AWS CodeBuild

User Guide

API Version 2016-10-06

AWS CodeBuild: User Guide

Table of Contents

What Is AWS CodeBuild?

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

AWS CodeBuild provides these benefits:

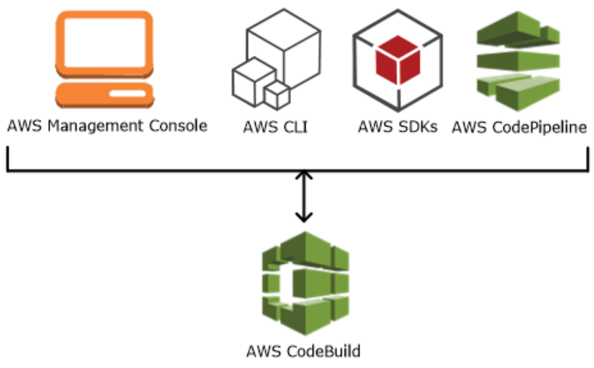
- **Fully managed** – AWS CodeBuild eliminates the need to set up, patch, update, and manage your own build servers.
- **On demand** – AWS CodeBuild scales on demand to meet your build needs. You pay only for the number of build minutes you consume.
- **Out of the box** – AWS CodeBuild provides preconfigured 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.

Topics

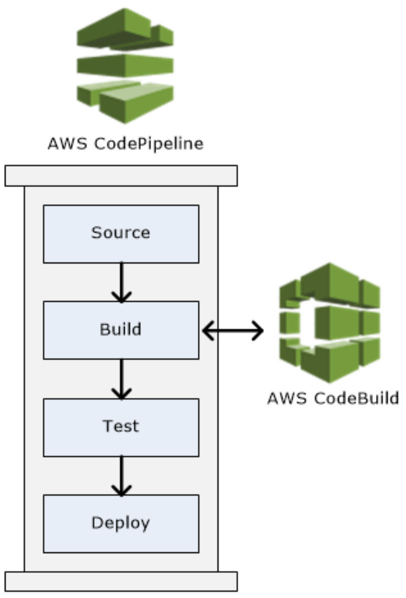
How to Run AWS CodeBuild

You can run AWS CodeBuild by using the AWS CodeBuild or AWS CodePipeline console. You can also automate the running of AWS CodeBuild by using the AWS Command Line Interface (AWS CLI) or the AWS SDKs. If an AWS SDK is not available for your programming language or platform, you can call the AWS CodeBuild HTTP API from your HTTP request/response functions.



To run AWS CodeBuild by using the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API, see Run AWS CodeBuild Directly (p. 38).

As the following diagram shows, you can add AWS CodeBuild as a build action to the build stage of a pipeline in AWS CodePipeline. AWS CodePipeline is a continuous delivery service that enables you 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 AWS CodePipeline to create a pipeline and then add an AWS CodeBuild build action, see Use AWS CodePipeline with AWS CodeBuild (p. 25). For more information about AWS CodePipeline, see the AWS CodePipeline User Guide.

Pricing for AWS CodeBuild

For information, see AWS CodeBuild Pricing.

How Do I Get Started with AWS CodeBuild?

We recommend that you complete the following steps:

1. **Learn** more about AWS CodeBuild by reading the information in Concepts (p. 4).
2. **Experiment** with AWS CodeBuild in an example scenario by following the instructions in Getting Started (p. 6).
3. **Use** AWS CodeBuild in your own scenarios by following the instructions in Plan a Build (p. 23).

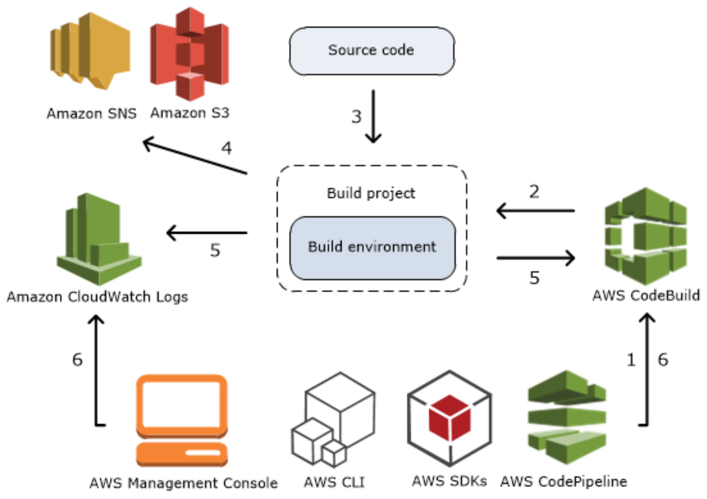
AWS CodeBuild Concepts

The following concepts are important for understanding how AWS CodeBuild works.

Topics

How AWS CodeBuild Works

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



1. As input, you must provide AWS CodeBuild with a build project. A *build project* defines how AWS CodeBuild will run a build. It includes information such as where to get the source code, the build environment to use, the 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 AWS CodeBuild uses to run a build. For more information, see:
   - Create a Build Project (p. 40)
   - Build Environment Reference (p. 82)

2. AWS CodeBuild uses the build project to create the build environment.
3. AWS CodeBuild downloads the source code into the build environment and then uses the build specification (build spec), as defined in the build project or included directly in the source code. A *build spec* is a collection of build commands and related settings, in YAML format, that AWS CodeBuild uses to run a build. For more information, see the Build Spec Reference (p. 77).
4. If the build is successful, the build environment uploads its output to an Amazon S3 bucket. The build environment can also perform tasks that you specify in the build spec (for example, sending build notifications to an Amazon SNS topic). For an example, see Build Notifications Sample (p. 118).
5. While the build is running, the build environment sends information to AWS CodeBuild and Amazon CloudWatch Logs.
6. While the build is running, you can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or the AWS CodeBuild HTTP API to get summarized build information from AWS CodeBuild and detailed build information from Amazon CloudWatch Logs. If you use AWS CodePipeline to run builds, you can get limited build information from AWS CodePipeline.

Next Steps

Now that you know more about AWS CodeBuild, we recommend that you complete the following steps:

1. **Experiment** with AWS CodeBuild in an example scenario by following the instructions in Getting Started (p. 6).
2. **Use** AWS CodeBuild in your own scenarios by following the instructions in Plan a Build (p. 23).

Getting Started with AWS CodeBuild

In this walkthrough, you will use AWS CodeBuild to build a collection of sample source code input files (which we call *build input artifacts* or *build input*) into a deployable version of the source code (which we call *build output artifact* or *build output*). Specifically, you will instruct AWS 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 walkthrough.

Important
Completing this walkthrough may 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, and CloudWatch Logs. For more information, see AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, and Amazon CloudWatch Pricing.

Topics

Step 1: Create or Use Amazon S3 Buckets to Store the Build Input and Output

To complete this walkthrough, you will need two Amazon S3 buckets:

- One of these buckets will store the build input (which we call the *input bucket*). In this walkthrough, we will name this input bucket `codebuild-`*`region-ID`*`-`*`account-ID`*`-input-bucket`, where *region-ID* represents the AWS region of the bucket, and *account-ID* represents your AWS account ID.

- The other bucket will store the build output (which we call the *output bucket*). In this walkthrough, we will name this output bucket `codebuild-region-ID-account-ID-output-bucket`.

If you chose a different name for either of these buckets, substitute it throughout this walkthrough.

These two buckets must be in the same AWS region as your builds. For example, if you instruct AWS CodeBuild to run a build in the US East (N. Virginia) region, then these buckets must be in the US Standard region (which is the AWS region that corresponds to buckets in the US East (N. Virginia) region).

To create a bucket, see Creating a Bucket in the *Amazon Simple Storage Service User Guide*.

> **Note**
> You could use a single bucket for this walkthrough. However, using two buckets can make it easier to see where the build input is coming from and where the build output is going. Although AWS CodeBuild also supports build input stored in AWS CodeCommit and GitHub repositories, this walkthrough does not show you how to use them. For more information, see Plan a Build (p. 23).

Step 2: Create the Source Code to Build

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

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

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

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

```
public class MessageUtil {
  private String message;

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

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

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

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

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

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

public class TestMessageUtil {

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

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

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

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

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

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

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

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

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

Step 3: Create the Build Spec

In this step, you will create a build specification (build spec) file. A *build spec* is a collection of build commands and related settings, in YAML format, that AWS CodeBuild uses to run a build. Without a build spec, AWS CodeBuild will not be able to successfully convert your build input into build output, nor will it be able to 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: 0.1

phases:
  install:
    commands:
      - echo Nothing to do in the install phase...
  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.

Tip
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 Build Spec Syntax (p. 77).

In this build spec declaration:

- `version` represents the version of the build spec standard being used. This build spec declaration uses the latest version, `0.1`.
- `phases` represents the build phases during which you can instruct AWS 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 additional build phase names.

  In this example, during the `build` phase, AWS CodeBuild runs the `mvn install` command. This command instructs Apache Maven to compile, test, and package the source code 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 walkthrough, the output of these `echo` commands can help you better understand how AWS CodeBuild runs commands and in which order. (Although all build phases are included in this example, you are not required to include an build phase if you do not plan to run any commands during that phase.) For each build phase included, AWS 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 AWS CodeBuild will upload to the output bucket. `files` represents the files to include in the build output. AWS CodeBuild will upload 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 will be different.

For more information, see the Build Spec Reference (p. 77).

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

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

Step 4: Add the Source Code and the Build Spec to the Input Bucket

In this step, you will 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 contained 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 AWS CodeCommit and GitHub repositories, you must store a build spec file 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 Amazon S3 buckets only, you must create a ZIP file that contains the source code and a build spec file at the root (top level) or include the build spec declaration as part of the build project definition.
For more information, see the Build Spec Reference (p. 77).

Step 5: Create the Build Project

In this step, you will create a build project that AWS CodeBuild will use to run the build. A *build project* defines how AWS CodeBuild will run a build. It includes information such as where to get the source code, the build environment to use, the 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 AWS CodeBuild uses to run a build. The build environment is expressed as a Docker image. (For more information, see the How Does a Docker Image Work? section in the Understand the Architecture topic on the Docker Docs website.) For this build environment, you'll instruct AWS CodeBuild to use a Docker image that contains a version of the Java Development Kit (JDK) and Apache Maven.

You can complete this step with the AWS CodeBuild console (p. 11) or with the AWS CLI (p. 12).

Note
You can work with AWS CodeBuild in several ways: through the AWS CodeBuild console, AWS CodePipeline, the AWS CLI, the AWS SDKs, or the AWS CodeBuild HTTP API. This walkthrough demonstrates how to use the AWS CodeBuild console and the AWS CLI. To learn how to use AWS CodePipeline, see Use AWS CodePipeline with AWS CodeBuild (p. 25). To learn how to use the AWS SDKs or the AWS CodeBuild HTTP API, see Run AWS CodeBuild Directly (p. 38).

To create the build project (console)

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In the AWS region selector, choose a region that supports AWS CodeBuild. For more information, see AWS CodeBuild in the "Regions and Endpoints" topic in the *Amazon Web Services General Reference*.
3. If a welcome page is displayed, choose **Get started**.

If a welcome page is not displayed, then on the navigation pane, choose **Build projects**, and then choose **Create project**.

4. On the **Configure your project** page, for **Project name**, type 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, substitute it throughout this walkthrough.

   Note
   On the **Configure project** page, you may see an error message similar to the following: **User: *`user-ARN`* is not authorized to perform: codebuild:ListProjects**. This is most likely because you signed in to the AWS Management Console as an IAM user that does not have sufficient permissions to use AWS CodeBuild in the console. 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:

   - 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 the AWS managed policies named **AWSCodeBuildAdminAccess**, **AmazonS3ReadOnlyAccess**, and **IAMFullAccess** 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 permission, and you are not able to 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. 101).
5. In **Source: What to build**, for **Source provider**, choose **Amazon S3**.
6. For **Bucket**, choose **codebuild-*`region-ID`*-*`account-ID`*-input-bucket**.
7. For **S3 object key**, type `MessageUtil.zip`.
8. In **Environment: How to build**, for **Environment image**, leave **Use an image managed by AWS CodeBuild** selected.
9. For **Operating system**, choose **Ubuntu**.
10. For **Runtime**, choose **Java**.
11. For **Version**, choose **aws/codebuild/java:openjdk-8**.
12. For **Build specification**, leave **Use the buildspec.yml in the source code root directory** selected.
13. In **Artifacts: Where to put the artifacts from this build project**, for **Artifacts type**, choose **Amazon S3**.
14. Leave **Artifacts name** blank.
15. For **Bucket name**, choose **codebuild-*`region-ID`*-*`account-ID`*-output-bucket**.
16. In **Service role**, leave **Create a service role in your account** selected, and leave **Role name** unchanged.
17. Choose **Continue**.
18. On the **Review** page, choose **Save**.

    Skip ahead to Step 6: Run the Build (p. 15).

To create the build project (AWS CLI)

1. Use the AWS CLI to run the `create-project` command, as follows:

```
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 substitute it for `create-project.json` throughout this walkthrough.

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

```
{
  "name": "codebuild-demo-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/java:openjdk-8",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "serviceIAMRole"
}
```

Replace *`serviceIAMRole`* with the Amazon Resource Name (ARN) of an AWS CodeBuild service role (for example, `arn:aws:iam::`*`account-ID`*`:role/`*`role-name`*). To create one, see Create an AWS CodeBuild Service Role (p. 156).

In this data:

- `name` represents a required identifier for this build project (in this example, `codebuild-demo-project`). If you use a different name, substitute it throughout this procedure. 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 will use, as specified by the Docker image repository type (in this example, `aws/codebuild/java:openjdk-8` for a Docker image in the AWS CodeBuild Docker images repository). `aws/codebuild/java` is the name of the Docker image. `openjdk-8` is the tag of the Docker image.

  To find more Docker images you can use in your scenarios, see the Build Environment Reference (p. 82).
- For `environment`, `computeType` is a required value that represents the computing resources AWS CodeBuild will use (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 walkthrough, so they are not shown here. For more information, see Create a Build Project (AWS CLI) (p. 43).

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

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

If successful, data similar to this will appear in the output.

```
{
  "project": {
    "name": "codebuild-demo-project",
    "serviceRole": "serviceIAMRole",
    "tags": [],
    "artifacts": {
      "packaging": "NONE",
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "message-util.zip"
    },
    "lastModified": 1472661575.244,
    "timeoutInMinutes": 60,
    "created": 1472661575.244,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/java:openjdk-8",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
    },
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project"
  }
}
```

- `project` represents information about this build project.
  - `tags` represents any tags that were declared.
  - `packaging` represents how the build output artifact will be stored in the output bucket. `NONE` means that a folder will be created inside of the output bucket and then the build output artifact will be stored inside of 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 AWS CodeBuild will stop 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 AWS CodeBuild to use during the build.
- `encryptionKey` represents the Amazon Resource Name (ARN) of the AWS KMS customer master key (CMK) that AWS 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 may 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 that does not have sufficient permissions to use AWS CodeBuild to create build projects. To fix this, configure the AWS CLI with credentials belonging to one of the following IAM entities:

- 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 the AWS managed policies named **AWSCodeBuildAdminAccess**, **AmazonS3ReadOnlyAccess**, and **IAMFullAccess** 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 permission, and you are not able to 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. 101).

Step 6: Run the Build

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

You can complete this step with the AWS CodeBuild console (p. 15) or the AWS CLI (p. 15).

To run the build (console)

1. If the **Build projects** page is not displayed, then in the navigation pane, choose **Build projects**.
2. In the list of build projects, choose **codebuild-demo-project**, and then choose **Start build**.
3. On the **Start new build** page, choose **Start build**.
4. Skip ahead to Step 7: View Summarized Build Information (p. 16).

To run the build (AWS CLI)

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

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

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

```
{
  "build": {
    "buildComplete": false,
    "initiator": "user-name",
    "artifacts": {
      "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-
bucket/message-util.zip"
    },
    "projectName": "codebuild-demo-project",
    "timeoutInMinutes": 60,
    "buildStatus": "IN_PROGRESS",
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/java:openjdk-8",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
    },
    "currentPhase": "SUBMITTED",
    "startTime": 1472848787.882,
    "id": "codebuild-demo-project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE",
    "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-
project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE"
  }
}
```

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

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

Step 7: View Summarized Build Information

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

You can complete this step with the AWS CodeBuild console (p. 16) or the AWS CLI (p. 17).

To view summarized build information (console)

1. If the **codebuild-demo-project:*build-ID*** page is not displayed, then in the navigation bar, choose **Build history**. Next, in the list of build projects, choose the **Build run** link that corresponds to **codebuild-demo-project** for **Project**. There should be only one matching link. (If you have

completed this walkthrough before, choose the link that corresponds to the most recent value in the **Completed** column.)

2. On the build details page, in **Phase details**, the following list of build phases should be displayed, with **Succeeded** in the **Status** column:

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

   In the page title area, a green box with **Succeeded** should be displayed.

   If you see a blue box with **In Progress** instead, choose the refresh button to see the latest progress.
3. Next to each build phase, the **Duration** value indicates how long that build phase lasted. The **Completed** value indicates when that build phase ended.

   If you expand a build phase, the phase's start and end times are displayed.

   Skip ahead to Step 8: View Detailed Build Information (p. 19).

To view summarized build information (AWS CLI)

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 will appear in the output.

```
{
  "buildsNotFound": [],
  "builds": [
    {
      "buildComplete": true,
      "phases": [
        {
          "phaseStatus": "SUCCEEDED",
          "endTime": 1472848788.525,
          "phaseType": "SUBMITTED",
          "durationInSeconds": 0,
          "startTime": 1472848787.882
        },
        ... The full list of build phases has been omitted for brevity ...
        {
          "phaseType": "COMPLETED",
          "startTime": 1472848878.079
        }
      ],
      "logs": {
        "groupName": "/aws/codebuild/codebuild-demo-project",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/
home?region=region-ID#logEvent:group=/aws/codebuild/codebuild-demo-
project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE",
        "streamName": "38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
      },
      "artifacts": {
        "md5sum": "MD5-hash",
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-
bucket/message-util.zip",
        "sha256sum": "SHA-256-hash"
      },
      "projectName": "codebuild-demo-project",
      "timeoutInMinutes": 60,
      "initiator": "user-name",
      "buildStatus": "SUCCEEDED",
      "environment": {
        "computeType": "BUILD_GENERAL1_SMALL",
        "image": "aws/codebuild/java:openjdk-8",
        "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 AWS 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), as well as `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 related build project's `packaging` value is set to `ZIP` (which you

did not set in this walkthrough) . You can use these hashes along with a checksum tool to confirm both 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, and then choose **Actions**, **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.

Step 8: View Detailed Build Information

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

You can complete this step with the AWS CodeBuild console (p. 19) or the AWS CLI (p. 19).

To view detailed build information (console)

1. With the build details page still displayed from the previous step, the last 20 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 walkthrough, most of the log events contain verbose information about AWS 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 type `"[INFO]"` in the **Filter events** box and then press Enter, only those events containing the characters `[INFO]` will be displayed. For more information, see Filter and Pattern Syntax in the *Amazon CloudWatch User Guide*.

Skip ahead to Step 9: Get the Build Output Artifact (p. 21).

To view detailed build information (AWS CLI)

1. Use your web browser to go to the `deepLink` location that appeared in the output in the previous step (for example, `https://console.aws.amazon.com/cloudwatch/home?region=`*`region-ID`*`#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE`).
2. In the CloudWatch Logs log stream, you can browse the log events. By default, only the last set of log events is displayed. To see earlier log events, scroll to the beginning of the list.
3. In this walkthrough, most of the log events contain verbose information about AWS 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 type `"[INFO]"` in the **Filter events** box and then press Enter, only those events containing the characters `[INFO]` will be 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 walkthrough.

