use either an access token, an app password, an OAuth app, or a Bitbucket connection to access the source provider.

Topics

- Bitbucket App connections
- Bitbucket app password or access token
- Bitbucket OAuth app

Bitbucket App connections

You can use Bitbucket to connect with CodeBuild. Bitbucket App connections are supported through AWS CodeConnections.

> **Note**
>
> CodeConnections is available in less regions than CodeBuild. You can use cross-region connections in CodeBuild. Connections created in opt-in regions, cannot be used in other regions. For more information, see AWS CodeConnections endpoints and quotas.

Topics

Step 1: Create a connection to Bitbucket (console)

Use these steps to use the CodeBuild console to add a connection for your project in Bitbucket.

To create a connection to Bitbucket

- Follow the instructions in the *Developer Tools User Guide* for Create a connection to Bitbucket.

> **Note**
>
> Instead of creating or using an existing connection in your account, you can use a connection shared from another AWS account. For more information, see Share connections with AWS accounts.

Step 2: Grant CodeBuild project IAM role access to use the connection

You can grant CodeBuild project IAM role access to use the Bitbucket tokens vended by your connection.

To grant CodeBuild project IAM role access

1. Create an IAM role for your CodeBuild project by following the instructions to Allow CodeBuild to interact with other AWS services for your CodeBuild project.
2. While following the instructions, add the following IAM policy to your CodeBuild project role to grant access to the connection.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:GetConnectionToken",
        "codeconnections:GetConnection"
      ],
      "Resource": [
        <connection-arn>
      ]
    }
  ]
}
```

Step 3: Configure CodeBuild to use the new connection

You can configure a connection as an account level credential and use it in a project.

AWS Management Console

To configure a connection as an account level credential in the AWS Management Console

1. For **Source provider**, choose **Bitbucket**.
2. For **Credential**, do one of the following:
   - Choose **Default source credential** to use your account's default source credential to apply to all projects.
     a. If you aren't connected to Bitbucket, choose **Manage default source credential**.
     b. For **Credential type**, choose **CodeConnections**.
     c. In **Connection**, choose to use an existing connection or create a new connection.
   - Choose **Custom source credential** to use a custom source credential to override your account's default settings.
     a. For **Credential type**, choose **CodeConnections**.
     b. In **Connection**, choose to use an existing connection or create a new connection.

AWS CLI

To configure a connection as an account level credential in the AWS CLI

- Open a terminal (Linux, macOS, or Unix) or command prompt (Windows). Use the AWS CLI to run the **import-source-credentials** command, specifying the `--auth-type`, `--server-type`, and `--token` for your connection.

  Use the following command:

  ```
  aws codebuild import-source-credentials --auth-type CODECONNECTIONS --server-type BITBUCKET --token <connection-arn>
  ```

For more information on setting up multiple tokens in your CodeBuild project, see Configure multiple tokens as source level credentials.

Bitbucket app password or access token

Prerequisites

Before you begin, you must add the proper permission scopes to your Bitbucket app password or access token.

For Bitbucket, your app password or access token must have the following scopes.

- **repository:read**: Grants read access to all the repositories to which the authorizing user has access.
- **pullrequest:read**: Grants read access to pull requests. If your project has a Bitbucket webhook, then your app password or access token must have this scope.
- **webhook**: Grants access to webhooks. If your project has a webhook operation, then your app password or access token must have this scope.

For more information, see Scopes for Bitbucket Cloud REST API and OAuth on Bitbucket Cloud on the Bitbucket website.

Connect Bitbucket with an app password (console)

To use the console to connect your project to Bitbucket using an app password, do the following when you create a project. For information, see Create a build project (console).

1. For **Source provider**, choose **Bitbucket**.
2. For **Credential**, do one of the following:
   - Choose to use account credentials to apply your account's default source credential to all projects.
     a. If you aren't connected to Bitbucket, choose **Manage account credential**.
     b. For **Credential type**, choose **App password**.
   - If you chose to use account level credentials for **Service**, choose which service you'd like to use to store your token and do the following:
     a. If you choose to use **Secrets Manager**, you can choose to use an existing secret connection or create a new secret, and then choose **Save**. For more information how to create a new secret, see Create and store a token in a Secrets Manager secret.
     b. If you choose to use **CodeBuild**, enter your Bitbucket username and app password, and then choose **Save**.
   - Select **Use override credentials for this project only** to use a custom source credential to override your account's credential settings.
     a. From the populated credential list, choose one of the options under **App password**.
     b. You can also create new App password token by selecting **create a new app password connection** in the description.

Connect Bitbucket with an access token (console)

To use the console to connect your project to Bitbucket using an access token, do the following when you create a project. For information, see Create a build project (console).

1. For **Source provider**, choose **Bitbucket**.
2. For **Credential**, do one of the following:
   - Choose to use account credentials to apply your account's default source credential to all projects.
     a. If you aren't connected to Bitbucket, choose **Manage account credential**.
     b. For **Credential type**, choose **Personal access token**.
   - If you chose to use account level credentials for **Service**, choose which service you'd like to use to store your token and do the following:

a. If you choose to use **Secrets Manager**, you can choose to use an existing secret connection or create a new secret, and then choose **Save**. For more information how to create a new secret, see Create and store a token in a Secrets Manager secret.

b. If you choose to use **CodeBuild**, enter your Bitbucket personal access token, and then choose **Save**.

- Select **Use override credentials for this project only** to use a custom source credential to override your account's credential settings.

  a. From the populated credential list, choose one of the options under **Personal access token**.

  b. You can also create new personal access token by selecting **create a new personal access token connection** in the description.

Connect Bitbucket with an app password or access token(CLI)

Follow these steps to use the AWS CLI to connect your project to Bitbucket using an app password or access token. For information about using the AWS CLI with AWS CodeBuild, see the Command line reference.

1. Run the **import-source-credentials** command:

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

JSON-formatted data appears in the output. Copy the data to a file (for example, *import-source-credentials.json*) in a location on the local computer or instance where the AWS CLI is installed. Modify the copied data as follows, and save your results.

```
{
    "serverType": "BITBUCKET",
    "authType": "auth-type",
    "shouldOverwrite": "should-overwrite",
    "token": "token",
    "username": "username"
    }
```

Replace the following:

- *server-type*: Required value. The source provider used for this credential. Valid values are GITHUB, BITBUCKET, GITHUB_ENTERPRISE, GITLAB, and GITLAB_SELF_MANAGED.
- *auth-type*: Required value. The type of authentication used to connect to a repository. Valid values are OAUTH, BASIC_AUTH, PERSONAL_ACCESS_TOKEN, CODECONNECTIONS, and SECRETS_MANAGER. For GitHub, only PERSONAL_ACCESS_TOKEN is allowed. BASIC_AUTH is only allowed with Bitbucket app password.
- *should-overwrite*: Optional value. Set to `false` to prevent overwriting the repository source credentials. Set to `true` to overwrite the repository source credentials. The default value is `true`.
- *token*: Required value. For GitHub or GitHub Enterprise Server, this is the personal access token. For Bitbucket, this is the personal access token or app password. For the auth-type CODECONNECTIONS, this is the connection ARN. For the auth-type SECRETS_MANAGER, this is the secret ARN.
- *username*: Optional value. This parameter is ignored for GitHub and GitHub Enterprise Server source providers.

2. To connect your account with an app password or an access token, switch to the directory that contains the `import-source-credentials.json` file you saved in step 1 and run the **import-source-credentials** command again.

```
aws codebuild import-source-credentials --cli-input-json file://import-source-
credentials.json
```

JSON-formatted data appears in the output with an Amazon Resource Name (ARN).

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
    }
```

> **Note**
>
> If you run the **import-source-credentials** command with the same server type and auth type a second time, the stored access token is updated.

After your account is connected with an app password, you can use `create-project` to create your CodeBuild project. For more information, see Create a build project (AWS CLI).

3. To view the connected app passwords or access tokens, run the **list-source-credentials** command.

```
aws codebuild list-source-credentials
```

A JSON-formatted `sourceCredentialsInfos` object appears in the output:

```
{
        "sourceCredentialsInfos": [
            {
                "authType": "auth-type",
                "serverType": "BITBUCKET",
                "arn": "arn"
            }
        ]
    }
```

The `sourceCredentialsObject` contains a list of connected source credentials information:

- The `authType` is the type of authentication used by credentials. This can be `OAUTH`, `BASIC_AUTH`, `PERSONAL_ACCESS_TOKEN`, `CODECONNECTIONS`, or `SECRETS_MANAGER`.
- The `serverType` is the type of source provider. This can be `GITHUB`, `GITHUB_ENTERPRISE`, `BITBUCKET`, `GITLAB`, or `GITLAB_SELF_MANAGED`.
- The `arn` is the ARN of the token.

4. To disconnect from a source provider and remove its app password or access tokens, run the **delete-source-credentials** command with its ARN.

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON-formatted data is returned with an ARN of the deleted credentials.

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
```

```
    }
```

Bitbucket OAuth app

Connect Bitbucket using OAuth (console)

To use the console to connect your project to Bitbucket using an OAuth app, do the following when you create a project. For information, see Create a build project (console).

1. For **Source provider**, choose **Bitbucket**.
2. For **Credential**, do one of the following:
   - Choose to use account credentials to apply your account's default source credential to all projects.
     a. If you aren't connected to Bitbucket, choose **Manage account credential**.
     b. For **Credential type**, choose **OAuth app**.
   - If you chose to use account level credentials for **Service**, choose which service you'd like to use to store your token and do the following:
     a. If you choose to use **Secrets Manager**, you can choose to use an existing secret connection or create a new secret, and then choose **Save**. For more information how to create a new secret, see Create and store a token in a Secrets Manager secret.
     b. If you choose to use **CodeBuild** and then choose **Save**.
   - Select **Use override credentials for this project only** to use a custom source credential to override your account's credential settings.
     a. From the populated credential list, choose one of the options under **OAuth app**.
     b. You can also create new OAuth app token by selecting **create a new Oauth app token connection** in the description.

To review your authorized OAuth apps, navigate to Application authorizations on Bitbucket, and verify that an application named `AWS CodeBuild (region)` is listed.

GitLab access in CodeBuild

For GitLab, you use a GitLab connection to access the source provider.

Topics

Connect CodeBuild to GitLab

Connections allow you to authorize and establish configurations that associate your third-party provider with your AWS resources using AWS CodeConnections. To associate your third-party repository as a source for your build project, you use a connection.

To add a GitLab or GitLab Self Managed source provider in CodeBuild, you can choose either to:

- Use the CodeBuild console **Create build project** wizard or **Edit Source** page to choose the **GitLab** or **GitLab Self Managed** provider option. See Create a connection to GitLab (console) to add the source provider. The console helps you create a connections resource.
- Use the CLI to create your connections resources, see Create a connection to GitLab (CLI) to create a connections resource with the CLI.

> **Note**
>
> You can also create a connection using the Developer Tools console under **Settings**. See Create a Connection.

> **Note**
>
> By authorizing this connection installation in GitLab, you grant our service permissions to process your data by accessing your account, and you can revoke the permissions at any time by uninstalling the application.

Create a connection to GitLab

This section describes how to connect GitLab to CodeBuild. For more information about GitLab connections, see Connect CodeBuild to GitLab.

Before you begin:

- You must have already created an account with GitLab.

> **Note**
>
> Connections only provide access to repositories owned by the account that was used to create and authorize the connection.

> **Note**
>
> You can create connections to a repository where you have the **Owner** role in GitLab, and then the connection can be used with the repository with resources such as CodeBuild. For repositories in groups, you do not need to be the group owner.

- To specify a source for your build project, you must have already created a repository on GitLab.

Topics

- Create a connection to GitLab (console)
- Create a connection to GitLab (CLI)

Create a connection to GitLab (console)

Use these steps to use the CodeBuild console to add a connection for your project (repository) in GitLab.

> **Note**
>
> Instead of creating or using an existing connection in your account, you can use a connection shared from another AWS account. For more information, see Share connections with AWS accounts.

To create or edit your build project

1. Sign in to the CodeBuild console.
2. Choose one of the following.

- Choose to create a build project. Follow the steps in Create a build project (console) to complete the first screen and in the **Source** section, under **Source Provider**, choose **GitLab**.
- Choose to edit an existing build project. Choose **Edit**, and then choose **Source**. In the **Edit Source** page, under **Source provider**, choose **GitLab**.

3. Choose one of the following:

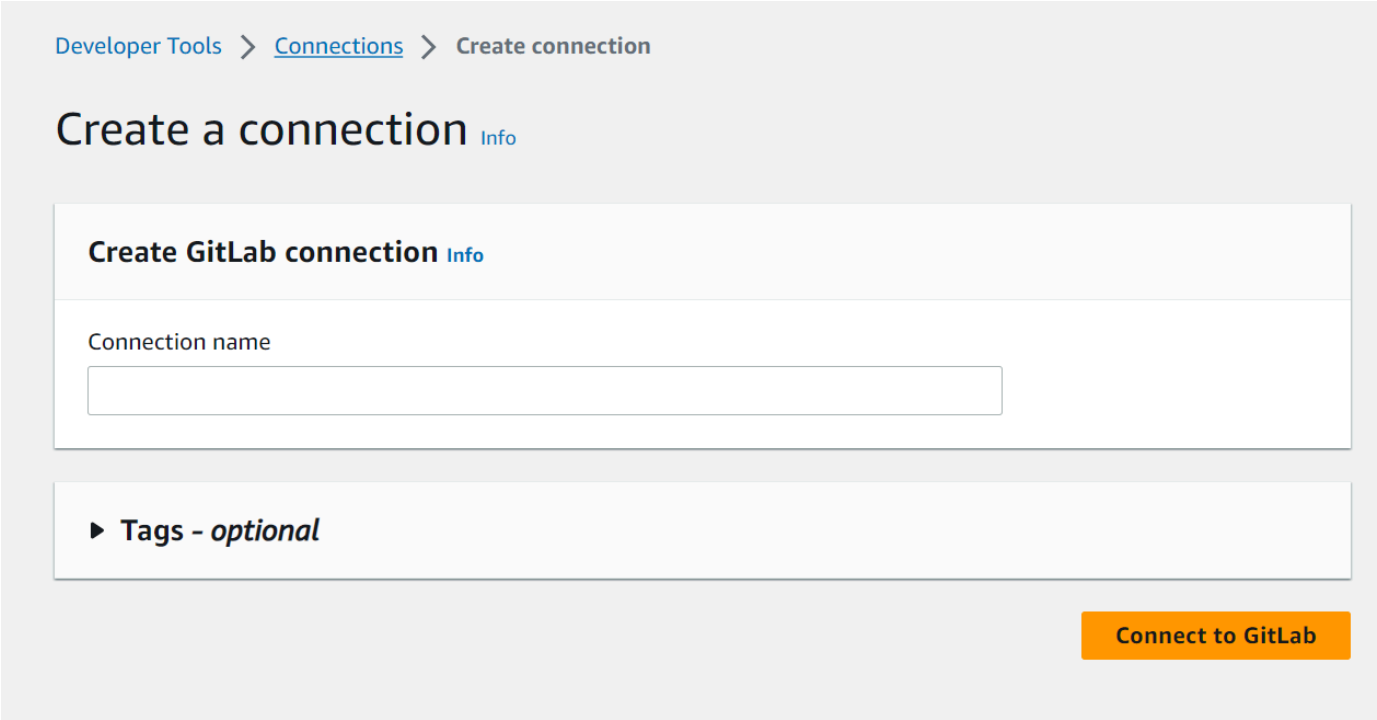
- Under **Connection**, choose **Default connection**. Default connection applies a default GitLab connection across all projects.
- Under **Connection**, choose **Custom connection**. Custom connection applies a custom GitLab connection that overrides your account's default settings.

4. Do one of the following:

- Under **Default connection** or **Custom connection**, if you have not already created a connection to your provider, choose **Create a new GitLab connection**. Proceed to step 5 to create the connection.
- Under **Connection**, if you have already created a connection to your provider, choose the connection. Proceed to step 10.

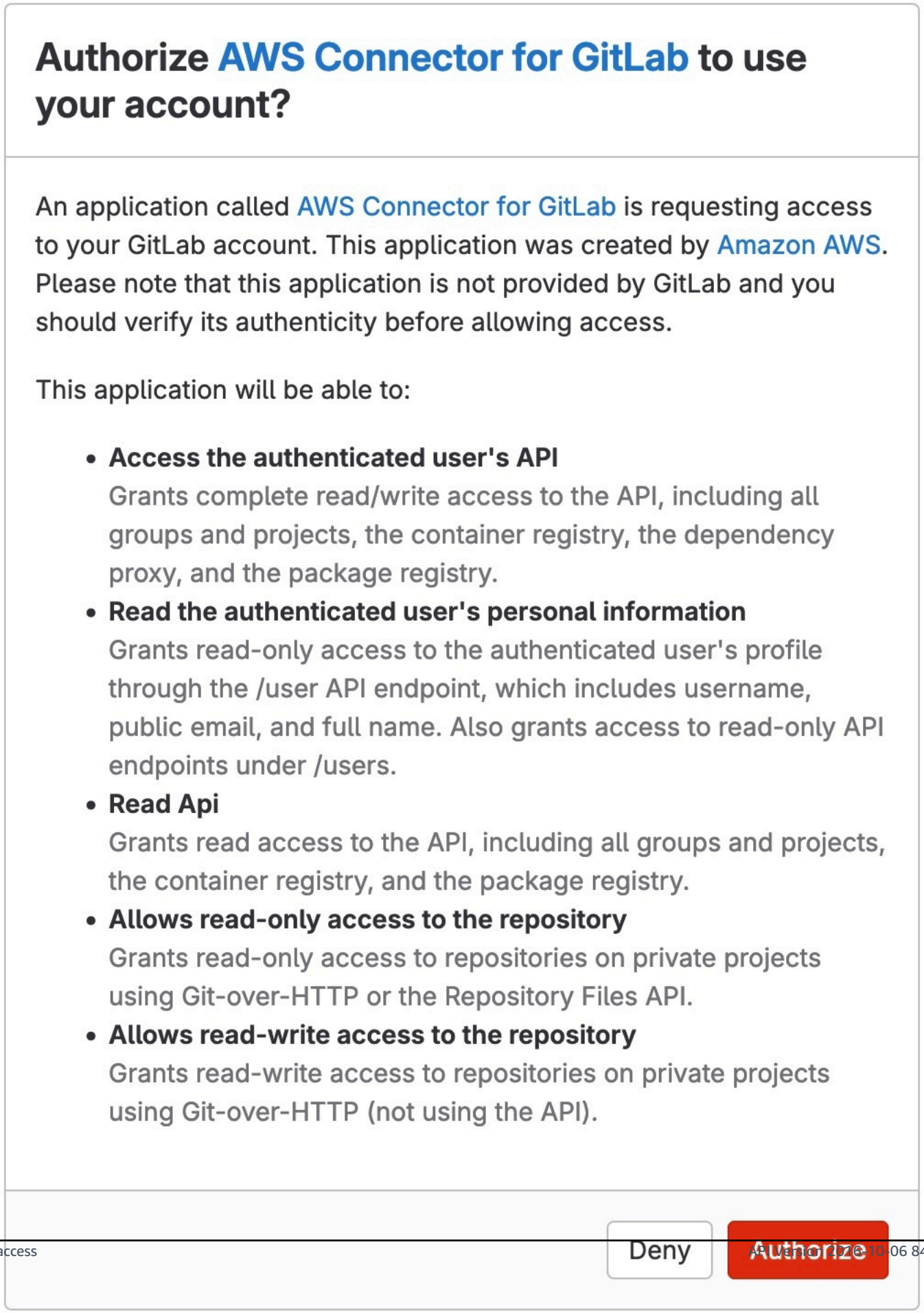
> **Note**
>
> If you close the pop-up window before a GitLab connection is created, you need to refresh the page.

5. To create a connection to a GitLab repository, under **Select a provider**, choose **GitLab**. In **Connection name**, enter the name for the connection that you want to create. Choose **Connect to GitLab**.



6. When the sign-in page for GitLab displays, log in with your credentials, and then choose **Sign in**.
7. If this is your first time authorizing the connection, an authorization page displays with a message requesting authorization for the connection to access your GitLab account.

   Choose **Authorize**.


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. The browser returns to the connections console page. Under **GitLab connection settings**, the new connection is shown in **Connection name**.
9. Choose **Connect**.

   After a GitLab connection is successfully created, a success banner will be displayed at the top.
10. On the **Create build project** page, in the **Default connection** or **Custom connection** drop-down list, make sure your connection ARN is listed. If not, choose the refresh button to have it appear.
11. In **Repository**, choose the name of your project in GitLab by specifying the project path with the namespace. For example, for a group-level repository, enter the repository name in the following format: `group-name/repository-name`. For more information about the path and namespace, see the `path_with_namespace` field in https://docs.gitlab.com/ee/api/projects.html#get-single-project. For more information about the namespace in GitLab, see https://docs.gitlab.com/ee/user/namespace/.

    > **Note**
    >
    > For groups in GitLab, you must manually specify the project path with the namespace. For example, for a repository named `myrepo` in a group `mygroup`, enter the following: `mygroup/myrepo`. You can find the project path with the namespace in the URL in GitLab.

12. In **Source version - optional**, enter a pull request ID, branch, commit ID, tag, or reference and a commit ID. For more information, see Source version sample with AWS CodeBuild.

    > **Note**
    >
    > We recommend that you choose Git branch names that don't look like commit IDs, such as `811dd1ba1aba14473856cee38308caed7190c0d` or `5392f7`. This helps you avoid Git checkout collisions with actual commits.

13. In **Git clone depth - optional**, you can create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

14. In **Build Status - optional**, select **Report build statuses to source provider when your builds start and finish** if you want the status of your build's start and completion reported to your source provider.

    To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

Create a connection to GitLab (CLI)

You can use the AWS Command Line Interface (AWS CLI) to create a connection.

To do this, use the **create-connection** command.

> **Important**
>
> A connection created through the AWS CLI or AWS CloudFormation is in `PENDING` status by default. After you create a connection with the CLI or AWS CloudFormation, use the console to edit the connection to make its status `AVAILABLE`.

To create a connection

- Follow the instructions in the *Developer Tools console User Guide* for Create a connection to GitLab (CLI).

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the aws:SourceArn and aws:SourceAccount global condition context keys in resource policies to limit the permissions that AWS CodeBuild gives another service to the resource. Use aws:SourceArn if you want only one resource to be associated with the cross-service access. Use aws:SourceAccount if you want to allow any resource in that account to be associated with the cross-service use.

The most effective way to protect against the confused deputy problem is to use the aws:SourceArn global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the aws:SourceArn global context condition key with wildcard characters (*) for the unknown portions of the ARN. For example, arn:aws:codebuild:*:*123456789012*:*.

If the aws:SourceArn value does not contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions.

The value of aws:SourceArn must be the CodeBuild project ARN.

The following example shows how you can use the aws:SourceArn and aws:SourceAccount global condition context keys in CodeBuild to prevent the confused deputy problem.

