# Table of Contents

What Is AWS CodeBuild? .......................................................................................................................................... 1  
  How to Run AWS CodeBuild ........................................................................................................................... 1  
  Pricing for AWS CodeBuild ............................................................................................................................ 2  
  How Do I Get Started with AWS CodeBuild? .................................................................................................. 2  
  Concepts ......................................................................................................................................................... 2  
  How AWS CodeBuild Works ............................................................................................................................ 3  
  Next Steps ...................................................................................................................................................... 3  

Getting Started ............................................................................................................................................. 4  
  Step 1: Create or Use Amazon S3 Buckets to Store the Build Input and Output ............................................ 4  
  Step 2: Create the Source Code to Build ....................................................................................................... 5  
  Step 3: Create the Build Spec ....................................................................................................................... 6  
  Step 4: Add the Source Code and the Build Spec to the Input Bucket ......................................................... 8  
  Step 5: Create the Build Project .................................................................................................................... 9  
  Step 6: Run the Build .................................................................................................................................... 12  
  Step 7: View Summarized Build Information ............................................................................................... 14  
  To view summarized build information (console) ....................................................................................... 14  
  To view summarized build information (AWS CLI) ................................................................................... 14  
  Step 8: View Detailed Build Information .................................................................................................... 16  
  Step 9: Get the Build Output Artifact ......................................................................................................... 18  
  Step 10: Clean Up ........................................................................................................................................ 18  
  Next Steps ..................................................................................................................................................... 19  

Samples ...................................................................................................................................................... 20  
  Use Case-Based Samples ................................................................................................................................ 21  
    Amazon ECR Sample .................................................................................................................................... 22  
    Docker Sample ........................................................................................................................................... 24  
    GitHub Enterprise Sample .......................................................................................................................... 30  
    GitHub Pull Request Sample ....................................................................................................................... 35  
    Use AWS Config with AWS CodeBuild Sample ....................................................................................... 37  
    Build Badges Sample ................................................................................................................................. 39  
    Build Notifications Sample ...................................................................................................................... 41  
    Docker in Custom Image Sample .............................................................................................................. 58  
    AWS CodeDeploy Sample ......................................................................................................................... 60  
    AWS Lambda Sample .............................................................................................................................. 63  
    Elastic Beanstalk Sample ......................................................................................................................... 64  
  Code-Based Samples ..................................................................................................................................... 70  
    C++ Sample ................................................................................................................................................ 71  
    Go Sample ................................................................................................................................................ 73  
    Maven Sample .......................................................................................................................................... 75  
    Node.js Sample ....................................................................................................................................... 77  
    Python Sample .......................................................................................................................................... 79  
    Ruby Sample ............................................................................................................................................ 81  
    Scala Sample ........................................................................................................................................... 83  
    Java Sample ............................................................................................................................................. 87  
    .NET Core in Linux Sample ..................................................................................................................... 89  

Plan a Build .................................................................................................................................................. 92  
  Build Spec Reference ..................................................................................................................................... 93  
    Build Spec File Name and Storage Location ............................................................................................ 95  
    Build Spec Syntax ..................................................................................................................................... 94  
    Build Spec Example .................................................................................................................................. 98  
    Build Spec Versions .................................................................................................................................. 100  
  Build Environment Reference ...................................................................................................................... 100  
    Docker Images Provided by AWS CodeBuild ......................................................................................... 101  
    Build Environment Compute Types ....................................................................................................... 107  
    Shells and Commands in Build Environments ....................................................................................... 108  

API Version 2016-10-06
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](https://aws.amazon.com/codebuild).

### Topics
- How to Run AWS CodeBuild (p. 1)
- Pricing for AWS CodeBuild (p. 2)
- How Do I Get Started with AWS CodeBuild? (p. 2)
- AWS CodeBuild Concepts (p. 2)

## 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.

To run AWS CodeBuild by using the AWS CodeBuild console, AWS CLI, AWS SDKs, see [Run AWS CodeBuild Directly](p. 111).

As the following diagram shows, you can add AWS CodeBuild as a build or test action to the build or test stage of a pipeline in AWS CodePipeline. AWS CodePipeline is a continuous delivery service that 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 or test action, see [Use AWS CodePipeline with AWS CodeBuild](p. 123). 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. 2).
2. **Experiment** with AWS CodeBuild in an example scenario by following the instructions in [Getting Started](p. 4).
3. **Use** AWS CodeBuild in your own scenarios by following the instructions in [Plan a Build](p. 92).

### AWS CodeBuild Concepts

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

**Topics**

- [How AWS CodeBuild Works](p. 3)
- [Next Steps](p. 3)
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. 141)
   - Build Environment Reference (p. 100)
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. 93).
4. If there is any build output, 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. 41).
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, or AWS SDKs, 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. 4).
2. Use AWS CodeBuild in your own scenarios by following the instructions in Plan a Build (p. 92).
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 (p. 4)
- Step 2: Create the Source Code to Build (p. 5)
- Step 3: Create the Build Spec (p. 6)
- Step 4: Add the Source Code and the Build Spec to the Input Bucket (p. 8)
- Step 5: Create the Build Project (p. 9)
- Step 6: Run the Build (p. 12)
- Step 7: View Summarized Build Information (p. 14)
- Step 8: View Detailed Build Information (p. 16)
- Step 9: Get the Build Output Artifact (p. 18)
- Step 10: Clean Up (p. 18)
- Next Steps (p. 19)

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 (Ohio) region, then these buckets must also be in the US East (Ohio) 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, GitHub, and Bitbucket repositories, this walkthrough does not show you how to use them. For more information, see Plan a Build (p. 92).

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.

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

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

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

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

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

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

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

public class TestMessageUtil {
    String message = "Robert";
    MessageUtil messageUtil = new MessageUtil(message);

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

API Version 2016-10-06
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.

```yaml
version: 0.2
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.

**Note**

Instead of including a build spec file in your source code, you can declare build commands separately when you create a build project. This is helpful if you want to build your source code with different build commands without updating your source code's repository each time. For more information, see Build Spec Syntax (p. 94).

In this build spec declaration:

- **version** represents the version of the build spec standard being used. This build spec declaration uses the latest version, `0.2`.
- **phases** represents the build phases during which you can instruct 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 compiled Java class files into a build output artifact. For completeness, a few `echo` commands are placed in each build phase in this example. When you view detailed build information later in this 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. 93).
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, GitHub, and Bitbucket repositories, by convention, you must store a build spec file named `buildspec.yml` in the root (top level) of each repository or include the build spec declaration as part of the build project definition. Do not create a ZIP file that contains the repository's source code and build spec file.

For build input stored in Amazon S3 buckets only, you must create a ZIP file that contains the source code and, by convention, a build spec file named `buildspec.yml` at the root (top level) or include the build spec declaration as part of the build project definition.

If you want to use a different name for your build spec file, or you want to reference a build spec in a location other than the root, you can specify a build spec override as part of the build project definition. For more information, see Build Spec File Name and Storage Location (p. 93).
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 Docker Overview 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. 9) or with the AWS CLI (p. 10).

Note
You can work with AWS CodeBuild in several ways: through the AWS CodeBuild console, AWS CodePipeline, the AWS CLI, or the AWS SDKs. 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. 123). To learn how to use the AWS SDKs, see Run AWS CodeBuild Directly (p. 111).

To create the build project (console)

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. 192).

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.
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.

Skip ahead to Step 6: Run the Build (p. 12).

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:

```json
{
    "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. 179).

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. 100).

• 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. 146).

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.

```json
{
  "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",
      "API Version 2016-10-06
      11"```
"type": "LINUX_CONTAINER",
"environmentVariables": []
},
"source": {
"type": "S3",
"location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
},
"arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-project"
}

- project represents information about this build project.
- tags represents any tags that were declared.
- packaging represents how the build output artifact 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. 192).

### 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. 13) or the AWS CLI (p. 13).
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.

To run the build (AWS CLI)

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

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

   Replace `project-name` with your build project name from the previous step (for example, codebuild-demo-project).

2. If successful, data similar to the following will appear in the output:

   ```json
   {
     "build": {
       "buildComplete": false,
       "initiator": "user-name",
       "artifacts": {
         "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/message-util.zip"
       },
       "projectName": "codebuild-demo-project",
       "timeoutInMinutes": 60,
       "buildStatus": "IN_PROGRESS",
       "environment": {
         "computeType": "BUILD_GENERAL1_SMALL",
         "image": "aws/codebuild/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.

API Version 2016-10-06
13
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. 14) or the AWS CLI (p. 14).

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. 16).

To view summarized build information (AWS CLI)

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

```bash
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.
To view summarized build information (AWS CLI)

```json
{
    "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=38calc4a-e9ca-4dcb-bef1-d52bfEXAMPLE",
                "streamName": "38calc4a-e9ca-4dcb-bef1-d52bfEXAMPLE"
            },
            "artifacts": {
                "md5sum": "MD5-hash",
                "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:38calc4a-e9ca-4dcb-bef1-d52bfEXAMPLE",
            "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-project:38calc4a-e9ca-4dcb-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. 16) or the AWS CLI (p. 16).

**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. 18).

**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=38calc4a-e9ca-4dcb-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.

```plaintext
...
[Container] 2016/04/15 17:49:42 Entering phase PRE_BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Entering phase BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering build phase...
[Container] 2016/04/15 17:49:42 Entering build phase...
[Container] 2016/04/15 17:49:42 Running command mvn install
[Container] 2016/04/15 17:49:44 [INFO] Scanning for projects...
[Container] 2016/04/15 17:49:44 [INFO]
[Container] 2016/04/15 17:49:44 [INFO]--------------------------------------------------------
[Container] 2016/04/15 17:49:44 [INFO]

... [Container] 2016/04/15 17:49:55 -------------------------------------------------------
[Container] 2016/04/15 17:49:55 -------------------------------------------------------
[Container] 2016/04/15 17:49:55 HiRobert
[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 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
... [Container] 2016/04/15 17:49:55 Results :
[Container] 2016/04/15 17:49:55

... [Container] 2016/04/15 17:49:55 --------------------------------------------------------

... [Container] 2016/04/15 17:49:55 Phase complete: BUILD Success: true
[Container] 2016/04/15 17:49:55 Running command echo Entering post_build phase...
[Container] 2016/04/15 17:49:55 Entering post_build phase...
[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

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. 18) or the Amazon S3 console (p. 18).

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. 18).

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. 92). If you don't feel ready yet, you might want to try building some of our samples. For more information, see Samples (p. 20).
AWS CodeBuild Samples

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon ECR Sample (p. 22)</td>
<td>Uses a Docker image in an Amazon ECR repository to use Apache Maven to produce a single JAR file.</td>
</tr>
<tr>
<td>Docker Sample (p. 24)</td>
<td>Uses a build image provided by AWS CodeBuild with Docker support to produce a Docker image with Apache Maven. Pushes the Docker image to a repository in Amazon ECR. You can adapt this sample to push the Docker image to Docker Hub.</td>
</tr>
<tr>
<td>?? (p. 30)</td>
<td>Uses AWS CodeBuild with GitHub Enterprise as the source repository, with certificates installed and webhooks enabled, to rebuild the source code every time a code change is pushed to the repository.</td>
</tr>
<tr>
<td>GitHub Pull Request Sample (p. 35)</td>
<td>Uses AWS CodeBuild with GitHub as the source repository, and webhooks enabled, to rebuild the source code every time a code change is pushed to the repository.</td>
</tr>
<tr>
<td>Use AWS Config with AWS CodeBuild Sample (p. 37)</td>
<td>Shows how to set up AWS Config. Lists which AWS CodeBuild resources are tracked and describes how to look up AWS CodeBuild projects in AWS Config.</td>
</tr>
<tr>
<td>Build Badges Sample (p. 39)</td>
<td>Shows how to set up AWS CodeBuild with build badges.</td>
</tr>
<tr>
<td>Build Notifications Sample (p. 41)</td>
<td>Uses Apache Maven to produce a single JAR file. Sends a build notification to subscribers of an Amazon SNS topic.</td>
</tr>
<tr>
<td>Docker in Custom Image Sample (p. 58)</td>
<td>Uses a custom Docker image to produce a Docker image.</td>
</tr>
<tr>
<td>AWS CodeDeploy Sample (p. 60)</td>
<td>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.</td>
</tr>
<tr>
<td>AWS Lambda Sample (p. 63)</td>
<td>Uses AWS CodeBuild along with Lambda, AWS CloudFormation, and AWS CodePipeline to build and deploy a serverless application that follows the AWS Serverless Application Model (AWS SAM) standard.</td>
</tr>
<tr>
<td>Elastic Beanstalk Sample (p. 64)</td>
<td>Uses Apache Maven to produce a single WAR file. Uses Elastic Beanstalk to deploy the WAR file to an Elastic Beanstalk instance.</td>
</tr>
</tbody>
</table>
You can use these code-based samples to experiment with AWS CodeBuild:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++ Sample (p. 71)</td>
<td>Uses C++ to output a single .out file.</td>
</tr>
<tr>
<td>Go Sample (p. 73)</td>
<td>Uses Go to output a single binary file.</td>
</tr>
<tr>
<td>Maven Sample (p. 75)</td>
<td>Uses Apache Maven to produce a single JAR file.</td>
</tr>
<tr>
<td>Node.js Sample (p. 77)</td>
<td>Uses Mocha to test whether an internal variable in code contains a specific string value. Produces a single .js file.</td>
</tr>
<tr>
<td>Python Sample (p. 79)</td>
<td>Uses Python to test whether an internal variable in code is set to a specific string value. Produces a single .py file.</td>
</tr>
<tr>
<td>Ruby Sample (p. 81)</td>
<td>Uses RSpec to test whether an internal variable in code is set to a specific string value. Produces a single .rb file.</td>
</tr>
<tr>
<td>Scala Sample (p. 83)</td>
<td>Uses sbt to produce a single JAR file.</td>
</tr>
<tr>
<td>Java Sample (p. 87)</td>
<td>Uses Apache Maven to produce a single WAR file.</td>
</tr>
<tr>
<td>.NET Core in Linux Sample (p. 89)</td>
<td>Uses .NET Core to build an executable file out of code written in C#.</td>
</tr>
</tbody>
</table>

AWS CodeBuild Use Case-Based Samples

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon ECR Sample (p. 22)</td>
<td>Uses a Docker image in an Amazon ECR repository to use Apache Maven to produce a single JAR file.</td>
</tr>
<tr>
<td>Docker Sample (p. 24)</td>
<td>Uses a build image provided by AWS CodeBuild with Docker support to produce a Docker image with Apache Maven. Pushes the Docker image to a repository in Amazon ECR. You can also adapt this sample to push the Docker image to Docker Hub.</td>
</tr>
<tr>
<td>GitHub Enterprise Sample (p. 30)</td>
<td>Uses AWS CodeBuild with GitHub Enterprise as the source repository, with certificates installed and webhooks enabled, to rebuild the source code every time a code change is pushed to the repository.</td>
</tr>
<tr>
<td>GitHub Pull Request Sample (p. 35)</td>
<td>Uses AWS CodeBuild with GitHub as the source repository, and webhooks enabled, to rebuild the source code every time a code change is pushed to the repository.</td>
</tr>
<tr>
<td>Use AWS Config with AWS CodeBuild Sample (p. 37)</td>
<td>Shows how to set up AWS Config. Lists which AWS CodeBuild resources are tracked and describes</td>
</tr>
</tbody>
</table>
### Amazon ECR Sample for AWS CodeBuild

This sample uses a Docker image in an Amazon Elastic Container Registry (Amazon ECR) image repository to build the [Maven Sample](#) 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 Elastic Container Registry Pricing](#).

### Running the Sample

To run this sample:

1. To create and push the Docker image to your image repository in Amazon ECR, complete the steps in the Running the Sample section of the [Docker Sample](#).
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](#).
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:GetRepositoryPolicy",
                "ecr:SetRepositoryPolicy"
            ],
            "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 **codebuild.amazonaws.com**. Leave **Everybody** cleared because you want to allow access to AWS CodeBuild only.
9. Skip the **All IAM entities** list.
10. For **Action**, select **Pull only actions**.

   All of the pull-only actions (**ecr:GetDownloadUrlForLayer**, **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": {
                "Service": "codebuild.amazonaws.com"
            },
            "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. 111).

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

   ```json
   { "name": "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-2.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.

**Related Resources**

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

**Docker Sample for AWS CodeBuild**

This sample produces as build output a Docker image and then pushes the Docker image to an Amazon Elastic Container Registry (Amazon ECR) image repository. You can adapt this sample to push the Docker image to Docker Hub. For more information, see Adapting the Sample to Push the Image to Docker Hub (p. 28).

To learn how to build a Docker image by using a custom Docker build image instead (`docker:dind` in Docker Hub), see our Docker in Custom Image Sample (p. 58).

This sample was tested referencing `golang:1.9`

This sample uses the new multi-stage Docker builds feature, which produces a Docker image as build output. It then pushes the Docker image to an Amazon ECR image repository. Multi-stage Docker image
AWS CodeBuild User Guide
Docker Sample

builds help to reduce the size of the final Docker image. For more information, see Use multi-stage builds with Docker.

Important
Running this sample 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 Elastic Container Registry Pricing.

Topics
• Running the Sample (p. 25)
• Directory Structure (p. 27)
• Files (p. 27)
• Adapting the Sample to Push the Image to Docker Hub (p. 28)
• Related Resources (p. 30)

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.

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

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

2. Create an image repository in Amazon ECR. Be sure to create the repository in the same AWS region where you 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 (\ldots) are used for brevity and to help you locate where to add the statement. Do not remove any statements, and do not type these ellipses into the policy.

```json
{
 "Statement": [
 
 ### BEGIN ADDING STATEMENT HERE ###

 { 
   "Action": [
     "ecr:BatchCheckLayerAvailability",
     "ecr:CompleteLayerUpload",
     "ecr:GetAuthorizationToken",
     "ecr:InitiateLayerUpload",
     "ecr:PutImage",
     "ecr:UploadLayerPart"
   ],
   "Resource": "*",
   "Effect": "Allow"
 },

 ### END ADDING STATEMENT HERE ###

..., 

 "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, GitHub, or Bitbucket 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. 111).

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

```json
{
 "name": "sample-docker-project",
 "source": {
   "type": "S3",
   "location": "codebuild-region-ID-account-ID-input-bucket/DockerSample.zip"
 },
 "artifacts": {
   "type": "NO_ARTIFACTS"
 },
 "environment": {
   "type": "LINUX_CONTAINER",
   "image": "aws/codebuild/docker:17.09.0",
   "computeType": "BUILD_GENERAL1_SMALL",
   "environmentVariables": {
     "name": "AWS_DEFAULT_REGION",
     "value": "region-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))`

**Note**

If you are using Docker prior to version 17.06, remove the --no-include-email option.

```yaml
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
      #AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
```
Dockerfile (in *(root directory name)*)

FROM golang:1.9 as builder
RUN go get -d -v golang.org/x/net/html
RUN go get -d -v github.com/alexellis/href-counter/
WORKDIR /go/src/github.com/alexellis/href-counter/.
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis/href-counter/app .
CMD ["./app"]

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:

   **Note**
   If you are using Docker prior to version 17.06, remove the --no-include-email option.

   ```
   ... 
   pre_build: 
   commands: 
   - echo Logging in to Amazon ECR...  
   - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
   build: 
   commands: 
   - echo Build started on `date`
   - echo Building the Docker image...  
   - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
   - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
   post_build: 
   commands: 
   - echo Build completed on `date`
   - echo Pushing the Docker image...  
   - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
   ...
   ```

2. With these Docker Hub-specific lines of code.

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

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

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

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

4. Follow the steps in Run AWS CodeBuild Directly (p. 111) 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.
GitHub Enterprise Sample for AWS CodeBuild

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

Prerequisites

1. Generate a personal access token that will be entered into your AWS CodeBuild project. We recommend that you create a new GitHub Enterprise user and generate a personal access token for this user. Copy it to your clipboard so that it can be used when you create your AWS CodeBuild project. For more information, see Creating a Personal Access Token in GitHub Enterprise on the GitHub Help website.

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

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

Linux/macOS clients:

From a terminal window, run the following command:

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

Replace the placeholders in the command with the following values:

- **HOST**: The IP address of your GitHub Enterprise repository.
- **PORTNUMBER**: The port number you are using to connect (for example, 443).
- **folder**: The folder where you downloaded your certificate.
- **filename**: The file name of your certificate file.

Important

Save the certificate as a .pem file.

Windows clients:
Download your certificate from GitHub Enterprise using your browser. To see the site's certificate details, choose the padlock icon. For information about how to export the certificate, see your browser documentation.

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

3. Upload your certificate file to an Amazon S3 bucket. For information about how to create an Amazon S3 bucket, see [How Do I Create an Amazon S3 Bucket?](#) For information about how to upload objects to an Amazon S3 bucket, see [How Do I Upload Files and Folders to a Bucket?](#)

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

### Create a Build Project with GitHub Enterprise as the Source Repository and Enable Webhooks (Console)


2. If a 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. In **Source: What to build**, for **Source provider**, choose **GitHub Enterprise**.

   - For **Personal Access Token**, paste the token you copied to your clipboard and choose **Save Token**. In **Repository URL**, enter the URL for your GitHub Enterprise repository.

     **Note**
     You only need to enter and save the personal access token once. All future AWS CodeBuild projects will use this token.

   - Select **Webhook** to rebuild every time a code change is pushed to this repository.

   - Select **Insecure SSL** to ignore SSL warnings while connecting to your GitHub Enterprise project repository.

     **Note**
     We recommend that you use **Insecure SSL** for testing only. It should not be used in a production environment.
5. 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.

For Build specification, do one of the following:

- Use the buildspec.yml file in the source code root directory.
- Override the build specification by inserting the build commands.

For more information, see the Build Spec Reference (p. 93).

For Certificate, choose **Install certificate from your S3**. For Bucket of certificate, choose the S3 bucket where your SSL certificate is stored. For Object key of certificate, type the name of your S3 object key.

6. 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**.
- To store the build output in an Amazon S3 bucket, choose **Amazon S3**, and then do the following:
• If you want to use your project name for the build output ZIP file or folder, leave **Artifacts name** blank. Otherwise, type the name in the **Artifacts name** box. By default, the artifact name is the project name. If you want to specify a different name, type it in the artifacts name box. If you want to output a ZIP file, then include the zip extension.

• For **Bucket name**, choose the name of the output bucket.

• If you chose **Insert build commands** earlier 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. 94)**.