```
...
[Container] 2016/04/15 17:49:42 Entering phase PRE_BUILD
```

```
[Container] 2016/04/15 17:49:42 Running command echo Entering pre_build
 phase...
[Container] 2016/04/15 17:49:42 Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Phase complete: PRE_BUILD Success: true
[Container] 2016/04/15 17:49:42 Entering phase BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering build phase...
[Container] 2016/04/15 17:49:42 Entering build phase...
[Container] 2016/04/15 17:49:42 Running command mvn install
[Container] 2016/04/15 17:49:44 [INFO] Scanning for projects...
[Container] 2016/04/15 17:49:44 [INFO]
[Container] 2016/04/15 17:49:44 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:44 [INFO] Building Message Utility Java Sample
 App 1.0
[Container] 2016/04/15 17:49:44 [INFO]
 ------------------------------------------------------------------------
...
[Container] 2016/04/15 17:49:55
 -------------------------------------------------------
[Container] 2016/04/15 17:49:55  T E S T S
[Container] 2016/04/15 17:49:55
 -------------------------------------------------------
[Container] 2016/04/15 17:49:55 Running TestMessageUtil
[Container] 2016/04/15 17:49:55 Inside testSalutationMessage()
[Container] 2016/04/15 17:49:55 Hi!Robert
[Container] 2016/04/15 17:49:55 Inside testPrintMessage()
[Container] 2016/04/15 17:49:55 Robert
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0,
 Skipped: 0, Time elapsed: 0.018 sec
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Results :
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0,
 Skipped: 0
...
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 [INFO] BUILD SUCCESS
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 [INFO] Total time: 11.845 s
[Container] 2016/04/15 17:49:56 [INFO] Finished at:
 2016-04-15T17:49:56+00:00
[Container] 2016/04/15 17:49:56 [INFO] Final Memory: 18M/216M
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 Phase complete: BUILD Success: true
[Container] 2016/04/15 17:49:56 Entering phase POST_BUILD
[Container] 2016/04/15 17:49:56 Running command echo Entering post_build
 phase...
[Container] 2016/04/15 17:49:56 Entering post_build phase...
[Container] 2016/04/15 17:49:56 Phase complete: POST_BUILD Success: true
[Container] 2016/04/15 17:49:57 Preparing to copy artifacts
[Container] 2016/04/15 17:49:57 Assembling file list
[Container] 2016/04/15 17:49:57 Expanding target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Found target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Creating zip artifact
```

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

Step 9: Get the Build Output Artifact

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

You can complete this step with the AWS CodeBuild console (p. 21) or the Amazon S3 console (p. 21).

To get the build output artifact (AWS CodeBuild console)

1. With the AWS CodeBuild console still open and the build details page still displayed from the previous step, expand **Build details**, and then choose the **Build 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 folder named `target`, where you will find the build output artifact file named `messageUtil-1.0.jar`.

   Skip ahead to Step 10: Clean Up (p. 21).

To get the build output artifact (Amazon S3 console)

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

Step 10: Clean Up

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

If you are using the IAM user to delete this bucket instead of an AWS root account or an administrator IAM user, then the user must have additional access permissions. (Using an AWS root account is not recommended.) Add the statement between the markers (*`### BEGIN ADDING STATEMENT HERE ###`* and *`### END ADDING STATEMENTS HERE ###`*) to an existing access policy for the user. Ellipses (`...`) are used for brevity and to help you locate where to add the statement. Do not remove any statements in the existing access policy, and do not type these ellipses into the existing policy.

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

Next Steps

In this walkthrough, 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 AWS CodeBuild in your own scenarios by following the instructions in Plan a Build (p. 23). If you don't feel ready yet, you might want to try building some of our samples. For more information, see Samples (p. 108).

Plan a Build for AWS CodeBuild

Before you run your build with AWS CodeBuild, you must answer these questions:

1. **Where is the source code located?** AWS CodeBuild currently supports building from the following source code repository providers. The source code must contain a build specification (build spec) file, or the build spec must be declared as part of a build project definition. A *build spec* is a collection of build commands and related settings, in YAML format, that AWS CodeBuild uses to run a build.

Repository provider	Required	Documentation
AWS CodeCommit	Repository name. (Optional) Commit ID associated with the source code.	See these topics in the *AWS CodeCommit User Guide*: Create an AWS CodeCommit Repository Create a Commit in AWS CodeCommit
Amazon S3	Input bucket name. Object name corresponding to the build input ZIP file that contains the source code. (Optional) Version ID associated with the build input ZIP file.	See these topics in the *Amazon S3 Getting Started Guide*: Create a Bucket Add an Object to a Bucket
GitHub	Repository name. (Optional) Commit ID associated with the source code.	See this topic on the GitHub Help website: Create a Repo

2. **Which build commands do you need to run and in what order?** By default, AWS CodeBuild downloads the build input from the provider you specify and uploads the build output to the bucket you specify. You use the build spec to instruct how to turn the downloaded build input into the expected build output. For more information, see the Build Spec Reference (p. 77).
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?

   AWS CodeBuild runs builds in build environments that use Docker images. These Docker images must be stored in a repository type supported by AWS CodeBuild. These include the AWS CodeBuild Docker image repository, Docker Hub, and Amazon EC2 Container Registry (Amazon ECR). For more information about the AWS CodeBuild Docker image repository, see Docker Images Provided by AWS CodeBuild (p. 82).
4. **Do you need AWS resources that aren't provided automatically by AWS CodeBuild? If so, which security policies will those resources need?** For example, you might need to modify the AWS CodeBuild service role to allow AWS CodeBuild to work with those resources.

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, AWS SDKs, or AWS CodeBuild HTTP API. For more information, see Run AWS CodeBuild Directly (p. 38).
- Create or identify a pipeline in AWS CodePipeline, and then add a build action that instructs AWS CodeBuild to run your build automatically. For more information, see Use AWS CodePipeline with AWS CodeBuild (p. 25).

Use AWS CodePipeline with AWS CodeBuild to Run Builds

You can automate your release process by using AWS CodePipeline to run your builds with AWS CodeBuild.

The following table lists tasks and the methods available for performing them. Using the AWS SDKs or AWS CodePipeline HTTP API to accomplish these tasks is outside the scope of this topic.

Task	**Available approaches**	**Approaches described in this topic**
Create a continuous delivery (CD) pipeline with AWS CodePipeline that automates builds with AWS CodeBuild	AWS CodePipeline console AWS CLI AWS SDKs AWS CodePipeline HTTP API	AWS CodePipeline console AWS CLI You can adapt the information in this topic to use the AWS SDKs or AWS CodePipeline HTTP API. For more information, reference the create pipeline action documentation for your programming language through the SDKs section of *Tools for Amazon Web Services* or see `CreatePipeline` in the *AWS CodePipeline API Reference*.
Add build automation with AWS CodeBuild to an existing pipeline in AWS CodePipeline	AWS CodePipeline console AWS CLI AWS SDKs AWS CodePipeline HTTP API	AWS CodePipeline console For the AWS CLI, you can adapt the information in this topic to create a pipeline that contains an AWS CodeBuild build action. For more information, see Edit a Pipeline (AWS CLI) in the *AWS CodePipeline User Guide*. You can adapt the information in this topic to use the AWS SDKs or AWS CodePipeline HTTP API. 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

1. Answer the questions in Plan a Build (p. 23).
2. If you are using an IAM user to access AWS 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 enables the user to create the pipeline in AWS 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 an AWS CodePipeline service role, if you do not already have one available in your AWS account. This service role enables AWS CodePipeline to interact with other AWS services, including AWS CodeBuild, on your behalf. For example, to create an AWS CodePipeline service role by using the AWS CLI, run the IAM `create-role` command:

   For Linux, OS X, or Unix:

   ```
   aws iam create-role --role-name AWS-CodePipeline-
   CodeBuild-Service-Role --assume-role-policy-document
    '{"Version":"2012-10-17","Statement":{"Effect":"Allow","Principal":
   {"Service":"codepipeline.amazonaws.com"},"Action":"sts:AssumeRole"}}'
   ```

   For Windows:

   ```
   aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-
   Role --assume-role-policy-document "{\"Version\":\"2012-10-17\",
   \"Statement\":{\"Effect\":\"Allow\",\"Principal\":{\"Service\":
   \"codepipeline.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}}"
   ```

   Note
   The IAM entity that creates this AWS CodePipeline service role must have permission in IAM to create service roles.
4. After you create an AWS CodePipeline service role or identify an existing one, you must add policy statements to it:

   - Add policy statements to the AWS CodePipeline service role as described in Edit a Policy for an IAM Service Role in the *AWS CodePipeline User Guide*.
   - Add the following policy statement (between `### BEGIN ADDING STATEMENT HERE ###` and `### END ADDING STATEMENT HERE ###`) to the AWS CodePipeline service role. This policy statement enables AWS CodePipeline to interact with AWS CodeBuild on your behalf. Ellipses (`...`) are used for brevity and to help you locate where to add the statement. Do not remove any statements, and do not type these ellipses into the existing policy.

   ```
   {
   ```

```
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
    },
    ### END ADDING STATEMENT HERE ###
    {
      "Action": [
        "s3:GetObject",
        ...
    },
    {
      "Action": [
        "s3:PutObject",
        ...
    },
    {
      "Action": [
        "codecommit:CancelUploadArchive",
        ...
    },
    ...
  ],
  "Version": "2012-10-17"
}
```

Note
The IAM entity that modifies this AWS CodePipeline service role must have permission in IAM to modify service roles.

5. Create and upload the source code to a repository type supported by AWS CodeBuild and AWS CodePipeline, such as AWS CodeCommit, Amazon S3, or GitHub. Make sure the source code contains a build spec file (or you can declare a build spec when you define a build project later in this topic), which provides instructions for building the source code. For more information, see the Build Spec Reference (p. 77).

   Important
   If you plan to use the pipeline to deploy the built source code, then the build output artifact must be compatible with the deployment system you will use.

   - For AWS CodeDeploy, see the AWS CodeDeploy Sample (p. 115) in this guide and see Prepare a Revision for AWS CodeDeploy in the *AWS CodeDeploy User Guide*.
   - For AWS Elastic Beanstalk, see the Elastic Beanstalk Sample (p. 127) in this guide and see Create an Application Source Bundle in the *AWS Elastic Beanstalk Developer Guide*.
   - For AWS OpsWorks, see Application Source and Using AWS CodePipeline with AWS OpsWorks in the *AWS OpsWorks User Guide*.

Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console)

1. Complete the steps in Prerequisites (p. 26).
2. Open the AWS CodePipeline console, at https://console.aws.amazon.com/codepipeline.

   You need to have already signed in to the AWS Management Console by using one of the following:

   - Your AWS root account. This is not recommended. For more information, see The 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:

   ```
   codepipeline:*
   iam:ListRoles
   iam:PassRole
   s3:CreateBucket
   s3:GetBucketPolicy
   s3:GetObject
   s3:ListAllMyBuckets
   s3:ListBucket
   s3:PutBucketPolicy
   codecommit:ListBranches
   codecommit:ListRepositories
   codedeploy:GetApplication
   codedeploy:GetDeploymentGroup
   codedeploy:ListApplications
   codedeploy:ListDeploymentGroups
   elasticbeanstalk:DescribeApplications
   elasticbeanstalk:DescribeEnvironments
   lambda:GetFunctionConfiguration
   lambda:ListFunctions
   opsworks:DescribeStacks
   opsworks:DescribeApps
   opsworks:DescribeLayers
   ```
3. In the AWS region selector, choose the region where your pipeline and related AWS resources are located. This region must also support AWS CodeBuild. For more information, see AWS CodeBuild in the "Regions and Endpoints" topic in the *Amazon Web Services General Reference*.
4. Create a pipeline as follows:

   If a welcome page is displayed, choose **Get started**.

   If an **All Pipelines** page is displayed, choose **Create pipeline**.
5. On the **Step 1: Name** page, for **Pipeline name**, type a name for the pipeline; for example, **`CodeBuildDemoPipeline`**. If you choose a different name, substitute it throughout this procedure. Choose **Next step**.
6. On the **Step 2: Source** page, for **Source provider**, do one of the following:

   - If your source code is stored in an Amazon S3 bucket, choose **Amazon S3**. For **Amazon S3 location**, type the path to the source code, using the format `s3://`*`bucket-name`*`/`*`path`*`/`*`to`*`/`*`file-name`*`.zip`. Choose **Next step**.

- If your source code is stored in an AWS CodeCommit repository, choose **AWS 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 represents the version of the source code you want to build. Choose **Next step**.
- 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 represents the version of the source code you want to build. Choose **Next step**.

7. On the **Step 3: Build** page, for **Build provider**, choose **AWS CodeBuild**.
8. If you already have a build project you want to use, choose **Select an existing build project**. For **Project name**, choose the name of the build project, and then skip ahead to step 17 in this procedure.

   Important
   If you choose an existing build project, it must have build output artifact settings already defined (even though AWS CodePipeline will override them). For more information, see the description of **Artifacts: Where to put the artifacts from this build project** in Create a Build Project (Console) (p. 40) or Change a Build Project's Settings (Console) (p. 54).
9. Choose **Create a new build project**.
10. For **Project name**, type a name for this build project. Build project names must be unique across each AWS account.
11. (Optional) Type a description in the **Description** box.
12. For **Environment image**, do one of the following:

    - To use a build environment based on a Docker image that is managed by AWS CodeBuild, choose **Use an image managed by AWS CodeBuild**. Make your selections from the **Operating system**, **Runtime**, and **Version** drop-down lists. For more information, see Docker Images Provided by AWS CodeBuild (p. 82).
    - To use a build environment based on a Docker image in an Amazon ECR repository in your AWS account, choose **Specify a Docker image**. For **Custom image type**, choose **Amazon ECR**. Use the **Amazon ECR repository** and **Amazon ECR image** drop-down lists to specify the desired 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 **Specify a Docker image**. For **Custom image type**, choose **Other**. In the **Custom image ID** box, type the Docker image ID, using the format *`docker-repo-name`*`/`*`docker-image-name`*`:`*`tag`*.
13. For **Build specification**, do one of the following:

    - If your source code includes a build spec file, choose **Use the buildspec.yml in the source code root directory**.
    - If your source code does not include a build spec file, choose **Insert build commands**. For **Build command**, type the commands you want to run during the build phase in the build environment; for multiple commands, separate each command with `&&`. For **Output files**, type the paths to the build output files in the build environment that you want to send to AWS CodePipeline; for multiple files, separate each file path with a comma. For more information, see the tooltips in the console.
14. For **AWS CodeBuild service role**, do one of the following:

    - If you do not have an AWS CodeBuild service role in your AWS account, choose **Create a service role in your account**. In the **Role name** box, type a name for the service role or leave the suggested name. (Service role names must be unique across your AWS account.)

      Note
      If you use the console to create an AWS CodeBuild service role, by default this service role works with this build project only. If you use the console to associate this service

role with another build project, this role will be updated to work with the other build project. A single AWS CodeBuild service role can work with up to ten build projects.

- If you have an AWS CodeBuild service role in your AWS account, choose **Choose an existing service role from your account**. In the **Role name** box, choose the name of the service role.

15. Expand **Advanced**.

    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 row**.
16. Choose **Save build project**.
17. After the build project is saved, choose **Next step**.
18. On the **Step 4: Beta** page, do one of the following:

    - If you do not want to deploy the build output artifact, for **Deployment provider**, choose **No Deployment**.
    - If you want to deploy the build output artifact, for **Deployment provider**, choose a deployment provider, and then specify the settings when prompted.
19. Choose **Next step**.
20. On the **Step 5: Service Role** page, for **Role name**, choose the AWS CodePipeline service role you created or identified as part of this topic's prerequisites.

    Do not use this page to create a AWS CodePipeline service role. If you do, the service role will not have the permissions required to work with AWS CodeBuild.
21. Choose **Next step**.
22. On the **Step 6: Review** page, choose **Create pipeline**.
23. After the pipeline runs successfully, you can get the build output artifact. With the pipeline displayed in the AWS CodePipeline console, in the **Build** action, rest your mouse pointer on the tooltip. Make a note of the value for **Output artifact** (for example, **MyAppBuild**).

    Tip
    You can also get the build output artifact by choosing the **Build artifacts** link on the build details page in the AWS CodeBuild console. To get to this page, skip the rest of the steps in this procedure, and see View Build Details (Console) (p. 65).
24. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
25. In the list of buckets, open the bucket used by the pipeline. The name of the bucket should follow the format `codepipline-`*`region-ID`*`-`*`random-number`*. You can use the AWS CLI to run the AWS CodePipeline `get-pipeline` command to get the name of the bucket, where *`my-pipeline-name`* is the display name of your pipeline:

    ```
    aws codepipeline get-pipeline --name my-pipeline-name
    ```

    In the output, the `pipeline` object contains an `artifactStore` object, which contains a `location` value with the name of the bucket.
26. Open the folder that matches the name of your pipeline (depending on the length of the pipeline's name, the folder name might be truncated), and then open the folder matching the value for **Output artifact** that you noted in step 23 of this procedure.
27. 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 will be in the extracted contents of the file.

28. If you instructed AWS 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 AWS CodeBuild (AWS CLI)

1. Complete the steps in Prerequisites (p. 26).
2. Create or identify a build project in AWS CodeBuild. For more information, see Create a Build Project (p. 40).

   Important
   The build project must define build output artifact settings (even though AWS CodePipeline will override them). For more information, see the description of `artifacts` in Create a Build Project (AWS CLI) (p. 43).
3. 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*.
4. 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 Amazon S3 input bucket and a build action that uses AWS CodeBuild:

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

In this JSON-formatted data:

- The value of `roleArn` must match the ARN of the AWS 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 Amazon S3 bucket. For settings for other source code repository types, see the AWS CodePipeline Pipeline Structure Reference in the *AWS CodePipeline User Guide*.
- The value of `ProjectName` is the name of the AWS CodeBuild build project you created earlier in this procedure.
- The value of `location` is the name of the Amazon S3 bucket used by this pipeline. For more information, see Create a Policy for an Amazon S3 Bucket to Use as the Artifact Store for AWS 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*.

5. Switch to the folder that contains the JSON file, and then run the AWS CodePipeline `create-pipeline` command, specifying the file name:

   ```
   aws codepipeline create-pipeline --cli-input-json file://create-
   pipeline.json
   ```

   Note
   You must create the pipeline in an AWS region that supports AWS CodeBuild. For more information, see AWS CodeBuild in the "Regions and Endpoints" topic in the *Amazon Web Services General Reference*.

   The JSON-formatted data appears in the output, and AWS CodePipeline creates the pipeline.

6. To get information about the pipeline's status, run the AWS CodePipeline `get-pipeline-state` command, specifying the name of the pipeline:

   ```
   aws codepipeline get-pipeline-state --name my-pipeline-name
   ```

   In the output, look for information that confirms the build was successful. Ellipses (`...`) are used to show data that has been omitted for brevity.

   ```
   {
     ...
     "stageStates": [
       ...
       {
         "actionStates": [
           {
             "actionName": "AWS CodeBuild",
             "latestExecution": {
               "status": "SUCCEEDED",
               ...
             },
             ...
           }
         ]
       }
     ]
   }
   ```

   If you run this command too early, you might not see any information about the build action. You might need to run this command multiple times until the pipeline has finished running the build action.

7. 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/.

   Tip
   You can also get the build output artifact by choosing the **Build artifacts** link on the related build details page in the AWS CodeBuild console. To get to this page, skip the rest of the steps in this procedure, and see View Build Details (Console) (p. 65).

8. In the list of buckets, open the bucket used by the pipeline. The name of the bucket should follow the format `codepipline-region-ID-random-number`. You can get the bucket name from the `create-pipeline.json` file or you can run the AWS 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.

9. Open the folder that matches the name of your pipeline (for example, *`my-pipeline-name`*).
10. In that folder, open the folder named `default`.
11. 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 will be in the extracted contents of the file.

Add an AWS CodeBuild Build Action to a Pipeline (AWS CodePipeline Console)

1. Open the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline.

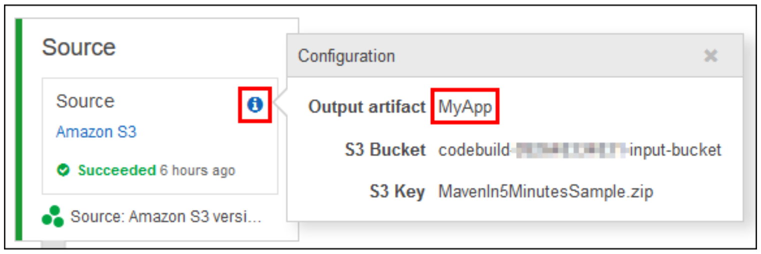
   You should have already signed in to the AWS Management Console by using one of the following:

   - Your AWS root account. This is not recommended. For more information, see The 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
   s3:ListBucket
   s3:PutBucketPolicy
   codecommit:ListBranches
   codecommit:ListRepositories
   codedeploy:GetApplication
   codedeploy:GetDeploymentGroup
   codedeploy:ListApplications
   codedeploy:ListDeploymentGroups
   elasticbeanstalk:DescribeApplications
   elasticbeanstalk:DescribeEnvironments
   lambda:GetFunctionConfiguration
   lambda:ListFunctions
   opsworks:DescribeStacks
   opsworks:DescribeApps
   opsworks:DescribeLayers
   ```

2. In the AWS region selector, choose the region where your pipeline is located. This region must also support AWS CodeBuild. For more information, see AWS CodeBuild in the "Regions and Endpoints" topic in the *Amazon Web Services General Reference*.

3. On the **All Pipelines** page, choose the name of the pipeline.
4. On the pipeline details page, in the **Source** action, rest your mouse pointer on the tooltip. Make a note of the value for **Output artifact** (for example, **MyApp**):

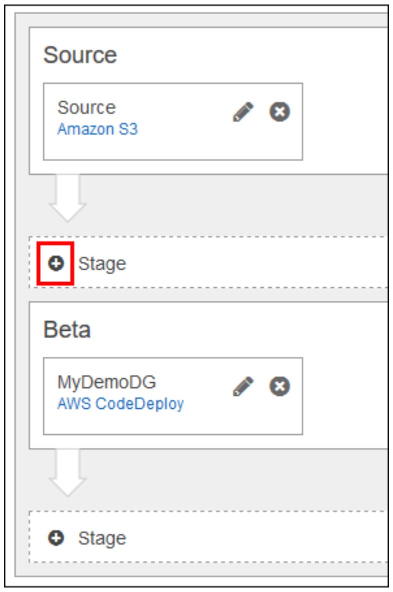


Output artifact MyApp
S3 Bucket codebuild-[redacted]-input-bucket
S3 Key MavenIn5MinutesSample.zip

Note
This procedure assumes you want to add a build action inside of a build stage between the **Source** and **Beta** stages. If you want to add the build action somewhere else, rest your mouse pointer 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**.

5. Choose **Edit**.
6. Between the **Source** and **Beta** stages, choose the add symbol (**+**) next to **Stage**.

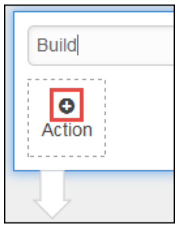
Note
This procedure assumes you want to add a new build stage to your pipeline. To add a build action to an existing stage, choose the edit (pencil) icon in the existing stage, and then skip to step 8 of this procedure.
This procedure also assumes you want to add a build stage between the **Source** and **Beta** stages. To add the build stage somewhere else, choose the add symbol in the desired place.



7. For **Enter stage name**, type the name of the build stage (for example, **`Build`**). If you choose a different name, use it throughout this procedure.
8. Inside of the selected stage, choose the add symbol (**+**) next to **Action**.

Note
This procedure assumes you want to add the build action inside of a build stage. To add the build action somewhere else, choose the add symbol in the desired place. You might

first need to choose the edit (pencil) icon in the existing stage where you want to add the build action.