```
{
    "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"
                }
            }
        }
    ]
}
```

Advanced topics

This section includes several advanced topics that are useful to more experienced AWS CodeBuild users.

Topics

Allow users to interact with CodeBuild

If you follow the steps in Getting started using the console to access AWS CodeBuild for the first time, you most likely do not need the information in this topic. However, as you continue using CodeBuild, you might want to do things such as give other users and groups in your organization the ability to interact with CodeBuild.

To allow an IAM user or group to interact with AWS CodeBuild, you must give them access permissions to CodeBuild. This section describes how to do this with the IAM console or the AWS CLI.

If you will access CodeBuild with your AWS root account (not recommended) or an administrator user in your AWS account, then you do not need to follow these instructions.

For information about AWS root accounts and administrator users, see The AWS account root user and Creating Your First AWS account root user and Group in the *user Guide*.

To add CodeBuild access permissions to an IAM group or user (console)

1. Open the IAM console at https://console.aws.amazon.com/iam/.

   You should have already signed in to the AWS Management Console by using one of the following:

   - Your AWS root account. This is not recommended. For more information, see The AWS account root user in the *user Guide*.
   - An administrator user in your AWS account. For more information, see Creating Your First AWS account root user and Group in the *user Guide*.
   - An user in your AWS account with permission to perform the following minimum set of actions:

     ```
     iam:AttachGroupPolicy
     iam:AttachUserPolicy
     iam:CreatePolicy
     iam:ListAttachedGroupPolicies
     iam:ListAttachedUserPolicies
     iam:ListGroups
     iam:ListPolicies
     iam:ListUsers
     ```

     For more information, see Overview of IAM Policies in the *user Guide*.
2. In the navigation pane, choose **Policies**.
3. To add a custom set of AWS CodeBuild access permissions to an IAM group or IAM user, skip ahead to step 4 in this procedure.

   To add a default set of CodeBuild access permissions to an IAM group or IAM user, choose **Policy Type**, **AWS Managed**, and then do the following:

   - To add full access permissions to CodeBuild, select the box named **AWSCodeBuildAdminAccess**, choose **Policy Actions**, and then choose **Attach**. Select the box next to the target IAM group or user, and then choose **Attach Policy**. Repeat this for the policies named **AmazonS3ReadOnlyAccess** and **IAMFullAccess**.

- To add access permissions to CodeBuild for everything except build project administration, select the box named **AWSCodeBuildDeveloperAccess**, choose **Policy Actions**, and then choose **Attach**. Select the box next to the target IAM group or user, and then choose **Attach Policy**. Repeat this for the policy named **AmazonS3ReadOnlyAccess**.
- To add read-only access permissions to CodeBuild, select the boxes named **AWSCodeBuildReadOnlyAccess**. Select the box next to the target IAM group or user, and then choose **Attach Policy**. Repeat this for the policy named **AmazonS3ReadOnlyAccess**.

You have now added a default set of CodeBuild access permissions to an IAM group or user. Skip the rest of the steps in this procedure.

4. Choose **Create Policy**.
5. On the **Create Policy** page, next to **Create Your Own Policy**, choose **Select**.
6. On the **Review Policy** page, for **Policy Name**, enter a name for the policy (for example, **`CodeBuildAccessPolicy`**). If you use a different name, be sure to use it throughout this procedure.
7. For **Policy Document**, enter the following, and then choose **Create Policy**.

```
{
  "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**
>
> This policy allows access to all CodeBuild actions and to a potentially large number of AWS resources. To restrict permissions to specific CodeBuild actions, change the value of `codebuild:*` in the CodeBuild policy statement. For more information, see Identity and access management. To restrict access to specific AWS resources, change the value of the `Resource` object. For more information, see Identity and access management.

8. In the navigation pane, choose **Groups** or **Users**.
9. In the list of groups or users, choose the name of the IAM group or IAM user to which you want to add CodeBuild access permissions.

10. For a group, on the group settings page, on the **Permissions** tab, expand **Managed Policies**, and then choose **Attach Policy**.

    For a user, on the user settings page, on the **Permissions** tab, choose **Add permissions**.

11. For a group, on the **Attach Policy** page, select **CodeBuildAccessPolicy**, and then choose **Attach Policy**.

    For a user, on the **Add permissions** page, choose **Attach existing policies directly**. Select **CodeBuildAccessPolicy**, choose **Next: Review**, and then choose **Add permissions**.

To add CodeBuild access permissions to an IAM group or user (AWS CLI)

1. Make sure you have configured the AWS CLI with the AWS access key and AWS secret access key that correspond to one of the IAM entities, as described in the previous procedure. For more information, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.
2. To add a custom set of AWS CodeBuild access permissions to an IAM group or IAM user, skip to step 3 in this procedure.

   To add a default set of CodeBuild access permissions to an IAM group or IAM user, do the following:

   Run one of the following commands, depending on whether you want to add permissions to an IAM group or user:

   ```
   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
   ```

   You must run the command three times, replacing *group-name* or *user-name* with the IAM group name or user name, and replacing *policy-arn* once for each of the following policy Amazon Resource Names (ARNs):

   - To add full access permissions to CodeBuild, use the following policy ARNs:
     - `arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess`
     - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`
     - `arn:aws:iam::aws:policy/IAMFullAccess`

- To add access permissions to CodeBuild for everything except build project administration, use the following policy ARNs:
  - `arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess`
  - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`
- To add read-only access permissions to CodeBuild, use the following policy ARNs:
  - `arn:aws:iam::aws:policy/AWSCodeBuildReadOnlyAccess`
  - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`

You have now added a default set of CodeBuild access permissions to an IAM group or user. Skip the rest of the steps in this procedure.

3. In an empty directory on the local workstation or instance where the AWS CLI is installed, create a file named `put-group-policy.json` or `put-user-policy.json`. If you use a different file name, be sure to use it throughout this procedure.

```
{
  "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**
>
> This policy allows access to all CodeBuild actions and to a potentially large number of AWS resources. To restrict permissions to specific CodeBuild actions, change the value of `codebuild:*` in the CodeBuild policy statement. For more information, see Identity and access management. To restrict access to specific AWS resources, change the value of the related `Resource` object. For more information, see Identity and access management or the specific AWS service's security documentation.

4. Switch to the directory where you saved the file, and then run one of the following commands. You can use different values for `CodeBuildGroupAccessPolicy` and `CodeBuildUserAccessPolicy`. If you use different values, be sure to use them here.

   For an IAM group:

```
aws iam put-group-policy --group-name group-name --policy-name
 CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json
```

For an user:

```
aws iam put-user-policy --user-name user-name --policy-name
 CodeBuildUserAccessPolicy --policy-document file://put-user-policy.json
```

In the preceding commands, replace *group-name* or *user-name* with the name of the target IAM group or user.

Allow CodeBuild to interact with other AWS services

If you follow the steps in Getting started using the console to access AWS CodeBuild for the first time, you most likely do not need the information in this topic. However, as you continue using CodeBuild, you might want to do things such as allow CodeBuild to interact with other AWS services.

To allow CodeBuild to interact with dependent AWS services on your behalf, you need an AWS CodeBuild service role. You can create a CodeBuild service role by using the CodeBuild or AWS CodePipeline consoles. For information, see:

- Create a build project (console)
- Create a pipeline that uses CodeBuild (CodePipeline console)
- Add a CodeBuild build action to a pipeline (CodePipeline console)
- Change a build project's settings (console)

If you do not plan to use these consoles, this section describes how to create a CodeBuild service role with the IAM console or the AWS CLI.

Important

CodeBuild uses the service role for all operations that are performed on your behalf. If the role includes permissions that the user shouldn't have, you can unintentionally escalate a user's permissions. Ensure that the role grants least privilege.

The service role described on this page contains a policy that grants the minimum permissions required to use CodeBuild. You may need to add additional permissions, depending on your use case.

To create a CodeBuild service role (console)

1. Open the IAM console at https://console.aws.amazon.com/iam/.

   You should have already signed in to the console by using one of the following:

   - Your AWS root account. This is not recommended. For more information, see The AWS account root user in the *user Guide*.
   - An administrator user in your AWS account. For more information, see Creating Your First AWS account root user and Group in the *user Guide*.
   - An user in your AWS account with permission to perform the following minimum set of actions:

     ```
     iam:AddRoleToInstanceProfile
     iam:AttachRolePolicy
     iam:CreateInstanceProfile
     iam:CreatePolicy
     iam:CreateRole
     iam:GetRole
     iam:ListAttachedRolePolicies
     iam:ListPolicies
     iam:ListRoles
     iam:PassRole
     iam:PutRolePolicy
     iam:UpdateAssumeRolePolicy
     ```

     For more information, see Overview of IAM Policies in the *user Guide*.
2. In the navigation pane, choose **Policies**.
3. Choose **Create Policy**.
4. On the **Create Policy** page, choose **JSON**.
5. For the JSON policy, enter the following, and then choose **Review Policy**:

   ```
   {
     "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**
>
> This policy contains statements that allow access to a potentially large number of AWS resources. To restrict AWS CodeBuild to access specific AWS resources, change the value of the `Resource` array. For more information, see the security documentation for the AWS service.

6. On the **Review Policy** page, for **Policy Name**, enter a name for the policy (for example, **`CodeBuildServiceRolePolicy`**), and then choose **Create policy**.

> **Note**
>
> If you use a different name, be sure to use it throughout this procedure.

7. In the navigation pane, choose **Roles**.
8. Choose **Create role**.
9. On the **Create role** page, with **AWS Service** already selected, choose **CodeBuild**, and then choose **Next:Permissions**.

10. On the **Attach permissions policies** page, select **CodeBuildServiceRolePolicy**, and then choose **Next: Review**.
11. On the **Create role and review** page, for **Role name**, enter a name for the role (for example, **`CodeBuildServiceRole`**), and then choose **Create role**.

To create a CodeBuild service role (AWS CLI)

1. Make sure you have configured the AWS CLI with the AWS access key and AWS secret access key that correspond to one of the IAM entities, as described in the previous procedure. For more information, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.
2. In an empty directory on the local workstation or instance where the AWS CLI is installed, create two files named `create-role.json` and `put-role-policy.json`. If you choose different file names, be sure to use them throughout this procedure.

   `create-role.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

> **Note**
>
> We recommend that you use the `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against the confused deputy problem. For example, you can edit the previous trust policy with the following condition blocks. The `aws:SourceAccount` is the owner of the CodeBuild project and the `aws:SourceArn` is the CodeBuild project ARN.

If you would like to restrict your service role to an AWS account, `create-role.json` might look similar to this:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "account-ID"
                    ]
                }
            }
        }
    ]
}
```

If you would like to restrict your service role to a specific CodeBuild project, `create-role.json` might look similar to this:

```
{
    "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**
>
> If you don't know or haven't decided on a name for your CodeBuild project and want a trust policy restriction on a particular ARN pattern, you can replace that portion of the ARN with a wildcard (*). After you create your project, you can then update the trust policy.

`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**
>
> This policy contains statements that allow access to a potentially large number of AWS resources. To restrict AWS CodeBuild to access specific AWS resources, change the value of the `Resource` array. For more information, see the security documentation for the AWS service.

3. Switch to the directory where you saved the preceding files, and then run the following two commands, one at a time, in this order. You can use different values for `CodeBuildServiceRole` and `CodeBuildServiceRolePolicy`, but be sure to use them here.

```
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
```

Encrypt build outputs using a customer managed key

If you follow the steps in Getting started using the console to access AWS CodeBuild for the first time, you most likely do not need the information in this topic. However, as you continue using CodeBuild, you might want to do things such as encrypt build artifacts.

For AWS CodeBuild to encrypt its build output artifacts, it needs access to a KMS key. By default, CodeBuild uses the AWS managed key for Amazon S3 in your AWS account.

If you do not want to use the AWS managed key, you must create and configure a customer managed key yourself. This section describes how to do this with the IAM console.

For information about customer managed keys, see AWS Key Management Service Concepts and Creating Keys in the *AWS KMS Developer Guide*.

To configure a customer managed key for use by CodeBuild, follow the instructions in the "How to Modify a Key Policy" section of Modifying a Key Policy in the *AWS KMS Developer Guide*. Then add the following statements (between *### BEGIN ADDING STATEMENTS HERE ###* and *### END ADDING STATEMENTS HERE ###*) to the key policy. Ellipses (`...`) are used for brevity and to help you locate where to add the statements. Do not remove any statements, and do not type these ellipses into the key policy.

```
{
  "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* represents the ID of the AWS region where the Amazon S3 buckets associated with CodeBuild are located (for example, `us-east-1`).

- *account-ID* represents the ID of the of the AWS account that owns the customer managed key.
- *CodeBuild-service-role* represents the name of the CodeBuild service role you created or identified earlier in this topic.

> **Note**
>
> To create or configure a customer managed key through the IAM console, you must first sign in to the AWS Management Console by using one of the following:
>
> - Your AWS root account. This is not recommended. For more information, see The Account Root User in the *user Guide*.
> - An administrator user in your AWS account. For more information, see Creating Your First AWS account root user and Group in the *user Guide*.
> - An user in your AWS account with permission to create or modify the customer managed key. For more information, see Permissions Required to Use the AWS KMS Console in the *AWS KMS Developer Guide*.

Interact with CodeBuild using the AWS CLI

If you follow the steps in Getting started using the console to access AWS CodeBuild for the first time, you most likely do not need the information in this topic. However, as you continue using CodeBuild, you might want to do things such as allow users to use the AWS CLI to interact with CodeBuild instead of (or in addition to) the CodeBuild console, the CodePipeline console, or the AWS SDKs.

To install and configure the AWS CLI, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

After installing the AWS CLI, complete the following tasks:

1. Run the following command to confirm whether your installation of the AWS CLI supports CodeBuild:

   ```
   aws codebuild list-builds
   ```

   If successful, information similar to the following will appear in the output:

```
{
  "ids": []
}
```

The empty square brackets indicate that you have not yet run any builds.

2. If an error is output, you must uninstall your current version of the AWS CLI and then install the latest version. For more information, see Uninstalling the AWS CLI and Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

Command line reference for AWS CodeBuild

The AWS CLI provides commands for automating AWS CodeBuild. Use the information in this topic as a supplement to the AWS Command Line Interface User Guide and the AWS CLI Reference for AWS CodeBuild.

Not what you're looking for? If you want to use the AWS SDKs to call CodeBuild, see the AWS SDKs and tools reference.

To use the information in this topic, you should have already installed the AWS CLI and configured it for use with CodeBuild, as described in Interact with CodeBuild using the AWS CLI.

To use the AWS CLI to specify the endpoint for CodeBuild, see Specify the AWS CodeBuild endpoint (AWS CLI).

Run this command to get a list of CodeBuild commands.

```
aws codebuild help
```

Run this command to get information about a CodeBuild command, where *command-name* is the name of the command.

```
aws codebuild command-name help
```

CodeBuild commands include:

- `batch-delete-builds`: Deletes one or more builds in CodeBuild. For more information, see Delete builds (AWS CLI).

- `batch-get-builds`: Gets information about multiple builds in CodeBuild. For more information, see View build details (AWS CLI).
- `batch-get-projects`: Gets information about one or more specified build projects. For more information, see View a build project's details (AWS CLI).
- `create-project`: Creates a build project. For more information, see Create a build project (AWS CLI).
- `delete-project`: Deletes a build project. For more information, see Delete a build project (AWS CLI).
- `list-builds`: Lists Amazon Resource Names (ARNs) for builds in CodeBuild. For more information, see View a list of build IDs (AWS CLI).
- `list-builds-for-project`: Gets a list of build IDs that are associated with a specified build project. For more information, see View a list of build IDs for a build project (AWS CLI).
- `list-curated-environment-images`: Gets a list of Docker images managed by CodeBuild that you can use for your builds. For more information, see Docker images provided by CodeBuild.
- `list-projects`: Gets a list of build project names. For more information, see View a list of build project names (AWS CLI).
- `start-build`: Starts running a build. For more information, see Run a build (AWS CLI).
- `stop-build`: Attempts to stop the specified build from running. For more information, see Stop a build (AWS CLI).
- `update-project`: Changes information about the specified build project. For more information, see Change a build project's settings (AWS CLI).

AWS SDKs and tools reference for AWS CodeBuild

To use one the AWS SDKs or tools to automate AWS CodeBuild, see the following resources.

If you want to use the AWS CLI to run CodeBuild, see the Command line reference.

Supported AWS SDKs and tools for AWS CodeBuild

The following AWS SDKs and tools support CodeBuild:

- The AWS SDK for C++. For more information, see the Aws::CodeBuild namespace section of the *AWS SDK for C++ API Reference*.

- The AWS SDK for Go. For more information, see the codebuild section of the *AWS SDK for Go API Reference*.
- The AWS SDK for Java. For more information, see the `com.amazonaws.services.codebuild` and `com.amazonaws.services.codebuild.model` sections of the AWS SDK for Java API reference.
- The AWS SDK for JavaScript in the browser and the AWS SDK for JavaScript in Node.js. For more information, see the Class: AWS.CodeBuild section of the *AWS SDK for JavaScript API Reference*.
- The AWS SDK for .NET. For more information, see the Amazon.CodeBuild and Amazon.CodeBuild.Model namespace sections of the *AWS SDK for .NET API Reference*.
- The AWS SDK for PHP. For more information, see the Namespace Aws\CodeBuild section of the *AWS SDK for PHP API Reference*.
- The AWS SDK for Python (Boto3). For more information, see the CodeBuild section of the *Boto 3 Documentation*.
- The AWS SDK for Ruby. For more information, see the Module: Aws::CodeBuild section of the *AWS SDK for Ruby API Reference*.
- The AWS Tools for PowerShell. For more information, see the AWS CodeBuild section of the *AWS Tools for PowerShell Cmdlet Reference*.

Using this service with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS CLI	AWS CLI code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples

SDK documentation	Code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	Tools for PowerShell code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

For examples specific to this service, see Code examples for CodeBuild using AWS SDKs.

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

Specify the AWS CodeBuild endpoint

You can use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to specify the endpoint used by AWS CodeBuild. There is an endpoint for each region in which CodeBuild is available. In addition to a regional endpoint, four regions also have a Federal Information Processing Standards (FIPS) endpoint. For more information about FIPS endpoints, see FIPS 140-2 overview.

Specifying an endpoint is optional. If you don't explicitly tell CodeBuild which endpoint to use, the service uses the endpoint associated with the region your AWS account uses. CodeBuild never

defaults to a FIPS endpoint. If you want to use a FIPS endpoint, you must associate CodeBuild with it using one of the following methods.

> **Note**
>
> You can use an alias or region name to specify an endpoint using an AWS SDK. If you use the AWS CLI, then you must use the full endpoint name.

For endpoints that can be used with CodeBuild, see CodeBuild regions and endpoints.

Topics

Specify the AWS CodeBuild endpoint (AWS CLI)

You can use the AWS CLI to specify the endpoint through which AWS CodeBuild is accessed by using the `--endpoint-url` argument in any CodeBuild command. For example, run this command to get a list of project build names using the Federal Information Processing Standards (FIPS) endpoint in the US East (N. Virginia) Region:

```
aws codebuild list-projects --endpoint-url https://codebuild-fips.us-
east-1.amazonaws.com
```

Include the `https://` at the begining of the endpoint.

The `--endpoint-url` AWS CLI argument is available to all AWS services. For more information about this and other AWS CLI arguments, see AWS CLI Command Reference.

Specify the AWS CodeBuild endpoint (AWS SDK)

You can use an AWS SDK to specify the endpoint through which AWS CodeBuild is accessed. Although this example uses the AWS SDK for Java, you can specify the endpoint with the other AWS SDKs.

Use the `withEndpointConfiguration` method when constructing the AWSCodeBuild client. Here is format to use:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("endpoint",
 "region")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

For information about `AWSCodeBuildClientBuilder`, see Class AWSCodeBuildClientBuilder.

The credentials used in `withCredentials` must be of type `AWSCredentialsProvider`. For more information, see Working with AWS credentials.

Do not include `https://` at the begining of the endpoint.

If you want to specify a non-FIPS endpoint, you can use the region instead of the actual endpoint. For example, to specify the endpoint in the US East (N. Virginia) region, you can use `us-east-1` instead of the full endpoint name, `codebuild.us-east-1.amazonaws.com`.

If you want to specify a FIPS endpoint, you can use an alias to simplify your code. Only FIPS endpoints have an alias. Other endpoints must be specified using their region or full name.

The following table lists the alias for each of the four available FIPS endpoints:

Region name	Region	Endpoint	Alias
US East (N. Virginia)	us-east-1	codebuild-fips.us-east-1.amazonaws.com	us-east-1-fips
US East (Ohio)	us-east-2	codebuild-fips.us-east-2.amazonaws.com	us-east-2-fips
US West (N. California)	us-west-1	codebuild-fips.us-west-1.amazonaws.com	us-west-1-fips
US West (Oregon)	us-west-2	codebuild-fips.us-west-2.amazonaws.com	us-west-2-fips

To specify use of the FIPS endpoint in the US West (Oregon) region using an alias:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-west-2-
fips", "us-west-2")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

To specify use of the non-FIPS endpoint in the US East (N. Virginia) region:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-east-1",
 "us-east-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

To specify use of the non-FIPS endpoint in the Asia Pacific (Mumbai) region:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1",
 "ap-south-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

Use AWS CodeBuild with AWS CodePipeline to test code and run builds

You can automate your release process by using AWS CodePipeline to test your code and run your builds with AWS CodeBuild.

The following table lists tasks and the methods available for performing them. Using the AWS SDKs to accomplish these tasks is outside the scope of this topic.

Task	Available approaches	Approaches described in this topic
Create a continuous delivery (CD) pipeline with CodePipeline that automates builds with CodeBuild	• CodePipeline console • AWS CLI • AWS SDKs	• Use the CodePipeline console • Use the AWS CLI • You can adapt the information in this topic to use the AWS SDKs. For more information, see the `create-pipeline` action documentation for your programming language in the SDKs section of *Tools for Amazon Web Services* or see `CreatePipeline` in the *AWS CodePipeline API Reference*.
Add test and build automation with CodeBuild to an existing pipeline in CodePipeline	• CodePipeline console • AWS CLI • AWS SDKs	• Use the CodePipeline console to add build automation • Use the CodePipeline console to add test automation • For the AWS CLI, you can adapt the information in this topic to create a pipeline that contains a CodeBuild build action or test action. For more information, see Edit a pipeline (AWS CLI) and the CodePipeline pipeline structure reference in the *AWS CodePipeline User Guide*. • You can adapt the information in this topic to use the AWS SDKs. For more information, see the `update-pipeline` action documentation for your programming language through the SDKs section of *Tools for Amazon Web Services* or see `UpdatePipeline` in the *AWS CodePipeline API Reference*.

Topics

Prerequisites

1. Answer the questions in Plan a build.
2. If you are using an user to access CodePipeline instead of an AWS root account or an administrator user, attach the managed policy named `AWSCodePipelineFullAccess` to the user (or to the IAM group to which the user belongs). Using an AWS root account is not recommended. This policy grants the user permission to create the pipeline in CodePipeline. For more information, see Attaching managed policies in the *user Guide*.

   > **Note**
   >
   > The IAM entity that attaches the policy to the user (or to the IAM group to which the user belongs) must have permission in IAM to attach policies. For more information, see Delegating permissions to administer IAM users, groups, and credentials in the *user Guide*.

3. Create a CodePipeline service role, if you do not already have one available in your AWS account. CodePipeline uses this service role to interact with other AWS services, including AWS CodeBuild, on your behalf. For example, to use the AWS CLI to create a CodePipeline service role, run the IAM `create-role` command:

   For Linux, macOS, or Unix:

   ```
   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"}}'
   ```

   For Windows:

   ```
   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**
>
> The IAM entity that creates this CodePipeline service role must have permission in IAM to create service roles.

