# Table of Contents

What is AWS CodeBuild? .......................................................................................................................... 1  
How to run CodeBuild ............................................................................................................................. 1  
Pricing for CodeBuild ............................................................................................................................ 2  
How do I get started with CodeBuild? .................................................................................................... 2  
Concepts .................................................................................................................................................... 2  
  How CodeBuild works .......................................................................................................................... 3  
  Next steps .............................................................................................................................................. 3  
Getting started ......................................................................................................................................... 5  
Getting started using the console ........................................................................................................... 5  
  Steps ..................................................................................................................................................... 5  
  Step 1: Create two S3 buckets .............................................................................................................. 5  
  Step 2: Create the source code .......................................................................................................... 6  
  Step 3: Create the buildspec file ........................................................................................................ 8  
  Step 4: Upload the source code and the buildspec file .................................................................... 9  
  Step 5: Create the build project ....................................................................................................... 10  
  Step 6: Run the build ......................................................................................................................... 11  
  Step 7: View summarized build information ................................................................................. 12  
  Step 8: View detailed build information .......................................................................................... 13  
  Step 9: Get the build output artifact ............................................................................................... 13  
  Step 10: Delete the S3 input bucket ................................................................................................. 14  
  Wrapping up ......................................................................................................................................... 14  
Getting started using the AWS CLI ....................................................................................................... 15  
  Steps ..................................................................................................................................................... 15  
  Step 1: Create two S3 buckets .............................................................................................................. 15  
  Step 2: Create the source code .......................................................................................................... 16  
  Step 3: Create the buildspec file ........................................................................................................ 18  
  Step 4: Upload the source code and the buildspec file .................................................................... 19  
  Step 5: Create the build project ....................................................................................................... 20  
  Step 6: Run the build ......................................................................................................................... 23  
  Step 7: View summarized build information ................................................................................. 24  
  Step 8: View detailed build information .......................................................................................... 26  
  Step 9: Get the build output artifact ............................................................................................... 27  
  Step 10: Delete the S3 input bucket ................................................................................................. 28  
  Wrapping up ......................................................................................................................................... 28  
Samples ....................................................................................................................................................... 29  
Windows samples ..................................................................................................................................... 30  
  Running the samples .......................................................................................................................... 30  
  Directory structure ............................................................................................................................... 31  
  Files ..................................................................................................................................................... 32  
Use case-based samples .......................................................................................................................... 48  
  Access token sample .......................................................................................................................... 49  
  Amazon ECR sample .......................................................................................................................... 53  
  Amazon EFS sample .......................................................................................................................... 56  
  AWS CodeDeploy sample ................................................................................................................. 59  
  AWS CodePipeline integration with multiple input sources and output artifacts sample ............... 63  
  AWS Config sample ........................................................................................................................... 65  
  AWS Elastic Beanstalk sample ......................................................................................................... 67  
  AWS Lambda sample .......................................................................................................................... 74  
  Bitbucket pull request and webhook filter sample ............................................................................ 75  
  Build badges sample .......................................................................................................................... 82  
  Build notifications sample ................................................................................................................. 85  
  Create a test report using the AWS CLI sample .............................................................................. 102  
  Docker in custom image sample ....................................................................................................... 107  
  Docker sample ................................................................................................................................... 109
Troubleshooting ............................................................................................................................. 359

Apache Maven builds reference artifacts from the wrong repository ........................................ 360
Build commands run as root by default ...................................................................................... 361
Builds might fail when file names have non-U.S. English characters ......................................... 361
Builds might fail when getting parameters from Amazon EC2 Parameter Store ......................... 361
Cannot access branch filter in the CodeBuild console ................................................................ 362
Cannot view build success or failure .......................................................................................... 362
Cannot view build success or failure .......................................................................................... 362
Cannot find and select the base image of the Windows Server Core 2016 platform ..................... 363
Earlier commands in buildspec files are not recognized by later commands .................................. 363
Error: "Access denied" when attempting to download cache .......................................................... 364
Error: "BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE" when using a custom build image .......... 364
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. Error: Code: 500" ........................ 365
Error: "Cannot connect to the Docker daemon" when running a build ........................................ 365
Error: "CodeBuild is experiencing an issue" when running a build ............................................ 366
Error: "CodeBuild is not authorized to perform: sts:AssumeRole" when creating or updating a build project .................................................................................................................................. 366
Error: "Error calling s3:GetBucketAcl: Either the bucket owner has changed or the service role no longer has permission to called s3:GetBucketAcl" .................................................................................. 367
Error: "Failed to upload artifacts: Invalid arn" when running a build ........................................... 367
Error: "Git clone failed: Unable to access 'your-repository-URL': SSL certificate problem: Self signed certificate" .................................................................................................................. 367
Error: "The bucket you are attempting to access must be addressed using the specified endpoint" when running a build .................................................................................................................. 368
Error: "The policy's default version was not created by enhanced zero click role creation or was not the most recent version created by enhanced zero click role creation." ........................................ 368
Error: "This build image requires selecting at least one runtime version." .................................... 369
Error: "QUEUED: INSUFFICIENT_SUBNET" when a build in a build queue fails ........................... 369
Error: "Unable to download cache: RequestError: Send request failed caused by: x509: Failed to load system roots and no roots provided" ....................................................................................... 370
Error: "Unable to download certificate from S3. AccessDenied" ................................................. 370
Error: "Unable to locate credentials" .......................................................................................... 370
RequestError timeout error when running CodeBuild in a proxy server ....................................... 371
The bourne shell (sh) must exist in build images ........................................................................ 372
Warning: "Skipping install of runtimes. runtime version selection is not supported by this build image" when running a build .................................................................................................................................. 372

Quotas .......................................................................................................................................... 374
Build projects .............................................................................................................................. 374
Builds .......................................................................................................................................... 374
Reports ........................................................................................................................................ 375

Third party notices for AWS CodeBuild for Windows ................................................................ 376
1) base Docker image—windowsservercore .............................................................................. 376
2) windows-base Docker image—choco .................................................................................... 377
3) windows-base Docker image—git --version 2.16.2 .............................................................. 377
4) windows-base Docker image—microsoft-build-tools --version 15.0.26320.2 .......................... 377
5) windows-base Docker image—nuget.commandline --version 4.5.1 ....................................... 377
6) windows-base Docker image—netfx-4.6.2-devpack ............................................................. 378
7) windows-base Docker image—visualsharptools, v 4.0 .......................................................... 381
9) windows-base Docker image—netfx-pcl-reference-assemblies-4.6 ......................................... 382
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](https://aws.amazon.com/codebuild).

**Topics**

- How to run CodeBuild (p. 1)
- Pricing for CodeBuild (p. 2)
- How do I get started with CodeBuild? (p. 2)
- AWS CodeBuild concepts (p. 2)

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.

To run CodeBuild by using the CodeBuild console, AWS CLI, or AWS SDKs, see [Run AWS CodeBuild directly](https://docs.aws.amazon.com/codebuild/latest/userguide/running-codebuild.html) (p. 177).

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 AWS CodePipeline with AWS CodeBuild (p. 195). 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 (p. 326).

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 (p. 2).
2. **Experiment** with CodeBuild in an example scenario by following the instructions in Getting started using the console (p. 5).
3. **Use** CodeBuild in your own scenarios by following the instructions in Plan a build (p. 148).

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 (p. 215)
   - Build environment reference (p. 165)
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 (p. 149).
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 (p. 85).
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 (p. 5).
2. **Use** CodeBuild in your own scenarios by following the instructions in Plan a build (p. 148).
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.

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 AWS CodePipeline with AWS CodeBuild (p. 195). For information about using the AWS SDKs, see Run AWS CodeBuild directly (p. 177).

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.

Steps

- Step 1: Create two S3 buckets (p. 5)
- Step 2: Create the source code (p. 6)
- Step 3: Create the buildspec file (p. 8)
- Step 4: Upload the source code and the buildspec file (p. 9)
- Step 5: Create the build project (p. 10)
- Step 6: Run the build (p. 11)
- Step 7: View summarized build information (p. 12)
- Step 8: View detailed build information (p. 13)
- Step 9: Get the build output artifact (p. 13)
- Step 10: Delete the S3 input bucket (p. 14)
- Wrapping up (p. 14)

Step 1: Create two S3 buckets

(Part of: Getting started with AWS CodeBuild using the console (p. 5))
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 (p. 148).

Next step

Step 2: Create the source code (p. 6)

**Step 2: Create the source code**

(Previous step: Step 1: Create two S3 buckets (p. 5))

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.

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

3. Create this file, name it \texttt{TestMessageUtil.java}, and then save it in the \texttt{/src/test/java} directory.

```java
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 \texttt{message} variable in the \texttt{MessageUtil} class to \texttt{Robert}. It then tests to see if the \texttt{message} variable was successfully set by checking whether the strings \texttt{Robert} and \texttt{Hi!Robert} appear in the output.

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

```xml
<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>
```
Step 3: Create the buildspec file

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 (p. 150).

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 (p. 149).

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
```

Next step

Step 4: Upload the source code and the buildspec file (p. 9)

**Step 4: Upload the source code and the buildspec file**

(Previous step: Step 3: Create the buildspec file (p. 8))
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 (p. 149).

**Next step**

Step 5: Create the build project (p. 10)

**Step 5: Create the build project**

(Previous step: Step 4: Upload the source code and the buildspec file (p. 9))

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**

Step 6: Run the build

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 IAM 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 IAM user in your AWS account. For more information, see Creating Your First IAM Admin User and Group in the IAM User Guide.
   - An IAM user in your AWS account with the AWSCodeBuildAdminAccess, AmazonS3ReadOnlyAccess, and IAMFullAccess managed policies attached to that IAM user or to an IAM group that the IAM user belongs to. If you do not have an IAM user or group in your AWS account with these permissions, and you cannot add these permissions to your IAM user or group, contact your AWS account administrator for assistance. For more information, see AWS managed (predefined) policies for AWS CodeBuild (p. 311).

   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 (p. 321).

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 2.

10. For Runtime(s), choose Standard.

11. For Image, choose aws/codebuild/amazonlinux2-x86_64-standard:2.0.

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.

Next step

Step 6: Run the build (p. 11)
In this step, you instruct AWS CodeBuild to run the build with the settings in the build project.

**To run the build**

2. In the navigation pane, choose **Build projects**.
3. In the list of build projects, choose **codebuild-demo-project**, and then choose **Start build**.
4. On the **Start build** page, choose **Start build**.

**Next step**

*Step 7: View summarized build information (p. 12)*

**Step 7: View summarized build information**

(Previous step: **Step 6: Run the build (p. 11)**)

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 details 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.

**Next step**

*Step 8: View detailed build information (p. 13)*
Step 8: View detailed build information

(Previous step: Step 7: View summarized build information (p. 12))

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 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 (p. 304).

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.

Next step

Step 9: Get the build output artifact (p. 13)

Step 9: Get the build output artifact

(Previous step: Step 8: View detailed build information (p. 13))

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, in Build Status, choose the View artifacts link. This opens the folder in Amazon S3 for the build output artifact. (If the build details page is not displayed, in the navigation bar, choose Build history, and then choose the Build run link.)
2. Open the target folder, 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/.
Step 10: Delete the S3 input 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.

Next step

Step 10: Delete the S3 input bucket (p. 14)

Step 10: Delete the S3 input bucket

(Previous step: Step 9: Get the build output artifact (p. 13))

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

If you are using the IAM user or an administrator IAM user to delete this bucket, 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 STATEMENTS HERE ###
    ]
}
```

Next step

Wrapping up (p. 14)

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 (p. 148). If you don’t feel ready yet, you might want to try building some of the samples. For more information, see Samples (p. 29).
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 AWS CodePipeline with AWS CodeBuild (p. 195). For information about using the AWS SDKs, see Run AWS CodeBuild directly (p. 177).

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.

Steps

- Step 1: Create two S3 buckets (p. 15)
- Step 2: Create the source code (p. 16)
- Step 3: Create the buildspec file (p. 18)
- Step 4: Upload the source code and the buildspec file (p. 19)
- Step 5: Create the build project (p. 20)
- Step 6: Run the build (p. 23)
- Step 7: View summarized build information (p. 24)
- Step 8: View detailed build information (p. 26)
- Step 9: Get the build output artifact (p. 27)
- Step 10: Delete the S3 input bucket (p. 28)
- Wrapping up (p. 28)

Step 1: Create two S3 buckets

(Part of: Getting started with AWS CodeBuild using the AWS CLI (p. 15))

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 (p. 148).

## Next step

Next step

Step 2: Create the source code (p. 16)

### Step 2: Create the source code

(Previous step: Step 1: Create two S3 buckets (p. 15))

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.

```java
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.

```java
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.
Step 3: Create the buildspec file

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.

```yaml
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 (p. 150).

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.
Step 4: Upload the source code and the buildspec file

- **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.

  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](p. 149).

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
```

**Next step**

**Step 4: Upload the source code and the buildspec file**

(Previous step: Step 3: Create the buildspec file (p. 18))

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
```
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:

```json
{
    "name": "codebuild-demo-project",
    "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "artifacts": {
```

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 (p. 149).

Next step

Step 5: Create the build project (p. 20)
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 Create a CodeBuild service role (p. 348).

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 (LINUX_CONTAINER is currently the only allowed value).
- 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:2.0` for a Docker image in the CodeBuild Docker images repository). `aws/codebuild/standard` is the name of the Docker image. `1.0` is the tag of the Docker image.

To find more Docker images you can use in your scenarios, see the Build environment reference (p. 165).
- 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) (p. 229).

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

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

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

```json
{
  "name": "codebuild-demo-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:2.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "serviceIAMRole"
}
```
Step 5: Create the build project

```
"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:2.0",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
    },
    "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "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 AWS KMS customer master key (CMK) 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 IAM 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 IAM user in your AWS account. For more information, see Creating Your First IAM Admin User and Group in the IAM User Guide.
- An IAM user in your AWS account with the AWSCodeBuildAdminAccess, AmazonS3ReadOnlyAccess, and IAMFullAccess managed policies attached to that IAM user or to an IAM group that the IAM user belongs to. If you do not have an IAM user or group in your AWS account with these permissions, and you cannot add these permissions...
to your IAM user or group, contact your AWS account administrator for assistance. For more information, see AWS managed (predefined) policies for AWS CodeBuild (p. 311).

Next step

Step 6: Run the build (p. 23)

Step 6: Run the build

(Previous step: Step 5: Create the build project (p. 20))

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:

   ```bash
   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:

   ```json
   {
     "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:2.0",
         "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.
Next step

Step 7: View summarized build information

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"
      }
    }
  ],
  "region": "region-ID",
  "status": "SUCCESS"
}
```

Make a note of the `id` value. You need it in the next step.
```json
"sha256sum": "SHA-256-hash",
"projectName": "codebuild-demo-project",
"timeoutInMinutes": 60,
"initiator": "user-name",
"buildStatus": "SUCCEEDED",
"environment": {
  "computeType": "BUILD_GENERAL1_SMALL",
  "image": "aws/codebuild/standard:2.0",
  "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.

**Next step**

**Step 8: View detailed build information (p. 26)**

*API Version 2016-10-06*
Step 8: View detailed build information

(Previous step: Step 7: View summarized build information (p. 24))

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 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 (p. 304).

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-d52fbEXAMPLE).
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 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]
------------------------------------------------------------------------

...  
[Container] 2016/04/15 17:49:55 -------------------------------------------------------
[Container] 2016/04/15 17:49:55 -------------------------------------------------------
```
Step 9: Get the build output 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.

Next step

Step 9: Get the build output artifact (p. 27)
Step 10: Delete the S3 input bucket

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

If you are using the IAM user or an administrator IAM user to delete this bucket, 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.

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

Next step

Wrapping up (p. 28)

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 (p. 148). If you don't feel ready yet, you might want to try building some of the samples. For more information, see Samples (p. 29).
## CodeBuild samples

These use case-based samples can be used to experiment with AWS CodeBuild:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon ECR sample (p. 53)</td>
<td>Uses a Docker image in an Amazon ECR repository to use Apache Maven to produce a single JAR file.</td>
</tr>
<tr>
<td>AWS Elastic Beanstalk sample (p. 67)</td>
<td>Uses Apache Maven to produce a single WAR file. Uses Elastic Beanstalk to deploy the WAR file to an Elastic Beanstalk instance.</td>
</tr>
<tr>
<td>Amazon EFS sample (p. 56)</td>
<td>Shows how to configure a buildspec file so that a CodeBuild project mounts and builds on an Amazon EFS file system.</td>
</tr>
<tr>
<td>AWS Lambda sample (p. 74)</td>
<td>Uses CodeBuild, Lambda, AWS CloudFormation, and CodePipeline to build and deploy a serverless application that follows the AWS Serverless Application Model (AWS SAM) standard.</td>
</tr>
<tr>
<td>Bitbucket pull request and webhook filter sample (p. 75)</td>
<td>Uses CodeBuild with Bitbucket as the source repository and webhooks enabled, to rebuild the source code every time a code change is pushed to the repository.</td>
</tr>
<tr>
<td>Build badges sample (p. 82)</td>
<td>Shows how to set up CodeBuild with build badges.</td>
</tr>
<tr>
<td>Build notifications sample (p. 85)</td>
<td>Uses Apache Maven to produce a single JAR file. Sends a build notification to subscribers of an Amazon SNS topic.</td>
</tr>
<tr>
<td>AWS CodeDeploy sample (p. 59)</td>
<td>Uses Apache Maven to produce a single JAR file. Uses CodeDeploy to deploy the JAR file to an Amazon Linux instance. You can also use CodePipeline to build and deploy the sample.</td>
</tr>
<tr>
<td>AWS CodePipeline integration with multiple input sources and output artifacts sample (p. 63)</td>
<td>Shows how to use AWS CodePipeline to create a build with multiple input sources and multiple output artifacts.</td>
</tr>
<tr>
<td>Host build output in an S3 bucket (p. 130)</td>
<td>Shows how to create a static website in an S3 bucket using unencrypted build artifacts.</td>
</tr>
<tr>
<td>Create a test report in CodeBuild using the AWS CLI sample (p. 102)</td>
<td>Uses the AWS CLI to create, run, and view the results of a test report.</td>
</tr>
<tr>
<td>Docker in custom image sample (p. 107)</td>
<td>Uses a custom Docker image to produce a Docker image.</td>
</tr>
<tr>
<td>Docker sample (p. 109)</td>
<td>Uses a build image provided by CodeBuild with Docker support to produce a Docker image with Apache Maven. Pushes the Docker image to a repository in Amazon ECR. You can also adapt this sample to push the Docker image to Docker Hub.</td>
</tr>
</tbody>
</table>
Microsoft Windows samples for CodeBuild

These samples use an AWS CodeBuild build environment running Microsoft Windows Server 2016, the .NET Framework, and the .NET Core SDK to build executables file out of code written in C#, 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.

**Running the samples**

**To run these 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, run the build, and follow the steps in Run AWS CodeBuild directly (p. 177).
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.)

```json
{
  "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_CONTAINER",
    "image": "aws/codebuild/windows-base:1.0",
    "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. 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 executable and other files.

   - The executable file for the C# sample using the .NET Framework, `CSharpHelloWorld.exe`, can be found in the `CSharpHelloWorld\bin\Debug` directory.
   - The executable file for the F# sample using the .NET Framework, `FSharpHelloWorld.exe`, can be found in the `FSharpHelloWorld\bin\Debug` directory.
   - The executable file for the Visual Basic sample using the .NET Framework, `VBHelloWorld.exe`, can be found in the `VBHelloWorld\bin\Debug` directory.
   - The executable file for the C# sample using .NET Core, `HelloWorldSample.dll`, can be found in the `bin\Debug\netcoreapp1.0` directory.

## Directory structure

These samples assume the following directory structures.

### C# and the .NET Framework

```
(root directory name)
|-- buildspec.yml
|-- CSharpHelloWorld.sln
  `-- CSharpHelloWorld
     |-- App.config
     |-- CSharpHelloWorld.csproj
     |-- Program.cs
     `-- Properties
        `-- AssemblyInfo.cs
```

### F# and the .NET Framework

```
(root directory name)
```
Files

These samples use the following files.

C# and the .NET Framework

buildspec.yml (in $(root directory name)):

```yaml
version: 0.2

env:
  variables:
    SOLUTION: .\CSharpHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.6.2

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:
```

Visual Basic and the .NET Framework

(root directory name)

```yaml
|-- buildspec.yml
|-- VBHelloWorld.sln
 `-- VBHelloWorld
    |-- App.config
    |-- HelloWorld.vbproj
    `-- My Project
       |-- Application.Designer.vb
       |-- Application.myapp
       |-- AssemblyInfo.vb
       |-- Resources.Designer.vb
       |-- Resources.resx
       |-- Settings.Designer.vb
       `-- Settings.settings
```
Files

- .\CSharpHelloWorld\bin\Debug/*

CSharpHelloWorld.sln (in \root directory name):

```xml
  <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>{2F8752D5-E628-4A38-AA7E-BC4B4E697CBB}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>CSharpHelloWorld</RootNamespace>
    <AssemblyName>CSharpHelloWorld</AssemblyName>
    <TargetFrameworkVersion>v4.6.2</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU'">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
  </PropertyGroup>
</Project>
```

App.config (in \root directory name)\CSharpHelloWorld:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.2" />
  </startup>
</configuration>
```

CSharpHelloWorld.csproj (in \root directory name)\CSharpHelloWorld:

```xml
<?xml version="1.0" encoding="utf-8" ?>
  <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>{2F8752D5-E628-4A38-AA7E-BC4B4E697CBB}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>CSharpHelloWorld</RootNamespace>
    <AssemblyName>CSharpHelloWorld</AssemblyName>
    <TargetFrameworkVersion>v4.6.2</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU'">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
  </PropertyGroup>
</Project>
```
<DefineConstants>DEBUG;TRACE</DefineConstants>
<ErrorReport>prompt</ErrorReport>
<WarningLevel>4</WarningLevel>
</PropertyGroup>

<Project Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
  <PlatformTarget>AnyCPU</PlatformTarget>
  <DebugType>pdbonly</DebugType>
  <Optimize>true</Optimize>
  <OutputPath>bin\Release\</OutputPath>
  <DefineConstants>TRACE</DefineConstants>
  <ErrorReport>prompt</ErrorReport>
  <WarningLevel>4</WarningLevel>
</PropertyGroup>

<ItemGroup>
  <Reference Include="System" />
  <Reference Include="System.Core" />
  <Reference Include="System.Xml.Linq" />
  <Reference Include="System.Data.DataSetExtensions" />
  <Reference Include="Microsoft.CSharp" />
  <Reference Include="System.Data" />  
  <Reference Include="System.Net.Http" />
  <Reference Include="System.Xml" />
</ItemGroup>

<ItemGroup>
  <Compile Include="Program.cs" />
  <Compile Include="Properties\AssemblyInfo.cs" />
</ItemGroup>

<ItemGroup>
  <None Include="App.config" />
</ItemGroup>

<Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.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 Name="AfterBuild" />

-->
using System.Runtime.CompilerServices;
using 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("CSharpHelloWorld")]
[assembly: AssemblyDescription(() )]
[assembly: AssemblyConfiguration(() )]
[assembly: AssemblyCompany(() )]
[assembly: AssemblyProduct("CSharpHelloWorld")]
[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("2f8752d5-e628-4a38-aa7e-bc4b4e697cbb")]

// 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")]

F# and the .NET Framework

buildspec.yml (in [root directory name]):

version: 0.2
env:
  variables:
    SOLUTION: .\FSharpHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.6.2

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:
    - .\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("(F2A719FB-5D33-465A-A702-920D77279786") = "FSharpHelloWorld", "FSharpHelloWorld \FSSharpHelloWorld.fsproj", 

```html
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release|Any CPU = Release|Any CPU
EndGlobalSection
GlobalSection(ProjectConfigurationPlatforms) = postSolution
  {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
  {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.Build.0 = Debug|Any CPU
  {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.ActiveCfg = Release|Any CPU
  {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.Build.0 = Release|Any CPU
EndGlobalSection
GlobalSection(SolutionProperties) = preSolution
  HideSolutionNode = FALSE
EndGlobalSection
EndGlobal
```

**App.config** *(root directory name)\FSharpHelloWorld*:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.2" />
  </startup>
</configuration>
```

**AssemblyInfo.fs** *(root directory name)\FSharpHelloWorld*:

```fsharp
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: AssemblyFileVersion("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"?>
  <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>v4.6.2</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>
  </ItemGroup>
</Project>

API Version 2016-10-06

37
Files

<ItemGroup>
  <Compile Include="AssemblyInfo.fs" />
  <Compile Include="Program.fs" />
  <None Include="App.config" />
</ItemGroup>

<PropertyGroup>
</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: 4.6.2

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\#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-C01C-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=v4.6.2" />  
  </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"?>
  <Import Project="$(MSBuildExtensionsPath)$(MSBuildToolsVersion)\Microsoft.Common.props"
      Condition="Exists('$(MSBuildExtensionsPath)$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
  </PropertyGroup>
<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 Name="AfterBuild" />
-->
</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"?>
  <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("1.0.0.0")>
<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
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.Microsoft.VisualBasic.HideModuleNameAttribute()> _
Friend Module Resources

    Private resourceMan As Global.System.Resources.ResourceManager

    '''<summary>
    '''  Returns the cached ResourceManager instance used by this class.
    '''</summary>
    Get
        If Object.ReferenceEquals(resourceMan, Nothing) Then
            Dim temp As Global.System.Resources.ResourceManager = New
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>
    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
        : and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.soap.base64
value : The object must be serialized with
        : 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>
AWS CodeBuild User guide

Files

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

API Version 2016-10-06
45
Option Strict On
Option Explicit On

Namespace My

"11.0.0.0"), 
 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

         _
         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 If
         Return defaultInstance
      End Get
      End Property
End Class
End Namespace

Namespace My

<Global.Microsoft.VisualBasic.HideModuleNameAttribute(), 
AP1 Version 2016-10-06
46
Friend Module MySettingsProperty

   Friend ReadOnly Property Settings() As Global.VBHelloWorld.My.MySettings
   Get
   End Get
   End Property
End Module
End Namespace

Settings.settings (in (root directory name)\VBHelloWorld\My Project):

<?xml version='1.0' encoding='utf-8'?>
  CurrentProfile="(Default)" UseMySettingsClassName="true">
  <Profiles>
    <Profile Name="(Default)" />
  </Profiles>
  <Settings />
</SettingsFile>

C# and .NET Core

buildspec.yml (in (root directory name)

version: 0.2
phases:
  build:
    commands:
      - dotnet restore
      - dotnet build
artifacts:
  files:
    - .\bin\Debug\netcoreapp1.0\*

HelloWorldSample.csproj (in (root directory name)

<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp1.0</TargetFramework>
  </PropertyGroup>
</Project>

Program.cs (in (root directory name)

using System;
namespace HelloWorldSample
{
  public static class Program
  {
    public static void Main()
    {
      Console.WriteLine("Hello World!");
    }
}
# CodeBuild use case-based samples

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access token sample  (p. 49)</td>
<td>Shows how to use access tokens in CodeBuild to connect to GitHub and Bitbucket.</td>
</tr>
<tr>
<td>Amazon ECR sample (p. 53)</td>
<td>Uses a Docker image in an Amazon ECR repository to use Apache Maven to produce a single JAR file.</td>
</tr>
<tr>
<td>Amazon EFS sample (p. 56)</td>
<td>Shows how to configure a buildspec file so that a CodeBuild project mounts and builds on an Amazon EFS file system.</td>
</tr>
<tr>
<td>AWS CodeDeploy sample (p. 59)</td>
<td>Uses Apache Maven to produce a single JAR file. Uses CodeDeploy to deploy the JAR file to an Amazon Linux instance. You can also use CodePipeline to build and deploy the sample.</td>
</tr>
<tr>
<td>AWS CodePipeline integration with multiple input sources and output artifacts sample  (p. 63)</td>
<td>Shows how to use AWS CodePipeline to create a build with multiple input sources and multiple output artifacts.</td>
</tr>
<tr>
<td>AWS Config sample (p. 65)</td>
<td>Shows how to set up AWS Config. Lists which CodeBuild resources are tracked and describes how to look up CodeBuild projects in AWS Config.</td>
</tr>
<tr>
<td>AWS Elastic Beanstalk sample (p. 67)</td>
<td>Uses Apache Maven to produce a single WAR file. Uses Elastic Beanstalk to deploy the WAR file to an Elastic Beanstalk instance.</td>
</tr>
<tr>
<td>AWS Lambda sample (p. 74)</td>
<td>Uses CodeBuild, Lambda, AWS CloudFormation, and CodePipeline to build and deploy a serverless application that follows the AWS Serverless Application Model (AWS SAM) standard.</td>
</tr>
<tr>
<td>Bitbucket pull request and webhook filter sample (p. 75)</td>
<td>Uses CodeBuild with Bitbucket as the source repository and webhooks enabled, to rebuild the source code every time a code change is pushed to the repository.</td>
</tr>
<tr>
<td>Build badges sample (p. 82)</td>
<td>Shows how to set up CodeBuild with build badges.</td>
</tr>
<tr>
<td>Build notifications sample (p. 85)</td>
<td>Uses Apache Maven to produce a single JAR file. Sends a build notification to subscribers of an Amazon SNS topic.</td>
</tr>
<tr>
<td>Create a test report using the AWS CLI sample (p. 102)</td>
<td>Uses the AWS CLI to create, run, and view the results of a test report.</td>
</tr>
<tr>
<td>Docker in custom image sample (p. 107)</td>
<td>Uses a custom Docker image to produce a Docker image.</td>
</tr>
</tbody>
</table>
Use access tokens with your source provider in CodeBuild

This sample shows you how to connect to GitHub or Bitbucket with an access token. For GitHub or GitHub Enterprise, you use a personal access token. For Bitbucket, you use an app password.