9. In the **Add action** pane, for **Action category**, choose **Build**.
10. In **Build actions**, for **Action name**, type a name for the action (for example, **`AWS CodeBuild`**). If you choose a different name, use it throughout this procedure.
11. For **Build provider**, choose **AWS CodeBuild**.
12. If you already have a build project in AWS CodeBuild, choose **Select an existing build project**. For **Project name**, choose the name of the build project, and then skip to step 21 of this procedure.

    Important
    If you choose an existing build project, it must have build output artifact settings already defined (even though AWS CodePipeline will override them). For more information, see the description of **Artifacts: Where to put the artifacts from this build project** in Create a Build Project (Console) (p. 40) or Change a Build Project's Settings (Console) (p. 54).
13. Choose **Create a new build project**.
14. For **Project name**, type a name for this build project. Build project names must be unique across each AWS account.
15. (Optional) Type a description in the **Description** box.
16. For **Environment image**, do one of the following:

    - To use a build environment based on a Docker image that is managed by AWS CodeBuild, choose **Use an image managed by AWS CodeBuild**. Make your selections from the **Operating system**, **Runtime**, and **Version** drop-down lists. For more information, see Docker Images Provided by AWS CodeBuild (p. 82).
    - To use a build environment based on a Docker image in an Amazon ECR repository in your AWS account, choose **Specify a Docker image**. For **Custom image type**, choose **Amazon ECR**. Use the **Amazon ECR repository** and **Amazon ECR image** drop-down lists to specify the desired Amazon ECR repository and Docker image in that repository.
    - To use a build environment based on a Docker image in Docker Hub, choose **Specify a Docker image**. For **Custom image type**, choose **Other**. In the **Custom image ID** box, type the Docker image ID, using the format *`docker-repo-name`*`/`*`docker-image-name`*`:`*`tag`*.
17. For **Build specification**, do one of the following:

    - If your source code includes a build spec file, choose **Use the buildspec.yml in the source code root directory**.
    - If your source code does not include a build spec file, choose **Insert build commands**. For **Build command**, type the commands you want to run during the build phase in the build environment; for multiple commands, separate each command with `&&`. For **Output files**, type the paths to the build output files in the build environment that you want to send to AWS CodePipeline; for multiple files, separate each file path with a comma. For more information, see the tooltips in the console.
18. For **AWS CodeBuild service role**, do one of the following:

- If you do not have an AWS CodeBuild service role in your AWS account, choose **Create a service role in your account**. In the **Role name** box, type a name for the service role or leave the suggested name. (Service role names must be unique across your AWS account.)

  Note
  If you use the console to create an AWS CodeBuild service role, by default this service role works with this build project only. If you use the console to associate this service role with another build project, this role will be updated to work with the other build project. A single AWS CodeBuild service role can work with up to ten build projects.
- If you have an AWS CodeBuild service role in your AWS account, choose **Choose an existing service role from your account**. In the **Role name** box, choose the name of the service role.

19. Expand **Advanced**.

    To specify a build timeout other than 60 minutes (the default), use the **hours** and **minutes** boxes to specify 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 row**.
20. Choose **Save build project**.
21. For **Input artifact #1**, type the value of **Output artifact** that you noted in step 4 of this procedure.
22. For **Output artifact #1**, type a name for the output artifact (for example, **`MyAppBuild`**).
23. Choose **Add action**.
24. Choose **Save pipeline changes**, and then choose **Save and continue**.
25. Choose **Release change**.
26. After the pipeline runs successfully, you can get the build output artifact. With the pipeline displayed in the AWS CodePipeline console, in the **Build** action, rest your mouse pointer on the tooltip. Make a note of the value for **Output artifact** (for example, **MyAppBuild**).

    Tip
    You can also get the build output artifact by choosing the **Build artifacts** link on the build details page in the AWS CodeBuild console. To get to this page, see View Build Details (Console) (p. 65), and then skip to step 31 of this procedure.
27. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
28. In the list of buckets, open the bucket used by the pipeline. The name of the bucket should follow the format `codepipline-`*`region-ID`*`-`*`random-number`*. You can use the AWS CLI to run the AWS CodePipeline `get-pipeline` command to get the name of the bucket:

    ```
    aws codepipeline get-pipeline --name my-pipeline-name
    ```

    In the output, the `pipeline` object contains an `artifactStore` object, which contains a `location` value with the name of the bucket.
29. Open the folder that matches the name of your pipeline (depending on the length of the pipeline's name, the folder name might be truncated), and then open the folder matching the value for **Output artifact** that you noted in step 26 of this procedure.
30. 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 will be in the extracted contents of the file.
31. If you instructed AWS CodePipeline to deploy the build output artifact, use the deployment provider's instructions to get to the build output artifact on the deployment targets.

Run AWS CodeBuild Directly

To set up, run, and monitor builds directly with AWS CodeBuild, you can use the AWS CodeBuild console, AWS CLI, AWS SDK, or AWS CodeBuild HTTP API.

Not what you're looking for? To use AWS CodePipeline to run AWS CodeBuild, see Use AWS CodePipeline with AWS CodeBuild (p. 25).

Topics

Prerequisites

Answer the questions in Plan a Build (p. 23).

Run AWS CodeBuild Directly (Console)

1. Create the build project. For information, see Create a Build Project (Console) (p. 40).
2. Run the build. For information, see Run a Build (Console) (p. 60).
3. Get information about the build. For information, see View Build Details (Console) (p. 65).

Run AWS CodeBuild Directly (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 90).

1. Create the build project. For information, see Create a Build Project (AWS CLI) (p. 43).
2. Run the build. For information, see Run a Build (AWS CLI) (p. 61).
3. Get information about the build. For information, see View Build Details (AWS CLI) (p. 65).

Run AWS CodeBuild Directly (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

Run AWS CodeBuild Directly (HTTP API)

1. Create the build project. For information, see Create a Build Project (HTTP API) (p. 48).
2. Run the build. For information, see Run a Build (HTTP API) (p. 64).
3. Get information about the build. For information, see View Build Details (HTTP API) (p. 66).

Working with Build Projects in AWS CodeBuild

A *build project* defines how AWS CodeBuild will run a build. It includes information such as where to get the source code, the build environment to use, the build commands to run, and where to store the build output.

You can perform these tasks when working with build projects:

Topics

Create a Build Project in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API to create a build project.

Topics

Prerequisites

Answer the questions in Plan a Build (p. 23).

Create a Build Project (Console)

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

2. If a welcome page is displayed, choose **Get started**.

   If a welcome page is not displayed, on the navigation pane, choose **Build projects**, and then choose **Create project**.
3. On the **Configure your project** page, for **Project name**, type a name for this build project. Build project names must be unique across each AWS account.
4. (Optional) Choose **Add description**, and type a description in the **Description** box.
5. In **Source: What to build**, for **Source provider**, choose the source code provider type, and then do one of the following:

   - If you chose **Amazon S3**, then for **Bucket**, choose the name of the input bucket that contains the source code. For **S3 object key**, type the name of the ZIP file that contains the source code.
   - If you chose **AWS CodeCommit**, then for **Repository**, choose the name of the repository.
   - If you chose **GitHub**, follow the instructions to connect (or reconnect) with GitHub. For **Repository**, choose the name of the repository that contains the source code.
6. In **Environment: How to build**, for **Environment image**, do one of the following:

   - To use a Docker image managed by AWS CodeBuild, choose **Use an image managed by AWS CodeBuild**, and then make selections from **Operating system**, **Runtime**, and **Version**.
   - To use another Docker image, choose **Specify a Docker image**. For **Custom image type**, choose **Other** or **Amazon ECR**. If you choose **Other**, then for **Custom image ID**, type the name and tag of the Docker image in Docker Hub, using the format `repository-name/image-name:image-tag`. If you choose **Amazon ECR**, then use **Amazon ECR repository** and **Amazon ECR image** to choose the Docker image in your AWS account.
7. Do one of the following:

   - If your source code includes a build spec file, choose **Use the buildspec.yml in the source code root directory**.
   - If your source code does not include a build spec file, choose **Insert build commands**. For **Build command**, type the commands you want to run in the `build` phase only. For multiple commands, separate each command by `&&`, for example `mvn test && mvn package`. To run commands in other phases, or if you have a particularly 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 Build Spec Reference (p. 77).
8. In **Artifacts: Where to put the artifacts from this build project**, for **Artifacts type**, do one of the following:

   - If you do not want to create any build output artifacts, choose **No artifacts**. You may want to do this, for example, if you only running build tests or you want to push a Docker image to an Amazon ECR repository.
   - To store the build output in an Amazon S3 bucket, choose **Amazon S3**. Then do the following:
     - Leave **Artifacts name** blank if you want to use your project name for the build output ZIP file or folder. Otherwise, to specify a name for the build output ZIP file or folder, type the name in the **Artifacts name** box. (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.)
     - For **Bucket name**, choose the name of the output bucket.
     - If you chose **Insert build commands** previously in this procedure, then for **Output files**, type 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 Build Spec Syntax (p. 77).
9. In **Service role**, do one of the following:

- If you do not have an AWS CodeBuild service role, choose **Create a new role in your account**. In **Role name**, accept the default name or type your own. Service role names must be unique across your AWS account.
- If you have an AWS CodeBuild service role, choose **Choose an existing role from your account**. In **Role name**, choose the service role.

  Note
  When you use the console to create or update a build project, you can create an AWS 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 will be updated to work with the other build project. A service role can work with up to ten build projects.

10. Expand **Show advanced settings**.

    Note
    If you arrived at this page by choosing **Get started** from a welcome page, then the **Show advanced settings** section will not be displayed. Skip to step 17 of this procedure.
    For information about changing default settings, see Change a Build Project's Settings (Console) (p. 54).
11. (Optional) For **Timeout**, specify a value between 5 minutes and 480 minutes (8 hours) after which AWS CodeBuild will stop the build if it is not complete. If **hours** and **minutes** are left blank, the default value of 60 minutes will be used.
12. (Optional) For **Encryption key**, do one of the following:
    - To use the AWS-managed customer master 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**, type the ARN of the customer-managed CMK. Use the format `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`*.
13. (Optional) If you chose **Amazon S3** for **Artifacts type** earlier in this procedure, then for **Artifacts packaging**, do one of the following:
    - To have AWS CodeBuild create a ZIP file containing the build output, choose **Zip**.
    - To have AWS CodeBuild create a folder containing the build output, choose **None**. (This is the default.)
14. For **Compute type**, choose one of the available options.
15. (Optional) For **Environment variables**, type the name and value of environment variables for builds to use. Use **Add row** to add an environment variable.

    Important
    Any environment variables you set will 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` will be 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 `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` will be replaced by the literal value `$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 build spec declaration takes lowest precedence.

16. (Optional) For **Tags**, type the name and value of any tags you want supporting AWS services to use. Use **Add row** to add a tag. You can add up to 50 tags.
17. Choose **Continue**.
18. On the **Review** page, do one of the following:

    - To run a build, choose **Save and build**.
    - To finish creating the build project without running a build, choose **Save**.

Create a Build Project (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 90).

1. Run the `create-project` command:

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

JSON-formatted data will appear 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. Then modify the copied data as follows, and save your results:

```
{
  "name": "project-name",
  "description": "description",
  "source": {
    "type": "source-type",
    "location": "source-location",
    "buildspec": "buildspec",
    "auth": {
      "type": "auth-type",
      "resource": "resource"
  },
  "artifacts": {
    "type": "artifacts-type",
    "location": "artifacts-location",
    "path": "path",
    "namespaceType": "namespaceType",
    "name": "artifacts-name",
    "packaging": "packaging"
  },
  "environment": {
    "type": "environment-type",
    "image": "image",
    "computeType": computeType,
    "environmentVariables": [
      {
        "name": "environmentVariable-name",
        "value": "environmentVariable-value"
      }
    ]
  },
  "serviceRole": "serviceRole",
  "timeoutInMinutes": "timeoutInMinutes",
  "encryptionKey": "encryptionKey",
  "tags": [
    {
      "key": "tag-key",
      "value": "tag-value"
    }
  ]
}
```

Replace the following:

- *`project-name`*: Required value. The name for this build project. This name must be unique across all of the build projects in your AWS account.
- *`description`*: Optional value. The description for this build project.
- For the required `source` object, information about this build project's source code settings. These settings include the following:
  - *`source-type`*: Required value. The type of repository that contains the source code to build. Valid values include `CODECOMMIT`, `CODEPIPELINE`, `GITHUB`, and `S3`.
  - *`source-location`*: Required value (unless you set *`source-type`* to `CODEPIPELINE`). The location of the source code for the specified repository type:
    - For AWS CodeCommit, the HTTPS clone URL to the repository that contains the source code and the build spec (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 build spec (for example, *`bucket-name`*`/`*`object-name`*`.zip`). This assumes that the ZIP file is in the root of the build input bucket. (If the ZIP file is in a folder inside of the bucket, use for example *`bucket-name`*`/`*`path`*`/`*`to`*`/`*`object-name`*`.zip` instead.)
    - For GitHub, the HTTPS clone URL, including the user name and personal access token, to the repository that contains the source code and the build spec (for example, `https://`*`login-user-name`*`:`*`personal-access-token`*`@github.com/`*`repo-owner-name`*`/`*`repo-name`*`.git`). For more information, see Creating an Access Token for Command-Line Use on the GitHub Help website.
    - For AWS CodePipeline, do not specify a `location` value for `source`. It will be ignored by AWS CodePipeline because when you create a pipeline in AWS CodePipeline, you specify the source code location in the Source stage of the pipeline.
  - *`buildspec`*: Optional value. The build specification definition or file to use. If this value is set, it must be a single string that conforms to both the build spec and YAML syntaxes. If this value is not provided or is set to an empty string, then the source code must contain a `buildspec.yml` file in its root directory. For more information, see the Build Spec Reference (p. 77).
  - *`auth`*: This object is used only by the AWS CodeBuild console. Do not specify values for *`auth-type`* or *`resource`*.
- For the required `artifacts` object, information about this build project's output artifact settings. These settings include the following:
  - *`artifacts-type`*: Required value. The type of build output artifact. Valid values include `CODEPIPELINE`, `NO_ARTIFACTS`, and `S3`.
  - *`artifacts-location`*: Required value (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`*, then this is name of the output bucket you created or identified in the prerequisites.
  - *`path`*: Optional value. The path and name of the build output ZIP file or folder:
    - If you specified `CODEPIPELINE` for *`artifacts-type`*, then 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`*, then 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`*, then AWS CodeBuild will use *`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`* and `MyArtifact.zip` for *`artifacts-name`*, then the path and name would be `MyPath/MyArtifact.zip`.
  - *`namespaceType`*: Optional value. 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`*, AWS CodeBuild will use *`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`*, then the path and name would be `MyPath/`*`build-ID`*`/MyArtifact.zip`.
  - *`artifacts-name`*: Required value (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`*, then 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`*, then the path and name would be `MyPath/MyArtifact.zip`.
  - *`packaging`*: Optional value. 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 `environment` object, information about this build project's participating build environment settings. These settings include:
  - *`environment-type`*: Required value. The type of build environment. The only allowed value is `LINUX_CONTAINER`.
  - *`image`*: Required value. The Docker image identifier this build environment will use. Typically, this identifier is expressed as *`image-name`*`:`*`tag`*. For example, in the Docker repository that AWS CodeBuild uses to manage its Docker images, this could be `aws/codebuild/java:openjdk-8`. 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`*.

   - *computeType*: Required value. A category corresponding to the number of CPU cores and memory this build environment will use. Allowed values include `BUILD_GENERAL1_SMALL`, `BUILD_GENERAL1_MEDIUM`, and `BUILD_GENERAL1_LARGE`.
   - 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 containing a `name` and `value` value of *environmentVariable-name* and *environmentVariable-value*.

     Important

     Any environment variables you set will 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` will be 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 `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` will be replaced by the literal value `$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 build spec declaration takes lowest precedence.
 - *serviceRole*: Required value. The ARN of the service role AWS CodeBuild will use to interact with services on behalf of the IAM user (for example, `arn:aws:iam::`*account-id*`:role/`*role-name*).
 - *timeoutInMinutes*: Optional value. The number of minutes, between 5 to 480 (8 hours), after which AWS CodeBuild will stop the build if it is not complete. If not specified, the default of 60 will be used. To determine if and when AWS 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 value. The alias or ARN of the AWS KMS customer master key (CMK) AWS CodeBuild will use 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 will be 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 AWS CodeBuild build project tags. Each tag is expressed as an object containing a `key` and `value` value of *tag-key* and *tag-value*.

 For an example, see To create the build project (AWS CLI) (p. 12) in Getting Started.

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 will appear 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,
    "environment": {
      "computeType": "computeType",
      "image": "image",
      "type": "environment-type",
      "environmentVariables": [
      {
        "name": "environmentVariable-name",
        "value": "environmentVariable-value"
      }
    },
    "source": {
      "type": "source-type",
      "location": "source-location",
      "buildspec": "buildspec"
      "auth": {
        "type": "auth-type",
        "resource": "resource"
      }
    },
    "encryptionKey": "encryptionKey",
    "arn": "arn"
  }
}
```

- The `project` object contains information about the new build project:
  - The `lastModified` value represents the time, in Unix time format, when information about the build project was last changed.
  - The `created` value represents the time, in Unix time format, when the build project was created.
  - The `arn` value represents the ARN of the build project.

Tip
Except for the build project name, you can change any of the build project's settings later. For more information, see Change a Build Project's Settings (AWS CLI) (p. 55).

To start running a build, see Run a Build (AWS CLI) (p. 61).

Create a Build Project (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

Create a Build Project (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `CreateProject` operation:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 581
X-Amz-Target: CodeBuild_20161006.CreateProject
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

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

If successful, a response is returned with body content similar to the following:

```
{
  "project": {
    "name": "sample-ant-helloworld-project",
    "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
    "tags": [],
    "artifacts": {
      "packaging": "ZIP",
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "sample-ant-helloworld-project",
    },
    "lastModified": 1475530648.823,
    "timeoutInMinutes": 60,
    "created": 1475530648.823,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "ant-image-ID",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/
AntHelloWorldWithTestsSample.zip"
    },
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/sample-ant-
helloworld-project"
  }
}
```

For descriptions of request and response data, see Create a Build Project (AWS CLI) (p. 43).

View a List of Build Project Names in AWS CodeBuild

To view a list of build projects in AWS CodeBuild, you can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API.

Topics

View a List of Build Project Names (Console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In the navigation pane, choose **Build projects**.

   Note
   By default, only the ten most recent build projects are displayed. To view more build projects, select a different value for **Projects per page** or select the back and forward arrows for **Viewing projects**.

View a List of Build Project Names (AWS CLI)

Run the `list-projects` command:

```
aws codebuild list-projects --sort-by sort-by --sort-order sort-order --next-
token next-token
```

In the preceding command, replace the following placeholders:

- *sort-by*: Optional string. 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. 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 a *next token*. To get the next batch of items in the list, run this command again, adding the next token to the call. To get all of the items in the list, keep running this command with each subsequent next token, until no more next tokens are returned.

For example, if you run this command:

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

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

```
{
  "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U
+AkMx8=",
  "projects": [
    "codebuild-demo-project",
    "codebuild-demo-project2",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project99"
  ]
}
```

If you run this command again:

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-
token Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=
```

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

```
{
  "projects": [
    "codebuild-demo-project100",
    "codebuild-demo-project101",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project122"
  ]
}
```

View a List of Build Project Names (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

View a List of Build Project Names (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `ListProjects` operation:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 2
X-Amz-Target: CodeBuild_20161006.ListProjects
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "sortBy": "NAME",
  "sortOrder": "ASCENDING"
}
```

If successful, a response is returned with body content similar to the following:

```
{
  "projects": [
    "sample-nodejs-project",
    "sample-ruby-project"
  ]
}
```

For descriptions of request and response data, see View a List of Build Project Names (AWS CLI) (p. 50).

View a Build Project's Details in AWS CodeBuild

To view the details of a build project in AWS CodeBuild, you can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API.

Topics

View a Build Project's Details (Console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In the navigation pane, choose **Build projects**.

   Note
   By default, only the ten most recent build projects are displayed. To view more build projects, select a different value for **Projects per page** or select the back and forward arrows for **Viewing projects**.
3. In the list of build projects, in the **Project** column, choose the link that corresponds to the build project.
4. On the **Build project: *`project-name`*** page, expand **Project details**.

View a Build Project's Details (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 90).

Run the `batch-get-projects` command:

```
aws codebuild batch-get-projects --names names
```

In the preceding command, replace the following placeholder:

- *`names`*: Required string. 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. 50).

For example, if you run this command:

```
aws codebuild batch-get-projects --names codebuild-demo-project codebuild-
demo-project2 my-other-demo-project
```

A result similar to the following might appear in the output. Ellipses (`...`) represent data omitted for brevity.

```
{
  "projectsNotFound": [
    "my-other-demo-project"
  ],
  "projects": [
    {
      ...
      "name": codebuild-demo-project,
      ...
    },
    {
      ...
      "name": codebuild-demo-project2",
      ...
    }
  ]
}
```

In the preceding output, the `projectsNotFound` array lists any build project names that were specified, but no information was 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. 43).

View a Build Project's Details (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

View a Build Project's Details (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `BatchGetProjects` operation:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 59
X-Amz-Target: CodeBuild_20161006.BatchGetProjects
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "names": [
    "sample-nodejs-project",
    "sample-ruby-project"
  ]
}
```

If successful, a response is returned with body content similar to the following. Ellipses (`...`) represent data omitted for brevity.

```
{
  "projects": [
    {
      ...
      "name": "sample-nodejs-project",
      ...
    }
    {
      ...
      "name": "sample-ruby-project",
      ...
    }
  ],
  "projectsNotFound": []
}
```

For descriptions of request and response data, see View a Build Project's Details (AWS CLI) (p. 52).

Change a Build Project's Settings in AWS CodeBuild

To change a build project's settings in AWS CodeBuild, you can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API.