4. After you create a CodePipeline service role or identify an existing one, you must add the default CodePipeline service role policy to the service role as described in Review the default CodePipeline service role policy in the *AWS CodePipeline User Guide*, if it isn't already a part of the policy for the role.

> **Note**
>
> The IAM entity that adds this CodePipeline service role policy must have permission in IAM to add service role policies to service roles.

5. Create and upload the source code to a repository type supported by CodeBuild and CodePipeline, such as CodeCommit, Amazon S3, Bitbucket, or GitHub. The source code should contain a buildspec file, but you can declare one when you define a build project later in this topic. For more information, see the Buildspec reference.

> **Important**
>
> If you plan to use the pipeline to deploy built source code, the build output artifact must be compatible with the deployment system you use.
>
> - For AWS OpsWorks, see Application source and Using CodePipeline with AWS OpsWorks in the *AWS OpsWorks User Guide*.

Create a pipeline that uses CodeBuild (CodePipeline console)

Use the following procedure to create a pipeline that uses CodeBuild to build and deploy your source code.

To create a pipeline that only tests your source code:

- Use the following procedure to create the pipeline, and then delete the Build and Beta stages from the pipeline. Then use the Add a CodeBuild test action to a pipeline (CodePipeline console) procedure in this topic to add to the pipeline a test action that uses CodeBuild.
- Use one of the other procedures in this topic to create the pipeline, and then use the Add a CodeBuild test action to a pipeline (CodePipeline console) procedure in this topic to add to the pipeline a test action that uses CodeBuild.

To use the create pipeline wizard in CodePipeline to create a pipeline that uses CodeBuild

1. Sign in to the AWS Management Console by using:

   - Your AWS root account. This is not recommended. For more information, see The account root user in the *user Guide*.
   - An administrator user in your AWS account. For more information, see Creating your first AWS account root user and group in the *user Guide*.
   - An user in your AWS account with permission to use the following minimum set of actions:

```
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. Open the AWS CodePipeline console at https://console.aws.amazon.com/codesuite/codepipeline/home.
3. In the AWS Region selector, choose the AWS Region where your build project AWS resources are located. This must be an AWS Region where CodeBuild is supported. For more information, see AWS CodeBuild in the *Amazon Web Services General Reference*.
4. Create a pipeline. If a CodePipeline information page is displayed, choose **Create pipeline**. If a **Pipelines** page is displayed, choose **Create pipeline**.
5. On the **Step 1: Choose pipeline settings** page, for **Pipeline name**, enter a name for the pipeline (for example, **`CodeBuildDemoPipeline`**). If you choose a different name, be sure to use it throughout this procedure.
6. For **Role name**, do one of the following:

   Choose **New service role**, and in **Role Name**, enter the name for your new service role.

   Choose **Existing service role**, and then choose the CodePipeline service role you created or identified as part of this topic's prerequisites.
7. For **Artifact store**, do one of the following:

   - Choose **Default location** to use the default artifact store, such as the S3 artifact bucket designated as the default, for your pipeline in the AWS Region you have selected for your pipeline.
   - Choose **Custom location** if you already have an existing artifact store you have created, such as an S3 artifact bucket, in the same AWS Region as your pipeline.

   > **Note**
   >
   > This is not the source bucket for your pipeline's source code. This is the artifact store for your pipeline. A separate artifact store, such as an S3 bucket, is required for each pipeline, in the same AWS Region as the pipeline.
8. Choose **Next**.
9. On the **Step 2: Add source stage** page, for **Source provider**, do one of the following:

   - If your source code is stored in an S3 bucket, choose **Amazon S3**. For **Bucket**, select the S3 bucket that contains your source code. For **S3 object key**, enter the name of the file the contains the source code (for example, *`file-name`*`.zip`). Choose **Next**.

- If your source code is stored in an AWS CodeCommit repository, choose **CodeCommit**. For **Repository name**, choose the name of the repository that contains the source code. For **Branch name**, choose the name of the branch that contains the version of the source code you want to build. Choose **Next**.
- If your source code is stored in a GitHub repository, choose **GitHub**. Choose **Connect to GitHub**, and follow the instructions to authenticate with GitHub. For **Repository**, choose the name of the repository that contains the source code. For **Branch**, choose the name of the branch that contains the version of the source code you want to build.

  Choose **Next**.

10. On the **Step 3: Add build stage** page, for **Build provider**, choose **CodeBuild**.
11. If you already have a build project you want to use, for **Project name**, choose the name of the build project and skip to the next step in this procedure.

    If you need to create a new CodeBuild build project, follow the instructions in Create a build project (console) and return to this procedure.

    If you choose an existing build project, it must have build output artifact settings already defined (even though CodePipeline overrides them). For more information, see Change a build project's settings (console).

    > **Important**
    >
    > If you enable webhooks for a CodeBuild project, and the project is used as a build step in CodePipeline, then two identical builds are created for each commit. One build is triggered through webhooks, and one through CodePipeline. Because billing is on a per-build basis, you are billed for both builds. Therefore, if you are using CodePipeline, we recommend that you disable webhooks in CodeBuild. In the AWS CodeBuild console, clear the **Webhook** box. For more information, see Change a build project's settings (console).

12. On the **Step 4: Add deploy stage** page, do one of the following:

    - If you do not want to deploy the build output artifact, choose **Skip**, and confirm this choice when prompted.
    - If you want to deploy the build output artifact, for **Deploy provider**, choose a deployment provider, and then specify the settings when prompted.

Choose **Next**.

13. On the **Review** page, review your choices, and then choose **Create pipeline**.
14. After the pipeline runs successfully, you can get the build output artifact. With the pipeline displayed in the CodePipeline console, in the **Build** action, choose the tooltip. Make a note of the value for **Output artifact** (for example, **MyAppBuild**).

    > **Note**
    >
    > You can also get the build output artifact by choosing the **Build artifacts** link on the build details page in the CodeBuild console. To get to this page, skip the rest of the steps in this procedure, and see View build details (console).

15. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
16. In the list of buckets, open the bucket used by the pipeline. The name of the bucket should follow the format `codepipeline-`*`region-ID`*`-`*`random-number`*. You can use the AWS CLI to run the CodePipeline **get-pipeline** command to get the name of the bucket, where *`my-pipeline-name`* is the display name of your pipeline:

    ```
    aws codepipeline get-pipeline --name my-pipeline-name
    ```

    In the output, the `pipeline` object contains an `artifactStore` object, which contains a `location` value with the name of the bucket.
17. Open the folder that matches the name of your pipeline (depending on the length of the pipeline's name, the folder name might be truncated), and then open the folder that matches the value for **Output artifact** that you noted earlier.
18. Extract the contents of the file. If there are multiple files in that folder, extract the contents of the file with the latest **Last Modified** timestamp. (You might need to give the file the `.zip` extension so that you can work with it in your system's ZIP utility.) The build output artifact is in the extracted contents of the file.
19. If you instructed CodePipeline to deploy the build output artifact, use the deployment provider's instructions to get to the build output artifact on the deployment targets.

Create a pipeline that uses CodeBuild (AWS CLI)

Use the following procedure to create a pipeline that uses CodeBuild to build your source code.

To use the AWS CLI to create a pipeline that deploys your built source code or that only tests your source code, you can adapt the instructions in Edit a pipeline (AWS CLI) and the CodePipeline pipeline structure reference in the *AWS CodePipeline User Guide*.

1. Create or identify a build project in CodeBuild. For more information, see Create a build project.

   > **Important**
   >
   > The build project must define build output artifact settings (even though CodePipeline overrides them). For more information, see the description of `artifacts` in Create a build project (AWS CLI).

2. Make sure you have configured the AWS CLI with the AWS access key and AWS secret access key that correspond to one of the IAM entities described in this topic. For more information, see Getting set up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.
3. Create a JSON-formatted file that represents the structure of the pipeline. Name the file `create-pipeline.json` or similar. For example, this JSON-formatted structure creates a pipeline with a source action that references an S3 input bucket and a build action that uses CodeBuild:

```
{
  "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
  }
}
```

In this JSON-formatted data:

- The value of `roleArn` must match the ARN of the CodePipeline service role you created or identified as part of the prerequisites.
- The values of `S3Bucket` and `S3ObjectKey` in `configuration` assume the source code is stored in an S3 bucket. For settings for other source code repository types, see the CodePipeline pipeline structure reference in the *AWS CodePipeline User Guide*.
- The value of `ProjectName` is the name of the CodeBuild build project you created earlier in this procedure.
- The value of `location` is the name of the S3 bucket used by this pipeline. For more information, see Create a policy for an S3 Bucket to use as the artifact store for CodePipeline in the *AWS CodePipeline User Guide*.
- The value of `name` is the name of this pipeline. All pipeline names must be unique to your account.

Although this data describes only a source action and a build action, you can add actions for activities related to testing, deploying the build output artifact, invoking AWS Lambda functions, and more. For more information, see the AWS CodePipeline pipeline structure reference in the *AWS CodePipeline User Guide*.

4. Switch to the folder that contains the JSON file, and then run the CodePipeline **create-pipeline** command, specifying the file name:

```
aws codepipeline create-pipeline --cli-input-json file://create-pipeline.json
```

> **Note**
>
> You must create the pipeline in an AWS Region where CodeBuild is supported. For more information, see AWS CodeBuild in the *Amazon Web Services General Reference*.

The JSON-formatted data appears in the output, and CodePipeline creates the pipeline.

5. To get information about the pipeline's status, run the CodePipeline **get-pipeline-state** command, specifying the name of the pipeline:

```
aws codepipeline get-pipeline-state --name <my-pipeline-name>
```

In the output, look for information that confirms the build was successful. Ellipses (`...`) are used to show data that has been omitted for brevity.

```
{
  ...
  "stageStates": [
    ...
    {
      "actionStates": [
        {
          "actionName": "CodeBuild",
          "latestExecution": {
            "status": "SUCCEEDED",
            ...
          },
          ...
        }
      ]
    }
  ]
}
```

If you run this command too early, you might not see any information about the build action. You might need to run this command multiple times until the pipeline has finished running the build action.

6. After a successful build, follow these instructions to get the build output artifact. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   > **Note**
   >
   > You can also get the build output artifact by choosing the **Build artifacts** link on the related build details page in the CodeBuild console. To get to this page, skip the rest of the steps in this procedure, and see View build details (console).

7. In the list of buckets, open the bucket used by the pipeline. The name of the bucket should follow the format `codepipeline-<region-ID>-<random-number>`. You can get the bucket name from the `create-pipeline.json` file or you can run the CodePipeline **get-pipeline** command to get the bucket's name.

   ```
   aws codepipeline get-pipeline --name <pipeline-name>
   ```

   In the output, the `pipeline` object contains an `artifactStore` object, which contains a `location` value with the name of the bucket.

8. Open the folder that matches the name of your pipeline (for example, *<pipeline-name>*).

9. In that folder, open the folder named `default`.

10. Extract the contents of the file. If there are multiple files in that folder, extract the contents of the file with the latest **Last Modified** timestamp. (You might need to give the file a `.zip` extension so that you can work with it in your system's ZIP utility.) The build output artifact is in the extracted contents of the file.

Add a CodeBuild build action to a pipeline (CodePipeline console)

1. Sign in to the AWS Management Console by using:

   - Your AWS root account. This is not recommended. For more information, see The account root user in the *user Guide*.
   - An administrator user in your AWS account. For more information, see Creating your first AWS account root user and group in the *user Guide*.
   - An user in your AWS account with permission to perform the following minimum set of actions:

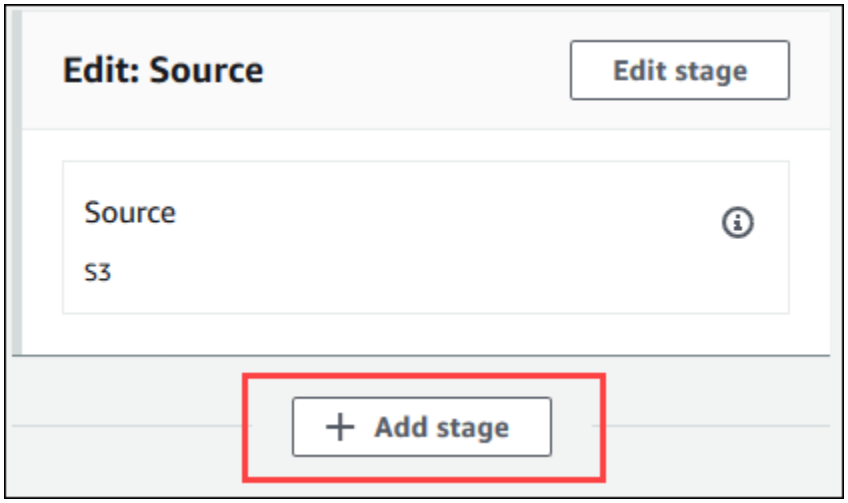
```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. Open the CodePipeline console at https://console.aws.amazon.com/codesuite/codepipeline/home.
3. In the AWS region selector, choose the AWS Region where your pipeline is located. This must be a Region where CodeBuild is supported. For more information, see CodeBuild in the *Amazon Web Services General Reference*.
4. On the **Pipelines** page, choose the name of the pipeline.
5. On the pipeline details page, in the **Source** action, choose the tooltip. Make a note of the value for **Output artifact** (for example, **MyApp**).

> **Note**
>
> This procedure shows you how to add a build action in a build stage between the **Source** and **Beta** stages. If you want to add the build action somewhere else, choose the tooltip on the action just before the place where you want to add the build action, and make a note of the value for **Output artifact**.

6. Choose **Edit**.
7. Between the **Source** and **Beta** stages, choose **Add stage**.

> **Note**
>
> This procedure shows you how to add a build stage between the **Source** and **Beta** stages to your pipeline. To add a build action to an existing stage, choose **Edit stage** in the stage, and then skip to step 8 of this procedure. To add the build stage somewhere else, choose **Add stage** in the desired place.



8. For **Stage name**, enter the name of the build stage (for example, **`Build`**). If you choose a different name, use it throughout this procedure.
9. Inside of the selected stage, choose **Add action**.

> **Note**
>
> This procedure shows you how to add the build action inside of a build stage. To add the build action somewhere else, choose **Add action** in the desired place. You might first need to choose **Edit stage** in the existing stage where you want to add the build action.

10. In **Edit action**, for **Action name**, enter a name for the action (for example, **`CodeBuild`**). If you choose a different name, use it throughout this procedure.
11. For **Action provider**, choose **CodeBuild**.

12. If you already have a build project you want to use, for **Project name**, choose the name of the build project and skip to the next step in this procedure.

    If you need to create a new CodeBuild build project, follow the instructions in Create a build project (console) and return to this procedure.

    If you choose an existing build project, it must have build output artifact settings already defined (even though CodePipeline overrides them). For more information, see the description of **Artifacts** in Create a build project (console) or Change a build project's settings (console).

    > **Important**
    >
    > If you enable webhooks for a CodeBuild project, and the project is used as a build step in CodePipeline, then two identical builds are created for each commit. One build is triggered through webhooks and one through CodePipeline. Because billing is on a per-build basis, you are billed for both builds. Therefore, if you are using CodePipeline, we recommend that you disable webhooks in CodeBuild. In the CodeBuild console, clear the **Webhook** box. For more information, see Change a build project's settings (console)

13. For **Input artifacts**, choose the output artifact that you noted earlier in this procedure.
14. For **Output artifacts**, enter a name for the output artifact (for example, **`MyAppBuild`**).
15. Choose **Add action**.
16. Choose **Save**, and then choose **Save** to save your changes to the pipeline.
17. Choose **Release change**.
18. After the pipeline runs successfully, you can get the build output artifact. With the pipeline displayed in the CodePipeline console, in the **Build** action, choose the tooltip. Make a note of the value for **Output artifact** (for example, **MyAppBuild**).

    > **Note**
    >
    > You can also get the build output artifact by choosing the **Build artifacts** link on the build details page in the CodeBuild console. To get to this page, see View build details (console), and then skip to step 31 of this procedure.

19. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

20. In the list of buckets, open the bucket used by the pipeline. The name of the bucket should follow the format `codepipeline-`*`region-ID`*`-`*`random-number`*. You can use the AWS CLI to run the CodePipeline **get-pipeline** command to get the name of the bucket:

    ```
    aws codepipeline get-pipeline --name my-pipeline-name
    ```

    In the output, the `pipeline` object contains an `artifactStore` object, which contains a `location` value with the name of the bucket.

21. Open the folder that matches the name of your pipeline (depending on the length of the pipeline's name, the folder name might be truncated), and then open the folder matching the value for **Output artifact** that you noted earlier in this procedure.

22. Extract the contents of the file. If there are multiple files in that folder, extract the contents of the file with the latest **Last Modified** timestamp. (You might need to give the file the `.zip` extension so that you can work with it in your system's ZIP utility.) The build output artifact is in the extracted contents of the file.

23. If you instructed CodePipeline to deploy the build output artifact, use the deployment provider's instructions to get to the build output artifact on the deployment targets.

Add a CodeBuild test action to a pipeline (CodePipeline console)

1. Sign in to the AWS Management Console by using:

    - Your AWS root account. This is not recommended. For more information, see The account root user in the *user Guide*.
    - An administrator user in your AWS account. For more information, see Creating your first AWS account root user and group in the *user Guide*.
    - An user in your AWS account with permission to perform the following minimum set of actions:

    ```
    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
```

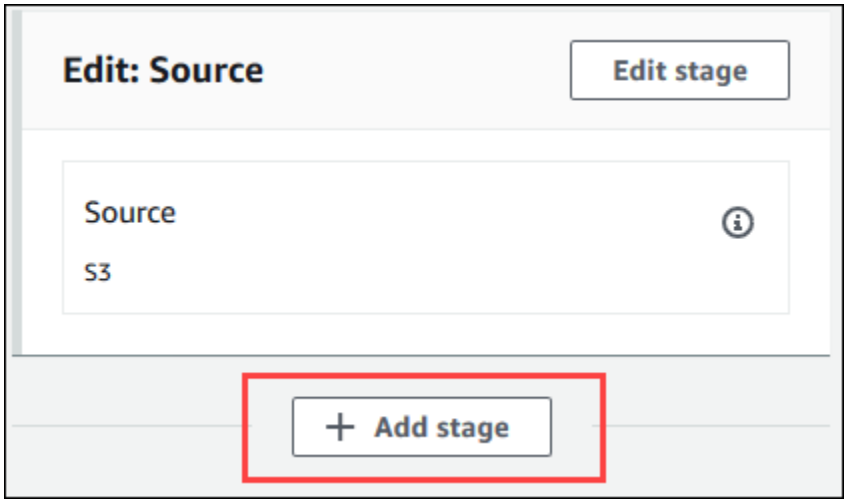
2. Open the CodePipeline console at https://console.aws.amazon.com/codesuite/codepipeline/home.
3. In the AWS region selector, choose the AWS Region where your pipeline is located. This must be an AWS Region where CodeBuild is supported. For more information, see AWS CodeBuild in the *Amazon Web Services General Reference*.
4. On the **Pipelines** page, choose the name of the pipeline.
5. On the pipeline details page, in the **Source** action, choose the tooltip. Make a note of the value for **Output artifact** (for example, **MyApp**).

> **Note**
>
> This procedure shows you how to add a test action inside of a test stage between the **Source** and **Beta** stages. If you want to add the test action somewhere else, rest your mouse pointer on the action just before, and make a note of the value for **Output artifact**.

6. Choose **Edit**.
7. Immediately after the **Source** stage, choose **Add stage**.

> **Note**
>
> This procedure shows you how to add a test stage immediately after the **Source** stage to your pipeline. To add a test action to an existing stage, choose **Edit stage** in the

stage, and then skip to step 8 of this procedure. To add the test stage somewhere else, choose **Add stage** in the desired place.



8. For **Stage name**, enter the name of the test stage (for example, **`Test`**). If you choose a different name, use it throughout this procedure.
9. In the selected stage, choose **Add action**.

> **Note**
>
> This procedure shows you how to add the test action in a test stage. To add the test action somewhere else, choose **Add action** in the desired place. You might first need to choose **Edit** in the existing stage where you want to add the test action.

10. In **Edit action**, for **Action name**, enter a name for the action (for example, **`Test`**). If you choose a different name, use it throughout this procedure.
11. For **Action provider**, under **Test**, choose **CodeBuild**.
12. If you already have a build project you want to use, for **Project name**, choose the name of the build project and skip to the next step in this procedure.

    If you need to create a new CodeBuild build project, follow the instructions in Create a build project (console) and return to this procedure.

> **Important**
>
> If you enable webhooks for a CodeBuild project, and the project is used as a build step in CodePipeline, then two identical builds are created for each commit. One build is triggered through webhooks and one through CodePipeline. Because billing is on a per-build basis, you are billed for both builds. Therefore, if you are using CodePipeline, we recommend that you disable webhooks in CodeBuild. In the CodeBuild console, clear the **Webhook**box. For more information, see Change a build project's settings (console)

13. For **Input artifacts**, select the value for **Output artifact** that you noted earlier in this procedure.
14. (Optional) If you want your test action to produce an output artifact, and you set up your buildspec accordingly, then for **Output artifact**, enter the value you want to assign to the output artifact.
15. Choose **Save**.
16. Choose **Release change**.
17. After the pipeline runs successfully, you can get the test results. In the **Test** stage of the pipeline, choose the **CodeBuild** hyperlink to open the related build project page in the CodeBuild console.
18. On the build project page, in **Build history**, choose the **Build run** hyperlink.
19. On the build run page, in **Build logs**, choose the **View entire log** hyperlink to open the build log in the Amazon CloudWatch console.
20. Scroll through the build log to view the test results.

Use AWS CodeBuild with Codecov

Codecov is a tool that measures the test coverage of your code. Codecov identifies which methods and statements in your code are not tested. Use the results to determine where to write tests to improve the quality of your code. Codecov is available for three of the source repositories supported by CodeBuild: GitHub, GitHub Enterprise Server, and Bitbucket. If your build project uses GitHub Enterprise Server, you must use Codecov Enterprise.

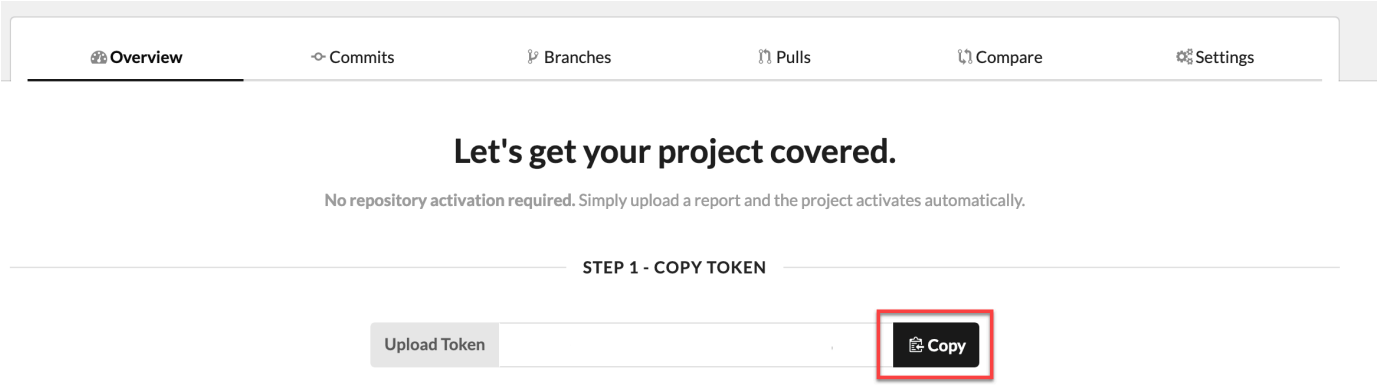
When you run a build of a CodeBuild project that is integrated with Codecov, Codecov reports that analyzes code in your repository are uploaded to Codecov. The build logs include a link to the reports. This sample shows you how to integrate a Python and a Java build project with Codecov. For a list of languages supported by Codecov, see Codecov supported languages on the Codecov website.

Integrate Codecov into a build project

Use the following procedure to integration Codecov into a build project.

To integrate Codecov with your build project

1. Go to https://codecov.io/signup and sign up for a GitHub or Bitbucket source repository. If you use GitHub Enterprise, see Codecov Enterprise on the Codecov website.
2. In Codecov, add the repository for which you want coverage.
3. When token information is displayed, choose **Copy**.



4. Add the copied token as an environment variable named `CODECOV_TOKEN` to your build project. For more information, see Change a build project's settings (console).
5. Create a text file named `my_script.sh` in your repository. Enter the following into the file:

```
#/bin/bash
bash <(curl -s https://codecov.io/bash) -t $CODECOV_TOKEN
```

6. Choose the **Python** or **Java** tab, as appropriate for your build project uses, and follow these steps.

   Java

   1. Add the following JaCoCo plugin to `pom.xml` in your repository.

```
<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. Enter the following commands in your buildspec file. For more information, see Buildspec syntax.

```
build:
  - mvn test -f pom.xml -fn
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

Python

Enter the following commands in your buildspec file. For more information, see Buildspec syntax.

```
build:
  - pip install coverage
```

```
  - coverage run -m unittest discover
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

7. Run a build of your build project. A link to Codecov reports generated for your project appears in your build logs. Use the link to view the Codecov reports. For more information, see Run AWS CodeBuild builds manually and Log AWS CodeBuild API calls with AWS CloudTrail. Codecov information in the build logs looks like the following:

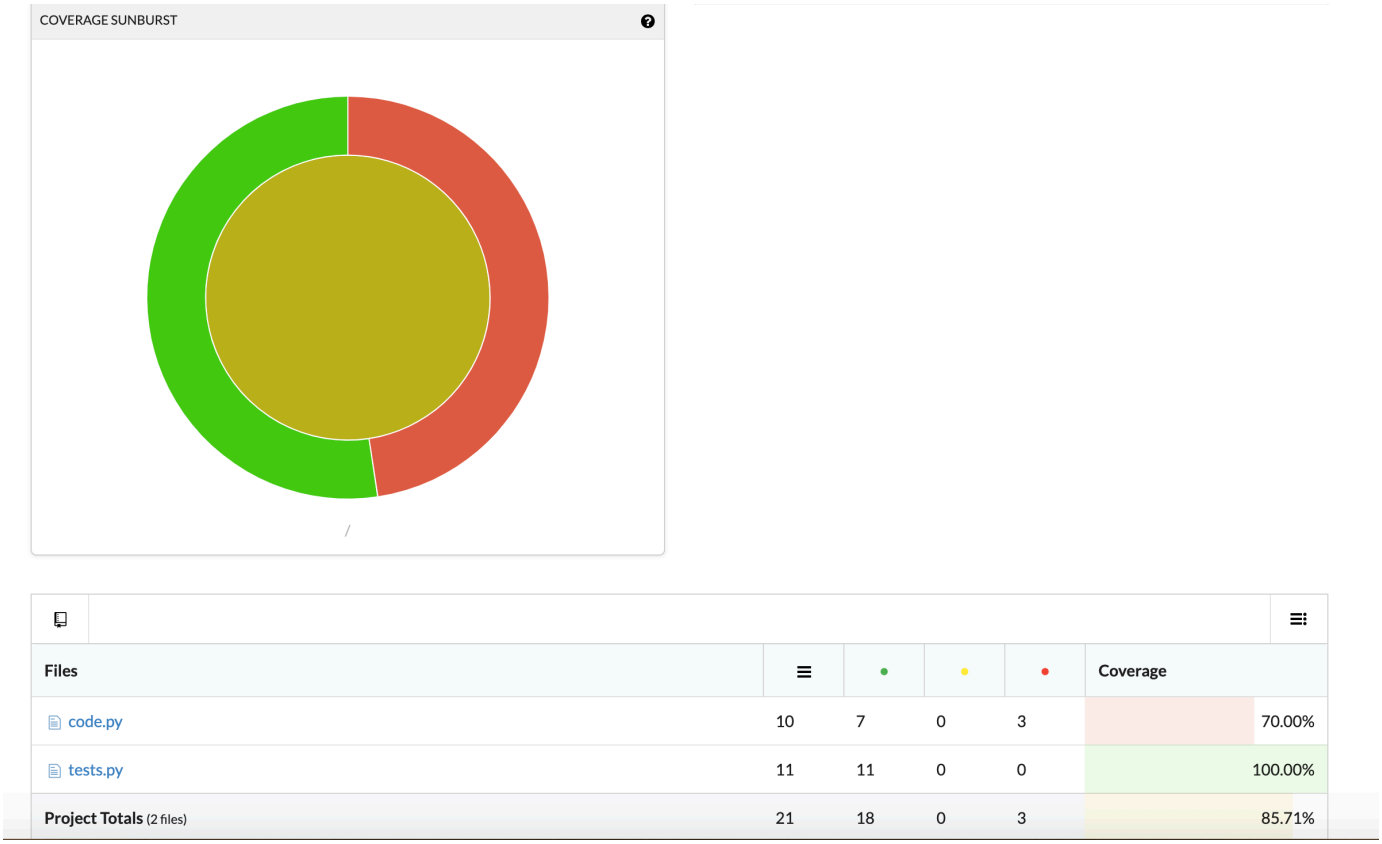
```
[Container] 2020/03/09 16:31:04 Running command bash my_script.sh


  _____          _
 / ____|        | |
| |     ___   __| | ___  ___ _____   __
| |    / _ \ / _` |/ _ \/ __/ _ \ \ / /
| |___| (_) | (_| |  __/ (_| (_) \ 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
```

The reports look like the following:



Use AWS CodeBuild with Jenkins

You can use the Jenkins plugin for AWS CodeBuild to integrate CodeBuild with your Jenkins build jobs. Instead of sending your build jobs to Jenkins build nodes, you use the plugin to send your build jobs to CodeBuild. This eliminates the need for you to provision, configure, and manage Jenkins build nodes.

Topics

Set up Jenkins

For information about setting up Jenkins with the AWS CodeBuild plugin, and to download the plugin source code, see https://github.com/awslabs/aws-codebuild-jenkins-plugin.

Install the plugin

If you already have a Jenkins server set up and would like to only install the AWS CodeBuild plugin, on your Jenkins instance, in the Plugin Manager, search for **`CodeBuild Plugin for Jenkins`**.

Use the plugin

To use AWS CodeBuild with sources from outside of a VPC

1. Create a project in the CodeBuild console. For more information, see Create a build project (console).
   - Choose the AWS Region where you want to run the build.
   - (Optional) Set the Amazon VPC configuration to allow the CodeBuild build container to access resources in your VPC.
   - Write down the name of your project. You need it in step 3.
   - (Optional) If your source repository is not natively supported by CodeBuild, you can set Amazon S3 as the input source type for your project.
2. In the IAMconsole, create an user to be used by the Jenkins plugin.
   - When you create credentials for the user, choose **Programmatic Access**.
   - Create a policy similar to the following and then attach the policy to your user.

```
{
  "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. Create a freestyle project in Jenkins.

   - On the **Configure** page, choose **Add build step**, and then choose **Run build on CodeBuild**.
   - Configure your build step.
     - Provide values for **Region**, **Credentials**, and **Project Name**.
     - Choose **Use Project source**.
     - Save the configuration and run a build from Jenkins.

4. For **Source Code Management**, choose how you want to retrieve your source. You might need to install the GitHub plugin (or the Jenkins plugin for your source repository provider) on your Jenkins server.

   - On the **Configure** page, choose **Add build step**, and then choose **Run build on AWS CodeBuild**.
   - Configure your build step.
     - Provide values for **Region**, **Credentials**, and **Project Name**.
     - Choose **Use Jenkins source**.
     - Save the configuration and run a build from Jenkins.

To use the AWS CodeBuild plugin with the Jenkins pipeline plugin

- On your Jenkins pipeline project page, use the snippet generator to generate a pipeline script that adds CodeBuild as a step in your pipeline. It should generate a script similar to this:

```
awsCodeBuild projectName: 'project', credentialsType: 'keys', region: 'us-west-2',
 sourceControlType: 'jenkins'
```

Use AWS CodeBuild with serverless applications

The AWS Serverless Application Model (AWS SAM) is an open-source framework for building serverless applications. For more information, see the AWS serverless application model repository on GitHub.

You can use AWS CodeBuild to package and deploy serverless applications that follow the AWS SAM standard. For the deployment step, CodeBuild can use AWS CloudFormation. To automate the building and deployment of serverless applications with CodeBuild and AWS CloudFormation, you can use AWS CodePipeline.

For more information, see Deploying Serverless Applications in the *AWS Serverless Application Model Developer Guide*.

Related resources

- For information about getting started with AWS CodeBuild, see Getting started with AWS CodeBuild using the console.
- For information about troubleshooting issues in CodeBuild, see Troubleshooting AWS CodeBuild.
- For information about quotas in CodeBuild, see Quotas for AWS CodeBuild.

Third party notices for AWS CodeBuild for Windows

When you use CodeBuild for Windows builds, you have the option to use some third party packages and modules to enable your built application to run on Microsoft Windows operating systems and to interoperate with some third party products. The following list contains the applicable third-party legal terms that govern your use of the specified third-party packages and modules.

Topics

1) base Docker image—windowsservercore

(license terms available at: https://hub.docker.com/_/microsoft-windows-servercore)

License: By requesting and using this Container OS Image for Windows containers, you acknowledge, understand, and consent to the following Supplemental License Terms:

MICROSOFT SOFTWARE SUPPLEMENTAL LICENSE TERMS

CONTAINER OS IMAGE

Microsoft Corporation (or based on where you live, one of its affiliates) (referenced as "us," "we," or "Microsoft") licenses this Container OS Image supplement to you ("Supplement"). You are licensed to use this Supplement in conjunction with the underlying host operating system software ("Host Software") solely to assist running the containers feature in the Host Software. The Host Software license terms apply to your use of the Supplement. You may not use it if you do not have a license for the Host Software. You may use this Supplement with each validly licensed copy of the Host Software.

ADDITIONAL LICENSING REQUIREMENTS AND/OR USE RIGHTS

Your use of the Supplement as specified in the preceding paragraph may result in the creation or modification of a container image ("Container Image") that includes certain Supplement components. For clarity, a Container Image is separate and distinct from a virtual machine or

virtual appliance image. Pursuant to these license terms, we grant you a restricted right to redistribute such Supplement components under the following conditions:

(i) you may use the Supplement components only as used in, and as a part of your Container Image,

(ii) you may use such Supplement components in your Container Image as long as you have significant primary functionality in your Container Image that is materially separate and distinct from the Supplement; and

(iii) you agree to include these license terms (or similar terms required by us or a hoster) with your Container Image to properly license the possible use of the Supplement components by your end-users.

We reserve all other rights not expressly granted herein.

By using this Supplement, you accept these terms. If you do not accept them, do not use this Supplement.

As part of the Supplemental License Terms for this Container OS Image for Windows containers, you are also subject to the underlying Windows Server host software license terms, which are located at: https://www.microsoft.com/en-us/useterms.

2) windows-base Docker image—choco

(license terms available at: https://github.com/chocolatey/choco/blob/master/LICENSE)

Copyright 2011 - Present RealDimensions Software, LLC

Licensed under the Apache License, version 2.0 (the "License"); you may not use these files except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or as agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

3) windows-base Docker image—git --version 2.16.2

(license terms available at: https://chocolatey.org/packages/git/2.16.2)

Licensed under GNU General Public License, version 2, available at: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

4) windows-base Docker image—microsoft-build-tools --version 15.0.26320.2

(license terms available at: https://www.visualstudio.com/license-terms/mt171552/)

MICROSOFT VISUAL STUDIO 2015 EXTENSIONS, VISUAL STUDIO SHELLS and C++ REDISTRIBUTABLE

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. They apply to the software named above. The terms also apply to any Microsoft services or updates for the software, except to the extent those have additional terms.

IF YOU COMPLY WITH THESE LICENSE TERMS, YOU HAVE THE RIGHTS BELOW.

1. **INSTALLATION AND USE RIGHTS**. You may install and use any number of copies of the software.
2. **TERMS FOR SPECIFIC COMPONENTS**.
   a. **Utilities**. The software may contain some items on the Utilities List at https://docs.microsoft.com/en-us/visualstudio/productinfo/2015-redistribution-vs. You may copy and install those items, if included with the software, on to yours or other third party machines, to debug and deploy your applications and databases you developed with the software. Please note that Utilities are designed for temporary use, that Microsoft may not be able to patch or update Utilities separately from the rest of the software, and that some Utilities by their nature may make it possible for others to access machines on which they are installed. As a result, you should delete all Utilities you have installed after you finish debugging or deploying your applications and databases. Microsoft is not responsible for any third party use or access of Utilities you install on any machine.
   b. **Microsoft Platforms**. The software may include components from Microsoft Windows; Microsoft Windows Server; Microsoft SQL Server; Microsoft Exchange; Microsoft Office; and Microsoft SharePoint. These components are governed by separate agreements and their own product support policies, as described in the license terms found in the installation directory for that component or in the "Licenses" folder accompanying the software.

c. **Third Party Components**. The software may include third party components with separate legal notices or governed by other agreements, as described in the ThirdPartyNotices file accompanying the software. Even if such components are governed by other agreements, the disclaimers and the limitations on and exclusions of damages below also apply. The software may also include components licensed under open source licenses with source code availability obligations. Copies of those licenses, if applicable, are included in the ThirdPartyNotices file. You may obtain this source code from us, if and as required under the relevant open source licenses, by sending a money order or check for $5.00 to: Source Code Compliance Team, Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052. Please write source code for one or more of the components listed below in the memo line of your payment:

- 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; or
- Visual Studio 2015 Isolated Shell.

We may also make a copy of the source code available at http://thirdpartysource.microsoft.com.

3. **DATA**. The software may collect information about you and your use of the software, and send that to Microsoft. Microsoft may use this information to provide services and improve our products and services. You may opt-out of many of these scenarios, but not all, as described in the product documentation. There are also some features in the software that may enable you to collect data from users of your applications. If you use these features to enable data collection in your applications, you must comply with applicable law, including providing appropriate notices to users of your applications. You can learn more about data collection and use in the help documentation and the privacy statement at https://privacy.microsoft.com/en-us/privacystatement. Your use of the software operates as your consent to these practices.
4. **SCOPE OF LICENSE**. The software is licensed, not sold. This agreement only gives you some rights to use the software. Microsoft reserves all other rights. Unless applicable law gives you

more rights despite this limitation, you may use the software only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the software that only allow you to use it in certain ways. You may not

- work around any technical limitations in the software;
- reverse engineer, decompile or disassemble the software, or attempt to do so, except and only to the extent required by third party licensing terms governing the use of certain open-source components that may be included with the software;
- remove, minimize, block or modify any notices of Microsoft or its suppliers in the software;
- use the software in any way that is against the law; or
- share, publish, rent or lease the software, or provide the software as a stand-alone hosted as solution for others to use.

5. **EXPORT RESTRICTIONS**. You must comply with all domestic and international export laws and regulations that apply to the software, which include restrictions on destinations, end users, and end use. For further information on export restrictions, visit (aka.ms/exporting).
6. **SUPPORT SERVICES**. Because this software is "as is," we may not provide support services for it.
7. **ENTIRE AGREEMENT**. This agreement, and the terms for supplements, updates, Internet-based services and support services that you use, are the entire agreement for the software and support services.
8. **APPLICABLE LAW**. If you acquired the software in the United States, Washington law applies to interpretation of and claims for breach of this agreement, and the laws of the state where you live apply to all other claims. If you acquired the software in any other country, its laws apply.
9. **CONSUMER RIGHTS; REGIONAL VARIATIONS**. This agreement describes certain legal rights. You may have other rights, including consumer rights, under the laws of your state or country. Separate and apart from your relationship with Microsoft, you may also have rights with respect to the party from which you acquired the software. This agreement does not change those other rights if the laws of your state or country do not permit it to do so. For example, if you acquired the software in one of the below regions, or mandatory country law applies, then the following provisions apply to you:

    a. **Australia**. You have statutory guarantees under the Australian Consumer Law and nothing in this agreement is intended to affect those rights.

    b. **Canada**. If you acquired this software in Canada, you may stop receiving updates by turning off the automatic update feature, disconnecting your device from the Internet (if and when you re-connect to the Internet, however, the software will resume checking for and installing

updates), or uninstalling the software. The product documentation, if any, may also specify how to turn off updates for your specific device or software.

c. **Germany and Austria**.

i. **Warranty**. The properly licensed software will perform substantially as described in any Microsoft materials that accompany the software. However, Microsoft gives no contractual guarantee in relation to the licensed software.

ii. **Limitation of Liability**. In case of intentional conduct, gross negligence, claims based on the Product Liability Act, as well as, in case of death or personal or physical injury, Microsoft is liable according to the statutory law.Subject to the foregoing clause (ii), Microsoft will only be liable for slight negligence if Microsoft is in breach of such material contractual obligations, the fulfillment of which facilitate the due performance of this agreement, the breach of which would endanger the purpose of this agreement and the compliance with which a party may constantly trust in (so-called "cardinal obligations"). In other cases of slight negligence, Microsoft will not be liable for slight negligence.

10 **DISCLAIMER OF WARRANTY. THE SOFTWARE IS LICENSED "AS-IS." YOU BEAR THE RISK OF USING IT. MICROSOFT GIVES NO EXPRESS WARRANTIES, GUARANTEES OR CONDITIONS. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT EXCLUDES THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**

11 **LIMITATION ON AND EXCLUSION OF DAMAGES. YOU CAN RECOVER FROM MICROSOFT AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO U.S. $5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.** This limitation applies to (a) anything related to the software, services, content (including code) on third party Internet sites, or third party applications; and (b) claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

EULA ID: VS2015_Update3_ShellsRedist_<ENU>

5) windows-base Docker image—nuget.commandline --version 4.5.1

(license terms available at: https://github.com/NuGet/Home/blob/dev/LICENSE.txt)

7) windows-base Docker image—netfx-4.6.2-devpack

MICROSOFT SOFTWARE SUPPLEMENTAL LICENSE TERMS

.NET FRAMEWORK AND ASSOCIATED LANGUAGE PACKS FOR MICROSOFT WINDOWS OPERATING SYSTEM

Microsoft Corporation (or based on where you live, one of its affiliates) licenses this supplement to you. If you are licensed to use Microsoft Windows operating system software (the "software"), you may use this supplement. You may not use it if you do not have a license for the software. You may use this supplement with each validly licensed copy of the software.

The following license terms describe additional use terms for this supplement. These terms and the license terms for the software apply to your use of the supplement. If there is a conflict, these supplemental license terms apply.

BY USING THIS SUPPLEMENT, YOU ACCEPT THESE TERMS. IF YOU DO NOT ACCEPT THEM, DO NOT USE THIS SUPPLEMENT.

If you comply with these license terms, you have the rights below.

1. **DISTRIBUTABLE CODE.** The supplement is comprised of Distributable Code. "Distributable Code" is code that you are permitted to distribute in programs you develop if you comply with the terms below.
   a. **Right to Use and Distribute**.

- You may copy and distribute the object code form of the supplement.
- Third Party Distribution. You may permit distributors of your programs to copy and distribute the Distributable Code as part of those programs.

b. **Distribution Requirements. For any Distributable Code you distribute, you must**

- add significant primary functionality to it in your programs;
- for any Distributable Code having a filename extension of .lib, distribute only the results of running such Distributable Code through a linker with your program;
- distribute Distributable Code included in a setup program only as part of that setup program without modification;
- require distributors and external end users to agree to terms that protect it at least as much as this agreement;
- display your valid copyright notice on your programs; and
- indemnify, defend, and hold harmless Microsoft from any claims, including attorneys' fees, related to the distribution or use of your programs.

c. **Distribution Restrictions. You may not**

- alter any copyright, trademark or patent notice in the Distributable Code;
- use Microsoft's trademarks in your programs' names or in a way that suggests your programs come from or are endorsed by Microsoft;
- distribute Distributable Code to run on a platform other than the Windows platform;
- include Distributable Code in malicious, deceptive or unlawful programs; or
- modify or distribute the source code of any Distributable Code so that any part of it becomes subject to an Excluded License. An Excluded License is one that requires, as a condition of use, modification or distribution, that
  - the code be disclosed or distributed in source code form; or
  - others have the right to modify it.

2. **SUPPORT SERVICES FOR SUPPLEMENT**. Microsoft provides support services for this software as described at www.support.microsoft.com/common/international.aspx.

8) windows-base Docker image—visualfsharptools, v 4.0

(license terms available at: https://github.com/dotnet/fsharp/blob/main/License.txt)

Licensed under the Apache License, version 2.0 (the "License"); you may not use these files except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or as agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

9) windows-base Docker image—netfx-pcl-reference-assemblies-4.6

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT .NET PORTABLE CLASS LIBRARY REFERENCE ASSEMBLIES – 4.6

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above. The terms also apply to any Microsoft

- updates,
- supplements,
- Internet-based services, and
- support services

for this software, unless other terms accompany those items. If so, those terms apply.

BY USING THE SOFTWARE, YOU ACCEPT THESE TERMS. IF YOU DO NOT ACCEPT THEM, DO NOT USE THE SOFTWARE.

IF YOU COMPLY WITH THESE LICENSE TERMS, YOU HAVE THE PERPETUAL RIGHTS BELOW.

1. **INSTALLATION AND USE RIGHTS**. You may install and use any number of copies of the software to design, develop and test your programs.
2. **ADDITIONAL LICENSING REQUIREMENTS AND/OR USE RIGHTS**.

a. **Distributable Code**. You may distribute the software in developer tool programs you develop, to enable customers of your programs to develop portable libraries for use with any device or operating system, if you comply with the terms below.

i. **Right to Use and Distribute. The software is "Distributable Code."**

- Distributable Code. You may copy and distribute the object code form of the software.
- Third Party Distribution. You may permit distributors of your programs to copy and distribute the Distributable Code as part of those programs.

ii. **Distribution Requirements. For any Distributable Code you distribute, you must**

- add significant primary functionality to it in your programs;
- require distributors and your customers to agree to terms that protect it at least as much as this agreement;
- display your valid copyright notice on your programs; and
- indemnify, defend, and hold harmless Microsoft from any claims, including attorneys' fees, related to the distribution or use of your programs.

iii. **Distribution Restrictions. You may not**

- alter any copyright, trademark or patent notice in the Distributable Code;
- use Microsoft's trademarks in your programs' names or in a way that suggests your programs come from or are endorsed by Microsoft;
- include Distributable Code in malicious, deceptive or unlawful programs; or
- modify or distribute the Distributable Code so that any part of it becomes subject to an Excluded License. An Excluded License is one that requires, as a condition of use, modification or distribution, that
  - the code be disclosed or distributed in source code form; or
  - others have the right to modify it.

3. **SCOPE OF LICENSE**. The software is licensed, not sold. This agreement only gives you some rights to use the software. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the software only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the software that only allow you to use it in certain ways. You may not

- work around any technical limitations in the software;
- reverse engineer, decompile or disassemble the software, except and only to the extent that applicable law expressly permits, despite this limitation;

- publish the software for others to copy; or
- rent, lease or lend the software.

4. **FEEDBACK**. You may provide feedback about the software. If you give feedback about the software to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.
5. **TRANSFER TO A THIRD PARTY**. The first user of the software may transfer it, and this agreement, directly to a third party. Before the transfer, that party must agree that this agreement applies to the transfer and use of the software. The first user must uninstall the software before transferring it separately from the device. The first user may not retain any copies.
6. **EXPORT RESTRICTIONS**. The software is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the software. These laws include restrictions on destinations, end users and end use. For additional information, see www.microsoft.com/exporting.
7. **SUPPORT SERVICES**. Because this software is "as is," we may not provide support services for it.
8. **ENTIRE AGREEMENT**. This agreement, and the terms for supplements, updates, Internet-based services and support services that you use, are the entire agreement for the software and any support services we provide.
9. **APPLICABLE LAW**.
   a. **United States**. If you acquired the software in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.
   b. **Outside the United States**. If you acquired the software in any other country, the laws of that country apply.
10. **LEGAL EFFECT**. This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the software. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.

11 **DISCLAIMER OF WARRANTY. THE SOFTWARE IS LICENSED "AS-IS." YOU BEAR THE RISK OF USING IT. MICROSOFT GIVES NO EXPRESS WARRANTIES, GUARANTEES OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS OR STATUTORY GUARANTEES UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT EXCLUDES THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**

FOR AUSTRALIA—YOU HAVE STATUTORY GUARANTEES UNDER THE AUSTRALIAN CONSUMER LAW AND NOTHING IN THESE TERMS IS INTENDED TO AFFECT THOSE RIGHTS.

12 **LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM MICROSOFT AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO U.S. $5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.**

This limitation applies to

- anything related to the software, services, content (including code) on third party Internet sites, or third party programs; and
- claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

10) windows-base Docker image—visualcppbuildtools v 14.0.25420.1

(license terms available at: https://www.visualstudio.com/license-terms/mt644918/)

MICROSOFT VISUAL C++ BUILD TOOLS

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT VISUAL C++ BUILD TOOLS

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. They apply to the software named above. The terms also apply to any Microsoft services or updates for the software, except to the extent those have different terms.

IF YOU COMPLY WITH THESE LICENSE TERMS, YOU HAVE THE RIGHTS BELOW.

1. **INSTALLATION AND USE RIGHTS**.
   a. One user may use copies of the software to develop and test their applications.
2. **DATA**. The software may collect information about you and your use of the software, and send that to Microsoft. Microsoft may use this information to provide services and improve our products and services. You may opt-out of many of these scenarios, but not all, as described in the product documentation. There are also some features in the software that may enable you to collect data from users of your applications. If you use these features to enable data collection in your applications, you must comply with applicable law, including providing appropriate notices to users of your applications. You can learn more about data collection and use in the help documentation and the privacy statement at http://go.microsoft.com/fwlink/?LinkID=528096. Your use of the software operates as your consent to these practices.
3. **TERMS FOR SPECIFIC COMPONENTS**.
   a. **Build Server**. The software may contain some Build Server components listed in BuildServer.TXT files, and/or any files listed on the BuildeServer list located following this Microsoft Software License Terms. You may copy and install those items, if included in the software, onto your build machines. You and others in your organization may use these items on your build machines solely for the purpose of compiling, building, verifying and archiving your applications or running quality or performance tests as part of the build process.
   b. **Microsoft Platforms**. The software may include components from Microsoft Windows; Microsoft Windows Server; Microsoft SQL Server; Microsoft Exchange; Microsoft Office; and Microsoft SharePoint. These components are governed by separate agreements and their own product support policies, as described in the license terms found in the installation directory for that component or in the "Licenses" folder accompanying the software.
   c. **Third Party Components**. The software may include third party components with separate legal notices or governed by other agreements, as described in the ThirdPartyNotices file accompanying the software. Even if such components are governed by other agreements, the disclaimers and the limitations on and exclusions of damages below also apply.
   d. **Package Managers**. The software may include package managers, like Nuget, that give you the option to download other Microsoft and third party software packages to use with your application. Those packages are under their own licenses, and not this agreement. Microsoft does not distribute, license or provide any warranties for any of the third party packages.

4. **SCOPE OF LICENSE**. The software is licensed, not sold. This agreement only gives you some rights to use the software. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the software only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the software that only allow you to use it in certain ways. For more information, see https://docs.microsoft.com/en-us/legal/information-protection/software-license-terms#1-installation-and-use-rights. You may not
   - work around any technical limitations in the software;
   - reverse engineer, decompile or disassemble the software, or attempt to do so, except and only to the extent required by third party licensing terms governing use of certain open source components that may be included with the software;
   - remove, minimize, block or modify any notices of Microsoft or its suppliers;
   - use the software in any way that is against the law; or
   - share, publish, rent or lease the software, or provide the software as a stand-alone hosted as solution for others to use.
5. **EXPORT RESTRICTIONS**. You must comply with all domestic and international export laws and regulations that apply to the software, which include restrictions on destinations, end users and end use. For further information on export restrictions, visit (aka.ms/exporting).
6. **SUPPORT SERVICES**. Because this software is "as is," we may not provide support services for it.
7. **ENTIRE AGREEMENT**. This agreement, and the terms for supplements, updates, Internet-based services and support services that you use, are the entire agreement for the software and support services.
8. **APPLICABLE LAW**. If you acquired the software in the United States, Washington law applies to interpretation of and claims for breach of this agreement, and the laws of the state where you live apply to all other claims. If you acquired the software in any other country, its laws apply.
9. **CONSUMER RIGHTS; REGIONAL VARIATIONS**. This agreement describes certain legal rights. You may have other rights, including consumer rights, under the laws of your state or country. Separate and apart from your relationship with Microsoft, you may also have rights with respect to the party from which you acquired the software. This agreement does not change those other rights if the laws of your state or country do not permit it to do so. For example, if you acquired the software in one of the below regions, or mandatory country law applies, then the following provisions apply to you:
   - **Australia**. You have statutory guarantees under the Australian Consumer Law and nothing in this agreement is intended to affect those rights.

- **Canada**. If you acquired this software in Canada, you may stop receiving updates by turning off the automatic update feature, disconnecting your device from the Internet (if and when you re-connect to the Internet, however, the software will resume checking for and installing updates), or uninstalling the software. The product documentation, if any, may also specify how to turn off updates for your specific device or software.
- **Germany and Austria**.
  - **Warranty**. The properly licensed software will perform substantially as described in any Microsoft materials that accompany the software. However, Microsoft gives no contractual guarantee in relation to the licensed software.
  - *Limitation of Liability*. In case of intentional conduct, gross negligence, claims based on the Product Liability Act, as well as, in case of death or personal or physical injury, Microsoft is liable according to the statutory law.

    Subject to the foregoing clause (ii), Microsoft will only be liable for slight negligence if Microsoft is in breach of such material contractual obligations, the fulfillment of which facilitate the due performance of this agreement, the breach of which would endanger the purpose of this agreement and the compliance with which a party may constantly trust in (so-called "cardinal obligations"). In other cases of slight negligence, Microsoft will not be liable for slight negligence.

10 **LEGAL EFFECT**. This agreement describes certain legal rights. You may have other rights under the laws of your state or country. This agreement does not change your rights under the laws of your state or country if the laws of your state or country do not permit it to do so. Without limitation of the foregoing, for Australia, **YOU HAVE STATUTORY GUARANTEES UNDER THE AUSTRALIAN CONSUMER LAW AND NOTHING IN THESE TERMS IS INTENDED TO AFFECT THOSE RIGHTS**

11 **DISCLAIMER OF WARRANTY. THE SOFTWARE IS LICENSED "AS-IS." YOU BEAR THE RISK OF USING IT. MICROSOFT GIVES NO EXPRESS WARRANTIES, GUARANTEES OR CONDITIONS. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT EXCLUDES THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**

12 **LIMITATION ON AND EXCLUSION OF DAMAGES. YOU CAN RECOVER FROM MICROSOFT AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO U.S. $5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.**

This limitation applies to (a) anything related to the software, services, content (including code) on third party Internet sites, or third party applications; and (b) claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

11) windows-base Docker image—microsoft-windows-netfx3-ondemand-package.cab

MICROSOFT SOFTWARE SUPPLEMENTAL LICENSE TERMS

MICROSOFT .NET FRAMEWORK 3.5 SP1 FOR MICROSOFT WINDOWS OPERATING SYSTEM

Microsoft Corporation (or based on where you live, one of its affiliates) licenses this supplement to you. If you are licensed to use Microsoft Windows operating system software (for which this supplement is applicable) (the "software"), you may use this supplement. You may not use it if you do not have a license for the software. You may use a copy of this supplement with each validly licensed copy of the software.

The following license terms describe additional use terms for this supplement. These terms and the license terms for the software apply to your use of the supplement. If there is a conflict, these supplemental license terms apply.

BY USING THIS SUPPLEMENT, YOU ACCEPT THESE TERMS. IF YOU DO NOT ACCEPT THEM, DO NOT USE THIS SUPPLEMENT.

If you comply with these license terms, you have the rights below.

1. **SUPPORT SERVICES FOR SUPPLEMENT**. Microsoft provides support services for this software as described at www.support.microsoft.com/common/international.aspx.
2. **MICROSOFT .NET BENCHMARK TESTING**. The software includes the .NET Framework, Windows Communication Foundation, Windows Presentation Foundation, and Windows Workflow

Foundation components of the Windows operating systems (.NET Components). You may conduct internal benchmark testing of the .NET Components. You may disclose the results of any benchmark test of the .NET Components, provided that you comply with the conditions set forth at http://go.microsoft.com/fwlink/?LinkID=66406.

Notwithstanding any other agreement you may have with Microsoft, if you disclose such benchmark test results, Microsoft shall have the right to disclose the results of benchmark tests it conducts of your products that compete with the applicable .NET Component, provided it complies with the same conditions set forth at http://go.microsoft.com/fwlink/?LinkID=66406.

12) windows-base Docker image—dotnet-sdk

(available at https://github.com/dotnet/core/blob/main/LICENSE.TXT)

The MIT License (MIT)

Copyright (c) Microsoft Corporation

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Use CodeBuild condition keys as IAM service role variables to control build access

With the CodeBuild build ARN, you can restrict build resource access by using context keys to scope down resource access in your CodeBuild service role. For CodeBuild, the keys that can be used to

control build access behavior are `codebuild:buildArn` and `codebuild:projectArn`. With the build project ARN, you can verify whether a call to your resource came from a specific build project. To verify this, use the `codebuild:buildArn` or `codebuild:projectArn` condition keys in an IAM identity-based policy.

To use the `codebuild:buildArn` or `codebuild:projectArn` condition keys in your policy, include it as a condition with any of the ARN condition operators. The value of the key must be an IAM variable that resolves to a valid ARN. In the example policy below, the only access allowed will be to the build project with the project ARN for the `${codebuild:projectArn}` IAM variable.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::bucket-name/${codebuild:projectArn}/*"
        }
}
```

Code examples for CodeBuild using AWS SDKs

The following code examples show how to use CodeBuild with an AWS software development kit (SDK).

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

- Basic examples for CodeBuild using AWS SDKs
  - Actions for CodeBuild using AWS SDKs
    - Use CreateProject with an AWS SDK or CLI
    - Use ListBuilds with an AWS SDK or CLI
    - Use ListProjects with an AWS SDK or CLI
    - Use StartBuild with an AWS SDK or CLI

Basic examples for CodeBuild using AWS SDKs

The following code examples show how to use the basics of AWS CodeBuild with AWS SDKs.

Examples

- Actions for CodeBuild using AWS SDKs
  - Use CreateProject with an AWS SDK or CLI
  - Use ListBuilds with an AWS SDK or CLI
  - Use ListProjects with an AWS SDK or CLI
  - Use StartBuild with an AWS SDK or CLI

Actions for CodeBuild using AWS SDKs

The following code examples demonstrate how to perform individual CodeBuild actions with AWS SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

The following examples include only the most commonly used actions. For a complete list, see the AWS CodeBuild API Reference.

Examples

- Use CreateProject with an AWS SDK or CLI
- Use ListBuilds with an AWS SDK or CLI
- Use ListProjects with an AWS SDK or CLI
- Use StartBuild with an AWS SDK or CLI

Use `CreateProject` with an AWS SDK or CLI

The following code examples show how to use `CreateProject`.

CLI

AWS CLI

Example 1: To create an AWS CodeBuild build project

The following `create-project` example creates a CodeBuild build project using source files from an S3 bucket

```
aws codebuild create-project \
    --name "my-demo-project" \
    --source "{\"type\": \"S3\",\"location\": \"codebuild-us-west-2-123456789012-
input-bucket/my-source.zip\"}" \
    --artifacts {"\"type\": \"S3\",\"location\": \"codebuild-us-
west-2-123456789012-output-bucket\""} \
    --environment "{\"type\": \"LINUX_CONTAINER\",\"image\": \"aws/codebuild/
standard:1.0\",\"computeType\": \"BUILD_GENERAL1_SMALL\"}" \
    --service-role "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role"
```

Output:

```
{
    "project": {
        "arn": "arn:aws:codebuild:us-west-2:123456789012:project/my-demo-
project",
        "name": "my-cli-demo-project",
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
        "lastModified": 1556839783.274,
        "badge": {
            "badgeEnabled": false
        },
        "queuedTimeoutInMinutes": 480,
        "environment": {
            "image": "aws/codebuild/standard:1.0",
            "computeType": "BUILD_GENERAL1_SMALL",
            "type": "LINUX_CONTAINER",
            "imagePullCredentialsType": "CODEBUILD",
            "privilegedMode": false,
            "environmentVariables": []
        },
        "artifacts": {
            "location": "codebuild-us-west-2-123456789012-output-bucket",
            "name": "my-cli-demo-project",
            "namespaceType": "NONE",
            "type": "S3",
            "packaging": "NONE",
            "encryptionDisabled": false
        },
        "source": {
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip",
            "insecureSsl": false
        },
        "timeoutInMinutes": 60,
        "cache": {
            "type": "NO_CACHE"
        },
        "created": 1556839783.274
    }
```

```
}
```

Example 2: To create an AWS CodeBuild build project using a JSON input file for the parameters

The following `create-project` example creates a CodeBuild build project by passing all of the required parameters in a JSON input file. Create the input file template by running the command with only the `--generate-cli-skeleton parameter`.

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

The input JSON file `create-project.json` contains the following content:

```
{
    "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:1.0",
        "computeType": "BUILD_GENERAL1_SMALL"
    },
    "serviceRole": "serviceIAMRole"
}
```

Output:

```
{
    "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:1.0",
            "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"
    }
}
```

For more information, see Create a Build Project (AWS CLI) in the *AWS CodeBuild User Guide*.

- For API details, see CreateProject in *AWS CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a project.

```
import {
  ArtifactsType,
  CodeBuildClient,
  ComputeType,
```

```
  CreateProjectCommand,
  EnvironmentType,
  SourceType,
} from "@aws-sdk/client-codebuild";

// Create the AWS CodeBuild project.
export const createProject = async (
  projectName = "MyCodeBuilder",
  roleArn = "arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin",
  buildOutputBucket = "xxxx",
  githubUrl = "https://...",
) => {
  const codeBuildClient = new CodeBuildClient({});

  const response = await codeBuildClient.send(
    new CreateProjectCommand({
      artifacts: {
        // The destination of the build artifacts.
        type: ArtifactsType.S3,
        location: buildOutputBucket,
      },
      // Information about the build environment. The combination of
 "computeType" and "type" determines the
      // requirements for the environment such as CPU, memory, and disk space.
      environment: {
        // Build environment compute types.
        // https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
compute-types.html
        computeType: ComputeType.BUILD_GENERAL1_SMALL,
        // Docker image identifier.
        // See https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-
ref-available.html
        image: "aws/codebuild/standard:7.0",
        // Build environment type.
        type: EnvironmentType.LINUX_CONTAINER,
      },
      name: projectName,
      // A role ARN with permission to create a CodeBuild project, write to the
 artifact location, and write CloudWatch logs.
      serviceRole: roleArn,
      source: {
        // The type of repository that contains the source code to be built.
        type: SourceType.GITHUB,
```

```
       // The location of the repository that contains the source code to be
built.
       location: githubUrl,
     },
   }),
 );
 console.log(response);
 //   {
 //     '$metadata': {
 //       httpStatusCode: 200,
 //       requestId: 'b428b244-777b-49a6-a48d-5dffedced8e7',
 //       extendedRequestId: undefined,
 //       cfId: undefined,
 //       attempts: 1,
 //       totalRetryDelay: 0
 //     },
 //     project: {
 //       arn: 'arn:aws:codebuild:us-east-1:xxxxxxxxxxxx:project/MyCodeBuilder',
 //       artifacts: {
 //         encryptionDisabled: false,
 //         location: 'xxxxxx-xxxxxxx-xxxxxx',
 //         name: 'MyCodeBuilder',
 //         namespaceType: 'NONE',
 //         packaging: 'NONE',
 //         type: 'S3'
 //       },
 //       badge: { badgeEnabled: false },
 //       cache: { type: 'NO_CACHE' },
 //       created: 2023-08-18T14:46:48.979Z,
 //       encryptionKey: 'arn:aws:kms:us-east-1:xxxxxxxxxxxx:alias/aws/s3',
 //       environment: {
 //         computeType: 'BUILD_GENERAL1_SMALL',
 //         environmentVariables: [],
 //         image: 'aws/codebuild/standard:7.0',
 //         imagePullCredentialsType: 'CODEBUILD',
 //         privilegedMode: false,
 //         type: 'LINUX_CONTAINER'
 //       },
 //       lastModified: 2023-08-18T14:46:48.979Z,
 //       name: 'MyCodeBuilder',
 //       projectVisibility: 'PRIVATE',
 //       queuedTimeoutInMinutes: 480,
 //       serviceRole: 'arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin',
 //       source: {
```

```
  //         insecureSsl: false,
  //         location: 'https://...',
  //         reportBuildStatus: false,
  //         type: 'GITHUB'
  //       },
  //       timeoutInMinutes: 60
  //     }
  //   }
  return response;
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see CreateProject in *AWS SDK for JavaScript API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListBuilds` with an AWS SDK or CLI

The following code examples show how to use `ListBuilds`.

C++

SDK for C++

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
//! List the CodeBuild builds.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
```

```
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
   Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

   Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
   listBuildsRequest.SetSortOrder(sortType);

   Aws::String nextToken; // Used for pagination.

   do {
       if (!nextToken.empty()) {
           listBuildsRequest.SetNextToken(nextToken);
       }

       Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
codeBuildClient.ListBuilds(
               listBuildsRequest);

       if (listBuildsOutcome.IsSuccess()) {
           const Aws::Vector<Aws::String> &ids =
listBuildsOutcome.GetResult().GetIds();
           if (!ids.empty()) {

               std::cout << "Information about each build:" << std::endl;
               Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
               getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
               Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
codeBuildClient.BatchGetBuilds(
                       getBuildsRequest);

               if (getBuildsOutcome.IsSuccess()) {
                   const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
getBuildsOutcome.GetResult().GetBuilds();
                   std::cout << builds.size() << " build(s) found." <<
std::endl;
                   for (auto val: builds) {
                       std::cout << val.GetId() << std::endl;
                   }
               } else {
                   std::cerr << "Error getting builds"
                             << getBuildsOutcome.GetError().GetMessage() <<
std::endl;
                   return false;
               }
```

```
            } else {
                std::cout << "No builds found." << std::endl;
            }

            // Get the next token for pagination.

            nextToken = listBuildsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error listing builds"
                      << listBuildsOutcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!nextToken.

            empty()

            );

    return true;
}
```

- For API details, see ListBuilds in *AWS SDK for C++ API Reference*.