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

   • If you do not want to use a cache, choose **No cache**.
   • To use a cache, choose **Amazon S3**, and then do the following:
     • For **Bucket**, choose the name of the Amazon S3 bucket where the cache is stored.
     • (Optional) For **Path prefix**, type an Amazon S3 path prefix. The **Path prefix** value is similar to a directory name that enables you to store the cache under the same directory in a bucket.

     **Important**
     Do not append "/" to the end of **Path prefix**.

   Using a cache saves considerable build time because reusable pieces of the build environment are stored in the cache and used across builds.

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

   • If you do not have an AWS CodeBuild service role, choose **Create a service role in your account**. In **Role name**, accept the default name or type your own.
   • If you have an AWS CodeBuild service role, choose **Choose an service 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 is updated to work with the other build project. A service role can work with up to 10 build projects.

9. In **VPC**, do one of the following:

   • If you are not using a VPC for your project, choose **No VPC**.
   • If you are using want AWS CodeBuild to work with your VPC:
     • For **VPC**, choose the VPC ID that AWS CodeBuild uses.
     • For **Subnets**, choose the subnets that include resources that AWS CodeBuild uses.
     • For **Security Groups**, choose the security groups that AWS CodeBuild uses to allow access to resources in the VPCs.

   For more information, see **Use AWS CodeBuild with Amazon Virtual Private Cloud (p. 112)**.

10. Expand **Show advanced settings** and set them as appropriate.

11. Choose **Continue**. On the **Review** page, choose **Save and build** or, to run the build later, choose **Save**.

12. If you enabled webhooks in **Source: What to Build**, then a **Create webhook** dialog box appears with values displayed for **Payload URL** and **Secret**.
**Important**

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

![Create webhook dialog box](image)

13. In GitHub Enterprise, choose the repository where your AWS CodeBuild project is stored, choose Settings, choose Hooks & services, and then choose Add webhook. Enter the payload URL and secret key, accept the defaults for the other fields, and choose Add webhook.

GitHub Pull Request Sample for AWS CodeBuild

AWS CodeBuild now supports webhooks, when the source repository is GitHub. This means that for an AWS CodeBuild build project that has its source code stored in a private GitHub repository, webhooks enables AWS CodeBuild to begin automatically rebuilding the source code every time a code change is pushed to the private repository.

Create a Build Project with GitHub as the Source Repository and Enable Webhooks (Console)

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. In Source: What to build, for Source provider, choose GitHub. Follow the instructions to connect (or reconnect) with GitHub and choose Authorize.
For **Webhook**, select the **Rebuild every time a code change is pushed to this repository** check box. You can select this check box only if, under **Repository**, you chose **Use a repository in my account**.

5. **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.

   For **Build specification**, do one of the following:
   
   - Use the buildspec.yml file in the source code root directory.
   - Override the build specification by inserting the build commands.

   For more information, see the **Build Spec Reference (p. 93)**.

6. **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**.
   - To store the build output in an Amazon S3 bucket, choose **Amazon S3**, and then do the following:
     
     - If you want to use your project name for the build output ZIP file or folder, leave **Artifacts name** blank. Otherwise, type the name in the **Artifacts name** box. (By default, the artifact name is the project name. If you want to specify a different name, type it in the artifacts name box. If you want to output a ZIP file, then include the zip extension.
     - For **Bucket name**, choose the name of the output bucket.
     - If you chose **Insert build commands** earlier 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. 94)**.

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

   - If you do not have an AWS CodeBuild service role, choose **Create a service role in your account**. In **Role name**, accept the default name or type your own.
If you have an AWS CodeBuild service role, choose **Choose an service 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 is updated to work with the other build project. A service role can work with up to 10 build projects.

8. Expand **Show advanced settings** and set them as appropriate.

9. Choose **Continue**. On the **Review** page, choose **Save and build** or, to run the build later, choose **Save**.

**Verification Checks**

1. On your **AWS CodeBuild project** page, do the following:

   - Choose **Project Details** and then choose the **Webhook URL** link. In your GitHub repository, on the **Settings** page, under **Webhooks**, verify that **Pull Request** and **Push** are both selected.

2. In GitHub, under **Accounts**, **Settings**, **Authorized OAuth Apps**, you should see the AWS CodeBuild region that has been authorized.

**Use AWS Config with AWS CodeBuild Sample**

AWS Config provides an inventory of your AWS resources and a history of configuration changes to these resources. AWS Config now supports AWS CodeBuild as an AWS resource, which means the service can track your AWS CodeBuild projects. For more information about AWS Config, see **What Is AWS Config?** in the **AWS Config Developer Guide**.

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

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

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

**Topics**

- **Prerequisites** (p. 38)
- **Set Up AWS Config** (p. 38)
- **Look Up AWS CodeBuild Projects** (p. 38)
• Viewing AWS CodeBuild Configuration Details in the AWS Config Console (p. 38)

Prerequisites

Create your AWS CodeBuild project(s). For more information, see Create a Build Project (p. 141).

Set Up AWS Config

• Setting up AWS Config (Console)
• Setting up AWS Config (AWS CLI)

Note
It can take up to 10 minutes before a user is able to see AWS CodeBuild projects in the AWS Config console.

Look Up AWS CodeBuild Projects

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

Viewing AWS CodeBuild Configuration Details in the AWS Config Console

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

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

For more information, see Viewing Configuration Details in the AWS Config Console in the AWS Config Developer Guide.

Example of a AWS CodeBuild Project in AWS Config:
AWS CodeBuild now supports the use of build badges, which provide an embeddable, dynamically generated image (badge) that displays the status of the latest build for a project. This image is accessible through a publicly available URL generated for your AWS CodeBuild project. This allows anyone to view the status of an AWS CodeBuild project. Build badges do not contain any security information, so they do not require authentication.

Create a Build Project with Build Badges Enabled (Console)

2. If a 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. 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. Select the Build Badge check box to make your project's build status visible and embeddable.
   - If you chose GitHub, follow the instructions to connect (or reconnect) with GitHub. On the GitHub Authorize application page, for Organization access, choose Request access next to each repository you want AWS CodeBuild to be able to access. After you choose Authorize application, back in the AWS CodeBuild console, for Repository, choose the name of the repository that contains the source code. Select the Build Badge check box to make your project's build status visible and embeddable.
   - If you chose Bitbucket, follow the instructions to connect (or reconnect) with Bitbucket. On the Bitbucket Confirm access to your account page, for Organization access, choose Grant access. After you choose Grant access, back in the AWS CodeBuild console, for Repository, choose the name of the repository that contains the source code. Select the Build Badge check box to make your project's build status visible and embeddable.
### Important
If you update your project source, then this could affect the accuracy of the project's build badges.

5. **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.

   For **Build specification**, do one of the following:

   - Use the `buildspec.yml` file in the source code root directory.
   - Override the build specification by inserting the build commands.

   For more information, see the [Build Spec Reference](#) (p. 93).

6. **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**.
   - To store the build output in an Amazon S3 bucket, choose **Amazon S3**, and then do the following:
     - If you want to use your project name for the build output ZIP file or folder, leave **Artifacts name** blank. Otherwise, type the name in the **Artifacts name** box. (By default, the artifact name is the project name. If you want to specify a different name, type it in the artifacts name box. If you want to output a ZIP file, then include the zip extension.
     - For **Bucket name**, choose the name of the output bucket.
   - If you chose **Insert build commands** earlier 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. 94).

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

   - If you do not have an AWS CodeBuild service role, choose **Create a service role in your account**. In **Role name**, accept the default name or type your own.
   - If you have an AWS CodeBuild service role, choose **Choose an service 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 is updated to work with the other build project. A service role can work with up to 10 build projects.

8. Expand **Show advanced settings** and set the other advanced settings as appropriate.

9. Choose **Continue**. On the **Review** page, choose **Save and build** or, to run the build later, choose **Save**.
Create a Build Project with Build Badges Enabled (CLI)

For information on creating a build project, see Create a Build Project (AWS CLI) (p. 146). To include build badges with your AWS CodeBuild project, you must specify `badgeEnabled` with a value of `true`.

Access Your AWS CodeBuild Build Badges

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

- In the AWS CodeBuild console, in the list of build projects, in the `Project` column, choose the link that corresponds to the build project. On the `Build project: project-name` page, expand `Project details`. The build badge URL appears under `Advanced`. For more information, see View a Build Project's Details (Console) (p. 155).
- In the AWS CLI, run the `batch-get-projects` command. The build badge URL is included in the project environment details section of the output. For more information, see View a Build Project's Details (AWS CLI) (p. 155).

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

Publish Your AWS CodeBuild Build Badges

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

Sample markdown code:

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

AWS CodeBuild Badge Statuses

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

Build Notifications Sample for AWS CodeBuild

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

**Important**
Running this sample may result in charges to your AWS account. These include possible charges for AWS CodeBuild and for AWS resources and actions related to Amazon CloudWatch and Amazon SNS. For more information, see AWS CodeBuild Pricing, Amazon CloudWatch Pricing, and Amazon SNS Pricing.
Running the Sample

To run this sample:

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

   ```json
   }
   ```

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

2. Create or identify a topic in Amazon SNS. AWS CodeBuild will use CloudWatch Events to send build notifications to this topic through Amazon SNS. To create a topic:

   2. Choose Create topic.
   3. In the Create new topic dialog box, for Topic name, type a name for the topic, for example CodeBuildDemoTopic. (If you choose a different name, substitute it throughout this sample.)
   4. Choose Create topic.
   5. On the Topic details: CodeBuildDemoTopic page, copy the Topic ARN value, as shown in the following screen shot. You will need this value for the next step.
Topic details

Publish to topic

Topic ARN:

Topic owner:

Region:

Display name:
For more information, see Create a Topic in the Amazon SNS Developer Guide.

3. Subscribe one or more recipients to the topic to receive email notifications. To subscribe a recipient to a topic:

1. With the Amazon SNS console open from the previous step, in the navigation pane, choose Subscriptions, and then choose Create subscription.
2. In the Create subscription dialog box, for Topic ARN, paste the topic ARN you copied from the previous step.
3. For Protocol, choose Email.
4. For Endpoint, type the recipient's full email address. Compare your results to the following screen shot.
5. Choose **Create Subscription**.

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

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

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

```json
{
    "Statement": [
        
        "Action": [  
            "events:*",  
            "iam:PassRole"
        ],  
        "Resource": "*",  
        "Effect": "Allow"
    ],  
    "Version": "2012-10-17"
}
```

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

5. Create a rule in CloudWatch Events. To do this, open the CloudWatch console, at https://console.aws.amazon.com/cloudwatch.

6. In the navigation pane, under **Events**, choose **Rules**, and then choose **Create rule**.

7. On the **Step 1: Create rule page**, the following should already be chosen: **Event Pattern** and **Build event pattern to match events by service**.

8. For **Service Name**, choose **CodeBuild**. For **Event Type**, **All Events** should already be chosen.

9. **Event Pattern Preview** should show the following code.

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

Compare your results so far to the following screen shot:
Step 1: Create Event Source

Create rules to invoke Target

Build or customize an Event Pattern

Schedule to invoke Target
10. Replace the code in Event Pattern Preview with one of the following two rule patterns by choosing Edit.

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

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

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

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

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

```json
{
    "source": ["aws.codebuild"],
    "detail-type": ["CodeBuild Build Phase Change"],
    "detail": {
        "completed-phase": ["SUBMITTED", "PROVISIONING", "DOWNLOAD_SOURCE", "INSTALL", "PRE_BUILD", "BUILD", "POST_BUILD", "UPLOAD_ARTIFACTS",
    }
}
```
In the preceding rule, make the following code changes as needed.

- To trigger an event for every build phase change (which may send up to 9 notifications for each build), either leave all of the values as shown in the `completed-phase` array, or remove the `completed-phase` array altogether.
- To trigger events only for individual build phase changes, remove the name of each build phase in the `completed-phase` array that you do not want to trigger an event for.
- To trigger an event for every build phase status change, either leave all of the values as shown in the `completed-phase-status` array, or remove the `completed-phase-status` array altogether.
- To trigger events only for individual build phase status changes, remove the name of each build phase status in the `completed-phase-status` array that you do not want to trigger an event for.
- To trigger events for all build projects, remove the `project-name` array.
- To trigger events for individual build projects, specify the name of each build project in the `project-name` array.

**Note**
If you want to trigger events for both build state changes and build phase changes, you must create two separate rules, one for build state changes and another for build phase changes. If you try to combine both rules into a single rule, the combined rule may produce unexpected results or stop working altogether.

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

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

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

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

For a rule with a detail-type value of CodeBuild Build Phase Change, type the following.
Note
To get other types of information, see the Build Notifications Input Format Reference (p. 54).

16. In the Input Template box, type one of the following input templates.

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

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

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

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

Compare your results so far to the following screen shot, which shows a rule with a detail-type value of CodeBuild Build State Change:
Step 1: Create rules to invoke Target

Event Source

Build or customize an Event Source

Schedule to invoke Target
17. Choose **Configure details**.

18. On the **Step 2: Configure rule details** page, type a **Name** and an optional **Description**. Leave the **Enabled** box checked for **State**.

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

Rule definition

Name* CodeBuild

Description

State Yes Enable
19. Choose **Create rule**.

20. Create build projects, run the builds, and view build information, for example by following the steps in *Run AWS CodeBuild Directly (p. 111)*.

21. Confirm that AWS CodeBuild is now successfully sending build notifications. For example, check to see if the build notification emails are now in your inbox.

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

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

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

**Related Resources**

- For more information about getting started with AWS CodeBuild, see *Getting Started with AWS CodeBuild (p. 4)*.
- For more information about troubleshooting problems with AWS CodeBuild, see *Troubleshooting Your VPC Setup (p. 114)*.
- For more information about limits in AWS CodeBuild, see *Limits for AWS CodeBuild (p. 211)*.

**Build Notifications Input Format Reference**

CloudWatch delivers notifications in JSON format.

Build state change notifications use the following format:

```json
{
    "version": "0",
    "id": "c030038d-8c4d-6141-9545-0ff7b7153EX",
    "detail-type": "CodeBuild Build State Change",
    "source": "aws.codebuild",
    "account": "123456789012",
    "time": "2017-09-01T16:14:28Z",
    "region": "us-west-2",
    "resources": [
    ],
    "detail": {
        "build-status": "SUCCEEDED",
        "project-name": "my-sample-project",
        "additional-information": {
            "artifact": {
                "md5sum": "da9c44c8a9a3cd4b443126e833168fEX",
                "sha256sum": "6ccc2ae1df9d155ba83c597051611c42d6e09c6329dcb14a312ceccc0a8e39EX",
                "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-artifact.zip"
            },
            "environment": {
                "image": "aws/codebuild/dot-net:1.1",
                "privileged-mode": false,
                "compute-type": "BUILD_GENERAL1_SMALL",
            }
        }
    }
}
```

API Version 2016-10-06

54
Android

API Version 2016-10-06

55
Build phase change notifications use the following format:

```json
{
  "version": "0",
  "id": "43ddc2bd-af76-9ca5-2dc7-b695e15adeEX",
  "detail-type": "CodeBuild Build Phase Change",
  "source": "aws.codebuild",
  "account": "123456789012",
  "time": "2017-09-01T16:14:21Z",
  "region": "us-west-2",
  "resources": [
  ],
  "detail": {
    "completed-phase": "COMPLETED",
    "project-name": "my-sample-project",
    "completed-phase-context": "[]",
    "additional-information": {
      "artifact": {
        "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
        "sha256sum": "6ccc2a1df9d155ba83c597051611432d60e09c6329dcb14a312cecc0a8e39EX",
        "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-artifact.zip"
      },
      "environment": {
```
"image": "aws/codebuild/dot-net:1.1",
"privileged-mode": false,
"compute-type": "BUILD_GENERAL1_SMALL",
"type": "LINUX_CONTAINER",
"environment-variables": []
},
"timeout-in-minutes": 60,
"build-complete": true,
"initiator": "MyCodeBuildDemoUser",
"build-start-time": "Sep 1, 2017 4:12:29 PM",
"source": {
"location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
"type": "S3"
},
"logs": {
"group-name": "/aws/codebuild/my-sample-project",
"stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
},
"phases": [
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:12:29 PM",
"end-time": "Sep 1, 2017 4:12:29 PM",
"duration-in-seconds": 0,
"phase-type": "SUBMITTED",
"phase-status": "SUCCEEDED"
},
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:12:29 PM",
"end-time": "Sep 1, 2017 4:13:05 PM",
"duration-in-seconds": 36,
"phase-type": "PROVISIONING",
"phase-status": "SUCCEEDED"
},
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:13:05 PM",
"end-time": "Sep 1, 2017 4:13:10 PM",
"duration-in-seconds": 4,
"phase-type": "DOWNLOAD_SOURCE",
"phase-status": "SUCCEEDED"
},
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:13:10 PM",
"end-time": "Sep 1, 2017 4:13:10 PM",
"duration-in-seconds": 0,
"phase-type": "INSTALL",
"phase-status": "SUCCEEDED"
},
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:13:10 PM",
"end-time": "Sep 1, 2017 4:13:10 PM",
"duration-in-seconds": 0,
"phase-type": "PRE_BUILD",
"phase-status": "SUCCEEDED"
},
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:13:10 PM",
"end-time": "Sep 1, 2017 4:13:10 PM",
"duration-in-seconds": 0,
"phase-type": "BUILD",
"phase-status": "SUCCEEDED"
},
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:13:10 PM",
"end-time": "Sep 1, 2017 4:13:10 PM",
"duration-in-seconds": 0,
"phase-type": "POST_BUILD",
"phase-status": "SUCCEEDED"
},
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:13:10 PM",
"end-time": "Sep 1, 2017 4:13:10 PM",
"duration-in-seconds": 0,
"phase-type": "TEST",
"phase-status": "SUCCEEDED"
},
{
"phase-context": [],
"start-time": "Sep 1, 2017 4:13:10 PM",
"end-time": "Sep 1, 2017 4:13:10 PM",
"duration-in-seconds": 0,
"phase-type": "PRODUCTION",
"phase-status": "SUCCEEDED"
}]}
Docker in Custom Image Sample for AWS CodeBuild

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

To learn how to build a Docker image by using a build image provided by AWS CodeBuild with Docker support instead, see our Docker Sample (p. 24).

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 (p. 59)
- Directory Structure (p. 59)
- Files (p. 59)
- Related Resources (p. 60)
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, GitHub, or Bitbucket 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. 111).

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

   ```json
   {
     "name": "sample-docker-custom-image-project",
     "source": {
       "type": "S3",
       "location": "codebuild-region-ID-account-ID-input-bucket/DockerCustomImageSample.zip"
     },
     "artifacts": {
       "type": "NO_ARTIFACTS"
     },
     "environment": {
       "type": "LINUX_CONTAINER",
       "image": "docker:dind",
       "computeType": "BUILD_GENERAL1_SMALL",
       "privilegedMode": true
     },
     "serviceRole": "arn:aws:iam::account-ID:role/role-name",
     "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
   }
   ```

3. To see the build results, look in the build's log for the string *Hello, World!* For more information, see View Build Details (p. 167).

Directory Structure

This sample assumes this directory structure.

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

Files

This sample uses these files.

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

version: 0.2
phases:
  install:
    commands:
      - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcp://0.0.0.0:2375 --storage-driver=overlay&
      - timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
  pre_build:
    commands:
      - docker build -t helloworld .
  build:
    commands:
      - docker images
      - docker run helloworld echo "Hello, World!"

Dockerfile (in root directory name)

FROM alpine
RUN ls

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

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.

```shell
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: 0.2
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
```

API Version 2016-10-06
61
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 or Bitbucket repository.

**Important**

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

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

```
|-- 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. 141).

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. 161). 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. 123).

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

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

AWS Lambda Sample for AWS CodeBuild

To define a standard model for serverless applications that use resources such as Lambda, AWS created the AWS Serverless Application Model (AWS SAM). For more information, see the AWS Serverless Application Model repository on GitHub.

You can use AWS CodeBuild to package and deploy serverless applications that follow the AWS SAM standard. For the deployment step, AWS CodeBuild can use AWS CloudFormation. To automate the building and deployment of serverless applications with AWS CodeBuild and AWS CloudFormation, you can use AWS CodePipeline.

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

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

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 deploys the WAR file to the instances in an Elastic Beanstalk environment. This sample is based on the Java Sample (p. 87).

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.

Create the Source Code

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

1. Download and install Maven. For information, see Downloading Apache Maven and Installing Apache Maven on the Apache Maven website.

2. Switch to an empty directory on your local computer or instance, and then run this Maven command.

```java
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
```

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

- Scenario A: Run AWS CodeBuild Manually and Deploy to Elastic Beanstalk Manually (p. 65)
- Scenario B: Use AWS CodePipeline to Run AWS CodeBuild and Deploy to Elastic Beanstalk (p. 67)
- Scenario C: Use the Elastic Beanstalk Command Line Interface (EB CLI) to Run AWS CodeBuild and Deploy to an Elastic Beanstalk Environment (p. 68)
Scenario A: Run AWS CodeBuild Manually and Deploy to Elastic Beanstalk Manually

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

Step A1: Add Files to the Source Code

In this step, you will add an Elastic Beanstalk configuration file and a build spec file to the code in Create the Source Code (p. 64). You will then upload the source code to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

1. Create a subdirectory named `.ebextensions` inside of the (root directory name)/my-web-app directory. In the `.ebextensions` subdirectory, create a file named `fix-path.config` with this content.

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

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

   ```yaml
   version: 0.2
   phases:
     post_build:
       commands:
         - mvn package
         - mv target/my-web-app.war my-web-app.war
   artifacts:
     files:
       - my-web-app.war
       - .ebextensions/**/*
   ```

3. Your file structure should now look like this.

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

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

**Step A2: Create the Build Project and Run the Build**

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

1. Create or identify an Amazon S3 output bucket to store the build output. If you're storing the source code in an Amazon S3 input bucket, the output bucket must be in the same AWS region as the input bucket.


   Use the AWS region selector to choose a region that supports AWS CodeBuild and matches the region where your Amazon S3 output bucket is stored.

3. Create a build project and then run a build. For more information, see Create a Build Project (Console) (p. 141) and Run a Build (Console) (p. 161). Leave all settings at their default values, except for these settings.

   - For **Environment: How to build**:
     - For **Environment image**, choose Use an image managed by AWS CodeBuild.
     - For **Operating system**, choose Ubuntu.
     - For **Runtime**, choose Java.

   - For **Artifacts: Where to put the artifacts from this build project**:
     - For **Artifacts name**, type a build output file name that’s easy for you to remember. Include the .zip extension.

   - For **Show advanced settings**:
     - For **Artifacts packaging**, choose Zip.

**Step A3: Create the Application and Environment and Deploy**

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


   Use the AWS region selector to choose the region that matches the one where your Amazon S3 output bucket is stored.

2. Create an Elastic Beanstalk application. For more information, see Managing and Configuring AWS Elastic Beanstalk Applications.

3. Create an Elastic Beanstalk environment for this application. For more information, see The Create New Environment Wizard. Leave all settings at their default values, except for these settings.

   - For **Platform**, choose Tomcat.

   - For **Application code**, choose Upload your code, and then choose Upload. For **Source code origin**, choose Public S3 URL, and then type the full URL to the build output ZIP file in the output bucket. Then choose Upload.

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

In this scenario, you will finish manually preparing and uploading the source code to be built. You will then use the AWS CodePipeline console to create a pipeline and an Elastic Beanstalk application and environment. After you create the pipeline, AWS CodePipeline automatically builds the source code and deploys the build output to the environment.

Step B1: Add a Build Spec File to the Source Code

In this step, you will create an add a build spec file to the code you created in Create the Source Code (p. 64). You will then upload the source code to an Amazon S3 input bucket or an AWS CodeCommit or GitHub repository.

1. Create a file named `buildspec.yml` with the following contents. Store the file inside of the (root directory name)/my-web-app directory.