Access token prerequisites

Before you begin, you must add the proper permission scopes to your access token.

For GitHub, your personal access token must have the following scopes.

- **repo**: Grants full control of private repositories.
- **repo:status**: Grants access to commit statuses.
- **admin:repo_hook**: Grants full control of repository hooks. This scope is not required if your token has the **repo** scope.
For more information, see Understanding Scopes for OAuth Apps on the GitHub website.

For Bitbucket, your app password 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 must have this scope.
- webhook: Grants access to webhooks. If your project has a webhook operation, then your app password must have this scope.

For more information, see Scopes for Bitbucket Cloud REST API and OAuth on Bitbucket Cloud on the Bitbucket website.

**Connect source providers with access tokens (console)**

To use the console to connect your project to GitHub or Bitbucket using access tokens, do the following while you create a project. For information, see Create a build project (console) (p. 216).

For GitHub:

1. For Source provider, choose GitHub.
2. For Repository, choose Connect with a GitHub personal access token.
3. In GitHub personal access token, enter your GitHub personal access token.
4. Choose Save token.

For Bitbucket:

1. For Source provider, choose Bitbucket.
**Note**
CodeBuild does not support Bitbucket Server.

2. For **Repository**, choose **Connect with a Bitbucket app password**.

3. In **Bitbucket username**, enter your Bitbucket user name.
4. In **Bitbucket app password**, enter your Bitbucket app password.
5. Choose **Save Bitbucket credentials**.

**Connect source providers with access tokens (CLI)**

Follow these steps to use the AWS CLI to connect your project to GitHub or Bitbucket using access tokens. For information about using the AWS CLI with AWS CodeBuild, see the Command line reference (p. 355).

1. Run the **import-source-credentials** command:

   ```bash
   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.
Replace the following:

- **server-type**: Required value. The source provider used for this credential. Valid values are GITHUB, GITHUB_ENTERPRISE, and BITBUCKET.
- **auth-type**: Required value. The type of authentication used to connect to a GitHub, GitHub Enterprise, or Bitbucket repository. Valid values include PERSONAL_ACCESS_TOKEN and BASIC_AUTH. You cannot use the CodeBuild API to create an OAUTH connection. You must use the CodeBuild console instead.
- **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, this is the personal access token. For Bitbucket, this is the app password.
- **username**: Optional value. The Bitbucket user name when authType is BASIC_AUTH. This parameter is ignored for other types of source providers or connections.

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.

```bash
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).

```json
{
  "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) (p. 229).

3. To view the connected access tokens, run the `list-source-credentials` command.

```bash
aws codebuild list-source-credentials
```

A JSON-formatted `sourceCredentialsInfos` object appears in the output:

```json
{
  "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`, or `PERSONAL_ACCESS_TOKEN`.
- The `serverType` is the type of source provider. This can be `GITHUB`, `GITHUB_ENTERPRISE`, or `BITBUCKET`.
- 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.

```bash
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON-formatted data is returned with an ARN of the deleted credentials.

```json
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

**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.

**Running the sample**

**To run this sample**

1. To create and push the Docker image to your image repository in Amazon ECR, complete the steps in the "Running the sample" section of the Docker sample (p. 109).
2. Create a Go project:
   a. Create the files as described in the Go project structure (p. 55) and Go project files (p. 56) 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 by following the steps in Run AWS CodeBuild directly (p. 177).
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.)

```json
{
    "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:2.0",
        "computeType": "BUILD_GENERAL1_SMALL"
    },
    "serviceRole": "arn:aws:iam::account-ID:role/service-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. 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 a cross-account Amazon ECR image.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccess",
      "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 by following the steps in Run AWS CodeBuild directly (p. 177).

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:latest",
    "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.
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)
}
```

Related resources

- For information about getting started with AWS CodeBuild, see Getting started with AWS CodeBuild using the console (p. 5).
- For information about troubleshooting issues in CodeBuild, see Troubleshooting AWS CodeBuild (p. 359).
- For information about quotas in CodeBuild, see Quotas for AWS CodeBuild (p. 374).

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

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, 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.

High-level steps

This sample covers the three high-level steps required to use Amazon EFS with AWS CodeBuild:

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.

**Note**

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

Create a VPC using AWS CloudFormation

Create your VPC with an AWS CloudFormation template.

1. Follow the instructions in AWS CloudFormation VPC template (p. 182) 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.

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.

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.

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.
10. From Environment type, choose Linux.
11. Select Privileged.

   Note
   By default, Docker containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.
12. Under Service role, choose New service role. In Role name, enter a name for the role CodeBuild creates for you.
13. Expand Additional configuration.
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 efs-1, the environment variable is CODEBUILD_EFS-1.
   - 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 name of a directory in your build container that mounts the file system. 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 buildspec commands into the editor. Replace `file-system-identifier` with the identifier you entered in step 17. Use capital letters (for example, CODEBUILD_EFS-1).

```yaml
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**.

**CodeBuild and Amazon EFS sample summary**

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.

**CodeDeploy sample for CodeBuild**

This sample instructs AWS CodeBuild to use Maven to produce as build output a single JAR file named `my-app-1.0-SNAPSHOT.jar`. This sample then uses CodeDeploy to deploy the JAR file to an Amazon Linux instance. You can also use AWS CodePipeline to automate the use of CodeDeploy to deploy the JAR file to an Amazon Linux instance. This sample is based on the Maven in 5 Minutes topic on the Apache Maven website.

**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, CloudWatch Logs, and Amazon EC2. For more information, see CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, Amazon CloudWatch Pricing, and Amazon EC2 Pricing.
Running the sample

To run this sample

1. Download and install Maven. For more 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.

   ```bash
   mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
   ```

   If successful, this directory structure and files is created.

   ```
   (root directory name)
   `-- my-app
       |-- pom.xml
       `-- src
           |-- main
           |   |-- java
           |       |-- com
           |           |-- mycompany
           |               |-- app
           |                   |-- App.java
           `-- test
               |-- java
               |   |-- com
               |       |-- mycompany
               |           |-- app
               |               |-- AppTest.java
   ```

3. Create a file with this content. Name the file `buildspec.yml`, and then add it to the `(root directory name)`/my-app directory.

   ```yaml
   version: 0.2
   phases:
   install:
       runtime-versions:
       java: corretto8
   build:
       commands:
       - echo Build started on `date`
       - mvn test
   post_build:
       commands:
       - echo Build completed on `date`
       - mvn package
   artifacts:
       files:
       - target/my-app-1.0-SNAPSHOT.jar
       - appspec.yml
       discard-paths: yes
   ```

4. Create a file with this content. Name the file `appspec.yml`, and then add it to the `(root directory name)`/my-app directory.

   ```yaml
   version: 0.0
   os: linux
   files:
   ```
- source: ./my-app-1.0-SNAPSHOT.jar
destination: /tmp

When finished, your directory structure and file should look like this.

```
(root directory name)
|-- my-app
  |-- buildspec.yml
  |-- appspec.yml
  |-- pom.xml
  `-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                 |-- App.java
    |-- test
    `-- java
        `-- com
            `-- mycompany
                `-- app
                    `-- AppTest.java
```

5. Create a ZIP file that contains the directory structure and files inside of `(root directory name)/my-app`, and then upload the ZIP file to a source code repository type supported by AWS CodeBuild and CodeDeploy, such as an S3 input bucket or a GitHub or Bitbucket repository.

**Important**

If you want to use CodePipeline to deploy the resulting build output artifact, you cannot upload the source code to a Bitbucket repository.

Do not add `(root directory name)` or `(root directory name)/my-app` to the ZIP file, just the directories and files inside of `(root directory name)/my-app`. The ZIP file should contain these directories and files:

```
CodeDeploySample.zip
|--buildspec.yml
|-- appspec.yml
|-- pom.xml
 `-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                 |-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

6. Create a build project by following the steps in Create a build project (p. 215).

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-codedeploy-project",
    "source": {
        "type": "S3",
```
7. If you plan to deploy the build output artifact with CodeDeploy, follow the steps in Run a build (p. 269). Otherwise, skip this step. (This is because if you plan to deploy the build output artifact with CodePipeline, CodePipeline uses CodeBuild to run the build automatically.)

8. Complete the setup steps for using CodeDeploy, including:
   - Grant the IAM user access to CodeDeploy and the AWS services and actions CodeDeploy depends on. For more information, see Provision an IAM User in the AWS CodeDeploy User Guide.
   - Create or identify a service role to enable CodeDeploy to identify the instances where it deploys the build output artifact. For more information, see Creating a Service Role for CodeDeploy in the AWS CodeDeploy User Guide.
   - Create or identify an IAM instance profile to enable your instances to access the S3 input bucket or GitHub repository that contains the build output artifact. For more information, see Creating an IAM Instance Profile for Your Amazon EC2 Instances in the AWS CodeDeploy User Guide.

9. Create or identify an Amazon Linux instance compatible with CodeDeploy where the build output artifact is deployed. For more information, see Working with Instances for CodeDeploy in the AWS CodeDeploy User Guide.

10. Create or identify a CodeDeploy application and deployment group. For more information, see Creating an Application with CodeDeploy in the AWS CodeDeploy User Guide.

11. Deploy the build output artifact to the instance.

    To deploy with CodeDeploy, see Deploying a Revision with CodeDeploy in the AWS CodeDeploy User Guide.

    To deploy with CodePipeline, see Use AWS CodePipeline with AWS CodeBuild (p. 195).

12. To find the build output artifact after the deployment is complete, sign in to the instance and look in the /tmp directory for the file named my-app-1.0-SNAPSHOT.jar.

**Related resources**

- For information about getting started with AWS CodeBuild, see Getting started with AWS CodeBuild using the console (p. 5).
- For information about troubleshooting issues in CodeBuild, see Troubleshooting AWS CodeBuild (p. 359).
- For information about quotas in CodeBuild, see Quotas for AWS CodeBuild (p. 374).
AWS CodePipeline integration with CodeBuild and multiple input sources and output artifacts sample

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 (p. 143).

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 AWS CodePipeline Pipeline Structure Reference in the AWS CodePipeline User Guide.

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

API Version 2016-10-06

63
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 (p. 172).

- 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 (p. 150).

```
version: 0.2
```

API Version 2016-10-06
phases:
  build:
    commands:
      - touch source1_file
      - cd $CODEBUILD_SRC_DIR_source2
      - touch source2_file

artifacts:
  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.

Use AWS Config with CodeBuild sample

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.

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

Topics
- Prerequisites (p. 65)
- Set up AWS Config (p. 65)
- Look up AWS CodeBuild projects (p. 66)
- Viewing AWS CodeBuild configuration details in the AWS Config console (p. 66)

Prerequisites

Create your AWS CodeBuild project. For instructions, see Create a build project (p. 215).

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.

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, choose Resources. 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.

Viewing AWS CodeBuild configuration details 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.

Example of a CodeBuild project in AWS Config:
AWS Elastic Beanstalk sample for CodeBuild

This sample instructs AWS CodeBuild to use Maven to produce as build output a single WAR file named `my-web-app.war`. This sample then deploys the WAR file to the instances in an AWS Elastic Beanstalk environment.

**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, CloudWatch Logs, and Amazon EC2. For more information, see CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, Amazon CloudWatch Pricing, and Amazon EC2 Pricing.

Create the source code

In this section, you use Maven to produce the source code. Later, you use CodeBuild to build a WAR file based on this source code.

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=my-web-app -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
   ```

   If successful, this directory structure and files are created.

   ```
   (root directory name)
   `-- my-web-app
       `-- pom.xml
       `-- src
         `-- main
           `-- resources
             `-- WEB-INF
               `-- web.xml
               `-- index.jsp
   ```
3. Create a subdirectory named `.ebextensions` in the `(<root directory name>/my-web-app` directory. In the `.ebextensions` subdirectory, create a file named `fix-path.config` with this content.

```plaintext
container_commands:
    fix_path:
        command: "unzip my-web-app.war 2>&1 > /var/log/my_last_deploy.log"
```

After you run Maven, continue with one of the following scenarios:

- **Scenario A: Run CodeBuild manually and deploy to Elastic Beanstalk manually** (p. 68)
- **Scenario B: Use CodePipeline to run CodeBuild and deploy to Elastic Beanstalk** (p. 70)
- **Scenario C: Use the Elastic Beanstalk CLI to run AWS CodeBuild and deploy to an Elastic Beanstalk environment** (p. 72)

**Scenario A: Run CodeBuild manually and deploy to Elastic Beanstalk manually**

In this scenario, you create and upload the source code. You then use the AWS CodeBuild and AWS Elastic Beanstalk consoles to build the source code, create an Elastic Beanstalk application and environment, and deploy the build output to the environment.

**Step a1: Add files to the source code**

In this step, you add an Elastic Beanstalk configuration file and a buildspec file to the code in Create the source code (p. 67). You then upload the source code to an S3 input bucket or a CodeCommit, GitHub, or Bitbucket repository.

1. Create a file named `buildspec.yml` with the following contents. Store the file in the `(<root directory name>/my-web-app` directory.

```plaintext
version: 0.2
phases:
    install:
        runtime-versions:
            java: corretto11
    post_build:
        commands:
            - mvn package
            - mvn target/my-web-app.war my-web-app.war
        artifacts:
            files:
                - my-web-app.war
                - .ebextensions/**/*
```

2. Your file structure should now look like this.

```
(<root directory name>)
    `|-- my-web-app
    |    `|-- .ebextensions
    |         `|-- fix-path.config
    |`-- src
        `|-- main
            |`-- resources
            |`-- webapp
```

API Version 2016-10-06
68
3. 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, it must be versioned. 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. For more information, see How to Configure Versioning on a Bucket in the Amazon S3 Developer Guide.

**Step a2: Create the build project and run the build**

In this step, you use the AWS CodeBuild console to create a build project and then run a build.

1. Create or choose an S3 output bucket to store the build output. If you’re storing the source code in an S3 input bucket, the output bucket must be in the same AWS region as the input bucket.
   Use the AWS region selector to choose an AWS Region where CodeBuild is supported. This must be the same Region where your S3 output bucket is stored.
3. Create a build project and then run a build. For more information, see Create a build project (console) (p. 216) and Run a build (console) (p. 270). 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/amazonlinux2-x86_64-standard:2.0.
   - For **Artifacts**:  
     - For **Type**, choose Amazon S3.
     - For **Bucket name**, enter the name of an S3 bucket.
     - For **Name**, enter a build output file name that’s easy for you to remember. Include the .zip extension.
   - For **Artifacts packaging**, choose Zip.

**Step a3: Create the application and environment and deploy**

In this step, you use the AWS Elastic Beanstalk console to create an application and environment. As part of creating the environment, you deploy the build output from the previous step to the environment.

   Use the AWS Region selector to choose the AWS Region where your S3 output bucket is stored.
2. Create an Elastic Beanstalk application. For more information, see Managing and Configuring AWS Elastic Beanstalk Applications in the AWS Elastic Beanstalk Developer Guide.
3. Create an Elastic Beanstalk environment for this application. For more information, see The Create New Environment Wizard in the AWS Elastic Beanstalk Developer Guide. Leave all settings at their default values, except for these settings.

- For Platform, choose Tomcat.
- For Application code, choose Upload your code, and then choose Upload. For Source code origin, choose Public S3 URL, and then enter the full URL to the build output ZIP file in the output bucket. Choose Upload.

4. After Elastic Beanstalk deploys the build output to the environment, you can see the results in a web browser. Go to the environment URL for the instance (for example, http://my-environment-name.random-string.region-ID.elasticbeanstalk.com). The web browser should display the text Hello World!

Scenario B: Use CodePipeline to run CodeBuild and deploy to Elastic Beanstalk

In this scenario, you complete the steps to prepare and upload the source code. You create a build project with CodeBuild and an Elastic Beanstalk application and environment with the AWS Elastic Beanstalk console. You then use the AWS CodePipeline console to create a pipeline. After you create the pipeline, CodePipeline builds the source code and deploys the build output to the environment.

Step b1: Add a buildspec file to the source code

In this step, you create and add a buildspec file to the code you created in Create the source code (p. 67). You then upload the source code to an S3 input bucket or a CodeCommit, GitHub, or Bitbucket repository.

1. 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: corretto11
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app.war my-web-app.war
    artifacts:
      files:
        - my-web-app.war
        - .ebextensions/**/*
        base-directory: 'target/my-web-app'
```

2. Your file structure should now look like this.

```
(root directory name)
  |-- my-web-app
      |-- .ebextensions
      |     `-- fix-path.config
      |-- src
      |    `-- main
      |         `-- resources
      |         `-- webapp
      |             `-- WEB-INF
      |                 `-- web.xml
```
3. 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, it must be versioned. 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. For more information, see How to Configure Versioning on a Bucket in the Amazon S3 Developer Guide.

Step b2: Create a build project

In this step, you create an AWS CodeBuild build project to use with your pipeline.

2. Create a build project. For more information, see Create a build project (console) (p. 216) and Run a build (console) (p. 270). 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/amazonlinux2-x86_64-standard:2.0.
   
   • For Artifacts:
     • For Type, choose Amazon S3.
     • For Bucket name, enter the name of an S3 bucket.
     • For Name, enter a build output file name that's easy for you to remember. Include the .zip extension.
     • For Artifacts packaging, choose Zip.

Step b3: Create an Elastic Beanstalk application and environment

In this step, you create an Elastic Beanstalk application and environment to use with CodePipeline.

2. Use the AWS Elastic Beanstalk console to create an application. For more information, see Managing and Configuring AWS Elastic Beanstalk Applications in the AWS Elastic Beanstalk Developer Guide.
3. Use the AWS Elastic Beanstalk console to create an environment. For more information, see The Create New Environment Wizard in the AWS Elastic Beanstalk Developer Guide. Except for Platform, leave all settings at their default values. For Platform, choose Tomcat.

Step b4: Create the pipeline and deploy

In this step, you use the AWS CodePipeline console to create a pipeline. After you create and run the pipeline, CodePipeline uses CodeBuild to build the source code. CodePipeline then uses Elastic Beanstalk to deploy the build output to the environment.
1. Create or identify a service role that CodePipeline, CodeBuild, and Elastic Beanstalk can use to access resources on your behalf. For more information, see Prerequisites (p. 196).


   Use the AWS Region selector to choose an AWS Region where CodeBuild is supported. If you’re storing the source code in an S3 input bucket, the output bucket must be in the same AWS region as the input bucket.

3. Create a pipeline. For information, see Create a pipeline that uses CodeBuild (CodePipeline console) (p. 197). Leave all settings at their default values, except for these settings.

   - On Add build stage, for Build provider, choose AWS CodeBuild. For Project name, choose the build project you just created.

   - On Add deploy stage, for Deploy provider, choose AWS Elastic Beanstalk.
     - For Application name, choose the Elastic Beanstalk application you just created.
     - For Environment name, choose the environment you just created.

4. After the pipeline has run successfully, you can see the results in a web browser. Go to the environment URL for the instance (for example, http://my-environment-name.random-string.region-ID.elasticbeanstalk.com). The web browser should display the text Hello World!

Now, whenever you make changes to the source code and upload those changes to the original S3 input bucket or to the CodeCommit, GitHub, or Bitbucket repository, CodePipeline detects the change and runs the pipeline again. This causes CodeBuild to rebuild the code and then causes Elastic Beanstalk to deploy the rebuilt output to the environment.

**Scenario C: Use the Elastic Beanstalk CLI to run AWS CodeBuild and deploy to an Elastic Beanstalk environment**

In this scenario, you complete the steps to prepare and upload the source code. You then run the Elastic Beanstalk CLI to create an Elastic Beanstalk application and environment, use CodeBuild to build the source code, and then deploy the build output to the environment. For more information, see Using the EB CLI with CodeBuild in the AWS Elastic Beanstalk Developer Guide.

**Step c1: Add files to the source code**

In this step, you add an Elastic Beanstalk configuration file and a buildspec file to the code you created in Create the source code (p. 67). You also create or identify a service role for the buildspec file.

1. Create or identify a service role that Elastic Beanstalk and the CLI can use on your behalf. For information, see Create a CodeBuild service role (p. 348).

2. 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: corretto11
     post_build:
       commands:
         - mvn package
         - mvn target/my-web-app.war my-web-app.war
   artifacts:
     files:
   ```

   API Version 2016-10-06
   72
In the preceding code, replace `my-service-role-name` with the name of the service role you created or identified earlier.

3. Your file structure should now look like this.

```
(root directory name)
|-- my-web-app
 |  |-- .ebextensions
 |     `-- fix-path.config
 |-- src
 |  |-- main
 |      |-- resources
 |      `-- webapp
 |          |-- WEB-INF
 |              |-- web.xml
 |              `-- index.jsp
|-- buildspec.yml
 `-- pom.xml
```

### Step c2: Install and run the EB CLI

1. If you have not already done so, install and configure the EB CLI on the same computer or instance where you created the source code. For information, see Install the Elastic Beanstalk Command Line Interface (EB CLI) and Configure the EB CLI in the AWS Elastic Beanstalk Developer Guide.

2. From the command line or terminal, run the `cd` command or similar to switch to your (root directory name)/my-web-app directory. Run the `eb init` command to configure the EB CLI.

```
  eb init
```

When prompted:

- Choose an AWS Region where AWS CodeBuild is supported and where you want to create your Elastic Beanstalk application and environment.
- Create an Elastic Beanstalk application, and enter a name for the application.
- Choose the Tomcat platform.
- Choose the Tomcat 8 Java 8 version.
- Choose whether you want to use SSH to set up access to your environment’s instances.

3. From the same directory, run the `eb create` command to create an Elastic Beanstalk environment.

```
  eb create
```

When prompted:

- Enter the name for the environment or accept the suggested name.
- Enter the DNS CNAME prefix for the environment or accept the suggested value.
- For this sample, accept the Classic load balancer type.

4. After you run the `eb create` command, the EB CLI does the following:

1. Creates a ZIP file from the source code and then uploads the ZIP file to an S3 bucket in your account.
2. Creates an Elastic Beanstalk application and application version.
3. Creates a CodeBuild project.
4. Runs a build based on the new project.
5. Deletes the project after the build is complete.
7. Deploys the build output to the environment.

5. After the EB CLI deploys the build output to the environment, you can see the results in a web browser. Go to the environment URL for the instance (for example, http://my-environment-name.random-string.region-ID.elasticbeanstalk.com). The web browser should display the text Hello World!. If you want, you can make changes to the source code and then run the `eb deploy` command from the same directory. The EB CLI performs the same steps as the `eb create` command, but it deploys the build output to the existing environment instead of creating a new environment.

Related resources

- For information about getting started with AWS CodeBuild, see Getting started with AWS CodeBuild using the console (p. 5).
- For information about troubleshooting issues in CodeBuild, see Troubleshooting AWS CodeBuild (p. 359).
- For information about quotas in CodeBuild, see Quotas for AWS CodeBuild (p. 374).

AWS Lambda sample for CodeBuild

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 Lambda-based Applications in the AWS Lambda Developer Guide. To experiment with a serverless application sample that uses CodeBuild along with AWS Lambda, AWS CloudFormation, and CodePipeline, see Automating Deployment of Lambda-based Applications in the AWS Lambda Developer Guide.

Related resources

- For information about getting started with AWS CodeBuild, see Getting started with AWS CodeBuild using the console (p. 5).
- For information about troubleshooting issues in CodeBuild, see Troubleshooting AWS CodeBuild (p. 359).
- For information about quotas in CodeBuild, see Quotas for AWS CodeBuild (p. 374).
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.

Topics

- Prerequisites (p. 75)
- Create a build project with Bitbucket as the source repository and enable webhooks (p. 75)
- Trigger a build with a Bitbucket webhook (p. 77)
- Filter Bitbucket webhook events (p. 77)

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.

Create a build project with Bitbucket as the source repository and enable webhooks

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

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. On the Create build project page, in Project configuration, 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 Bitbucket.
Follow the instructions to connect or reconnect, and then choose **Grant access**.

**Note**
CodeBuild does not support Bitbucket Server.

5. **Choose Use a repository in my account.** You cannot use a webhook if you use a public Bitbucket repository.
6. In Primary source webhook events, select Rebuild every time a code change is pushed to this repository. You can select this check box only if you chose Repository in my Bitbucket account.  

Note: If a build is triggered by a Bitbucket webhook, the Report build status setting is ignored. The build status is always sent to Bitbucket.

7. Choose other settings for your project. For more information about source provider options and settings, see Choose source provider.

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

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.


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 “Create a build project with Bitbucket as the source repository and enable webhooks” (p. 75).

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.

Filter 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 triggered for specified branches only.
You can specify more than one webhook filter group. A build is triggered if the filters on one or more filter groups 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, and PULL_REQUEST_MERGED. 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.

<table>
<thead>
<tr>
<th>X-Event-Key Header value</th>
<th>Event type</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo:push</td>
<td>PUSH</td>
</tr>
<tr>
<td>pullrequest:created</td>
<td>PULL_REQUEST_CREATED</td>
</tr>
<tr>
<td>pullrequest:updated</td>
<td>PULL_REQUEST_UPDATED</td>
</tr>
<tr>
<td>pullrequest:fulfilled</td>
<td>PULL_REQUEST_MERGED</td>
</tr>
</tbody>
</table>

  **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.

- One or more optional filters. Use a regular expression to specify a filter. For an event to trigger a build, every filter 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.

  **Note**
  You can find the webhook payload in the webhook settings of your Bitbucket repository.

**Topics**
- Filter Bitbucket webhook events (console) (p. 78)
- Filter Bitbucket webhook events (SDK) (p. 80)
- Filter Bitbucket webhook events (AWS CloudFormation) (p. 82)

**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) (p. 216)](#) and [WebhookFilter in the AWS CodeBuild API Reference](#).

In this example, a webhook filter group triggers a build for pull requests only:

- 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/master$` and head references that match `^refs/heads/branch1!`.
- The second filter group specifies push requests on branches with Git reference names that match the regular expression `^refs/heads/branch1$`.

Using an example of two filter groups, a build is triggered when one or both evaluate to true:
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 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.

**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"
    }
  ]
]
```

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 are 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/master$` and head references that match `^refs/heads/myBranch$`.

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$"  
    },  
    { "type": "BASE_REF",  
      "pattern": "^refs/heads/master$"  
    }  
  ], 
  [ 
    { "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 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/username, where `username` is your Bitbucket user name.

```
"filterGroups": [ 
  [ 
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED, PULL_REQUEST_MERGED"  
    },  
    { "type": "HEAD_REF",  
      "pattern": "^refs/heads/myBranch$"  
    }  
  ] 
]
```
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/master$ 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/.*.

```
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:2.0
    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/master$
          ExcludeMatchedPattern: false
        - Type: ACTOR_ACCOUNT_ID
          Pattern: 12345
          ExcludeMatchedPattern: true
        - Type: EVENT
          Pattern: PUSH
        - Type: HEAD_REF
          Pattern: ^refs/heads/.*
```

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.

Create a build project with build badges enabled (console)

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. On the Create build project page, in Project configuration, 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 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. For **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** (p. 149).

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** (p. 150).

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.

## Create a build project with build badges enabled (CLI)

For information about creating a build project, see **Create a build project (AWS CLI)** (p. 229). To include build badges with your AWS CodeBuild project, you must specify **badgeEnabled** with a value of `true`.

## Access your 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)** (p. 244).

• 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)** (p. 244).

**Important**
The build badge request URL is for the master branch, but you can specify any branch in your source repository that you have used to run a build.

---

**API Version 2016-10-06**

84
Publish your CodeBuild build badges

You can include your build badge request URL in a markdown file in your preferred repository (for example, GitHub or CodeCommit) to display the status of the latest build.

Sample markdown code:

```
![Build Status](https://codebuild.us-east-1.amazon.com/badges?uuid=...&branch=master)
```

CodeBuild badge 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.

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.

Running the 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 IAM 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.

```json
{
    "Statement": [
        `### BEGIN ADDING STATEMENT HERE ###`
        {
            "Action": [
                "sns:CreateTopic",
                "sns:GetTopicAttributes",
                "sns:List*"]
        `### END ADDING STATEMENT HERE ###`
    ]
}
```
### END ADDING STATEMENT HERE ###

...
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.
Create subscription

Topic ARN

Protocol

Endpoint
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 IAM user instead of an AWS root account or an administrator IAM 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.

```json
{
    "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*.


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**:

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

Compare your results:
Step 1: Create Event Source

Create rules to invoke Target Schedule to invoke Target

Build or customize an Event Pattern
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.

```json
{
    "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.