CLI

AWS CLI

To get a list of AWS CodeBuild builds IDs.

The following `list-builds` example gets a list of CodeBuild IDs sorted in ascending order.

```
aws codebuild list-builds --sort-order ASCENDING
```

The output includes a `nextToken` value which indicates that there is more output available.

```
{
    "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"
    ]
}
```

Run this command again and provide the `nextToken` value in the previous response as a parameter to get the next part of the output. Repeat until you don't receive a `nextToken` value in the response.

```
aws codebuild list-builds --sort-order ASCENDING --next-
token 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```

Next part of the output:

```
{
    "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"
    ]
}
```

For more information, see View a List of Build IDs (AWS CLI) in the *AWS CodeBuild User Guide*

- For API details, see ListBuilds in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListProjects` with an AWS SDK or CLI

The following code examples show how to use `ListProjects`.

C++

SDK for C++

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
//! List the CodeBuild projects.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType
 sortType,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Next token for pagination.
    Aws::Vector<Aws::String> allProjects;

    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
 codeBuildClient.ListProjects(
                listProjectsRequest);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
 outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(),
 projects.end());
```

```
            nextToken = outcome.GetResult().GetNextToken();
        }

        else {
            std::cerr << "Error listing projects" <<
 outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (!nextToken.empty());

    std::cout << allProjects.size() << " project(s) found." << std::endl;
    for (auto project: allProjects) {
        std::cout << project << std::endl;
    }

    return true;
}
```

- For API details, see ListProjects in *AWS SDK for C++ API Reference*.

CLI

AWS CLI

To get a list of AWS CodeBuild build project names.

The following `list-projects` example gets a list of CodeBuild build projects sorted by name in ascending order.

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

The output includes a `nextToken` value which indicates that there is more output available.

```
{
    "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"
    ]
}
```

Run this command again and provide the `nextToken` value from the previous response as a parameter to get the next part of the output. Repeat until you don't receive a `nextToken` value in the response.

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-
token Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=

{
    "projects": [
        "codebuild-demo-project100",
        "codebuild-demo-project101",

  ... The full list of build project names has been omitted for brevity ...
        "codebuild-demo-project122"

  ]
}
```

For more information, see View a List of Build Project Names (AWS CLI) in the *AWS CodeBuild User Guide*.

- For API details, see ListProjects in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StartBuild` with an AWS SDK or CLI

The following code examples show how to use `StartBuild`.

C++

SDK for C++

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
//! Start an AWS CodeBuild project build.
/*!
  \param projectName: A CodeBuild project name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome =
 codeBuildClient.StartBuild(
            startBuildRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
                  << std::endl;
    }

    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see StartBuild in *AWS SDK for C++ API Reference*.

CLI

AWS CLI

To start running a build of an AWS CodeBuild build project.

The following `start-build` example starts a build for the specified CodeBuild project. The build overrides both the project's setting for the number of minutes the build is allowed to be queued before it times out and the project's artifact settings.

```
aws codebuild start-build \
    --project-name "my-demo-project" \
    --queued-timeout-in-minutes-override 5 \
    --artifacts-override {"\"type\": \"S3\",\"location\":
 \"arn:aws:s3:::artifacts-override\",\"overrideArtifactName\":true"}
```

Output:

```
{
    "build": {
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
        "buildStatus": "IN_PROGRESS",
        "buildComplete": false,
        "projectName": "my-demo-project",
        "timeoutInMinutes": 60,
        "source": {
            "insecureSsl": false,
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip"
        },
        "queuedTimeoutInMinutes": 5,
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "currentPhase": "QUEUED",
        "startTime": 1556905683.568,
        "environment": {
            "computeType": "BUILD_GENERAL1_MEDIUM",
            "environmentVariables": [],
            "type": "LINUX_CONTAINER",
```

```
            "privilegedMode": false,
            "image": "aws/codebuild/standard:1.0",
            "imagePullCredentialsType": "CODEBUILD"
        },
        "phases": [
            {
                "phaseStatus": "SUCCEEDED",
                "startTime": 1556905683.568,
                "phaseType": "SUBMITTED",
                "durationInSeconds": 0,
                "endTime": 1556905684.524
            },
            {
                "startTime": 1556905684.524,
                "phaseType": "QUEUED"
            }
        ],
        "logs": {
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logEvent:group=null;stream=null"
        },
        "artifacts": {
            "encryptionDisabled": false,
            "location": "arn:aws:s3:::artifacts-override/my-demo-project",
            "overrideArtifactName": true
        },
        "cache": {
            "type": "NO_CACHE"
        },
        "id": "my-demo-project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE",
        "initiator": "my-aws-account-name",
        "arn": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-
project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE"
    }
}
```

For more information, see Run a Build (AWS CLI) in the *AWS CodeBuild User Guide*.

- For API details, see StartBuild in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Troubleshooting AWS CodeBuild

Use the information in this topic to help you identify, diagnose, and address issues. To learn how to log and monitor CodeBuild builds to troubleshoot issues, see Logging and monitoring.

Topics

- Error: "QUEUED: INSUFFICIENT_SUBNET" when a build in a build queue fails
- Error: "Unable to download cache: RequestError: Send request failed caused by: x509: Failed to load system roots and no roots provided"
- Error: "Unable to download certificate from S3. AccessDenied"
- Error: "Unable to locate credentials"
- RequestError timeout error when running CodeBuild in a proxy server
- The bourne shell (sh) must exist in build images
- Warning: "Skipping install of runtimes. runtime version selection is not supported by this build image" when running a build
- Error: "Unable to verify JobWorker identity" when opening the CodeBuild console
- Build failed to start
- Accessing GitHub metadata in locally cached builds
- AccessDenied: The bucket owner for the report group does not match the owner of the S3 bucket...
- Error: "Your credentials lack one or more required privilege scopes" when creating a CodeBuild project with CodeConnections
- Error: "Sorry, no terminal at all requested - can't get input" when building with the Ubuntu install command

Apache Maven builds reference artifacts from the wrong repository

Issue: When you use Maven with an AWS CodeBuild-provided Java build environment, Maven pulls build and plugin dependencies from the secure central Maven repository at https://repo1.maven.org/maven2. This happens even if your build project's `pom.xml` file explicitly declares other locations to use instead.

Possible cause: CodeBuild-provided Java build environments include a file named `settings.xml` that is preinstalled in the build environment's `/root/.m2` directory. This `settings.xml` file contains the following declarations, which instruct Maven to always pull build and plugin dependencies from the secure central Maven repository at https://repo1.maven.org/maven2.

```
<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>
```

Recommended solution: Do the following:

1. Add a `settings.xml` file to your source code.
2. In this `settings.xml` file, use the preceding `settings.xml` format as a guide to declare the repositories you want Maven to pull the build and plugin dependencies from instead.
3. In the `install` phase of your build project, instruct CodeBuild to copy your `settings.xml` file to the build environment's `/root/.m2` directory. For example, consider the following snippet from a `buildspec.yml` file that demonstrates this behavior.

```
version 0.2

phases:
  install:
    commands:
```

```
      - cp ./settings.xml /root/.m2/settings.xml
```

Build commands run as root by default

Issue: AWS CodeBuild runs your build commands as the root user. This happens even if your related build image's Dockerfile sets the USER instruction to a different user.

Cause: By default, CodeBuild runs all build commands as the root user.

Recommended solution: None.

Builds might fail when file names have non-U.S. English characters

Issue: When you run a build that uses files with file names that contain non-U.S. English characters (for example, Chinese characters), the build fails.

Possible cause: Build environments provided by AWS CodeBuild have their default locale set to `POSIX`. `POSIX` localization settings are less compatible with CodeBuild and file names that contain non-U.S. English characters and can cause related builds to fail.

Recommended solution: Add the following commands to the `pre_build` section of your buildspec file. These commands make the build environment use U.S. English UTF-8 for its localization settings, which is more compatible with CodeBuild and file names that contain non-U.S. English characters.

For build environments based on Ubuntu:

```
pre_build:
  commands:
    - export LC_ALL="en_US.UTF-8"
    - locale-gen en_US en_US.UTF-8
    - dpkg-reconfigure -f noninteractive locales
```

For build environments based on Amazon Linux:

```
pre_build:
  commands:
```

```
    - export LC_ALL="en_US.utf8"
```

Builds might fail when getting parameters from Amazon EC2 Parameter Store

Issue: When a build tries to get the value of one or more parameters stored in Amazon EC2 Parameter Store, the build fails in the DOWNLOAD_SOURCE phase with the error `Parameter does not exist`.

Possible cause: The service role the build project relies on does not have permission to call the `ssm:GetParameters` action or the build project uses a service role that is generated by AWS CodeBuild and allows calling the `ssm:GetParameters` action, but the parameters have names that do not start with `/CodeBuild/`.

Recommended solutions:

- If the service role was not generated by CodeBuild, update its definition to allow CodeBuild to call the `ssm:GetParameters` action. For example, the following policy statement allows calling the `ssm:GetParameters` action to get parameters with names starting with `/CodeBuild/`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/CodeBuild/*"
    }
  ]
}
```

- If the service role was generated by CodeBuild, update its definition to allow CodeBuild to access parameters in Amazon EC2 Parameter Store with names other than those starting with `/CodeBuild/`. For example, the following policy statement allows calling the `ssm:GetParameters` action to get parameters with the specified name:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/PARAMETER_NAME"
    }
  ]
}
```

Cannot access branch filter in the CodeBuild console

Issue: The branch filter option is not available in the console when you create or update an AWS CodeBuild project.

Possible cause: The branch filter option is deprecated. It has been replaced by webhook filter groups, which provide more control over the webhook events that trigger a new build in CodeBuild.

Recommended solution: To migrate a branch filter that you created before the introduction of webhook filters, create a webhook filter group with a `HEAD_REF` filter with the regular expression `^refs/heads/branchName$`. For example, if your branch filter regular expression was `^branchName$`, then the updated regular expression you put in the `HEAD_REF` filter is `^refs/heads/branchName$`. For more information, see Bitbucket webhook events and Filter GitHub webhook events (console).

Cannot view build success or failure

Issue: You cannot see the success or failure of a retried build.

Possible cause: The option to report your build's status is not enabled.

Recommended solutions: Enable **Report build status** when you create or update a CodeBuild project. This option tells CodeBuild to report back the status when you trigger a build. For more information, see reportBuildStatus in the *AWS CodeBuild API Reference*.

Build status not reported to source provider

Issue: After allowing build status reporting to a source provider, such as GitHub or Bitbucket, the build status is not updated.

Possible cause: The user associated with the source provider does not have write access to the repo.

Recommended solutions: To be able to report the build status to the source provider, the user associated with the source provider must have write access to the repo. If the user does not have write access, the build status cannot be updated. For more information, see Source provider access.

Cannot find and select the base image of the Windows Server Core 2019 platform

Issue: You cannot find or select the base image of the Windows Server Core 2019 platform.

Possible cause: You are using an AWS Region that does not support this image.

Recommended solutions: Use one of the following AWS Regions where the base image of the Windows Server Core 2019 platform is supported:

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)
- Europe (Ireland)

Earlier commands in buildspec files are not recognized by later commands

Issue: The results of one or more commands in your buildspec file are not recognized by later commands in the same buildspec file. For example, a command might set a local environment variable, but a command run later might fail to get the value of that local environment variable.

Possible cause: In buildspec file version 0.1, AWS CodeBuild runs each command in a separate instance of the default shell in the build environment. This means that each command runs in isolation from all other commands. By default, then, you cannot run a single command that relies on the state of any previous commands.

Recommended solutions: We recommend that you use build spec version 0.2, which solves this issue. If you must use buildspec version 0.1, we recommend that you use the shell command chaining operator (for example, && in Linux) to combine multiple commands into a single command. Or include a shell script in your source code that contains multiple commands, and then call that shell script from a single command in the buildspec file. For more information, see Shells and commands in build environments and Environment variables in build environments.

Error: "Access denied" when attempting to download cache

Issue: When attempting to download the cache on a build project that has cache enabled, you receive an `Access denied` error.

Possible causes:

- You have just configured caching as part of your build project.
- The cache has recently been invalidated through the `InvalidateProjectCache` API.
- The service role being used by CodeBuild does not have `s3:GetObject` and `s3:PutObject` permissions to the S3 bucket that is holding the cache.

Recommended solution: For first time use, it's normal to see this immediately after updating the cache configuration. If this error persists, then you should check to see if your service role has `s3:GetObject` and `s3:PutObject` permissions to the S3 bucket that is holding the cache. For more information, see Specifying S3 permissions in the *Amazon S3 Developer Guide*.

Error: "BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE" when using a custom build image

Issue: When you try to run a build that uses a custom build image, the build fails with the error `BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE`.

Possible cause: *The build image's overall uncompressed size is larger than the build environment compute type's available disk space. To check your build image's size, use Docker to run the `docker images REPOSITORY:TAG` command. For a list of available disk space by compute type, see Build environment compute modes and types.*

> **Recommended solution:** Use a larger compute type with more available disk space, or reduce the size of your custom build image.

Possible cause: *AWS CodeBuild does not have permission to pull the build image from your Amazon Elastic Container Registry (Amazon ECR).*

> **Recommended solution:** Update the permissions in your repository in Amazon ECR so that CodeBuild can pull your custom build image into the build environment. For more information, see the Amazon ECR sample.

Possible cause: *The Amazon ECR image you requested is not available in the AWS Region that your AWS account is using.*

Recommended solution: Use an Amazon ECR image that is in the same AWS Region as the one your AWS account is using.

Possible cause: *You are using a private registry in a VPC that does not have public internet access. CodeBuild cannot pull an image from a private IP address in a VPC. For more information, see Private registry with AWS Secrets Manager sample for CodeBuild.*

Recommended solution: If you use a private registry in a VPC, make sure the VPC has public internet access.

Possible cause: *If the error message contains "**toomanyrequests**", and the image is obtained from Docker Hub, this error means the Docker Hub pull limit has been reached.*

Recommended solution: Use a Docker Hub private registry, or obtain your image from Amazon ECR. For more information about using a private registry, see Private registry with AWS Secrets Manager sample for CodeBuild. For more information about using Amazon ECR, see Amazon ECR sample for CodeBuild .

Error: "Build container found dead before completing the build. build container died because it was out of memory, or the Docker image is not supported. ErrorCode: 500"

Issue: When you try to use a Microsoft Windows or Linux container in AWS CodeBuild, this error occurs during the PROVISIONING phase.

Possible causes:

- The container OS version is not supported by CodeBuild.
- HTTP_PROXY, HTTPS_PROXY, or both are specified in the container.

Recommended solutions:

- For Microsoft Windows, use a Windows container with a container OS that is version microsoft/windowsservercore:10.0.x (for example, microsoft/windowsservercore:10.0.14393.2125).

- For Linux, clear the HTTP_PROXY and HTTPS_PROXY settings in your Docker image, or specify the VPC configuration in your build project.

Error: "Cannot connect to the Docker daemon" when running a build

Issue: Your build fails and you receive an error similar to `Cannot connect to the Docker daemon at unix:/var/run/docker.sock. Is the docker daemon running?` in the build log.

Possible cause: You are not running your build in privileged mode.

Recommended solution: To fix this error, you must enable privileged mode and update your buildspec using the following instructions.

To run your build in privileged mode, follow these steps:

1. Open the CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In the navigation pane, choose **Build projects**, then choose your build project.
3. From **Edit**, choose **Environment**.
4. Choose **Additional configuration**.
5. From **Privileged**, select **Enable this flag if you want to build Docker images or want your builds to get elevated privileges.**.
6. Choose **Update environment**.
7. Choose **Start build** to retry your build.

You'll also need to start the Docker daemon inside your container. The `install` phase of your buildspec might look similar to this.

```
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"
```

For more information about the OverlayFS storage driver referenced in the buildspec file, see Use the OverlayFS storage driver on the Docker website.

> **Note**
>
> If the base operating system is Alpine Linux, in the `buildspec.yml` add the `-t` argument to `timeout`:
>
> ```
> - timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
> ```

To learn more about how to build and run a Docker image by using AWS CodeBuild, see Docker in custom image sample for CodeBuild.

Error: "CodeBuild is not authorized to perform: sts:AssumeRole" when creating or updating a build project

Issue: When you try to create or update a build project, you receive the error `Code:InvalidInputException, Message:CodeBuild is not authorized to perform: sts:AssumeRole on arn:aws:iam::`*`account-ID`*`:role/`*`service-role-name`*.

Possible causes:

- The AWS Security Token Service (AWS STS) has been deactivated for the AWS region where you are attempting to create or update the build project.
- The AWS CodeBuild service role associated with the build project does not exist or does not have sufficient permissions to trust CodeBuild.
- The AWS CodeBuild service role casing associated with the build project does not match with the actual IAM role.

Recommended solutions:

- Make sure AWS STS is activated for the AWS region where you are attempting to create or update the build project. For more information, see Activating and deactivating AWS STS in an AWS Region in the *IAM User Guide*.
- Make sure the target CodeBuild service role exists in your AWS account. If you are not using the console, make sure you did not misspell the Amazon Resource Name (ARN) of the service role

when you created or updated the build project. Note that IAM roles are case sensitive, so check that the casing of the IAM role is correct.

- Make sure the target CodeBuild service role has sufficient permissions to trust CodeBuild. For more information, see the trust relationship policy statement in Allow CodeBuild to interact with other AWS services.

Error: "Error calling GetBucketAcl: Either the bucket owner has changed or the service role no longer has permission to called s3:GetBucketAcl"

Issue: When you run a build, you receive an error about a change in ownership of an S3 bucket and `GetBucketAcl` permissions.

Possible cause: You added the `s3:GetBucketAcl` and `s3:GetBucketLocation` permissions to your IAM role. These permissions secure your project's S3 bucket and ensure that only you can access it. After you added these permissions, the owner of the S3 bucket changed.

Recommended solution: Verify you are an owner of the S3 bucket, and then add permissions to your IAM role again. For more information, see Secure access to S3 buckets.

Error: "Failed to upload artifacts: Invalid arn" when running a build

Issue: When you run a build, the `UPLOAD_ARTIFACTS` build phase fails with the error `Failed to upload artifacts: Invalid arn`.

Possible cause: Your S3 output bucket (the bucket where AWS CodeBuild stores its output from the build) is in an AWS Region different from the CodeBuild build project.

Recommended solution: Update the build project's settings to point to an output bucket that is in the same AWS Region as the build project.

Error: "Git clone failed: Unable to access '`your-repository-URL`': SSL certificate problem: Self signed certificate"

Issue: When you try to run a build project, the build fails with this error.

Possible cause: Your source repository has a self-signed certificate, but you have not chosen to install the certificate from your S3 bucket as part of your build project.

Recommended solutions:

- Edit your project. For **Certificate**, choose **Install certificate from S3**. For **Bucket of certificate**, choose the S3 bucket where your SSL certificate is stored. For **Object key of certificate**, enter the name of your S3 object key.
- Edit your project. Select **Insecure SSL** to ignore SSL warnings while connecting to your GitHub Enterprise Server project repository.

> **Note**
>
> We recommend that you use **Insecure SSL** for testing only. It should not be used in a production environment.

Error: "The bucket you are attempting to access must be addressed using the specified endpoint" when running a build

Issue: When you run a build, the `DOWNLOAD_SOURCE` build phase fails with the error `The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint.`

Possible cause: Your pre-built source code is stored in an S3 bucket, and that bucket is in an AWS Region different from the AWS CodeBuild build project.

Recommended solution: Update the build project's settings to point to a bucket that contains your pre-built source code. Make sure that bucket is in the same AWS Region as the build project.

Error: "This build image requires selecting at least one runtime version."

Issue: When you run a build, the `DOWNLOAD_SOURCE` build phase fails with the error `YAML_FILE_ERROR: This build image requires selecting at least one runtime version`.

Possible cause: Your build uses version 1.0 or later of the Amazon Linux 2 (AL2) standard image, or version 2.0 or later of the Ubuntu standard image, and a runtime is not specified in the buildspec file.

Recommended solution: If you use the `aws/codebuild/standard:2.0` CodeBuild managed image, you must specify a runtime version in the `runtime-versions` section of the buildspec file. For example, you might use the following buildspec file for a project that uses PHP:

```
version: 0.2

phases:
  install:
    runtime-versions:
        php: 7.3
  build:
    commands:
      - php --version
artifacts:
  files:
    -  README.md
```

> **Note**
>
> If you specify a `runtime-versions` section and use an image other than Ubuntu Standard Image 2.0 or later, or the Amazon Linux 2 (AL2) standard image 1.0 or later, the build issues the warning, "`Skipping install of runtimes. Runtime version selection is not supported by this build image.`"

For more information, see Specify runtime versions in the buildspec file.

Error: "QUEUED: INSUFFICIENT_SUBNET" when a build in a build queue fails

Issue: A build in a build queue fails with an error similar to `QUEUED: INSUFFICIENT_SUBNET`.

Possible causes: The IPv4 CIDR block specified for your VPC uses a reserved IP address. The first four IP addresses and the last IP address in each subnet CIDR block are not available for you to use

and cannot be assigned to an instance. For example, in a subnet with CIDR block `10.0.0.0/24`, the following five IP addresses are reserved:

- `10.0.0.0:` Network address.
- `10.0.0.1`: Reserved by AWS for the VPC router.
- `10.0.0.2`: Reserved by AWS. The IP address of the DNS server is always the base of the VPC network range plus two; however, we also reserve the base of each subnet range plus two. For VPCs with multiple CIDR blocks, the IP address of the DNS server is located in the primary CIDR. For more information, see Amazon DNS server in the *Amazon VPC User Guide*.
- `10.0.0.3`: Reserved by AWS for future use.
- `10.0.0.255`: Network broadcast address. We do not support broadcast in a VPC. This address is reserved.

Recommended solutions: Check if your VPC uses a reserved IP address. Replace any reserved IP address with one that is not reserved. For more information, see VPC and subnet sizing in the *Amazon VPC User Guide*.

Error: "Unable to download cache: RequestError: Send request failed caused by: x509: Failed to load system roots and no roots provided"

Issue: When you try to run a build project, the build fails with this error.

Possible cause: You configured caching as part of your build project and are using an older Docker image that includes an expired root certificate.

Recommended solution: Update the Docker image that is being used in your AWS CodeBuild the project. For more information, see Docker images provided by CodeBuild.

Error: "Unable to download certificate from S3. AccessDenied"

Issue: When you try to run a build project, the build fails with this error.

Possible causes:

- You have chosen the wrong S3 bucket for your certificate.

- You have entered the wrong object key for your certificate.

Recommended solutions:

- Edit your project. For **Bucket of certificate**, choose the S3 bucket where your SSL certificate is stored.
- Edit your project. For **Object key of certificate**, enter the name of your S3 object key.

Error: "Unable to locate credentials"

Issue: When you try to run the AWS CLI, use an AWS SDK, or call another similar component as part of a build, you get build errors that are directly related to the AWS CLI, AWS SDK, or component. For example, you might get a build error such as `Unable to locate credentials`.

Possible causes:

- The version of the AWS CLI, AWS SDK, or component in the build environment is incompatible with AWS CodeBuild.
- You are running a Docker container within a build environment that uses Docker, and the container does not have access to the AWS credentials by default.

Recommended solutions:

- Make sure your build environment has the following version or higher of the AWS CLI, AWS SDK, or component.
  - 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 SDK for 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

- If you need to run a Docker container in a build environment and the container requires AWS credentials, you must pass through the credentials from the build environment to the container. In your buildspec file, include a Docker `run` command such as the following. This example uses the `aws s3 ls` command to list your available S3 buckets. The `-e` option passes through the environment variables required for your container to access AWS credentials.