Topics

Change a Build Project's Settings (Console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
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 change, choose **Actions**, and then choose **Update**.
   - Choose the link for the build project you want to change, and then choose **Edit project**.

   Note
   Only the most recent 10 build projects are displayed by default. To view more build projects, choose a different value for **Projects per page** or choose the back and forward arrows for **Viewing projects**.
4. On the project details page, to add a description or change the existing description, type the new or replacement description in the **Description** box.

   For more information about this and the following settings, see Create a Build Project (Console) (p. 40).
5. To change information about the source code location, in the **Source: What to build** area, choose **Update source**. Then change the displayed fields depending on the source provider type (for example, **Source provider**, **Bucket**, **S3 object**, or **Repository**).
6. To change information about the build environment, in the **Environment: How to build** area, choose **Update image**. Then change the displayed fields depending on the build environment type (for example, **Environment image**, **Operating system**, **Runtime**, **Version**, **Custom image type**, **Custom image ID**, **Amazon ECR repository**, or **Amazon ECR image**).

7. Do one of the following:

   - If your source code previously did not include a buildspec.yml file but does now, choose **Update build specification**, and then choose **Use buildspec.yml from source code**.
   - If your source code previously included a buildspec.yml file but now it does not, choose **Update build specification**, then choose **Insert build commands**, and then type the commands in **Build commands**.
8. To change information about the build output artifact location and name, in **Artifacts: Where to put the artifacts from this build project**, change the values of **Artifacts type**, **Artifact name**, **Bucket name**, or **Output files**.
9. To change information about the AWS CodeBuild service role, in **Service role**, change the values of **Create a role**, **Choose an existing service role from your account**, or **Role name**.

   Note
   When you use the console to create or update a build project, you can create an AWS 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 will be updated to work with the other build project. A service role can work with up to ten build projects.
10. To change information about the build timeout, in **Show advanced settings**, for **Timeout**, change the values of **hours** and **minutes**. If **hours** and **minutes** are left blank, the default value will be 60 minutes.
11. To change information about the AWS KMS customer master key (CMK), in **Show advanced settings**, change the value of **Encryption key**.

    Important
    If you leave **Encryption key** blank, then AWS CodeBuild will use the AWS-managed customer master key (CMK) for Amazon S3 in your AWS account instead.
12. To change information about the way build output artifacts are stored, in **Show advanced settings**, change the value of **Artifacts packaging**.
13. To change the amount of memory and vCPUs that will be used to run builds, in **Show advanced settings**, change the value of **Compute type**.
14. To change information about environment variables you want builds to use, in **Show advanced settings**, for **Environment variables**, change the values of **Name** and **Value**. Use **Add row** to add an environment variable. Choose the delete (**X**) icon next to an environment variable you no longer want to use.
15. To change information about tags for this build project, in **Show advanced settings**, 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.
16. Choose **Update**.

Change a Build Project's Settings (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 90).

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

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

   JSON-formatted data will appear 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. 43), and save your results.

> **Note**
> In the JSON-formatted data, you must provide the name of the build project that you want to change settings for. 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 containing the file you just saved, and run the `update-project` command again.

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

3. If successful, data similar to that as described in Create a Build Project (AWS CLI) (p. 43) will appear in the output.

Change a Build Project's Settings (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

Change a Build Project's Settings (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `UpdateProject` operation as follows:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 630
X-Amz-Target: CodeBuild_20161006.UpdateProject
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "name": "sample-ant-helloworld-project",
  "description": "This is a sample build project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/
AntHelloWorldWithTestsSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "name": "AntHelloWorldWithTestsOutputArtifact.zip",
    "packaging": "ZIP"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "ant-image-ID",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

If successful, a response is returned with body content similar to the following:

```
{
  "project": {
    "name": "sample-ant-helloworld-project",
    "description": "This is a sample build project",
    "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
    "created": 1475530648.823,
    "artifacts": {
      "packaging": "ZIP",
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "AntHelloWorldWithTestsOutputArtifact.zip"
    },
    "lastModified": 1475530781.439,
    "timeoutInMinutes": 60,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "ant-image-ID",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/
AntHelloWorldWithTestsSample.zip"
    },
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/sample-ant-
helloworld-project"
  }
}
```

For descriptions of request and response data, see Change a Build Project's Settings (AWS CLI) (p. 55).

Delete a Build Project in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API to delete a build project in AWS CodeBuild.

Warning
If you delete a build project, it cannot be recovered. All information about builds will also be deleted and cannot be recovered.

Topics

- Delete a Build Project (AWS SDKs) (p. 58)
- Delete a Build Project (HTTP API) (p. 58)

Delete a Build Project (Console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
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, choose **Actions**, and then choose **Delete**.
   - Choose the link for the build project you want to delete, and then choose **Delete**.

   Note
   Only the most recent 10 build projects are displayed by default. To view more build projects, select a different value for **Projects per page** or select the back and forward arrows for **Viewing projects**.

Delete a Build Project (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 90).

1. Run the `delete-project` command:

   ```
   aws codebuild delete-project --name name
   ```

   Replace the following placeholder:

   - *name*: Required string. The name of the build project to delete. To get a list of available build projects, run the `list-projects` command. For more information, see View a List of Build Project Names (AWS CLI) (p. 50).
2. If successful, no data and no errors appear in the output.

Delete a Build Project (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

Delete a Build Project (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `DeleteProject` operation as follows:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
```

```
Content-Length: 43
X-Amz-Target: CodeBuild_20161006.DeleteProject
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "name": "sample-ant-helloworld-project"
}
```

For a description of the request data, see Delete a Build Project (AWS CLI) (p. 58).

If successful, no data and no errors are returned in the response.

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).

You can perform these tasks when working with builds:

Topics

Run a Build in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API to run a build in AWS CodeBuild.

Topics

Run a Build (Console)

To use AWS CodePipeline to run a build with AWS CodeBuild, skip these steps and follow the instructions in Use AWS CodePipeline with AWS CodeBuild (p. 25).

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
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 new build** page, do one of the following:

   - For AWS CodeCommit or GitHub, for the optional **Source version** value, type the commit ID that corresponds to the version of the input artifact you want to build. If **Source version** is left blank, then the latest commit will be used.
   - For Amazon S3, for the optional **Source version** value, type the version ID that corresponds to the version of the input artifact you want to build. If **Source version** is left blank, then the latest version will be used.

4. Expand **Show advanced options**.

   - If you want to change the output artifacts type for this build only, choose the replacement type in **Artifacts type**.
   - If you want to change the name of the output artifact for this build only, type the replacement name in **Artifacts name**.
   - If you want to change the name of the output bucket for this build only, choose the replacement name in **Bucket name**.
   - If you want to change the way output artifacts are packaged for this build only, choose the replacement packaging type in **Artifacts packaging**.
   - If you want to change the build timeout for this build only, specify the new value in **Timeout**.

5. Expand **Environment variables**.

   If you want to change the environment variables for this build only, change the values of **Name** and **Value**. Use **Add row** to add a new environment variable for this build only. Choose the delete (**X**) icon next to an environment variable if you do not want to use it in this build.

   - **Important**
     Any environment variables you set will 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` will be 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 `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` will be replaced by the literal value `$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, its 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 build spec declaration takes lowest precedence.

6. Choose **Start build**.

   For detailed information about this build, see View Build Details (Console) (p. 65).

Run a Build (AWS CLI)

Note
To use AWS CodePipeline to run a build with AWS CodeBuild, skip these steps and follow the instructions in Create a Pipeline that Uses AWS CodeBuild (AWS CLI) (p. 31).
For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 90).

1. Run the `start-build` command in one of the following ways:

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

   Use this if you want to run a build that uses the latest version of the build input artifact and the build project's existing settings.

   ```
   aws codebuild start-build --generate-cli-skeleton
   ```

   Use this if you want to run a build with an earlier version of the build input artifact or if you want to override the settings for the build output artifacts, environment variables, build spec, 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. 49).

3. If you run the `start-build` command with the `--generate-cli-skeleton` option, JSON-formatted data will appear in the output. Copy the data to a file (for example, *`start-build.json`*) in a location on the local computer or instance where the AWS CLI is installed. Modify the copied data to match the following format, and save your results:

   ```
   {
     "projectName": "projectName",
     "sourceVersion": "sourceVersion",
     "artifactsOverride": {
       "type": "type",
       "location": "location",
       "path": "path",
       "namespaceType": "namespaceType",
       "name": "artifactsOverride-name",
       "packaging": "packaging"
     },
     "environmentVariablesOverride": [
       {
         "name": "environmentVariablesOverride-name",
         "value": "value"
       }
     ],
     "buildspecOverride": "buildspecOverride",
     "timeoutInMinutesOverride": timeoutInMinutesOverride
   }
   ```

   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 build input to be built. If not specified, the latest version will be used. If specified:
     - For AWS CodeCommit or GitHub: the commit ID to use.
     - For Amazon S3: the version ID of the object representing the build input ZIP file to use.
   - *`type`*: Optional string. The build output artifact type that overrides for this build the one defined in the build project.
   - *`location`*: Optional string. The build output artifact location that overrides for this build the one defined in the build project.
   - *`path`*: Optional string. The build output artifact path that overrides for this build the one defined in the build project.

- *namespaceType*: Optional string. The build output artifact path type that overrides for this build the one defined in the build project.
- *name*: Optional string. The build output artifact name that overrides for this build the one defined in the build project.
- *packaging*: Optional string. The build output artifact packaging type that overrides for this build the one defined in the build project.
- *environmentVariablesOverride-name*: Optional string. The name of an environment variable in the build project whose value you want to override for this build.
- *value*: Optional string. The value of the environment variable defined in the build project you want to override for this build.

  Important
  Any environment variables you set will 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` will be 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 `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` will be replaced by the literal value `$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 build spec declaration takes lowest precedence.
- *buildspecOverride*: Optional string. A build spec declaration that overrides for this build the one defined in the build project.
- *timeoutInMinutesOverride*: Optional number. The number of build timeout minutes that overrides for this build the one defined in the build project.

For information about valid values for these placeholders, see Create a Build Project (AWS CLI) (p. 43). To get a list of the latest settings for a build project, see View a Build Project's Details (p. 51).

4. Switch to the directory containing the file you just saved, and run the `start-build` command again.

```
aws codebuild start-build --cli-input-json file://start-build.json
```

5. If successful, data similar to that described in the To run the build (AWS CLI) (p. 15) procedure in Getting Started will appear in the output.

To work with detailed information about this build, make a note of the `id` value in the output, and then see View Build Details (AWS CLI) (p. 65).

Run a Build (AWS SDKs)

To use AWS CodePipeline to run a build with AWS CodeBuild, skip these steps and follow the instructions in Use AWS CodePipeline with AWS CodeBuild to Run Builds (p. 25) instead.

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

Run a Build (HTTP API)

To use AWS CodePipeline to run a build with AWS CodeBuild, skip these steps and follow the instructions in Use AWS CodePipeline with AWS CodeBuild to Run Builds (p. 25).

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `StartBuild` operation as follows:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 52
X-Amz-Target: CodeBuild_20161006.StartBuild
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "projectName": "sample-maven-in-5-minutes-project"
}
```

If successful, a response is returned with body content similar to this:

```
{
  "build": {
    "buildComplete": false,
    "artifacts": {
      "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/
MavenOutputArtifact.zip"
    },
    "projectName": "sample-maven-in-5-minutes-project",
    "timeoutInMinutes": 60,
    "buildStatus": "IN_PROGRESS",
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/java:openjdk-8",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-992648334831-input-bucket/
MavenIn5MinutesSample.zip"
    },
    "currentPhase": "SUBMITTED",
    "startTime": 147553131.256,
    "id": "sample-maven-in-5-minutes-project:build-ID",
    "arn": "arn:aws:codebuild:region-ID:account-ID:build/sample-maven-in-5-
minutes-project:build-ID"
```

```
  }
}
```

For descriptions of the request and response data, see Run a Build (AWS CLI) (p. 61).

View Build Details in AWS CodeBuild

To view details about builds managed by AWS CodeBuild, you can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API.

Topics

View Build Details (Console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
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 that corresponds to the build.
   - In the navigation pane, choose **Build projects**. In the list of build projects, in the **Project** column, choose the link that corresponds to the name of the build project. Then, in the list of builds, in the **Build run** column, choose the link that corresponds to the build.

   Note
   By default, only the ten most recent builds or build projects are displayed. To view more builds or build projects, select a different value for **Builds per page** or **Projects per page** or select the back and forward arrows for **Viewing builds** or **Viewing projects**.

View Build Details (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 90).

Run the `batch-get-builds` command:

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

Replace the following placeholder:

- *`ids`*: Required string. One or more build IDs to view details about. To specify more than one build ID, separate each build ID with a space. You can specify up to 100 build IDs. To get a list of build IDs, see one or more of the following topics:
  - View a List of Build IDs (AWS CLI) (p. 70)
  - View a List of Build IDs for a Build Project (AWS CLI) (p. 72)

For example, if you run this command:

```
aws codebuild batch-get-builds --ids codebuild-demo-
project:e9c4f4df-3f43-41d2-ab3a-60fe2EXAMPLE codebuild-
demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE my-other-
project:813bb6c6-891b-426a-9dd7-6d8a3EXAMPLE
```

If the command is successful, data similar to that described in the To view summarized build information (AWS CLI) (p. 17) procedure in Getting Started will appear in the output.

View Build Details (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

View Build Details (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `BatchGetBuilds` operation:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 145
X-Amz-Target: CodeBuild_20161006.BatchGetBuilds
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "ids": [
    "sample-codedeploy-project:build-ID"
  ]
}
```

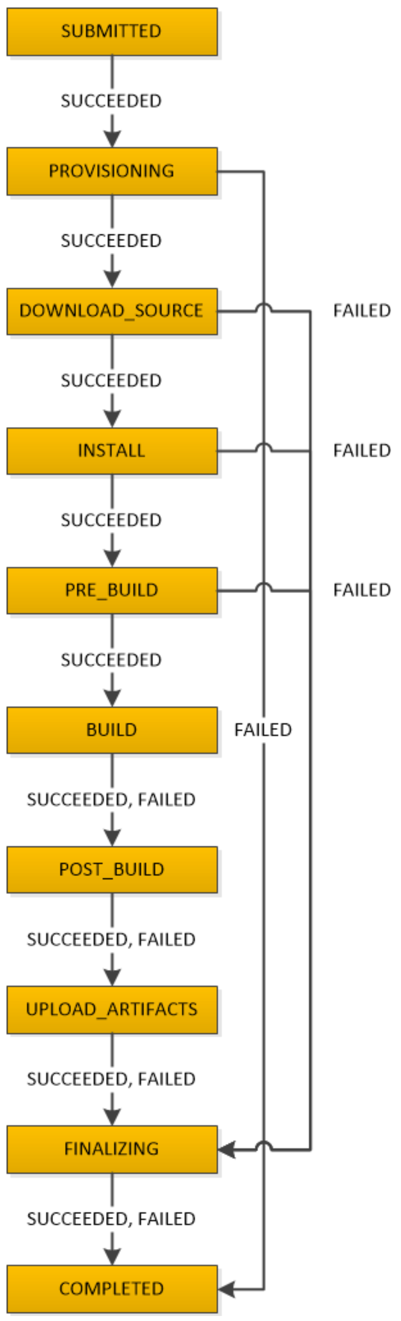
If successful, a response is returned with body content similar to the following:

```
{
  "builds": [
    {
      "buildComplete": true,
      "phases": [
        {
          "phaseStatus": "SUCCEEDED",
          "endTime": 1473358443.027,
          "phaseType": "SUBMITTED",
          "durationInSeconds": 0,
          "startTime": 1473358442.564
        },
        {
          "contexts": [],
          "phaseType": "PROVISIONING",
          "phaseStatus": "SUCCEEDED",
          "durationInSeconds": 28,
          "startTime": 1473358443.027,
          "endTime": 1473358471.934
        },
        {
          "contexts": [],
          "phaseType": "DOWNLOAD_SOURCE",
          "phaseStatus": "SUCCEEDED",
          "durationInSeconds": 3,
          "startTime": 1473358471.934,
          "endTime": 1473358475.846
        },
        {
          "contexts": [],
          "phaseType": "INSTALL",
          "phaseStatus": "SUCCEEDED",
          "durationInSeconds": 42,
          "startTime": 1473358475.846,
          "endTime": 1473358518.758
        },
        {
          "contexts": [],
          "phaseType":"PRE_BUILD",
          "phaseStatus": "SUCCEEDED",
          "durationInSeconds": 0,
          "startTime": 1473358518.758,
          "endTime": 1473358518.878
        },
        {
          "contexts": [],
          "phaseType": "BUILD",
          "phaseStatus": "SUCCEEDED",
          "durationInSeconds": 12,
          "startTime": 1473358518.878,
          "endTime": 1473358531.861
        },
        {
          "contexts": [],
          "phaseType": "POST_BUILD",
          "phaseStatus": "SUCCEEDED",
          "durationInSeconds": 0,
          "startTime": 1473358531.861,
          "endTime": 1473358531.969
        },
        {
          "contexts": [],
          "phaseType": "UPLOAD_ARTIFACTS",
          "phaseStatus": "SUCCEEDED",
          "durationInSeconds": 0,
          "startTime": 1473358531.969,
          "endTime": 1473358532.922
        },
        {
          "contexts": [],
          "phaseType": "FINALIZING",
          "phaseStatus": "SUCCEEDED",
          "durationInSeconds": 2,
          "startTime": 1473358532.922,
          "endTime": 1473358535.852
        },
        {
          "phaseType": "COMPLETED",
          "startTime": 1473358535.852
        }
      ],
      "logs": {
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-east-1#logEvent:group=/aws/codebuild/sample-codedeploy-
project;stream=86303ef6-a542-4359-9143-c50bcEXAMPLE",
        "groupName": "/aws/codebuild/sample-codedeploy-project",
        "streamName": "86303ef6-a542-4359-9143-c50bcEXAMPLE"
      },
      "artifacts": {
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-
bucket/CodeDeployOutputArtifact.zip"
      },
      "projectName": "sample-codedeploy-project",
      "timeoutInMinutes": 60,
      "buildStatus": "SUCCEEDED",
      "environment": {
        "computeType": "BUILD_GENERAL1_SMALL",
        "image": "aws/codebuild/java:openjdk-8",
        "type": "LINUX_CONTAINER"
      },
      "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/
CodeDeploySample.zip"
      },
      "currentPhase": "COMPLETED",
      "startTime": 1473358442.564,
      "endTime": 1473358535.852,
      "id": "sample-codedeploy-project:build-ID",
      "arn": "arn:aws:codebuild:region-ID:account-ID:build/sample-codedeploy-
project:build-ID"
    }
  ]
}
```

For descriptions of request and response data, see View Build Details (AWS CLI) (p. 65).

Build Phase Transitions

Builds in AWS CodeBuild proceed in phases:



View a List of Build IDs in AWS CodeBuild

To view a list of build IDs for builds managed by AWS CodeBuild, you can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API.

Topics

View a List of Build IDs (Console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In the navigation pane, choose **Build history**.

   Note
   By default, only the ten most recent builds are displayed. To view more builds, select a different value for **Builds per page** or select the back and forward arrows for **Viewing builds**.

View a List of Build IDs (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 90).

- Run the `list-builds` command:

```
aws codebuild list-builds --sort-order sort-order --next-token next-token
```

  In the preceding command, replace the following placeholders:

  - *sort-order*: Optional string. 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 would be returned, along with a unique string called a *next token*. To get the next batch of items in the list, run this command again, adding the next token to the call. To get all of the items in the list, keep running this command with each subsequent next token, until no more next tokens are returned.

  For example, if you run this command:

```
aws codebuild list-builds --sort-order ASCENDING
```

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

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

  If you run this command again:

```
aws codebuild list-builds --sort-order ASCENDING --next-token
 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```

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

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

View a List of Build IDs (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

View a List of Build IDs (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `ListBuilds` operation as follows:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 2
X-Amz-Target: CodeBuild_20161006.ListBuilds
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "sortOrder": "ASCENDING"
}
```

If successful, a response is returned with body content similar to the following:

```
{
  "ids": [
    "sample-ant-helloworld-project-2:3c3f9435-7ecf-4993-8063-ee942EXAMPLE",
    "sample-maven-in-5-minutes-project:0a6b900f-ab38-4d95-aeb7-74407EXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
  ],
  "nextToken": "AQECAHg3...The full token has been omitted for
 brevity...0PpX7X3f"
}
```

For descriptions of request and response data, see View a List of Build IDs (AWS CLI) (p. 70).

View a List of Build IDs for a Build Project in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API to view a list of build IDs for a build project in AWS CodeBuild.

Topics

View a List of Build IDs for a Build Project (Console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. In the navigation pane, choose **Build projects**. In the list of build projects, in the **Project** column, choose the build project.

Note
By default, only the ten most recent builds or build projects are displayed. To view more builds or build projects, select a different value for **Builds per page** or **Projects per page** or select the back and forward arrows for **Viewing builds** or **Viewing projects**.

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. 90).

Run the `list-builds-for-project` command, as follows:

```
aws codebuild list-builds-for-project --project-name project-name --sort-
order sort-order --next-token next-token
```

In the preceding command, replace the following placeholders:

- *`project-name`*: Required string. 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. 50).
- *`sort-order`*: Optional string. 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 a *next token*. To get the next batch of items in the list, run this command again, adding the next token to the call. To get all of the items in the list, keep running this command with each subsequent next token that is returned, until no more next tokens are returned.

For example, if you run this command similar to this:

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project
 --sort-order ASCENDING
```

A result like the following might appear in the output:

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"
    "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"
  ]
}
```

If you run this command again:

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project
 --sort-order ASCENDING --next-token 4AEA6u7J...The full token has been
 omitted for brevity...MzY2OA==
```

A result like the following might be output:

```
{
  "ids": [
    "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
    "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
  ]
}
```

View a List of Build IDs for a Build Project (AWS SDKs)

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs Reference (p. 92).