```json
{
    "source": [
        "aws.codebuild"
    ],
    "detail-type": [
        "CodeBuild Build Phase Change"
    ],
    "detail": {
        "completed-phase": [
            "SUBMITTED",
            "PROVISIONING",
            "DOWNLOAD_SOURCE",
            "INSTALL",
            "PRE_BUILD",
            "BUILD",
            "POST_BUILD",
            "UPLOAD_ARTIFACTS"
        ]
    }
}
```
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.

**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.

   ```json
   {"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.
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:

   ```json
   {"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](#) (p. 98).

   Compare your results so far to the following, which shows a rule with a detail-type value of **CodeBuild Build State Change**:

   "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>'."
Step 1: Creating

Create rules to invoke

Event Source

Build or customize an Event Source

Schedule to invoke Target

Event Pattern
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.

Compare your results so far to the following screen shot:
Step 2: Configure

Rule definition

Name* CodeBuild

Description

State [ ] Enabled
19. Choose **Create rule**.
20. Create build projects, run the builds, and view build information by following the steps in Run AWS CodeBuild directly (p. 177).
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**.

**Related resources**

- For information about getting started with AWS CodeBuild, see Getting started with AWS CodeBuild using the console (p. 5).
- For information about troubleshooting issues in CodeBuild, see Troubleshooting AWS CodeBuild (p. 359).
- For information about quotas in CodeBuild, see Quotas for AWS CodeBuild (p. 374).

**Build notifications input format reference**

CloudWatch delivers notifications in JSON format.

Build state change notifications use the following format:

```json
{
  "version": "0",
  "id": "c030038d-8c4d-6141-9545-0ff7b7153EX",
  "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": {
      "md5sum": "da9c44c8a9a3c3db4b43116e833168fEX",
      "sha256sum": "6ccc2a1df9d155ba83c5797051611c42d6e09c6329dcb14a312cecc0a8e39EX",
      "location": "arn:aws:s3:::codebuild-:123456789012-output-bucket/my-output-artifact.zip"
    },
    "environment": {
      "image": "aws/codebuild/standard:2.0",
      "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",
},
"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 4:14:21 PM",
  "duration-in-seconds": 70,
  "phase-type": "BUILD",
  "phase-status": "SUCCEEDED"
  },
Build phase change notifications use the following format:

```json
{
  "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": "da9c44c8a9a3cd4b43126e823168fEX",
        "sha256sum": "6cccc2ae1df9d155ba8c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
        "location": "arn:aws:s3:::codebuild-:123456789012-output-bucket/my-output-artifact.zip"
      }
    }
  }
}
```

API Version 2016-10-06

100
"environment": {
  "image": "aws/codebuild/standard:2.0",
  "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": "0745a7a9-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 4:13:10 PM",
    "duration-in-seconds": 0,
    "phase-type": "BUILD",
    "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": "POST_BUILD",
    "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": "COMPLETE",
    "phase-status": "SUCCEEDED"
  }
]
Create a test report in CodeBuild using the AWS CLI sample

The test reporting feature is in preview release for CodeBuild and is subject to change.

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
- Prerequisites (p. 103)
• Create a report group (p. 103)
• Configure a project with a report group (p. 104)
• Run and view results of a report (p. 105)

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 format of your test cases can be JUnit XML, Cucumber JSON, Visual Studio TRX, or TestNG XML. Create your test cases with any test framework that can create test files in one of those formats (for example, Surefire JUnit plugin, TestNG, and 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 (p. 300).

Create a report group

2. Create a folder in your S3 bucket where your test results are exported.
3. Copy the following into CreateReportGroupInput.json. For bucket, use the name of the S3 bucket. For path, enter the path to the folder in your S3 bucket.

```
{
    "name": "report-name",
    "type": "TEST",
    "exportConfig": {
        "type": "S3",
        "s3": {
            "bucket": "bucket-name",
            "path": "path-to-folder",
            "packaging": "NONE"
        }
    }
}
```

4. Run the following command in the directory that contains CreateReportGroupInput.json. For region, specify your AWS Region (for example, us-east-2).

```
aws codebuild create-report-group \
    --cli-input-json file://CreateReportGroupInput.json \
    --region us-east-2
```
--region your-region

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.

```json
{
  "reportGroup": {
    "name": "report-name",
    "type": "TEST",
    "exportConfig": {
      "type": "S3",
      "s3": {
        "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
  }
}
```

**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.

   ```yaml
   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 a test report (p. 286) and Report group naming (p. 293).

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.

```
{
  "create-project-with-report":
  {
    "name": "test-report-project",
    "description": "sample-test-report-project",
    "source": {
      "type": "your-repository-type",
      "location": "https://github.com/your-repository/your-folder"
    },
    "artifacts": {
      "type": "NO_ARTIFACTS"
    },
    "cache": {
      "type": "NO_CACHE"
    },
    "environment": {
      "type": "LINUX_CONTAINER",
      "image": "aws/codebuild/standard:2.0",
      "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
  --region your-region
```

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. Make a note of the build ID that appears in the output. Its format is test-report>:build-id.

```
aws codebuild start-build
  --project-name "test-project"
  --region your-region
```

2. Run the following command to get information about your build, including the ARN of your report. For --ids, specify your build ID. Make a note of the report ARN in the output.

```
aws codebuild batch-get-builds
  --ids "build-id"
  --region your-region
```

3. Run the following command to get details about your reports. For --report-group-arn, specify your report ARN.

```
aws codebuild batch-get-reports
  --report-arns report-group-arn
```

API Version 2016-10-06
Create a test report using the AWS CLI sample

--region your-region

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": {
        "type": "S3",
        "s3": {
          "bucket": "your-s3-bucket",
          "path": "path-to-your-report-results",
          "packaging": "NONE",
          "encryptionKey": "encryption-key"
        }
      },
      "expired": 1575916770.0,
      "truncated": false,
      "executionId": "arn:aws:codebuild:us-west-2:123456789012:build/name-of-build-project:2c254862-ddf6-4831-a53f-6839a73829c1",
      "type": "TEST",
      "arn": "report-arn",
      "testSummary": {
        "durationInNanoSeconds": 6657770,
        "total": 11,
        "statusCounts": {
          "FAILED": 3,
          "SKIPPED": 7,
          "ERROR": 0,
          "SUCCEEDED": 1,
          "UNKNOWN": 0
        }
      }
    },
    {
      "reportsNotFound": []
    }
  ]
}
```

4. 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 \
--region your-region
```

The output looks like the following.

```
{
  "testCases": [
    {
      "status": "FAILED",
      "name": "Test case 1",
      "expired": 1575916770.0,
      "reportArn": "report-arn",
      "prefix": "Cucumber tests for agent",
    }
  ]
}
```
Docker in custom image sample for CodeBuild

This 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 Docker sample (p. 109).

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
- Running the sample (p. 107)
- Directory structure (p. 108)
- Files (p. 108)
- Related resources (p. 56)

Running the sample

To run this 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 by following the steps in Run AWS CodeBuild directly (p. 177).

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

**Note**
By default, Docker containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.

3. To see the build results, look in the build's log for the string Hello, World!. For more information, see View build details (p. 278).

**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. For more information about the OverlayFS storage driver referenced in the buildspec file, see Use the OverlayFS storage driver on the Docker website.

```
bustspec.yml (in (root directory name))

version: 0.2
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"
  pre_build:
    commands:
      - docker build -t helloworld .
  build:
    commands:
      - docker images
      - docker run helloworld echo "Hello, World!"
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"
```

Dockerfile (in **root directory name**)  

```plaintext
FROM maven:3.3.9-jdk-8
RUN echo "Hello World"
```

Related resources

- For information about getting started with AWS CodeBuild, see Getting started with AWS CodeBuild using the console (p. 5).
- For information about troubleshooting issues in CodeBuild, see Troubleshooting AWS CodeBuild (p. 359).
- For information about quotas in CodeBuild, see Quotas for AWS CodeBuild (p. 374).

Docker 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 Adapting the sample to push the image to Docker Hub (p. 113).

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 (p. 107).

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

- Running the sample (p. 110)
- Directory structure (p. 112)
- Files (p. 112)
- Adapting the sample to push the image to Docker Hub (p. 113)
- Related resources (p. 56)
Running the sample

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 IAM user instead of an AWS root account or an administrator IAM 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 IAM User Guide.

```json
{
    "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.

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.

```json
{
    "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 ###
        ...
    ]
}
```
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.

**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. Follow the steps in Run AWS CodeBuild directly (p. 177) to 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:2.0.

d. Because you use this build project to build a Docker image, select Privileged.

**Note**
By default, Docker containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.

e. 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.)

```json
{
  "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:2.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [
      {
        "name": "AWS_DEFAULT_REGION",
        "value": "region-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)`)  

**Note**

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

```
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
  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
```

API Version 2016-10-06
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"]
```

### Adapting the sample to push the image to Docker Hub

To push the Docker image to Docker Hub instead of Amazon ECR, edit this sample's code.

**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 --no-include-email --region $AWS_DEFAULT_REGION)
   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:
```

**API Version 2016-10-06**

113
- 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:

   ```json
   ...
   "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:

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

4. Follow the steps in Run AWS CodeBuild directly (p. 177) to 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.
Related resources

- For information about getting started with AWS CodeBuild, see Getting started with AWS CodeBuild using the console (p. 5).
- For information about troubleshooting issues in CodeBuild, see Troubleshooting AWS CodeBuild (p. 359).
- For information about quotas in CodeBuild, see Quotas for AWS CodeBuild (p. 374).

GitHub Enterprise sample for CodeBuild

AWS CodeBuild supports GitHub Enterprise as a source repository. This sample shows how to set up your CodeBuild projects when your GitHub Enterprise 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 repository.

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 in GitHub Enterprise on the GitHub Help website.

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

   Select scopes
   Scopes define the access for personal tokens. Read more about OAuth scopes.

   ![Select scopes](image)

   1. repo
      - Full control of private repositories
      - Access commit status
      - Access deployment status
      - Access public repositories

2. Download your certificate from GitHub Enterprise. 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 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:**

   ```
   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 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.
Use your browser to download your certificate from GitHub Enterprise. 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.

Create a build project with GitHub Enterprise as the source repository and enable webhooks (console)

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. On the Create build project page, in **Project configuration**, 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**.
   - For **Personal Access Token**, paste the token you copied to your clipboard and choose **Save Token**. In **Repository URL**, enter the URL for your GitHub Enterprise repository.
     **Note**
     You only need to enter and save the personal access token once. All future AWS CodeBuild projects use this token.
   - 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 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/dockerrimage 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 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 (p. 178).

8. For 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 (p. 149).

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** (p. 150).

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. The `ARM_CONTAINER` and `LINUX_GPU_CONTAINER` environment types and the `BUILD_GENERAL1_2XLARGE` compute type do not support the use of a local cache.

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** (p. 150). For more information about caching, see **Build caching in AWS CodeBuild** (p. 245).

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

12. If you enabled webhooks in **Source**, a **Create webhook** dialog box is displayed with values for **Payload URL** and **Secret**.

  **Important**
  The **Create webhook** dialog box appears only once. Copy the payload URL and secret key. You need them when you add a webhook in GitHub Enterprise.
  If you need to generate a payload URL and secret key again, you must first delete the webhook from your GitHub Enterprise repository. In your CodeBuild project, clear the **Webhook** check box and then choose **Save**. You can then create or update a CodeBuild project with the **Webhook** check box selected. The **Create webhook** dialog box appears again.

13. In GitHub Enterprise, choose the repository where your CodeBuild project is stored.

14. Choose **Settings**, choose **Hooks & services**, and then choose **Add webhook**.

15. Enter the payload URL and secret key, accept the defaults for the other fields, and then choose **Add webhook**.
16. Return to your CodeBuild project. Close the Create webhook dialog box and choose Start build.

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.

**Note**
We recommend that you use a filter group to specify which GitHub users can trigger a build in a public repository. This can prevent a user from triggering an unexpected build. For more information, see Filter GitHub webhook events (p. 123).

Create a build project with GitHub as the source repository and enable webhooks (console)

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**:

   On the **Create build project** page, in **Project configuration**, 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**.

   Choose **Repository in my GitHub account**.

   In **GitHub repository**, enter the URL for your GitHub repository.

   ![GitHub repository configuration](image)

   6. In **Primary source webhook events**, select **Rebuild every time a code change is pushed to this repository**. You can select this check box only if you chose **Repository in my GitHub account**.
7. 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 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**.

8. 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.

9. For **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 (p. 149)**.

10. 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 (p. 150).

11. Expand Additional configuration and set options as appropriate.
12. Choose Create build project. On the Review page, choose Start build to run the build.

Verification checks

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 Build details.
4. In Source, 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.

Filter 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 triggered for specified branches only.

Note
Filter groups work the same way in GitHub and GitHub Enterprise.

You can create one or more webhook filter groups to specify which webhook events trigger a build. A build is triggered if all the filters on one or more filter groups 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, and PULL_REQUEST_MERGED. 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.

<table>
<thead>
<tr>
<th>X-GitHub-Event Header value</th>
<th>Webhook event payload action value</th>
<th>Event type</th>
</tr>
</thead>
<tbody>
<tr>
<td>pull_request</td>
<td>opened</td>
<td>PULL_REQUEST_CREATED</td>
</tr>
<tr>
<td>pull_request</td>
<td>reopened</td>
<td>PULL_REQUEST_REOPENED</td>
</tr>
<tr>
<td>pull_request</td>
<td>synchronize</td>
<td>PULL_REQUEST_UPDATED</td>
</tr>
<tr>
<td>pull_request</td>
<td>closed, and the merged field is true</td>
<td>PULL_REQUEST_MERGED</td>
</tr>
<tr>
<td>push</td>
<td>n/a</td>
<td>PUSH</td>
</tr>
</tbody>
</table>

**Note**
The PULL_REQUEST_REOPENED event type can be used with GitHub and GitHub Enterprise only.

- One or more optional filters. Use a regular expression to specify a filter. For an event to trigger a build, every filter 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 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 and GitHub Enterprise push events only.

**Note**
You can find the webhook payload in the webhook settings of your GitHub repository.

**Topics**
- Filter GitHub webhook events (console) (p. 124)
- Filter GitHub webhook events (SDK) (p. 127)
- Filter GitHub webhook events (AWS CloudFormation) (p. 129)

**Filter GitHub 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) (p. 216)](https://docs.aws.amazon.com/codebuild/latest/userguide/build-project.html) and [WebhookFilter](https://docs.aws.amazon.com/codebuild/latest/APIReference/API_WebhookFilter.html) in the [AWS CodeBuild API Reference](https://docs.aws.amazon.com/codebuild/latest/APIReference/).

In this example, a webhook filter group triggers a build for pull requests only:

![Webhook event filter group 1](image)

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/master$` and head references that match `^refs/heads/branch1$`.
- 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 a change is made by a specified GitHub or GitHub Enterprise 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.

---

**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`.

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"
    }
  ]
]
```

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/master$ 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$"
    },
    {
      "type": "BASE_REF",
      "pattern": "^refs/heads/master$"
    }
  ],
  [
    {
      "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_REOPENED, PULL_REQUEST_MERGED"
    },
    {
      "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"
    }
  ]
]
```
You can create a filter that triggers a build only when a change is made by a specified GitHub or GitHub Enterprise 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.

```json
"filterGroups": [
  {
    "type": "EVENT",
    "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED"
  },
  {
    "type": "ACTOR_ACCOUNT_ID",
    "pattern": "actor-account-id"
  }
]
```

### Filter GitHub 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 a 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/master$` 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/.*`.

```yaml
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:2.0
    Source:
      Type: GITHUB
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - Type: EVENT
```
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.
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, 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 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. For **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](p. 149).

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. For **Bucket name**, choose the name of the S3 bucket you configured to function as a website in step 1.

12. 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](p. [138]).

13. Select **Disable artifacts encryption**.

14. Expand **Additional configuration** and choose options as appropriate.

15. Choose **Create build project**. On the build project page, in **Build history**, choose **Start build** to run the build.

16. (Optional) Follow the instructions in [Example: Speed Up Your Website with Amazon CloudFront](p. 269) in the [Amazon S3 Developer Guide](p. 215).

### 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. This sample shows 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](p. 166).

**Note**
If you use Docker in your build container, your build must run in privileged mode. For more information, see [Run a build in AWS CodeBuild](p. 269) and [Create a build project in AWS CodeBuild](p. 215).

### Update your runtime version

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 if you use the Amazon Linux 2 standard image:
phases:
  install:
    runtime-versions:
      java: corretto8

- A `runtime-versions` section that specifies version 11 of Java if you use the Amazon Linux 2 standard image:

phases:
  install:
    runtime-versions:
      java: corretto11

- A `runtime-versions` section that specifies version 8 of Java if you use the Ubuntu standard image 2.0:

phases:
  install:
    runtime-versions:
      java: openjdk8

- A `runtime-versions` section that specifies version 11 of Java if you use the Ubuntu standard image 2.0:

phases:
  install:
    runtime-versions:
      java: openjdk11

The following examples show how you to specify different versions of Node.js using the Ubuntu standard image 2.0 or the Amazon Linux 2 standard image 2.0:

- A `runtime-versions` section that specifies Node.js version 8:

phases:
  install:
    runtime-versions:
      nodejs: 8

- A `runtime-versions` section that specifies Node.js version 10:

phases:
  install:
    runtime-versions:
      nodejs: 10

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

1. Follow steps 1 and 2 in Create the source code (p. 67) to generate source code. If successful, a directory named `my-web-app` is created with your source files.

2. Create a file named `buildspec.yml` with the following contents. Store the file in the `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.yml` 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
```

3. 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. Create a build project. For more information, see Create a build project (console) (p. 216) and Run a build (console) (p. 270). 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/amazonlinux2-x86_64-standard:2.0`.

6. Choose **Start build**.

7. On **Build configuration**, accept the defaults, and then choose **Start build**.
8. 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/*;
```

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

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

10. 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 a runtime dependency**

This example shows how to specify a runtime and a dependency runtime. For example, any supported Android runtime version is dependent on the Java runtime version 8. For example, if you specify Android version 29 and use Amazon Linux 2 or Ubuntu, you can also specify Java version 8. If you do not specify the dependent runtime, CodeBuild attempts to choose it for you.

The build project in this example uses source code in the GitHub AWS Samples repository. The source code uses the Android version 28 runtime and the build project uses Amazon Linux 2, so the buildspec also specifies Java version 8.

2. Create a build project. For more information, see Create a build project (console) (p. 216) and Run a build (console) (p. 270). Leave all settings at their default values, except for these settings.

- For **Environment**:
  - For **Source provider**, choose GitHub.
  - For **Repository**, choose Public repository.
  - For **Environment image**, choose Managed image.
  - For **Operating system**, choose Amazon Linux 2.
  - For **Runtime(s)**, choose Standard.
  - For **Image**, choose aws/codebuild/amazonlinux2-x86_64-standard:2.0.

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:

```plaintext
version: 0.2
phases:
  install:
    runtime-versions:
      android: 29
      java: corretto8
  build:
    commands:
      - ./gradlew assembleDebug
artifacts:
  files:
    - app/build/outputs/apk/app-debug.apk
```

The runtime-versions section specifies both Android version 29 and Java version 8 runtimes.

5. Choose **Create build project**.

6. Choose **Start build**.

7. On **Build configuration**, accept the defaults, and then choose **Start build**.

8. After the build is complete, view the build output on the **Build logs** tab. You should see output similar to the following. It shows that Android version 29 and Java version 8 are installed:

```
[Container] 2019/05/14 23:21:42 Entering phase DOWNLOAD_SOURCES
[Container] Date Time Running command echo "Installing Android version 29 ..."
Installing Android version 29 ...

[Container] Date Time Running command echo "Installing Java version 8 ..."
Installing Java version 8 ...
```

**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.

```javascript
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.

```json
{
    "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 10 and Go version 1.12 runtimes.

```yaml
version: 0.2
phases:
  install:
    runtime-versions:
      golang: 1.13
      nodejs: 10
  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:
```

API Version 2016-10-06
137
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`.

11. Create a build project. For more information, see Create a build project (console) (p. 216) and Run a build (console) (p. 270). 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/amazonlinux2-x86_64-standard:2.0`.

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 '1.12' based on manual selections...
[Container] Date Time Selecting 'nodejs' runtime version '10' based on manual selections...
[Container] Date Time Running command echo "Installing Go version 1.12 ..."
Installing Go version 1.12 ...
[Container] Date Time Running command echo "Installing Node.js version 10 ..."
Installing Node.js version 10 ...
[Container] Date Time Running command n 10.15.3
[Container] Date Time Moving to directory /codebuild/output/src819694850/src
[Container] Date Time Registering with agent
```

API Version 2016-10-06
138
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, 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}).

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

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 123456789012345678901234678901234567890. 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: 
\texttt{refs/heads/branchname}^{\{(full-commit-SHA)\}} \text{(for example, refs/heads/master}^{\{(123456789012345678901234678901234567890)}). If you do this, CodeBuild downloads only the specified branch to find the version.

\textbf{Note}
To speed up the \texttt{DOWNLOAD_SOURCE} phase of your build, you can also set \texttt{Git clone depth} to a low number. CodeBuild downloads fewer versions of your repository.

\textbf{To specify a GitHub repository version with a commit ID}

1. Open the AWS CodeBuild console at \url{https://console.aws.amazon.com/codesuite/codebuild/home}.
2. Create a build project. For information, see Create a build project (console) (p. 216) and Run a build (console) (p. 270). 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 \url{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 \texttt{aws/codebuild/amazonlinux2-x86_64-standard:2.0}.
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:

   \begin{verbatim}
   version: 0.2
   phases:
   install:
     runtime-versions:
       ruby: 2.6
   build:
     commands:
       - echo \$CODEBUILD_RESOLVED_SOURCE_VERSION
   \end{verbatim}

   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 \texttt{echo} command displays the version of the source code stored in the \texttt{CODEBUILD_RESOLVED_SOURCE_VERSION} environment variable.
5. On Build configuration, accept the defaults, and then choose Start build.
6. For Source version, enter \texttt{046e8b67481d53bdc86c3f6affdd5d1afaee6d369}. This is the SHA of a commit in the \url{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.
• 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.

These steps show you how to create a build using the same version of the source. This time, the version of the source is specified using a reference with the commit ID.

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

1. From the left navigation pane, choose **Build projects**, and then choose the project you created earlier.

2. Choose **Start build**.

3. In **Source version**, enter 
   \[
   \text{refs/heads/master}^{046e8b67481d53bdc86c3f6affdd5d1afaee6d369}
   \] This is the same commit ID and a reference to a branch in the format \[
   \text{refs/heads/branchname}^{\text{full-commit-SHA}}
   \].

4. Choose **Start build**.

5. 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
   046e8b67481d53bdc86c3f6affdd5d1afaee6d369
   [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.

### 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.

### 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.

• 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:latest
   docker login
   docker push your-username/repository-name
   ```

2. Follow the steps in Creating a Basic Secret in the AWS Secrets Manager User Guide. In step 3, in Select secret type, do the following:
   a. Choose Other type of secrets.
   
   ![Secret type selection](image)

   b. In Secret key/value, create one key-value pair for your Docker Hub user name and one key-value pair for your Docker Hub password.
   c. For Secret name, enter a name, such as dockerhub. You can enter an optional description to help you remember that this is a secret for Docker Hub.
   d. Leave Disable automatic rotation selected because the keys correspond to your Docker Hub credentials.
   e. Choose Store secret.
   f. When you review your settings, write down the ARN to use later in this sample.

   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 AWS Identity and Management 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) (p. 216).

    **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**, choose **Custom image**.
    b. For **Environment type**, choose **Linux** or **Windows**.
    c. For **Custom image type**, choose **Other location**.
    d. In **Other location**, enter the image location and 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.

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 this 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.

1. Upload your sources to one or more S3 buckets, CodeCommit, GitHub, GitHub Enterprise, or Bitbucket repositories.
2. Choose which source is the primary source. This is the source in which CodeBuild looks for and executes your buildspec file.
3. Create a build project. For more information, see Create a build project in AWS CodeBuild (p. 215).
4. Follow the instructions in Run AWS CodeBuild directly (p. 177) to 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:

```json
{
    "name": "sample-project",
    "source": {
        "type": "S3",
        "location": "bucket/sample.zip"
    }
}
```
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_sourceIdentifier`. For more information, see Environment variables in build environments (p. 172).

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 `artifactIdentifier`.

**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:
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 `start-build` command in the AWS CLI might look like:

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

**Project without a source sample**

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:

```json
{
  "name": "project-name",
  "source": {
    "type": "NO_SOURCE",
    "buildspec": "version: 0.2
phases:
  build:
    commands:
      - command"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:2.0",
    "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) (p. 229).

To learn how to create a pipeline that uses multiple source inputs to CodeBuild to create multiple output artifacts, see AWS CodePipeline integration with CodeBuild and multiple input sources and output artifacts sample (p. 63).
Use semantic versioning to name build artifacts sample

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) (p. 216).
   - 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) (p. 229).
   - 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:

```yaml
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 (p. 172).

```yaml
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:

```yaml
version: 0.2
env:
```

---

**AWS CodeBuild User guide**

**Use semantic versioning to name build artifacts sample**
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 (p. 172).

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 (p. 149).
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.

<table>
<thead>
<tr>
<th>Repository provider</th>
<th>Required</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeCommit</td>
<td>Repository name. (Optional) Commit ID associated with the source code.</td>
<td>See these topics in the AWS CodeCommit User Guide: Create a CodeCommit Repository Create a Commit in CodeCommit</td>
</tr>
<tr>
<td>Amazon S3</td>
<td>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.</td>
<td>See these topics in the Amazon S3 Getting Started Guide: Create a Bucket Add an Object to a Bucket</td>
</tr>
<tr>
<td>GitHub</td>
<td>Repository name. (Optional) Commit ID associated with the source code.</td>
<td>See this topic on the GitHub Help website: Create a Repo</td>
</tr>
<tr>
<td>Bitbucket</td>
<td>Repository name. (Optional) Commit ID associated with the source code.</td>
<td>See this topic on the Bitbucket Cloud documentation website: Create a repository</td>
</tr>
</tbody>
</table>

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 (p. 149).

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 (p. 166).

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 (p. 178).

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 AWS CodeBuild directly (p. 177).
- 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 AWS CodePipeline with AWS CodeBuild (p. 195).

**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 (p. 3).

**Topics**

- Buildspec file name and storage location (p. 149)
- Buildspec syntax (p. 150)
- Buildspec example (p. 162)
- Buildspec versions (p. 164)

**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:
AWS CodeBuild User guide
Buildspec syntax

• 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 (p. 215) or Change a build project's
settings (p. 252).
• 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 a build (p. 269).
• 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

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:
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: yes
proxy:
upload-artifacts: yes
logs: yes
phases:
install:
run-as: Linux-user-name
runtime-versions:
runtime: version
runtime: version
commands:
- command

API Version 2016-10-06
150


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 (p. 164).

- **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 execute 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.

- **env**: Optional sequence. Represents information for one or more custom environment variables.
  - **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, especially AWS access key IDs and secret access keys, 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 a build in AWS CodeBuild (p. 269).
- 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 (p. 215) and Change a build project's settings in AWS CodeBuild (p. 252).
- The value in the buildspec declaration takes lowest precedence.

- **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 Systems Manager Parameter Store Console Walkthrough 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 Create a CodeBuild service role (p. 348).

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 a build in AWS CodeBuild (p. 269).
- 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 (p. 215) and Change a build project's settings in AWS CodeBuild (p. 252).
- The value in the buildspec declaration takes lowest precedence.

- secrets-manager: Required if env specified, and you want to retrieve custom environment variables stored in AWS Secrets Manager. Specify a Secrets Manager reference-key using the following pattern:

  secret-id:json-key:version-stage:version-id

  - secret-id: 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: Specifies the key name of the 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: 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: 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.

For more information, see What Is AWS Secrets Manager in the AWS Secrets Manager User Guide.

- 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.

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_.
• git-credential-helper: Optional mapping. Used to indicate if CodeBuild uses its Git credential helper to provide Git credentials. Set true if it is used. Otherwise, no or not specified. For more information, see the gitcredentials documentation 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 (p. 189).

• upload-artifacts: Optional mapping. Set to true if you want your build in an explicit proxy server to upload artifacts. The default is no.

• logs: Optional mapping. Set to true for your build in an 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 (p. 172).

• 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 the install buildspec file are run as User-1 except commands in the install phase, which are run as User-2.

The allowed build phase names are:

• 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.