```
docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI your-image-tag aws s3 ls
```

- If you are building a Docker image and the build requires AWS credentials (for example, to download a file from Amazon S3), you must pass through the credentials from the build environment to the Docker build process as follows.

  1. In your source code's Dockerfile for the Docker image, specify the following `ARG` instructions.

     ```
     ARG AWS_DEFAULT_REGION
     ARG AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
     ```

  2. In your buildspec file, include a Docker `build` command such as the following. The `--build-arg` options sets the environment variables required for your Docker build process to access the AWS credentials.

     ```
     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 timeout error when running CodeBuild in a proxy server

Issue: You receive a `RequestError` error similar to one of the following:

- `RequestError: send request failed caused by: Post https://logs.<your-region>.amazonaws.com/: dial tcp 52.46.158.105:443: i/o timeout` from CloudWatch Logs.

- `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 refused` from Amazon S3.

Possible causes:

- `ssl-bump` is not configured properly.
- Your organization's security policy does not allow you to use `ssl_bump`.
- Your buildspec file does not have proxy settings specified using a `proxy` element.

Recommended solutions:

- Make sure `ssl-bump` is configured properly. If you use Squid for your proxy server, see Configure Squid as an explicit proxy server.
- Follow these steps to use private endpoints for Amazon S3 and CloudWatch Logs:

  1. In your private subnet routing table, remove the rule you added that routes traffic destined for the internet to your proxy server. For information, see Creating a subnet in your VPC in the *Amazon VPC User Guide*.
  2. Create a private Amazon S3 endpoint and CloudWatch Logs endpoint and associate them with the private subnet of your Amazon VPC. For information, see VPC endpoint services in the *Amazon VPC User Guide*.
  3. Confirm **Enable Private DNS Name** in your Amazon VPC is selected. For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

- If you do not use `ssl-bump` for an explicit proxy server, add a proxy configuration to your buildspec file using a `proxy` element. For more information, see Run CodeBuild in an explicit proxy server and Buildspec syntax.

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
```

The bourne shell (sh) must exist in build images

Issue: You are using a build image that is not provided by AWS CodeBuild, and your builds fail with the message `Build container found dead before completing the build`.

Possible cause: The Bourne shell (`sh`) is not included in your build image. CodeBuild needs `sh` to run build commands and scripts.

Recommended solution: If `sh` in not present in your build image, be sure to include it before you start any more builds that use your image. (CodeBuild already includes `sh` in its build images.)

Warning: "Skipping install of runtimes. runtime version selection is not supported by this build image" when running a build

Issue: When you run a build, the build log contains this warning.

Possible cause: Your build does not use version 1.0 or later of the Amazon Linux 2 (AL2) standard image, or version 2.0 or later of the Ubuntu standard image, and a runtime is specified in a `runtime-versions` section in your buildspec file.

Recommended solution: Be sure your buildspec file does not contain a `runtime-versions` section. The `runtime-versions` section is only required if you use the Amazon Linux 2 (AL2) standard image or later or the Ubuntu standard image version 2.0 or later.

Error: "Unable to verify JobWorker identity" when opening the CodeBuild console

Issue: When you open the CodeBuild console, an "Unable to verify JobWorker identity" error message is displayed.

Possible cause: The IAM role that is used for console access has a tag with `jobId` as the key. This tag key is reserved for CodeBuild and will cause this error if it is present.

Recommended solution: Change any custom IAM role tags that have the key `jobId` to have a different key, such as `jobIdentifier`.

Build failed to start

Issue: When starting a build, you receive a **Build failed to start** error message.

Possible cause: The number of concurrent builds has been reached.

Recommended solutions: Wait until other builds are complete, or increase the concurrrent build limit for the project, and start the build again. For more information, see Project configuration.

Accessing GitHub metadata in locally cached builds

Issue: In some cases, the .git directory in a cached build is a text file and not a directory.

Possible causes: When local source caching is enabled for a build, CodeBuild creates a gitlink for the `.git` directory. This means that the `.git` directory is actually a text file containing the path to the directory.

Recommended solutions: In all cases, use the following command to obtain the Git metadata directory. This command will work no matter the format of `.git`:

```
git rev-parse --git-dir
```

AccessDenied: The bucket owner for the report group does not match the owner of the S3 bucket...

Issue: When uploading test data to an Amazon S3 bucket, CodeBuild is unable to write the test data to the bucket.

Possible causes:

- The account specified for the report group bucket owner does not match the owner of the Amazon S3 bucket.
- The service role does not have write access to the bucket.

Recommended solutions:

- Change the report group bucket owner to match the owner of the Amazon S3 bucket.

- Modify the service role to allow write access to the Amazon S3 bucket.

Error: "Your credentials lack one or more required privilege scopes" when creating a CodeBuild project with CodeConnections

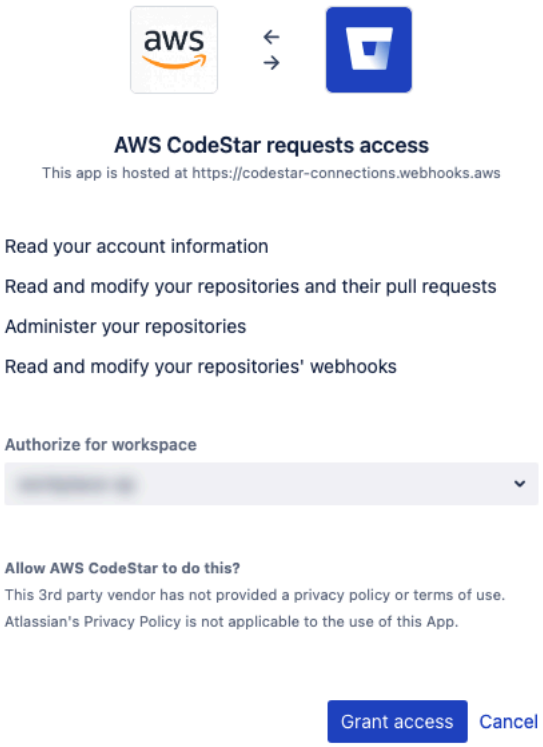
Issue: When creating a CodeBuild project with CodeConnections, you don't have permission to install a Bitbucket webhook.

Possible causes:

- The new permission scope may not have been accepted in your Bitbucket account.

Recommended solutions:

- To accept the new permission, you should have received any email with a subject titled **Action required - Scopes for AWS CodeStar have changed** sent by Bitbucket, `notifications-noreply@bitbucket.org`. The email contains a link to grant the webhook permissions to your existing CodeConnections Bitbucket app installation.
- If you cannot locate the email, you can grant the permission by navigating to `https://bitbucket.org/site/addons/reauthorize?account=`*`<workspace-name>`*`&addon_key=aws-codestar`, or `https://bitbucket.org/site/addons/reauthorize?addon_key=aws-codestar` and selecting the workspace you'd like to grant the webhook permission to.



Error: "Sorry, no terminal at all requested - can't get input" when building with the Ubuntu install command

Issue: If you're running GPU container privileged builds, you may be installing the NVIDIA Container Toolkit following this procedure. In the latest CodeBuild image release, CodeBuild pre-installs and configures docker with `nvidia-container-toolkit` in the latest `amazonlinux` and `ubuntu` curated image. Following this procedure will cause builds with the Ubuntu install command to fail with following error:

```
Running command curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | gpg --
dearmor --no-tty -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
gpg: Sorry, no terminal at all requested - can't get input
curl: (23) Failed writing body
```

Possible causes: The gpg key already exists at the same location.

Recommended solutions: The `nvidia-container-toolkit` is already installed in the image. If you see this error, you can skip the install and restart docker process in your buildspec.

Quotas for AWS CodeBuild

The following tables list the current quotas in AWS CodeBuild. These quotas are for each supported AWS Region for each AWS account, unless otherwise specified.

Service quotas

The following are the default quotas for the AWS CodeBuild service.

Name	Default	Adjuse	Description
Associated tags per project	Each supported Region: 50	No	Maximum number of tags you can associate with a build project
Build projects	Each supported Region: 5,000	Yes	Maximum number of build projects
Build timeout in minutes	Each supported Region: 2,160	No	Maximum build timeout in minutes
Concurrent request for information about builds	Each supported Region: 100	No	Maximum number of builds you can request information about at any one time using the AWS CLI or an AWS SDK.
Concurrent requests for information on build projects	Each supported Region: 100	No	Maximum number of build projects you can request information about at any one time using the AWS CLI or an AWS SDK.
Concurrently running builds for ARM Lambda/10GB environment	Each supported Region: 1	Yes	Maximum number of concurrently

Name	Default	Adjustable	Description
			running builds for ARM Lambda/10GB environment
Concurrently running builds for ARM Lambda/1GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM Lambda/1GB environment
Concurrently running builds for ARM Lambda/2GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM Lambda/2GB environment
Concurrently running builds for ARM Lambda/4GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM Lambda/4GB environment
Concurrently running builds for ARM Lambda/8GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM Lambda/8GB environment
Concurrently running builds for ARM/2XLarge environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM/2XLarge environment

Name	Default	Adjustable	Description
Concurrently running builds for ARM/Large environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM/Large environment
Concurrently running builds for ARM/Medium environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM/Medium environment
Concurrently running builds for ARM/Small environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM/Small environment
Concurrently running builds for ARM/XLarge environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for ARM/XLarge environment
Concurrently running builds for Linux GPU Large environment	Each supported Region: 0	Yes	Maximum number of concurrently running builds for Linux GPU/Large environment
Concurrently running builds for Linux GPU Small environment	Each supported Region: 0	Yes	Maximum number of concurrently running builds for Linux GPU/Small environment
Concurrently running builds for Linux Lambda/10GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux Lambda/10GB environment

Name	Default	Adjustable	Description
Concurrently running builds for Linux Lambda/1GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux Lambda/1GB environment
Concurrently running builds for Linux Lambda/2GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux Lambda/2GB environment
Concurrently running builds for Linux Lambda/4GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux Lambda/4GB environment
Concurrently running builds for Linux Lambda/8GB environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux Lambda/8GB environment
Concurrently running builds for Linux/2XLarge environment	Each supported Region: 0	Yes	Maximum number of concurrently running builds for Linux/2XLarge environment
Concurrently running builds for Linux/ Large environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux/Large environment

Name	Default	Adjustable	Description
Concurrently running builds for Linux/Medium environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux/Medium environment
Concurrently running builds for Linux/Small environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux/Small environment
Concurrently running builds for Linux/XLarge environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Linux/XLarge environment
Concurrently running builds for Windows Server 2019/Large environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Windows Server 2019/Large environment
Concurrently running builds for Windows Server 2019/Medium environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Windows Server 2019/Medium environment
Concurrently running builds for Windows Server 2022/2XLarge environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Windows Server 2022/2XLarge environment

Name	Default	Adjustable	Description
Concurrently running builds for Windows Server 2022/Large environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Windows Server 2022/Large environment
Concurrently running builds for Windows Server 2022/Medium environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Windows Server 2022/Medium environment
Concurrently running builds for Windows Server 2022/XLarge environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Windows Server 2022/XLarge environment
Concurrently running builds for Windows/Large environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Windows/Large environment
Concurrently running builds for Windows/Medium environment	Each supported Region: 1	Yes	Maximum number of concurrently running builds for Windows/Medium environment
Minimum period for build timeout in minutes	Each supported Region: 5	No	Minimum build timeout in minutes
Security groups under VPC configuration	Each supported Region: 5	No	Security groups available for VPC configuration

Name	Default	Adjuste	Description
Subnets under VPC configuration	Each supported Region: 16	No	Subnets available for VPC configuration

> **Note**
>
> Internal metrics will determine the default quotas for concurrent running builds.

Quotas for the maximum number of concurrent running builds vary, depending on the compute type. For some platforms and compute types, the default is 20. To request a higher concurrent build quota, or if you get a "Cannot have more than X active builds for the account" error, use the link above to make the request. For more information on pricing, see AWS CodeBuild pricing.

Other limits

Build projects

Resource	Default
Allowed characters in a build project description	Any
Allowed characters in a build project name	The letters `A-Z` and `a-z`, the numbers `0-9`, and the special characters `-` and `_`
Length of a build project name	2 to 150 characters, inclusive
Maximum length of a build project description	255 characters
Maximum number of reports you can add to a project	5

Resource	Default
Number of minutes you can specify in a build project for the build timeout of all related builds	5 to 2160 (36 hours)

Builds

Resource	Default
Maximum time the history of a build is retained	1 year
Number of minutes you can specify for the build timeout of a single build	5 to 2160 (36 hours)

Compute fleets

Resource	Default
Concurrent number of compute fleets	10
Concurrently running instances for ARM/Small environment fleets	1
Concurrently running instances for ARM/Large environment fleets	1
Concurrently running instances for Linux/Small environment fleets	1
Concurrently running instances for Linux/Medium environment fleets	1
Concurrently running instances for Linux/Large environment fleets	1

Resource	Default
Concurrently running instances for Linux/XLarge environment fleets	1
Concurrently running instances for Linux/2XLarge environment fleets	0
Concurrently running instances for Linux GPU/Small environment fleets	0
Concurrently running instances for Linux GPU/Large environment fleets	0
Concurrently running instances for Windows Server 2019/Medium environment fleets	1
Concurrently running instances for Windows Server 2019/Large environment fleets	1
Concurrently running instances for Windows Server 2022/Medium environment fleets	1
Concurrently running instances for Windows Server 2022/Large environment fleets	1
Concurrently running instances for Mac ARM/Medium environment fleets	1
Concurrently running instances for Mac ARM/Large environment fleets	1

Reports

Resource	Default
Maximum duration a test report is available after it is created	30 days

Resource	Default
Maximum length of a test case message	5,000 characters
Maximum length of a test case name	1,000 characters
Maximum number of report groups per AWS account	5,000
Maximum number of test cases per report	500

Tags

Tag limits apply to tags on CodeBuild build projects and CodeBuild report group resources.

Resource	Default
Resource tag key names	Any combination of Unicode letters, numbers, spaces, and allowed characters in UTF-8 between 1 and 127 characters in length. Allowed characters are `+ - = . _ : / @` Tag key names must be unique, and each key can only have one value. A tag key name cannot: • begin with `aws:` • consist only of spaces • end with a space • contain emojis or any of the following characters: `? ^ * [ \ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;`
Resource tag values	Any combination of Unicode letters, numbers, spaces, and allowed characters in UTF-8 between 0 and 255 characters in length. Allowed characters are `+ - = . _ : / @`

Resource	Default
	A key can only have one value, but many keys can have the same value. A tag key value cannot contain emojis or any of the following characters: ? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;

AWS CodeBuild User Guide document history

The following table describes the important changes to the documentation since the last release of AWS CodeBuild. For notification about updates to this documentation, you can subscribe to an RSS feed.

- **Latest API version:** 2016-10-06

Change	Description	Date
New compute type: CUSTOM_INSTANCE_TYPE	CodeBuild now allows you to create a reserved capacity fleet with a specific instance type by using CUSTOM_INSTANCE_TYPE .	April 23, 2025
New support for CodeBuild sandbox	Added information about using the new CodeBuild sandbox. See Debug builds with CodeBuild sandbox.	April 7, 2025
New Windows environment types	CodeBuild now supports Windows XL and 2XL environment types. For more information, see Build environment compute types.	March 31, 2025
Updated Amazon S3 caching	CodeBuild now supports a new caching behavior for Amazon S3 caching.	March 28, 2025
New content: GitHub Actions runner configuration options	CodeBuild now supports CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATIO	March 11, 2025

	N_NAME for registration at the enterprise level.	
New content: Add new webhook filter type	Add support for a new webhook filter type (ORGANIZATION_NAME).	March 11, 2025
New content: Tutorial for Apple code signing with Fastlane with S3 certificate storage	Add new tutorial for Apple code signing with Fastlane in CodeBuild using S3 for certificate storage	February 5, 2025
New content: Tutorial for Apple code signing with Fastlane with GitHub certificate storage	Add new tutorial for Apple code signing with Fastlane in CodeBuild using GitHub for certificate storage	February 5, 2025
New content: Buildkite runner	Add new content for the Buildkite runner	January 31, 2025
New content: Buildkite manual webhooks	Add support for Buildkite manual webhooks.	January 31, 2025
New content: Batch build buildspec reference	Add support for batch builds in reserved capacity fleets and Lambda environments.	January 8, 2025
New content: Execute parallel tests in batch builds	Add new content for parallel tests in batch builds.	January 2, 2025
New content: Retry builds automatically	CodeBuild now supports auto-retry for webhook builds.	December 18, 2024
New content: Configure a private registry credential for self-hosted runners	Add support for setting registry credentials when using custom images from non-private registries.	December 13, 2024

New content: GitHub Actions runner configuration options	CodeBuild GitHub Actions self-hosted runners now allow you to register your runners at the organization level and configure a specific runner group ID.	December 12, 2024
New content: Add on-failure attribute RETRY	CodeBuild now allows you to configure an on-failure attribute to RETRY in your buildspec.	December 12, 2024
New content: GitLab manual webhooks	Add support for GitLab manual webhooks.	December 11, 2024
Updated content: Updated aliases	Update aliases for Linux-based standard runtime images.	November 22, 2024
Updated content: Label overrides supported with the CodeBuild-hosted GitLab runner	Add support for custom image label overrides for GitLab runners.	November 22, 2024
Updated content: Label overrides supported with the CodeBuild-hosted GitHub Actions runner	Add support for custom image label overrides for GitHub Actions runners.	November 22, 2024
Updated content: AWS managed (predefined) policies for AWS CodeBuild	The AWSCodeBuildAdminAccess, AWSCodeBuildDeveloperAccess, and AWSCodeBuildReadOnlyAccess policies have been updated. The original resource `arn:aws:codebuild:*` has been updated to `arn:aws:codebuild:*:*:project/*` .	November 15, 2024

Updated content: Reserved capacity	Reserved capacity fleets now support non-container builds: ARM EC2, Linux EC2, and Windows EC2.	November 12, 2024
Updated content: Reserved capacity	Reserved capacity fleets now support attribute-based compute.	November 6, 2024
New content: Retry builds automatically	CodeBuild now allows you to enable auto-retry for your builds.	October 25, 2024
New content: Run CodeBuild in a managed proxy server for reserved capacity fleets	Add proxy configurations support for reserved capacity fleets.	October 15, 2024
New content: Self-managed GitLab runners	Add new content for self-managed GitLab runners	September 17, 2024
New content: GitLab group webhooks	Add support for GitLab group webhooks.	September 17, 2024
New content: Run buildspec commands the INSTALL, PRE_BUILD, and POST_BUILD phases	Add support for `-with-buildspec .`	August 20, 2024
Updated content: Reserved capacity	Reserved capacity fleets now support macOS.	August 19, 2024
New content: GitHub App connections	Add support for GitHub App connections.	August 14, 2024
New content: Bitbucket App connections	Add support for Bitbucket App connections.	August 14, 2024