```yaml
version: 0.2
phases:
  post_build:
    commands:
    - mvn package
artifacts:
  files:
    - '**/*'
  base-directory: 'target/my-web-app'
```

2. Your file structure should now look like this.

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

3. Upload this contents of the my-web-app directory to an Amazon S3 input bucket or an AWS CodeCommit, GitHub, or Bitbucket 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.

Step B2: Create the Pipeline and Deploy

In this step, you will use the AWS CodePipeline and Elastic Beanstalk consoles to create a pipeline, an application, and an environment. After you create the pipeline and it runs, AWS CodePipeline uses AWS CodeBuild to build the source code, and then it uses Elastic Beanstalk to deploy the build output to the environment.
1. Create or identify a service role that AWS CodePipeline, AWS CodeBuild, and Elastic Beanstalk can use to do their work on your behalf. For more information, see Prerequisites (p. 124).


   Use the AWS region selector to choose a region that supports AWS CodeBuild and, if you're storing the source code in an Amazon S3 input bucket, choose the region that matches the one where your input bucket is stored.

3. Create a pipeline. For information, see Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console) (p. 125). Leave all settings at their default values, except for these settings.

   - For Step 3: Build, for Configure your project, choose Create a new build project. For Environment: How to build:
     - For Environment image, choose Use an image managed by AWS CodeBuild.
     - For Operating system, choose Ubuntu.
     - For Runtime, choose Java.

   - For Step 4: Beta, for Deployment provider, choose AWS Elastic Beanstalk.
     - For the application, choose the create a new one in Elastic Beanstalk link. This opens the Elastic Beanstalk console. For more information, see Managing and Configuring AWS Elastic Beanstalk Applications. After you create the application, return to the AWS CodePipeline console, and then select the application you just created.

   - For the environment, choose the create a new one in Elastic Beanstalk link. This opens the Elastic Beanstalk console. For more information, see The Create New Environment Wizard. Leave all but one setting at their default values: for Platform, choose Tomcat. After you create the environment, return to the AWS CodePipeline console, and then select the environment you just created.

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

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

**Scenario C: Use the Elastic Beanstalk Command Line Interface (EB CLI) to Run AWS CodeBuild and Deploy to an Elastic Beanstalk Environment**

In this scenario, you will finish manually preparing and uploading the source code to be built. You will then run the EB CLI to create an Elastic Beanstalk application and environment, use AWS CodeBuild to build the source code, and then deploy the build output to the environment. For more information, see Using the EB CLI with AWS CodeBuild in the AWS Elastic Beanstalk Developer Guide.

**Step C1: Add Files to the Source Code**

In this step, you will add an Elastic Beanstalk configuration file and a build spec file to the code you created in Create the Source Code (p. 64). You will also create or identify a service role for the build spec file.
1. Create or identify a service role that Elastic Beanstalk and the EB CLI can use on your behalf. For information, see Create an AWS CodeBuild Service Role (p. 179).

2. Create a subdirectory named .ebextensions inside of the (root directory name)/my-web-app directory. In the .ebextensions subdirectory, create a file named fix-path.config with this content.

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

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

   ```
   version: 0.2
   phases:
     post_build:
       commands:
         - mvn package
         - mv target/my-web-app.war my-web-app.war
   artifacts:
     files:
       - my-web-app.war
       - .ebextensions/**/*
   eb_codebuild_settings:
     CodeBuildServiceRole: my-service-role-name
     ComputeType: BUILD_GENERAL1_SMALL
     Image: aws/codebuild/java:openjdk-8
     Timeout: 60
   ```

   In the preceding code, replace my-service-role-name with the name of the service role you created or identified earlier.

4. Your file structure should now look like this.

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

**Step C2: Install and Run the EB CLI**

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

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

When prompted:

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

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

```
 eb create
```

When prompted:

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

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

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

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

If you want, you can make changes to the source code and then run the `eb deploy` command from the same directory. The EB CLI performs the same steps as the `eb create` command, but it deploys the build output to the existing environment instead of creating a new environment.

**Related Resources**

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

**AWS CodeBuild Code-Based Samples**

Refer to these code-based samples to experiment with AWS CodeBuild:
**Name** | **Description**
--- | ---
**C++ Sample (p. 71)** | Uses C++ to output a single .out file.
**Go Sample (p. 73)** | Uses Go to output a single binary file.
**Maven Sample (p. 75)** | Uses Apache Maven to produce a single JAR file.
**Node.js Sample (p. 77)** | Uses Mocha to test whether an internal variable in code contains a specific string value. Produces a single .js file.
**Python Sample (p. 79)** | 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. 81)** | 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. 83)** | Uses sbt to produce a single JAR file.
**Java Sample (p. 87)** | Uses Apache Maven to produce a single WAR file.
**.NET Core in Linux Sample (p. 89)** | Uses .NET Core to build an executable file out of code written in C#.

---

**C++ Hello World Sample for AWS CodeBuild**

This C++ sample produces as build output a single binary file named `hello.out`.

**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 (p. 71)
- Directory Structure (p. 72)
- Files (p. 72)
- Related Resources (p. 73)

**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, GitHub, or Bitbucket 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. 111).

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

```json
{
  "name": "sample-c-plus-plus-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/CPlusPlusSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "CPlusPlusOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/ubuntu-base:14.04",
    "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 `CPlusPlusOutputArtifact.zip` file to your local computer or instance, and then extract the contents of the file. In the extracted contents, get the `hello.out` file.

### Directory Structure

This sample assumes this directory structure.

```
(root directory name)
|-- buildspec.yml
 `-- hello.cpp
```

### Files

This sample uses these files.

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

```yaml
version: 0.2
phases:
  install:
    commands:
      - apt-get update -y
      - apt-get install -y build-essential
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the C++ code...
```

API Version 2016-10-06
72
Go Sample

```bash
- g++ hello.cpp -o hello.out

post_build:
  commands:
    - echo Build completed on `date`

artifacts:
  files:
    - hello.out
```

```c++
#include <iostream>

int main()
{
  std::cout << "Hello, World!\n";
}
```

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

Go Hello World Sample for AWS CodeBuild

This Go sample produces as build output a single binary file named `hello`.

**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 (p. 73)
- Directory Structure (p. 74)
- Files (p. 74)
- Related Resources (p. 75)

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, GitHub, or Bitbucket 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. 111).

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

```json
{
  "name": "sample-go-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "GoOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/golang:1.7.3",
    "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 GoOutputArtifact.zip file to your local computer or instance, and then extract the contents of the file. In the extracted contents, get the `hello` file.

### Directory Structure

This sample assumes this directory structure.

```
(root directory name)
|   |-- buildspec.yml
|   `-- hello.go
```

### Files

This sample uses these files.

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

```yaml
version: 0.2
phases:
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Go code...
      - go build hello.go
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - hello
```
```go
package main
import "fmt"

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

Related Resources

- For more information about getting started with AWS CodeBuild, see [Getting Started with AWS CodeBuild](p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see [Troubleshooting Your VPC Setup](p. 114).
- For more information about limits in AWS CodeBuild, see [Limits for AWS CodeBuild](p. 211).

**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
```
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: 0.2
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, GitHub, or Bitbucket 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. 111).

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.)
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.

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

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 (p. 77)
- Directory Structure (p. 78)
- Files (p. 78)
- Related Resources (p. 79)

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, GitHub, or Bitbucket 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. 111).

   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.)

   ```json
   {
     "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.

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

**Files**

This sample uses these files.

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

```yaml
version: 0.2
```
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))

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

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

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

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 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 (p. 80)
- Directory Structure (p. 80)
- Files (p. 80)
- Related Resources (p. 81)
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, GitHub, or Bitbucket 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. 111).

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

   ```json
   {
     "name": "sample-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

```
version: 0.2
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

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

HelloWorld_tst.py

```
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()
```

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

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 (p. 82)
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, GitHub, or Bitbucket repository.

   **Important**
   
   Do not upload \( \text{(root directory name)} \), just the files inside of \( \text{(root directory name)} \).

   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 \( \text{(root directory name)} \) to the ZIP file, just the files inside of \( \text{(root directory name)} \).

2. Create a build project, run the build, and view related build information by following the steps in Run AWS CodeBuild Directly (p. 111).

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

   ```json
   {
     "name": "sample-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` *(root directory name)*

```
version: 0.2
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` *(root directory name)*

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

`HelloWorld_spec.rb` *(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
```

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

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 (p. 84)
- Directory Structure (p. 85)
- Files (p. 85)
- Related Resources (p. 87)

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, GitHub, or Bitbucket 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. 111).

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

   ```json
   {
       "name": "sample-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
| `-- macros
| `-- `-- main
|   `-- scala
|     `-- Macros.scala
|-- project
| `-- Build.scala
| `-- build.properties
```

Files

This sample uses these files.

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

```
version: 0.2
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` *(root directory name)/core/src/main/scala*

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

`Macros.scala` *(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.scalam Macros",
    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.scalam Macros" % "paradise" % "2.1.0-M5" cross CrossVersion.full),
              "org.scalam Macros" % "quasiquotes" % "2.1.0-M5" cross CrossVersion.binary)
        }
      }
    )
  )

  lazy val core: Project = Project("core", file("core")
    settings = buildSettings
  )
)

build.properties (in (root directory name)/project)
sbt.version=0.13.7

**Related Resources**

- For more information about getting started with AWS CodeBuild, see [Getting Started with AWS CodeBuild](#) (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see [Troubleshooting Your VPC Setup](#) (p. 114).
- For more information about limits in AWS CodeBuild, see [Limits for AWS CodeBuild](#) (p. 211).

**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: 0.2
   phases:
     post_build:
       commands:
         - echo Build completed on `date`
   ```

API Version 2016-10-06
87
- mvn package
artifacts:
  files:
    - target/my-web-app.war
discard-paths: yes

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

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

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

```markdown
WebArchiveHelloWorldSample.zip
  |-- buildspec.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. 111).

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

```json
{
  "name": "sample-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.

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).

.NET Core in Linux Sample for AWS CodeBuild

This sample uses a AWS CodeBuild build environment running .NET Core to build an executable file out of code written in C#.

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. 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, GitHub, or Bitbucket 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. 111).
   If you use the AWS CLI to create the build project, the JSON-formatted input to the `create-project` command might look similar to this. (Replace the placeholders with your own values.)

```json
{
    "name": "sample-dot-net-core-project",
    "source": {
```
3. To get the build output artifact, in your Amazon S3 output bucket, download the `dot-net-core-output-artifact.zip` file to your local computer or instance. Extract the contents to get to the executable file `HelloWorldSample.dll`, which be found in the `bin\Debug\netcoreapp2.0` directory.

## Directory Structure

This sample assumes this directory structure.

```
(root directory name)
|-- buildspec.yml
  |-- HelloWorldSample.csproj
   `-- Program.cs
```

## Files

This sample uses these files.

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

```yaml
version: 0.2
phases:  
  build:  
    commands:  
      - dotnet restore  
      - dotnet build  
  artifacts:  
    files:  
      - bin/Debug/netcoreapp2.0/*
```

**HelloWorldSample.csproj (in (root directory name))**

```xml
<Project Sdk="Microsoft.NET.Sdk">  
  <PropertyGroup>  
    <OutputType>Exe</OutputType>  
    <TargetFramework>netcoreapp2.0</TargetFramework>  
  </PropertyGroup>  
  <Project>
```

**Program.cs (in (root directory name))**
using System;

namespace HelloWorldSample {
    public static class Program {
        public static void Main() {
            Console.WriteLine("Hello, World!");
        }
    }
}

Related Resources

- For more information about getting started with AWS CodeBuild, see Getting Started with AWS CodeBuild (p. 4).
- For more information about troubleshooting problems with AWS CodeBuild, see Troubleshooting Your VPC Setup (p. 114).
- For more information about limits in AWS CodeBuild, see Limits for AWS CodeBuild (p. 211).
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.

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

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. 93).

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 Elastic Container Registry (Amazon ECR). For more information about the AWS CodeBuild Docker image repository, see Docker Images Provided by AWS CodeBuild (p. 101).

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.

5. Do you want AWS CodeBuild to work with your VPC? If so, you need the VPC ID, the subnet IDs, and security group IDs for your VPC configuration. For more information, see Use AWS CodeBuild with Amazon Virtual Private Cloud (p. 112).

After you have answered these questions, you should have the settings and resources you need to run a build successfully. To run your build, you can:

- Use the AWS CodeBuild console, AWS CLI, or AWS SDKs. For more information, see Run AWS CodeBuild Directly (p. 111).
- Create or identify a pipeline in AWS CodePipeline, and then add a build or test action that instructs AWS CodeBuild to automatically test your code, run your build, or both. For more information, see Use AWS CodePipeline with AWS CodeBuild (p. 123).

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. 3).

Topics
- Build Spec File Name and Storage Location (p. 93)
- Build Spec Syntax (p. 94)
- Build Spec Example (p. 98)
- Build Spec Versions (p. 100)

Build Spec File Name and Storage Location

If you include a build spec as part of the source code, by default, the build spec file must be named buildspec.yml and be placed in the root of your source directory.

You can also override the default build spec file name and location. For example, you can:

- Use a different build spec file for different builds in the same repository, such as buildspec_debug.yml and buildspec_release.yml.
- Store a build spec file somewhere other than the root of your source directory, such as config/buildspec.yml.

You can specify only one build spec for a build project, regardless of the build spec file's name.

To override the default build spec file name, location, or both, do one of the following:
• Run the AWS CLI `create-project` or `update-project` command, setting the `buildspec` value to the path to the alternate build spec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. You can also do the equivalent with the create project operation in the AWS SDKs. For more information, see Create a Build Project (p. 141) or Change a Build Project's Settings (p. 156).

• Run the AWS CLI `start-build` command, setting the `buildspecOverride` value to the path to the alternate build spec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. You can also do the equivalent with the start build operation in the AWS SDKs. For more information, see Run a Build (p. 161).

• In an AWS CloudFormation template, set the `BuildSpec` property of `Source` in a resource of type `AWS::CodeBuild::Project` to the path to the alternate build spec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. For more information, see the "BuildSpec" property in AWS CodeBuild Project Source in the AWS CloudFormation User Guide.

# Build Spec Syntax

Build specs must be expressed in YAML format.

The build spec has the following syntax:

```yaml
version: 0.2
env:
  variables:
    key: "value"
  parameter-store:
    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
base-directory: location
```

The build spec contains the following:

- **version**: Required mapping. Represents the build spec version. We recommend you use **0.2**.
**Note**
Version 0.1 is still supported. However, we recommend that you use version 0.2 whenever possible. For more information, see [Build Spec Versions](#).

- **env**: Optional sequence. Represents information for one or more custom environment variables.

- **variables**: Required if **env** is specified, and you want to define custom environment variables in plain text. Contains a mapping of *key/value* scalars, where each mapping represents a single custom environment variable in plain text. *key* is the name of the custom environment variable, and *value* is that variable's value.

**Important**
We strongly discourage using environment variables to store sensitive values, especially AWS access key IDs and secret access keys. Environment variables can be displayed in plain text using tools such as the AWS CodeBuild console and the AWS CLI. For sensitive values, we recommend you use the **parameter-store** mapping instead, as described later in this section.

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.

- **parameter-store**: Required if **env** is specified, and you want to retrieve custom environment variables stored in Amazon EC2 Systems Manager Parameter Store. Contains a mapping of *key/value* scalars, where each mapping represents a single custom environment variable stored in Amazon EC2 Systems Manager Parameter Store. *key* is the name you will use later in your build commands to refer to this custom environment variable, and *value* is the name of the custom environment variable stored in Amazon EC2 Systems Manager Parameter Store. To store sensitive values, see [Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough](#) in the [Amazon EC2 Systems Manager User Guide](#).

**Important**
To allow AWS CodeBuild to retrieve custom environment variables stored in Amazon EC2 Systems Manager Parameter Store, you must add the `ssm:GetParameters` action to your AWS CodeBuild service role. For more information, see [Create an AWS CodeBuild Service Role](#).

Any environment variables you retrieve from Amazon EC2 Systems Manager Parameter Store 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 retrieve 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 retrieve 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 store 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 AWS CodeBuild will run during each phase of the build.

  **Note**
  In build spec version 0.1, AWS CodeBuild runs each command in a separate instance of the default shell in the build environment. This means that each command runs in isolation from all other commands. Therefore, by default, you cannot run a single command that relies on the state of any previous commands (for example, changing directories or setting environment variables). To get around this limitation, we recommend you use version 0.2, which solves this issue. If you must use build spec version 0.1 for some reason, we recommend the approaches in Shells and Commands in Build Environments (p. 108).

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.

  **Important**
  Commands in some build phases might not be run if commands in earlier build phases fail. For example, if a command fails during the `install` phase, none of the commands in the `pre_build`, `build`, and `post_build` phases will be run for that build's lifecycle. For more information, see Build Phase Transitions (p. 168).

- **artifacts**: Optional 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. This sequence is not required if, for example, you are building and pushing a Docker image to Amazon ECR, or you are running unit tests on your source code but not building it.

- **files**: Required sequence. Represents the locations 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 or, if set, the base directory. Locations can include the following:

- A single file (for example, `my-file.jar`).
- A single file in a subdirectory (for example, `my-subdirectory/my-file.jar` or `my-parent-subdirectory/my-subdirectory/my-file.jar`).
- `**/*` represents all files recursively.
- `my-subdirectory/*` represents all files in a subdirectory named `my-subdirectory`.
- `my-subdirectory/**/*` represents all files recursively starting from a subdirectory named `my-subdirectory`.

When you specify build output artifact locations, AWS CodeBuild can locate the original build location in the build environment. You do not have to prepend your build artifact output locations with the path to the original build location or specify `./*` or similar. If you want to know the path to this location, you can run a command such as `echo $CODEBUILD_SRC_DIR` during a build. The location for each build environment might be slightly different.

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

- `base-directory`: Optional mapping. Represents one or more top-level directories, relative to the original build location, that AWS CodeBuild uses to determine which files and subdirectories to include in the build output artifact. Valid values include:
  - A single top-level directory (for example, `my-directory`).
  - `'my-directory*'` represents all top-level directories with names starting with `my-directory`.

Matching top-level directories are not included in the build output artifact, only their files and subdirectories.

You can use `files` and `discard-paths` to further restrict which files and subdirectories are included. For example, for the following directory structure:

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

And for the following `artifacts` sequence:

```
artifacts:
    files:
        - '/my-file3.txt'
    base-directory: my-build2
```

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

```
my-subdirectory
    |-- my-file3.txt
```

While for the following `artifacts` sequence:

```
artifacts:
    files:
        - '**/*'
```

API Version 2016-10-06
base-directory: 'my-build*'  
discard-paths: yes

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

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

- cache Optional sequence. Represents information about where AWS CodeBuild can prepare the files for uploading cache to an Amazon S3 cache bucket. This sequence is not required if the cache type of the project is No Cache.
- paths: Required sequence. Represents the locations of the cache. Contains a sequence of scalars, with each scalar representing a separate location where AWS CodeBuild can find build output artifacts, relative to the original build location or, if set, the base directory. Locations can include the following:
  - A single file (for example, my-file.jar).
  - A single file in a subdirectory (for example, my-subdirectory/my-file.jar or my-parent-subdirectory/my-subdirectory/my-file.jar).
  - '**/*' represents all files recursively.
  - my-subdirectory/* represents all files in a subdirectory named my-subdirectory.
  - my-subdirectory/**/* represents all files recursively starting from a subdirectory named my-subdirectory.

Important
Because a 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, or the AWS SDKs 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. There is an example in the next section.

If you use the AWS CodeBuild or AWS CodePipeline consoles instead of a buildspec.yml file, you can insert commands for the build phase only. Instead of using the preceding syntax, you list, in a single line, all of the commands 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 instead of a buildspec.yml file to specify the locations of the build output artifacts in the build environment. Instead of using the preceding syntax, you list, in a single line, all of the locations. For multiple locations, separate each location with a comma (for example, appspec.yml, target/my-app.jar).

Build Spec Example

Here is an example of a buildspec.yml file.