• runtime-versions: Optional sequence. A runtime version is supported with the Ubuntu standard image 2.0 or later and the Amazon Linux 2 standard image 1.0 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, java: openjdk11, ruby: 2.6, nodejs: 12.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 2.0, then the following specifies that version 8 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 (p. 166).

```yaml
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.
- The following supported runtimes can be specified.

### Ubuntu 18.04 and Amazon Linux 2 platforms runtimes

<table>
<thead>
<tr>
<th>Runtime name</th>
<th>Version/versions</th>
<th>How to specify</th>
</tr>
</thead>
<tbody>
<tr>
<td>android</td>
<td>• 28</td>
<td>• android: 28</td>
</tr>
<tr>
<td>android</td>
<td>• 29</td>
<td>• android: 29</td>
</tr>
<tr>
<td>dotnet</td>
<td>• 3.0</td>
<td>• dotnet: 3.0</td>
</tr>
<tr>
<td>dotnet</td>
<td>• 3.1</td>
<td>• dotnet: 3.1</td>
</tr>
</tbody>
</table>
### Buildspec syntax

<table>
<thead>
<tr>
<th>Language</th>
<th>Version</th>
<th>Version</th>
<th>Language</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>golang</td>
<td>1.13</td>
<td>1.14</td>
<td>golang</td>
<td>1.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>golang</td>
<td>1.14</td>
<td></td>
<td>golang</td>
<td>1.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nodejs</td>
<td>10</td>
<td>12</td>
<td>nodejs</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java</td>
<td>openjdk8</td>
<td>openjdk11</td>
<td>java</td>
<td>openjdk8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>openjdk11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java</td>
<td>corretto8</td>
<td>corretto11</td>
<td>java</td>
<td>corretto8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>corretto11</td>
</tr>
<tr>
<td>Language</td>
<td>Version</td>
<td>Image Version</td>
<td>Syntax</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>---------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>php</td>
<td>• 7.3</td>
<td></td>
<td>• php: 7.3</td>
<td></td>
</tr>
<tr>
<td>php</td>
<td>• 7.4</td>
<td></td>
<td>• php: 7.4</td>
<td></td>
</tr>
<tr>
<td>python</td>
<td>• 3.7</td>
<td></td>
<td>• python: 3.7</td>
<td></td>
</tr>
<tr>
<td>python</td>
<td>• 3.8</td>
<td></td>
<td>• python: 3.8</td>
<td></td>
</tr>
<tr>
<td>ruby</td>
<td>• 2.6</td>
<td></td>
<td>• ruby: 2.6</td>
<td></td>
</tr>
<tr>
<td>ruby</td>
<td>• 2.7</td>
<td></td>
<td>• ruby: 2.7</td>
<td></td>
</tr>
</tbody>
</table>
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."

- `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.

- `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.

  - `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.

- `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.

  - `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.

- `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.

  - `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.

Important
Commands in some build phases might not be run if commands in earlier build phases fail. For example, if a command fails during the `install` phase, none of the commands in the `pre_build`, `build`, and `post_build` phases are run for that build's lifecycle. For more information, see Build phase transitions (p. 279).

- `finally`: Optional block. Commands specified in a `finally` block are executed after commands in the `commands` block. The commands in a `finally` block are executed 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.

- `report-name-or-arn`: Optional sequence. Represents information about where you want the files with your test results. A project can have a maximum of five report groups. Specify a name for a new report group or the ARN of an existing 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-in-buildspec`. For more information, see Report group naming (p. 293).

- `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`.

API Version 2016-10-06
- `my-subdirectory/**/*` represents all files recursively starting from a subdirectory named `my-subdirectory`.

- **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.

- **discard-paths**: Optional mapping. Represents whether paths to test result files uploaded to an S3 bucket are discarded. **yes** if paths are discarded. Otherwise, **no** or not specified (the default). For example, if a path to a test result is `com/myapp/mytests/TestResult.xml`, specifying **yes** shortens this path to `TestResult.xml`.

- **file-format**: Optional mapping. Represents the test file format. If not specified, `JUnitXml` is used. The valid values are:
  - `JUnitXml`
  - `CucumberJson`
  - `VisualStudioTrx`
  - `TestNGXml`

- **artifacts**: Optional sequence. Represents information about where CodeBuild can find the build output and how CodeBuild prepares it for uploading to the Amazon 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.

- **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.

- **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)](p. 216).

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.
This is an example of an artifact name that uses a CodeBuild environment variable. For more information, see Environment variables in build environments (p. 172).

This is an example of an artifact name that uses a CodeBuild environment variable with the artifact's creation date appended to it.

- **discard-paths**: Optional mapping. Represents whether paths to files in the build output artifact are discarded. `yes` if paths are discarded; otherwise, `no` or not specified (the default). For example, if a path to a file in the build output artifact would be `com/mycompany/app/HelloWorld.java`, then specifying `yes` would shorten this path to simply `HelloWorld.java`.

- **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-build1
|   |-- my-file1.txt
|   `-- my-build2
|       |-- my-file2.txt
|       `-- my-subdirectory
|           `-- my-file3.txt
```

And for the following **artifacts** sequence:
`artifacts:
    files:
        - '*/my-file3.txt'
    base-directory: my-build2`

The following subdirectory and file would be included in the build output artifact:

```bash
my-subdirectory
  `-- my-file3.txt```

While for the following `artifacts` sequence:

```bash
artifacts:
    files:
        - '**/*'
    base-directory: 'my-build2'
    discard-paths: yes
```

The following files would be included in the build output artifact:

```bash
|-- my-file1.txt
|-- my-file2.txt
 `-- my-file3.txt```

- `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. For example, if your project has the following structure:

```json
{
    "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:

```bash
version: 0.2
phases:
    build:
        commands:
            - echo Building...
    artifacts:
        secondary-artifacts:
```
artifact1:
  files:
  - directory/file
  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 Amazon S3 cache bucket. This sequence is not required if the cache type of the project is No 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

The test reporting feature is in preview release for CodeBuild and is subject to change.

Here is an example of a buildspec.yml file.

```yaml
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...
      - 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/messageUtil-1.0.jar
        discard-paths: yes
      artifact2:
        files:
          - target/messageUtil-1.0.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\nvariables:\n  JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64\n  LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword\nphases:\ninstall:\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\npre_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\nbuilt:
  commands:\n    - echo Entered the build phase...\n    - mvn install\n  finally:\n    - echo This always runs even if the install command fails\npost_build:
  commands:\n    - echo Entered the post_build phase...\n    - 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/messageUtil-1.0.jar
        discard-paths: yes
      artifact2:
        files:
          - target/messageUtil-1.0.jar
        discard-paths: yes
  cache:
    paths:
      - '/root/.m2/**/*'
```

API Version 2016-10-06

163
- echo This always runs even if the login command fails
- echo Entered the build phase...
- mvn install finally:
- echo This always runs even if the install command fails
- echo Entered the post_build phase...
- echo Build completed on `date`

```
reports:
  reportGroupJunitXml:
    files:
      - "**/*"
    base-directory: 'target/tests/reports'
    discard-paths: false

reportGroupCucumberJson:
    files:
      - 'cucumber/target/cucumber-tests.xml'

file-format: CucumberJson
```

artifacts:
  files:
    - target/messageUtil-1.0.jar
  discard-paths: yes
  secondary-artifacts:
    artifact1:
      files:
       - target/messageUtil-1.0.jar
      discard-paths: yes
    artifact2:
      files:

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.
## 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* (p. 3).

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* (p. 166).
- To get the identifier of a publicly available Docker image stored in Docker Hub, see *Searching for Images* 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* (p. 53).

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 types* (p. 170).

**Topics**
- *Docker images provided by CodeBuild* (p. 166)
- *Build environment compute types* (p. 170)
- *Shells and commands in build environments* (p. 172)
- *Environment variables in build environments* (p. 172)
- *Background tasks in build environments* (p. 174)

---

### Version Changes

<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 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. |
Docker images provided by CodeBuild

AWS CodeBuild manages the following Docker images that are available in the CodeBuild and AWS CodePipeline consoles.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Image identifier</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Linux 2</td>
<td>aws/codebuild/amazonlinux2-x86_64-standard:2.0</td>
<td>al2/standard/2.0</td>
</tr>
<tr>
<td>Amazon Linux 2</td>
<td>aws/codebuild/amazonlinux2-x86_64-standard:3.0</td>
<td>al2/standard/3.0</td>
</tr>
<tr>
<td>Amazon Linux 2</td>
<td>aws/codebuild/amazonlinux2-aarch64-standard:1.0</td>
<td>al2/aarch64/standard/1.0</td>
</tr>
<tr>
<td>Ubuntu 18.04</td>
<td>aws/codebuild/standard:2.0</td>
<td>ubuntu/standard/2.0</td>
</tr>
<tr>
<td>Ubuntu 18.04</td>
<td>aws/codebuild/standard:3.0</td>
<td>ubuntu/standard/3.0</td>
</tr>
<tr>
<td>Ubuntu 18.04</td>
<td>aws/codebuild/standard:4.0</td>
<td>ubuntu/standard/4.0</td>
</tr>
<tr>
<td>Windows Server Core 2016</td>
<td>aws/codebuild/windows-base:2.0</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The latest version of each image 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/amazonlinux2-x86_64-standard:2.0` instead of a more granular version, such as `aws/codebuild/amazonlinux2-x86_64-standard:2.0-1.0.0`.

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.

When you specify a runtime in the `runtime-versions` section of your buildspec file, you can specify its major version, its major version with the latest minor version, or the latest major and minor version. The following table lists the available runtimes and how to specify them.

**Ubuntu 18.04 and Amazon Linux 2 platforms runtimes**

<table>
<thead>
<tr>
<th>Runtime name</th>
<th>Version/versions</th>
<th>How to specify a version</th>
</tr>
</thead>
<tbody>
<tr>
<td>android</td>
<td>• 28</td>
<td>• android: 28</td>
</tr>
<tr>
<td>Container</td>
<td>Version 1</td>
<td>Version 2</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>android</td>
<td>29</td>
<td>android: 29</td>
</tr>
<tr>
<td>dotnet</td>
<td>3.0</td>
<td>dotnet: 3.0</td>
</tr>
<tr>
<td>dotnet</td>
<td>3.1</td>
<td>dotnet: 3.1</td>
</tr>
<tr>
<td>golang</td>
<td>1.13, 1.14</td>
<td>golang: 1.13, 1.14</td>
</tr>
</tbody>
</table>
## Docker images provided by CodeBuild

<table>
<thead>
<tr>
<th>Language</th>
<th>Version</th>
<th>Image Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>golang</strong></td>
<td>1.14</td>
<td>golang: 1.14</td>
</tr>
<tr>
<td><strong>nodejs</strong></td>
<td>10</td>
<td>nodejs: 10</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>nodejs: 12</td>
</tr>
<tr>
<td><strong>java</strong></td>
<td>openjdk8</td>
<td>java: openjdk8</td>
</tr>
<tr>
<td></td>
<td>openjdk11</td>
<td>java: openjdk11</td>
</tr>
<tr>
<td><strong>java</strong></td>
<td>corretto8</td>
<td>java: corretto8</td>
</tr>
<tr>
<td></td>
<td>corretto11</td>
<td>java: corretto11</td>
</tr>
<tr>
<td><strong>php</strong></td>
<td>7.3</td>
<td>php: 7.3</td>
</tr>
<tr>
<td><strong>php</strong></td>
<td>7.4</td>
<td>php: 7.4</td>
</tr>
</tbody>
</table>
### Python

<table>
<thead>
<tr>
<th>Version</th>
<th>Python:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7</td>
<td>3.7</td>
</tr>
<tr>
<td>3.8</td>
<td>3.8</td>
</tr>
</tbody>
</table>

### Ruby

<table>
<thead>
<tr>
<th>Version</th>
<th>Ruby:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6</td>
<td>2.6</td>
</tr>
<tr>
<td>2.7</td>
<td>2.7</td>
</tr>
</tbody>
</table>

**Note**

The `aws/codebuild/amazonlinux2-aarch64-standard:1.0` image does not support the Android Runtime (ART).

The base image of the Windows Server Core 2016 contains the following runtimes.

<table>
<thead>
<tr>
<th>Runtime name</th>
<th>Version in <code>windows-base:2.0</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>dotnet</td>
<td>2.2, 3.1</td>
</tr>
<tr>
<td>golang</td>
<td>1.13</td>
</tr>
<tr>
<td>nodejs</td>
<td>10.18, 12.14</td>
</tr>
<tr>
<td>java</td>
<td>openjdk11</td>
</tr>
</tbody>
</table>
## Build environment compute types

AWS CodeBuild provides build environments with the following available memory, vCPUs, and disk space:

**Operating system: Linux**

<table>
<thead>
<tr>
<th>Compute type</th>
<th>computeType value</th>
<th>Memory</th>
<th>vCPUs</th>
<th>Disk space</th>
<th>Environment type</th>
</tr>
</thead>
<tbody>
<tr>
<td>build.general1.small</td>
<td>BUILDGENERAL1_SMALL</td>
<td>2</td>
<td>64 GB</td>
<td>64 GB</td>
<td>LINUX_CONTAINER</td>
</tr>
<tr>
<td>build.general1.medium</td>
<td>BUILDGENERAL1_MEDIUM</td>
<td>4</td>
<td>128 GB</td>
<td>128 GB</td>
<td>LINUX_CONTAINER</td>
</tr>
<tr>
<td>build.general1.large</td>
<td>BUILDGENERAL1_LARGE</td>
<td>8</td>
<td>128 GB</td>
<td>128 GB</td>
<td>LINUX_CONTAINER</td>
</tr>
<tr>
<td>build.general1.xlarge</td>
<td>BUILDGENERAL1_XLARGE</td>
<td>32</td>
<td>50 GB</td>
<td>50 GB</td>
<td>LINUX_GPU_CONTAINER</td>
</tr>
<tr>
<td>build.general1.2xlarge</td>
<td>BUILDGENERAL1_2XLARGE</td>
<td>8</td>
<td>50 GB</td>
<td>50 GB</td>
<td>ARM_CONTAINER</td>
</tr>
<tr>
<td>build.general1.4xlarge</td>
<td>BUILDGENERAL1_4XLARGE</td>
<td>72</td>
<td>824 GB (SSD)</td>
<td>824 GB (SSD)</td>
<td>LINUX_CONTAINER</td>
</tr>
</tbody>
</table>

The disk space listed for each build environment is available only in the directory specified by the CODEBUILD_SRC_DIR environment variable.

Note

The base image of the Windows Server Core 2016 platform is available in the US East (N. Virginia), US East (Ohio), US West (Oregon), and Europe (Ireland) regions only.

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 (p. 162).

CodeBuild frequently updates the list of Docker 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) (p. 216) or Change a build project's settings (console) (p. 253).
- For the AWS CLI, run the list-curated-environment-images command:

  ```bash
  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 (p. 356).
### Build environment compute types

**Note**

Some environment and compute types have limitations:

- The environment type `LINUX_GPU_CONTAINER` is available only in regions US East (N. Virginia), US West (Oregon), Canada (Central), Europe (Ireland), Europe (London), Europe (Frankfurt), Asia Pacific (Tokyo), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), China (Beijing), and China (Ningxia).

- The environment type `ARM_CONTAINER` is available only in regions US East (N. Virginia), US East (Ohio), US West (Oregon), Europe (Ireland), Asia Pacific (Mumbai), Asia Pacific (Tokyo), Asia Pacific (Sydney), and Europe (Frankfurt).

- The compute type `build.general1.2xlarge` is available only in regions US East (N. Virginia), US East (Ohio), US West (N. California), US West (Oregon), Canada (Central), South America (São Paulo), Europe (Stockholm), Europe (Ireland), Europe (London), Europe (Paris), Europe (Frankfurt), Middle East (Bahrain), Asia Pacific (Hong Kong), Asia Pacific (Tokyo), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Mumbai), China (Beijing), and China (Ningxia).

- For the compute type `build.general1.2xlarge`, Docker images up to 100 GB uncompressed are supported.

#### Operating system: Windows

<table>
<thead>
<tr>
<th>Compute type</th>
<th>computeType value</th>
<th>Memory</th>
<th>vCPUs</th>
<th>Disk space</th>
<th>Environment type</th>
</tr>
</thead>
<tbody>
<tr>
<td>build.general</td>
<td>BUILD_GENERAL1_MEDIUM</td>
<td>4</td>
<td>128 GB</td>
<td></td>
<td>WINDOWS_CONTAINER</td>
</tr>
<tr>
<td>build.general</td>
<td>BUILD_GENERAL1_LARGE</td>
<td>8</td>
<td>128 GB</td>
<td></td>
<td>WINDOWS_CONTAINER</td>
</tr>
</tbody>
</table>

**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.

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) (p. 216) or Change a build project's settings (console) (p. 253).

- 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) (p. 229) or Change a build project's settings (AWS CLI) (p. 264).

- 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 (p. 356).

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 (p. 56). If you want to manipulate container disk space during a build, then the build must run in privileged mode.

**Note**

By default, Docker containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.
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 (p. 149).
- 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) (p. 216).
- 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) (p. 253).
- 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 (p. 215) or Change a build project's settings (p. 252).
- 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 a build (p. 269).

You can specify any Shell 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 run in a Windows Server Core 2016 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.
Environment variables in build environments

- **CODEBUILD_BUILD_ID**: The CodeBuild ID of the build (for example, codebuild-demo-project:ble6661e-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 IAM 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_LOG_PATH**: The log stream name in CloudWatch Logs for the build.
- **CODEBUILD_RESOLVED_SOURCE_VERSION**: An identifier for the version of a build's source code. Its format depends on the source code repository:
  - For CodeCommit, GitHub, GitHub Enterprise, and Bitbucket, it is the commit ID. For these repositories, CODEBUILD_RESOLVED_SOURCE_VERSION is only available after the DOWNLOAD_SOURCE phase.
  - For CodePipeline, it is the source revision is provided by CodePipeline. For CodePipeline, the CODEBUILD_RESOLVED_SOURCE_VERSION environment variable may not always be available.
  - For Amazon S3, this does not apply.
- **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, then this might be empty.
- **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, 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 build that is triggered by a webhook pull request event, it is pr/pull-request-number.
- **CODEBUILD_SRC_DIR**: The directory path that CodeBuild uses for the build (for example, /tmp/src123456789/src).
  
  **Note**
  If you use a secondary source, the environment variable for its 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 (p. 143).
- **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_PREV_COMMIT**: The ID of the most recent commit before the webhook push event that triggers the current build.
You can also provide build environments with your own environment variables. For more information, see the following topics:

- Use AWS CodePipeline with AWS CodeBuild (p. 195)
- Create a build project (p. 215)
- Change a build project's settings (p. 252)
- Run a build (p. 269)
- Buildspec reference (p. 149)

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 and secret access keys. 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 Systems Manager Parameter Store Console Walkthrough in the Amazon EC2 Systems Manager User Guide. To retrieve them, see the parameter-store mapping in Buildspec syntax (p. 150).

### 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 $!
  ```
Test and debug locally with the AWS CodeBuild agent

This topic provides information about how to run the AWS CodeBuild agent and subscribe to notifications about new versions of the agent.

Test and debug on a local machine with the CodeBuild agent

You can use the AWS CodeBuild agent to test and debug builds on a local machine.

To use the agent

1. Download the `codebuild.sh` script.
2. Run the script and specify your container images and output directory:

   ```
   codebuild_build.sh [-i image_name] [-a artifact_output_directory] [options]
   ```

The CodeBuild agent is available from https://hub.docker.com/r/amazon/aws-codebuild-local/. Its Secure Hash Algorithm (SHA) signature is 94467b3eeac4184d28a38febc27a1530691527dd49c17e30ad1b6331d791e82f5. You can use this to identify the version of the agent. To see the agent's SHA signature, run the following command:

   ```
   docker inspect amazon/aws-codebuild-local
   ```

Receive notifications for new CodeBuild agent versions

You can subscribe to Amazon SNS notifications so you know when new versions of the AWS CodeBuild agent are released. Follow the steps in this procedure to subscribe to these notifications.

To subscribe to the CodeBuild agent notifications

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:
   
   For Topic ARN, use the following Amazon Resource Name (ARN):

```
To unsubscribe from CodeBuild agent notifications

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 AWS CodeBuild directly

You can use the AWS CodeBuild console, AWS CLI, or AWS SDK to set up, run, and monitor builds directly with CodeBuild.

Not what you’re looking for? To use AWS CodePipeline to run CodeBuild, see Use AWS CodePipeline with AWS CodeBuild (p. 195).

Topics
• Prerequisites (p. 177)
• Run AWS CodeBuild directly (p. 177)

Prerequisites

Answer the questions in Plan a build (p. 148).

Run AWS CodeBuild directly

1. Create the build project. To use the console, see Create a build project (console) (p. 216). To use the AWS CLI, see Create a build project (AWS CLI) (p. 229).
2. Run the build. To use the console, see Run a build (console) (p. 270). To use the AWS CLI, see Run a build (AWS CLI) (p. 273).
3. Get information about the build. To use the console, see View build details (console) (p. 278). To use the AWS CLI, see View build details (AWS CLI) (p. 278).
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.

**Note**

VPC connectivity from CodeBuild is not supported in Windows.

**Topics**
- Use cases (p. 178)
- Allowing Amazon VPC access in your CodeBuild projects (p. 178)
- Best practices for VPCs (p. 179)
- Troubleshooting your VPC setup (p. 180)
- Use VPC endpoints (p. 180)
- AWS CloudFormation VPC template (p. 182)
- Use AWS CodeBuild with a proxy server (p. 186)

**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.

**Allowing 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) (p. 216). 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) (p. 229). 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 (p. 320).

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 to 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.

## 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. 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 execute 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. 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 (p. 182).

Troubleshooting 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 Create a CodeBuild service role (p. 348).

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 Accessing AWS Services Through PrivateLink in the Amazon VPC User Guide.

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.
Creating 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.

```json
{
    "Statement": [
        {
            "Action": [
                "codebuild:StartBuild",
                "codebuild:BatchGetBuilds",
                "codebuild:ListBuildsForProject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:codebuild:region:account-id:project/project-name"
        }
    ]
}
```
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.

---

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

API Version 2016-10-06
184
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:
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 a build project’s settings in AWS CodeBuild (p. 252) and Create a build project in AWS CodeBuild (p. 215).

- **Transparent proxy**: If you use a transparent proxy server, no special configuration is required.

**Topics**

- Components required to run CodeBuild in a proxy server (p. 186)
- Run CodeBuild in an explicit proxy server (p. 189)
- Run CodeBuild in a transparent proxy server (p. 192)
- Run a package manager and other tools in a proxy server (p. 193)

**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 communication 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:
Run CodeBuild in an explicit proxy server

Topics

- Configure Squid as an explicit proxy server (p. 189)
- Create a CodeBuild project (p. 190)
- Explicit proxy server sample squid.conf file (p. 190)

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.

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.

```bash
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.

```bash
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:

```bash
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
   ```

   2. Add `proxy` to your buildspec file. For more information, see Buildspec syntax (p. 150).

   ```
   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 (p. 371).

   For more information, see Explicit proxy server sample `squid.conf` file (p. 190) 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 (p. 215) and Change a build project's settings in AWS CodeBuild (p. 252).

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
```
# Run CodeBuild in an explicit proxy server

```plaintext
# add all URLS to be whitelisted for download source and commands to be executed 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 execute 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.
```

API Version 2016-10-06
# 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.

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
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 --port 3129
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 --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

To execute 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 execute in the build environment.

```
// acl allowed_sites dstdomain ppa.launchpad.net # Required for apt-get to execute in the build environment
acl apt_get dstdom_regex .*\.launchpad.net # Required for CodeBuild to execute apt-get in the build environment
acl apt_get dstdom_regex .*\.ubuntu.com # Required for CodeBuild to execute 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";
```

**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 proxy server.

```
// acl allowed_sites dstdomain ppa.launchpad.net # Required to execute apt-get in the build environment
acl allowed_sites dstdomain google.com # Required for access to a website. 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`.

```
echo 'Acquire::http::Proxy "http://<private-ip-of-proxy-server>:3128";
```
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.

```plaintext
acl allowed_sites dstdomain ppa.launchpad.net # Required to execute 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.

```plaintext
maven clean install -DproxySet=true -DproxyHost=<private-ip-of-proxy-server> -DproxyPort=3128
```
Use CodePipeline with AWS CodeBuild 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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Available approaches</th>
<th>Approaches described in this topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a continuous delivery (CD) pipeline with CodePipeline that automates builds with CodeBuild</td>
<td>• CodePipeline console</td>
<td>• Use the CodePipeline console (p. 197)</td>
</tr>
<tr>
<td></td>
<td>• AWS CLI</td>
<td>• Use the AWS CLI (p. 199)</td>
</tr>
<tr>
<td></td>
<td>• AWS SDKs</td>
<td>• 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.</td>
</tr>
</tbody>
</table>

| | Add test and build automation with CodeBuild to an existing pipeline in CodePipeline | • Use the CodePipeline console to add build automation (p. 202) |
| | • AWS CLI | • Use the CodePipeline console to add test automation (p. 206) |
| | • AWS SDKs | • 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 SDKspipeline. For more information, reference 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 (p. 196)
- Create a pipeline that uses CodeBuild (CodePipeline console) (p. 197)
- Create a pipeline that uses CodeBuild (AWS CLI) (p. 199)
- Add a CodeBuild build action to a pipeline (CodePipeline console) (p. 202)
- Add a CodeBuild test action to a pipeline (CodePipeline console) (p. 206)
Prerequisites

1. Answer the questions in Plan a build (p. 148).

2. If you are using an IAM user to access CodePipeline instead of an AWS root account or an administrator IAM 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 IAM 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 IAM 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:

   ```bash
   ```

   For Windows:

   ```bash
   ```

   **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, or GitHub. (CodePipeline does not currently support Bitbucket.) 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 (p. 149).

   **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 CodeDeploy, see the AWS CodeDeploy sample (p. 59) in this guide and Prepare a Revision for CodeDeploy in the AWS CodeDeploy User Guide.
   - For AWS Elastic Beanstalk, see the AWS Elastic Beanstalk sample (p. 67) in this guide and Create an Application Source Bundle in the AWS Elastic Beanstalk Developer 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) (p. 206) 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) (p. 206) 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 IAM User Guide.
   - An administrator IAM user in your AWS account. For more information, see Creating Your First IAM Admin User and Group in the IAM User Guide.
   - An IAM user in your AWS account with permission to use the following minimum set of actions:


```plaintext
codepipeline::*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codemlmit:ListBranches
codemlmit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:GetApplications
codedeploy:GetDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

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 ahead to step 22 in this procedure. Otherwise, use the following steps to create a project in CodeBuild.

   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 Create a build project (console) (p. 216) or Change a build project's settings (console) (p. 253).

   **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) (p. 253).
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)** (p. 278).

15. Open the Amazon S3 console at [https://console.aws.amazon.com/s3/](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:

   ```bash
   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 (p. 215).

   **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) (p. 229).
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:

```json
{
   "pipeline": {
      "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
      "stages": [
         {
            "name": "Source",
            "actions": [
               {
                  "name": "Source",
                  "actionTypeId": {
                     "category": "Source",
                     "owner": "AWS",
                     "version": "1",
                     "provider": "S3"
                  },
                  "outputArtifacts": [
                     {
                        "name": "MyApp"
                     }
                  ],
                  "configuration": {
                     "S3Bucket": "my-input-bucket-name",
                     "S3ObjectKey": "my-source-code-file-name.zip"
                  },
                  "runOrder": 1
               }
            ]
         },
         {
            "name": "Build",
            "actions": [
               {
                  "name": "Build",
                  "actionTypeId": {
                     "category": "Build",
                     "owner": "AWS",
                     "version": "1",
                     "provider": "CodeBuild"
                  },
                  "outputArtifacts": [
                     {
                        "name": "default"
                     }
                  ],
                  "configuration": {
                     "ProjectName": "my-build-project-name"
                  },
                  "runOrder": 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:

```bash
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:

```bash
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.

```json
{
...  
"stageStates": [
...
  
  "actionStates": [
    
    "actionName": "CodeBuild",
    "latestExecution": {
      "status": "SUCCEEDED",
    }"name": "my-pipeline-name",
"version": 1
}
}
Add a CodeBuild build action to a pipeline (CodePipeline console)

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) (p. 278).

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 my-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, `my-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 *IAM User Guide*.
   - An administrator IAM user in your AWS account. For more information, see Creating Your First IAM Admin User and Group in the *IAM User Guide*.
   - An IAM 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
   ```

   API Version 2016-10-06
Add a CodeBuild build action to a pipeline (CodePipeline console)


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 the build action 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 in CodeBuild, for Project name, choose the name of the build project, and then skip to step 22 of 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) (p. 216) or Change a build project's settings (console) (p. 253).

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) (p. 253).


14. 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.

15. For Project name, enter a name for this build project. Build project names must be unique across each AWS account.

16. (Optional) Enter a description.

17. For Environment, do one of the following:

- To use a build environment based on a Docker image that is managed by CodeBuild, choose Managed image. Make your selections from the Operating system, Runtime, and Runtime version drop-down lists. For more information, see Docker images provided by CodeBuild (p. 166).

- To use a build environment based on a Docker image in an Amazon ECR repository in your AWS account, choose Custom image. For Environment type, choose an environment type, and then choose Amazon ECR. Use the Amazon ECR repository and Amazon ECR image drop-down lists to choose the Amazon ECR repository and Docker image in that repository.

- To use a build environment based on a publicly available Docker image in Docker Hub, choose Other location. In Other location, enter the Docker image ID, using the format docker repository/docker-image-name.

Select Privileged only if you plan to use this build project to build Docker images, and the build environment image you chose is not one provided by CodeBuild with Docker support. 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 as needed. You can do this by running the following build commands to initialize the Docker daemon in the install phase of your buildspec. (Do not run the following build commands if you chose a build environment image provided by CodeBuild with Docker support.)

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcp://127.0.0.1:2375 --storage-driver=overlay &
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

18. 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.
Add a CodeBuild build action to a pipeline (CodePipeline console)

19. Expand **Additional configuration**.

   To specify a build timeout other than 60 minutes (the default), use the **hours** and **minutes** boxes to set a timeout between 5 and 480 minutes (8 hours).

   For **Compute**, choose one of the available options.

   For **Environment variables**, use **Name** and **Value** to specify any optional environment variables for the build environment to use. To add more environment variables, choose **Add environment variable**.

   **Important**
   We strongly discourage storing sensitive values, especially AWS access key IDs and secret access keys, in environment variables. Environment variables can be displayed in plain text in the CodeBuild console and AWS CLI.

   To store and retrieve sensitive values, we recommend your build commands use the AWS CLI to interact with the Amazon EC2 Systems Manager Parameter Store. The AWS CLI is already installed and configured on all build environments provided by CodeBuild. For more information, see **Systems Manager Parameter Store** and **Systems Manager Parameter Store CLI Walkthrough** in the *Amazon EC2 Systems Manager User Guide*.

20. For **Buildspec**, do one of the following:

   - If your source code includes a buildspec file, choose **Use a buildspec file**.
   - If your source code does not include a buildspec file, choose **Insert build commands**. For **Build commands**, enter the commands you want to run during the build phase in the build environment. For multiple commands, separate each command with `&&` for Linux-based build environments or `;` for Windows-based build environments. For **Output files**, enter the paths to the build output files in the build environment that you want to send to CodePipeline. For multiple files, separate each file path with a comma.

21. Choose **Create build project**.

22. Return to the CodePipeline console.

23. For **Input artifacts**, choose the output artifact that you noted earlier in this procedure.

24. For **Output artifacts**, enter a name for the output artifact (for example, **MyAppBuild**).

25. Choose **Add action**.

26. Choose **Save**, and then choose **Save** to save your changes to the pipeline.

27. Choose **Release change**.

28. 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)** (p. 278), and then skip to step 31 of this procedure.

29. Open the Amazon S3 console at [https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).

30. 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:
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 IAM User Guide.
   - An administrator IAM user in your AWS account. For more information, see Creating Your First IAM Admin User and Group in the IAM User Guide.
   - An IAM user in your AWS account with permission to perform the following minimum set of actions:

```bash
codepipeline:*
im:ListRoles
im:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:GetApplications
codedeploy:GetDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```


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 in CodeBuild, for Project name, choose the name of the build project, and then skip to step 22 of 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) (p. 253)


14. 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.