View a List of Build IDs for a Build Project (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `ListBuildsForProject` operation as follows:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 52
X-Amz-Target: CodeBuild_20161006.ListBuildsForProject
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "projectName": "sample-maven-in-5-minutes-project"
}
```

If successful, a response with body content like the following is returned:

```
{
  "ids": [
    "sample-maven-in-5-minutes-project:build-ID",
    ... The full list of build IDs has been omitted for brevity ...
  ]
}}
```

For descriptions of request and response data, see View a List of Build IDs for a Build Project (AWS CLI) (p. 72).

Stop a Build in AWS CodeBuild

To stop a build in AWS CodeBuild, you can use the AWS CodeBuild console, AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API.

Topics

Stop a Build (Console)

1. Open the AWS CodeBuild console at https://console.aws.amazon.com/codebuild/.
2. Do one of the following:
   - If the ***`build-project-name`*****:*****`build-ID`*** page is displayed, choose **Stop**.

- In the navigation pane, choose **Build history**. In the list of builds, choose the box that corresponds to the build, and then choose **Stop**.
- In the navigation pane, choose **Build projects**. In the list of build projects, in the **Project** column, choose the link that corresponds to the build project's name. In the list of builds, choose the box that corresponds to the build, and then choose **Stop**.

Note
By default, only the most recent 10 builds or build projects are displayed. To view more builds or build projects, select a different value for **Builds per page** or **Projects per page** or select the back and forward arrows for **Viewing builds** or **Viewing projects**.
If AWS CodeBuild cannot successfully stop a build (for example, the build process is already complete), the **Stop** button will be disabled or may be missing altogether.

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. 70)
  - View a List of Build IDs for a Build Project (AWS CLI) (p. 72)

If AWS CodeBuild successfully stops the build, the `buildStatus` value in the `build` object in the output will be `STOPPED`.

If AWS CodeBuild cannot successfully stop the build (for example, the build is already complete), the `buildStatus` value in the `build` object in the output will be 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 Reference (p. 92).

Stop a Build (HTTP API)

For more information about using the AWS CodeBuild HTTP API, see the HTTP API Reference (p. 93).

Call the `StopBuild` operation as follows:

Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 80
X-Amz-Target: CodeBuild_20161006.StopBuild
```

```
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/us-
east-1/codebuild/aws4_request, SignedHeaders=content-type;host;user-agent;x-
amz-date;x-amz-target, Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{
  "id": "sample-maven-in-5-minutes-project:build-ID"
}
```

For descriptions of the request data, see Stop a Build (AWS CLI) (p. 75).

If AWS CodeBuild successfully stops the build, a `build` object will be returned in the response body. The `buildStatus` value in the `build` object will be `STOPPED`.

If AWS CodeBuild cannot successfully stop the build (for example, the build has already been successfully completed), the `buildStatus` value in the `build` object will be the final build status (for example, `SUCCEEDED`).

Build Specification Reference for AWS CodeBuild

This topic provides important reference information about build specifications (build specs). A *build spec* is a collection of build commands and related settings, in YAML format, that AWS CodeBuild uses to run a build. You can include a build spec as part of the source code or you can define a build spec when you create a build project. For information about how a build spec works, see How AWS CodeBuild Works (p. 4).

Topics

Build Spec File Name and Storage Location

If you include a build spec as part of the source code, the build spec file must be named `buildspec.yml`.

If you include a build spec as part of the source code, for an AWS CodeCommit or GitHub repository, it must be stored in the root (top level) of the repository that contains the source code. For an Amazon S3 bucket, the build spec must be stored in the root (top level) of the build input ZIP file that contains the source code.

Build Spec Syntax

Build specs must be expressed in YAML format.

The build spec has the following syntax:

```
version: 0.1

environment_variables:
  plaintext:
    key: "value"
    key: "value"

phases:
  install:
    commands:
      - command
      - command
  pre_build:
    commands:
      - command
      - command
  build:
    commands:
      - command
      - command
  post_build:
    commands:
      - command
      - command
artifacts:
  files:
    - location
    - location
  discard-paths: yes
```

The build spec contains the following:

- `version`: Required mapping. Represents the build spec version. We recommend you use `0.1`.

  Note
  Version 0.0 is still supported, but no longer documented. We recommend that you use version 0.1 whenever possible. For more information, see Build Spec Versions (p. 81).

- `environment_variables`: Optional sequence. Represents information for one or more custom environment variables.
  - `plaintext`: Required if `environment_variables` is specified. Contains a mapping of *`key`*/*`value`* scalars, where each mapping represents a single custom environment variable.

    Important
    Any environment variables you set will 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` will be 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 `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` will be replaced by the literal value `$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 build spec declaration takes lowest precedence.

- `phases`: Required sequence. Represents the commands that AWS CodeBuild will run during each phase of the build. The allowed build phase names are:
  - `install`: Optional sequence. Represents the commands, if any, that AWS CodeBuild will run during installation. We recommend 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.

- `commands`: Required sequence if `install` is specified. Contains a sequence of scalars, where each scalar represents a single command that AWS CodeBuild will run during installation. AWS 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 AWS CodeBuild will run before the build. For example, you might use this phase to log 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 AWS CodeBuild will run before the build. AWS 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 AWS CodeBuild will run 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 AWS CodeBuild will run during the build. AWS 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 AWS CodeBuild will run 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 AWS CodeBuild will run after the build. AWS CodeBuild runs each command, one at a time, in the order listed, from beginning to end.

- `artifacts` Required sequence. Represents information about where AWS CodeBuild can find the build output and how AWS CodeBuild will prepare it for uploading to the Amazon S3 output bucket.
  - `files`: Required sequence. Represents the locations containing the build output artifacts in the build environment. Contains a sequence of scalars, with each scalar representing a separate location where AWS CodeBuild can find build output artifacts, relative to the original build location. 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`*.
  - `discard-paths`: Optional mapping. Represents whether paths to files in the build output artifact are preserved. `yes` if paths are preserved; 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`.

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 is invalid, builds might fail immediately. You can use a YAML validator to test whether your build spec declarations are valid YAML.

If you use the AWS CLI, the AWS SDKs, or the AWS CodeBuild HTTP API to declare a build spec when you create or update a build project, the build spec must be a single string expressed in YAML format, along with required whitespace and newline escape characters. For an example, see the next section.

If you use the AWS CodeBuild or AWS CodePipeline consoles to insert build commands instead of using a buildspec.yml file, you can only insert commands for the `build` phase. Instead of using the preceding syntax, you list in a single line all of the commands 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 AWS CodeBuild or AWS CodePipeline consoles to specify the locations of the build output artifacts in the build environment instead of using a buildspec.yml file. To do this, 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 `appspec.yml, target/my-app.jar`.

Build Spec Example

Here is an example of a buildspec.yml file.

```
version: 0.1

environment_variables:
  plaintext:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"

phases:
  install:
    commands:
      - apt-get update -y
      - apt-get install -y maven
  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
  discard-paths: yes
```

Here is an example of the preceding build spec, expressed as a single string, for use with the AWS CLI, the AWS SDKs, or the AWS CodeBuild HTTP API.

```
"version: 0.1\n\nenvironment_variables:\n  plaintext:\n    JAVA_HOME: "/
usr/lib/jvm/java-8-openjdk-amd64"\n\nphases:\n  install:\n    commands:\n
      - apt-get update -y\n      - apt-get install -y maven\n  pre_build:
\n    commands:\n      - echo Nothing to do in the pre_build phase...\n
 build:\n    commands:\n      - echo Build started on `date`\n      - mvn
 install\n  post_build:\n    commands:\n      - echo Build completed on
 `date`\nartifacts:\n  files:\n    - target/messageUtil-1.0.jar\n  discard-
paths: yes"
```

Here is an example of the commands in the `build` phase, for use with the AWS CodeBuild or AWS CodePipeline consoles.

```
echo Build started on `date` && mvn install
```

In these examples:

- A custom environment variable with the key of `JAVA_HOME` and the value of `/usr/lib/jvm/java-8-openjdk-amd64` will be set.
- You cannot change these build phase names. The only commands that will be run in this example are `apt-get update -y` and `apt-get install -y maven` (to install Apache Maven) and `mvn install` (to compile, test, and package the source code into a build output artifact and to perform other actions, such as install the build output artifact in its internal repository). The `echo` commands are included here to show how AWS CodeBuild runs commands and the order in which it runs them.
- `files` represents the set of files to upload to the build output location. In this example, AWS CodeBuild will upload 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` will be 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.

Build Spec Versions

The following table lists the build spec versions and the changes between versions.

Version	Changes
0.1	The `containers` and *`tag`* sequences have been removed. The `during_build` sequence has been renamed to `build`. The `type` mapping has been removed. All other sequences and mappings from version 0.0 are unchanged.
0.0	This is the initial build spec version. You can still use build specs that declare version 0.0. However, AWS CodeBuild will ignore the required `containers` and *`tag`* sequences in version 0.0 build specs. Version 0.0 will no longer be documented.

Build Environment Reference for AWS CodeBuild

When you call AWS CodeBuild to run a build, you must provide information about the build environment AWS CodeBuild will use. A *build environment* represents a combination of operating system, programming language runtime, and tools that AWS CodeBuild uses to run a build. For information about how a build environment works in an AWS CodeBuild solution, see How AWS CodeBuild Works (p. 4).

A build environment contains a Docker image. For information, see "Docker images" in the Understanding the architecture topic on the Docker Docs website.

When you provide information to AWS CodeBuild about the build environment, you specify the identifier of a Docker image in a repository type that is supported by AWS CodeBuild. These include the AWS CodeBuild Docker image repository, publicly available images in Docker Hub, and Amazon EC2 Container Registry (Amazon ECR) repositories in your AWS account:

- We recommend that you use Docker images stored in the AWS CodeBuild Docker image repository, because they are optimized for use with the service. For more information, see Docker Images Provided by AWS CodeBuild (p. 82).
- To get the identifier of a publicly available Docker image stored in Docker Hub, see the Searching for images section on the Docker Docs website.
- To learn how to work with Docker images stored in Amazon ECR repositories in your AWS account, see our Amazon ECR Sample (p. 109).

In addition to a Docker image identifier, you also specify a set of computing resources that the build environment will use. For more information, see Build Environment Compute Types (p. 88).

Docker Images Provided by AWS CodeBuild

AWS CodeBuild manages the following Docker images:

Platform	Programming language or framework	Runtime version	Additional components	Image identifier
Amazon Linux 2016.03, 64-bit v2.1.3	Golang	1.5.3	Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7	`aws/codebuild/eb-go-1.5-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Golang	1.5.3	Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7	`aws/codebuild/eb-go-1.5-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Java	1.7.0	Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7	`aws/codebuild/eb-java-7-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Java	1.7.0	Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7	`aws/codebuild/eb-java-7-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Java	1.8.0	Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7	`aws/codebuild/eb-java-8-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Java	1.8.0	Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7	`aws/codebuild/eb-java-8-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Node.js	4.4.6	Git 2.7.4, npm 2.15.5	`aws/codebuild/eb-nodejs-4.4.6-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.3	Python	2.6.9	meld3 1.0.2, pip 7.1.2, setuptools 18.4	`aws/codebuild/eb-python-2.6-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Python	2.6.9	meld3 1.0.2, pip 7.1.2, setuptools 18.4	`aws/codebuild/eb-python-2.6-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Python	2.7.10	meld3 1.0.2, pip 7.1.2, setuptools 18.4	`aws/codebuild/eb-python-2.7-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Python	2.7.10	meld3 1.0.2, pip 7.1.2, setuptools 18.4	`aws/codebuild/eb-python-2.7-amazonlinux-64:2.1.6`

Platform	Programming language or framework	Runtime version	Additional components	Image identifier
Amazon Linux 2016.03, 64-bit v2.1.3	Python	3.4.3	meld3 1.0.2, pip 7.1.2, setuptools 18.4	`aws/codebuild/eb-python-3.4-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Python	3.4.3	meld3 1.0.2, pip 7.1.2, setuptools 18.4	`aws/codebuild/eb-python-3.4-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Ruby	1.9.3	Bundler, RubyGems	`aws/codebuild/eb-ruby-1.9-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Ruby	1.9.3	Bundler, RubyGems	`aws/codebuild/eb-ruby-1.9-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Ruby	2.0.0	Bundler, RubyGems	`aws/codebuild/eb-ruby-2.0-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Ruby	2.0.0	Bundler, RubyGems	`aws/codebuild/eb-ruby-2.0-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Ruby	2.1.9	Bundler, RubyGems	`aws/codebuild/eb-ruby-2.1-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Ruby	2.1.9	Bundler, RubyGems	`aws/codebuild/eb-ruby-2.1-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Ruby	2.2.5	Bundler, RubyGems	`aws/codebuild/eb-ruby-2.2-amazonlinux-64:2.1.3`
Amazon Linux 2016.03, 64-bit v2.1.6	Ruby	2.2.5	Bundler, RubyGems	`aws/codebuild/eb-ruby-2.2-amazonlinux-64:2.1.6`
Amazon Linux 2016.03, 64-bit v2.1.3	Ruby	2.3.1	Bundler, RubyGems	`aws/codebuild/eb-ruby-2.3-amazonlinux-64:2.1.3`

Platform	Programming language or framework	Runtime version	Additional components	Image identifier
Amazon Linux 2016.03, 64-bit v2.1.6	Ruby	2.3.1	Bundler, RubyGems	`aws/codebuild/eb-ruby-2.3-amazonlinux-64:2.1.6`
Ubuntu 14.04	(Base image)		AWS CLI, Git 1.9.1	`aws/codebuild/ubuntu-base:14.04`
Ubuntu 14.04	Android	24.4.1	AWS CLI, Git 1.9.1, Java 6, pip 8.1.2, Python 2.7	`aws/codebuild/android-java-6:24.4.1`
Ubuntu 14.04	Android	24.4.1	AWS CLI, Git 1.9.1, Java 7, pip 8.1.2, Python 2.7	`aws/codebuild/android-java-7:24.4.1`
Ubuntu 14.04	Android	24.4.1	AWS CLI, Git 1.9.1, Java 8, pip 8.1.2, Python 2.7	`aws/codebuild/android-java-8:24.4.1`
Ubuntu 14.04	Docker	1.11.2	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/docker:1.11.2`
Ubuntu 14.04	Docker	1.12.1	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/docker:1.12.1`
Ubuntu 14.04	Golang	1.5.4	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/golang:1.5.4`
Ubuntu 14.04	Golang	1.6.3	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/golang:1.6.3`
Ubuntu 14.04	Golang	1.7.3	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/golang:1.7.3`
Ubuntu 14.04	Java	6	Apache Ant 1.9.6, Apache Maven 3.3.3, AWS CLI, Git 1.9.1, Gradle 2.7, pip 8.1.2, Python 2.7	`aws/codebuild/java:openjdk-6`

Platform	Programming language or framework	Runtime version	Additional components	Image identifier
Ubuntu 14.04	Java	7	Apache Ant 1.9.6, Apache Maven 3.3.3, AWS CLI, Git 1.9.1, Gradle 2.7, pip 8.1.2, Python 2.7	`aws/codebuild/java:openjdk-7`
Ubuntu 14.04	Java	8	Apache Ant 1.9.6, Apache Maven 3.3.3, AWS CLI, Git 1.9.1, Gradle 2.7, pip 8.1.2, Python 2.7	`aws/codebuild/java:openjdk-8`
Ubuntu 14.04	Node.js	4.3.2	AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7	`aws/codebuild/nodejs:4.3.2`
Ubuntu 14.04	Node.js	4.4.7	AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7	`aws/codebuild/nodejs:4.4.7`
Ubuntu 14.04	Node.js	5.12.0	AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7	`aws/codebuild/nodejs:5.12.0`
Ubuntu 14.04	Node.js	6.3.1	AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7	`aws/codebuild/nodejs:6.3.1`
Ubuntu 14.04	Node.js	7.0.0	AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7	`aws/codebuild/nodejs:7.0.0`
Ubuntu 14.04	Python	2.7.12	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/python:2.7.12`
Ubuntu 14.04	Python	3.3.6	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/python:3.3.6`
Ubuntu 14.04	Python	3.4.5	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/python:3.4.5`
Ubuntu 14.04	Python	3.5.2	AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7	`aws/codebuild/python:3.5.2`
Ubuntu 14.04	Ruby	2.1.10	AWS CLI, Bundler 1.13.5, Git 1.9.1, pip 8.1.2, Python 2.7, RubyGems 2.6.7	`aws/codebuild/ruby:2.1.10`

Platform	Programming language or framework	Runtime version	Additional components	Image identifier
Ubuntu 14.04	Ruby	2.2.5	AWS CLI, Bundler 1.13.5, Git 1.9.1, pip 8.1.2, Python 2.7, RubyGems 2.6.7	`aws/codebuild/ruby:2.2.5`
Ubuntu 14.04	Ruby	2.3.1	AWS CLI, Bundler 1.13.5, Git 1.9.1, pip 8.1.2, Python 2.7, RubyGems 2.6.7	`aws/codebuild/ruby:2.3.1`

You can use a build specification to install additional components that you need (for example, the AWS CLI, Apache Maven, Apache Ant, Mocha, RSpec, or similar) during the `install` build phase. For more information, see Build Spec Example (p. 80).

Note
AWS CodeBuild frequently updates the list of Docker images. To get the most current list, do one of the following:

- In the AWS CodeBuild console, in the create project wizard or update project page, in **Environment: How to build**, for **Environment image**, choose **Use an image managed by AWS CodeBuild**. Choose from the **Operating system**, **Runtime**, and **Version** drop-down lists. For more information, see Create a Build Project (Console) (p. 40) or Change a Build Project's Settings (Console) (p. 54).
- In the AWS CodePipeline console, in the create pipeline wizard on the **Step 3: Build** page, or in the **AWS CodeBuild** section of the **Add action** or **Edit action** pane for an existing pipeline, choose **Create a new build project**. In **Environment: How to build**, for **Environment image**, choose **Use an image managed by AWS CodeBuild**. Choose from the **Operating system**, **Runtime**, and **Version** drop-down lists. For more information, see Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console) (p. 28) or Add an AWS CodeBuild Build Action to a Pipeline (AWS CodePipeline Console) (p. 34).
- For the AWS CLI, run the `list-curated-environment-images` command as follows:

```
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 Reference (p. 92).
- For AWS CodeBuild HTTP API, call the `ListCuratedEnvironmentImages` operation as follows:

  Sample headers:

```
POST / HTTP/1.1
Host: codebuild.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 2
X-Amz-Target: CodeBuild_20161006.ListCuratedEnvironmentImages
X-Amz-Date: 20161006T213734Z
User-Agent: aws-cli/1.10.3 Python/2.7.9 Windows/8 botocore/1.3.25
Content-Type: application/x-amz-json-1.1
```

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAJTTFEXAMPLE/20161003/
us-east-1/codebuild/aws4_request, SignedHeaders=content-
type;host;user-agent;x-amz-date;x-amz-target,
 Signature=bc48f52bEXAMPLE
```

Sample request body:

```
{}
```

For more information on the Docker images that contain `eb-` in their identifer, see Supported Platforms and Platform History in the *AWS Elastic Beanstalk Developer Guide*. Note that Docker image containing `eb-` in their identifer are not available in the AWS CodeBuild and AWS CodePipeline consoles.
To confirm the version of a component installed on a Docker image, during a build you can run a build command that outputs the component's version number. For example, include one or more of the following commands in your build specification:

- For Apache Ant, run `ant -version`.
- For Apache Maven, run `mvn -version`.
- For the AWS CLI, run `aws --version`.
- For Bundler, run `bundle version`.
- For Git, run `git --version`.
- For Gradle, run `gradle --version`.
- For Java, run `java -version`.
- For NPM, run `npm --version`.
- For pip, run `pip --version`.
- For Python, run `python --version`.
- For RubyGems, run `gem --version`.
- For setuptools, run `easy_install --version`.

For example, the following build command (entered through the AWS CodeBuild or AWS CodePipeline console as part of a build project's settings) outputs the versions of the AWS CLI, Git, pip, and Python on a Docker image that has these components installed: `aws --version && git --version && pip --version && python --version`.

Build Environment Compute Types

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

Compute type	computeType value	Memory	vCPUs	Disk space
build.general1.small	`BUILD_GENERAL1_SMALL`	3 GB	2	64 GB
build.general1.medium	`BUILD_GENERAL1_MEDIUM`	7 GB	4	128 GB
build.general1.large	`BUILD_GENERAL1_LARGE`	15 GB	8	128 GB

To select one of these compute types, do one of the following:

- In the AWS CodeBuild console, in the create project wizard or update project page, expand **Show advanced settings**, and then choose one of the available **Compute type** options. For more information, see Create a Build Project (Console) (p. 40) or Change a Build Project's Settings (Console) (p. 54).
- In the AWS CodePipeline console, in the create pipeline wizard on the **Step 3: Build** page, or in the **AWS CodeBuild** section of the **Add action** or **Edit action** pane for an existing pipeline, choose **Create a new build project**, expand **Advanced**, and then choose one of the available **Compute type** options. For more information, see Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console) (p. 28) or Add an AWS CodeBuild Build Action to a Pipeline (AWS CodePipeline Console) (p. 34).
- For the AWS CLI, run the `create-project` or `update-project` command, specifying the `environment` object's `computeType` value as listed in the preceding table. For more information, see Create a Build Project (AWS CLI) (p. 43) or Change a Build Project's Settings (AWS CLI) (p. 55).
- For the AWS SDKs, call the equivalent of the `CreateProject` or `UpdateProject` operation for your target programming language, specifying the equivalent of the `environment` object's `computeType` value, as listed in the preceding table, for your target programming language. For more information, see the AWS SDKs Reference (p. 92).
- For AWS CodeBuild HTTP API, call the `CreateProject` or `UpdateProject` operation, specifying the `environment` object's `computeType` value as listed in the preceding table. For more information, see Create a Build Project (HTTP API) (p. 48) or Change a Build Project's Settings (HTTP API) (p. 56).