```
version: 0.2

env:
  variables:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
    LOGIN_PASSWORD: "dockerLoginPassword"

phases:
  install:
    commands:
```
- echo Entered the install phase...
- apt-get update -y
- apt-get install -y maven

pre_build:
  commands:
    - echo Entered the pre_build phase...
    - docker login -u User -p $LOGIN_PASSWORD

build:
  commands:
    - echo Entered the build phase...
    - echo Build started on `date`
    - mvn install

post_build:
  commands:
    - echo Entered the post_build phase...
    - 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, or the AWS SDKs.

"version: 0.2

variables:
  JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  LOGIN_PASSWORD: "dockerLoginPassword"

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 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 in plain text with the key of JAVA_HOME and the value of /usr/lib/jvm/java-8-openjdk-amd64 will be set.
- A custom environment variable named dockerLoginPassword you stored in Amazon EC2 Systems Manager Parameter Store will be referenced later in build commands by using the key LOGIN_PASSWORD.
- You cannot change these build phase names. The commands that will be run in this example are apt-get update -y and apt-get install -y maven (to install Apache Maven), mvn install (to compile, test, and package the source code into a build output artifact and to perform other actions, such as install the build output artifact in its internal repository), docker login (to log in to Docker with the password that corresponds to the value of the custom environment variable dockerLoginPassword you set in Amazon EC2 Systems Manager Parameter Store), and several echo commands. The echo commands are included here to show how AWS CodeBuild runs commands and the order in which it runs them.
- files represents the files to upload to the build output location. In this example, 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.

<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td><code>environment_variables</code> has been renamed to <code>env.plaintext</code> has been renamed to <code>variables</code>. In version 0.1, AWS CodeBuild runs each build command in a separate instance of the default shell in the build environment. In version 0.2, this issue is addressed; AWS CodeBuild runs all build commands in the same instance of the default shell in the build environment.</td>
</tr>
<tr>
<td>0.1</td>
<td>This is the initial definition of the build specification format.</td>
</tr>
</tbody>
</table>

## 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, see How AWS CodeBuild Works (p. 3).

A build environment contains a Docker image. For information, see Docker Glossary: Image 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 supported repository type. These include the AWS CodeBuild Docker image repository, publicly available images in Docker Hub, and Amazon Elastic 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. 101).
- To get the identifier of a publicly available Docker image stored in Docker Hub, see Searching for images on the Docker Docs website.
- To learn how to work with Docker images stored in Amazon ECR repositories in your AWS account, see our Amazon ECR Sample (p. 22).

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. 107).

**Topics**

- Docker Images Provided by AWS CodeBuild (p. 101)
- Build Environment Compute Types (p. 107)
Docker Images Provided by AWS CodeBuild

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

<table>
<thead>
<tr>
<th>Platform</th>
<th>Programming language or framework</th>
<th>Runtime version</th>
<th>Additional components</th>
<th>Image identifier</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu 14.04</td>
<td>(Base image)</td>
<td></td>
<td>AWS CLI, Git 1.9.1</td>
<td>aws/codebuild/ubuntu-base:14.04</td>
<td>ubuntu/ubuntu-base/14.04</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Android</td>
<td>24.4.1</td>
<td>AWS CLI, Git 1.9.1, Java 8, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/android-java-8:24.4.1</td>
<td>ubuntu/android-java-8/24.4.1</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Docker</td>
<td>1.12.1</td>
<td>AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7, Docker-compose 1.16.1</td>
<td>aws/codebuild/docker:1.12.1</td>
<td>ubuntu/docker/1.12.1</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Docker</td>
<td>17.09.0</td>
<td>AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7, Docker-compose 1.16.1</td>
<td>aws/codebuild/docker:17.09.0</td>
<td>ubuntu/docker/17.09.0</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Golang</td>
<td>1.7.3</td>
<td>AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/golang:1.7.3</td>
<td>ubuntu/golang/1.7.3</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Golang</td>
<td>1.6.3</td>
<td>AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/golang:1.6.3</td>
<td>ubuntu/golang/1.6.3</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Golang</td>
<td>1.5.4</td>
<td>AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/golang:1.5.4</td>
<td>ubuntu/golang/1.5.4</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Java</td>
<td>8</td>
<td>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</td>
<td>aws/codebuild/java:openjdk-8</td>
<td>ubuntu/java/openjdk-8</td>
</tr>
<tr>
<td>Platform</td>
<td>Programming language or framework</td>
<td>Runtime version</td>
<td>Additional components</td>
<td>Image identifier</td>
<td>Definition</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-----------------</td>
<td>------------</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Java</td>
<td>7</td>
<td>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</td>
<td>aws/codebuild/java:openjdk-7</td>
<td>ubuntu/java/openjdk-7</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Java</td>
<td>6</td>
<td>Apache Ant 1.9.6, Apache Maven 3.2.5, AWS CLI, Git 1.9.1, Gradle 2.7, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/java:openjdk-6</td>
<td>ubuntu/java/openjdk-6</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Node.js</td>
<td>7.0.0</td>
<td>AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/nodejs:7.0.0</td>
<td>ubuntu/nodejs/7.0.0</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Node.js</td>
<td>6.3.1</td>
<td>AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/nodejs:6.3.1</td>
<td>ubuntu/nodejs/6.3.1</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Node.js</td>
<td>5.12.0</td>
<td>AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/nodejs:5.12.0</td>
<td>ubuntu/nodejs/5.12.0</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Node.js</td>
<td>4.4.7</td>
<td>AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/nodejs:4.4.7</td>
<td>ubuntu/nodejs/4.4.7</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Node.js</td>
<td>4.3.2</td>
<td>AWS CLI, Git 1.9.1, NPM, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/nodejs:4.3.2</td>
<td>ubuntu/nodejs/4.3.2</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Python</td>
<td>3.5.2</td>
<td>AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/python:3.5.2</td>
<td>ubuntu/python/3.5.2</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Python</td>
<td>3.4.5</td>
<td>AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/python:3.4.5</td>
<td>ubuntu/python/3.4.5</td>
</tr>
<tr>
<td>Ubuntu 14.04</td>
<td>Python</td>
<td>3.3.6</td>
<td>AWS CLI, Git 1.9.1, pip 8.1.2, Python 2.7</td>
<td>aws/codebuild/python:3.3.6</td>
<td>ubuntu/python/3.3.6</td>
</tr>
</tbody>
</table>
AWS CodeBuild also manages the following Docker images that are not in the AWS CodeBuild and AWS CodePipeline consoles.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Programming language or framework</th>
<th>Runtime version</th>
<th>Additional components</th>
<th>Image identifier</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Golang</td>
<td>1.6</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-go-1.6-amazonlinux-64:2.3.2</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Golang</td>
<td>1.5.3</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-go-1.5-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Platform</td>
<td>Programming language or framework</td>
<td>Runtime version</td>
<td>Additional components</td>
<td>Image identifier</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------</td>
<td>----------------</td>
<td>---------------------------------------</td>
<td>-------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Golang</td>
<td>1.5.3</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-go-1.5-amazonlinux-64:2.1.3</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.4.3</td>
<td>Java</td>
<td>1.8.0</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-java-8-amazonlinux-64:2.4.3</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Java</td>
<td>1.8.0</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-java-8-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Java</td>
<td>1.8.0</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-java-8-amazonlinux-64:2.1.3</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.4.3</td>
<td>Java</td>
<td>1.7.0</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-java-7-amazonlinux-64:2.4.3</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Java</td>
<td>1.7.0</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-java-7-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Java</td>
<td>1.7.0</td>
<td>Apache Maven 3.3.3, Apache Ant 1.9.6, Gradle 2.7</td>
<td>aws/codebuild/eb-java-7-amazonlinux-64:2.1.3</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v4.0.0</td>
<td>Node.js</td>
<td>6.10.0</td>
<td>Git 2.7.4, npm 2.15.5</td>
<td>aws/codebuild/eb-nodejs-6.10.0-amazonlinux-64:4.0.0</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Node.js</td>
<td>4.4.6</td>
<td>Git 2.7.4, npm 2.15.5</td>
<td>aws/codebuild/eb-nodejs-4.4.6-amazonlinux-64:2.1.3</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Python</td>
<td>3.4.3</td>
<td>meld3 1.0.2, pip 7.1.2, setuptools 18.4</td>
<td>aws/codebuild/eb-python-3.4-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Python</td>
<td>3.4.3</td>
<td>meld3 1.0.2, pip 7.1.2, setuptools 18.4</td>
<td>aws/codebuild/eb-python-3.4-amazonlinux-64:2.1.3</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Python</td>
<td>3.4</td>
<td>meld3 1.0.2, pip 7.1.2, setuptools 18.4</td>
<td>aws/codebuild/eb-python-3.4-amazonlinux-64:2.3.2</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Python</td>
<td>2.7.10</td>
<td>meld3 1.0.2, pip 7.1.2, setuptools 18.4</td>
<td>aws/codebuild/eb-python-2.7-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Platform</td>
<td>Programming language or framework</td>
<td>Runtime version</td>
<td>Additional components</td>
<td>Image identifier</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Python</td>
<td>2.7.10</td>
<td>meld3 1.0.2, pip 7.1.2, setuptools 18.4</td>
<td>aws/codebuild/eb-python-2.7-amazonlinux-64:2.1.3</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Python</td>
<td>2.7</td>
<td>meld3 1.0.2, pip 7.1.2, setuptools 18.4</td>
<td>aws/codebuild/eb-python-2.7-amazonlinux-64:2.3.2</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Python</td>
<td>2.6.9</td>
<td>meld3 1.0.2, pip 7.1.2, setuptools 18.4</td>
<td>aws/codebuild/eb-python-2.6-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Python</td>
<td>2.6</td>
<td>meld3 1.0.2, pip 7.1.2, setuptools 18.4</td>
<td>aws/codebuild/eb-python-2.6-amazonlinux-64:2.3.2</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Ruby</td>
<td>2.3.1</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.3-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Ruby</td>
<td>2.3</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.3-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Ruby</td>
<td>2.2.5</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.2-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Ruby</td>
<td>2.2</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.2-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Ruby</td>
<td>2.1.9</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.1-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Ruby</td>
<td>2.1</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.1-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Ruby</td>
<td>2.1</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.1-amazonlinux-64:2.1.6</td>
<td></td>
</tr>
</tbody>
</table>
### Docker Images Provided by AWS CodeBuild

<table>
<thead>
<tr>
<th>Platform</th>
<th>Programming language or framework</th>
<th>Runtime version</th>
<th>Additional components</th>
<th>Image identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Ruby</td>
<td>2.0</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.0-amazonlinux-64:2.3.2</td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Ruby</td>
<td>2.0.0</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.0-amazonlinux-64:2.1.6</td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Ruby</td>
<td>2.0.0</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-2.0-amazonlinux-64:2.1.3</td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.6</td>
<td>Ruby</td>
<td>1.9.3</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-1.9-amazonlinux-64:2.1.6</td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.1.3</td>
<td>Ruby</td>
<td>1.9.3</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-1.9-amazonlinux-64:2.1.3</td>
</tr>
<tr>
<td>Amazon Linux 2016.03, 64-bit v2.3.2</td>
<td>Ruby</td>
<td>1.9</td>
<td>Bundler, RubyGems</td>
<td>aws/codebuild/eb-ruby-1.9-amazonlinux-64:2.3.2</td>
</tr>
</tbody>
</table>

For more information about the Docker images that contain `eb-` in their identifier, see Supported Platforms and Platform History in the AWS Elastic Beanstalk Developer Guide. Docker images that contain `eb-` in their identifier are available for use in Elastic Beanstalk; but are not available in the AWS CodeBuild and AWS CodePipeline consoles.

You can use a build specification to install other components (for example, the AWS CLI, Apache Maven, Apache Ant, Mocha, RSpec, or similar) during the install build phase. For more information, see Build Spec Example (p. 98).

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, 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. 141) or Change a Build Project's Settings (Console) (p. 156).
- 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, 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. 125) or Add an AWS CodeBuild Build Action to a Pipeline (AWS CodePipeline Console) (p. 132).
- For the AWS CLI, run the list-curated-environment-images command:

```bash
aws codebuild list-curated-environment-images
```
- For the AWS SDKs, call the ListCuratedEnvironmentImages operation for your target programming language. For more information, see the AWS SDKs and Tools Reference (p. 186).
To confirm the version of a component installed on a Docker image, you can run a command during a build. The version number for the component will appear in the output. 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`.

The following build command (entered through the AWS CodeBuild or AWS CodePipeline console as part of a build project's settings) returns 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:

<table>
<thead>
<tr>
<th>Compute type</th>
<th>computeType value</th>
<th>Memory</th>
<th>vCPUs</th>
<th>Disk space</th>
</tr>
</thead>
<tbody>
<tr>
<td>build.general1.small</td>
<td>BUILD_GENERAL1_SMALL</td>
<td>3 GB</td>
<td>2</td>
<td>64 GB</td>
</tr>
<tr>
<td>build.general1.medium</td>
<td>BUILD_GENERAL1_MEDIUM</td>
<td>7 GB</td>
<td>4</td>
<td>128 GB</td>
</tr>
<tr>
<td>build.general1.large</td>
<td>BUILD_GENERAL1_LARGE</td>
<td>15 GB</td>
<td>8</td>
<td>128 GB</td>
</tr>
</tbody>
</table>

**Note**

For custom build environment images, AWS CodeBuild supports Docker images up to 10 GB uncompressed, regardless of the compute type. To check your build image's size, use Docker to run the `docker images REPOSITORY:TAG` command.

To choose one of these compute types:

- In the AWS CodeBuild console, in the **Create Project** wizard or **Update project** page, expand **Show advanced settings**, and then choose one of the options from **Compute type**. For more information, see Create a Build Project (Console) (p. 141) or Change a Build Project's Settings (Console) (p. 156).

- In the AWS CodePipeline console, in the **Create Pipeline** wizard on the **Step 3: Build** page, or in the **Add action** or **Edit action** pane, choose **Create a new build project**, expand **Advanced**, and then choose one of the options in **Compute type**. For more information, see Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console) (p. 125) or Add an AWS CodeBuild Build Action to a Pipeline (AWS CodePipeline Console) (p. 132).
For the AWS CLI, run the `create-project` or `update-project` command, specifying the `computeType` value of the environment object. For more information, see Create a Build Project (AWS CLI) (p. 146) or Change a Build Project's Settings (AWS CLI) (p. 159).

For the AWS SDKs, call the equivalent of the `CreateProject` or `UpdateProject` operation for your target programming language, specifying the equivalent of `computeType` value of the environment object. For more information, see the AWS SDKs and Tools Reference (p. 186).

### Shells and Commands in Build Environments

You provide a set of commands for AWS CodeBuild to run in a build environment during the lifecycle of a build (for example, installing build dependencies and testing and compiling your source code). There are several ways to specify these commands:

- Create a build spec file and include it with your source code. In this file, specify the commands you want to run in each phase of the build lifecycle. For more information, see the Build Specification Reference for AWS CodeBuild (p. 93).

- Use the AWS CodeBuild or AWS CodePipeline console to create a build project. In *Insert build commands*, for *Build command*, specify the commands you want to run in the build phase. For more information, see the description of *Insert build commands* in Create a Build Project (Console) (p. 141) or Use AWS CodePipeline with AWS CodeBuild (p. 123).

- Use the AWS CodeBuild console to change the settings of a build project. In *Insert build commands*, for *Build command*, specify the commands you want to run in the build phase. For more information, see the description of *Insert build commands* in Change a Build Project's Settings (Console) (p. 156).

- Use the AWS CLI, or AWS SDKs to create a build project or change the settings of a build project. Reference the source code that contains a build spec file with your commands, or specify a single string that includes the contents of an equivalent build spec file. For more information, see the description for the `buildspec` value in Create a Build Project (p. 141) or Change a Build Project's Settings (p. 156).

- Use the AWS CLI, or AWS SDKs to start a build, specifying a build spec file or a single string that includes the contents of an equivalent build spec file. For more information, see the description for the `buildspecOverride` value in Run a Build (p. 161).

You can specify any command that is supported by the build environment’s default shell (- `sh` is the default shell in curated image). In build spec version 0.1, AWS CodeBuild runs each command in a separate instance of the default shell in the build environment. This means that each command runs in isolation from all other commands. Therefore, by default, you cannot run a single command that relies on the state of any previous commands (for example, changing directories or setting environment variables). To get around this limitation, we recommend you use version 0.2, which solves this issue. If you must use version 0.1 for some reason, we recommend the following approaches:

- Include a shell script in your source code that contains the commands you want to run in a single instance of the default shell. For example, you could include a file named `my-script.sh` in your source code that contains commands such as `cd MyDir; mkdir -p mySubDir; cd mySubDir; pwd;`. Then, in your build spec file, specify the command `. /my-script.sh`.

- In your build spec file, or for the console’s *Build command* setting for the build phase only, specify a single command that includes all of the commands you want to run in a single instance of the default shell (for example, `cd MyDir && mkdir -p mySubDir && cd mySubDir && pwd`).

If AWS CodeBuild encounters an error, the error might be more difficult to troubleshoot compared to running a single command in its own instance of the default shell.
Environment Variables in Build Environments

AWS CodeBuild provides several environment variables that you can use in your build commands:

- **AWS_DEFAULT_REGION**: The AWS region where the build is running (for example, `us-east-1`). This environment variable is used primarily by the AWS CLI.
- **AWS_REGION**: The AWS region where the build is running (for example, `us-east-1`). This environment variable is used primarily by the AWS SDKs.
- **CODEBUILD_BUILD_ARN**: The Amazon Resource Name (ARN) of the build (for example, `arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`).
- **CODEBUILD_BUILD_ID**: The AWS CodeBuild ID of the build (for example, `codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`).
- **CODEBUILD_BUILD_IMAGE**: The AWS CodeBuild build image identifier (for example, `aws/codebuild/java:openjdk-8`).
- **CODEBUILD_BUILD_SUCCEEDING**: Whether the current build is succeeding. Set to 0 if the build is failing, or 1 if the build is succeeding.
- **CODEBUILD_INITIATOR**: The entity that started the build. If AWS CodePipeline started the build, this is the pipeline's name, for example `codepipeline/my-demo-pipeline`. If an IAM user started the build, this is the user's name, for example `MyUserName`. If the Jenkins plugin for AWS CodeBuild started the build, this is the string `CodeBuild-Jenkins-Plugin`.
- **CODEBUILD_KMS_KEY_ID**: The identifier of the AWS KMS key that AWS CodeBuild is using to encrypt the build output artifact (for example, `arn:aws:kms:region-ID:account-ID:key/key-ID` or `alias/key-alias`).
- **CODEBUILD_RESOLVED_SOURCE_VERSION**: For builds run by AWS CodePipeline, the commit ID or Amazon S3 version ID of the source code to be built. Note that this value is available only if the pipeline's related Source action is based on an Amazon S3, AWS CodeCommit, or GitHub repository.
- **CODEBUILD_SOURCE_REPO_URL**: The URL to the input artifact or source code repository. For Amazon S3, this is `s3://` followed by the bucket name and path to the input artifact. For AWS CodeCommit and GitHub, this is the repository's clone URL.
- **CODEBUILD_SOURCE_VERSION**: For Amazon S3, the version ID associated with the input artifact. For AWS CodeCommit, the commit ID or branch name associated with the version of the source code to be built. For GitHub, the commit ID, branch name, or tag name associated with the version of the source code to be built.
- **CODEBUILD_SRC_DIR**: The directory path that AWS CodeBuild uses for the build (for example, `/tmp/src123456789/src`).
- **HOME**: This environment variable is always set to `/root`.

You can also provide build environments with your own environment variables. For more information, see the following topics:

- Use AWS CodePipeline with AWS CodeBuild (p. 123)
- Create a Build Project (p. 141)
- Change a Build Project's Settings (p. 156)
- Run a Build (p. 161)
- Build Spec Reference (p. 93)

**Important**
We strongly discourage using environment variables to store sensitive values, especially AWS access key IDs and secret access keys. Environment variables can be displayed in plain text using tools such as the AWS CodeBuild console and the AWS CLI.
For sensitive values, we recommend you store them in the Amazon EC2 Systems Manager Parameter Store and then retrieve them from your build spec. To store sensitive values, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the Amazon EC2 Systems Manager User Guide. To retrieve them, see the parameter-store mapping in Build Spec Syntax (p. 94).

To list all of the available environment variables in a build environment, you can run the `printenv` command during a build. With the exception of those previously listed, environment variables that start with `CODEBUILD_` are for AWS CodeBuild internal use. They should not be used in your build commands.

**Background Tasks in Build Environments**

You can run background tasks in build environments. To do this, in your build spec, use the `nohup` command to run a command as a task in the background, even if the build process exits the shell. Use the `disown` command to forcibly stop a running background task.

**Examples:**

- Start a background process and wait for it to complete later:

  ```bash
  nohup sleep 30 & ; echo $! > pidfile
  wait $(cat pidfile)
  ```

- Start a background process and do not wait for it to ever complete:

  ```bash
  nohup sleep 30 & ; disown $!
  ```

- Start a background process and kill it later:

  ```bash
  nohup sleep 30 & ; echo $! > pidfile
  kill $(cat pidfile)
  ```
Run AWS CodeBuild Directly

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

Not what you're looking for? To use AWS CodePipeline to run AWS CodeBuild, see Use AWS CodePipeline with AWS CodeBuild (p. 123).

Topics
- Prerequisites (p. 111)
- Run AWS CodeBuild Directly (Console) (p. 111)
- Run AWS CodeBuild Directly (AWS CLI) (p. 111)

Prerequisites

Answer the questions in Plan a Build (p. 92).

Run AWS CodeBuild Directly (Console)

1. Create the build project. For information, see Create a Build Project (Console) (p. 141).
2. Run the build. For information, see Run a Build (Console) (p. 161).
3. Get information about the build. For information, see View Build Details (Console) (p. 167).

Run AWS CodeBuild Directly (AWS CLI)

For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 185).

1. Create the build project. For information, see Create a Build Project (AWS CLI) (p. 146).
2. Run the build. For information, see Run a Build (AWS CLI) (p. 163).
3. Get information about the build. For information, see View Build Details (AWS CLI) (p. 168).
Use AWS CodeBuild with Amazon Virtual Private Cloud

Typically, resources in an VPC are not accessible by AWS CodeBuild. To enable access, you must provide additional VPC-specific configuration information as part of your AWS CodeBuild project configuration. This includes the VPC ID, the VPC subnet IDs, and the VPC security group IDs. VPC-enabled builds are then able to access resources inside your VPC. For more information about setting up a VPC in Amazon VPC, see the VPC User Guide.

Topics
- Use Cases (p. 112)
- Enabling Amazon VPC Access in Your AWS CodeBuild Projects (p. 112)
- Best Practices for VPCs (p. 113)
- Troubleshooting Your VPC Setup (p. 114)
- AWS CloudFormation VPC Template (p. 114)

Use Cases

VPC connectivity from AWS CodeBuild builds makes it possible to:

- Run integration tests from your build against data in an Amazon RDS database that's isolated on a private subnet.
- Query data in an Amazon ElastiCache cluster directly from tests.
- Interact with internal web services hosted on Amazon EC2, Amazon ECS, or services that use internal Elastic Load Balancing.
- Retrieve dependencies from self-hosted, internal artifact repositories, such as PyPI for Python, Maven for Java, and npm for Node.js.
- Access objects in an Amazon S3 bucket configured to allow access through an Amazon VPC endpoint only.
- Query external web services that require fixed IP addresses through the Elastic IP address of the NAT gateway or NAT instance associated with your subnet(s).

Your builds can access any resource that's hosted in your VPC.

Enabling Amazon VPC Access in Your AWS CodeBuild Projects

Include these settings in your VPC configuration:

- For **VPC ID**, choose the VPC ID that AWS CodeBuild uses.
- For **Subnets**, choose the subnets that include resources that AWS CodeBuild uses.
- For **Security Groups**, choose the security groups that AWS CodeBuild uses to allow access to resources in the VPCs.
Create a build project (console)

For information about creating a build project, see Create a Build Project (Console) (p. 141). When you create or change your AWS CodeBuild project, in VPC, choose your VPC ID, subnets, and security groups.

Create a build project (AWS CLI)

For information about creating a build project, see Create a Build Project (AWS CLI) (p. 146). If you are using the AWS CLI with AWS CodeBuild, the service role used by AWS CodeBuild to interact with services on behalf of the IAM user must have the following policy attached: Allow a User to Create a VPC Network Interface (p. 196).

The vpcConfig object should include your vpcId, securityGroupIds, and subnets.

- **vpcId**: Required value. The VPC ID that AWS CodeBuild uses. To get a list of all Amazon VPC IDs in your region, run this command:

  ```
  aws ec2 describe-vpcs
  ```

- **subnets**: Required value. The subnet IDs that include resources used by AWS CodeBuild. To obtain these IDs, run this command:

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

  **Note**
  Replace us-east-1 with your region.

- **securityGroupIds**: Required value. The security group IDs used by AWS CodeBuild to allow access to resources in the VPCs. To obtain these IDs, run this command:

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

  **Note**
  Replace us-east-1 with your region.

Best Practices for VPCs

Use this checklist when setting up a VPC to work with AWS CodeBuild.

- Set up your VPC with public and private subnets and a NAT gateway. For more information, see VPC with Public and Private Subnets (NAT).

  **Important**
  You need a NAT gateway or NAT instance in order to use AWS CodeBuild with your Amazon VPC so that AWS CodeBuild can reach public endpoints (for example, to execute CLI commands when running builds). You cannot use the internet gateway instead of a NAT gateway or a NAT instance because AWS CodeBuild does not support assigning elastic IP addresses to the network interfaces that it creates, and auto-assigning a public IP address is not supported by Amazon EC2 for any network interfaces created outside of Amazon EC2 instance launches.

- Include multiple Availability Zones with your VPC.
- Make sure that your security groups have no inbound (ingress) traffic allowed to your builds. For more information, see Security Groups Rules.
- Set up separate subnets for your builds.
When you set up your AWS CodeBuild projects to access your VPC, choose private subnets only.

For more information about setting up a VPC in Amazon VPC, see the Amazon VPC User Guide.

For more information about using AWS CloudFormation to configure an Amazon VPC to use the AWS CodeBuild VPC feature, see the AWS CloudFormation VPC Template (p. 114).

Troubleshooting Your VPC Setup

When troubleshooting VPC issues, use the information that appears in the error message to help you identify, diagnose, and address issues.

The following are some guidelines to assist you when troubleshooting a common AWS CodeBuild VPC error: "Build does not have internet connectivity. Please check subnet network configuration".

1. Make sure that your internet gateway is attached to VPC.
2. Make sure that the route table for your public subnet points to the internet gateway.
3. Make sure that your network ACLs allow traffic to flow.
4. Make sure that your security groups allow traffic to flow.
5. Troubleshoot your NAT gateway.
6. Make sure that the route table for private subnets points to the NAT gateway.
7. Make sure that the service role used by AWS CodeBuild to interact with services on behalf of the IAM user has the following policy attached to it: Allow a User to Create a VPC Network Interface (p. 196).

AWS CloudFormation VPC Template

AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly, by using template files to create and delete a collection of resources together as a single unit (a stack). For more information, see the AWS CloudFormation User Guide.

The following is a AWS CloudFormation YAML template for configuring an Amazon VPC to use AWS CodeBuild VPC feature.

```yaml
Description:
This template deploys a VPC, with a pair of public and private subnets spread across two Availability Zones. It deploys an Internet Gateway, with a default route on the public subnets. It deploys a pair of NAT Gateways (one in each AZ), and default routes for them in the private subnets.

Parameters:

EnvironmentName:
Description: An environment name that will be prefixed to resource names
```
Type: String

VpcCIDR:
  Description: Please enter the IP range (CIDR notation) for this VPC
  Type: String
  Default: 10.192.0.0/16

PublicSubnet1CIDR:
  Description: Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone
  Type: String
  Default: 10.192.10.0/24

PublicSubnet2CIDR:
  Description: Please enter the IP range (CIDR notation) for the public subnet in the second Availability Zone
  Type: String
  Default: 10.192.11.0/24

PrivateSubnet1CIDR:
  Description: Please enter the IP range (CIDR notation) for the private subnet in the first Availability Zone
  Type: String
  Default: 10.192.20.0/24

PrivateSubnet2CIDR:
  Description: Please enter the IP range (CIDR notation) for the private subnet in the second Availability Zone
  Type: String
  Default: 10.192.21.0/24

Resources:

VPC:
  Type: AWS::EC2::VPC
Properties:
  CidrBlock: !Ref VpcCIDR
  Tags:
    - Key: Name
      Value: !Ref EnvironmentName

InternetGateway:
  Type: AWS::EC2::InternetGateway
  Properties:
    Tags:
    - Key: Name
      Value: !Ref EnvironmentName

InternetGatewayAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC

PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet1CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
VpcId: !Ref VPC
AvailabilityZone: !Select [ 1, !GetAZs '' ]
CidrBlock: !Ref PublicSubnet2CIDR
MapPublicIpOnLaunch: true
Tags:
  - Key: Name
    Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet1CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet2CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
Properties:
    Domain: vpc

NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
        Domain: vpc

NatGateway1:
    Type: AWS::EC2::NatGateway
    Properties:
        AllocationId: !GetAtt NatGateway1EIP.AllocationId
        SubnetId: !Ref PublicSubnet1

NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
        AllocationId: !GetAtt NatGateway2EIP.AllocationId
        SubnetId: !Ref PublicSubnet2

PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
        VpcId: !Ref VPC
        Tags:
            - Key: Name
              Value: !Sub ${EnvironmentName} Public Routes

DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
RouteTableId: !Ref PublicRouteTable
DestinationCidrBlock: 0.0.0.0/0
GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2

PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ1)

DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:
Type: AWS::EC2::SubnetRouteTableAssociation

Properties:
  RouteTableId: !Ref PrivateRouteTable1
  SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:

Type: AWS::EC2::RouteTable

Properties:
  VpcId: !Ref VPC
  Tags:
    - Key: Name
      Value: !Sub ${EnvironmentName} Private Routes (AZ2)

DefaultPrivateRoute2:

Type: AWS::EC2::Route

Properties:
  RouteTableId: !Ref PrivateRouteTable2
  DestinationCidrBlock: 0.0.0.0/0
  NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:
  RouteTableId: !Ref PrivateRouteTable2
  SubnetId: !Ref PrivateSubnet2

NoIngressSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:
  GroupName: "no-ingress-sg"
  GroupDescription: "Security group with no ingress rule"
  VpcId: !Ref VPC
Outputs:

VPC:

  Description: A reference to the created VPC
  Value: !Ref VPC

PublicSubnets:

  Description: A list of the public subnets
  Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

PrivateSubnets:

  Description: A list of the private subnets
  Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

PublicSubnet1:

  Description: A reference to the public subnet in the 1st Availability Zone
  Value: !Ref PublicSubnet1

PublicSubnet2:

  Description: A reference to the public subnet in the 2nd Availability Zone
  Value: !Ref PublicSubnet2

PrivateSubnet1:

  Description: A reference to the private subnet in the 1st Availability Zone
  Value: !Ref PrivateSubnet1

PrivateSubnet2:

  Description: A reference to the private subnet in the 2nd Availability Zone
  Value: !Ref PrivateSubnet2

NoIngressSecurityGroup:

  Description: Security group with no ingress rule
  Value: !Ref NoIngressSecurityGroup
Use AWS CodePipeline with AWS CodeBuild to Test Code and Run Builds

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

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Available approaches</th>
<th>Approaches described in this topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a continuous delivery (CD) pipeline with AWS CodePipeline that automates builds with AWS CodeBuild</td>
<td>AWS CodePipeline console, AWS CLI, AWS SDKs</td>
<td>• Using the AWS CodePipeline console (p. 125) • Using the AWS CLI (p. 128) • You can adapt the information in this topic to use the AWS SDKs. 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.</td>
</tr>
<tr>
<td>Add test and build automation with AWS CodeBuild to an existing pipeline in AWS CodePipeline</td>
<td>AWS CodePipeline console, AWS CLI, AWS SDKs</td>
<td>• Add build automation by using the AWS CodePipeline console (p. 132) • Add test automation by using the AWS CodePipeline console (p. 136) • For the AWS CLI, you can adapt the information in this topic to create a pipeline that contains an AWS CodeBuild build action or test action. For more information, see Edit a Pipeline (AWS CLI) and the AWS CodePipeline Pipeline Structure Reference in the AWS CodePipeline User Guide. • You can adapt the information in this topic to use the AWS SDKs pipeline. 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.</td>
</tr>
</tbody>
</table>

Topics

- Prerequisites (p. 124)
- Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console) (p. 125)
- Create a Pipeline that Uses AWS CodeBuild (AWS CLI) (p. 128)
- Add an AWS CodeBuild Build Action to a Pipeline (AWS CodePipeline Console) (p. 132)
- Add an AWS CodeBuild Test Action to a Pipeline (AWS CodePipeline Console) (p. 136)
Prerequisites

1. Answer the questions in Plan a Build (p. 92).

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, macOS, or Unix:
   
   ```bash
   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:
   
   ```bash
   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 a policy statement to it. Add the default AWS CodePipeline service role policy to the service role as described in Review the Default AWS CodePipeline Service Role Policy in the AWS CodePipeline User Guide.

   **Note**
   The IAM entity that adds this AWS CodePipeline service role policy must have permission in IAM to add service role policies to service roles.

5. Create and upload the source code to a repository type supported by AWS CodeBuild and AWS CodePipeline, such as AWS CodeCommit, Amazon S3, or GitHub. (AWS CodePipeline does not currently support Bitbucket.) 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. 93).

   **Important**
   If you plan to use the pipeline to deploy 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. 60) 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. 64) in this guide and see Create an Application Source Bundle in the AWS Elastic Beanstalk Developer Guide.
Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console)

Use the following procedure to create a pipeline that uses AWS CodeBuild to build and deploy your source code.

To create a pipeline that only tests your source code, your options are to:

- Use the following procedure to create the pipeline, and then delete the Build and Beta stages from the pipeline. Then use the Add an AWS CodeBuild Test Action to a Pipeline (AWS CodePipeline Console) (p. 136) procedure in this topic to add to the pipeline a test action that uses AWS CodeBuild.
- Use one of the other procedures in this topic to create the pipeline, and then use the Add an AWS CodeBuild Test Action to a Pipeline (AWS CodePipeline Console) (p. 136) procedure in this topic to add to the pipeline a test action that uses AWS CodeBuild.

To use the create pipeline wizard in AWS CodePipeline to create a pipeline that uses AWS CodeBuild

1. Complete the steps in Prerequisites (p. 124).

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:

```plaintext
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:GetApplications
codedeploy:GetDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

API Version 2016-10-06
125
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.

   Note
   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. 141) or Change a Build Project's Settings (Console) (p. 156).

   Important
   If you enable webhooks for an AWS CodeBuild project, and the project is used as a build step in AWS CodePipeline, then two identical builds will be created for each commit. One build is triggered through webhooks, and one through AWS CodePipeline. Because billing is on a per-build basis, you will be billed for both builds. Therefore, if you are using AWS CodePipeline, we recommend that you disable webhooks in CodeBuild. In the AWS CodeBuild console, clear the Webhook box. For more information, see step 9 in Change a Build Project's Settings (Console) (p. 156)

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. 101).

   • 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).

Select the Privileged check box only if you plan to use this build project to build Docker images, and the build environment image you chose is not one provided by AWS CodeBuild with Docker support. Otherwise, all associated builds that attempt to interact with the Docker daemon will fail. Note that you must also start the Docker daemon so that your builds can interact with it as needed. One way to do this is to initialize the Docker daemon in the install phase of your build spec by running the following build commands. (Do not run the following build commands if you chose a build environment image provided by AWS CodeBuild with Docker support.)

```
-nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcp://0.0.0.0:2375 --storage-driver=overlay
-timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

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.

Important
We strongly discourage using environment variables to store sensitive values, especially AWS access key IDs and secret access keys. Environment variables can be displayed in plain text using tools such as the AWS CodeBuild console and the AWS CLI. To store and retrieve sensitive values, we recommend your build commands use the AWS CLI to interact with the Amazon EC2 Systems Manager Parameter Store. The AWS CLI comes preinstalled and preconfigured on all build environments provided by AWS CodeBuild. For
Create a Pipeline that Uses AWS CodeBuild (AWS CLI)

Use the following procedure to create a pipeline that uses AWS CodeBuild to build your source code.

16. Choose **Save build project**.
17. After the build project is saved, choose **Next step**.
18. On the **Step 4: Deploy** 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 an 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).

   **Note**
   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. 167).
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 codepipeline-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:

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

   In the output, the pipeline object contains an artifactStore object, which contains a location value with the name of the bucket.
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.
To use the AWS CLI to create a pipeline that deploys your built source code or that only tests your source code, you can adapt the instructions in Edit a Pipeline (AWS CLI) and the AWS CodePipeline Pipeline Structure Reference in the AWS CodePipeline User Guide.

1. Complete the steps in Prerequisites (p. 124).

2. Create or identify a build project in AWS CodeBuild. For more information, see Create a Build Project (p. 141).

   **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. 146).

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:

```json
{
    "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": "CodeBuild"
                        },
                        "configuration": {
                            "command": [
                                "build"  
                            ],
                            "artifacts": {
                                "s3": {
                                    "location": "s3://my-input-bucket-name/my-build-artifact.zip"
                                }
                            }  
                        },
                        "runOrder": 2
                    }
                ]
            }
        ]
    }
}
```
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:
In the output, look for information that confirms the build was successful. Ellipses (…) are used to show data that has been omitted for brevity.

```json
{
  ...
  "stageStates": [
    ...
    {
      "actionStates": [
        {
          "actionName": "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/.

   **Note**
   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. 167).

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

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

In the output, the `pipeline` object contains an `artifactStore` object, which contains a `location` value with the name of the bucket.

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)


   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: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):

   ![Source action with Output artifact](image)

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

Note
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. 141) or Change a Build Project’s Settings (Console) (p. 156).

Important
If you enable webhooks for an AWS CodeBuild project, and the project is used as a build step in AWS CodePipeline, then two identical builds will be created for each commit. One build is triggered through webhooks; and one through AWS CodePipeline. Because billing is on a per-build basis, you will be billed for both builds. Therefore, if you are using AWS CodePipeline, we recommend that you disable webhooks in CodeBuild. In the AWS CodeBuild console, uncheck the webhook box. For more information, see step 9 in Change a Build Project’s Settings (Console) (p. 156)

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. 101).
   - 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.

Select the **Privileged** check box only if you plan to use this build project to build Docker images, and the build environment image you chose is not one provided by AWS CodeBuild with Docker support. Otherwise, all associated builds that attempt to interact with the Docker daemon will fail. Note that you must also start the Docker daemon so that your builds can interact with it as needed. One way to do this is to initialize the Docker daemon in the **install** phase of your build spec by running the following build commands. (Do not run the following build commands if you chose a build environment image provided by AWS CodeBuild with Docker support.)

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
  host=tcp://0.0.0.0:2375 --storage-driver=overlay&
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

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**.

**Important**

We strongly discourage using environment variables to store sensitive values, especially AWS access key IDs and secret access keys. Environment variables can be displayed in plain text using tools such as the AWS CodeBuild console and the AWS CLI. To store and retrieve sensitive values, we recommend your build commands use the AWS CLI to interact with the Amazon EC2 Systems Manager Parameter Store. The AWS CLI comes preinstalled and preconfigured on all build environments provided by AWS CodeBuild. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store CLI Walkthrough in the *Amazon EC2 Systems Manager User Guide*.

20. 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**).

**Note**

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. 167), and then skip to step 31 of this procedure.

27. Open the Amazon S3 console at [https://console.aws.amazon.com/s3/](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 codepipeline-**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.

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


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:

```plaintext
codepipeline:*  
iampass:ListRoles  
iampass:PassRole  
s3:createBucket  
s3:getBucketPolicy  
s3:getObject  
s3:listAllMyBuckets  
s3:listBucket  
s3:PutBucketPolicy  
codecommit:listBranches  
codecommit:listRepositories  
codedeploy:listApplications  
codedeploy:listDeploymentGroups  
elasticbeanstalk:listApplications  
elasticbeanstalk:listEnvironments  
lambda:getFunctionConfiguration  
lambda:listFunctions  
opworks:listStacks  
opworks:listApps  
opworks:listLayers
```

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

5. Choose Edit.
6. Immediately after the Source stage, choose add (+) next to Stage.

Note
This procedure assumes you want to add a test stage to your pipeline. To add a test stage 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 test stage immediately after the Source stage. To add the test stage somewhere else, choose the add symbol in the desired place.

7. For Enter stage name, type the name of the test stage (for example, Test). If you choose a different name, use it throughout this procedure.

8. Inside of the selected stage, choose add (+) next to Action.

Note
This procedure assumes you want to add the test action inside of a test stage. To add the test 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 test action.

9. In the Add action pane, for Action category, choose Test.

10. In Test actions, for Action name, type a name for the action (for example, Test). If you choose a different name, use it throughout this procedure.

11. For Test 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 enable webhooks for an AWS CodeBuild project, and the project is used as a build step in AWS CodePipeline, then two identical builds will be created for each commit. One build is triggered through webhooks; and one through AWS CodePipeline. Because billing is on a per-build basis, you will be billed for both builds. Therefore, if you are using AWS CodePipeline, we recommend that you disable webhooks in CodeBuild. In the AWS CodeBuild console, clear the webhook box. For more information, see step 9 in Change a Build Project’s Settings (Console) (p. 156)

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. 101).
   - 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. (Optional) 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).

Select the **Privileged** check box only if you plan to use this build project to build Docker images, and the build environment image you chose is not one provided by AWS CodeBuild with Docker.
support. Otherwise, all associated builds that attempt to interact with the Docker daemon will fail. Note that you must also start the Docker daemon so that your builds can interact with it as needed. One way to do this is to initialize the Docker daemon in the install phase of your build spec by running the following build commands. (Do not run the following build commands if you chose a build environment image provided by AWS CodeBuild with Docker support.)

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcp://0.0.0.0:2375 --storage-driver=overlay&
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

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**.

**Important**

We strongly discourage using environment variables to store sensitive values, especially AWS access key IDs and secret access keys. Environment variables can be displayed in plain text using tools such as the AWS CodeBuild console and the AWS CLI. To store and retrieve sensitive values, we recommend your build commands use the AWS CLI to interact with the Amazon EC2 Systems Manager Parameter Store. The AWS CLI comes preinstalled and preconfigured on all build environments provided by AWS CodeBuild. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store CLI Walkthrough in the Amazon EC2 Systems Manager User Guide.

20. Choose **Save build project**.
21. For **Input artifacts #1**, type the **Output artifact** value you noted in step 4 of this procedure.
22. (Optional) If you want your test action to produce an output artifact, and you set up your build spec accordingly, then for **Output artifact #1**, type the value you want to assign to the output artifact.
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 test results. In the pipeline's **Test** stage, choose the AWS CodeBuild hyperlink to open the related build project page in the AWS CodeBuild console.

27. On the build project page, in the **Build history** area, choose the related **Build run** hyperlink.
28. On the build run page, in the **Build logs** area, choose the **View entire log** hyperlink to open the related build log in the Amazon CloudWatch console.
29. Scroll through the build log to view the test results.
Use AWS CodeBuild with Jenkins

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. This eliminates the need for you to provision, configure, and manage Jenkins build nodes.

To get the plugin, download the aws-codebuild.hpi file. For instructions for installing, configuring, and running the plugin from within Jenkins, see the AWS CodeBuild CodeBuild Plugin page on the Jenkins website.
Working with Build Projects and Builds in AWS CodeBuild

To get started, follow the steps in Create a Build Project (p. 141), and then follow the steps in Run a Build (p. 161). For more information about build projects and builds, see the following topics.

Topics
• Working with Build Projects (p. 141)
• Working with Builds in AWS CodeBuild (p. 161)

Working with Build Projects

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 (p. 141)
• View a List of Build Project Names in AWS CodeBuild (p. 153)
• View a Build Project's Details in AWS CodeBuild (p. 155)
• Change a Build Project's Settings in AWS CodeBuild (p. 156)
• Delete a Build Project in AWS CodeBuild (p. 160)

Create a Build Project in AWS CodeBuild

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

Topics
• Prerequisites (p. 141)
• Create a Build Project (Console) (p. 141)
• Create a Build Project (AWS CLI) (p. 146)
• Create a Build Project (AWS SDKs) (p. 153)

Prerequisites

Answer the questions in Plan a Build (p. 92).

Create a Build Project (Console)

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. Select the **Build Badge** check box to make your project’s build status visible and embeddable. For more information, see **Build Badges Sample (p. 39)**. Change the value for **Git clone depth**. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

   - If you chose **GitHub**, follow the instructions to connect (or reconnect) with GitHub. On the GitHub **Authorize application** page, for **Organization access**, choose **Request access** next to each repository you want AWS CodeBuild to be able to access. After you choose **Authorize application**, back in the AWS CodeBuild console, for **Repository**, choose the name of the repository that contains the source code.

     - Change the value for **Git clone depth**. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

     - For **GitHub webhooks**, select **Webhook** if you want AWS CodeBuild to rebuild the source code every time a code change is pushed to this repository.

     - Select **Build Badge** to make your project’s build status visible and embeddable. For more information, see **Build Badges Sample (p. 39)**.

   - If you chose **GitHub Enterprise**, see **GitHub Enterprise Sample (p. 30)** for information about prerequisites.

   - For **Personal Access Token**, paste the token that was copied to your clipboard as part of the prerequisites. Choose **Save Token**. In **Repository URL**, enter the URL for your GitHub Enterprise repository.

     **Note**

     You only need to enter and save the personal access token once. All future AWS CodeBuild projects will use this token.

     - Change the value for **Git clone depth**. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.

     - Select **Webhook** if you want AWS CodeBuild to rebuild the source code every time a code change is pushed to this repository.

     - Select **Insecure SSL** to ignore SSL warnings while connecting to your GitHub Enterprise project repository.

     **Note**

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

     - Select **Build Badge** to make your project’s build status visible and embeddable.