15. For Project name, enter a name for this build project. Build project names must be unique across each AWS account.

16. (Optional) Enter a description.

17. For Environment, do one of the following:
Add a CodeBuild test action to a pipeline (CodePipeline console)

- To use a build environment based on a Docker image that is managed by CodeBuild, choose Managed image. Make your selections from the Operating system, Runtime, and Runtime version drop-down lists. For more information, see Docker images provided by CodeBuild (p. 166).
- To use a build environment based on a Docker image in an Amazon ECR repository in your AWS account, choose Custom image. For Environment type, choose an environment type, and then choose Amazon ECR. Use the Amazon ECR repository and Amazon ECR image drop-down lists to choose the Amazon ECR repository and Docker image in that repository.
- To use a build environment based on a publicly available Docker image in Docker Hub, choose Other location. In Other location, enter the Docker image ID, using the format docker repository/docker-image-name.

Select Privileged only if you plan to use this build project to build Docker images, and the build environment image you chose is not one provided by CodeBuild with Docker support. 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 as needed. You can do this by running the following build commands to initialize the Docker daemon in the install phase of your buildspec. (Do not run the following build commands if you chose a build environment image provided by CodeBuild with Docker support.)

```bash
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

18. 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.

19. Expand Additional configuration.

To specify a build timeout other than 60 minutes (the default), use the hours and minutes boxes to set a timeout between 5 and 480 minutes (8 hours).

For Compute, choose one of the available options.

For Environment variables, use Name and Value to specify any optional environment variables for the build environment to use. To add more environment variables, choose Add environment variable.

**Important**
We strongly discourage storing sensitive values, especially AWS access key IDs and secret access keys, in environment variables. Environment variables can be displayed in plain text in the CodeBuild console and AWS CLI.
To store and retrieve sensitive values, we recommend your build commands use the AWS CLI to interact with the Amazon EC2 Systems Manager Parameter Store. The AWS CLI is already installed and configured on all build environments provided by CodeBuild. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store CLI Walkthrough in the Amazon EC2 Systems Manager User Guide.
20. For **Buildspec**, do one of the following:

- If your source code includes a buildspec file, choose **Use a buildspec file**.
- If your source code does not include a buildspec file, choose **Insert build commands**. For **Build commands**, enter the commands you want to run during the build phase in the build environment. For multiple commands, separate each command with `&&` for Linux-based build environments or `;` for Windows-based build environments. For **Output files**, enter the paths to the build output files in the build environment that you want to send to CodePipeline. For multiple files, separate each file path with a comma.

21. Choose **Create build project**.

22. Return to the CodePipeline console.

23. For **Input artifacts**, select the value for **Output artifact** that you noted earlier in this procedure.

24. (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.

25. Choose **Save**.

26. Choose **Release change**.

27. 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.

28. On the build project page, in **Build history**, choose the **Build run** hyperlink.

29. On the build run page, in **Build logs**, choose the **View entire log** hyperlink to open the build log in the Amazon CloudWatch console.

30. Scroll through the build log to view the test results.
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.

Setting up Jenkins

For information about setting up Jenkins with the AWS CodeBuild plugin, see the Simplify Your Jenkins Builds with CodeBuild blog post on the AWS DevOps Blog. You can download the CodeBuild Jenkins plugin from https://github.com/awslabs/aws-codebuild-jenkins-plugin.

Installing 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.

Using 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) (p. 216).
   - 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 IAM console, create an IAM 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.

```json
{
   "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}}"]
      }
   ]
}
```
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:

```python
awsCodeBuild(projectName: 'project', credentialsType: 'keys', region: 'us-west-2', sourceControlType: 'jenkins')
```
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, and Bitbucket. If your build project uses GitHub Enterprise, 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

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) (p. 253).
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.
2. Enter the following commands in your buildspec file. For more information, see Buildspec syntax (p. 150).

```
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 (p. 150).

```
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 a build in AWS CodeBuild (p. 269) and Logging AWS CodeBuild API calls with AWS CloudTrail (p. 327). Codecov information in the build logs looks like the following:

```
[Container] 2020/03/09 16:31:04 Running command bash my_script.sh

Bash-20200309-bc4d7e6
```

API Version 2016-10-06
213
AWS CodeBuild detected.

... The full list of Codecov log entries has been omitted for brevity ...

- View reports at
  https://codecov.io/github/user/test_py/commit/commit-id

[Container] 2020/03/09 16:31:07 Phase complete: POST_BUILD State: SUCCEEDED

The reports look like the following:

```
<table>
<thead>
<tr>
<th>Files</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>code.py</td>
<td>10</td>
</tr>
<tr>
<td>tests.py</td>
<td>11</td>
</tr>
<tr>
<td><strong>Project Totals</strong></td>
<td>21</td>
</tr>
</tbody>
</table>
```

API Version 2016-10-06
Working with build projects and builds in AWS CodeBuild

To get started, follow the steps in Create a build project (p. 215), and then follow the steps in Run a build (p. 269). For more information about build projects and builds, see the following topics.

Topics
- Working with build projects (p. 215)
- Working with builds in AWS CodeBuild (p. 269)

Working with 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 (p. 215)
- Create a notification rule (p. 240)
- View a list of build project names in AWS CodeBuild (p. 242)
- View a build project's details in AWS CodeBuild (p. 244)
- Build caching in AWS CodeBuild (p. 245)
- Create AWS CodeBuild triggers (p. 249)
- Edit AWS CodeBuild triggers (p. 251)
- Change a build project's settings in AWS CodeBuild (p. 252)
- Delete a build project in AWS CodeBuild (p. 265)
- Working with shared projects (p. 266)

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
- Prerequisites (p. 216)
- Create a build project (console) (p. 216)
- Create a build project (AWS CLI) (p. 229)
- Create a build project (AWS SDKs) (p. 240)
- Create a build project (AWS CloudFormation) (p. 240)
Prerequisites

Answer the questions in Plan a build (p. 148).

Create a build project (console)

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:

   On the Create build project page, in Project configuration, 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.

   In Description, enter an optional description for your project.

   Select Build badge to make your project's build status visible and embeddable. For more information, see Build badges sample (p. 82).

   **Note**
   Build badge does not apply if your source provider is Amazon S3.

   Expand Additional configuration.

   (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.

5. In Source:

   For Source provider, choose the source code provider type. Use the following table to make selections appropriate for your source provider:

   **Note**
   CodeBuild does not support Bitbucket Server.

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Bucket, choose the name of the input bucket that contains the source code.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For 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</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Create a build project

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>a forward slash (/) to download everything in the S3 bucket.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choose <strong>Connect using OAuth</strong> or <strong>Connect with a Bitbucket app password</strong> and follow the instructions to connect (or reconnect) to Bitbucket.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Choose <strong>Connect using OAuth</strong> or <strong>Connect with a GitHub personal access token</strong> and follow the instructions to connect (or reconnect) to GitHub and authorize access to AWS CodeBuild.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Create a build project

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Personal Access token, see GitHub Enterprise sample (p. 115) for information about how to copy a personal access token to your clipboard. Paste the token in the text field, and then choose Save Token. <strong>Note</strong> You only need to enter and save the personal access token once. CodeBuild uses this token in all future projects.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From Repository, choose the repository you want to use.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Create a build project

For **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](p. 139).

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Repository, choose a public repository or a repository in your account.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use <strong>Repository URL</strong> only if you use a public repository. The URL must contain the source provider's name. For example, a Bitbucket URL must contain bitbucket.org.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Create a build project

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon S3</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CodeCommit</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bitbucket</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GitHub</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GitHub Enterprise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

If your source provider is Amazon S3, for **Source version**, enter the version ID of the object that represents the build of your input file. If your source provider is GitHub or GitHub Enterprise, enter a pull request, branch, commit ID, tag, or reference and a commit ID. If your source provider is Bitbucket, enter a branch, commit ID, tag, or reference and a commit ID. For more information, see [Source version sample with AWS CodeBuild](p. 139).

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**.

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Create a build project

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Use Git submodules if you want to include Git submodules in your repository.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

The status of a build triggered by a webhook is always reported to your source provider.
<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Rebuild every time a code change is pushed to this repository</strong> if you want CodeBuild to build the source code every time a code change is pushed to this repository. Webhooks are allowed only with your own Bitbucket, GitHub, or GitHub Enterprise repository.</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

API Version 2016-10-06

222
### Create a build project

If you chose **Rebuild every time a code change is pushed to this repository**, in **Event type**, choose an event that you want to trigger a build. You use regular expressions to create a filter. If no filter is specified, all update and create pull requests, and all push events, trigger a build. For more information, see [Filter GitHub webhook events](https://docs.aws.amazon.com/codebuild/latest/userguide/filtering-event-types.html) and [Filter Bitbucket webhook events](https://docs.aws.amazon.com/codebuild/latest/userguide/filtering-event-types.html).

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you chose <strong>Rebuild every time a code change is pushed to this repository</strong>, in <strong>Event type</strong>, choose an event that you want to trigger a build. You use regular expressions to create a filter. If no filter is specified, all update and create pull requests, and all push events, trigger a build. For more information, see <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/filtering-event-types.html">Filter GitHub webhook events</a> and <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/filtering-event-types.html">Filter Bitbucket webhook events</a>.</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Choose <strong>Insecure SSL</strong> to ignore SSL warnings while connecting to your GitHub Enterprise project repository.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

For each secondary source you want:

a. Choose **Add source**.

b. For **Source identifier**, enter a value that is fewer than 128 characters and contains only alphanumeric characters and underscores.

c. For **Source provider**, choose the source code provider type. Use the table earlier in this step to make selections appropriate for your secondary source provider.
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 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*.

   (Optional) Select **Privileged** only if you plan to use this build project to build Docker images, and the build environment image you chose is not provided by CodeBuild with Docker support. 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 containers do not allow access to any devices. Privileged mode grants a build project’s Docker container access to all devices. For more information, see **Runtime Privilege and Linux Capabilities** on the Docker Docs website.

   ```bash
   - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
     host=tcp://127.0.0.1:2375 --storage-driver=overlay&
   - timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
   ```

   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.

   Expand **Additional configuration**.

   (Optional) For **Timeout**, specify a value between 5 minutes and 480 minutes (8 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.

   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 (p. 178).

If you want to use one or more elastic file systems:

- 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 `efs-1`, the environment variable is `CODEBUILD_EFS-1`.

- 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 name of a directory in your build container that mounts the file system. 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.

For **Compute**, choose one of the available options.

For **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`. 
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.

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.

7. In **Buildspec**:

   For **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** (p. 149).

8. In **Artifacts**:

   For **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** (p. 150).

   - 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** (p. 150).

   - If you do not want your build artifacts encrypted, select **Remove artifacts encryption**.

   For each secondary set of artifacts you want:

   a. For **Artifact identifier**, enter a value that is fewer than 128 characters and contains only alphanumeric characters and underscores.
b. Choose Add artifact.

c. Follow the previous steps to configure your secondary artifacts.

d. Choose Save artifact.

Expand Additional configuration.

(Optional) For Encryption key, do one of the following:

- To use the AWS-managed customer managed key (CMK) 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 CMK to encrypt the build output artifacts, in Encryption key, enter the ARN of the CMK. Use the format arn:aws:kms:region-ID:account-ID:key/key-ID.

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. The ARM_CONTAINER and LINUX_GPU_CONTAINER environment types and the BUILD_GENERAL1_2XLARGE compute type do not support the use of a local cache.

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 (p. 150). For more information about caching, see Build caching in AWS CodeBuild (p. 245).

9. In Logs, choose the logs you want to create. You can create Amazon CloudWatch Logs, Amazon S3 logs, or both.

   If you want Amazon CloudWatch Logs logs:
   - Select CloudWatch logs.
   - In Group name, enter the name of your Amazon CloudWatch Logs log group.
   - In Stream name, enter your Amazon CloudWatch Logs log stream name.

   If you want Amazon S3 logs:
   - Select S3 logs.
   - From Bucket, choose the name of the S3 bucket for your logs.
   - In Path prefix, enter the prefix for your logs.
Create a build project

(Optional) If you chose Amazon S3 for Type in Artifacts earlier in this procedure, then for Artifacts packaging, do one of the following:

- To have CodeBuild create a ZIP file that contains the build output, choose Zip.
- To have CodeBuild create a folder that contains the build output, choose None. (This is the default.)
- Select Remove S3 log encryption if you do not want your S3 logs encrypted.

10. Choose Create build project.

Create a build project (AWS CLI)

For information about using the AWS CLI with CodeBuild, see the Command line reference (p. 355).

1. Run the create-project command:

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

JSON-formatted data appears in the output. Copy the data to a file (for example, create-project.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.

```json
{
    "name": "project-name",
    "description": "description",
    "source": {
        "type": "source-type",
        "location": "source-location",
        "gitCloneDepth": "gitCloneDepth",
        "buildspec": "buildspec",
        "InsecureSsl": "InsecureSsl",
        "reportBuildStatus": "reportBuildStatus",
        "gitSubmodulesConfig": {
            "fetchSubmodules": "fetchSubmodules"
        },
        "auth": {
            "type": "auth-type",
            "resource": "resource"
        }
    },
    "sourceVersion": "source-version",
    "secondarySourceVersions": {
        "sourceIdentifier": "secondary-source-identifier",
        "sourceVersion": "secondary-source-version"
    },
    "artifacts": {
        "type": "artifacts-type",
        "location": "artifacts-location",
        "path": "path",
        "namespaceType": "namespaceType",
        "name": "artifacts-name",
        "overrideArtifactName": "override-artifact-name",
        "packaging": "packaging"
    },
    "cache": {
        "type": "cache-type",
        "location": "cache-location",
```

API Version 2016-10-06
229
"mode": ["cache-mode"],
"logsConfig": {
  "cloudWatchLogs": {
    "status": "cloudwatch-logs-status",
    "groupName": "group-name",
    "streamName": "stream-name"
  },
  "s3Logs": {
    "status": "s3-logs-status",
    "location": "s3-logs-location",
    "encryptionDisabled": "s3-logs-encryptionDisabled"
  }
},
"secondaryArtifacts": [
  {
    "type": "artifacts-type",
    "location": "artifacts-location",
    "path": "path",
    "namespaceType": "namespaceType",
    "name": "artifacts-name",
    "packaging": "packaging",
    "artifactIdentifier": "artifact-identifier"
  }
],
"secondarySources": [
  {
    "type": "source-type",
    "location": "source-location",
    "gitCloneDepth": "gitCloneDepth",
    "buildspec": "buildspec",
    "InsecureSsl": "InsecureSsl",
    "reportBuildStatus": "reportBuildStatus",
    "auth": {
      "type": "auth-type",
      "resource": "resource"
    },
    "sourceIdentifier": "source-identifier"
  }
],
"serviceRole": "serviceRole",
"vpcConfig": {
  "securityGroupIds": [
    "security-group-id"
  ],
  "subnets": [
    "subnet-id"
  ],
  "vpcId": "vpc-id"
},
"fileSystemLocations": [
  {
    "type": "EFS",
    "location": "EFS-DNS-name-1:/directory-path",
    "mountPoint": "mount-point",
    "identifier": "efs-identifier",
    "mountOptions": "efs-mount-options"
  },
  {
    "type": "EFS",
    "location": "EFS-DNS-name-2:/directory-path",
    "mountPoint": "mount-point",
    "identifier": "efs-identifier",
Replace the following:

- **project-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.

For the required **source** object, 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 CodeBuild **secondarySources** object. 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, BITBUCKET, S3, and 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 Project without a source sample (p. 145).

- **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 a forward slash (/), followed by the name of the ZIP file that contains the source code and the buildspec (for example, bucket-name/object-name.zip). This assumes that the ZIP file is in the root of
Create a build project

1. When you use the console to connect (or reconnect) with GitHub, on the GitHub Authorize application page, for Organization access, choose Request access next to each repository you want CodeBuild to be able to access.

2. 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, 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 account. To do this, use the CodeBuild console to create a build project.

1. Create a personal access token in GitHub Enterprise.

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 in GitHub Enterprise 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 account.

• 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.

• 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.

• buildspec: Optional. The build specification definition or file to use. 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 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 (p. 149).

• auth: This object is used by the CodeBuild console only. Do not specify values for auth-type (unless source-type is set to GITHUB) or resource.

• reportBuildStatus: Optional. 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, or Bitbucket, an invalidInputException is thrown.

• gitSubmodulesConfig: Optional. Information about the Git submodules configuration. Used with CodeCommit, GitHub, GitHub Enterprise, and Bitbucket only. 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.

- **InsecureSsl**: Optional. Used with GitHub Enterprise only. Set this value to `true` to ignore
  TSL warnings while connecting to your GitHub Enterprise 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-version**: Optional. A 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 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 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** (p. 139).

- **secondarySourceVersions**: Optional. An array of `projectSourceVersion` objects. If
  `secondarySourceVersions` is specified at the build level, then they take precedence over this.
  - **secondary-source-identifier**: An identifier for a source in the build project.
  - **secondary-source-version**: A `sourceVersion` object.

For the required `artifacts` object, 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 `CodeBuild
secondaryArtifacts` object. These settings include the following:

- **artifacts-type**: Required. The type of build output artifact. Valid values include
  `CODEPIPELINE`, `NO_ARTIFACTS`, and `S3`.

- **artifacts-location**: Required unless you set `artifacts-type` to `CODEPIPELINE` or `NO_ARTIFACTS`. The location of the build output artifact:
  - If you specified `CODEPIPELINE` for `artifacts-type`, do not specify a location for artifacts.
  - If you specified `NO_ARTIFACTS` for `artifacts-type`, do not specify a location for artifacts.
  - If you specified `S3` for `artifacts-type`, this is the name of the output bucket you created
    or identified in the prerequisites.

- **path**: Optional. The path and name of the build output ZIP file or folder:
  - If you specified `CODEPIPELINE` for `artifacts-type`, do not specify a path for artifacts.
  - If you specified `NO_ARTIFACTS` for `artifacts-type`, do not specify a path for artifacts.
  - If you specified `NO_ARTIFACTS` for `artifacts-type`, do not specify a path for artifacts.
  - If you specified `S3` for `artifacts-type`, this is the path inside of `artifacts-location`
to the build output ZIP file or folder. If you do not specify a value for `path`, CodeBuild
  uses `namespaceType` (if specified) and `artifacts-name` to determine the path and
  name of the build output ZIP file or folder. For example, if you specify `MyPath` for `path`
Create a build project

and MyArtifact.zip for artifacts-name, the path and name would be MyPath/MyArtifact.zip.

- namespaceType: Optional. The path and name of the build output ZIP file or folder:
  - If you specified CODEPIPELINE for artifacts-type, do not specify a namespaceType for artifacts.
  - If you specified NO_ARTIFACTS for artifacts-type, do not specify a namespaceType for artifacts.
  - If you specified S3 for artifacts-type, 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 artifacts-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 artifacts-name, the path and name would be MyPath/build-ID/MyArtifact.zip.

- artifacts-name: Required unless you set artifacts-type to CODEPIPELINE or NO_ARTIFACTS. The path and name of the build output ZIP file or folder:
  - If you specified CODEPIPELINE for artifacts-type, do not specify a name for artifacts.
  - If you specified NO_ARTIFACTS for artifacts-type, do not specify a name for artifacts.
  - If you specified S3 for artifacts-type, this is the name of the build output ZIP file or folder inside of artifacts-location. For example, if you specify MyPath for path and MyArtifact.zip for artifacts-name, the path and name would be MyPath/MyArtifact.zip.

- override-artifact-name: Optional boolean. If set to true, the name specified in the artifacts block of the buildspec file overrides artifacts-name. For more information, see Build specification reference for CodeBuild (p. 149).

- packaging: Optional. The type of build output artifact to create:
  - If you specified CODEPIPELINE for artifacts-type, do not specify a packaging for artifacts.
  - If you specified NO_ARTIFACTS for artifacts-type, do not specify a packaging for artifacts.
  - If you specified S3 for artifacts-type, valid values include ZIP and NONE. To create a ZIP file that contains the build output, use ZIP. To create a folder that contains the build output, use NONE. The default value is NONE.

- For the required cache object, information about this build project's cache settings. For information, see Build caching (p. 245). These settings include the following.

  - cache-type: Required. Valid values are S3, NO_CACHE, or LOCAL_CACHE.

  - cache-location: Required only if you set CacheType to S3. If you specified Amazon S3 for CacheType, this is the ARN of the S3 bucket and the path prefix. For example, if your S3 bucket name is my-bucket, and your path prefix is build-cache, then acceptable formats for your CacheLocation are my-bucket/build-cache or arn:aws:s3:::my-bucket/build-cache.

  - cache-mode: Required if you set CacheType to LOCAL. You can specify one or more of the following local cache modes: LOCAL_SOURCE_CACHE, LOCAL_DOCKER_LAYER_CACHE, LOCAL_CUSTOM_CACHE.

  Note
  Docker layer cache mode is available for Linux only. If you choose it, your project must run in privileged mode. The ARM_CONTAINER and LINUX_GPU_CONTAINER environment types and the BUILD_GENERAL1_2XLARGE compute type do not support the use of a local cache.

API Version 2016-10-06

234
• For the `logsConfig` object, information about where this build's logs are located:
  • `cloudwatch-logs-status`: Required. Valid values are ENABLED or DISABLED. If its value is ENABLED, the following values are required. For more information, see Working with Log Groups and Log Streams in the Amazon CloudWatch Logs User Guide.
  • `group-name`: The name of the CloudWatch Logs group.
  • `stream-name`: The name of the CloudWatch Logs stream.
  • `s3-logs-status`: Required. Valid values are ENABLED or DISABLED.
  • `s3-logs-location`: Required if `s3-logs-status` is ENABLED. This is the ARN of an S3 bucket and the path prefix. For example, if your S3 bucket name is `my-bucket`, and your path prefix is `build-log`, then acceptable formats for your `s3-logs-location` are `my-bucket/build-log` or `arn:aws:s3:::my-bucket/build-log`.
  • `s3-logs-encryptionDisabled`: Optional boolean. If set to true, your S3 build log output is not encrypted. By default, S3 build logs are encrypted.

• For the optional `secondaryArtifacts` object, information about the settings of a secondary artifact for a build project. You can add up to 12 secondary artifacts. The `secondaryArtifacts` uses many of the same settings used by the `CodeBuild artifacts object` object. The settings are:
  • `artifacts-type`: Required. This setting is also used by the `artifacts object`. See `CodeBuild artifact object's type property`.
  • `artifacts-location`: Required. This setting is also used by the `artifacts object`. See `CodeBuild artifact object's location property`.
  • `path`: Optional. This setting is also used by the `artifacts object`. See `CodeBuild artifact object's path property`.
  • `namespaceType`: Optional. This setting is also used by the `artifacts object`. See `CodeBuild artifact object's namespaceType property`.
  • `artifacts-name`: Required. This setting is also used by the `artifacts object`. See `CodeBuild artifact object's name property`.
  • `packaging`: Optional. This setting is also used by the `artifacts object`. See `CodeBuild artifact object's packaging property`.
  • `artifact-identifier`: Required. A unique string identifier for a secondary artifact.

• For the optional `secondarySources` object, information about the settings of a secondary source for a build project. You can add up to 12 `secondarySources`. The `secondarySources` object uses many of the same settings used by the `CodeBuild source object`. They include the following:
  • `source-type`: Required. This setting is also used by the `sources object`. See `CodeBuild source object's type property`.
  • `source-location`: Required. This setting is also used by the `sources object`. See `CodeBuild source object's location property`.
  • `gitCloneDepth`: Optional. This setting is also used by the `sources object`. See `CodeBuild source object's location property`.
  • `buildspec`: Optional. This setting is also used by the `sources object`. See `CodeBuild source object's buildspec property`.
  • `auth`: This setting is also used by the `sources object`. See `CodeBuild source object's auth property`.
  • `reportBuildStatus`: Optional. This setting is also used by the `sources object`. See `CodeBuild source object's reportBuildStatus property`.
  • `InsecureSsl`: Optional. This setting is also used by the `sources object`. See `CodeBuild source object's insecureSsl property`.
  • `source-identifier`: Required. A unique string identifier for a secondary source.
• **serviceRole**: Required. The ARN of the service role CodeBuild uses to interact with services on behalf of the IAM user (for example, `arn:aws:iam::account-id:role/role-name`).

• For the optional **vpcConfig** object, information about your VPC configuration. These settings include:
  • **vpcId**: Required. The VPC ID that CodeBuild uses. Run this command to get a list of all VPC IDs in your Region:

    ```bash
    aws ec2 describe-vpcs
    ```

  • **subnets**: Required. The subnet IDs that include resources used by CodeBuild. Run this command to get these IDs:

    ```bash
    aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
    ```

  If you are using a Region other than `us-east-1`, be sure to use it when you run the command.

  • **securityGroupIds**: Required. The security group IDs used by CodeBuild to allow access to resources in the VPCs. Run this command to get these IDs:

    ```bash
    aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
    ```

  If you are using a Region other than `us-east-1`, be sure to use it when you run the command.

• For the optional **fileSystemLocations** object, information about your Amazon EFS configuration. These settings include:
  • **type**: Required. This value must be EFS.

  • **location**: Required. The location specified in the format `EFS-DNS-name:/directory-path`.

  • **mountPoint**: Required. The name of a directory in your build container that mounts the file system. If this directory does not exist, CodeBuild creates it during the build.

  • **identifier**: Required. A unique file system identifier. CodeBuild uses this to create an environment variable that identifies the file system. The environment variable format is `CODEBUILD_file-system-identifier` in capital letters. For example, if you enter `efs-1`, the resulting environment variable is `CODEBUILD_EFS-1`.

  • **mountOptions**: Optional. If you leave this 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.

• For the required **environment** object, information about this project's build environment settings. These settings include:
  • **environment-type**: Required. The type of build environment. Valid values are `ARM_CONTAINER`, `LINUX_CONTAINER`, `LINUX_GPU_CONTAINER`, and `WINDOWS_CONTAINER`.

  • **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:2.0`. 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 Recommended Docker Images provided by CodeBuild (p. 166).

  • **computeType**: Required. A category that corresponds to the number of CPU cores and memory used by this build environment. Allowed values include `BUILD_GENERAL1_SMALL`, `BUILD_GENERAL1_MEDIUM`, `BUILD_GENERAL1_LARGE`, and `BUILD_GENERAL1_2XLARGE`. `BUILD_GENERAL1_2XLARGE` is only supported with the `LINUX_CONTAINER` environment type.

  • **certificate**: Optional. The ARN of the 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 S3 bucket name is `my-bucket`, your path prefix is `cert`, and your object key name is `certificate.pem`, then acceptable formats for your certificate are `my-bucket/cert/certificate.pem` or `arn:aws:s3:::my-bucket/cert/certificate.pem`.

- For the optional `environmentVariables` array, information about any 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 `environmentVariable-name`, `environmentVariable-value`, and `environmentVariable-type`.

Console and AWS CLI users can see an environment variable. If you have no concerns about the visibility of your environment variable, set `environmentVariable-name` and `environmentVariable-value`, and then set `environmentVariable-type` to `PLAINTEXT`.

We recommend 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. For `environmentVariable-name`, for that stored parameter, set an identifier for CodeBuild to reference.

If you use Amazon EC2 Systems Manager Parameter Store, for `environmentVariable-value`, set the parameter's name as stored in the Parameter Store. Set `environmentVariable-type` to `PARAMETER_STORE`. Using a parameter named `/CodeBuild/dockerLoginPassword` as an example, set `environmentVariable-name` to `LOGIN_PASSWORD`. Set `environmentVariable-value` to `/CodeBuild/dockerLoginPassword`. Set `environmentVariable-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](https://docs.aws.amazon.com/systemsmanager/latest/userguide/systems-manager-parameter-store.html) and [Systems Manager Parameter Store Console Walkthrough](https://docs.aws.amazon.com/systemsmanager/latest/userguide/systems-manager-parameter-store-console-walkthrough.html) 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 `environmentVariable-value`, set the parameter's name as stored in Secrets Manager. Set `environmentVariable-type` to `SECRETS_MANAGER`. Using a secret named `/CodeBuild/dockerLoginPassword` as an example, set `environmentVariable-name` to `LOGIN_PASSWORD`. Set `environmentVariable-value` to `/CodeBuild/dockerLoginPassword`. Set `environmentVariable-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.

- Use the optional `registryCredential` to specify information about credentials that provide access to a private Docker registry.
  - `credential-arn-or-name`: 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.
  - `credential-provider`: The only valid value is `SECRETS_MANAGER`.

When this is set:

- `imagePullCredentials` must be set to `SERVICE_ROLE`.
- Images cannot be curated or an Amazon ECR image.

- `imagePullCredentialsType-value`: Optional. The type of credentials CodeBuild uses to pull images in your build. There are two valid values:
  - `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.

- You must specify `privilegedMode` with a value of `true` only if you plan to use this build project to build Docker images, and the build environment image you specified is not provided by CodeBuild with Docker support. 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 containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see [Runtime Privilege and Linux Capabilities](#) on the Docker Docs website.

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcps://127.0.0.1:2375 --storage-driver=overlay&
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

- **badgeEnabled**: Optional. To include build badges with your CodeBuild project, you must specify `badgeEnabled` with a value of `true`. For more information, see [Build badges sample with CodeBuild](#).

- **timeoutInMinutes**: Optional. The number of minutes, between 5 to 480 (8 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`.

- **encryptionKey**: Optional. The alias or ARN of the AWS KMS customer managed key (CMK) 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 CMK for Amazon S3 is used.

- For the optional `tags` array, information about any 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 that contains a `key` and `value` of `tag-key` and `tag-value`.

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

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

3. If successful, data similar to the following appears in the output:

```
{
  "project": {
    "name": "project-name",
    "description": "description",
    "serviceRole": "serviceRole",
    "tags": [
      {
        "key": "tags-key",
        "value": "tags-value"
      }
    ],
    "artifacts": {
      "namespaceType": "namespaceType",
      "packaging": "packaging",
      "path": "path",
      "type": "artifacts-type",
      "location": "artifacts-location",
      "name": "artifacts-name"
    },
    "lastModified": lastModified,
    "timeoutInMinutes": timeoutInMinutes,
    "created": created
  }
}
```
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.
notifications. For more information, see What Are Notifications? in the Developer Tools Console User Guide.

You can use the console or the AWS CLI to create notification rules for AWS CodeBuild.

To create a notification rule (console)

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 in the Developer Tools Console User Guide.
6. In Events that trigger notifications, select the events for which you want to send notifications.

<table>
<thead>
<tr>
<th>Category</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build state</td>
<td>Failed</td>
</tr>
<tr>
<td></td>
<td>Succeeded</td>
</tr>
<tr>
<td></td>
<td>In-progress</td>
</tr>
<tr>
<td></td>
<td>Stopped</td>
</tr>
<tr>
<td>Build phase</td>
<td>Failure</td>
</tr>
<tr>
<td></td>
<td>Success</td>
</tr>
</tbody>
</table>
7. In Targets, choose Create SNS topic.

   **Note**
   When you create the topic, the policy that allows CodeBuild to publish events to the topic is applied for you. Using a topic created specifically for CodeBuild notifications also helps ensure that you only add users to the subscription list for that topic that you want to see notifications about this build project.

   After the codestar-notifications- prefix, enter a name for the topic, and then choose Submit.

   **Note**
   If you want to use an existing Amazon SNS topic instead of creating a new one, in Targets, choose its ARN. Make sure the topic has the appropriate access policy and that the subscriber list contains only those users who are allowed to see information about the build project. For more information, see Configure Amazon SNS Topics for Notifications and Understanding Notification Contents and Security in the Developer Tools Console User Guide.
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 in the Developer Tools Console User Guide. You can also set up integration between notifications and AWS Chatbot to send notifications to Amazon Chime chatrooms or Slack channels. For more information, see
Configure Integration Between Notifications and AWS Chatbot in the Developer Tools Console User Guide.

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:

   ```bash
   aws codestar-notifications 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 Amazon SNS target you want for the rule. The following example shows a notification rule named `MyNotificationRule` for a build project named `MyBuildProject` in an AWS account 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:

   ```json
   {
     "Name": "MyNotificationRule",
     "EventTypeIds": ["codebuild-project-build-state-succeeded"],
     "Targets": [
       {
         "TargetType": "SNS",
       }
     ],
     "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:

   ```bash
   aws codestar-notifications 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:

   ```json
   {
     "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/dc82df7a-EXAMPLE"
   }
   ```

View a list of 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) (p. 243)
- View a list of build project names (AWS CLI) (p. 243)
View a list of build project names (AWS SDKs) (p. 244)

View a list of build project names (console)

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:

```bash
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:

```bash
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

A result similar to the following might appear in the output:

```json
{
  "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:
View a build project's details

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) (p. 244)
- View a build project's details (AWS CLI) (p. 244)
- View a build project's details (AWS SDKs) (p. 245)

### View a build project's details (console)

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 select a different value for **Projects per page**.
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) (p. 243).

For example, if you run this command:

```bash
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.

```json
{
  "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) (p. 229).

For more information about using the AWS CLI with AWS CodeBuild, see the Command line reference (p. 355).

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 (p. 356).

Build caching in AWS CodeBuild

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 (p. 246)
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 intermediate build artifacts that are more expensive to build than to download. This is not the best option for large build artifacts because they can take a long time to transfer over your network, which can affect build performance.

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 large intermediate build artifacts because the cache is immediately available on the build host. 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 (GitHub, GitHub Enterprise, 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.

Note

By default, Docker containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.

- 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 are overridden if a source item has the same name. Directories are specified using cache paths in the buildspec file. For more information, see Buildspec syntax (p. 150).

Note

The ARM_CONTAINER and LINUX_GPU_CONTAINER environment types and the BUILD_GENERAL1_2XLARGE compute type do not support the use of a local cache. For more information, see Build environment compute types (p. 170).

Topics

- Specify local caching (CLI) (p. 247)
- Specify local caching (console) (p. 247)
- Specify local caching (AWS CloudFormation) (p. 249)
You can use the AWS CLI, console, SDK, or AWS CloudFormation to specify a local cache.

**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:
  
  ```bash
  --cache type=LOCAL,mode=[LOCAL_SOURCE_CACHE]
  ```

- To specify a Docker layer cache:
  
  ```bash
  --cache type=LOCAL,mode=[LOCAL_DOCKER_LAYER_CACHE]
  ```

- To specify a custom cache:
  
  ```bash
  --cache type=LOCAL,mode=[LOCAL_CUSTOM_CACHE]
  ```

For more information, see [Create a build project (AWS CLI) (p. 229)](https://docs.aws.amazon.com/codebuild/latest/userguide/create-project.html).

**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.
Cache type

- Local

Select one or more local caches:

- Docker layer cache
  Caches existing Docker layers

- Source cache
  Caches .git metadata sources

- Custom cache
  Caches directories specified
For more information, see Create a build project (console) (p. 216).

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: myBucket
      Name: myArtifact
      EncryptionDisabled: true
      OverrideArtifactName: true
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:2.0
      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 containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.

For more information, see Create a build project (AWS CloudFormation) (p. 240).

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.

You create a trigger after you create a project.

To create a trigger

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.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Required Parameters</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hourly</td>
<td>Start minute</td>
<td>Use the Start minute drop-down menu.</td>
</tr>
<tr>
<td>Daily</td>
<td>Start minute</td>
<td>Use the Start minute drop-down menu.</td>
</tr>
<tr>
<td></td>
<td>Start hour</td>
<td>Use the Start hour drop-down menu.</td>
</tr>
<tr>
<td>Weekly</td>
<td>Start minute</td>
<td>Use the Start minute drop-down menu.</td>
</tr>
<tr>
<td></td>
<td>Start hour</td>
<td>Use the Start hour drop-down menu.</td>
</tr>
<tr>
<td></td>
<td>Start day</td>
<td>Use the Start day drop-down menu.</td>
</tr>
<tr>
<td>Custom</td>
<td>Cron expression</td>
<td>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.</td>
</tr>
</tbody>
</table>

API Version 2016-10-06

250
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 480 minutes (8 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**.

---

**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` (p. 249).

**To edit a trigger**

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.
### Change a build project's settings

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 [Working with test report permissions](p. 298).

**Topics**
- Change a build project's settings (console) (p. 253)
- Change a build project's settings (AWS CLI) (p. 264)
- Change a build project's settings (AWS SDKs) (p. 265)

---

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Required Parameters</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hourly</td>
<td>Start minute</td>
<td>Use the <strong>Start minute</strong> drop-down menu.</td>
</tr>
<tr>
<td>Daily</td>
<td>Start minute, Start hour</td>
<td>Use the <strong>Start minute</strong> drop-down menu. Use the <strong>Start hour</strong> drop-down menu.</td>
</tr>
<tr>
<td>Weekly</td>
<td>Start minute, Start hour, Start day</td>
<td>Use the <strong>Start minute</strong> drop-down menu. Use the <strong>Start hour</strong> drop-down menu. Use the <strong>Start day</strong> drop-down menu.</td>
</tr>
<tr>
<td>Custom</td>
<td>Cron expression</td>
<td>Enter a cron expression in <strong>Cron expression</strong>. 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 <code>0 9 ? * MON-FRI *</code> schedules a build every weekday at 9:00 A.M. For more information, see Cron Expressions in the Amazon CloudWatch Events User Guide.</td>
</tr>
</tbody>
</table>

7. Select **Enable this trigger**.

**Note**
You can use the Amazon CloudWatch console at [https://console.aws.amazon.com/cloudwatch/](https://console.aws.amazon.com/cloudwatch/) to edit source version, timeout, and other options that are not available in AWS CodeBuild.
Change a build project's settings (console)

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.
4. To change the project's description, in Project configuration, choose Edit, and then enter a description.
   
   Choose Update configuration.

   For more information about settings referred to in this procedure, see Create a build project (console) (p. 216).

5. To change information about the source code location, in Source, choose Edit. Use the following table to make selections appropriate for your source provider, and then choose Update source.

   Note
   CodeBuild does not support Bitbucket Server.

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Bucket, choose the name of the input bucket that contains the source code.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For 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.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choose Connect using OAuth or Connect with a Bitbucket app password and follow the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Provider</td>
<td>Amazon S3</td>
<td>CodeCommit</td>
<td>Bitbucket</td>
<td>GitHub</td>
<td>GitHub Enterprise</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>instructions to connect (or reconnect) to Bitbucket.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choose <strong>Connect using OAuth</strong> or <strong>Connect with a GitHub personal access token</strong> and follow the instructions to connect (or reconnect) to GitHub and authorize access to AWS CodeBuild.</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Source Provider</td>
<td>Amazon S3</td>
<td>CodeCommit</td>
<td>Bitbucket</td>
<td>GitHub</td>
<td>GitHub Enterprise</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-------------</td>
<td>-----------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>For <strong>Personal Access token</strong>, see GitHub Enterprise sample (p. 115) for information about how to copy a personal access token to your clipboard. Paste the token in the text field, and then choose <strong>Save Token</strong>. <strong>Note</strong> You only need to enter and save the personal access token once. CodeBuild uses this token in all future projects.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>From <strong>Repository</strong>, choose the repository you want to use.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Provider</td>
<td>Amazon S3</td>
<td>CodeCommit</td>
<td>Bitbucket</td>
<td>GitHub</td>
<td>GitHub Enterprise</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>For 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 (p. 139).</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For Repository, choose a public repository or a repository in your account.</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use Repository URL only if you use a public repository. The URL must contain the source provider's name. For example, a Bitbucket URL must contain bitbucket.org.</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

API Version 2016-10-06
256
### Change a build project’s settings

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>If your source provider is Amazon S3, for <strong>Source version</strong>, enter the version ID of the object that represents the build of your input file. If your source provider is GitHub or GitHub Enterprise, enter a pull request, branch, commit ID, tag, or reference and a commit ID. If your source provider is Bitbucket, enter a branch, commit ID, tag, or reference and a commit ID. For more information, see [Source version sample with AWS CodeBuild](p. 139).</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Choose <strong>Git clone depth</strong> to create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose <strong>Full</strong>.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
## Source Provider

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Use Git submodules</strong> if you want to include Git submodules in your repository.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Select <strong>Report build statuses to source provider when your builds start and finish</strong> if you want the status of your build's start and completion reported to your source provider.</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Note**
The status of a build triggered by a webhook is always reported to your source provider.
### Source Provider

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Rebuild every time a code change is pushed to this repository</strong> if you want CodeBuild to build the source code every time a code change is pushed to this repository. Webhooks are allowed only with your own Bitbucket, GitHub, or GitHub Enterprise repository.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Source Provider</td>
<td>Amazon S3</td>
<td>CodeCommit</td>
<td>Bitbucket</td>
<td>GitHub</td>
<td>GitHub Enterprise</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>If you chose <strong>Rebuild every time a code change is pushed to this repository</strong>, in <strong>Event type</strong>, choose an event that you want to trigger a build. You use regular expressions to create a filter. If no filter is specified, all update and create pull requests, and all push events, trigger a build. For more information, see <a href="#">Filter GitHub webhook events</a> (p. 123) and <a href="#">Filter Bitbucket webhook events</a> (p. 77).</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>If you chose <strong>Webhook</strong>, choose <strong>Rotate webhook secret key</strong> if you want GitHub to rotate your secret key every time a code change triggers a build.</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Change a build project's settings

<table>
<thead>
<tr>
<th>Source Provider</th>
<th>Amazon S3</th>
<th>CodeCommit</th>
<th>Bitbucket</th>
<th>GitHub</th>
<th>GitHub Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose Insecure SSL to ignore SSL warnings while connecting to your GitHub Enterprise project repository.</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

To change whether CodeBuild can modify the service role you use for this project, select or clear **Allow AWS CodeBuild to modify this service role so it can be used with this build project**. If you clear it, you must use a service role with CodeBuild permissions attached to it. For more information, see Add CodeBuild access permissions to an IAM group or IAM user (p. 344) and Create a CodeBuild service role (p. 348).

6. To change information about the build environment, in **Environment**, choose **Edit**. Make changes appropriate for the build environment type (for example, **Environment image**, **Operating system**, **Runtime**, **Runtime version**, **Custom image**, **Other location**, **Amazon ECR repository**, or **Amazon ECR image**).

7. If you plan to use this build project to build Docker images and the specified build environment is not provided by CodeBuild with Docker support, select **Privileged**. 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 as needed. You can do this by running the following build commands to initialize the Docker daemon in the **install** phase of your buildspec file. (Do not run the following build commands if the specified build environment image is provided by CodeBuild with Docker support.)

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcp://127.0.0.1:2375 --storage-driver=overlay&
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

8. To change information about the CodeBuild service role, in **Service role**, change the values for **New service role**, **Existing service role**, or **Role name**.

**Note**
By default, Docker containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.

9. To change information about the build timeout, in **Additional configuration**, for **Timeout**, change the values for **hours** and **minutes**. If **hours** and **minutes** are left blank, the default value is 60 minutes.

10. To change information about the VPC you created in Amazon VPC, in **Additional configuration**, change the values for **VPC**, **Subnets**, and **Security groups**.

11. To change information about a file system you created in Amazon EFS, in **Additional configuration**, change its values for **Identifier**, **ID**, **Directory path**, **Mount point**, and **Mount options**. For more information, see Amazon Elastic File System sample for AWS CodeBuild (p. 56).

API Version 2016-10-06
261
12. To change the amount of memory and vCPUs that are used to run builds, in **Additional configuration**, change the value for **Compute**.

13. To change information about environment variables you want builds to use, in **Additional configuration**, for **Environment variables**, change the values for **Name**, **Value**, and **Type**. Use **Add environment variable** to add an environment variable. Choose **Remove** next to an environment variable you no longer want to use.

Others can see environment variables by using the CodeBuild console and the AWS CLI. 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**, type `/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:
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.

14. To change information about tags for this build project, in **Additional configuration**, for **Tags**, change the values of **Name** and **Value**. Use **Add row** to add a tag. You can add up to 50 tags. Choose the delete (X) icon next to a tag you no longer want to use.

15. Choose **Update environment**.

16. To change the project’s build specifications, in **Buildspec**, choose **Edit**. 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 previously did not include a buildspec.yml file but does now, choose **Use a buildspec file**.
- If your source code previously included a buildspec.yml file but does not now, choose **Insert build commands**, and in **Build commands**, enter the commands.

17. Choose **Update buildspec**.

18. To change information about the build output artifact location and name, in **Artifacts**, choose **Edit**, and then change the values for **Type**, **Name**, **Path**, **Namespace type**, or **Bucket name**.

19. To change information about the AWS KMS customer managed key (CMK), in **Additional configuration**, change the value for **Encryption key**.

**Important**
If you leave **Encryption key** blank, CodeBuild uses the AWS-managed CMK for Amazon S3 in your AWS account instead.

20. Using a cache saves 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* (p. 150). To change information about the cache, expand **Additional configuration**. In **Cache type**, do one of the following:

- If you previously chose a cache, but do not want to use one now, choose **No cache**.
• If you previously chose **No cache** but now want to use one, choose **Amazon S3**, and then do the following:
  • For **Cache 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. You use it to store the cache under the same directory in a bucket.

**Important**
Do not append a forward slash (/) to the end of **Path prefix**.

21. To change your log settings, in **Logs**, select or clear **CloudWatch logs** and **S3 logs**.

   If you select **CloudWatch logs**:
   • In **Group name**, enter the name of your Amazon CloudWatch Logs group.
   • In **Stream name**, enter your Amazon CloudWatch Logs stream name.

   If you select **S3 logs**:
   • From **Bucket**, choose the name of the S3 bucket for your logs.
   • In **Path prefix**, enter the prefix for your logs.
   • Select **Remove S3 log encryption** if you do not want your S3 logs encrypted.

22. To change information about the way build output artifacts are stored, in **Additional configuration**, change the value of **Artifacts packaging**.

23. To change whether build artifacts are encrypted, use **Disable artifacts encryption**.

24. Choose **Update artifacts**.

### Change a build project's settings (AWS CLI)

For information about using the AWS CLI with AWS CodeBuild, see the [Command line reference](#) (p. 355).

1. Run the `update-project` command as follows:

   ```bash
   aws codebuild update-project --generate-cli-skeleton
   ```

   JSON-formatted data appears in the output. Copy the data to a file (for example, `update-project.json`) in a location on the local computer or instance where the AWS CLI is installed. Then modify the copied data as described in [Create a build project (AWS CLI)](#) (p. 229), and save your results.

   **Note**
   In the JSON-formatted data, you must provide the name of the build project. All other settings are optional. You cannot change the build project's name, but you can change any of its other settings.

2. Switch to the directory that contains the file you just saved, and run the `update-project` command again.

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

3. If successful, data similar to that described in [Create a build project (AWS CLI)](#) (p. 229) appears in the output.
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 (p. 356).

Delete a build project 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) (p. 265)
- Delete a build project (AWS CLI) (p. 265)
- Delete a build project (AWS SDKs) (p. 266)

Delete a build project (console)

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:

   ```bash
   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) (p. 243).

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 (p. 355).
Delete a build project (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference (p. 356).

Working with shared 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.

Contents

- Prerequisites for sharing projects (p. 266)
- Prerequisites for accessing shared projects shared with you (p. 266)
- Related services (p. 266)
- Sharing a project (p. 267)
- Unsharing a shared project (p. 268)
- Identifying a shared project (p. 268)
- Shared project permissions (p. 269)

Prerequisites for sharing projects

To share a project, your AWS account must own it. You cannot share a project that has been shared with you.

Prerequisites for accessing shared projects shared with you

To access a shared report group, a consumer's IAM role requires the BatchGetProjects permission. You can attach the following policy to their IAM role:

```json
{
    "Effect": "Allow",
    "Resource": [
        "*
    ],
    "Action": [
        "codebuild:BatchGetProjects"
    ]
}
```

For more information, see Using identity-based policies for AWS CodeBuild (p. 310).

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.
Sharing a project

The consumer can use the AWS CLI but not the 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.

**To share a project that you own (CodeBuild console)**

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.

   ```json
   {
   "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 project with the ARN `arn:aws:codebuild:us-`
west-2:123456789012:project/my-project to Alice and the root user for the AWS account identified by 123456789012.

```json
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS": [
          "arn:aws:iam::123456789012:user/Alice",
          "123456789012"
        ]
      },
      "Resource": "arn:aws:codebuild:us-west-2:123456789012:project/my-project"
    }
  ]
}
```

3. Run the following command.

```bash
aws codebuild put-resource-policy --resource-arn project-arn --policy file://policy.json
```

---

**Unsharing 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:

```bash
aws codebuild delete-resource-policy --resource-arn project-arn
```

**Identifying 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.

Working with 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 Builds (p. 374).

• Builds are queued if the number of concurrently running builds reaches its limit. The maximum number of builds in a queue is five times the concurrent build limit. For more information, see Builds (p. 374).

• 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 a build in AWS CodeBuild (p. 269).

• 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 a build in AWS CodeBuild (p. 269)
• View build details in AWS CodeBuild (p. 278)
• View a list of build IDs in AWS CodeBuild (p. 280)
• View a list of build IDs for a build project in AWS CodeBuild (p. 282)
• Stop a build in AWS CodeBuild (p. 283)
• Delete builds in AWS CodeBuild (p. 284)

Run a build in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs to run a build in CodeBuild.

Topics

• Run a build (console) (p. 270)
• Run a build (AWS CLI) (p. 273)
Run a build (console)

To use AWS CodePipeline to run a build with CodeBuild, skip these steps and follow the instructions in Use AWS CodePipeline with AWS CodeBuild (p. 195).

2. Do one of the following:
   - If you just finished creating a build project, the Build project: project-name page should be displayed. Choose Start build.
   - If you created a build project earlier, in the navigation pane, choose Build projects. Choose the build project, and then choose Start build.
3. On the Start build page, do one of the following:
   - For Amazon S3, for the optional Source version value, enter the version ID for the version of the input artifact you want to build. If Source version is left blank, the latest version is used.
   - For CodeCommit, for Reference type, choose Branch, Git tag, or Commit ID. Next, choose the branch, Git tag, or enter a commit ID to specify the version of your source code. For more information, see Source version sample with AWS CodeBuild (p. 139). Change the value for Git clone depth. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose Full.
   - For GitHub or GitHub Enterprise, for the optional Source version value, enter a commit ID, pull request ID, branch name, or tag name for 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. Change the value for Git clone depth. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose Full.
   - For Bitbucket, for the optional Source version value, enter a commit ID, branch name, or tag name for 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. Change the value for Git clone depth. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose Full.
   - To use a different source provider for this build only, choose Advanced build options. For more information about source provider options and settings, see Choose source provider.
4. Choose Advanced build overrides.

   Here you can change settings for this build only. The settings in this section are optional.

   Under Source, you can:
   - Choose Add source to add a secondary source.
   - Choose Remove source to remove a secondary source.
   - Use Source provider and Source version to modify settings for a source.

   Under Environment, you can:
   - Override settings for Environment image, Operating system, Runtime, and Runtime version.
   - Select or clear Privileged.
Note
By default, Docker containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.

- In Service role, you can change the service role that CodeBuild uses to call dependent AWS services for you. Choose New service role to have CodeBuild create a service role for you.
- Choose Override build specification to use a different build specification.
- Change the value for Timeout.
- Change the value for Compute.
- From Certificate, choose a different setting.

Under Buildspec, you can:

- Choose Use a buildspec file to use a buildspec.yml 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 by its ARN (for example, arn:aws:s3:::my-codebuild-sample2/buildspec.yml).
- Choose Insert build commands to enter commands you want to run during the build phase.

Under Build Artifacts, you can:

- From Type, choose a different artifacts type.
- In Name, enter a different output artifact name.
- If you want a name specified in the buildspec file to override any name specified in the console, select Enable semantic versioning. The name 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 Buildspec syntax (p. 150).
- In Path, enter a different output artifact path.
- In Namespace type, choose a different type. Choose Build ID to insert the build ID into the path of the build output file (for example, My-Path/Build-ID/My-Artifact.zip). Otherwise, choose None.
- From Bucket name choose a different S3 bucket for your output artifacts.
- If you do not want your build artifacts encrypted, select Disable artifacts encryption.
- Select Artifacts packaging, and then choose Zip to put the build artifact files in a compressed file. To put the build artifact files in the specified S3 bucket individually (not compressed), choose None.
- Under Cache, from Type, choose a different cache setting.
- To override secondary artifacts for this build only:
  - To remove a secondary artifact, in Secondary artifacts, choose the X in its row.
  - To add a secondary artifact, choose Add artifact, and then enter the information for your secondary artifact. For more information, see step 8 in Create a build project (console) (p. 216).

Under Logs, you can override your log settings by selecting or clearing CloudWatch Logs and S3 logs.

- If you enable CloudWatch logs:
• In **Group name**, enter the name of your Amazon CloudWatch Logs group.
• In **Stream name**, enter your Amazon CloudWatch Logs stream name.

• If you enable **S3 logs**:
  • From **Bucket**, choose the name of the S3 bucket for your logs.
  • In **Path prefix**, enter the prefix for your logs.

Under **Service role**, you can change the service role that CodeBuild uses to call dependent AWS services for you. Choose **Create a role** to have CodeBuild create a service role for you.

5. Expand **Environment variables override**.

If you want to change the environment variables for this build only, change the values for **Name**, **Value**, and **Type**. Choose **Add environment variable** to add a new environment variable for this build only. Choose **Remove environment variable** to remove an environment variable you do not want to use in this build.

Others can see an environment variable by using the CodeBuild console and the AWS CLI. 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. For **Type**, choose **Parameter**. For **Name**, type 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**
We recommend that you store parameters in Amazon EC2 Systems Manager Parameter Store 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 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. 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 **Create a 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.

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 **Create a service role in your account**, then 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/`.

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...
Run a build

For detailed information about this build, see View build details (console) (p. 278).

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) (p. 199).
For more information about using the AWS CLI with CodeBuild, see the Command line reference (p. 355).

1. Run the `start-build` command in one of the following ways:

   ```bash
   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.

   ```bash
   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 a list of build project names (p. 242).

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 12 hours 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:

   ```json
   {
   "projectName": "projectName",
   "sourceVersion": "sourceVersion",
   "artifactsOverride": {
   "type": "type",
   "location": "location",
   "path": "path",
   "namespaceType": "namespaceType",
   "name": "artifactsOverride-name",
   }}
   ```
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 **artifactsOveride**.
  - **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.
Run a build

- **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 (p. 149).

- 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 12 hours 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 TSL setting specified in the build project. The insecure TSL setting determines whether to ignore TSL warnings while connecting to the project source code. This override applies only if the build's source is GitHub Enterprise.

- **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, 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.

**Important**

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 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 **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) (p. 229). For a list of the latest settings for a build project, see View a build project's details (p. 244).

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 (p. 23) procedure appears in the output.

To work with detailed information about this build, make a note of the \textit{id} value in the output, and then see View build details (AWS CLI) (p. 278).

**Start running builds automatically (AWS CLI)**

If your source code is stored in a GitHub or a GitHub Enterprise 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 \texttt{create-webhook} command as follows:

```
aws codebuild create-webhook --project-name
```

- where \texttt{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"
   }
}
```

- where \texttt{url} is the URL to the GitHub webhook.

For GitHub Enterprise, information similar to the following appears in the output:

```
{
   "webhook": {"id": 12345678901, "name": "GitHub Enterprise Webhook", "enabled": true, "events": ["push"], "lastUpdated": "2023-01-01T12:00:00Z", "url": "https://example.com/webhook-url"}
}
```

1. Copy the secret key and payload URL from the output. You need them to add a webhook in GitHub Enterprise.
2. In GitHub Enterprise, choose the repository where your CodeBuild project is stored. Choose \textit{Settings}, choose \textit{Hooks & services}, and then choose \textit{Add webhook}.
3. Enter the payload URL and secret key, accept the defaults for the other fields, and then choose \textit{Add webhook}.

**Stop running builds automatically (AWS CLI)**

If your source code is stored in a GitHub or a GitHub Enterprise 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) (p. 277).
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
```

- 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 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 CodePipeline with AWS CodeBuild to test code and run builds (p. 195) instead.

For information about using CodeBuild with the AWS SDKs, see the AWS SDKs and tools reference (p. 356).

**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) (p. 278)
- View build details (AWS CLI) (p. 278)
- View build details (AWS SDKs) (p. 279)
- Build phase transitions (p. 279)

**View build details (console)**

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 (p. 355).
Run the **batch-get-builds** command:

```bash
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) (p. 281)
  - View a list of build IDs for a build project (AWS CLI) (p. 282)

For example, if you run this command:

```bash
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 (p. 24) 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 (p. 356).

**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) (p. 281)
- View a list of build IDs (AWS CLI) (p. 281)
View a list of build IDs (AWS SDKs) (p. 282)

**View a list of build IDs (console)**

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 (p. 355).

- Run the **list-builds** command:

  ```bash
  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:

  ```bash
  aws codebuild list-builds --sort-order ASCENDING
  ```

  A result similar to the following might appear in the output:

  ```json
  {
  "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:

  ```bash
  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:
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 (p. 356).

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) (p. 282)
- View a list of build IDs for a build project (AWS CLI) (p. 282)
- View a list of build IDs for a build project (AWS SDKs) (p. 283)

View a list of build IDs for a build project (console)

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 (p. 355).

Run the list-builds-for-project command, as follows:

```bash
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) (p. 243).
• **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:

```bash
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:

```bash
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:a5405d2b-1ab3-44df-a2e2d-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 (p. 356).

**Stop a build 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) (p. 284)
- Stop a build (AWS CLI) (p. 284)
- Stop a build (AWS SDKs) (p. 284)
Stop a build (console)

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) (p. 281)
  - View a list of build IDs for a build project (AWS CLI) (p. 282)

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 (p. 356).

Delete builds in AWS CodeBuild

You can use the AWS CLI or the AWS SDKs to delete builds in AWS CodeBuild.

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) (p. 281)
  - View a list of build IDs for a build project (AWS CLI) (p. 282)

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:

```json
{
    "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::123456789012n: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](p. 356).
Working with test reporting in AWS CodeBuild

The test reporting feature is in preview release for CodeBuild and is subject to change.

You can create reports in AWS 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 test file format can be JUnit XML or Cucumber JSON. Create your test cases with any test framework that can create files in one of those formats (for example, Surefire JUnit plugin, TestNG, and 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. 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 S3 bucket. Exported test files do not expire. Information about the S3 bucket is specified when you create the report group.

Topics
- Create a test report (p. 286)
- Working with report groups (p. 287)
- Working with reports (p. 297)
- Working with test report permissions (p. 298)
- View test reports (p. 301)

Create a test report

The test reporting feature is in preview release for CodeBuild and is subject to change.

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 (p. 215).
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-
Working with report groups

The test reporting feature is in preview release for CodeBuild and is subject to change.