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 or the AWS CodeBuild HTTP API to call AWS CodeBuild, see the AWS SDKs Reference (p. 92) or the HTTP API Reference (p. 93).

To use the information in this topic, you should have already installed the AWS CLI and configured it for use with AWS CodeBuild, as described in Install and Configure the AWS CLI (p. 163).

Run this command to get a list of AWS CodeBuild commands.

```
aws codebuild help
```

Run this command to get information about a AWS CodeBuild command, where *command-name* is the name of the command.

```
aws codebuild command-name help
```

AWS CodeBuild commands include:

- `batch-get-builds`: Gets information about multiple builds in AWS CodeBuild. For more information, see View Build Details (AWS CLI) (p. 65).
- `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. 52).
- `create-project`: Creates a build project. For more information, see Create a Build Project (AWS CLI) (p. 43).
- `delete-project`: Deletes a build project. For more information, see Delete a Build Project (AWS CLI) (p. 58).
- `list-builds`: Lists Amazon Resource Names (ARNs) for builds in AWS CodeBuild. For more information, see View a List of Build IDs (AWS CLI) (p. 70).
- `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. 72).

- `list-curated-environment-images`: Gets a list of Docker images managed by AWS CodeBuild that you can use for your builds. For more information, see Docker Images Provided by AWS CodeBuild (p. 82).
- `list-projects`: Gets a list of build project names. For more information, see View a List of Build Project Names (AWS CLI) (p. 50).
- `start-build`: Starts running a build. For more information, see Run a Build (AWS CLI) (p. 61).
- `stop-build`: Attempts to stop the specified build from running. For more information, see Stop a Build (AWS CLI) (p. 75).
- `update-project`: Changes information about the specified build project. For more information, see Change a Build Project's Settings (AWS CLI) (p. 55).

AWS SDKs Reference for AWS CodeBuild

To use one the AWS SDKs to automate AWS CodeBuild, see the following resources.

Not what you're looking for? If an AWS SDK is not available for your programming language or platform, you can automate AWS CodeBuild directly from your code by using the AWS CodeBuild HTTP API. For more information, see the HTTP API Reference (p. 93). If you want to use the AWS CLI to run AWS CodeBuild, see the Command Line Reference (p. 90). If you want to use the AWS Tools for Windows PowerShell to run AWS CodeBuild, see the AWS CodeBuild section of the *AWS Tools for Windows PowerShell Cmdlet Reference*

The following AWS SDKs support AWS CodeBuild:

- 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.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*.

AWS CodeBuild HTTP API Reference

You can automate AWS CodeBuild directly from your code through a set of HTTP-based APIs. This is especially useful if your target AWS SDK does not yet support AWS CodeBuild.

Not what you're looking for? If you want to use the AWS SDKs to automate AWS CodeBuild, see the AWS SDKs Reference (p. 92). If you want to use the AWS CLI to run AWS CodeBuild, see the Command Line Reference (p. 90).

AWS CodeBuild HTTP API operations include:

- `BatchGetBuilds`: Gets information about one or more builds. For more information, see View Build Details (HTTP API) (p. 66).
- `BatchGetProjects`: Gets information about one or more build projects. For more information, see View a Build Project's Details (HTTP API) (p. 53).
- `CreateProject`: Creates a build project. For more information, see Create a Build Project (HTTP API) (p. 48).
- `DeleteProject`: Deletes a build project. For more information, see Delete a Build Project (HTTP API) (p. 58).
- `ListBuilds`: Gets a list of build IDs, with each build ID representing a single build. For more information, see View a List of Build IDs (HTTP API) (p. 71).
- `ListBuildsForProject`: Gets a list of build IDs for the specified build project, with each build ID representing a single build. For more information, see View a List of Build IDs for a Build Project (HTTP API) (p. 73).
- `ListCuratedEnvironmentImages`: Gets information about Docker images that are managed by AWS CodeBuild. For more information, see Docker Images Provided by AWS CodeBuild (p. 82).
- `ListProjects`: Gets a list of build project names, with each build project name representing a single build project. For more information, see View a List of Build Project Names (HTTP API) (p. 51).
- `StartBuild`: Starts running a build. For more information, see Run a Build (HTTP API) (p. 64).
- `StopBuild`: Attempts to stop running a build. For more information, see Stop a Build (HTTP API) (p. 75).
- `UpdateProject`: Changes the settings of a build project. For more information, see Change a Build Project's Settings (HTTP API) (p. 56).

For more information, see the AWS CodeBuild API Reference.

When you call AWS CodeBuild HTTP API operations, you must sign each request so that AWS can identify who sent it. You use Signature Version 4 to sign requests.

For example, the following sample Python code file (`getV4Sig.py`) uses Signature Version 4 to call the AWS CodeBuild HTTP API:

```
import sys, os, base64, datetime, hashlib, hmac
import requests # You may need to run 'pip install requests' to get this
 module.

def sign(key, msg):
  return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()

def getSignatureKey(key, date_stamp, regionName, serviceName):
  kDate = sign(('AWS4' + key).encode('utf-8'), date_stamp)
  kRegion = sign(kDate, regionName)
  kService = sign(kRegion, serviceName)
  kSigning = sign(kService, 'aws4_request')
  return kSigning

method = sys.argv[1] # The HTTP method, for example, 'POST'.
service = sys.argv[2] # The AWS CodeBuild service abbreviation, for example,
 'codebuild'.
host = sys.argv[3] # The AWS CodeBuild service host, 'codebuild.us-
east-1.amazonaws.com'.
region = sys.argv[4] # The target AWS region, for example, 'us-east-1'.
endpoint = sys.argv[5] # The AWS CodeBuild service endpoint, 'https://
codebuild.us-east-1.amazonaws.com/'.
content_type = sys.argv[6] # The content type, for example, 'application/x-
amz-json-1.1'.
amz_target = sys.argv[7] # The target AWS CodeBuild HTTP API, for example,
 'CodeBuild_20161006.ListBuildsForProject'.
access_key = sys.argv[8] # Your AWS access key ID.
secret_key = sys.argv[9] # Your AWS secret access key.
request_parameters = sys.argv[10] # The request body, for example,
 '{"projectName":"codebuild-demo-project"}' for Unix/Linux, or
 '{\"projectName\":\"codebuild-demo-project\"}' for Windows.
t = datetime.datetime.utcnow()
amz_date = t.strftime('%Y%m%dT%H%M%SZ')
date_stamp = t.strftime('%Y%m%d')
canonical_uri = '/'
canonical_querystring = ''
canonical_headers = 'content-type:' + content_type + '\n' + 'host:' + host +
 '\n' + 'x-amz-date:' + amz_date + '\n' + 'x-amz-target:' + amz_target + '\n'
signed_headers = 'content-type;host;x-amz-date;x-amz-target'
payload_hash = hashlib.sha256(request_parameters).hexdigest()
canonical_request = method + '\n' + canonical_uri + '\n' +
 canonical_querystring + '\n' + canonical_headers + '\n' + signed_headers +
 '\n' + payload_hash
algorithm = 'AWS4-HMAC-SHA256'
credential_scope = date_stamp + '/' + region + '/' + service + '/' +
 'aws4_request'
string_to_sign = algorithm + '\n' +  amz_date + '\n' +  credential_scope +
 '\n' +  hashlib.sha256(canonical_request).hexdigest()
signing_key = getSignatureKey(secret_key, date_stamp, region, service)
signature = hmac.new(signing_key, (string_to_sign).encode('utf-8'),
 hashlib.sha256).hexdigest()
```

```
authorization_header = algorithm + ' ' + 'Credential=' + access_key + '/'
 + credential_scope + ', ' +  'SignedHeaders=' + signed_headers + ', ' +
 'Signature=' + signature

headers = {'Content-Type':content_type,
  'X-Amz-Date':amz_date,
  'X-Amz-Target':amz_target,
  'Authorization':authorization_header}

print '\nBEGIN REQUEST++++++++++++++++++++++++++++++++++++'
print 'Request URL = ' + endpoint

r = requests.post(endpoint, data=request_parameters, headers=headers)

print '\nRESPONSE++++++++++++++++++++++++++++++++++++'
print 'Response code: %d\n' % r.status_code
print r.text
```

If you call the preceding code (from Windows) with parameters similar to the following:

```
python getV4Sig.py POST codebuild codebuild.us-east-1.amazonaws.com us-
east-1 https://codebuild.us-east-1.amazonaws.com/ application/x-amz-json-1.1
 CodeBuild_20161006.ListBuildsForProject my-AWS-access-key-ID my-AWS-secret-
access-key {\"projectName\":\"codebuild-demo-project\"}
```

You would get output similar to the following:

```
BEGIN REQUEST++++++++++++++++++++++++++++++++++++
Request URL = https://codebuild.us-east-1.amazonaws.com/

RESPONSE++++++++++++++++++++++++++++++++++++
Response code: 200

{"ids":["codebuild-demo-project:469eb862-09fd-4ec4-8e6b-
afa0aEXAMPLE","codebuild-demo-project:6bbab563-4a82-4d38-adf6-a01d9EXAMPLE"]}
```

For more information, see Signature Version 4 Signing Process in the *AWS General Reference*.

Authentication and Access Control for 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 Amazon S3 buckets and viewing Amazon CloudWatch Logs for builds. The following sections describe how you can use AWS Identity and Access Management (IAM) and AWS CodeBuild to help secure access to your resources:

- Authentication (p. 96)
- Access Control (p. 97)

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 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 AWS 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. AWS 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 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 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 Amazon 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 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 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 AWS CodeBuild. We recommend that you read the overview first.

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 Guide*.

When you grant permissions, you decide who is getting the permissions, the resources they can access, and the actions that can be performed on those resources.

Topics

AWS CodeBuild Resources and Operations

In AWS CodeBuild, the primary resource is a build project. In a policy, you use an Amazon Resource Name (ARN) to identify the resource the policy applies to. Builds are also resources and have ARNs associated with them. For more information, see Amazon Resource Names (ARN) and AWS Service Namespaces in the *Amazon Web Services General Reference*.

Resource type	ARN format
Build project	`arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*
Build	`arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:build/`*`build-ID`*
All AWS CodeBuild resources	`arn:aws:codebuild:*`
All AWS CodeBuild resources owned by the specified account in the specified region	`arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:*`

Note
Most AWS services treat a colon (:) or a forward slash (/) as the same character in ARNs. However, AWS 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-1:123456789012: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 AWS CodeBuild API actions accept multiple resources (for example, `BatchGetProjects`). To specify multiple resources in a single statement, separate their ARNs with commas, as follows:

```
"Resource": [
  "arn:aws:codebuild:us-east-1:123456789012:project/myBuildProject",
  "arn:aws:codebuild:us-east-1:123456789012:project/myOtherBuildProject"
]
```

AWS CodeBuild provides a set of operations to work with the AWS CodeBuild resources. For a list, see AWS CodeBuild Permissions Reference (p. 106).

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 AWS CodeBuild resource.
- If you create an IAM user in your AWS account and grant permissions to create AWS CodeBuild resources to that user, the user can create AWS CodeBuild resources. However, your AWS account, to which the user belongs, owns the AWS CodeBuild resources.
- If you create an IAM role in your AWS account with permissions to create AWS CodeBuild resources, anyone who can assume the role can create AWS CodeBuild resources. Your AWS account, to which the role belongs, owns the AWS 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 Guide*. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide Guide*.

Policies attached to an IAM identity are referred to as identity-based policies (IAM polices). Policies attached to a resource are referred to as resource-based policies. AWS CodeBuild supports identity-based (IAM policies) only.

Identity-Based Policies (IAM 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 Guide*.

In AWS 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 AWS CodeBuild, see Overview of Managing Access Permissions to Your AWS CodeBuild Resources (p. 97).

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, AWS 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. 98) and AWS CodeBuild Permissions Reference (p. 106).

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 Guide*.

For a table showing all of the AWS CodeBuild API actions and the resources they apply to, see the AWS CodeBuild Permissions Reference (p. 106).

Using Identity-Based Policies (IAM 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 AWS CodeBuild resources. For more information, see Overview of Managing Access Permissions to Your AWS CodeBuild Resources (p. 97).

Topics

The following shows an example of a permissions policy that allows a user to get information about build projects only in the `us-east-1` region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-1:123456789012:project/my*"

    }
  ]
}
```

Permissions Required to Use the AWS CodeBuild Console

A user who uses the AWS CodeBuild console must have a minimum set of permissions that allows the user to describe other AWS resources for the AWS account. You must have permissions from the following services:

- AWS CodeBuild
- Amazon CloudWatch
- AWS CodeCommit (if you are storing your source code in an AWS CodeCommit repository)
- Amazon EC2 Container Registry (Amazon ECR) (if you are using a build environment that relies on a Docker image in an Amazon ECR repository)
- Amazon EC2 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.

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 AWS CodeBuild including permissions to administrate AWS CodeBuild build projects.
- **AWSCodeBuildDeveloperAccess** – Provides access to AWS CodeBuild but does not allow build project administration.

- **AWSCodeBuildReadOnlyAccess** – Provides read-only access to AWS CodeBuild.

To access build output artifacts that AWS CodeBuild creates, you must also attach the AWS managed policy named **AmazonS3ReadOnlyAccess**.

To create and manage AWS 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 AWS CodeBuild actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

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 AWS 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. 101).

You can use the following sample IAM policies to limit AWS CodeBuild access for your IAM users and roles.

> **Note**
> All of the following examples use the US East (N. Virginia) region (us-east-1) and contain a fictitious AWS account ID.

Topics

Allow a User to Get Information About Build Projects

The following example policy statement allows a user to get information about build projects only in the `us-east-1` region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-1:123456789012:project/my*"

    }
  ]
}
```

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-1` region for account `123456789012` and using only the specified AWS CodeBuild service role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-1:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam:123456789012:role/CodeBuildServiceRole"
    }
  ]
}
```

Allow a User to Delete Build Projects

The following example policy statement allows a user to delete build projects only in the `us-east-1` region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteProject",
      "Resource": "arn:aws:codebuild:us-east-1:123456789012:project/my*"
    }
  ]
}
```

Allow a User to Get a List of Build Project Names

The following example policy statement allows a user to get a list of build project names for the same account:

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

Allow a User to Change Information About Build Projects

The following example policy statement allows a user to change information about build projects with any name but only in the `us-east-1` region for account `123456789012` and using only the specified AWS CodeBuild service role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateProject",
      "Resource": "arn:aws:codebuild:us-east-1:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam:123456789012:role/CodeBuildServiceRole"
    }
  ]
}
```

Allow a User to Get Information About Builds

The following example policy statement allows a user to get information about builds only in the `us-east-1` region for account `123456789012` for the build projects named `my-build-project` and `my-other-build-project`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetBuilds",
      "Resource": [
        "arn:aws:codebuild:us-east-1:123456789012:project/my-build-project",
        "arn:aws:codebuild:us-east-1:123456789012:project/my-other-build-
project"
      ]
    }
  ]
}
```

Allow a User to Get a List of Build IDs for a Build Project

The following example policy statement allows a user to get a list of build IDs only in the `us-east-1` region for account `123456789012` for the build projects named `my-build-project` and `my-other-build-project`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListBuildsForProject",
      "Resource": [
        "arn:aws:codebuild:us-east-1:123456789012:project/my-build-project",
        "arn:aws:codebuild:us-east-1:123456789012:project/my-other-build-
project"
      ]
    }
  ]
}
```

Allow a User to Get a List of Build IDs

The following example policy statement allows a user to get a list of all build IDs for the same account:

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

Allow a User to Begin Running Builds

The following example policy statement allows a user to run builds only in the `us-east-1` region for account `123456789012` for build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
      "Resource": "arn:aws:codebuild:us-east-1:123456789012:project/my*"
    }
  ]
}
```

Allow a User to Attempt to Stop Builds

The following example policy statement allows a user to attempt to stop running builds only in the `us-east-1` region for account `123456789012` for any build project that starts with the name `my`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StopBuild",
      "Resource": "arn:aws:codebuild:us-east-1:123456789012:project/my*"
    }
  ]
}
```

Allow a User to Get Information About Docker Images that Are Managed by AWS CodeBuild

The following example policy statement allows a user to get information about all Docker images that are managed by AWS CodeBuild:

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

AWS CodeBuild Permissions Reference

You can use the following table as a reference when you are setting up Access Control (p. 97) 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` 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 AWS CodeBuild actions and `codebuild:Batch*` specifies all AWS 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-1:123456789012:projects/my*
```

AWS CodeBuild API Operations and Required Permissions for Actions

`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`*

`CreateProject`

Actions: `codebuild:CreateProject`, `iam:PassRole`

Required to create build projects.

Resources: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*, `arn:aws:iam:`*`account-ID`*`:role/`*`role-name`*

`DeleteProject`

Action: `codebuild:DeleteProject`

Required to delete build projects.

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: *

`StartBuild`

Action: `codebuild:StartBuild`

Required to start running builds.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

`StopBuild`

Action: `codebuild:StopBuild`

Required to attempt to stop running builds.

Resource: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

`UpdateProject`

Actions: `codebuild:UpdateProject`, `iam:PassRole`

Required to change information about builds.

Resources: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*, `arn:aws:iam:`*`account-ID`*`:role/`*`role-name`*

AWS CodeBuild Samples

Use these samples to experiment with AWS CodeBuild:

Name	Description
Amazon ECR Sample (p. 109)	Uses a Docker image in an Amazon ECR repository to use Apache Maven to produce a single JAR file.
Ant Sample (p. 111)	Uses JUnit to test whether specific string values are output. Uses Apache Ant to produce a single JAR file.
AWS CodeDeploy Sample (p. 115)	Uses Apache Maven to produce a single JAR file. Uses AWS CodeDeploy to deploy the JAR file to an Amazon Linux instance. You can also use AWS CodePipeline to build and deploy the sample.
Build Notifications Sample (p. 118)	Uses Apache Maven to produce a single JAR file. Sends a build notification to subscribers of an Amazon SNS topic.
Docker Sample (p. 122)	Uses Docker 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.
Elastic Beanstalk Sample (p. 127)	Uses Apache Maven to produce a single WAR file. Uses Elastic Beanstalk to deploy the WAR file to an Elastic Beanstalk instance. You can also use AWS CodePipeline to build and deploy the sample.
Maven Sample (p. 130)	Uses Apache Maven to produce a single JAR file.
Node.js Sample (p. 133)	Uses Mocha to test whether an internal variable in code contains a specific string value. Produces a single .js file.

Name	Description
Python Sample (p. 135)	Uses Python to test whether an internal variable in code is set to a specific string value. Produces a single .py file.
Ruby Sample (p. 137)	Uses RSpec to test whether an internal variable in code is set to a specific string value. Produces a single .rb file.
Scala Sample (p. 139)	Uses sbt to produce a single JAR file.
WAR Sample (p. 142)	Uses Apache Maven to produce a single WAR file.

Amazon ECR Sample for AWS CodeBuild

This sample uses a Docker image in an Amazon EC2 Container Registry (Amazon ECR) image repository to build the Maven Sample (p. 130) for AWS CodeBuild.

Important
Running this sample may 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 AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, Amazon CloudWatch Pricing, and Amazon EC2 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. 122).
2. To create and upload the source code to be built, complete steps 1 through 4 of the Running the Sample section of the Maven Sample (p. 130).
3. Assign permissions to your image repository in Amazon ECR so that AWS CodeBuild can pull the repository's Docker image into the build environment:

    1. If you are using an IAM user instead of an AWS root account or an administrator IAM user to work with Amazon ECR, add the 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 access to managing permissions for 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. For more information, see Working with Inline Policies Using the AWS Management Console in the *IAM User Guide*.

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "ecr:*RepositoryPolicy"
      ],
      "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. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.
3. Choose **Repositories**.
4. In the list of repository names, choose the name of the repository you created or selected.
5. Choose the **Permissions** tab, choose **Add**, and then create a statement.
6. For **Sid**, type an identifier (for example, **`CodeBuildAccess`**).
7. For **Effect**, leave **Allow** selected because you want to allow access to AWS CodeBuild.
8. For **Principal**, type **`201349592320,570169269855,964771811575`**. (These are the internal AWS account IDs used by AWS CodeBuild to access Amazon ECR repositories in supported AWS regions). Leave **Everybody** cleared because you want to allow access to AWS CodeBuild only.

   Note
   The preceding internal AWS account IDs are for the US East (N. Virginia), EU (Ireland), and US West (Oregon) regions, respectively. To deny AWS CodeBuild access to Amazon ECR repositories in specific AWS regions, remove the internal AWS account IDs for the AWS regions where you want to deny access.
9. Skip the **All IAM entities** list.
10. For **Action**, select **Pull only actions**.

    All of the pull-only actions (**ecr:DownloadUrlForLayer**, **ecr:BatchGetImage**, and **ecr:BatchCheckLayerAvailability**) will be selected.
11. Choose **Save all**.

    This policy will be displayed in **Policy document**.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::201349592320:root",
          "arn:aws:iam::570169269855:root",
          "arn:aws:iam::964771811575: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. 38).

   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/MavenIn5MinutesSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "MavenIn5MinutesOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "account-ID.dkr.ecr.us-east-1.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 Amazon S3 output bucket.
6. Download the *`MavenIn5MinutesOutputArtifact`*`.zip` file to your local computer or instance, and then extract the contents of the *`MavenIn5MinutesOutputArtifact`*`.zip` file. In the extracted contents, open the `target` folder to get the `my-app-1.0-SNAPSHOT.jar` file.

Ant Hello World Sample for AWS CodeBuild

This Ant sample uses JUnit to test whether specific string values appear in the output. It produces a single JAR file named `HelloWorld.jar`. This sample is based on the Tutorial: Hello World with Apache Ant on the Apache Ant website.