   - If you chose **Bitbucket**, follow the instructions to connect (or reconnect) with Bitbucket. On the Bitbucket **Confirm access to your account** page, for **Organization access**, choose **Grant access**. After you choose **Grant access**, back in the AWS CodeBuild console, for **Repository**, choose the name of the repository that contains the source code. Select **Build Badge** to make your project’s build status visible and embeddable. For more information, see **Build Badges Sample (p. 39)**. Change the value for **Git clone depth**. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.
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.

   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, or if you want to run build commands different from the ones specified for the build phase in the buildspec.yml file in the source code's root directory, choose Insert build commands. For Build command, type the commands you want to run in the build phase. For multiple commands, separate each command by && (for example, mvn test && mvn package). To run commands in other phases, or if you have a long list of commands for the build phase, add a buildspec.yml file to the source code root directory, add the commands to the file, and then choose Use the buildspec.yml file in the source code root directory.

   For more information, see the Build Spec Reference (p. 93).

7. 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 might want to do this if you're only running build tests or you want to push a Docker image to an Amazon ECR repository.
   • To store the build output in an Amazon S3 bucket, choose Amazon S3, and then do the following:
     • If you want to use your project name for the build output ZIP file or folder, leave Artifacts name blank. Otherwise, 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 earlier 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. 94).

8. In Cache, do one of the following:

   • If you do not want to use a cache, choose No cache.
   • To use a cache, choose Amazon S3, and then do the following:
     • For Bucket, choose the name of the Amazon S3 bucket where the cache is stored.
     • (Optional) For Path prefix, type an Amazon S3 path prefix. The Path prefix value is similar to a directory name that enables you to store the cache under the same directory in a bucket.

       **Important**
       Do not append "/" to the end of Path prefix.
Using a cache saves considerable build time because reusable pieces of the build environment are stored in the cache and used across builds.

9. In **Service role**, do one of the following:
   - If you do not have an AWS CodeBuild service role, choose **Create a service role in your account**. In **Role name**, accept the default name or type your own.
   - 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 is updated to work with the other build project. A service role can work with up to 10 build projects.

10. In **VPC**, do one of the following:
   - If you are not using a VPC for your project, choose **No VPC**.
   - If you are using want AWS CodeBuild to work with your VPC:
     - For **VPC**, choose the VPC ID that AWS CodeBuild uses.
     - For **Subnets**, choose the subnets that include resources that AWS CodeBuild uses.
     - For **Security Groups**, choose the security groups that AWS CodeBuild uses to allow access to resources in the VPCs.

   For more information, see **Use AWS CodeBuild with Amazon Virtual Private Cloud** (p. 112).

11. 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 is not displayed. Skip to step 20 of this procedure.
   For information about changing default settings, see **Change a Build Project's Settings (Console)** (p. 156).

12. (Optional) For **Timeout**, specify a value between 5 minutes and 480 minutes (8 hours) after which AWS CodeBuild stops the build if it is not complete. If **hours** and **minutes** are left blank, the default value of 60 minutes is used.

13. (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 CMK. Use the format `arn:aws:kms:region-ID:account-ID:key/key-ID`.

14. (Optional) Select **Privileged** only if you plan to use this build project to build Docker images, and the build environment image you chose is not provided by AWS CodeBuild with Docker support. Otherwise, all associated builds that attempt to interact with the Docker daemon fail. You must also start the Docker daemon so that your builds can interact with it. One way to do this is to initialize the Docker daemon in the `install` phase of your build spec by running the following build commands. Do not run these commands if you chose a build environment image provided by AWS CodeBuild with Docker support.

   ```sh
   - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcp://0.0.0.0:2375 --storage-driver=overlay
   ```
15. (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.)

16. For Compute type, choose one of the available options.

17. For Environment variables, type the name, value, and type of each environment variable for builds to use. Use Add row to add an environment variable.

   **Note**
   AWS CodeBuild will set the environment variable for your AWS region automatically. If you do not add them to your buildspec.yml, then the following environment variables must be set:

   - AWS_ACCOUNT_ID
   - IMAGE_REPO_NAME
   - IMAGE_TAG

   Others can see environment variables by using the AWS CodeBuild console and the AWS CLI. If you have no concerns about the visibility of your environment variable, set the Name and Value fields, and then set Type to Plaintext.

   We recommend that you store an environment variable with a sensitive value, such as an AWS access key ID, an AWS secret access key, or a password as a parameter in Amazon EC2 Systems Manager Parameter Store. For Type, choose Parameter Store. For Name, type an identifier for AWS CodeBuild to reference. For Value, type the parameter's name as stored in Amazon EC2 Systems Manager Parameter Store. Using a parameter named /CodeBuild/dockerLoginPassword as an example, for Type, choose Parameter Store. For Name, type LOGIN_PASSWORD. For Value, type /CodeBuild/dockerLoginPassword.

   **Important**
   We recommend that you store parameters in Amazon EC2 Systems Manager Parameter Store with parameter names that start with /CodeBuild/ (for example, /CodeBuild/dockerLoginPassword). You can use the AWS CodeBuild console to create a parameter in Amazon EC2 Systems Manager. Choose Create a parameter, and then follow the instructions in the dialog box. (In that dialog box, for KMS key, you can optionally specify the ARN of an AWS KMS key in your account. Amazon EC2 Systems Manager uses this key to encrypt the parameter's value during storage and decrypt during retrieval.) If you use the AWS CodeBuild console to create a parameter, the console starts the parameter name with /CodeBuild/ as it is being stored. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the Amazon EC2 Systems Manager User Guide.

   If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the ssm:GetParameters action. If you chose Create a service role in your account earlier, then AWS CodeBuild includes this action in the default service role for your build project automatically. However, if you chose Choose an existing service role from your account, then you must include this action to your service role separately.

   If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store with parameter names that do not start with /CodeBuild/, and you chose Create a service role in your account, then you must update that service role to allow access to parameter names that do not start with /CodeBuild/. This is because that service role allows access only to parameter names that start with /CodeBuild/.
Environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named `MY_VAR` with a value of `my_value`, and you set an environment variable named `MY_VAR` with a value of `other_value`, then `my_value` is replaced by `other_value`. Similarly, if the Docker image already contains an environment variable named `PATH` with a value of `/usr/local/sbin:/usr/local/bin`, and you set an environment variable named `PATH` with a value of `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`.

Do not set any environment variable with a name that begins with `CODEBUILD_`. This prefix is reserved for internal use.

If an environment variable with the same name is defined in multiple places, the value is determined as follows:

- The value in the start build operation call takes highest precedence.
- The value in the build project definition takes next precedence.
- The value in the build spec declaration takes lowest precedence.

18. (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.

19. Choose Continue.

20. 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 information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 185).

1. Run the create-project command:

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

JSON-formatted data appears in the output. Copy the data to a file (for example, `create-project.json`) in a location on the local computer or instance where the AWS CLI is installed. Modify the copied data as follows, and save your results.

```
{
  "name": "project-name",
  "description": "description",
  "source": {
    "type": "source-type",
    "location": "source-location",
    "gitCloneDepth": "gitCloneDepth",
    "buildspec": "buildspec",
    "auth": {
      "type": "auth-type",
      "resource": "resource",
      "badgeEnabled": "badgeEnabled",
      "InsecureSsl": "InsecureSsl"
    }
  },
  "artifacts": {
    "type": "artifacts-type",
    "location": "artifacts-location",
    "path": "path",
    "namespaceType": "namespaceType",
```

API Version 2016-10-06

146
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, GITHUB_ENTERPRISE, BITBUCKET, 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 `bucket-name/path/to/object-name.zip` instead.)

- For GitHub, the HTTPS clone URL to the repository that contains the source code and the build spec. Also, you must connect your AWS account to your GitHub account. To do this, use the AWS CodeBuild console to create a build project. When you use the console to connect (or reconnect) with GitHub, on the GitHub **Authorize application** page, for **Organization access**, choose **Request access** next to each repository you want AWS CodeBuild to be able to access. Choose **Authorize application**. (After you have connected to your GitHub account, you do not need to finish creating the build project. You can close the AWS CodeBuild console.) To instruct AWS CodeBuild to use this connection, in the `source` object, set the `auth` object's `type` value to `OAUTH`.

- For GitHub Enterprise, the HTTP or HTTPS clone URL to the repository that contains the source code and the build spec. You must also connect your AWS account to your GitHub Enterprise account. To do this, use the AWS CodeBuild console to create a build project. First, create a personal access token in GitHub Enterprise. Copy this token to your clipboard, so that it can be used when creating your AWS CodeBuild project. For more information, see Creating a Personal Access Token in GitHub Enterprise on the GitHub Help website. When you use the console to create your AWS CodeBuild project, in **Source: What to build**, for **Source provider**, choose GitHub Enterprise. For **Personal Access Token**, paste the token that was copied to your clipboard. Choose **Save Token**. Your AWS CodeBuild account is now connected to your GitHub Enterprise account.

- For Bitbucket, the HTTPS clone URL to the repository that contains the source code and the build spec. You must also connect your AWS account to your Bitbucket account. To do this, use the AWS CodeBuild console to create a build project. When you use the console to connect (or reconnect) with Bitbucket, on the Bitbucket **Confirm access to your account** page, choose **Grant access**. (After you have connected to your Bitbucket account, you do not need to finish creating the build project. You can close the AWS CodeBuild console.) To instruct AWS CodeBuild to use this connection, in the `source` object, set the `auth` object's `type` value to `OAUTH`.

- For AWS CodePipeline, do not specify a `location` value for `source`. It is 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.

- `gitCloneDepth`: Optional value. The depth of history to download. Minimum value is 0. If this value is 0, greater than 25, or not provided, then the full history is downloaded with each build project. If your source type is Amazon S3, this value is not supported.

- `buildspec`: Optional value. The build specification definition or file to use. If this value is set, it can be either an inline build spec definition or the path to an alternate build spec file relative to the value of the built-in `CODEBUILD_SRC_DIR` environment variable. 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 Build Spec File Name and Storage Location (p. 93).

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

- `badgeEnabled`: Optional value. To include build badges with your AWS CodeBuild project, you must specify `badgeEnabled` with a value of `true`. For more information, see Build Badges Sample with AWS CodeBuild (p. 39).

- `InsecureSsl`: Optional value. This is used with GitHub Enterprise only. Set this value to `true` to ignore SSL warnings while connecting to your GitHub Enterprise project repository. The default value is `false`. `InsecureSsl` should be used for testing purposes only. It should not be used in a production environment.

- 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 uses namespaceType (if specified) and **artifacts-name** to determine the path and name of the build output ZIP file or folder. For example, if you specify MyPath for path 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 uses path (if specified) and **artifacts-name** to determine the path and name of the build output ZIP file or folder. For example, if you specify MyPath for path, BUILD_ID for namespaceType, and MyArtifact.zip for **artifacts-name**, then the path and name would be MyPath/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 cache object, information about this build project's cache settings. These settings include the following:
  
  - **CacheType**: Required value. Valid values are S3 or NONE.
• CacheLocation: Required value unless you set CacheType to NONE. If you specified S3 for CacheType, then this is the ARN of the S3 bucket and the path prefix. For example, if your Amazon S3 bucket name is my-bucket, and your path prefix is build-cache, then acceptable formats for your CacheLocation are my-bucket/build-cache or aws:s3:::my-bucket/build-cache.

• serviceRole: Required value. The ARN of the service role AWS CodeBuild uses to interact with services on behalf of the IAM user (for example, arn:aws:iam::account-id:role/role-name).

For the optional vpcConfig object, information about your VPC configuration. These settings include:

• vpcId: Required value. The VPC ID that AWS CodeBuild uses. To get a list of all Amazon VPC IDs in your region, run this command:

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

• subnets: Required value. The subnet IDs that include resources used by AWS CodeBuild. To obtain these IDs, run this command:

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

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

• securityGroupIds: Required value. The security group IDs used by AWS CodeBuild to allow access to resources in the VPCs. To obtain these IDs, run this command:

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

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

For the required environment object, information about this project's 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 used by this build environment. 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. For more information, see Docker Images Provided by AWS CodeBuild (p. 101).

• computeType: Required value. A category corresponding to the number of CPU cores and memory used by this build environment. Allowed values include BUILD_GENERAL1_SMALL, BUILD_GENERAL1_MEDIUM, and BUILD_GENERAL1_LARGE.

• certificate: Optional value. The ARN of the AWS S3 bucket, path prefix and object key containing the PEM encoded certificate. The object key can be either just the .pem file or a .zip file containing the pem encoded certificate. For example, if your Amazon S3 bucket name is my-bucket, your path prefix is cert, and your object key name is certificate.pem, then acceptable formats for your certificate are my-bucket/cert/certificate.pem or arn:aws:s3:::my-bucket/cert/certificate.pem.

For the optional environmentVariables array, information about any environment variables you want to specify for this build environment. Each environment variable is
expressed as an object containing a name, value, and type of `environmentVariable-name`, `environmentVariable-value`, and `environmentVariable-type`.

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

We recommend you store an environment variable with a sensitive value, such as an AWS access key ID, an AWS secret access key, or a password as a parameter in Amazon EC2 Systems Manager Parameter Store. For `environmentVariable-name`, for that stored parameter, set an identifier for AWS CodeBuild to reference. For `environmentVariable-value`, set the parameter's name as stored in Amazon EC2 Systems Manager Parameter Store. Set `environmentVariable-type` to PARAMETER_STORE. Using a parameter named `/CodeBuild/dockerLoginPassword` as an example, set `environmentVariable-name` to LOGIN_PASSWORD. Set `environmentVariable-value` to `/CodeBuild/dockerLoginPassword`. Set `environmentVariable-type` to PARAMETER_STORE.

**Important**

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the ssm:GetParameters action. If you chose Create a service role in your account earlier, then AWS CodeBuild includes this action in the default service role for your build project automatically. However, if you chose Choose an existing service role from your account, then you must include this action to your service role separately.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store with parameter names that do not start with `/CodeBuild/`, and you chose Create a service role in your account, then you must update that service role to allow access to parameter names that do not start with `/CodeBuild/`. This is because that service role allows access only to parameter names that start with `/CodeBuild/`. Any environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named MY_VAR with a value of my_value, and you set an environment variable named MY_VAR with a value of other_value, then my_value is replaced by other_value. Similarly, if the Docker image already contains an environment variable named PATH with a value of /usr/local/sbin:/usr/local/bin, and you set an environment variable named PATH with a value of $PATH:/usr/share/ant/bin, then /usr/local/sbin:/usr/local/bin is replaced by the literal value $PATH:/usr/share/ant/bin. Do not set any environment variable with a name that begins with CODEBUILD_. This prefix is reserved for internal use.

If an environment variable with the same name is defined in multiple places, the value is determined as follows:

- The value in the start build operation call takes highest precedence.
- The value in the build project definition takes next precedence.
- The value in the build spec declaration takes lowest precedence.

- You must specify `privilegedMode` with a value of `true` only if you plan to use this build project to build Docker images, and the build environment image you specified is not provided by AWS CodeBuild with Docker support. Otherwise, all associated builds that attempt to interact with the Docker daemon fail. You must also start the Docker daemon so that your builds can interact with it. One way to do this is to initialize the Docker daemon in the install phase of your build spec by running the following build commands. Do not run these commands if you specified a build environment image provided by AWS CodeBuild with Docker support.

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

API Version 2016-10-06
151
Create a Build Project

- **timeoutInMinutes**: Optional value. The number of minutes, between 5 to 480 (8 hours), after which AWS CodeBuild stops the build if it is not complete. If not specified, the default of 60 is 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 uses to encrypt the build output. If you specify an alias, use the format `arn:aws:kms:region-ID:account-ID:key/key-ID` or, if an alias exists, use the format `alias/key-alias`. If not specified, the AWS-managed CMK for Amazon S3 is used.

- For the optional `tags` array, information about any tags you want to associate with this build project. You can specify up to 50 tags. These tags can be used by any AWS service that supports AWS CodeBuild build project tags. Each tag is expressed as an object containing a `key` and `value` of `tag-key` and `tag-value`.

  For an example, see To create the build project (AWS CLI) (p. 10).

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

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

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

   ```json
   {
     "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",
             "type": "environmentVariable-type"
           }
         ]
       },
       "source": {
         "type": "source-type",
         "location": "source-location",
       }
   }
   ```
• 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 `createdAt` value represents the time, in Unix time format, when the build project was created.
  • The `arn` value represents the ARN of the build project.

**Note**
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. 159).

To start running a build, see Run a Build (AWS CLI) (p. 163).

If your source code is stored in a GitHub repository, and you want AWS CodeBuild to rebuild the source code every time a code change is pushed to the repository, see Start Running Builds Automatically (AWS CLI) (p. 166).

**Create a Build Project (AWS SDKs)**

For information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and Tools Reference (p. 186).

**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, or AWS SDKs.

**Topics**

- View a List of Build Project Names (Console) (p. 153)
- View a List of Build Project Names (AWS CLI) (p. 154)
- View a List of Build Project Names (AWS SDKs) (p. 155)

**View a List of Build Project Names (Console)**

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 and Tools Reference (p. 186).

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, or AWS SDKs.

Topics
- View a Build Project's Details (Console) (p. 155)
- View a Build Project's Details (AWS CLI) (p. 155)
- View a Build Project's Details (AWS SDKs) (p. 156)

View a Build Project's Details (Console)
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. 185).

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. 154).

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.

```json
{
```
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. 146).

**View a Build Project's Details (AWS SDKs)**

For more information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and Tools Reference (p. 186).

**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, or AWS SDKs.

**Topics**

- Change a Build Project's Settings (Console) (p. 156)
- Change a Build Project's Settings (AWS CLI) (p. 159)
- Change a Build Project's Settings (AWS SDKs) (p. 160)

**Change a Build Project's Settings (Console)**

2. In the navigation pane, choose Build projects.
3. Do one of the following:
   - Choose the radio button next to the build project you want to change, choose Actions, and then choose Update.
   - Choose the link for the build project you want to change, and then choose Edit project.

**Note**

By default, only the 10 most recent build projects are displayed. 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, type a description in Description.

For more information about settings referred to in this procedure, see Create a Build Project (Console) (p. 141).
5. To change information about the source code location, in the **Source: What to build** area, choose **Update source**. Change the displayed fields depending on the source provider type (for example, **Source provider**, **Bucket**, **S3 object**, or **Repository**).
   - If your Source provider is AWS CodeCommit, BitBucket, GitHub, or GitHub Enterprise, you can change the value for **Git clone depth**. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose **Full**.
   - If your source code is stored in a GitHub or GitHub Enterprise repository, and you want AWS CodeBuild to rebuild the source code every time a code change is pushed to the repository, select **Webhook**.
   - If your **Source provider** is AWS CodeCommit, BitBucket, GitHub, or GitHub Enterprise, a **Build Badge** check box is available. Select **Build Badge** to make your project's build status visible and embeddable. For more information, see **Build Badges Sample** (p. 39).

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

6. To change information about the build environment, in **Environment: How to build**, choose **Update image**. Make changes appropriate for 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 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 cache, in **Cache**, do one of the following:
   - If you previously chose a cache but now you do not want to use a cache, choose **No cache**.
   - If you previously chose **No cache** but now you want to use a cache, choose **Amazon S3**, and then do the following:
     - For **Bucket**, choose the name of the Amazon S3 bucket where the cache is stored.
     - (Optional) For **Path prefix**, type an Amazon S3 path prefix. The **Path prefix** value is similar to a directory name that enables you to store the cache under the same directory in a bucket.

   **Important**
   Do not append "/" to the end of **Path prefix**.

   Using a cache saves considerable build time because reusable pieces of the build environment are stored in the cache and used across builds.

10. 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 is updated to work with the other build project. A service role can work with up to 10 build projects.

11. In **VPC**, do one of the following:
If you are not using a VPC for your project, choose **No VPC**. If you want AWS CodeBuild to work with your VPC:
- For **VPC**, choose the VPC ID that AWS CodeBuild uses.
- For **Subnets**, choose the subnets that include resources that AWS CodeBuild uses.
- For **Security Groups**, choose the security groups that AWS CodeBuild uses to allow access to resources in the VPCs.

For more information, see [Use AWS CodeBuild with Amazon Virtual Private Cloud](p. 112).

12. 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 is 60 minutes.

13. 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, AWS CodeBuild uses the AWS-managed CMK for Amazon S3 in your AWS account instead.

14. If you plan to use this build project to build Docker images and the specified build environment is not provided by AWS CodeBuild with Docker support, in **Show advanced settings**, select **Privileged**. Otherwise, all associated builds that attempt to interact with the Docker daemon fail. You must also start the Docker daemon so that your builds can interact with it as needed. One way to do this is to initialize the Docker daemon in the install phase of your build spec by running the following build commands. (Do not run the following build commands if the specified build environment image is provided by AWS CodeBuild with Docker support.)

   ```
   - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcp://0.0.0.0:2375 --storage-driver=overlay&
   - timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
   ```

15. To change information about the way build output artifacts are stored, in **Show advanced settings**, change the value of **Artifacts packaging**.

16. To change the amount of memory and vCPUs that are used to run builds, in **Show advanced settings**, change the value of **Compute type**.

17. To change information about environment variables you want builds to use, in **Show advanced settings**, for **Environment variables**, change the values for **Name**, **Value**, and **Type**. Use **Add row** to add an environment variable. Choose the delete (X) button next to an environment variable you no longer want to use.

   Others can see environment variables by using the AWS CodeBuild console and the AWS CLI. If you have no concerns about the visibility of your environment variable, set the **Name** and **Value** fields, and then set **Type** to **Plaintext**.

   We recommend that you store an environment variable with a sensitive value, such as an AWS access key ID, an AWS secret access key, or a password as a parameter in Amazon EC2 Systems Manager Parameter Store. For **Type**, choose **Parameter Store**. For **Name**, type an identifier for AWS CodeBuild to reference. For **Value**, type the parameter's name as stored in Amazon EC2 Systems Manager Parameter Store. Using a parameter named `/CodeBuild/dockerLoginPassword` as an example, for **Type** choose **Parameter Store**. For **Name**, type `LOGIN_PASSWORD`. For **Value**, type `/CodeBuild/dockerLoginPassword`.