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 5 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 (p. 287)
- Update a report group (p. 291)
- Specify test files (p. 292)
- Specify test commands (p. 293)
- Report group naming (p. 293)
- Working with shared report groups (p. 294)

Create a report group

The test reporting feature is in preview release for CodeBuild and is subject to change.
You can use the AWS 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 Working with test report permissions (p. 298).

Topics
- Create a report group (buildspec) (p. 288)
- Create a report group (CLI) (p. 288)
- Create a report group (console) (p. 289)
- Create a report group (AWS CloudFormation) (p. 290)

Create a report group (buildspec)

The test reporting feature is in preview release for CodeBuild and is subject to change.

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 (p. 291).

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:

   ```json
   reports:
     new-report-group: #surefire junit reports
     files:
       - '**/*'
     base-directory: 'surefire/target/surefire-reports'
   
   For more information, see Specify test files (p. 292) 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 (p. 293).
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 (p. 293).

Create a report group (CLI)

The test reporting feature is in preview release for CodeBuild and is subject to change.

To create a test report

- Create a file named CreateReportGroup.json.

1. 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 S3 bucket.
Create a report group

```

```

Replace bucket-name with your S3 bucket name and path with the path in your S3 bucket to where you want to export the files. 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 master key (CMK). For more information, see Update a report group (p. 291).

- Use the following JSON to specify that your test report does not export raw test files:

```

```

2. Run the following command:

```

aws codebuild create-report-group \
--cli-input-json file://CreateReportGroupInput.json \
--region us-east-2
```

Create a report group (console)

The test reporting feature is in preview release for CodeBuild and is subject to change.

To create a test report

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. If you want to upload the raw data of your test report results to an S3 bucket:
   a. Select Backup to Amazon S3.
   b. For S3 bucket name, enter the name of the S3 bucket.
   c. For Path prefix, enter the path in your S3 bucket where you want to upload your test results.
   d. Select Compress test result data in a zip file to compress your raw test result data files.
   e. Expand Additional configuration to display encryption options. Choose one of the following:
Create a report group

- **Default AWS managed key** to use a customer master key (CMK) for Amazon S3 that is managed by the AWS Key Management Service. In CodeBuild, the default CMK is for Amazon S3 and uses the format `aws/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 CMK 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 (p. 304).

6. Choose **Create report group**.

### Create a report group (AWS CloudFormation)

The test reporting feature is in preview release for CodeBuild and is subject to change.

**To create a test report**

You can use an AWS CloudFormation template file to create and provision a report group. For more information, see the 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 S3 bucket.

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: S3
        S3Destination:
          Bucket: my-s3-bucket-name
          Path: path-to-folder-for-exported-files
          Packaging: ZIP
          EncryptionKey: my-KMS-encryption-key
          EncryptionDisabled: false
```
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 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:
  - A customer master key (CMK) for Amazon S3 that is managed by the AWS Key Management Service.
  - A CMK that you create and configure.

For more information, see Data encryption (p. 304).

Update a report group (console)

To update a report group
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 customer master key (CMK) for Amazon S3 that is managed by the AWS Key Management Service. In CodeBuild, the default CMK is for Amazon S3 and uses the format aws/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 CMK 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 option 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)

To update a report group
1. Create a file named UpdateReportGroupInput.json.
2. Copy the following into UpdateReportGroupInput.json:

```json
{
  "BackupToS3": true,
  "S3BucketName": "my-bucket",
  "PathPrefix": "reportgroup1",
  "CompressTestResultData": true,
  "Encryption": {
    "DefaultAWSManagedKey": true,
    "KMSEncryptionKey": "arn:aws:kms:us-west-2:123456789012:key/abcdefg12345",
    "DisableEncryption": false
  }
}
```
Specify test files

The test reporting feature is in preview release for CodeBuild and is subject to change.

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 S3 bucket are discarded.

```json
reports:
  files:
```
Specify test commands

The test reporting feature is in preview release for CodeBuild and is subject to change.

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:

```bash
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
```

Report group naming

The test reporting feature is in preview release for CodeBuild and is subject to change.

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`.

**Working with shared 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.

**Contents**

- Prerequisites for sharing report groups (p. 294)
- Prerequisites for accessing report groups shared with you (p. 294)
- Related services (p. 294)
- Sharing a report group (p. 295)
- Unsharing a shared report group (p. 296)
- Identifying a shared report group (p. 296)
- Shared report group permissions (p. 297)

**Prerequisites for sharing report groups**

To share a report group, your AWS account must own it. You cannot share a report group that has been shared with you.

**Prerequisites for accessing 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:

```json
{
  "Effect": "Allow",
  "Resource": [
    "*
  ],
  "Action": [
    "codebuild:BatchGetReportGroups"
  ]
}
```

For more information, see Using identity-based policies for AWS CodeBuild (p. 310).

**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](https://docs.aws.amazon.com/ram/latest/userguide/).
Sharing 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.

To share a report group that you own (CodeBuild console)

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)


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:

```json
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS":"consumer-aws-account-id-or-user"
         },
         "Action":[
            "codebuild:BatchGetReportGroups",
            "codebuild:BatchGetReports",
            "codebuild:ListBuildsForProject",
            "codebuild:DescribeTestCases",
            "arn-of-report-group-to-share"
         ],
         "Resource":"
      }
   ]
}
```

API Version 2016-10-06
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.

```json
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect":"Allow",
         "Principal": { "AWS": [ "arn:aws:iam::123456789012:user/Alice", "123456789012" ] },
      }
   ]
}
```

3. Run the following command.

```
aws codebuild put-resource-policy --resource-arn report-group-arn --policy file://policy.json
```

### Unsharing 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
```

### Identifying 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`:

```bash
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:

```bash
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:

```bash
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.

Working with reports

The test reporting feature is in preview release for CodeBuild and is subject to change.
A report contains the results of test cases that are specified for one report group. A test report is created during the run of a build project. You specify a report group, test case files, and commands to run the test cases in its buildspec file. Each time the test cases run, a new test report is created in the report group.

A test report expires 30 days after it is created. You cannot view an expired test report, but you can export the test results to raw test result files in an S3 bucket. Exported raw test files do not expire. For more information, see Update a report group (p. 291).

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 Working with test report permissions (p. 298).
  - 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.

**Working with test report permissions**

The test reporting feature is in preview release for CodeBuild and is subject to change.

This topic describes important information about permissions related to test reporting.

**Topics**

- Create a role for test reports (p. 299)
- Permissions for test reporting operations (p. 300)
- Test reporting permissions examples (p. 300)
Create a role for test reports

The test reporting feature is in preview release for CodeBuild and is subject to change.

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

**Note**

BatchPutTestCases, CreateReport, and UpdateReport 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:

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

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:

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

The following restricts permissions to only report groups created by running builds of a project named my-project:

```json
{
```
Permissions for test reporting operations

The test reporting feature is in preview release for CodeBuild and is subject to change.

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 (p. 321).

Test reporting permissions examples

The test reporting feature is in preview release for CodeBuild and is subject to change.

For information about sample policies related to test reporting, see the following:

- Allow a user to change a report group (p. 317)
- Allow a user to create a report group (p. 315)
- Allow a user to delete a report (p. 316)
- Allow a user to delete a report group (p. 316)
- Allow a user to get information about report groups (p. 314)
- Allow a user to get information about reports (p. 315)
- Allow a user to get a list of report groups (p. 318)
- Allow a user to get a list of reports (p. 318)
- Allow a user to get a list of reports for a report group (p. 319)
- Allow a user to get a list of test cases for a report (p. 319)
View test reports

The test reporting feature is in preview release for CodeBuild and is subject to change.

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 (p. 301)
- View test reports for a report group (p. 301)
- View test reports in your AWS account (p. 302)

View test reports for a build

The test reporting feature is in preview release for CodeBuild and is subject to change.

To view test reports for a build
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

The test reporting feature is in preview release for CodeBuild and is subject to change.

To view test reports in a report group
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**

The test reporting feature is in preview release for CodeBuild and is subject to change.

**To view test reports in your AWS account**

2. In the navigation pane, choose Report history.
3. Choose a test report to see its details.
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 (p. 303)
- Identity and access management in AWS CodeBuild (p. 305)
- Logging and monitoring in AWS CodeBuild (p. 327)
- Compliance validation for AWS CodeBuild (p. 343)
- Resilience in AWS CodeBuild (p. 343)
- Infrastructure security in AWS CodeBuild (p. 343)

Data protection in AWS CodeBuild

AWS CodeBuild conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that 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 TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls in AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers’ account numbers, into free-form 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 CodeBuild or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don’t include credentials information in the URL to validate your request to that server.
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 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 (p. 304).

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

Topics
- Data encryption (p. 304)
- Key management (p. 304)
- Traffic privacy (p. 305)

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 customer master keys (CMKs) for Amazon S3 that are managed by the AWS Key Management Service. If you do not want to use these CMKs, you must create and configure a customer-managed CMK. 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 (p. 172)

  - You can specify a customer-managed CMK 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 CMKs managed by AWS.

- **Encryption of data in-transit** - All communication between customers and CodeBuild and between CodeBuild and its downstream dependencies is protected using TSL connections that are signed using the Signature Version 4 signing process. All CodeBuild endpoints use SHA-256 certificates that are managed by AWS Certificate Manager Private Certificate Authority. For more information, see Signature Version 4 Signing Process and What is ACM PCA.

- **Build artifact encryption** - CodeBuild requires access to an AWS KMS CMK in order to encrypt its build output artifacts. By default, CodeBuild uses an AWS Key Management Service CMK for Amazon S3 in your AWS account. If you do not want to use this CMK, you must create and configure a customer-managed CMK. For more information Creating KMS Keys.

Key management

You can protect your content from unauthorized use through encryption. Store your encryption keys in AWS Secrets Manager, and then give CodeBuild permission to obtain the encryption keys from your Secrets Manager account. For more information, see Create and configure an AWS KMS
CMK for CodeBuild (p. 353), Create a build project in AWS CodeBuild (p. 215), Run a build in AWS CodeBuild (p. 269), and Tutorial: Storing and Retrieving a Secret.

Use the CODEBUILD_KMS_KEY environment variable in a build command for your AWS KMS key. For more information, see Environment variables in build environments (p. 172).

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 (p. 141).

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 (p. 180). 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:

- Authentication (p. 305)
- Access control (p. 306)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your root credentials and they provide complete access to all of your AWS resources.

  **Important**
  
  For security reasons, we recommend that you use the root credentials only to create an administrator user, which is an IAM user with full permissions to your AWS account. Then, you can use this administrator user to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the IAM User Guide.

- **IAM user** – An IAM user is simply an identity in your AWS account that has custom permissions (for example, permission to create build projects in CodeBuild). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

  In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the AWS SDKs or by using the AWS Command Line Interface (AWS CLI). The AWS SDKs and AWS CLI tools use the access keys to cryptographically sign your request. If you don’t use the AWS tools, you must sign the request yourself. CodeBuild supports Signature Version 4, a protocol for authenticating inbound API requests.
For more information about authenticating requests, see the Signature Version 4 Signing Process in the AWS General Reference.

- **IAM role** – An IAM role is similar to an IAM user, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
  - **Federated user access** – Instead of creating an IAM user, you can use preexisting user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as federated users. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the IAM User Guide.
  - **Cross-account access** – You can use an IAM role in your account to grant another AWS account permissions to access your account's resources. For an example, see Tutorial: Delegate Access Across AWS Accounts Using IAM Roles in the IAM User Guide.
  - **AWS service access** – You can use an IAM role in your account to grant permissions to an AWS service to access your account’s resources. For example, you can create a role that allows Amazon Redshift to access an S3 bucket on your behalf and then load data stored in the bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the IAM User Guide.
  - **Applications running on Amazon EC2** – Instead of storing access keys in the Amazon EC2 instance for use by applications running on the instance and making AWS API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an Amazon EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the Amazon EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the IAM User Guide.

**Access control**

You can have valid credentials to authenticate your requests, but unless you have permissions, you cannot create or access AWS CodeBuild resources. For example, you must have permissions to create, view, or delete build projects and to start, stop, or view builds.

The following sections describe how to manage permissions for CodeBuild. We recommend that you read the overview first.

- Overview of managing access permissions to your AWS CodeBuild resources (p. 306)
- Using identity-based policies for AWS CodeBuild (p. 310)
- AWS CodeBuild permissions reference (p. 321)
- Viewing resources in the console (p. 326)

**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 (p. 307)
• Understanding resource ownership (p. 308)
• Managing access to resources (p. 308)
• Specifying policy elements: Actions, effects, and principals (p. 309)

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.

<table>
<thead>
<tr>
<th>Resource type</th>
<th>ARN format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build project</td>
<td>arn:aws:codebuild:region-ID:account-ID:project/project-name</td>
</tr>
<tr>
<td>All CodeBuild resources</td>
<td>arn:aws:codebuild:*</td>
</tr>
<tr>
<td>All CodeBuild resources owned by the specified account in the specified AWS Region</td>
<td>arn:aws:codebuild:region-ID:account-ID:*</td>
</tr>
</tbody>
</table>

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": [
```
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 IAM 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 IAM 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 (IAM policies) only.

Identity-based policies

You can attach policies to IAM identities.

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to view build projects and other AWS CodeBuild resources in the AWS CodeBuild console, you can attach a permissions policy to a user or group that the user belongs to.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:

  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
  2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. The principal
in the trust policy must also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see Access Management in the IAM User Guide.

In CodeBuild, identity-based policies are used to manage permissions to the resources related to the deployment process. For example, you can control access to build projects.

You can create IAM policies to restrict the calls and resources that users in your account have access to, and then attach those policies to IAM users. For more information about how to create IAM roles and to explore example IAM policy statements for CodeBuild, see Overview of managing access permissions to your AWS CodeBuild resources (p. 306).

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 be 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 Add CodeBuild access permissions to an IAM group or IAM user (p. 344) and Create a CodeBuild service role (p. 348).

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 (p. 307) and AWS CodeBuild permissions reference (p. 321).

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 (p. 321).
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 (p. 306).

Topics
- Permissions required to use the AWS CodeBuild console (p. 310)
- Permissions required for the AWS CodeBuild console to connect to source providers (p. 311)
- AWS managed (predefined) policies for AWS CodeBuild (p. 311)
- AWS CodeBuild managed policies and notifications (p. 311)
- Customer-managed policy examples (p. 313)

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",
    }
  ]
}
```

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)
- 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 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 IAM 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. 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 IAM users or groups that require those permissions.

AWS 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 full access managed policies

The AWSCodeBuildFullAccess managed policy includes the following statements to allow full access to notifications. Users with this managed policy applied can also create and manage Amazon SNS topics.
for notifications, subscribe and unsubscribe users to topics, and list topics to choose as targets for notification rules.

```
{
  "Sid": "CodeStarNotificationsReadWriteAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-notifications:CreateNotificationRule",
    "codestar-notifications:DescribeNotificationRule",
    "codestar-notifications:UpdateNotificationRule",
    "codestar-notifications:DeleteNotificationRule",
    "codestar-notifications:Subscribe",
    "codestar-notifications:Unsubscribe"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:*"
    }
  }
},
{
  "Sid": "CodeStarNotificationsListAccess",
  "Effect": "Allow",
  "Action": [
    "codestar-notifications:ListNotificationRules",
    "codestar-notifications:ListTargets",
    "codestar-notifications:ListTagsforResource"
  ],
  "Resource": "*
},
{
  "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
  "Effect": "Allow",
  "Action": [
    "sns:CreateTopic",
    "sns:SetTopicAttributes"
  ],
  "Resource": "arn:aws:sns:*:*:codestar-notifications*"
},
{
  "Sid": "SNSTopicListAccess",
  "Effect": "Allow",
  "Action": [
    "sns:ListTopics"
  ],
  "Resource": "*
}
```

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": {
```

API Version 2016-10-06
312
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.

For more information about IAM and notifications, see Identity and Access Management for AWS CodeStar Notifications in the Developer Tools Console User Guide.

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 permissions specific to the console. For information, see Permissions required to use the AWS CodeBuild console (p. 310).

You can use the following sample IAM policies to limit CodeBuild access for your IAM users and roles.

Topics
- Allow a user to get information about build projects (p. 314)
- Allow a user to get information about report groups (p. 314)
- Allow a user to get information about reports (p. 315)
- Allow a user to create build projects (p. 315)
- Allow a user to create a report group (p. 315)
- Allow a user to delete a report group (p. 316)
- Allow a user to delete a report (p. 316)
- Allow a user to delete build projects (p. 316)
- Allow a user to get a list of build project names (p. 316)
- Allow a user to change information about build projects (p. 317)
- Allow a user to change a report group (p. 317)
- Allow a user to get information about builds (p. 317)
- Allow a user to get a list of build IDs for a build project (p. 318)
- Allow a user to get a list of build IDs (p. 318)
- Allow a user to get a list of report groups (p. 318)
- Allow a user to get a list of reports (p. 318)
- Allow a user to get a list of reports for a report group (p. 319)
- Allow a user to get a list of test cases for a report (p. 319)
- Allow a user to start running builds (p. 319)
- Allow a user to attempt to stop builds (p. 320)
- Allow a user to attempt to delete builds (p. 320)
- Allow a user to get information about Docker images that are managed by CodeBuild (p. 320)
- Allow CodeBuild access to AWS services required to create a VPC network interface (p. 320)
- Use a deny statement to prevent AWS CodeBuild from disconnecting from source providers (p. 321)

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",
        }
    ]
}
```

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:
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",
        }
    ]
}
```

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",
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam:123456789012:role/CodeBuildServiceRole"
        }
    ]
}
```

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",
        }
    ]
}
```
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:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codebuild:DeleteReportGroup",
      }
   ]
}
```

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:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codebuild:DeleteReport",
      }
   ]
}
```

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:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codebuild:DeleteProject",
      }
   ]
}
```

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:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codebuild:ListProjects",
         "Condition": { "StringEquals": { "aws:PrincipalARN": "arn:aws:sts::123456789012:principal" } }
      }
   ]
}
```
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",
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam:123456789012:role/CodeBuildServiceRole"
    }
  ]
}
```

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",
    }
  ]
}
```

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",
    }
  ]
}
```
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:

```json
{
   "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:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codebuild:ListBuilds",
         "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:

```json
{
   "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:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codebuild:GetBuildReport",
         "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:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:ListReports",
            "Resource": "*"
        }
    ]
}
```

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:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:DescribeTestCases",
        }
    ]
}
```

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:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:StartBuild",
        }
    ]
}
```
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",
    }
  ]
}
```

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",
    }
  ]
}
```

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 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",
      "Resource": "*"
    }
  ]
}
```
"Effect": "Allow",
"Action": [
  "ec2:CreateNetworkInterface",
  "ec2:DescribeDhcpOptions",
  "ec2:DescribeNetworkInterfaces",
  "ec2:DeleteNetworkInterface",
  "ec2:DescribeSubnets",
  "ec2:DescribeSecurityGroups",
  "ec2:DescribeVpcs"
],
"Resource": "*
},
"Condition": {
  "StringEquals": {
    "ec2:AuthorizedService": "codebuild.amazonaws.com"
  }
}
}

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 (p. 311).

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {"Effect": "Deny",
     "Action": "codebuild:DeleteOAuthToken",
     "Resource": "*
    }
  ]
}
```

AWS CodeBuild permissions reference

You can use the following table as a reference when you are setting up Access control (p. 306) and writing permissions policies that you can attach to an IAM identity (identity-based policies).

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

API Version 2016-10-06
321
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.


- **BatchGetReports**
  
  **Action**: codebuild:BatchGetReports
  
  Required to get information about reports.


- **CreateProject**
  
  **Actions**: codebuild:CreateProject, iam:PassRole
  
  Required to create build projects.

CreateReportGroup

**Action:** codebuild:CreateReportGroup

Required to create a report group.

**Resource:** arn:aws:codebuild:*:*:report-group/report-name

CreateWebhook

**Action:** codebuild:CreateWebhook

Required to create a webhook.

**Resource:** arn:aws:codebuild:*:*:project/project-name

DeleteReport

**Action:** codebuild:DeleteReport

Required to delete a report.

**Resource:** arn:aws:codebuild:*:*:report-group/report-name

DeleteReportGroup

**Action:** codebuild:DeleteReportGroup

Required to delete a report group.

**Resource:** arn:aws:codebuild:*:*:report-group/report-name

DeleteSourceCredentials

**Action:** codebuild:DeleteSourceCredentials

Required to delete a set of SourceCredentialsInfo objects that contain information about credentials for a GitHub, GitHub Enterprise, or Bitbucket repository.

**Resource:** *

DeleteWebhook

**Action:** codebuild:DeleteWebhook

Required to delete a webhook.

**Resource:** arn:aws:codebuild:*:*:project/project-name

DescribeTestCases

**Action:** codebuild:DescribeTestCases

Required to return a paginated list of test cases.

**Resource:** arn:aws:codebuild:*:*:report-group/report-name

ImportSourceCredentials

**Action:** codebuild:ImportSourceCredentials

Required to import a set of SourceCredentialsInfo objects that contain information about credentials for a GitHub, GitHub Enterprise, 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

**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.


**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.
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 IAM users. For information about applying tags to a project using the console or AWS CLI, see Create a build project in AWS CodeBuild (p. 215). 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.

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 IAM users. The aws:ResourceTag condition key is used to control access to resources based on their tags.

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

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.

Logging and monitoring in AWS CodeBuild

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
• Logging AWS CodeBuild API calls with AWS CloudTrail (p. 327)
• Monitoring AWS CodeBuild (p. 329)

Logging 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.

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 IAM 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` Element in the AWS CloudTrail User Guide.

**Understanding 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 (p. 304).

The following example shows a CloudTrail log entry that demonstrates creating a build project in CodeBuild.

```json
{
  "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",
  "eventSourceArn": "arn:aws:codebuild:us-west-2:account-ID:project/project-name",
  "eventSource": "codebuild.amazonaws.com",
  "eventName": "CreateProject",
  "awsRegion": "us-west-2",
  "userId": "account-ID:iam-user",
  "resources": [
    "arn:aws:codebuild:us-west-2:account-ID:project/project-name"
  ],
  "detail": {
    "codeBuild": {
      "projectId": "project-name",
      "projectName": "project-name"
    }
  }
}
```
Monitoring AWS CodeBuild

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 only. To see metrics for a project, specify ProjectName for the dimension in CloudWatch.
- **AWS account level:** These metrics are for all builds in one account. To see metrics at the AWS account level, do not enter a dimension in CloudWatch.

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 executing builds in a build project or an AWS account over time.
Metrics displayed in the CodeBuild console are always from the past three days. You can use the CloudWatch console to view CodeBuild metrics over different durations.

For more information, see Monitoring builds with CloudWatch metrics (p. 332).

**CodeBuild CloudWatch Metrics**

The following metrics can be tracked per AWS account or build project.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuildDuration</td>
<td>Measures the duration of the build's BUILD phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td>Builds</td>
<td>Measures the number of builds triggered.</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Sum</td>
</tr>
<tr>
<td>DownloadSourceDuration</td>
<td>Measures the duration of the build's DOWNLOAD_SOURCE phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td>Duration</td>
<td>Measures the duration of all builds over time.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td>FailedBuilds</td>
<td>Measures the number of builds that failed because of client error or a timeout.</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Sum</td>
</tr>
<tr>
<td>FinalizingDuration</td>
<td>Measures the duration of the build's FINALIZING phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td>InstallDuration</td>
<td>Measures the duration of the build's INSTALL phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
</tbody>
</table>
### Metric Description

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td><strong>PostBuildDuration</strong></td>
<td>Measures the duration of the build's POST_BUILD phase</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td><strong>PreBuildDuration</strong></td>
<td>Measures the duration of the build's PRE_BUILD phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td><strong>ProvisioningDuration</strong></td>
<td>Measures the duration of the build's PROVISIONING phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td><strong>QueuedDuration</strong></td>
<td>Measures the duration of the build's QUEUED phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td><strong>SubmittedDuration</strong></td>
<td>Measures the duration of the build's SUBMITTED phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
<tr>
<td><strong>SucceededBuilds</strong></td>
<td>Measures the number of successful builds.</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Sum</td>
</tr>
<tr>
<td><strong>UploadArtifactsDuration</strong></td>
<td>Measures the duration of the build's UPLOAD_ARTIFACTS phase.</td>
</tr>
<tr>
<td></td>
<td>Units: Seconds</td>
</tr>
<tr>
<td></td>
<td>Valid CloudWatch statistics: Average (recommended), Maximum, Minimum</td>
</tr>
</tbody>
</table>
CodeBuild CloudWatch dimensions

ProjectName is the only AWS CodeBuild metrics dimension. If it is specified, then the metrics are for that project. If it is not specified, then the metrics are for the current AWS account.

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:

- **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 Monitoring builds with CloudWatch alarms (p. 337). For more information about alarms, see Creating Amazon CloudWatch Alarms in the Amazon CloudWatch User Guide.

Monitoring builds with CloudWatch metrics

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 access metrics.

Access build metrics (CodeBuild console)

The graphs in the CodeBuild console show three days of metrics. You cannot customize the metrics or the graphs used to display them. Use the Amazon CloudWatch console to view your build metrics if you want to edit them.

To access AWS account level metrics

2. In the navigation pane, choose **Account metrics**.

To access project-level metrics

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.

Access build metrics (Amazon CloudWatch console)

You can customize the metrics and the graphs used to display them.
To access 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.
Metrics

Favorites

Add a dashboard
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.

To access 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.
Metrics

Favorites

Add a dashboard
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*.

**Monitoring 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**.
Create Alarm

1. Select Metric

Browse Metrics

CloudWatch Metrics

Your CloudWatch metric will be...

CodeBuild Metrics: 4
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**.
1. Select Metric
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*.

The following shows an alarm that sends an Amazon SNS notification to a list named `codebuild-sns-notifications` when one or more failed builds are detected over 15 minutes. The 15 minutes is calculated by multiplying the five minute period by the three specified data points. The information displayed for a failed builds alarm at the project level or account level is identical.
Create Alarm

1. Select Metric

Alarm Threshold

Provide the details and appropriate threshold.

Name: Account
8. Choose **Create Alarm**.

### 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** – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

### 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 the AWS global network security procedures that are described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

You use AWS published API calls to access CodeBuild through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

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.
Advanced topics

This section includes several advanced topics that are useful to more experienced AWS CodeBuild users.

Topics

• Advanced setup (p. 344)
• Command line reference for AWS CodeBuild (p. 355)
• AWS SDKs and tools reference for AWS CodeBuild (p. 356)
• Specify the AWS CodeBuild endpoint (p. 356)

Advanced setup

If you follow the steps in Getting started using the console (p. 5) 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 IAM groups and users in your organization access to CodeBuild, modify existing service roles in IAM or customer master keys in AWS KMS to access CodeBuild, or set up the AWS CLI across your organization’s workstations to access CodeBuild. This topic describes how to complete the related setup steps.

We assume you already have an AWS account. However, if you do not already have one, go to http://aws.amazon.com, choose Sign In to the Console, and follow the online instructions.

Topics

• Add CodeBuild access permissions to an IAM group or IAM user (p. 344)
• Create a CodeBuild service role (p. 348)
• Create and configure an AWS KMS CMK for CodeBuild (p. 353)
• Install and configure the AWS CLI (p. 354)

Add CodeBuild access permissions to an IAM group or IAM user

To access AWS CodeBuild with an IAM group or IAM user, you must add access permissions. 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 IAM user in your AWS account, then you do not need to follow these instructions.

For information about AWS root accounts and administrator IAM users, see The Account Root User and Creating Your First IAM Admin User and Group in the IAM User Guide.

To add CodeBuild access permissions to an IAM group or IAM 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:
AWS CodeBuild User guide
Add CodeBuild access permissions
to an IAM group or IAM user

- Your AWS root account. This is not recommended. For more information, see The Account Root
  User in the IAM User Guide.
- An administrator IAM user in your AWS account. For more information, see Creating Your First IAM
  Admin User and Group in the IAM User Guide.
- An IAM user in your AWS account with permission to perform the following minimum set of
  actions:

```xml
<actions>
  <name>iam:AttachGroupPolicy</name>
  <name>iam:AttachUserPolicy</name>
  <name>iam:CreatePolicy</name>
  <name>iam:ListAttachedGroupPolicies</name>
  <name>iam:ListAttachedUserPolicies</name>
  <name>iam:ListGroups</name>
  <name>iam:ListPolicies</name>
  <name>iam:ListUsers</name>
</actions>
```

For more information, see Overview of IAM Policies in the IAM 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 IAM 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 IAM user, and then choose Attach Policy. Repeat
  this for the policy named AmazonS3ReadAccess.
- To add read-only access permissions to CodeBuild, select the boxes named
  AWSCodeBuildReadOnlyAccess. Select the box next to the target IAM group or IAM 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 IAM 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.

```json
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Sid": "CodeBuildDefaultPolicy",
    "Effect": "Allow",
    "Action": [
      "codebuild:*",
      "iam:PassRole"
    ],
  },
  {
    "Sid": "CodeBuildReadOnlyPolicy",
    "Effect": "Allow",
    "Action": [
      "codebuild:*",
      "iam:PassRole"
    ],
  }
]}
```

API Version 2016-10-06
Add CodeBuild access permissions to an IAM group or IAM user

```json
"Resource": "*
},
{
"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: Reivew, and then choose Add permissions.

To add CodeBuild access permissions to an IAM group or IAM 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.