New content: Multiple access tokens in CodeBuild	Add support for sourcing access tokens to third party providers from secrets in AWS Secrets Manager or through AWS CodeConnections connections.	August 14, 2024
Updated content: Reserved capacity	Reserved capacity fleets now support ARM Medium, ARM XLarge, and ARM 2XLarge compute types.	August 5, 2024
Updated content: Reserved capacity	CodeBuild now supports VPC connectivity for reserved capacity fleets on Windows.	August 1, 2024
New ARM compute types	CodeBuild now supports ARM Medium, ARM XLarge, and ARM 2XLarge compute types. For more information, see Build environment compute types.	July 10, 2024
Updated content: SHA signature	Update the Secure Hash Algorithm (SHA) signature for the x86_64 and ARM.	June 19, 2024
New content: GitHub global and organization webhooks	Add support for GitHub global and organization webhooks.	June 17, 2024
New content: Add new webhook filter type	Add support for a new webhook filter type (`REPOSITORY_NAME`).	June 17, 2024
Updated disk space	The `ARM Small` and `ARM Large` compute types now have increased disk space.	June 4, 2024

New content: GitHub manual webhooks	Add support for GitHub manual webhooks.	May 23, 2024
Updated content: Reserved capacity	CodeBuild now supports VPC connectivity for reserved capacity fleets on Amazon Linux.	May 15, 2024
Updated content: Lambda compute images	Add Lambda support for .NET 8 (`al-lambda/aarch64/dotnet8` and `al-lambda/x86_64/dotnet8`)	May 8, 2024
Updated quota: Build timeout	Update maximum build timeout quota to 2160 minutes (36 hours).	May 1, 2024
Updated content: AWS managed (predefined) policies for AWS CodeBuild	The AWSCodeBuildAdminAccess, AWSCodeBuildDeveloperAccess, and AWSCodeBuildReadOnlyAccess policies have been updated to reflect the AWS CodeConnections rebranding.	April 30, 2024
New content: Bitbucket app password or access token	Add support for Bitbucket access tokens.	April 11, 2024
New content: Auto-discover reports in CodeBuild	CodeBuild now supports report auto-discover.	April 4, 2024
New content: Self-hosted GitHub Actions runners	Add new content for self-hosted GitHub Actions runners	April 2, 2024
New content: GitLab connections	Add support for GitLab and GitHub Self Managed connections.	March 25, 2024

New content: Add new webhook events and filter types	Add support for new webhook events (`RELEASED` and `PRERELEASED`) and filter types (`TAG_NAME` and `RELEASE_NAME`).	March 15, 2024
New content: Add a new webhook event: `PULL_REQUEST_CLOSED`	Add support for a new webhook event: `PULL_REQUEST_CLOSED`.	February 20, 2024
Updated content: Docker images provided by CodeBuild	Add support for Windows Server Core 2019 (`windows-base:2019-3.0`)	February 7, 2024
Updated content: Docker images provided by CodeBuild	Add support for new runtimes for Amazon Linux 2023 (`al2/aarch64/standard/3.0`)	January 29, 2024
New content: Reserved capacity	CodeBuild now supports reserved capacity fleets in CodeBuild.	January 18, 2024
New compute type	CodeBuild now supports a Linux XLarge compute type. For more information, see Build environment compute types.	January 8, 2024
Updated content: Docker images provided by CodeBuild	Add support for new runtimes for Amazon Linux 2 (`al2/standard/5.0`) and Ubuntu (`ubuntu/standard/7.0`)	December 14, 2023
Updated content: Docker images provided by CodeBuild	Add support for new Lambda compute images	December 8, 2023

New content: AWS Lambda compute	Add new content for the AWS Lambda compute	November 6, 2023
Updated content: Docker images provided by CodeBuild	Add support for Amazon Linux 2 (`al2/standard/5.0`)	May 17, 2023
Changes to managed policies for CodeBuild	Details about updates to AWS managed politices for CodeBuild are now available. For more information, see CodeBuild updates to AWS managed policies.	May 16, 2023
Updated content: Docker images provided by CodeBuild	Remove support for Amazon Linux 2 (`al2/standard/3.0`) and add support for Amazon Linux 2 (`al2/standard/corretto8`) and Amazon Linux 2 (`al2/standard/corretto11`)	May 9, 2023
Updated content: Docker images provided by CodeBuild	Add support for Ubuntu 22.04 (`ubuntu/standard/7.0`)	April 13, 2023
Updated content: Docker images provided by CodeBuild	Remove support for Ubuntu 18.04 (`ubuntu/standard/4.0`) and Amazon Linux 2 (`al2/aarch64/standard/1.0`)	March 31, 2023

Updated content: Remove VPC limitation	Removing the following limitation: If you configure CodeBuild to work with a VPC, local caching is not supported. Starting 02/28/22, your VPC build will take longer since a new Amazon EC2 instance will be used for each build.	March 1, 2023
Updated content: Docker images provided by CodeBuild	Remove support for Ubuntu 18.04 (`ubuntu/standard/3.0`) and Amazon Linux 2 (`al2/standard/2.0`)	June 30, 2022
Amazon ECR Sample: Restrict image access	When CodeBuild credentials are used to pull an Amazon ECR image, you can restrict image access to a specific CodeBuild project. For more information, see Amazon ECR sample.	March 10, 2022
Added region support	The `ARM_CONTAINER` compute type is now supported in the following additional regions: Asia Pacific (Seoul), Canada (Central), Europe (London), and Europe (Paris). For more information, see Build environment compute types.	March 10, 2022

New VPC limitation	If you configure CodeBuild to work with a VPC, local caching is not supported. Starting 02/28/22, your VPC build will take longer since a new Amazon EC2 instance will be used for each build.	February 25, 2022
Batch report mode	CodeBuild now allows you to select how batch build statuses are sent to the source provider for a project. For more information, see Batch report mode.	October 4, 2021
New compute type	CodeBuild now supports a small ARM compute type. For more information, see Build environment compute types.	September 13, 2021
Public build projects	CodeBuild now allows you to make the build results for your build projects available to the public without requiring access to an AWS account. For more information, see Public build projects.	August 11, 2021
Session debugging for batch builds	CodeBuild now supports session debugging for batch builds. For more information, see build-graph and build-list.	March 3, 2021

Project level concurrent build limit	CodeBuild now allows you to limit the number of concurrent builds for a build project. For more information, see Project configuration and concurrentBuildLimit.	February 16, 2021
New buildspec property: s3-prefix	CodeBuild now provides the `s3-prefix` buildspec property for artifacts that allows you to specify a path prefix for artifacts that are uploaded to Amazon S3. For more information, see s3-prefix.	February 9, 2021
New buildspec property: on-failure	CodeBuild now provides the `on-failure` buildspec property for build phases that allows you to determine what happens when a build phase fails. For more information, see on-failure.	February 9, 2021
New buildspec property: exclude-paths	CodeBuild now provides the `exclude-paths` buildspec property for artifacts that allows you to exclude paths from your build artifacts. For more information, see exclude-paths.	February 9, 2021

New buildspec property: enable-symlinks	CodeBuild now provides the `enable-symlinks` buildspec property for artifacts that allows you to preserve symbolic links in a ZIP artifact. For more information, see enable-symlinks.	February 9, 2021
Buildspec artifact name enhancement	CodeBuild now allows the `artifacts/name` property to contain path information. For more information, see name.	February 9, 2021
Code coverage reporting	CodeBuild now provides code coverage reports. For more information, see Code coverage reports.	July 30, 2020
Batch builds	CodeBuild now supports running concurrent and coordinated builds of a project. For more information, see Batch builds in CodeBuild.	July 30, 2020
Windows Server 2019 image	CodeBuild now provides a Windows Server Core 2019 build image. For more information, see Docker images provided by CodeBuild .	July 20, 2020

Session Manager	CodeBuild now allows you to pause a running build and then use AWS Systems Manager Session Manager to connect to the build container and view the state of the container. For more informati on, see Session Manager.	July 20, 2020
Updated topic	CodeBuild now supports specifying a shell to use in their build environments in the buildspec file. For more information, see Build specification reference.	June 25, 2020
Test reporting with test frameworks	Added several topics the describe how to generate CodeBuild test reports with several test frameworks. For more information, see Test reporting with test framework s.	May 29, 2020
Updated topics	CodeBuild now supports adding tags to report groups. For more information, see ReportGroup.	May 21, 2020
Support for test reporting	CodeBuild support for test reporting is now generally available.	May 21, 2020

Updated topics	CodeBuild now supports creating create webhook filters for Github and Bitbucket that trigger builds only when the head commit message matches the specified expression. For more information, see GitHub pull request and webhook filter sample and Bitbucket pull request and webhook filter sample.	May 6, 2020
New topics	CodeBuild now supports sharing build project and report group resources. For more information, see Working with shared projects and Working with shared report groups.	December 13, 2019
New and updated topics	CodeBuild now supports test reporting during the run of a build project. For more information, see Working with test reporting, Create a test report, and Create a test report using the AWS CLI sample.	November 25, 2019
Updated topic	CodeBuild now supports Linux GPU and Arm environme nt types, and the `2xlarge` compute type. For more information, see Build environment compute types.	November 19, 2019

Updated topics	CodeBuild now supports build numbers on all builds, exporting environment variables, and AWS Secrets Manager integration. For more information, see Exported variables and Secrets Manager in Buildspec syntax.	November 6, 2019
New topic	CodeBuild now supports notification rules. You can use notification rules to notify users of important changes in build projects. For more information, see Create a notification rule.	November 5, 2019
Updated topics	CodeBuild now supports the Android version 29 and Go version 1.13 runtimes. For more information, see Docker images provided by CodeBuild and Buildspec syntax.	September 10, 2019
Updated topics	When you create a project, you can now choose the Amazon Linux 2 (AL2) managed image. For more information, see Docker images provided by CodeBuild and Runtime versions in buildspec file sample for CodeBuild.	August 16, 2019

Updated topic	When you create a project, you can now choose to disable encryption of S3 logs and, if you use a Git-based source repository, include Git submodules. For more information, see Create a build project in CodeBuild.	March 8, 2019
New topic	CodeBuild now supports local caching. You can specify local caching in one or more of four modes when you create a build. For more informati on, see Build caching in CodeBuild.	February 21, 2019
New topics	CodeBuild now supports webhook filter groups to specify events that trigger a build. For more information, see Filter GitHub webhook events and Filter Bitbucket webhook events.	February 8, 2019
New topic	The CodeBuild User Guide now shows how to use CodeBuild with a proxy server. For more information, see Use CodeBuild with a proxy server.	February 4, 2019

Updated topics	CodeBuild now supports using an Amazon ECR image that is in another AWS account. Several topics have been updated to reflect this change, including Amazon ECR sample for CodeBuild, Create a build project, and Create a CodeBuild service role.	January 24, 2019
Support for private Docker registries	CodeBuild now supports using a Docker image that is stored in a private registry as your runtime environme nt. For more information, see Private registry with AWS Secrets Manager sample.	January 24, 2019
Updated topic	CodeBuild now supports using an access token to connect to GitHub (with a personal access token) and Bitbucket (with an app password) repositories. For more information, see Create a build project (console) and Use access tokens with your source provider.	December 6, 2018
Updated topic	CodeBuild now supports new build metrics that measure the duration of each phase in a build. For more information, see CodeBuild CloudWatch metrics.	November 15, 2018

VPC endpoint policy topic	Amazon VPC endpoints for CodeBuild now support policies. For more informati on, see Create a VPC endpoint policy for CodeBuild .	November 9, 2018
Updated content	Topics have been updated to reflect the new console experience.	October 30, 2018
Amazon EFS sample	CodeBuild can mount an Amazon EFS file system during a build using commands in a project's buildspec file. For more information, see Amazon EFS sample for CodeBuild.	October 26, 2018
Bitbucket webhooks	CodeBuild now supports webhooks when you use Bitbucket for your repositor y. For more information, see Bitbucket pull request sample for CodeBuild.	October 2, 2018
S3 logs	CodeBuild now supports build logs in an S3 bucket. Previousl y, you could only build logs using CloudWatch Logs. For more information, see Create project.	September 17, 2018

Multiple inpout sources and multiple output artifacts	CodeBuild now supports projects that use more than one input source and publish more than one set of artifacts . For more information, see Multiple input sources and input artifacts sample and CodePipeline integration with CodeBuild and multiple input sources and output artifacts sample.	August 30, 2018
Semantic versioning sample	The CodeBuild User Guide now has a use case-base d sample that demonstra tes how to use semantic versioning to create artifact names at build time. For more information, see Use semantic versioning to name build artifacts sample.	August 14, 2018
New static website sample	The CodeBuild User Guide now has a use case-based sample that demonstrates how to host build output in an S3 bucket. The sample takes advantage of the recent support of unencrypted build artifacts. For more informati on, see Create a static website with build output hosted in an S3 bucket.	August 14, 2018

Support for overriding an artifact name with semantic versioning	You can now use semantic versioning to specify a format that CodeBuild uses to name build artifacts. This is useful because a build artifact with a hard-coded name overwrites previous build artifacts that use the same hard-coded name. For example, if a build is triggered multiple times a day, you can now add a timestamp to its artifact name. Each build artifact name is unique and does not overwrite the artifacts of previous builds.	August 7, 2018
Support of unencrypted build artifacts	CodeBuild now supports builds with unencrypted build artifacts. For more information, see Create a build project (console).	July 26, 2018
Support for Amazon CloudWatch metrics and alarms	CodeBuild now provides integration with CloudWatch metrics and alarms. You can use the CodeBuild or CloudWatch console to monitor builds at the project and account level. For more information, see Monitoring builds.	July 19, 2018

Support for reporting a build's status	CodeBuild can now report the status of a build's start and completion to your source provider. For more informati on, see Create a build project in CodeBuild.	July 10, 2018
Environment variables added to CodeBuild documentation	The Environment variables in build environments page was updated with the CODEBUILD _BUILD_ID, CODEBUILD _LOG_PATH, and CODEBUILD _START_TIME environment variables.	July 9, 2018
Support for a `finally` block in the buildspec file	The CodeBuild documenta tion was updated with details about the optional `finally` block in a buildspec file. Commands in the finally block always run after the commands in its correspon ding commands block. For more information, see Buildspec syntax.	June 20, 2018
CodeBuild agent update notifications	The CodeBuild documenta tion was updated with details about how you can use Amazon SNS to be notified when new versions of the CodeBuild agent are released. For more information, see Receive notifications for new AWS CodeBuild agent versions.	June 15, 2018

Earlier updates

The following table describes important changes in each release of the *AWS CodeBuild User Guide* before June 2018.

Change	Description	Date
Support for Windows builds	CodeBuild now supports builds for the Microsoft Windows Server platform, including a prepackaged build environment for the .NET Core 2.0 on Windows. For more information, see Run Microsoft Windows samples for CodeBuild .	May 25, 2018
Support for build idempotency	When you run the `start-build` command with the AWS Command Line Interface (AWS CLI), you can specify that the build is idempotent. For more information, see Run a build (AWS CLI) .	May 15, 2018
Support for overriding more build project settings	You can now override more build project settings when you create a build. The overrides are only for that build. For more information, see Run AWS CodeBuild builds manually .	May 15, 2018
VPC Endpoint support	You can now use VPC endpoints to improve the security of your builds. For	March 18, 2018

Change	Description	Date
	more information, see Use VPC endpoints .	
Support of triggers	You can now create triggers to schedule builds at regular frequencies. For more information, see Create AWS CodeBuild triggers .	March 28, 2018
FIPS endpoints documentation	You can now learn about how to use the AWS Command Line Interface (AWS CLI) or an AWS SDK to tell CodeBuild to use one of four Federal Information Processing Standards (FIPS) endpoints . For more information, see Specify the AWS CodeBuild endpoint .	March 28, 2018
AWS CodeBuild available in Asia Pacific (Mumbai), Europe (Paris), and South America (São Paulo)	AWS CodeBuild is now available in the Asia Pacific (Mumbai), Europe (Paris), and South America (São Paulo) regions. For more information, see AWS CodeBuild in the *Amazon Web Services General Reference*.	March 28, 2018
GitHub Enterprise Server support	CodeBuild can now build from source code stored in a GitHub Enterprise Server repository. For more information, see Run the GitHub Enterprise Server sample .	January, 25, 2018

Change	Description	Date
Git clone depth support	CodeBuild now supports the creation of a shallow clone with a history truncated to the specified number of commits. For more informati on, see Create a build project .	January, 25, 2018
VPC support	VPC-enabled builds are now able to access resources inside your VPC. For more informati on, see VPC support .	November, 27, 2017
Dependency caching support	CodeBuild now supports the dependency caching. This allows CodeBuild to save certain reusable pieces of the build environment in the cache and use this across builds.	November, 27, 2017
Build badges support	CodeBuild now supports the use of build badges, which provide an embeddable, dynamically generated image (badge) that displays the status of the latest build for a project. For more information, see Build badges sample .	November 27, 2017

Change	Description	Date
AWS Config integration	AWS Config now supports CodeBuild as an AWS resource, which means the service can track your CodeBuild projects. For more information about AWS Config, see AWS Config sample .	October 20, 2017
Automatically rebuild updated source code in GitHub repositories	If your source code is stored in a GitHub repository, you can enable AWS CodeBuild to rebuild your source code whenever a code change is pushed to the repository. For more information, see Run the GitHub pull request and webhook filter sample .	September 21, 2017

Change	Description	Date
New ways for storing and retrieving sensitive or large environment variables in Amazon EC2 Systems Manager Parameter Store	You can now use the AWS CodeBuild console or the AWS CLI to retrieve sensitive or large environment variables stored in Amazon EC2 Systems Manager Parameter Store. You can also now use the AWS CodeBuild console to store these types of environment variables in Amazon EC2 Systems Manager Parameter Store. Previously, you could only retrieve these types of environment variables by including them in a buildspec or by running build commands to automate the AWS CLI. You could only store these types of environme nt variables by using the Amazon EC2 Systems Manager Parameter Store console. For more informati on, see Create a build project , Change build project settings , and Run builds manually .	September 14, 2017
Build deletion support	You can now delete builds in AWS CodeBuild. For more information, see Delete builds .	August 31, 2017

Change	Description	Date
Updated way to retrieve sensitive or large environment variables stored in Amazon EC2 Systems Manager Parameter Store by using a buildspec	AWS CodeBuild now makes it easier to use a buildspec to retrieve sensitive or large environment variables stored in Amazon EC2 Systems Manager Parameter Store. Previously, you could only retrieve these types of environment variables by running build commands to automate the AWS CLI. For more information, see the `parameter-store` mapping in Buildspec syntax .	August 10, 2017
AWS CodeBuild supports Bitbucket	CodeBuild can now build from source code stored in a Bitbucket repository. For more information, see Create a build project and Run builds manually .	August 10, 2017
AWS CodeBuild available in US West (N. California), Europe (London), and Canada (Central)	AWS CodeBuild is now available in the US West (N. California), Europe (London), and Canada (Central) regions. For more information, see AWS CodeBuild in the *Amazon Web Services General Reference*.	June 29, 2017

Change	Description	Date
Alternate buildspec file names and locations supported	You can now specify an alternate file name or location of a buildspec file to use for a build project, instead of a default buildspec file named `buildspec.yml` at the root of the source code. For more information, see Buildspec file name and storage location .	June 27, 2017
Updated build notifications sample	CodeBuild now provides built-in support for build notifications through Amazon CloudWatch Events and Amazon Simple Notification Service (Amazon SNS). The previous Build notifications sample has been updated to demonstrate this new behavior.	June 22, 2017
Docker in custom image sample added	A sample showing how to use CodeBuild and a custom Docker build image to build and run a Docker image has been added. For more information, see the Docker in custom image sample .	June 7, 2017

Change	Description	Date
Fetch source code for GitHub pull requests	When you run a build with CodeBuild that relies on source code stored in a GitHub repository, you can now specify a GitHub pull request ID to build. You can also specify a commit ID, a branch name, or a tag name instead. For more informati on, see the **Source version** value in Run a build (console) or the `sourceVersion` value in Run a build (AWS CLI) .	June 6, 2017
Build specification version updated	A new version of the buildspec format has been released. Version 0.2 addresses the issue of CodeBuild running each build command in a separate instance of the default shell. Also in version 0.2, `environment_variab les` is renamed to `env`, and `plaintext` is renamed to `variables` . For more information, see Build specification reference for CodeBuild .	May 9, 2017

Change	Description	Date
Dockerfiles for build images available in GitHub	Definitions for many of the build images provided by AWS CodeBuild are available as Dockerfiles in GitHub. For more information, see the Definition column of the table in Docker images provided by CodeBuild .	May 2, 2017
AWS CodeBuild available in Europe (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Sydney), and Asia Pacific (Tokyo)	AWS CodeBuild is now available in the Europe (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Sydney), and Asia Pacific (Tokyo) regions. For more information, see AWS CodeBuild in the *Amazon Web Services General Reference*.	March 21, 2017
CodePipeline test action support for CodeBuild	You can now add to a pipeline in CodePipeline a test action that uses CodeBuild. For more information, see Add a CodeBuild test action to a pipeline (CodePipeline console) .	March 8, 2017

Change	Description	Date
Buildspec files support fetching build output from within selected top-level directories	Buildspec files now enable you to specify individual top-level directories whose contents you can instruct CodeBuild to include in build output artifacts. You do this by using the `base-directory` mapping. For more information, see Buildspec syntax .	February 8, 2017
Built-in environment variables	AWS CodeBuild provides additional built-in environment variables for your builds to use. These include environment variables describing the entity that started the build, the URL to the source code repository, the source code's version ID, and more. For more information, see Environment variables in build environments .	January 30, 2017
AWS CodeBuild available in US East (Ohio)	AWS CodeBuild is now available in the US East (Ohio) region. For more information, see AWS CodeBuild in the *Amazon Web Services General Reference*.	January 19, 2017

Change	Description	Date
Shell and command behaviors information	CodeBuild runs each command you specify in a separate instance of a build environment's default shell. This default behavior can produce some unexpecte d side effects for your commands. We recommend some approaches to work around this default behavior if needed. For more informati on, see Shells and commands in build environments .	December 9, 2016
Environment variables information	CodeBuild provides several environment variables that you can use in your build commands. You can also define your own environment variables. For more informati on, see Environment variables in build environments .	December 7, 2016
Troubleshooting topic	Troubleshooting informati on is now available. For more information, see Troublesh ooting AWS CodeBuild .	December 5, 2016
Jenkins plugin initial release	This is the initial release of the CodeBuild Jenkins plugin. For more information, see Use AWS CodeBuild with Jenkins .	December 5, 2016
User Guide initial release	This is the initial release of the *CodeBuild User Guide*.	December 1, 2016