   **Important**
   We recommend that you store parameters in Amazon EC2 Systems Manager Parameter Store with parameter names that start with `/CodeBuild/` (for example, `/CodeBuild/dockerLoginPassword`). You can use the AWS CodeBuild console to create a parameter in Amazon EC2 Systems Manager. Choose **Create a parameter**, and then follow the
instructions in the dialog box. (In that dialog box, for KMS key, you can optionally specify the ARN of an AWS KMS key in your account. Amazon EC2 Systems Manager uses this key to encrypt the parameter’s value during storage and decrypt during retrieval.) If you use the AWS CodeBuild console to create a parameter, the console starts the parameter name with /CodeBuild/ as it is being stored. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the Amazon EC2 Systems Manager User Guide.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the ssm:GetParameters action. If you chose Create a service role in your account earlier, then AWS CodeBuild includes this action in the default service role for your build project automatically. However, if you chose Choose an existing service role from your account, then you must include this action to your service role separately.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store with parameter names that do not start with /CodeBuild/, and you chose Create a service role in your account, then you must update that service role to allow access to parameter names that do not start with /CodeBuild/. This is because that service role allows access only to parameter names that start with /CodeBuild/.

Environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named MY_VAR with a value of my_value, and you set an environment variable named MY_VAR with a value of other_value, then my_value is replaced by other_value. Similarly, if the Docker image already contains an environment variable named PATH with a value of /usr/local/sbin:/usr/local/bin, and you set an environment variable named PATH with a value of $PATH:/usr/share/ant/bin, then /usr/local/sbin:/usr/local/bin is replaced by the literal value $PATH:/usr/share/ant/bin.

Do not set any environment variable with a name that begins with CODEBUILD_. This prefix is reserved for internal use.

If an environment variable with the same name is defined in multiple places, 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.

18. 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.

19. 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. 185).

1. Run the update-project command as follows:

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

JSON-formatted data appears in the output. Copy the data to a file (for example, update-project.json) in a location on the local computer or instance where the AWS CLI is installed. Then modify the copied data as described in Create a Build Project (AWS CLI) (p. 146), 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. 146) appears in the output.

Change a Build Project's Settings (AWS SDKs)

For information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and Tools Reference (p. 186).

Delete a Build Project in AWS CodeBuild

You can use the AWS CodeBuild console, AWS CLI, or AWS SDKs 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 (Console) (p. 160)
- Delete a Build Project (AWS CLI) (p. 160)
- Delete a Build Project (AWS SDKs) (p. 161)

Delete a Build Project (Console)

2. In the navigation pane, choose Build projects.
3. Do one of the following:
   - Choose the radio button next to the build project you want to delete, 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. 185).
1. Run the `delete-project` command:

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

   Replace the following placeholder:

   - `name`: Required string. The name of the build project to delete. To get a list of available build projects, run the `list-projects` command. For more information, see View a List of Build Project Names (AWS CLI) (p. 154).

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 and Tools Reference (p. 186).

---

**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 (p. 161)
- View Build Details in AWS CodeBuild (p. 167)
- View a List of Build IDs in AWS CodeBuild (p. 169)
- View a List of Build IDs for a Build Project in AWS CodeBuild (p. 171)
- Stop a Build in AWS CodeBuild (p. 172)
- Delete Builds in AWS CodeBuild (p. 173)

**Run a Build in AWS CodeBuild**

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

**Topics**

- Run a Build (Console) (p. 161)
- Run a Build (AWS CLI) (p. 163)
- Start Running BuildsAutomatically (AWS CLI) (p. 166)
- Stop Running Builds Automatically (AWS CLI) (p. 167)
- Run a Build (AWS SDKs) (p. 167)

**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. 125).

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 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, the latest version will be used.
   - For AWS CodeCommit, for the optional Source version value, for Branch, choose the name of the branch that contains the version of the source code you want to build. For Source version, accept the displayed HEAD commit ID or type a different one. If Source version is blank, the default branch's HEAD commit ID is used. You cannot type a tag name for Source version. To specify a tag, type the tag's commit ID. Change the value for Git clone depth. This will create a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose Full.
   - For GitHub or GitHub Enterprise, for the optional Source version value, type a commit ID, a pull request ID, a branch name, or a tag name that corresponds to the version of the source code you want to build. If you specify a branch name, the branch's HEAD commit ID is used. If Source version is blank, the default branch's HEAD commit ID is used. Change the value for Git clone depth. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose Full.
   - For Bitbucket, for the optional Source version value, type a commit ID, a branch name, or a tag name that corresponds to the version of the source code you want to build. If you specify a branch name, the branch's HEAD commit ID is used. If Source version is blank, the default branch's HEAD commit ID is used. Change the value for Git clone depth. This creates a shallow clone with a history truncated to the specified number of commits. If you want a full clone, choose Full.

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 for Name, Value, and Type. Use Add row to add a new environment variable for this build only. Choose the delete (X) button next to an environment variable you do not want to use in this build.

   Others can see an environment variable by using the AWS CodeBuild console and the AWS CLI. If you have no concerns about the visibility of your environment variable, set the Name and Value fields, and then set Type to Plaintext.

   We recommend that you store an environment variable with a sensitive value, such as an AWS access key ID, an AWS secret access key, or a password as a parameter in Amazon EC2 Systems Manager Parameter Store. For Type, choose Parameter Store. For Name, type an identifier for AWS
Important

We recommend that you store parameters in Amazon EC2 Systems Manager Parameter Store with parameter names that start with `/CodeBuild/` (for example, `/CodeBuild/dockerLoginPassword`). You can use the AWS CodeBuild console to create a parameter in Amazon EC2 Systems Manager. Choose Create a parameter, and then follow the instructions in the dialog box. (In that dialog box, for KMS key, you can optionally specify the ARN of an AWS KMS key in your account. Amazon EC2 Systems Manager uses this key to encrypt the parameter's value during storage and decrypt during retrieval.) If you use the AWS CodeBuild console to create a parameter, the console starts the parameter with `/CodeBuild/` as it is being stored. For more information, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the Amazon EC2 Systems Manager User Guide.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the `ssm:GetParameters` action. If you chose Create a service role in your account earlier, then AWS CodeBuild includes this action in the default service role for your build project automatically. However, if you chose Choose an existing service role from your account, then you must include this action in your service role separately.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store with parameter names that do not start with `/CodeBuild/`, and you chose Create a service role in your account, then you must update that service role to allow access to parameter names that do not start with `/CodeBuild/`. Any environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named `MY_VAR` with a value of `my_value`, and you set an environment variable named `MY_VAR` with a value of `other_value`, then `my_value` is replaced by `other_value`. Similarly, if the Docker image already contains an environment variable named `PATH` with a value of `/usr/local/sbin:/usr/local/bin`, and you set an environment variable named `PATH` with a value of `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`.

Do not set any environment variable with a name that 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. 167).

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. 128). For more information about using the AWS CLI with AWS CodeBuild, see the Command Line Reference (p. 185).

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. 153).

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

```json
{
  "projectName": "projectName",
  "sourceVersion": "sourceVersion",
  "artifactsOverride": {
    "type": "type",
    "location": "location",
    "path": "path",
    "namespaceType": "namespaceType",
    "name": "artifactsOverride-name",
    "packaging": "packaging"
  },
  "environmentVariablesOverride": [
    {
      "name": "environmentVariablesOverride-name",
      "value": "value",
      "type": "environmentVariablesOverride-type"
    }
  ],
  "buildspecOverride": "buildspecOverride",
  "gitCloneDepthOverride": "gitCloneDepthOverride",
  "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 source code to be built, as follows:
  - For Amazon S3, the version ID that corresponds to the version of the input ZIP file you want to build. If sourceVersion is not specified, then the latest version is used.
  - For AWS CodeCommit, the commit ID that corresponds to the version of the source code you want to build. If sourceVersion is not specified, the default branch's HEAD commit ID is used. (You cannot specify a tag name for sourceVersion, but you can specify the tag's commit ID.)
  - For GitHub, the commit ID, pull request ID, branch name, or tag name that corresponds to the version of the source code you want to build. If a pull request ID is specified, it must use the format pr/pull-request-ID (for example, pr/25). If a branch name is specified, the branch's HEAD commit ID is used. If sourceVersion is not specified, the default branch's HEAD commit ID is used.
For Bitbucket, the commit ID, branch name, or tag name that corresponds to the version of the source code you want to build. If a branch name is specified, the branch's HEAD commit ID is used. If `sourceVersion` is not specified, the default branch's HEAD commit ID is used.

- **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 that you want to override for this build.
- **environmentVariablesOverride-type**: Optional string. The type of environment variable in the build project whose value you want to override for this build.

**Important**

We recommend you store an environment variable with a sensitive value, such as an AWS access key ID, an AWS secret access key, or a password as a parameter in Amazon EC2 Systems Manager Parameter Store. AWS CodeBuild can use a parameter stored in Amazon EC2 Systems Manager Parameter Store only if that parameter's name starts with `/CodeBuild/` (for example, `/CodeBuild/dockerLoginPassword`). You can use the AWS CodeBuild console to create a parameter in Amazon EC2 Systems Manager. Choose **Create a parameter**, and then follow the instructions in the dialog box. (In that dialog box, for **KMS key**, you can optionally specify the ARN of an AWS KMS key in your account. Amazon EC2 Systems Manager uses this key to encrypt the parameter's value during storage and decrypt during retrieval.) If you use the AWS CodeBuild console to create a parameter, the console starts the parameter with `/CodeBuild/` as it is being stored. However, if you use the Amazon EC2 Systems Manager Parameter Store console to create a parameter, you must start the parameter's name with `/CodeBuild/`, and you must set **Type** to **Secure String**. For more information, see **Systems Manager Parameter Store** and **Systems Manager Parameter Store Console Walkthrough** in the **Amazon EC2 Systems Manager User Guide**.

If your build project refers to parameters stored in Amazon EC2 Systems Manager Parameter Store, the build project's service role must allow the `ssm:GetParameters` action. If you chose **Create a new service role in your account**, then AWS CodeBuild includes this action in the default service role for your build project automatically. However, if you chose **Choose an existing service role from your account**, then you must include this action in your service role separately.

Environment variables you set replace existing environment variables. For example, if the Docker image already contains an environment variable named `MY_VAR` with a value of `my_value`, and you set an environment variable named `MY_VAR` with a value of `other_value`, then `my_value` is replaced by `other_value`. Similarly, if the Docker image already contains an environment variable named `PATH` with a value of `/usr/local/sbin:/usr/local/bin`, and you set an environment variable named `PATH` with a value of `$PATH:/usr/share/ant/bin`, then `/usr/local/sbin:/usr/local/bin` is replaced by the literal value `$PATH:/usr/share/ant/bin`. Do not set any environment variable with a name that begins with `CODEBUILD_`. This prefix is reserved for internal use.
If an environment variable with the same name is defined in multiple places, the environment variable's value is determined as follows:

- The value in the start build operation call takes highest precedence.
- The value in the build project definition takes next precedence.
- The value in the 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. If this value is set, it can be either an inline build spec definition or the path to an alternate build spec file relative to the value of the built-in `CODEBUILD_SRC_DIR` environment variable.

- `gitCloneDepthOverride`: Optional string. The value of the Git clone depth in the build project whose value you want to override for this build. If your source type is Amazon S3, this value is not supported.

- `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. 146). For a list of the latest settings for a build project, see View a Build Project's Details (p. 155).

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

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

5. If successful, data similar to that described in the To run the build (AWS CLI) (p. 13) procedure appears in the output.

To work with detailed information about this build, make a note of the id value in the output, and then see View Build Details (AWS CLI) (p. 168).

**Start Running Builds Automatically (AWS CLI)**

If your source code is stored in a GitHub or a GitHub Enterprise repository, you can use GitHub webhooks to have AWS CodeBuild rebuild your source code whenever a code change is pushed to the repository.

Run the `create-webhook` command as follows:

```
aws codebuild create-webhook --project-name
```

- where `project-name` is the name of the build project that contains the source code to be rebuilt.

For GitHub, information similar to the following appears in the output:

```
{
   "webhook": {
      "url": "url"
   }
}
```

- where `url` is the URL to the GitHub webhook.

For GitHub Enterprise, information similar to the following appears in the output:

```
API Version 2016-10-06
166
```
Copy the secret key and payload URL from the output. You need them to add a webhook in GitHub Enterprise. In GitHub Enterprise, choose the repository where your AWS CodeBuild project is stored, choose Settings, choose Hooks & services, and then choose Add webhook. Enter the payload URL and secret key, accept the defaults for the other fields, and then choose Add webhook.

Stop Running Builds Automatically (AWS CLI)

If your source code is stored in a GitHub or a GitHub Enterprise repository, you can set up GitHub webhooks to have AWS CodeBuild rebuild your source code whenever a code change is pushed to the repository. For more information, see Start Running Builds Automatically (AWS CLI) (p. 166).

If you have enabled this behavior, you can turn it off by running the delete-webhook command as follows:

```bash
aws codebuild delete-webhook --project-name
```

- where `project-name` is the name of the build project that contains the source code to be rebuilt.

If this command is successful, no information and no errors appear in the output.

**Note**

This deletes the webhook from your AWS CodeBuild project only. You should also delete the webhook from your GitHub or GitHub Enterprise repository.

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 Test Code and Run Builds (p. 123) instead.

For information about using AWS CodeBuild with the AWS SDKs, see the AWS SDKs and Tools Reference (p. 186).

View Build Details in AWS CodeBuild

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

**Topics**

- View Build Details (Console) (p. 167)
- View Build Details (AWS CLI) (p. 168)
- View Build Details (AWS SDKs) (p. 168)
- Build Phase Transitions (p. 168)

View Build Details (Console)


API Version 2016-10-06

167
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. 185).

Run the `batch-get-builds` command:

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

Replace the following placeholder:

- **ids**: Required string. One or more build IDs to view details about. To specify more than one build ID, separate each build ID with a space. You can specify up to 100 build IDs. To get a list of build IDs, see one or more of the following topics:
  - View a List of Build IDs (AWS CLI) (p. 170)
  - View a List of Build IDs for a Build Project (AWS CLI) (p. 171)

For example, if you run this command:

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

If the command is successful, data similar to that described in the To view summarized build information (AWS CLI) (p. 14) 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 and Tools Reference (p. 186).

### Build Phase Transitions

Builds in AWS CodeBuild proceed in phases:
An important point to note here is that the `UPLOAD_ARTIFACTS` phase is always attempted, even if the `BUILD` phase fails.

**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, or AWS SDKs.

**Topics**
- View a List of Build IDs (Console) (p. 170)
- View a List of Build IDs (AWS CLI) (p. 170)
View a List of Build IDs

View a List of Build IDs (AWS SDKs) (p. 171)

View a List of Build IDs (Console)

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. 185).

- Run the list-builds command:

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

  In the preceding command, replace the following placeholders:

  - **sort-order**: Optional string. 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:

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

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

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

  If you run this command again:

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

  A result similar to the following might appear in the output:
{ "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 and Tools Reference (p. 186).

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

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

Topics
- View a List of Build IDs for a Build Project (Console) (p. 171)
- View a List of Build IDs for a Build Project (AWS CLI) (p. 171)
- View a List of Build IDs for a Build Project (AWS SDKs) (p. 172)

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

2. In the navigation pane, choose Build projects. In the list of build projects, in the 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. 185).

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

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

In the preceding command, replace the following placeholders:
- `project-name`: Required string. 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. 154).
• **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:

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

A result like the following might appear in the output:

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

If you run this command again:

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

A result like the following might be output:

```json
{
  "ids": [
    "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE",
    "codebuild-demo-project:a5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...,
    "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
  ]
}
```

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

For more information about using AWS CodeBuild with the AWS SDKs, see the [AWS SDKs and Tools Reference](p. 186).

### Stop a Build in AWS CodeBuild

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

**Topics**

- Stop a Build (Console) (p. 173)
- Stop a Build (AWS CLI) (p. 173)
- Stop a Build (AWS SDKs) (p. 173)
Stop a Build (Console)

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:

  ```bash
  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. 170)
    • View a List of Build IDs for a Build Project (AWS CLI) (p. 171)

  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 and Tools Reference (p. 186).

Delete Builds in AWS CodeBuild

To delete builds in AWS CodeBuild, you can use the AWS CLI, or the AWS SDKs.

Delete Builds (AWS CLI)

Run the `batch-delete-builds` command:

```bash
aws codebuild batch-delete-builds --ids ids
```
In the preceding command, replace the following placeholder:

- **ids**: Required string. The IDs of the builds to delete. To specify multiple builds, separate each build ID with a space. To get a list of build IDs, see the following topics:
  - View a List of Build IDs (AWS CLI) (p. 170)
  - View a List of Build IDs for a Build Project (AWS CLI) (p. 171)

If successful, a `buildsDeleted` array appears in the output, containing the Amazon Resource Name (ARN) of each build that was successfully deleted. Information about builds that were not successfully deleted appears in output within a `buildsNotDeleted` array.

For example, if you run this command:

```bash
aws codebuild batch-delete-builds --ids my-demo-build-project:f8b888d2-5e1e-4032-8645-b115195648EX my-other-demo-build-project:a18bc6ee-e499-4887-b36a-8c90349c7eEX
```

Information similar to the following appears in the output:

```json
```

### Delete Builds (AWS SDKs)

For information about using AWS CodeBuild with the AWS SDKs, see the [AWS SDKs and Tools Reference](p. 186).
Advanced Topics

This section includes several advanced topics that are useful to more experienced AWS CodeBuild users.

Topics
- Advanced Setup (p. 175)
- Command Line Reference for AWS CodeBuild (p. 185)
- AWS SDKs and Tools Reference for AWS CodeBuild (p. 186)
- Authentication and Access Control for AWS CodeBuild (p. 186)
- Logging AWS CodeBuild API Calls with AWS CloudTrail (p. 199)

Advanced Setup

If you follow the steps in Getting Started (p. 4) 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 (p. 175)
- Create an AWS CodeBuild Service Role (p. 179)
- Create and Configure an AWS KMS CMK for AWS CodeBuild (p. 183)
- Install and Configure the AWS CLI (p. 184)

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 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.
Add AWS CodeBuild Access Permissions to an IAM Group or IAM User

- 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:ListGroup  
  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 access 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**.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildDefaultPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*",
        "iam:PassRole"
      ],
      "Resource": "**"
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": ["cloudwatch:*"]
    }
  ]
}
```
Add AWS CodeBuild Access Permissions to an IAM Group or IAM User

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. 186). To restrict access to specific AWS resources, change the value of the Resource object. For more information, see Authentication and Access Control (p. 186).

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

For a user, on the user settings page, on the Permissions tab, choose Add permissions.
11. For a group, on the Attach Policy page, select CodeBuildAccessPolicy, and then choose Attach Policy.

For a user, on the Add permissions page, choose Attach existing policies directly. Select CodeBuildAccessPolicy, choose Next: Review, and then choose Add permissions.

To add 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 access 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.

```json
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "CodeBuildAccessPolicy",
        "Effect": "Allow",
        "Action": [
            "codebuild:*",
            "iam:PassRole"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchLogsAccessPolicy",
        "Effect": "Allow",
        "Action": [
            "logs:FilterLogEvents",
            "logs:GetLogEvents"
        ],
        "Resource": "*"
    },
    {
        "Sid": "S3AccessPolicy",
        "Effect": "Allow",
        "Action": [
            "s3:CreateBucket",
            "s3:GetObject",
            "s3:List*",
            "s3:PutObject"
        ],
        "Resource": "*"
    }
]}
```
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. 141)
- Create a Pipeline that Uses AWS CodeBuild (AWS CodePipeline Console) (p. 125)
- Add an AWS CodeBuild Build Action to a Pipeline (AWS CodePipeline Console) (p. 132)
- Change a Build Project's Settings (Console) (p. 156)

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 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:

```bash
iam:AddRoleToInstanceProfile
iam:AttachRolePolicy
```

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:

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

For an IAM user:

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

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

API Version 2016-10-06

179
AWS CodeBuild User Guide
Create an AWS CodeBuild Service Role

iam:CreateInstanceProfile
iam:CreatePolicy
iam:CreateRole
iam:GetRole
iam:ListAttachedRolePolicies
iam:ListPolicies
iam:ListRoles
iam:PassRole
iam:PutRolePolicy
iam:UpdateAssumeRolePolicy

For more information, see Overview of IAM Policies in the IAM User Guide.

2. In the navigation pane, choose Policies.
3. Choose Create Policy.
4. On the Create Policy page, choose JSON.
5. For the JSON Policy, type the following, and then choose Review Policy:

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

Note
This policy contains statements that allow access to a potentially large number of AWS resources. To restrict AWS CodeBuild to access specific AWS resources, change the value of the Resource array. For more information, see the security documentation for the AWS service.

6. On the Review Policy page, for Policy Name, type a name for the policy (for example, CodeBuildServiceRolePolicy), and then choose Create policy.

Note
If you use a different name, substitute it throughout this procedure.

7. In the navigation pane, choose Roles.

8. Choose Create role.

9. On the Create role page, with AWS Service already selected, choose CodeBuild, as the service that will use this role, and then choose Next:Permissions.

10. On the Attach permissions policies page, select CodeBuildServiceRolePolicy, and then choose Next: Review.

11. On the Create role and review page, for Role name, type a name for the role (for example, CodeBuildServiceRole), and then choose Create role.

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:

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

put-role-policy.json:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchLogsPolicy",
            "Effect": "Allow",
            "Action": [  
```
Create an AWS CodeBuild Service Role

```json
{
    "Statement": [
        {
            "Sid": "CloudTrailAccess",
            "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.

```bash
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.

```json
{
    "Version": "2012-10-17",
    "Id": "...",
    "Statement": [
        ### BEGIN ADDING STATEMENTS HERE ###
        {
            "Sid": "Allow access through Amazon S3 for all principals in the account that are authorized to use Amazon S3",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.region-ID.amazonaws.com",
                    "kms:CallerAccount": "account-ID"
                }
            }
        },
        {
            "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 ###
}
```

API Version 2016-10-06

183
• **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 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, or the AWS SDKs. To install and configure the AWS CLI, see Getting Set Up with the AWS Command Line Interface in the AWS Command Line Interface User Guide.

1. Run the following command to confirm whether your installation of the AWS CLI supports AWS CodeBuild:

   ```bash
   aws codebuild list-builds
   ```

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

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

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

Command Line Reference for AWS CodeBuild

The AWS CLI provides commands for automating AWS CodeBuild. Use the information in this topic as a supplement to the AWS Command Line Interface User Guide and the AWS CLI Reference for AWS CodeBuild.

Not what you're looking for? If you want to use the AWS SDKs to call AWS CodeBuild, see the AWS SDKs and Tools Reference (p. 186).

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. 184).