API Version 2016-10-06

346
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 IAM user:

```bash
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 IAM 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 IAM 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.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*",
        "iam:PassRole"
      ],
      "Resource": "*
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "*
    },
    {
      "Sid": "S3AccessPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "*
    }
  ]
}
```

API Version 2016-10-06

347
"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 (p. 305). To restrict access to specific AWS resources, change the value of the related Resource object. For more information, see Identity and access management (p. 305) 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:

```bash
aws iam put-group-policy --group-name group-name --policy-name CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json
```

For an IAM user:

```bash
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 IAM user.

Create a CodeBuild service role

You need an AWS CodeBuild service role so that CodeBuild can interact with dependent AWS services on your behalf. You can create a CodeBuild service role by using the CodeBuild or AWS CodePipeline consoles. For information, see:

- Create a build project (console) (p. 216)
- Create a pipeline that uses CodeBuild (CodePipeline console) (p. 197)
- Add a CodeBuild build action to a pipeline (CodePipeline console) (p. 202)
- Change a build project's settings (console) (p. 253)
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.

**Note**
The service role described on this page contains a policy that grants the minimum permissions required to use CodeBuild. You might need to add additional permissions depending on your use case. For example, if you want to use CodeBuild with Amazon Virtual Private Cloud, then the service role you create requires the permissions in the following policy: Create a CodeBuild service role (p. 348).

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 Account Root User in the IAM User Guide.
   • An administrator IAM user in your AWS account. For more information, see Creating Your First IAM Admin User and Group in the IAM User Guide.
   • An IAM user in your AWS account with permission to perform the following minimum set of actions:

```json
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 IAM 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:

```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": "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": "*
]"}
Create a CodeBuild service role

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:

```json
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": { 
   "Service": "codebuild.amazonaws.com"
   },
   "Action": "sts:AssumeRole"
  }
 ]
}
```

put-role-policy.json:

```json
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CloudWatchLogsPolicy",
   "Effect": "Allow",
   "Action": [
   "logs:CreateLogGroup",
   "logs:CreateLogStream",
   "logs:PutLogEvents"
   ]
  }
 ]
}
```
Create a CodeBuild service role

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

```bash
aws iam create-role --role-name CodeBuildServiceRole --assume-role-policy-document file://create-role.json
```
Create and configure an AWS KMS CMK for CodeBuild

For AWS CodeBuild to encrypt its build output artifacts, it needs access to an AWS KMS customer master key (CMK). By default, CodeBuild uses the AWS-managed CMK for Amazon S3 in your AWS account.

If you do not want to use this CMK, you must create and configure a customer-managed CMK yourself. This section describes how to do this with the IAM console.

For information about CMKs, see AWS Key Management Service Concepts and Creating Keys in the AWS KMS Developer Guide.

To configure a CMK 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.

```json
{
   "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"
            }
         }
      },
      ### END ADDING STATEMENTS HERE ###
      {
         "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": "*"
      }
   ]
}
```
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 AWS account that owns the CMK.

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 CMK 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 IAM User Guide.
- An administrator IAM user in your AWS account. For more information, see Creating Your First IAM Admin User and Group in the IAM User Guide.
- An IAM user in your AWS account with permission to create or modify the CMK. For more information, see Permissions Required to Use the AWS KMS Console in the AWS KMS Developer Guide.

### Install and configure the AWS CLI
To access AWS CodeBuild, you can use the AWS CLI with—or instead of—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.

1. Run the following command to confirm whether your installation of the AWS CLI supports CodeBuild:

```bash
aws codebuild list-builds
```

If successful, information similar to the following will appear in the output:

```json
{
  "ids": []
}
```
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 (p. 356).

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 Install and configure the AWS CLI (p. 354).

To use the AWS CLI to specify the endpoint for CodeBuild, see Specify the AWS CodeBuild endpoint (AWS CLI) (p. 357).

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) (p. 284).
- `batch-get-builds`: Gets information about multiple builds in CodeBuild. For more information, see View build details (AWS CLI) (p. 278).
- `batch-get-projects`: Gets information about one or more specified build projects. For more information, see View a build project's details (AWS CLI) (p. 244).
- `create-project`: Creates a build project. For more information, see Create a build project (AWS CLI) (p. 229).
- `delete-project`: Deletes a build project. For more information, see Delete a build project (AWS CLI) (p. 265).
- `list-builds`: Lists Amazon Resource Names (ARNs) for builds in CodeBuild. For more information, see View a list of build IDs (AWS CLI) (p. 281).
- `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) (p. 282).
- `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 (p. 166).
- `list-projects`: Gets a list of build project names. For more information, see View a list of build project names (AWS CLI) (p. 243).
- `start-build`: Starts running a build. For more information, see Run a build (AWS CLI) (p. 273).
- `stop-build`: Attempts to stop the specified build from running. For more information, see Stop a build (AWS CLI) (p. 284).
• **update-project**: Changes information about the specified build project. For more information, see Change a build project's settings (AWS CLI) (p. 264).

## AWS SDKs and tools reference for AWS CodeBuild

To use one of 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 (p. 355).

### 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.

## 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) (p. 357)
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 beginning 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 a format to use:

```java
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 beginning 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:

<table>
<thead>
<tr>
<th>Region name</th>
<th>Region</th>
<th>Endpoint</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
<td>codebuild-fips.us-east-1.amazonaws.com</td>
<td>us-east-1-fips</td>
</tr>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
<td>codebuild-fips.us-east-2.amazonaws.com</td>
<td>us-east-2-fips</td>
</tr>
</tbody>
</table>
Specify the AWS CodeBuild endpoint (AWS SDK)

<table>
<thead>
<tr>
<th>Region name</th>
<th>Region</th>
<th>Endpoint</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>US West (N. California)</td>
<td>us-west-1</td>
<td>codebuild-fips.us-west-1.amazonaws.com</td>
<td>us-west-1-fips</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
<td>codebuild-fips.us-west-2.amazonaws.com</td>
<td>us-west-2-fips</td>
</tr>
</tbody>
</table>

To specify use of the FIPS endpoint in the US West (Oregon) region using an alias:

```java
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:

```java
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:

```java
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    .withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1", "ap-south-1")).
    .withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    .build();
```
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 (p. 327).

Topics
- Apache Maven builds reference artifacts from the wrong repository (p. 360)
- Build commands run as root by default (p. 361)
- Builds might fail when file names have non-U.S. English characters (p. 361)
- Builds might fail when getting parameters from Amazon EC2 Parameter Store (p. 361)
- Cannot access branch filter in the CodeBuild console (p. 362)
- Cannot view build success or failure (p. 362)
- Cannot view build success or failure (p. 362)
- Cannot find and select the base image of the Windows Server Core 2016 platform (p. 363)
- Earlier commands in buildspec files are not recognized by later commands (p. 363)
- Error: "Access denied" when attempting to download cache (p. 364)
- Error: "BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE" when using a custom build image (p. 364)
- 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" (p. 365)
- Error: "Cannot connect to the Docker daemon" when running a build (p. 365)
- Error: "CodeBuild is experiencing an issue" when running a build (p. 366)
- Error: "CodeBuild is not authorized to perform: sts:AssumeRole" when creating or updating a build project (p. 366)
- Error: "Error calling GetBucketAcl: Either the bucket owner has changed or the service role no longer has permission to called s3:GetBucketAcl" (p. 367)
- Error: "Failed to upload artifacts: Invalid arn" when running a build (p. 367)
- Error: "Git clone failed: Unable to access 'your-repository-URL': SSL certificate problem: Self signed certificate" (p. 367)
- Error: "The bucket you are attempting to access must be addressed using the specified endpoint" when running a build (p. 368)
- Error: "The policy's default version was not created by enhanced zero click role creation or was not the most recent version created by enhanced zero click role creation."
- Error: "This build image requires selecting at least one runtime version."
- Error: "QUEUED: INSUFFICIENT_SUBNET" when a build in a build queue fails (p. 369)
- Error: "Unable to download cache: RequestError: Send request failed caused by: x509: Failed to load system roots and no roots provided" (p. 370)
- Error: "Unable to download certificate from S3. AccessDenied" (p. 370)
- Error: "Unable to locate credentials" (p. 370)
- RequestError timeout error when running CodeBuild in a proxy server (p. 371)
- The bourne shell (sh) must exist in build images (p. 372)
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.

```xml
<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>
    </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.

```yaml
version 0.2
phases:
  install:
```
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:

```bash
pre_build:
  commands:
  - export LC_ALL="en_US.UTF-8"
  - locale-gen en_US en_US.UTF-8
  - dpkg-reconfigure locales
```

For build environments based on Amazon Linux:

```bash
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/`:

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "ssm:GetParameters",
        "Effect": "Allow",
      }
    ]
  }
  ```

- 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:

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "ssm:GetParameters",
        "Effect": "Allow",
      }
    ]
  }
  ```

---

**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 Filter Bitbucket webhook events (console) (p. 78) and Filter GitHub webhook events (console) (p. 124).

---

**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.

---

**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.

---

**Cannot find and select the base image of the Windows Server Core 2016 platform**

**Issue:** You cannot find or select the base image of the Windows Server Core 2016 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 2016 platform is supported:

- US East (N. Virginia)
- US East (Ohio)
- US West (N. California)

---

**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 (p. 172) and Environment variables in build environments (p. 172).
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 causes:**

- 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 types (p. 170).
- AWS CodeBuild does not have permission to pull the build image from your Amazon Elastic Container Registry (Amazon ECR).
- The Amazon ECR image you requested is not available in the AWS Region that your AWS account is using.
- 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 (p. 141).

**Recommended solutions:**

- Use a larger compute type with more available disk space, or reduce the size of your custom build image.
- 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 (p. 53).
- Use an Amazon ECR image that is in the same AWS Region as the one your AWS account is using.
- If you use a private registry in a VPC, make sure the VPC has public internet access.
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:** Follow these steps to run your build in privileged mode:

2. In the navigation pane, choose Build projects, and then choose your build project.
3. From Edit, choose Environment.
4. Choose Override images, and then choose Environment.
5. Specify your environment image, operating system, runtime, and image. These settings should match the settings for the build that failed.

**Note**
By default, Docker containers do not allow access to any devices. Privileged mode grants a build project's Docker container access to all devices. For more information, see Runtime Privilege and Linux Capabilities on the Docker Docs website.

7. Choose Update environment.
8. Choose Start build to retry your build.
Error: "CodeBuild is experiencing an issue" when running a build

Issue: When you try to run a build project, you receive this error during the build's PROVISIONING phase.

Possible cause: Your build is using environment variables that are too large for AWS CodeBuild. CodeBuild can raise errors when the length of all environment variables (all names and values added together) reach a combined maximum of around 5,500 characters.

Recommended solution: Use Amazon EC2 Systems Manager Parameter Store to store large environment variables and then retrieve them from your buildspec file. Amazon EC2 Systems Manager Parameter Store can store an individual environment variable (name and value added together) that is a combined 4,096 characters or less. To store large environment variables, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the Amazon EC2 Systems Manager User Guide. To retrieve them, see the parameter-store mapping in Buildspec syntax (p. 150).

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.

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.
- Make sure the target CodeBuild service role has sufficient permissions to trust CodeBuild. For more information, see the trust relationship policy statement in Create a CodeBuild service role (p. 348).
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 (p. 309).

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 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: "The policy's default version was not created by enhanced zero click role creation or was not the most recent version created by enhanced zero click role creation."

**Issue:** When you try to update a project in the console, the update failed with this error:

**Possible causes:**

- You have updated the policies attached to the target AWS CodeBuild service role.
- You have selected an earlier version of a policy attached to the target CodeBuild service role.

**Recommended solutions:**

- Edit your CodeBuild project and clear the Allow CodeBuild to modify this service role so it can be used with this build project check box. Verify the CodeBuild service role you are using has sufficient permissions. If you edit your CodeBuild project again, you must clear this check box again. For more information, see Create a CodeBuild service role (p. 348).
- Follow these steps to edit your CodeBuild project to use a new service role:
  1. Open the IAM console and create a new service role. For more information, see Create a CodeBuild service role (p. 348).
  3. In the navigation pane, choose Build projects.
  4. Choose the button next to your build project, choose Edit, and then choose Environment.
  5. For Service role, choose the role you created.
  6. Choose Update environment.
**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:

```yaml
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.
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 (p. 166).

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:


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 (p. 189).
- 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 (AWS PrivateLink) 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 (p. 189) and Buildspec syntax (p. 150).

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 is 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.
Warning: "Skipping install of runtimes.
runtime version selection is not supported
by this build image" when running a build

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.
# 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.

## Build projects

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowed characters in a build project description</td>
<td>Any</td>
</tr>
<tr>
<td>Allowed characters in a build project name</td>
<td>The letters A–Z and a–z, the numbers 0–9, and the special characters – and _</td>
</tr>
<tr>
<td>Length of a build project name</td>
<td>2 to 255 characters, inclusive</td>
</tr>
<tr>
<td>Maximum length of a build project description</td>
<td>255 characters</td>
</tr>
<tr>
<td>Maximum number of build projects</td>
<td>5,000</td>
</tr>
<tr>
<td>Maximum number of build projects you can request information about at one time using the AWS CLI or AWS SDKs</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of reports you can add to a project</td>
<td>5</td>
</tr>
<tr>
<td>Maximum number of tags you can associate with a build project</td>
<td>50</td>
</tr>
<tr>
<td>Number of minutes you can specify in a build project for the build timeout of all related builds</td>
<td>5 to 480 (8 hours)</td>
</tr>
<tr>
<td>Number of security groups you can add under VPC configuration</td>
<td>1 to 5</td>
</tr>
<tr>
<td>Number of subnets you can add under VPC configuration</td>
<td>1 to 16</td>
</tr>
</tbody>
</table>

## Builds

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of builds you can request information about at one time using the AWS CLI or AWS SDKs</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of concurrent running builds*</td>
<td>60</td>
</tr>
<tr>
<td>Maximum time the history of a build is retained</td>
<td>1 year</td>
</tr>
</tbody>
</table>
Number of minutes you can specify for the build timeout of a single build

5 to 480 (8 hours)

* 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. For a new account, the quota can be 1—5. To request a higher concurrent build quota or if you get a "Cannot have more than X active builds for the account" error, contact AWS Support.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum duration a test report is available after it is created</td>
<td>30 days</td>
</tr>
<tr>
<td>Maximum number of report groups per AWS account</td>
<td>1000</td>
</tr>
<tr>
<td>Maximum number of test cases per report</td>
<td>500</td>
</tr>
</tbody>
</table>
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 (p. 376)
- 2) windows-base Docker image—choco (p. 377)
- 3) windows-base Docker image—git --version 2.16.2 (p. 377)
- 4) windows-base Docker image—microsoft-build-tools --version 15.0.26320.2 (p. 377)
- 5) windows-base Docker image—nuget.commandline --version 4.5.1 (p. 380)
- 7) windows-base Docker image—netfx-4.6.2-devpack (p. 380)
- 8) windows-base Docker image—visualfsharptools, v 4.0 (p. 381)
- 9) windows-base Docker image—netfx-pcl-reference-assemblies-4.6 (p. 382)
- 10) windows-base Docker image—visualcppbuildtools v 14.0.25420.1 (p. 384)
- 11) windows-base Docker image—microsoft-windows-netfx3-ondemand-package.cab (p. 386)
- 12) windows-base Docker image—dotnet-sdk (p. 387)

1) base Docker image—windowsservercore

(license terms available at: https://hub.docker.com/r/microsoft/windowsservercore/)

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) (referred 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/usertools.

---

### 2) windows-base Docker image—choco

(license terms available at: https://github.com/chocolatey/chocolatey.org/blob/master/LICENSE.txt)

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
4) windows-base Docker image—microsoft-build-tools --version 15.0.26320.2

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 reconnect 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.

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)

Copyright (c) .NET Foundation. All rights reserved.

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.

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—visualfsharp tools, v 4.0

(license terms available at: https://raw.githubusercontent.com/Microsoft/visualfsharp/master/License.txt)

Copyright (c) Microsoft Corporation. All rights reserved.

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
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.

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 BuildServer 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.
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](http://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](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](http://go.microsoft.com/fwlink/?LinkId=66406).

12) windows-base Docker image—dotnet-sdk

(available at [https://github.com/dotnet/core/blob/master/LICENSE.TXT](https://github.com/dotnet/core/blob/master/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.
# AWS CodeBuild User Guide

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  
- **Latest documentation update:** March 20, 2020

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New topics (p. 388)</td>
<td>CodeBuild now supports sharing build project and report group resources. For more information, see <a href="#">Working with shared projects</a> and <a href="#">Working with shared report groups</a>.</td>
<td>December 13, 2019</td>
</tr>
<tr>
<td>New and updated topics (p. 388)</td>
<td>CodeBuild now supports test reporting during the run of a build project. For more information, see <a href="#">Working with test reporting</a>, <a href="#">Create a test report</a>, and <a href="#">Create a test report using the AWS CLI sample</a>.</td>
<td>November 25, 2019</td>
</tr>
<tr>
<td>Updated topic (p. 388)</td>
<td>CodeBuild now supports Linux GPU and Arm environment types, and the <code>2xlarge</code> compute type. For more information, see <a href="#">Build environment compute types</a>.</td>
<td>November 19, 2019</td>
</tr>
<tr>
<td>Updated topics (p. 388)</td>
<td>CodeBuild now supports build numbers on all builds, exporting environment variables, and AWS Secrets Manager integration. For more information, see <a href="#">Exported variables</a> and <a href="#">Secrets Manager in Buildspec syntax</a>.</td>
<td>November 6, 2019</td>
</tr>
<tr>
<td>New topic (p. 388)</td>
<td>CodeBuild now supports notification rules. You can use notification rules to notify users of important changes in build projects. For more information, see <a href="#">Create a notification rule</a>.</td>
<td>November 5, 2019</td>
</tr>
<tr>
<td>Updated topics (p. 388)</td>
<td>CodeBuild now supports the Android version 29 and Go version 1.13 runtimes. For more information, see <a href="#">Docker images</a>.</td>
<td>September 10, 2019</td>
</tr>
<tr>
<td>Updated topics (p. 388)</td>
<td>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.</td>
<td>August 16, 2019</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Updated topic (p. 388)</td>
<td>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.</td>
<td>March 8, 2019</td>
</tr>
<tr>
<td>New topic (p. 388)</td>
<td>CodeBuild now supports local caching. You can specify local caching in one or more of four modes when you create a build. For more information, see Build caching in CodeBuild.</td>
<td>February 21, 2019</td>
</tr>
<tr>
<td>New topics (p. 388)</td>
<td>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.</td>
<td>February 8, 2019</td>
</tr>
<tr>
<td>New topic (p. 388)</td>
<td>The CodeBuild User Guide now shows how to use CodeBuild with a proxy server. For more information, see Use CodeBuild with a proxy server.</td>
<td>February 4, 2019</td>
</tr>
<tr>
<td>Updated topics (p. 388)</td>
<td>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.</td>
<td>January 24, 2019</td>
</tr>
<tr>
<td>Support for private Docker registries (p. 388)</td>
<td>CodeBuild now supports using a Docker image that is stored in a private registry as your runtime environment. For more information, see Private registry with AWS Secrets Manager sample.</td>
<td>January 24, 2019</td>
</tr>
<tr>
<td>Updated topic (p. 388)</td>
<td>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.</td>
<td>December 6, 2018</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Updated topic (p. 388)</td>
<td>CodeBuild now supports new build metrics that measure the duration of each phase in a build. For more information, see CodeBuild CloudWatch metrics.</td>
<td>November 15, 2018</td>
</tr>
<tr>
<td>VPC endpoint policy topic (p. 388)</td>
<td>Amazon VPC endpoints for CodeBuild now support policies. For more information, see Create a VPC endpoint policy for CodeBuild.</td>
<td>November 9, 2018</td>
</tr>
<tr>
<td>Updated content (p. 388)</td>
<td>Topics have been updated to reflect the new console experience.</td>
<td>October 30, 2018</td>
</tr>
<tr>
<td>Amazon EFS sample (p. 388)</td>
<td>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.</td>
<td>October 26, 2018</td>
</tr>
<tr>
<td>Bitbucket webhooks (p. 388)</td>
<td>CodeBuild now supports webhooks when you use Bitbucket for your repository. For more information, see Bitbucket pull request sample for CodeBuild.</td>
<td>October 2, 2018</td>
</tr>
<tr>
<td>S3 logs (p. 388)</td>
<td>CodeBuild now supports build logs in an S3 bucket. Previously, you could only build logs using CloudWatch Logs. For more information, see Create project.</td>
<td>September 17, 2018</td>
</tr>
<tr>
<td>Multiple input sources and multiple output artifacts (p. 388)</td>
<td>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.</td>
<td>August 30, 2018</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>Semantic versioning sample (p. 388)</td>
<td>The CodeBuild User Guide now has a use case-based sample that demonstrates how to use semantic versioning to create artifact names at build time. For more information, see Use semantic versioning to name build artifacts sample.</td>
<td>August 14, 2018</td>
</tr>
<tr>
<td>New static website sample (p. 388)</td>
<td>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 information, see Create a static website with build output hosted in an S3 bucket.</td>
<td>August 14, 2018</td>
</tr>
<tr>
<td>Support for overriding an artifact name with semantic versioning (p. 388)</td>
<td>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.</td>
<td>August 7, 2018</td>
</tr>
<tr>
<td>Support of unencrypted build artifacts (p. 388)</td>
<td>CodeBuild now supports builds with unencrypted build artifacts. For more information, see Create a build project (console).</td>
<td>July 26, 2018</td>
</tr>
<tr>
<td>Support for Amazon CloudWatch metrics and alarms (p. 388)</td>
<td>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.</td>
<td>July 19, 2018</td>
</tr>
<tr>
<td>Support for reporting a build's status (p. 388)</td>
<td>CodeBuild can now report the status of a build's start and completion to your source provider. For more information, see Create a build project in CodeBuild.</td>
<td>July 10, 2018</td>
</tr>
</tbody>
</table>
Earlier updates

The following table describes important changes in each release of the AWS CodeBuild User Guide before June 2018.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Windows builds</td>
<td>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 Microsoft Windows samples for CodeBuild (p. 30).</td>
<td>May 25, 2018</td>
</tr>
<tr>
<td>Support for build idempotency</td>
<td>When you run the <code>start-build</code> 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) (p. 273).</td>
<td>May 15, 2018</td>
</tr>
<tr>
<td>Support for overriding more build project settings</td>
<td>You can now override more build project settings when you create a build. The overrides are only for that build. For more</td>
<td>May 15, 2018</td>
</tr>
</tbody>
</table>
## Earlier updates

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC Endpoint support</td>
<td>You can now use VPC endpoints to improve the security of your builds. For more information, see <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/using-vpc-connections.html">Use VPC endpoints (p. 180)</a>.</td>
<td>March 18, 2018</td>
</tr>
<tr>
<td>Support of triggers</td>
<td>You can now create triggers to schedule builds at regular frequencies. For more information, see <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/triggers.html">Create AWS CodeBuild triggers (p. 249)</a>.</td>
<td>March 28, 2018</td>
</tr>
<tr>
<td>FIPS endpoints documentation</td>
<td>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 <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/specifying-endpoints.html">Specify the AWS CodeBuild endpoint (p. 356)</a>.</td>
<td>March 28, 2018</td>
</tr>
<tr>
<td>AWS CodeBuild available in Asia Pacific (Mumbai), Europe (Paris), and South America (São Paulo)</td>
<td>AWS CodeBuild is now available in the Asia Pacific (Mumbai), Europe (Paris), and South America (São Paulo) regions. For more information, see <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/about-codebuild.html">AWS CodeBuild in the Amazon Web Services General Reference</a>.</td>
<td>March 28, 2018</td>
</tr>
<tr>
<td>GitHub Enterprise support</td>
<td>CodeBuild can now build from source code stored in a GitHub Enterprise repository. For more information, see <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/sample-code/ghe-sample.html">GitHub Enterprise sample (p. 115)</a>.</td>
<td>January 25, 2018</td>
</tr>
<tr>
<td>Git clone depth support</td>
<td>CodeBuild now supports the creation of a shallow clone with a history truncated to the specified number of commits. For more information, see <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/build-project-reference.html">Create a build project (p. 215)</a>.</td>
<td>January 25, 2018</td>
</tr>
<tr>
<td>VPC support</td>
<td>VPC-enabled builds are now able to access resources inside your VPC. For more information, see <a href="https://docs.aws.amazon.com/codebuild/latest/userguide/vpc-support.html">VPC support (p. 178)</a>.</td>
<td>November 27, 2017</td>
</tr>
<tr>
<td>Dependency caching support</td>
<td>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.</td>
<td>November 27, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Build badges support</td>
<td>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 (p. 82).</td>
<td>November 27, 2017</td>
</tr>
<tr>
<td>AWS Config integration</td>
<td>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 (p. 65).</td>
<td>October 20, 2017</td>
</tr>
<tr>
<td>Automatically rebuild updated source code in GitHub repositories</td>
<td>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 GitHub pull request and webhook filter sample (p. 120).</td>
<td>September 21, 2017</td>
</tr>
<tr>
<td>New ways for storing and retrieving sensitive or large environment variables in Amazon EC2 Systems Manager Parameter Store</td>
<td>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 environment variables by using the Amazon EC2 Systems Manager Parameter Store console. For more information, see Create a build project (p. 215), Change a build project's settings (p. 252), and Run a build (p. 269).</td>
<td>September 14, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Build deletion support</td>
<td>You can now delete builds in AWS CodeBuild. For more information, see Delete builds (p. 284).</td>
<td>August 31, 2017</td>
</tr>
<tr>
<td>Updated way to retrieve sensitive or large environment variables stored in Amazon EC2 Systems Manager Parameter Store by using a buildspec</td>
<td>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 (p. 150).</td>
<td>August 10, 2017</td>
</tr>
<tr>
<td>AWS CodeBuild supports Bitbucket</td>
<td>CodeBuild can now build from source code stored in a Bitbucket repository. For more information, see Create a build project (p. 215) and Run a build (p. 269).</td>
<td>August 10, 2017</td>
</tr>
<tr>
<td>AWS CodeBuild available in US West (N. California), Europe (London), and Canada (Central)</td>
<td>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.</td>
<td>June 29, 2017</td>
</tr>
<tr>
<td>Alternate buildspec file names and locations supported</td>
<td>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 (p. 149).</td>
<td>June 27, 2017</td>
</tr>
<tr>
<td>Updated build notifications sample</td>
<td>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 (p. 85) has been updated to demonstrate this new behavior.</td>
<td>June 22, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>Docker in custom image sample added</td>
<td>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 (p. 107).</td>
<td>June 7, 2017</td>
</tr>
<tr>
<td>Fetch source code for GitHub pull requests</td>
<td>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 information, see the Source version value in Run a build (console) (p. 270) or the sourceVersion value in Run a build (AWS CLI) (p. 273).</td>
<td>June 6, 2017</td>
</tr>
<tr>
<td>Build specification version updated</td>
<td>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_variables is renamed to env, and plaintext is renamed to variables. For more information, see Build specification reference for CodeBuild (p. 149).</td>
<td>May 9, 2017</td>
</tr>
<tr>
<td>Dockerfiles for build images available in GitHub</td>
<td>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 (p. 166).</td>
<td>May 2, 2017</td>
</tr>
<tr>
<td>AWS CodeBuild available in Europe (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Sydney), and Asia Pacific (Tokyo)</td>
<td>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.</td>
<td>March 21, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>CodePipeline test action support for CodeBuild</td>
<td>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) (p. 206).</td>
<td>March 8, 2017</td>
</tr>
<tr>
<td>Buildspec files support fetching build output from within selected top-level directories</td>
<td>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 (p. 150).</td>
<td>February 8, 2017</td>
</tr>
<tr>
<td>Built-in environment variables</td>
<td>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 (p. 172).</td>
<td>January 30, 2017</td>
</tr>
<tr>
<td>AWS CodeBuild available in US East (Ohio)</td>
<td>AWS CodeBuild is now available in the US East (Ohio) region. For more information, see AWS CodeBuild in the Amazon Web Services General Reference.</td>
<td>January 19, 2017</td>
</tr>
<tr>
<td>AWS Lambda sample</td>
<td>A reference was added to a sample showing how to use CodeBuild along with Lambda, AWS CloudFormation, and CodePipeline to build and deploy a serverless application that follows the AWS Serverless Application Model (AWS SAM) standard. For more information, see the AWS Lambda sample (p. 74).</td>
<td>December 20, 2016</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Shell and command behaviors</td>
<td>CodeBuild runs each command you specify in a separate instance of a build</td>
<td>December 9, 2016</td>
</tr>
<tr>
<td>information</td>
<td>environment's default shell. This default behavior can produce some</td>
<td></td>
</tr>
<tr>
<td></td>
<td>unexpected side effects for your commands. We recommend some approaches to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>work around this default behavior if needed. For more information, see</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shells and commands in build environments (p. 172).</td>
<td></td>
</tr>
<tr>
<td>Environment variables</td>
<td>CodeBuild provides several environment variables that you can use in your</td>
<td>December 7, 2016</td>
</tr>
<tr>
<td>information</td>
<td>build commands. You can also define your own environment variables. For</td>
<td></td>
</tr>
<tr>
<td></td>
<td>more information, see Environment variables in build environments (p. 172).</td>
<td></td>
</tr>
<tr>
<td>Troubleshooting topic</td>
<td>Troubleshooting information is now available. For more information, see</td>
<td>December 5, 2016</td>
</tr>
<tr>
<td></td>
<td>Troubleshooting AWS CodeBuild (p. 359).</td>
<td></td>
</tr>
<tr>
<td>Jenkins plugin initial release</td>
<td>This is the initial release of the CodeBuild Jenkins plugin. For more</td>
<td>December 5, 2016</td>
</tr>
<tr>
<td></td>
<td>information, see Use AWS CodeBuild with Jenkins (p. 210).</td>
<td></td>
</tr>
<tr>
<td>User Guide initial release</td>
<td>This is the initial release of the CodeBuild User Guide.</td>
<td>December 1, 2016</td>
</tr>
</tbody>
</table>
AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.