Important
Running this sample may 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, and CloudWatch Logs. For more information, see AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, and Amazon CloudWatch Pricing.

Topics

Running the Sample

To run this sample:

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

    Important
    Do not upload *(root directory name)*, just the directories and files inside of *(root directory name)*.
    If you are using an Amazon 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)* to the ZIP file, just the directories and files inside of *(root directory name)*.

2. Create an associated build project, run the build, and view related build information by following the steps in Run AWS CodeBuild Directly (p. 38).

    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-ant-helloworld-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/AntHelloWorldSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "AntHelloWorldOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/java:openjdk-8",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "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 Amazon S3 output bucket, download the *AntHelloWorldOutputArtifact*`.zip` file to your local computer or instance, and then extract the contents of the file. In the extracted contents, open the `build/jar` folder to get the `HelloWorld.jar` file.

Directory Structure

This sample assumes this directory structure.

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

```
          |-- HelloMessageUtil.java
          |-- MessageUtil.java
          `-- TestMessageUtil.java
```

Files

This sample uses these files.

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

```
version: 0.1

phases:
  install:
    commands:
      - echo Downloading JUnit JAR file...
      - mkdir lib
      - wget http://central.maven.org/maven2/junit/junit/4.10/junit-4.10.jar
 -O ./lib/junit-4.10.jar
  pre_build:
    commands:
      - echo Creating directories...
      - mkdir build
      - mkdir build/classes
      - mkdir build/jar
  build:
    commands:
      - echo Build started on `date`
      - ant
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
    files:
      - build/jar/HelloWorld.jar
```

`build.xml` (in *(root directory name)*)

```
<?xml version="1.0"?>
<project name="HelloWorld" default="run" basedir=".">
  <property name="build.dir" location="build"/>
  <property name="classes.dir" value="${build.dir}/classes"/>
  <property name="jar.dir" value="${build.dir}/jar"/>
  <property name="lib.dir" location="lib"/>
  <property name="src.dir" location="src"/>
  <property name="main-class" value="HelloMessageUtil"/>
  <path id="classpath.base"/>
  <path id="classpath.test">
    <pathelement location="${lib.dir}/junit-4.10.jar"/>
    <pathelement location="${src.dir}"/>
    <pathelement location="${classes.dir}"/>
    <path refid="classpath.base"/>
  </path>
  <target name="compile">
    <javac srcdir="${src.dir}" destdir="${classes.dir}" verbose="true">
      <classpath refid="classpath.test"/>
    </javac>
  </target>
  <target name="jar" depends="compile">
    <jar destfile="${jar.dir}/${ant.project.name}.jar"
 basedir="${classes.dir}">
      <manifest>
        <attribute name="Main-Class" value="${main-class}"/>
      </manifest>
    </jar>
  </target>
  <target name="test" depends="jar">
    <junit printsummary="yes">
      <classpath refid="classpath.test"/>
      <formatter type="brief" usefile="false"/>
      <test name="TestMessageUtil"/>
    </junit>
  </target>
  <target name="run" depends="test">
    <java jar="${jar.dir}/${ant.project.name}.jar" fork="true"/>
  </target>
</project>
```

`HelloMessageUtil.java` (in *`(root directory name)`*`/src`)

```
public class HelloMessageUtil {

  public static void main(String[] args) {

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

    System.out.println("Inside main()");
    messageUtil.printMessage();
    messageUtil.salutationMessage();
  }
}
```

`MessageUtil.java` (in *`(root directory name)`*`/src`)

```
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;
  }
}
```

`TestMessageUtil.java` (in *(root directory name)*/src)

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

AWS CodeDeploy Sample for AWS 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 AWS CodeDeploy to deploy the JAR file to an Amazon Linux instance. (Alternatively, you can use AWS CodePipeline to automate the use of AWS 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 may 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 EC2. For more information, see AWS 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.

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

If successful, this directory structure and files will be 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.

```
version: 0.1

phases:
  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.

```
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 AWS CodeDeploy, such as an Amazon S3 input bucket or a GitHub repository.

   Important
   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. 40).

   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",
    "location": "codebuild-region-ID-account-ID-input-
bucket/CodeDeploySample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "CodeDeployOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/java:openjdk-8",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

7. If you plan to deploy the build output artifact with AWS CodeDeploy, then follow the steps in Run a Build (p. 60). Otherwise, skip this step. (This is because if you plan to deploy the build output artifact with AWS CodePipeline, then AWS CodePipeline will use AWS CodeBuild to run the build automatically.)
8. Complete the setup steps for using AWS CodeDeploy, including:

   - Grant the IAM user access to AWS CodeDeploy and the AWS services and actions AWS 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 AWS CodeDeploy to identify the instances where it will deploy the build output artifact. For more information, see Creating a Service Role for AWS CodeDeploy in the *AWS CodeDeploy User Guide*.
   - Create or identify an IAM instance profile to enable your instances to access the Amazon 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 AWS CodeDeploy where the build output artifact will be deployed. For more information, see Working with Instances for AWS CodeDeploy in the *AWS CodeDeploy User Guide*.
10. Create or identify an AWS CodeDeploy application and deployment group. For more information, see Creating an Application with AWS CodeDeploy in the *AWS CodeDeploy User Guide*.
11. Deploy the build output artifact to the instance.

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

    To deploy with AWS CodePipeline, see Use AWS CodePipeline with AWS CodeBuild (p. 25).
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`.

Build Notifications Sample for AWS CodeBuild

This sample uses Apache Maven to produce as build output a single JAR file named `my-app-1.0-SNAPSHOT.jar`. This sample is based on the Maven in 5 Minutes topic on the Apache Maven website. After the build is complete, AWS CodeBuild sends a build notification through Amazon SNS to subscribers.

Important
Running this sample may 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 SNS. For more information, see AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service 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.

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "sns:CreateTopic",
        "sns:List*",
        "sns:Publish",
        "sns:SetTopicAttributes",
        "sns:Subscribe"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

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

2. Create or identify a topic in Amazon SNS. AWS CodeBuild will send build notifications to this topic. For more information, see Create a Topic in the *Amazon SNS Developer Guide*.
3. Subscribe one or more recipients to the topic. For more information, see Subscribe to a Topic in the *Amazon SNS Developer Guide*.
4. Add the following statement (between *### BEGIN ADDING STATEMENT HERE ###* and *### END ADDING STATEMENT HERE ###*) to the policy you attached to your AWS CodeBuild service role. 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. This statement enables AWS CodeBuild to send build notifications to Amazon SNS topics.

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

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

5. Download and install Maven on your local computer or instance. For more information, see Downloading Apache Maven and Installing Apache Maven on the Apache Maven website.

6. 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-app -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

   If successful, this directory structure and files will be created.

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

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

```
version: 0.1

environment_variables:
  plaintext:
    AWS_DEFAULT_REGION: "region-ID"
    SNS_TOPIC_ARN: "arn:aws:sns:region-ID:account-ID:topic-name"

phases:
  build:
    commands:
      - echo Build started on `date`
      - mvn test
  post_build:
    commands:
      - echo Build completed on `date`
      - mvn package
      - aws sns publish --topic-arn $SNS_TOPIC_ARN --subject 'AWS
 CodeBuild - Build Completed' --message 'The build has completed.
 For build details, go to https://console.aws.amazon.com/cloudwatch/
home?region=us-east-1#logStream:group=/aws/codebuild/sample-build-
notifications-project in Amazon CloudWatch Logs.'
artifacts:
  files:
    - target/my-app-1.0-SNAPSHOT.jar
```

Replace *region-ID*, *account-ID*, and *topic-name* with your AWS region ID, your AWS account ID, and your Amazon SNS topic name.

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

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

8. Upload this contents of the `my-app` directory to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

   Important
   Do not upload *(root directory name)* or *(root directory name)*`/my-app`, just the directories and files inside of *(root directory name)*`/my-app`.
   If you are using an Amazon 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-app` to the ZIP file, just the directories and files inside of *(root directory name)*`/my-app`.

9. Create a build project, run the build, and view build information by following the steps in Run AWS CodeBuild Directly (p. 38).

   For example, 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-build-notifications-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/BuildNotificationsSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "BuildNotificationsOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/java:openjdk-8",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

10. Confirm that AWS CodeBuild successfully sent the build notification. For example, if you subscribed your email address to the Amazon SNS topic, check to see if the build notification email is now in your inbox.
11. To get the build output artifact, open your Amazon S3 output bucket.
12. Download the *`BuildNotificationsOutputArtifact`*`.zip` file to your local computer or instance, and then extract the contents of the file. In the extracted contents, open the `target` folder to get the `my-app-1.0-SNAPSHOT.jar` file.

Docker Sample for AWS CodeBuild

This sample produces as build output a Docker image and then pushes the Docker image to an Amazon EC2 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. 125).

This sample was tested referencing the `maven:3.3.9-jdk-8` Docker image in Docker Hub.

Important
Running this sample may 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 AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, Amazon CloudWatch Pricing, and Amazon EC2 Container Registry Pricing.

Topics

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 enables creating 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*.

```
{
  "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 will be creating your build environment and running 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 will 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 enables AWS CodeBuild to upload Docker images to Amazon ECR repositories. Ellipses (`...`) are used for brevity and to help you locate where to add the statement. Do not remove any statements, and do not type these ellipses into the policy.

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:CompleteLayerUpload",
        "ecr:GetAuthorizationToken",
        "ecr:InitiateLayerUpload",
        "ecr:PutImage",
        "ecr:UploadLayerPart"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

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

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

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

5. Create a build project, run the build, and view related build information by following the steps in Run AWS CodeBuild Directly (p. 38).

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

```
{
  "name": "sample-docker-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/DockerSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/docker:1.12.1",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [
      {
        "name": "AWS_DEFAULT_REGION",
        "value": "region-ID"
      },
      {
        "name": "AWS_ACCOUNT_ID",
        "value": "account-ID"
      },
      {
        "name": "IMAGE_REPO_NAME",
        "value": "Amazon-ECR-repo-name"
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
    ]
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

6. Confirm that AWS CodeBuild successfully pushed the Docker image to the repository:

   1. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.
   2. Choose **Repositories**.
   3. Choose the repository name. The image should be listed on the **Images** tab.

Directory Structure

This sample assumes this directory structure.

```
(root directory name)
    |-- buildspec.yml
    `-- Dockerfile
```

Files

This sample uses these files.

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

```
version: 0.1

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

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

```
FROM maven:3.3.9-jdk-8

RUN echo "Hello World"
```

Adapting the Sample to Push the Image to Docker Hub

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

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 --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...
      - docker login --username="$DOCKER_HUB_USERNAME" --
password="$DOCKER_HUB_PASSWORD"
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $IMAGE_REPO_NAME:$IMAGE_TAG
...
```

2. Upload the modified code to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

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

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

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

With these lines of code, which specifies and separates your Docker Hub credentials and repository name from the source code.

```
...
    "environmentVariables": [
      {
        "name": "DOCKER_HUB_USERNAME",
        "value": "your-Docker-Hub-user-name"
      },
      {
        "name": "DOCKER_HUB_PASSWORD",
        "value": "your-Docker-Hub-password"
      },
      {
        "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. 38) 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.

AWS Elastic Beanstalk Sample for AWS 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 uses Elastic Beanstalk to deploy the WAR file to an instance. (Alternatively, you can use AWS CodePipeline to automate the use of Elastic Beanstalk to deploy the WAR file to an instance.) This sample is based on the WAR Sample (p. 142).

Important
Running this sample may 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 EC2. For more information, see AWS 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.

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

   If successful, this directory structure and files will be created.

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

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

   ```
   version: 0.1

   phases:
     post_build:
       commands:
         - echo Build completed on `date`
         - mvn package
   artifacts:
     files:
       - target/my-web-app.war
       - move-war.sh
     discard-paths: yes
   ```

4. Create a file with this content. Name the file `move-war.sh`, and then add it to the *(root directory name)*`/my-web-app` directory.

   ```
   #!/bin/sh

   sudo mv /var/lib/tomcat8/webapps/ROOT/my-web-app.war /var/lib/tomcat8/
   webapps/
   ```

   You will use this file later to move the deployed build output artifact to the correct location on the instance.

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

   ```
   (root directory name)
   ```

```
     `-- my-web-app
           |-- buildpsec.yml
           |-- move-war.sh
           |-- pom.xml
           `-- src
                 `-- main
                       |-- resources
                       `-- webapp
                             |-- WEB-INF
                             |     `-- web.xml
                             `-- index.jsp
```

5. Upload this contents of the `my-web-app` directory to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

   Important
   Do not upload *`(root directory name)`* or *`(root directory name)`*`/my-web-app`, just the directories and files inside of *`(root directory name)`*`/my-web-app`.
   If you are using an Amazon 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 inside of *`(root directory name)`*`/my-web-app`. For example, the ZIP file should contain these directories and files:

```
ElasticBeanstalkSample.zip
     |-- buildpsec.yml
     |-- move-war.sh
     |-- pom.xml
     `-- src
           `-- main
                 |-- resources
                 `-- webapp
                       |-- WEB-INF
                       |     `-- web.xml
                       `-- index.jsp
```

6. Create a build project by following the steps in Create a Build Project (p. 40).

   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-elastic-beanstalk-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/ElasticBeanstalkSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "ElasticBeanstalkOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/java:openjdk-8",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

7. If you plan to deploy the build output artifact with Elastic Beanstalk, then follow the steps in Run a Build (p. 60). Otherwise, skip this step. (This is because if you plan to deploy the build output artifact with AWS CodePipeline, then AWS CodePipeline will use AWS CodeBuild to run the build automatically.)
8. Complete the setup steps for using Elastic Beanstalk, including:
   - If you are using an IAM user instead of an AWS root account (not recommended) or an IAM administrator to access Elastic Beanstalk, grant the user access to Elastic Beanstalk and the AWS services and actions Elastic Beanstalk depends on. For more information, see Elastic Beanstalk User Policy in the *AWS Elastic Beanstalk Developer Guide*.
   - Create or identify a service role to enable Elastic Beanstalk to interact with dependent AWS resources. For more information, see Elastic Beanstalk Service Role in the *AWS Elastic Beanstalk Developer Guide*.
   - Create or identify an IAM instance profile to enable your Elastic Beanstalk instances to interact with dependent AWS resources. For more information, see Elastic Beanstalk Instance Profile in the *AWS Elastic Beanstalk Developer Guide*.
9. Create or identify an Elastic Beanstalk application. For more information, see Create an Application in the *AWS Elastic Beanstalk Developer Guide*.
10. Create or identify an Elastic Beanstalk environment. For this sample, your environment should contain a single-instance web server environment with a predefined configuration of Tomcat. For more information, see The New Environment Wizard and Using the AWS Elastic Beanstalk Tomcat Platform in the *AWS Elastic Beanstalk Developer Guide*.
11. Deploy the build output artifact to the instance.

    If you use the Elastic Beanstalk **New Environment** wizard, the build output artifact should be deployed automatically.

    To deploy with AWS CodePipeline, see Use AWS CodePipeline with AWS CodeBuild (p. 25).
12. To find the build output artifact after the deployment is complete, sign in to the instance and look in the `/tmp/deployment/application/ROOT` and `/var/lib/tomcat8/webapps/ROOT` directories for the files named `move-war.sh` and `my-web-app.war`.
13. To see the results in a web browser:
    1. Sign in to the instance and then run these two commands, one at a time, to move the deployed build output artifact to the correct location on the instance.

       ```
       cd /var/lib/tomcat8/webapps/ROOT
       ```

       ```
       sh move-war.sh
       ```
    2. From a web browser, go to the instance's environment URL prefix followed by `/my-web-app/` (for example, `http://my-environment-name-env.us-east-1.elasticbeanstalk.com/my-web-app/`).

       The web browser should display the text `Hello World!`.

Maven in 5 Minutes Sample for AWS CodeBuild

This Maven sample produces as build output a single JAR file named `my-app-1.0-SNAPSHOT.jar`. This sample is based on the Maven in 5 Minutes topic on the Apache Maven website.

Important
Running this sample may 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, and CloudWatch Logs. For more information, see AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, and Amazon CloudWatch 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.

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

   If successful, this directory structure and files will be 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.

```
version: 0.1

phases:
  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
```

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

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

4. Upload this contents of the `my-app` directory to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

   Important
   Do not upload *`(root directory name)`* or *`(root directory name)`*`/my-app`, just the directories and files inside of *`(root directory name)`*`/my-app`.
   If you are using an Amazon 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-app` to the ZIP file, just the directories and files inside of *`(root directory name)`*`/my-app`.

5. Create a build project, run the build, and view related build information by following the steps in Run AWS CodeBuild Directly (p. 38).

   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-maven-in-5-minutes-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/MavenIn5MinutesSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "MavenIn5MinutesOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/java:openjdk-8",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

6. To get the build output artifact, open your Amazon S3 output bucket.
7. Download the *MavenIn5MinutesOutputArtifact*.zip file to your local computer or instance, and then extract the contents of the *MavenIn5MinutesOutputArtifact*.zip file. In the extracted contents, open the target folder to get the my-app-1.0-SNAPSHOT.jar file.
8. Download the main.zip file to your local computer or instance, and then extract the contents of the main.zip file. In the extracted contents, open the target folder to get the my-app-1.0-SNAPSHOT.jar file.

Node.js Hello World Sample for AWS CodeBuild

This Node.js sample tests whether an internal variable in code starts with the string Hello. It produces as build output a single file named HelloWorld.js.

Important
Running this sample may 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, and CloudWatch Logs. For more information, see AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, and Amazon CloudWatch Pricing.

Topics

Running the Sample

To run this sample:

1. On your local computer or instance, create the files as described in the Directory Structure and Files sections of this topic, and then upload them to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

   Important
   Do not upload *(root directory name)*, just the files inside of *(root directory name)*.
   If you are using an Amazon 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. 38).

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

```
{
  "name": "sample-nodejs-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/NodeJSSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "NodeJSOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/nodejs:6.3.1",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "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, open your Amazon S3 output bucket.
4. Download the *`NodeJSOutputArtifact`*`.zip` file to your local computer or instance, and then extract the contents of the file. In the extracted contents, get the `HelloWorld.js` file.

Directory Structure

This sample assumes this directory structure.

```
(root directory name)
    |-- buildspec.yml
    `-- HelloWorld.js
```

Files

This sample uses these files.

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

```
version: 0.1

phases:
  install:
    commands:
      - echo Installing Mocha...
      - npm install -g mocha
  pre_build:
    commands:
      - echo Installing source NPM dependencies...
      - npm install unit.js
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Node.js code
      - mocha HelloWorld.js
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - HelloWorld.js
```

`HelloWorld.js` (in *`(root directory name)`*)

```
var test = require('unit.js');
var str = 'Hello, world!';

test.string(str).startsWith('Hello');

if (test.string(str).startsWith('Hello')) {
  console.log('Passed');
}
```

Python Hello World Sample for AWS CodeBuild

This Python sample tests whether an internal variable in code contains the string `Hello world!`. It produces as build output a single file named `HelloWorld.py`.

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

Topics

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 Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

   Important
   Do not upload *(root directory name)*, just the files inside of *(root directory name)*.
   If you are using an Amazon 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. 38).

   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-python-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/PythonSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "PythonOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/python:3.5.2",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "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, open your Amazon S3 output bucket.
4. Download the `PythonOutputArtifact.zip` file to your local computer or instance, and then extract the contents of the file. In the extracted contents, get the `HelloWorld.py` file.

Directory Structure

This sample assumes this directory structure.

```
(root directory name)
    |-- buildspec.yml
    |-- HelloWorld.py
    `-- HelloWorld_tst.py
```

Files

This sample uses these files.

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

```
version: 0.1

phases:
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Python code...
      - python HelloWorld_tst.py
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - HelloWorld.py
```

`HelloWorld.py` (in *(root directory name)*)

```
class HelloWorld:
  def __init__(self):
    self.message = 'Hello world!'
```

`HelloWorld_tst.py` (in *(root directory name)*)

```
import unittest
from HelloWorld import HelloWorld
```

```
class MyTestCase(unittest.TestCase):
  def test_default_greeting_set(self):
    hw = HelloWorld()
    self.assertEqual(hw.message, 'Hello world!')

if __name__ == '__main__':
  unittest.main()
```

Ruby Hello World Sample for AWS CodeBuild

This Ruby sample tests whether an internal variable in code contains the string `Hello, world!`. It produces as build output a single file named `HelloWorld.rb`.

Important
Running this sample may 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, and CloudWatch Logs. For more information, see AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, and Amazon CloudWatch Pricing.

Topics

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 Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

   Important
   Do not upload *`(root directory name)`*, just the files inside of *`(root directory name)`*.
   If you are using an Amazon 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. 38).

   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-ruby-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/RubySample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "RubyOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/ruby:2.3.1",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "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, open your Amazon S3 output bucket.
4. Download the *`RubyOutputArtifact`*`.zip` file to your local computer or instance, and then extract the contents of the file. In the extracted contents, get the `HelloWorld.rb` file.

Directory Structure

This sample assumes this directory structure.

```
(root directory name)
    |-- buildspec.yml
    |-- HelloWorld.rb
    `-- HelloWorld_spec.rb
```

Files

This sample uses these files.

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

```
version: 0.1

phases:
  install:
    commands:
      - echo Installing RSpec...
      - gem install rspec
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Ruby code...
      - rspec HelloWorld_spec.rb
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - HelloWorld.rb
```

`HelloWorld.rb` (in *`(root directory name)`*)

```
class HelloWorld
  def say_hello()
    return 'Hello, world!'
  end
end
```

`HelloWorld_spec.rb` (in *`(root directory name)`*)

```
require './HelloWorld'

describe HelloWorld do
  context "When testing the HelloWorld class" do
    it "The say_hello method should return 'Hello World'" do
      hw = HelloWorld.new
      message = hw.say_hello
      puts 'Succeed' if expect(message).to eq "Hello, world!"
    end
  end
end
```

Scala Hello World Sample for AWS CodeBuild

This Scala sample produces as build output a single JAR file named `core_2.11-1.0.0.jar`.

Important
Running this sample may 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, and CloudWatch Logs. For more information, see AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, and Amazon CloudWatch Pricing.

Topics

Running the Sample

To run this sample:

1. Identify a Docker image that contains sbt, a build tool for Scala projects. To find a compatible Docker image, search Docker Hub for **`sbt`**.
2. Create the directory structure and files as described in the Directory Structure and Files sections of this topic, and then upload them to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

   Important
   Do not upload *`(root directory name)`*, just the directories and files inside of *`(root directory name)`*.
   If you are using an Amazon 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)`* to the ZIP file, just the directories and files inside of *`(root directory name)`*.
3. Create a build project, run the build, and view related build information by following the steps in Run AWS CodeBuild Directly (p. 38).

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

4. To get the build output artifact, open your Amazon S3 output bucket.
5. Download the *`ScalaOutputArtifact`*`.zip` file to your local computer or instance, and then extract the contents of the file. In the extracted contents, open the `core/target/scala-2.11` folder to get the `core_2.11-1.0.0.jar` file.

Directory Structure

This sample assumes this directory structure.

```
(root directory name)
    |-- buildspec.yml
    |-- core
    |     `-- src
    |           `-- main
    |                 `-- scala
    |                       `-- Test.scala
    |-- macros
    |     `-- src
    |           `-- main
    |                 `-- scala
    |                       `-- Macros.scala
    `-- project
          |-- Build.scala
          `-- build.properties
```

Files

This sample uses these files.

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

```
version: 0.1

phases:
  build:
    commands:
      - echo Build started on `date`
      - echo Run the test and package the code...
      - sbt run
  post_build:
    commands:
      - echo Build completed on `date`
      - sbt package
artifacts:
  files:
    - core/target/scala-2.11/core_2.11-1.0.0.jar
```

`Test.scala` (in *`(root directory name)`*`/core/src/main/scala`)

```
object Test extends App {
  Macros.hello
}
```

`Macros.scala` (in *`(root directory name)`*`/macros/src/main/scala`)

```
import scala.language.experimental.macros
import scala.reflect.macros.Context

object Macros {
  def impl(c: Context) = {
    import c.universe._
    c.Expr[Unit](q"""println("Hello World")""")
  }

  def hello: Unit = macro impl
}
```

`Build.scala` (in *`(root directory name)`*`/project`)

```
import sbt._
import Keys._

object BuildSettings {
  val buildSettings = Defaults.defaultSettings ++ Seq(
    organization := "org.scalamacros",
    version := "1.0.0",
    scalaVersion := "2.11.8",
    crossScalaVersions := Seq("2.10.2", "2.10.3", "2.10.4", "2.10.5",
 "2.10.6", "2.11.0", "2.11.1", "2.11.2", "2.11.3", "2.11.4", "2.11.5",
 "2.11.6", "2.11.7", "2.11.8"),
    resolvers += Resolver.sonatypeRepo("snapshots"),
    resolvers += Resolver.sonatypeRepo("releases"),
    scalacOptions ++= Seq()
  )
}

object MyBuild extends Build {
  import BuildSettings._

  lazy val root: Project = Project(
    "root",
    file("."),
    settings = buildSettings ++ Seq(
      run <<= run in Compile in core)
  ) aggregate(macros, core)

  lazy val macros: Project = Project(
    "macros",
    file("macros"),
    settings = buildSettings ++ Seq(
      libraryDependencies <+= (scalaVersion)("org.scala-lang" % "scala-
reflect" % _),
      libraryDependencies := {
        CrossVersion.partialVersion(scalaVersion.value) match {
          // if Scala 2.11+ is used, quasiquotes are available in the
 standard distribution
          case Some((2, scalaMajor)) if scalaMajor >= 11 =>
            libraryDependencies.value
          // in Scala 2.10, quasiquotes are provided by macro paradise
          case Some((2, 10)) =>
            libraryDependencies.value ++ Seq(
              compilerPlugin("org.scalamacros" % "paradise" % "2.1.0-M5"
 cross CrossVersion.full),
              "org.scalamacros" %% "quasiquotes" % "2.1.0-M5" cross
 CrossVersion.binary)
        }
      }
    )
  )

  lazy val core: Project = Project(
    "core",
    file("core"),
    settings = buildSettings
  ) dependsOn(macros)
}
```

`build.properties` (in *`(root directory name)`*`/project`)

```
sbt.version=0.13.7
```

WAR Hello World Sample for AWS CodeBuild

This Maven sample produces as build output a single Web application ARchive (WAR) file named `my-web-app.war`.

Important
Running this sample may 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, and CloudWatch Logs. For more information, see AWS CodeBuild Pricing, Amazon S3 Pricing, AWS Key Management Service Pricing, and Amazon CloudWatch 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.

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

   If successful, this directory structure and files will be created.

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

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

   ```
   version: 0.1

   phases:
     post_build:
       commands:
         - echo Build completed on `date`
         - mvn package
   artifacts:
     files:
       - target/my-web-app.war
     discard-paths: yes
   ```

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

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

4. Upload this contents of the `my-web-app` directory to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

   Important
   Do not upload *`(root directory name)`* or *`(root directory name)`*`/my-web-app`, just the directories and files inside of *`(root directory name)`*`/my-web-app`.
   If you are using an Amazon 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 inside of *(root directory name)*/my-web-app. For example, the ZIP file should contain these directories and files:

```
WebArchiveHelloWorldSample.zip
     |-- buildpsec.yml
     |-- pom.xml
     `-- src
           `-- main
                 |-- resources
                 `-- webapp
                       |-- WEB-INF
                       |     `-- web.xml
                       `-- index.jsp
```

5. Create a build project, run the build, and view related build information by following the steps in Run AWS CodeBuild Directly (p. 38).

   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-web-archive-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/WebArchiveHelloWorldSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "WebArchiveHelloWorldOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/java:openjdk-8",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

6. To get the build output artifact, open your Amazon S3 output bucket.
7. Download the *WebArchiveHelloWorldOutputArtifact*.zip file to your local computer or instance, and then extract the contents of the file. In the extracted contents, open the `target` folder to get the `my-web-app.war` file.

Using the AWS CodeBuild Jenkins Plugin

Jenkins is a continuous integration and continuous delivery application that you can use to build and test your software projects continuously. For more information, see Meet Jenkins on the Jenkins website.

At a functional level, there are two components to Jenkins: a scheduler that creates and runs your build jobs; and a build platform, namely, a set of distributed build nodes. For more information, see Distributed builds on the Jenkins website.

The AWS CodeBuild Jenkins Plugin enables you to integrate AWS 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 AWS CodeBuild instead. This eliminates the need for you to provision, configure, and manage Jenkins build nodes.

For instructions on how to download, install, configure, and run the plugin from within Jenkins, see the awslabs/aws-codebuild-jenkins-plugin repository on GitHub.

Logging AWS CodeBuild API Calls with AWS CloudTrail

AWS CodeBuild is integrated with CloudTrail, a service that captures API calls made by or on behalf of AWS CodeBuild in your AWS account and delivers the log files to an Amazon S3 bucket you specify. CloudTrail captures API calls from the AWS CodeBuild console, the AWS CLI, the AWS SDKs, and the AWS CodeBuild HTTP API. Using the information collected by CloudTrail, you can determine which request was made to AWS CodeBuild, the source IP address from which the request was made, who made the request, when it was made, and so on. To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

AWS CodeBuild Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, calls made to AWS CodeBuild actions are tracked in log files. AWS CodeDeploy records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

All of the AWS CodeBuild actions are logged and documented in the Command Line Reference (p. 90) and the HTTP API Reference (p. 93). For example, calls to create build projects and run builds generate entries in CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, see the `userIdentity` field in the CloudTrail Event Reference.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, Amazon S3 server-side encryption (SSE) is used to encrypt your log files.

You can have CloudTrail publish Amazon SNS notifications when new log files are delivered. For more information, see Configuring Amazon SNS Notifications for CloudTrail.

You can also aggregate AWS CodeBuild log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see Receiving CloudTrail Log Files from Multiple Regions.

Understanding AWS CodeBuild Log File Entries

CloudTrail log files can contain one or more log entries where each entry is made up of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order. That is, they are not an ordered stack trace of the public calls.

The following example shows a CloudTrail log entry that demonstrates creating a build project in AWS CodeBuild:

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "FederatedUser",
    "principalId": "account-ID:user-name",
    "arn": "arn:aws:sts::account-ID:federated-user/user-name",
    "accountId": "account-ID",
    "accessKeyId": "access-key-ID",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2016-09-06T17:59:10Z"
      },
      "sessionIssuer": {
        "type": "IAMUser",
        "principalId": "access-key-ID",
        "arn": "arn:aws:iam::account-ID:user/user-name",
        "accountId": "account-ID",
        "userName": "user-name"
      }
    }
  },
  "eventTime": "2016-09-06T17:59:11Z",
  "eventSource": "codebuild.amazonaws.com",
  "eventName": "CreateProject",
  "awsRegion": "region-ID",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "user-agent",
  "requestParameters": {
    "awsActId": "account-ID"
  },
  "responseElements": {
    "project": {
      "environment": {
        "image": "image-ID",
        "computeType": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
      },
      "name": "codebuild-demo-project",
      "description": "This is my demo project",
      "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project:project-ID",
      "encryptionKey": "arn:aws:kms:region-ID:key-ID",
      "timeoutInMinutes": 10,
      "artifacts": {
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-
bucket",
        "type": "S3",
        "packaging": "ZIP",
        "outputName": "MyOutputArtifact.zip"
      },
      "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
      "lastModified": "Sep 6, 2016 10:59:11 AM",
      "source": {
        "type": "GITHUB",
        "location": "https://github.com/my-repo.git"
      },
      "created": "Sep 6, 2016 10:59:11 AM"
    }
  },
  "requestID": "9d32b228-745b-11e6-98bb-23b67EXAMPLE",
  "eventID": "581f7dd1-8d2e-40b0-aeee-0dbf7EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "account-ID"
}
```

Troubleshooting AWS CodeBuild

Use the information in this topic to help you identify, diagnose, and address issues as you use AWS CodeBuild.

Topics

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 following 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 AWS 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 "To activate or deactivate AWS STS in a region" in Activating and Deactivating AWS STS in an AWS Region in the *IAM User Guide*.
- Make sure the target AWS CodeBuild service role exists in your AWS account. If you are not using the console, make sure you did not misspell the service role's Amazon Resource Name (ARN) when creating or updating the build project.
- Make sure the target AWS CodeBuild service role has sufficient permissions to trust AWS CodeBuild. For more information, see the trust relationship policy statement as described in Create an AWS CodeBuild Service Role (p. 156).

Limits for AWS CodeBuild

The following tables list the current limits in AWS CodeBuild. These limits are for each supported AWS region for each AWS account, unless otherwise specified.

Build Projects

Resource	Default limit
Maximum number of build projects	1,000
Length of a build project name	2 to 255 characters, inclusive
Allowed characters in a build project name	The letters `A-Z` and `a-z`, the numbers `0-9`, and the special characters `-` and `_`
Maximum length of a build project description	255 characters
Allowed characters in a build project description	Any
Maximum number of build projects you can request information about at any one time by using the AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API	100
Maximum number of tags you can associate with a build project	50
Number of minutes you can specify in a build project for the build timeout of all related builds	5 to 480 (8 hours)

Builds

Resource	Default limit
Maximum number of concurrent running builds	20

Resource	Default limit
Maximum number of builds you can request information about at any one time by using the AWS CLI, AWS SDKs, or AWS CodeBuild HTTP API	100
Number of minutes you can specify for the build timeout of a single build	5 to 480 (8 hours)

Setting Up AWS CodeBuild

If you follow the steps in Getting Started (p. 6) to access AWS CodeBuild for the first time, most likely you will not need to reference the information in this topic. However, as you continue using AWS CodeBuild, you will want to do things such as give IAM groups and users in your organization access to AWS CodeBuild, modify existing service roles in IAM or customer master keys in AWS KMS to access AWS CodeBuild, or set up the AWS CLI across your organization's workstations to access AWS 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 AWS 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 AWS 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 AWS CodeBuild access permissions to an IAM group or IAM user (console)

1. Open the Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam/.

   You should have already signed in to the AWS Management Console by using one of the following:

   - Your AWS root account. This is not recommended. For more information, see The 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:

```
iam:AttachGroupPolicy
iam:AttachUserPolicy
iam:CreatePolicy
iam:ListAttachedGroupPolicies
iam:ListAttachedUserPolicies
iam:ListGroups
iam:ListPolicies
iam:ListUsers
```

  For more information, see Overview of IAM Policies in the *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 AWS CodeBuild access permissions to an IAM group or IAM user, choose **Policy Type**, **AWS Managed**, and then do the following:

   - To add full acess permissions to AWS CodeBuild, select the box named **AWSCodeBuildAdminAccess**. Then choose **Policy Actions**, **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 AWS CodeBuild for everything except build project administration, select the box named **AWSCodeBuildDeveloperAccess**. Then choose **Policy Actions**, **Attach**. Select the box next to the target IAM group or IAM user, and then choose **Attach Policy**. Repeat this for the policy named **AmazonS3ReadOnlyAccess**.
   - To add read-only access permissions to AWS 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 AWS 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**, type a name for the policy (for example, **`CodeBuildAccessPolicy`**). If you use a different name, substitute it throughout this procedure.
7. For **Policy Document**, type the following, and then choose **Create Policy**.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildDefaultPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*",
        "iam:PassRole"
      ],
      "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": "*"
    }
  ]
}
```

Note
This policy allows access to all AWS CodeBuild actions and to a potentially large number of AWS resources. To restrict permissions to specific AWS CodeBuild actions, change the value of `codebuild:*` in the AWS CodeBuild policy statement. For more information, see Authentication and Access Control (p. 96). To restrict access to specific AWS resources, change the value of the `Resource` object. For more information, see Authentication and Access Control (p. 96).

8. In the navigation pane, choose **Groups** or **Users**.
9. In the list of groups, choose the name of the IAM group or IAM user to which you want to add AWS CodeBuild access permissions.
10. On the group settings page or user settings page, on the **Permissions** tab, expand **Managed Policies**, and then choose **Attach Policy**.
11. On the **Attach Policy** page, select **CodeBuildAccessPolicy**, and then choose **Attach Policy**.

To add AWS 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*.
2. To add a custom set of AWS CodeBuild access permissions to an IAM group or IAM user, skip ahead to step 3 in this procedure.

   To add a default set of AWS 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:

```
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 acess permissions to AWS 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 AWS 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 AWS 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 AWS 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, substitute it throughout this procedure.

```
{
  "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:CreateBucket",
        "s3:GetObject",
        "s3:List*",
        "s3:PutObject"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
> This policy allows access to all AWS CodeBuild actions and to a potentially large number of AWS resources. To restrict permissions to specific AWS CodeBuild actions, change the value of `codebuild:*` in the AWS CodeBuild policy statement. For more information, see Authentication and Access Control (p. 96). To restrict access to specific AWS resources, change the value of the related `Resource` object. For more information, see Authentication and Access Control (p. 96) 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, substitute them here.

   For an IAM group:

   ```
   aws iam put-group-policy --group-name group-name --policy-name
    CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json
   ```

   For an IAM user:

   ```
   aws iam put-user-policy --user-name user-name --policy-name
    CodeBuildUserAccessPolicy --policy-document file://put-user-policy.json
   ```

   In the preceding commands, replace *`group-name`* or *`user-name`* with the name of the target IAM group or IAM user.

Create an AWS CodeBuild Service Role

You need an AWS CodeBuild service role so that AWS CodeBuild can interact with dependent AWS services on your behalf. You can create an AWS CodeBuild service role by using the AWS CodeBuild or AWS CodePipeline consoles. For information, see:

- Create a Build Project (Console) (p. 40)
- Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console) (p. 28)
- Add an AWS CodeBuild Build Action to a Pipeline (AWS CodePipeline Console) (p. 34)
- Change a Build Project's Settings (Console) (p. 54)

If you do not plan to use these consoles, this section describes how to create an AWS CodeBuild service role with the IAM console or the AWS CLI.

To create an AWS CodeBuild service role (console)

1. Open the Identity and Access Management (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:

```
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, next to **Create Your Own Policy**, choose **Select**.
5. On the **Review Policy** page, for **Policy Name**, type a name for the policy (for example, **`CodeBuildServiceRolePolicy`**). If you use a different name, substitute it throughout this procedure.
6. For **Policy Document**, type the following, and then choose **Create Policy**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "S3PutObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

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.

7. In the navigation pane, choose **Roles**.
8. Choose **Create New Role**.
9. On the **Set Role Name** page, for **Role Name**, type a name for the service role (for example, **`CodeBuildServiceRole`**). If you use a different name, substitute it throughout this procedure.
10. Choose **Next Step**.
11. On the **Select Role Type** page, with **AWS Service Roles** already selected, next to **Amazon EC2**, choose **Select**.

    Note
    You must choose a role type, such as **Amazon EC2**, to proceed. This is because an AWS CodeBuild role type is not yet available on this page. You will change this role type setting later in this procedure.
12. On the **Attach Policy** page, select **CodeBuildServiceRolePolicy**, and then choose **Next Step**.
13. Choose **Create Role**.
14. In the list of roles, choose **CodeBuildServiceRole**.
15. On the **Trust Relationships** tab, choose **Edit Trust Relationship**.
16. On the **Edit Trust Relationship** page, completely replace the contents of the **Policy Document** box with the following, and then choose **Update Trust Policy**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Note
This policy statement establishes a trust relationship with AWS CodeBuild in all supported AWS regions. You can establish a trust relationship with AWS CodeBuild in fewer supported AWS regions. To do this, completely replace the contents of the **Policy Document** box with the following policy statement, remove the principals for the AWS regions where you don't want to establish a trust relationship with AWS CodeBuild, and then choose **Update Trust Policy**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codebuild.us-east-1.amazonaws.com",
          "codebuild.us-west-2.amazonaws.com",
          "codebuild.eu-west-1.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

To create an AWS 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, substitute them throughout this procedure.

   `create-role.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Note

This policy statement establishes a trust relationship with AWS CodeBuild in all supported AWS regions. You can establish a trust relationship with AWS CodeBuild in fewer supported AWS regions. To do this, use the following policy statement, removing the principals for the AWS regions where you don't want to establish a trust relationship with AWS CodeBuild:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codebuild.us-east-1.amazonaws.com",
          "codebuild.us-west-2.amazonaws.com",
          "codebuild.eu-west-1.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

`put-role-policy.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "S3PutObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

> **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 substitute them here.

```
aws iam create-role --role-name CodeBuildServiceRole --assume-role-policy-
document file://create-role.json
```

```
aws iam put-role-policy --role-name CodeBuildServiceRole --policy-name
 CodeBuildServiceRolePolicy --policy-document file://put-role-policy.json
```

Create and Configure an AWS KMS CMK for AWS CodeBuild

For AWS CodeBuild to encrypt its build output artifacts, it needs access to an AWS KMS customer master key (CMK). By default, AWS 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 AWS CodeBuild, follow the instructions in the "How to Modify a Key Policy" section of Modifying a Key Policy in the *AWS KMS Developer Guide*. Then add the following statements (between *`### BEGIN ADDING STATEMENTS HERE ###`* and *`### END ADDING STATEMENTS HERE ###`*) to the key policy. Ellipses (`. . .`) are used for brevity and to help you locate where to add the statements. Do not remove any statements, and do not type these ellipses into the key policy.

```
{
  "Version": "2012-10-17",
  "Id": "...",
  "Statement": [
    ### BEGIN ADDING STATEMENTS HERE ###
    {
      "Sid": "Allow access through Amazon S3 for all principals in the
 account that are authorized to use Amazon S3",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.region-ID.amazonaws.com",
          "kms:CallerAccount": "account-ID"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-ID:role/CodeBuild-service-role"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    ### END ADDING STATEMENTS HERE ###
    {
      "Sid": "Enable IAM User Permissions",
      ...
    },
    {
      "Sid": "Allow access for Key Administrators",
      ...
    },
    {
      "Sid": "Allow use of the key",
      ...
    },
    {
      "Sid": "Allow attachment of persistent resources",
      ...
    }
  ]
}
```

- *region-ID* represents the ID of the AWS region where the Amazon S3 buckets associated with AWS CodeBuild are located (for example, `us-east-1`).
- *account-ID* represents the ID of the of the AWS account that owns the CMK.
- *CodeBuild-service-role* represents the name of the AWS 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 AWS CodeBuild console, the AWS CodePipeline console, the AWS SDKs, or the AWS CodeBuild HTTP API. 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 AWS CodeBuild:

   ```
   aws codebuild list-builds
   ```

   If successful, information similar to the following will appear in the output:

   ```
   {
     "ids": []
   }
   ```

   The empty square brackets indicate that you have not yet run any builds.
2. If an error is output, you must uninstall your current version of the AWS CLI and then install the latest version. For more information, see Uninstalling the AWS CLI and Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

AWS CodeBuild User Guide Document History

Here is a list of important changes to the *AWS CodeBuild User Guide*.

- **Latest API version:** 2016-10-06
- **Latest documentation update:** December 5, 2016

Change	Description	Date Changed
Troubleshooting topic added	Troubleshooting information is now available. For more information, see Troubleshooting AWS CodeBuild (p. 149).	December 5, 2016
Jenkins plugin initial release	This is the initial release of the AWS CodeBuild Jenkins Plugin. For more information, see Using the AWS CodeBuild Jenkins Plugin (p. 145).	December 5, 2016
User Guide initial release	This is the initial release of the *AWS CodeBuild User Guide*.	December 1, 2016

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.