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

```shell
aws codebuild help
```

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

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

AWS CodeBuild commands include:

- **batch-delete-builds**: Deletes one or more builds in AWS CodeBuild. For more information, see Delete Builds (AWS CLI) (p. 173).
- **batch-get-builds**: Gets information about multiple builds in AWS CodeBuild. For more information, see View Build Details (AWS CLI) (p. 168).
- **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. 155).
- **create-project**: Creates a build project. For more information, see Create a Build Project (AWS CLI) (p. 146).
- **delete-project**: Deletes a build project. For more information, see Delete a Build Project (AWS CLI) (p. 160).
- **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. 170).
- **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. 171).
- **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. 101).
- **list-projects**: Gets a list of build project names. For more information, see View a List of Build Project Names (AWS CLI) (p. 154).
- **start-build**: Starts running a build. For more information, see Run a Build (AWS CLI) (p. 163).
- **stop-build**: Attempts to stop the specified build from running. For more information, see Stop a Build (AWS CLI) (p. 173).
- **update-project**: Changes information about the specified build project. For more information, see Change a Build Project's Settings (AWS CLI) (p. 159).
AWS SDKs and Tools Reference for AWS CodeBuild

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

If you want to use the AWS CLI to run AWS CodeBuild, see the Command Line Reference (p. 185).

Supported AWS SDKs and Tools for AWS CodeBuild

The following AWS SDKs and tools support AWS CodeBuild:

- The AWS SDK for C++. For more information, see the Aws::CodeBuild namespace section of the AWS SDK for C++ API Reference.
- The AWS SDK for Go. For more information, see the codebuild section of the AWS SDK for Go API Reference.
- The AWS SDK for Java. For more information, see the com.amazonaws.services.codebuild and com.amazonaws.services.codebuild.model sections of the AWS SDK for Java API Reference.
- The AWS SDK for JavaScript in the Browser and the AWS SDK for JavaScript in Node.js. For more information, see the Class: AWS.AWS CodeBuild section of the AWS SDK for JavaScript API Reference.
- The AWS SDK for .NET. For more information, see the Amazon.CodeBuild and Amazon.CodeBuild.Model namespace sections of the AWS SDK for .NET API Reference.
- The AWS SDK for PHP. For more information, see the Namespace Aws\CodeBuild section of the AWS SDK for PHP API Reference.
- The AWS SDK for Python (Boto3). For more information, see the CodeBuild section of the Boto 3 Documentation.
- The AWS SDK for Ruby. For more information, see the Module: Aws::CodeBuild section of the AWS SDK for Ruby API Reference.
- The AWS Tools for PowerShell. For more information, see the AWS CodeBuild section of the AWS Tools for PowerShell Cmdlet Reference.

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. 186)
- Access Control (p. 187)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your root credentials and they provide complete access to all of your AWS resources.
Important

For security reasons, we recommend that you use the root credentials only to create an administrator user, which is an IAM user with full permissions to your AWS account. Then, you can use this administrator user to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the IAM User Guide.

- **IAM user** – An IAM user is simply an identity in your AWS account that has custom permissions (for example, permission to create build projects in 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.
  - **Cross-account access** – You can use an IAM role in your account to grant another AWS account permissions to access your account’s resources. For an example, see Tutorial: Delegate Access Across AWS Accounts Using IAM Roles in the IAM User Guide.
  - **AWS service access** – You can use an IAM role in your account to grant permissions to an AWS service to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an 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.
  - **Applications running on Amazon EC2** – Instead of storing access keys in the Amazon EC2 instance for use by applications running on the instance and making AWS API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an Amazon EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the Amazon EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the IAM User Guide.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions, you cannot create or access AWS CodeBuild resources. For example, you must have permissions to create, view, or delete build projects and to start, stop, or view builds.

The following sections describe how to manage permissions for AWS CodeBuild. We recommend that you read the overview first.

- Overview of Managing Access Permissions to Your AWS CodeBuild Resources (p. 188)
- Using Identity-Based Policies (IAM Policies) for AWS CodeBuild (p. 191)
- AWS CodeBuild Permissions Reference (p. 197)
Overview of Managing Access Permissions to Your AWS CodeBuild Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

**Note**
An account administrator (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the IAM User Guide.

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

**Topics**
- AWS CodeBuild Resources and Operations (p. 188)
- Understanding Resource Ownership (p. 189)
- Managing Access to Resources (p. 189)
- Specifying Policy Elements: Actions, Effects, and Principals (p. 190)

### AWS CodeBuild Resources and Operations

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

<table>
<thead>
<tr>
<th>Resource type</th>
<th>ARN format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build project</td>
<td>arn:aws:codebuild:region-ID:account-ID:project/project-name</td>
</tr>
<tr>
<td>All AWS CodeBuild resources</td>
<td>arn:aws:codebuild:*</td>
</tr>
<tr>
<td>All AWS CodeBuild resources owned by the specified account in the specified region</td>
<td>arn:aws:codebuild:region-ID:account-ID:*</td>
</tr>
</tbody>
</table>

**Note**
Most AWS services treat a colon (:) or a forward slash (/) as the same character in ARNs. However, 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-2:123456789012:project/myBuildProject"
To specify all resources, or if an API action does not support ARNs, use the wildcard character (*) in the `Resource` element as follows:

```
"Resource": "*"
```

Some 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-2:123456789012:project/myBuildProject",
  "arn:aws:codebuild:us-east-2:123456789012: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. 197).

**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. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the IAM User Guide.

Policies attached to an IAM identity are referred to as identity-based policies (IAM policies). Policies attached to a resource are referred to as resource-based policies. 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.

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. 188).

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. 188) and AWS CodeBuild
Permissions Reference (p. 197).

The following are the basic policy elements:

- **Resource** – You use an Amazon Resource Name (ARN) to identify the resource that the policy applies
to.
- **Action** – You use action keywords to identify resource operations you want to allow or deny. For
  example, the codebuild:CreateProject permission gives the user permissions to perform the
  CreateProject operation.
- **Effect** – You specify the effect, either allow or deny, when the user requests the action. If you don't
  explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny
  access to a resource. You might do this to make sure a user cannot access a resource, even if a different
  policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user the policy is attached to is the implicit
  principal. For resource-based policies, you specify the user, account, service, or other entity that you
  want to receive permissions.

To learn more about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the IAM
User Guide.

For a table showing all of the AWS CodeBuild API actions and the resources they apply to, see the AWS
CodeBuild Permissions Reference (p. 197).
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. 188).

Topics

- Permissions Required to Use the AWS CodeBuild Console (p. 191)
- Permissions Required for the AWS CodeBuild Console to Connect to Source Providers Using OAuth (p. 192)
- AWS Managed (Predefined) Policies for AWS CodeBuild (p. 192)
- Customer-Managed Policy Examples (p. 192)

The following shows an example of a permissions policy that allows a user to get information about build projects only in the `us-east-2` region for account `123456789012` for any build project that starts with the name `my`:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
    }
  ]
}
```

Permissions Required to Use the AWS CodeBuild Console

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

- AWS CodeBuild
- Amazon CloudWatch
- AWS CodeCommit (if you are storing your source code in an AWS CodeCommit repository)
- Amazon Elastic Container Registry (Amazon ECR) (if you are using a build environment that relies on a Docker image in an Amazon ECR repository)
- Amazon Elastic Container Service (Amazon ECS) (if you are using a build environment that relies on a Docker image in an Amazon ECR repository)
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service (Amazon S3)

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended.
Permissions Required for the AWS CodeBuild Console to Connect to Source Providers Using OAuth

The AWS CodeBuild console uses the following API actions to connect to source providers (for example, GitHub repositories) using OAuth.

- codebuild:ListConnectedOAuthAccounts
- codebuild:ListRepositories
- codebuild:PersistOAuthToken

To associate source providers using OAuth (such as GitHub repositories) with your build projects, you must use the AWS CodeBuild console. To do this, you must first add the preceding API actions to IAM access policies associated with the IAM user you use to access the AWS CodeBuild console.

These API actions are not intended to be called by your code. Therefore, these API actions are not included in the AWS CLI and AWS SDKs.

AWS Managed (Predefined) Policies for AWS CodeBuild

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the IAM User Guide.

The following AWS managed policies, which you can attach to users in your account, are specific to AWS CodeBuild.

- AWSCodeBuildAdminAccess – Provides full access to 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. 191).

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

Topics

- Allow a User to Get Information About Build Projects (p. 193)
• Allow a User to Create Build Projects (p. 193)
• Allow a User to Delete Build Projects (p. 193)
• Allow a User to Get a List of Build Project Names (p. 194)
• Allow a User to Change Information About Build Projects (p. 194)
• Allow a User to Get Information About Builds (p. 194)
• Allow a User to Get a List of Build IDs for a Build Project (p. 195)
• Allow a User to Get a List of Build IDs (p. 195)
• Allow a User to Begin Running Builds (p. 195)
• Allow a User to Attempt to Stop Builds (p. 196)
• Allow a User to Attempt to Delete Builds (p. 196)
• Allow a User to Get Information About Docker Images that Are Managed by AWS CodeBuild (p. 196)
• Allow a User to Create a VPC Network Interface (p. 196)

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-2 region for account 123456789012 for any build project that starts with the name my:

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

Allow a User to Create Build Projects

The following example policy statement allows a user to create build projects with any name but only in the us-east-2 region for account 123456789012 and using only the specified AWS CodeBuild service role:

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

Allow a User to Delete Build Projects

The following example policy statement allows a user to delete build projects only in the us-east-2 region for account 123456789012 for any build project that starts with the name my:
Using Identity-Based Policies (IAM Policies)

Allow a User to Get a List of Build Project Names

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

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

Allow a User to Change Information About Build Projects

The following example policy statement allows a user to change information about build projects with any name but only in the us-east-2 region for account 123456789012 and using only the specified AWS CodeBuild service role:

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

Allow a User to Get Information About Builds

The following example policy statement allows a user to get information about builds only in the us-east-2 region for account 123456789012 for the build projects named my-build-project and my-other-build-project:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
```
"Effect": "Allow",
"Action": "codebuild:BatchGetBuilds",
"Resource": [
  "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",
  "arn:aws:codebuild:us-east-2:123456789012:project/my-other-build-project"
]
]}
}

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

The following example policy statement allows a user to get a list of build IDs only in the us-east-2 region for account 123456789012 for the build projects named my-build-project and my-other-build-project:

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListBuildsForProject",
      "Resource": [
        "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",
        "arn:aws:codebuild:us-east-2:123456789012:project/my-other-build-project"
      ]
    }
  ]
}

Allow a User to Get a List of Build IDs

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

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

Allow a User to Begin Running Builds

The following example policy statement allows a user to run builds only in the us-east-2 region for account 123456789012 for build project that starts with the name my:

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
    }
  ]
}
Allow a User to Attempt to Stop Builds

The following example policy statement allows a user to attempt to stop running builds only in the us-east-2 region for account 123456789012 for any build project that starts with the name my:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    
    "Effect": "Allow",
    "Action": "codebuild:StopBuild",
  ]
}
```

Allow a User to Attempt to Delete Builds

The following example policy statement allows a user to attempt to delete builds only in the us-east-2 region for account 123456789012 for any build project that starts with the name my:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    
    "Effect": "Allow",
    "Action": "codebuild:BatchDeleteBuilds",
  ]
}
```

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:

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

Allow a User to Create a VPC Network Interface

The following example policy statement allows a user to create a network interface in an Amazon VPC:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    
    "Effect": "Allow",
    "Action": "codebuild:*",
  ]
}
```
"Action": [
  "ec2:CreateNetworkInterface",
  "ec2:DescribeDhcpOptions",
  "ec2:DescribeNetworkInterfaces",
  "ec2:DeleteNetworkInterface",
  "ec2:DescribeSubnets",
  "ec2:DescribeSecurityGroups",
  "ec2:DescribeVpcs"
],
"Resource": "*
},

{ "Effect": "Allow",
"Action": [ "ec2:CreateNetworkInterfacePermission"
],
"Resource": "arn:aws:ec2:{{region}}:{{account-id}}:network-interface/**",
"Condition": {
  "StringEquals": {
    "ec2:Subnet": [ "arn:aws:ec2:{{region}}:{{account-id}}:subnet/[[subnets]]"
    ],
    "ec2:AuthorizedService": "codebuild.amazonaws.com"
  }
}
}

AWS CodeBuild Permissions Reference

You can use the following table as a reference when you are setting up Access Control (p. 187) 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-2:123456789012:project/my*

AWS CodeBuild API Operations and Required Permissions for Actions

BatchDeleteBuilds

**Action:** codebuild:BatchDeleteBuilds

Required to delete builds.
**Resource:** arn:aws:codebuild:region-ID:account-ID:project/project-name

**Action:** codebuild:BatchGetBuilds

Required to get information about builds.

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

**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,
ar: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

---

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, and the AWS SDKs. 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 CodeBuild 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. 185). 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:

```json
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "FederatedUser",
        "principalId": "account-ID:user-name",
        "arn": "arn:aws:sts::account-ID:federated-user/user-name",
        "accountId": "account-ID",
        "accessKeyId": "access-key-ID",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2016-09-06T17:59:10Z"
            },
            "sessionIssuer": {
                "type": "IAMUser",
                "principalId": "access-key-ID",
                "arn": "arn:aws:iam::account-ID:user/user-name",
                "accountId": "account-ID",
                "userName": "user-name"
            }
        },
        "eventTime": "2016-09-06T17:59:11Z",
        "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",
                "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"
}
Troubleshooting AWS CodeBuild

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

Topics
- Error: "CodeBuild is not authorized to perform: sts:AssumeRole" When Creating or Updating a Build Project (p. 202)
- Error: "The bucket you are attempting to access must be addressed using the specified endpoint..." When Running a Build (p. 203)
- Error: "Failed to upload artifacts: Invalid Arn" When Running a Build (p. 203)
- Error: "Unable to Locate Credentials" (p. 203)
- Earlier Commands in Build Specs Are Not Recognized by Later Commands (p. 204)
- Apache Maven Builds Reference Artifacts from the Wrong Repository (p. 205)
- Build Commands Run as root by Default (p. 206)
- The Bourne Shell (sh) Must Exist in Build Images (p. 206)
- Error: "AWS CodeBuild is experiencing an issue" When Running a Build (p. 206)
- Error: "BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE" When Using a Custom Build Image (p. 207)
- Builds May Fail When File Names Have Non-US English Characters (p. 207)
- Builds May Fail When Getting Parameters from Amazon EC2 Parameter Store (p. 208)
- "Access denied" Error Message When Attempting to Download Cache (p. 208)
- Error: "Unable to download cache: RequestError: send request failed caused by: x509: failed to load system roots and no roots provided" (p. 209)
- Error: "Unable to download certificate from S3. AccessDenied" (p. 209)
- Error: "Git Clone Failed: unable to access 'your-repository-URL': SSL certificate problem: self signed certificate" (p. 210)
- Error: "The policy's default version was not created by enhanced zero click role creation or was not the most recent version created by enhanced zero click role creation." (p. 210)

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:
Error: "The bucket you are attempting to access must be addressed using the specified endpoint..." When Running a Build

**Issue:** When you run a build, the **DOWNLOAD_SOURCE** build phase fails with the error "The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint."

**Possible cause:** Your pre-built source code is stored in an Amazon S3 bucket, and that bucket is in a different AWS region than the AWS CodeBuild build project.

**Recommended solution:** Update the build project's settings to point to a bucket that contains your pre-built source code, and that bucket is in the same region as the build project.

Error: "Failed to upload artifacts: Invalid arn" When Running a Build

**Issue:** When you run a build, the **UPLOAD_ARTIFACTS** build phase fails with the error "Failed to upload artifacts: Invalid arn".

**Possible cause:** Your Amazon S3 output bucket (the bucket where AWS CodeBuild stores its output from the build) is in a different AWS region than the AWS CodeBuild build project.

**Recommended solution:** Update the build project's settings to point to an output bucket that is in the same region as the build project.

Error: "Unable to Locate Credentials"

**Issue:** When you try to run the AWS CLI, use an AWS SDK, or call another similar component as part of a build, you get build errors that are directly related to the AWS CLI, AWS SDK, or component. For example, you may get a build error such as "unable to locate credentials."

**Possible causes:**

- The version of the AWS CLI, AWS SDK, or component in the build environment is incompatible with AWS CodeBuild.
- You are running a Docker container within a build environment that uses Docker, and that Docker container does not have access to the necessary AWS credentials by default.
Recommended solutions:

- Make sure your build environment has the following version or higher of the AWS CLI, AWS SDK, or component.
  - AWS CLI: 1.10.47
  - AWS SDK for C++: 0.2.19
  - AWS SDK for Go: 1.2.5
  - AWS SDK for Java: 1.11.16
  - AWS SDK for JavaScript: 2.4.7
  - AWS SDK for PHP: 3.18.28
  - AWS SDK for Python (Boto3): 1.4.0
  - AWS SDK for Ruby: 2.3.22
  - Botocore: 1.4.37
  - CoreCLR: 3.2.6-beta
  - Node.js: 2.4.7
- If you need to run a Docker container in a build environment and that container requires AWS credentials, you must pass through the credentials from the build environment to the container. In your build spec, include a Docker `run` command such as the following, which in this example uses the `aws s3 ls` command to list your available Amazon S3 buckets. The `-e` option passes through the necessary environment variables for your container to access AWS credentials.

```
docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI your-image-tag
aws s3 ls
```
- If you are building a Docker image and the build requires AWS credentials (for example, to download a file from Amazon S3), you must pass through the credentials from the build environment to the Docker build process as follows.
  1. In your source code's Dockerfile for the Docker image, specify the following `ARG` instructions.

```
ARG AWS_DEFAULT_REGION
ARG AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```
  2. In your build spec, include a Docker `build` command such as the following. The `--build-arg` options will set the necessary environment variables for your Docker build process to access the AWS credentials.

```
docker build --build-arg AWS_DEFAULT_REGION=$AWS_DEFAULT_REGION --build-arg AWS_CONTAINER_CREDENTIALS_RELATIVE_URI=$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI -t your-image-tag
```

Earlier Commands in Build Specs Are Not Recognized by Later Commands

**Issue:** The results of one or more commands in your build spec are not recognized by later commands in the same build spec. For example, a command might set a local environment variable, but a command run later might fail to get the value of that local environment variable.

**Possible cause:** In build spec version 0.1, AWS CodeBuild runs each command in a separate instance of the default shell in the build environment. This means that each command runs in isolation from all other commands. By default, then, you cannot run a single command that relies on the state of any previous commands.
Recommended solutions: We recommend you use build spec version 0.2, which solves this issue. If you must use build spec version 0.1 for some reason, we recommend using the shell command chaining operator (for example, && in Linux) to combine multiple commands into a single command. Or include a shell script in your source code that contains multiple commands, and then call that shell script from a single command in the build spec. For more information, see Shells and Commands in Build Environments (p. 108) and Environment Variables in Build Environments (p. 109).

Apache Maven Builds Reference Artifacts from the Wrong Repository

Issue: When you use Maven with an AWS CodeBuild provided Java build environment, Maven pulls build and plugin dependencies from the secure central Maven repository at https://repo1.maven.org/maven2. This happens even if your build project's pom.xml file explicitly declares other locations to use instead.

Possible cause: AWS CodeBuild provided Java build environments include a file named settings.xml that is preinstalled in the build environment's /root/.m2 directory. This settings.xml file contains the following declarations, which instruct Maven to always pull build and plugin dependencies from the secure central Maven repository at https://repo1.maven.org/maven2.

```xml
<settings>
  <activeProfiles>
    <activeProfile>securecentral</activeProfile>
  </activeProfiles>
  <profiles>
    <profile>
      <id>securecentral</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
</settings>
```

Recommended solution: Do the following:

1. Add a settings.xml file to your source code.
2. In this settings.xml file, use the preceding settings.xml format as a guide to declare the repositories you want Maven to pull the build and plugin dependencies from instead.
3. In the install phase of your build project, instruct AWS CodeBuild to copy your settings.xml file to the build environment's /root/.m2 directory. For example, consider the following snippet from a buildspec.yml file that demonstrates this behavior.

API Version 2016-10-06
205
Build Commands Run as root by Default

**Issue:** AWS CodeBuild runs your build commands as the root user. This happens even if your related build image's Dockerfile sets the `USER` instruction to a different user.

**Cause:** AWS CodeBuild runs all build commands as the root user by default.

**Recommended solution:** None.

The Bourne Shell (sh) Must Exist in Build Images

**Issue:** You are using a build image that is not provided by AWS CodeBuild, and your builds fail with the message "build container found dead before completing the build."

**Possible cause:** The Bourne shell (`sh`) is not included in your build image. AWS CodeBuild needs `sh` to run build commands and scripts.

**Recommended solution:** If `sh` is not present in your build image, be sure to include it before you start any more builds that use your image. (AWS CodeBuild already includes `sh` in its build images.)

Error: "AWS CodeBuild is experiencing an issue" When Running a Build

**Issue:** When you try to run a build project, you receive the following error during the build's PROVISIONING phase: "AWS CodeBuild is experiencing an issue."

**Possible cause:** Your build is using environment variables that are too large for AWS CodeBuild. AWS CodeBuild can raise errors once the length of all environment variables (all names and values added together) reach a combined maximum of around 5,500 characters.

**Recommended solution:** Use Amazon EC2 Systems Manager Parameter Store to store large environment variables and then retrieve them from your build spec. Amazon EC2 Systems Manager Parameter Store can store an individual environment variable (name and value added together) that is a combined 4,096 characters or less. To store large environment variables, see Systems Manager Parameter Store and Systems Manager Parameter Store Console Walkthrough in the Amazon EC2 Systems Manager User Guide. To retrieve them, see the parameter-store mapping in Build Spec Syntax (p. 94).
Error: "BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE" When Using a Custom Build Image

**Issue:** When you try to run a build that uses a custom build image, the build fails with the error `BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE`.

**Possible causes:**

- The build image's overall uncompressed size is larger than the build environment compute type's available disk space. To check your build image's size, use Docker to run the `docker images REPOSITORY:TAG` command. For a list of available disk space by compute type, see Build Environment Compute Types (p. 107).

- AWS CodeBuild does not have permission to pull the build image from your Amazon Elastic Container Registry (Amazon ECR).

**Recommended solutions:**

- Use a larger compute type with more available disk space, or reduce the size of your custom build image.

- Update the permissions in your repository in Amazon ECR so that AWS CodeBuild can pull your custom build image into the build environment. For more information, see the Amazon ECR Sample (p. 22).

Builds May Fail When File Names Have Non-US English Characters

**Issue:** When you run a build that uses files with file names containing non-US English characters (for example, Chinese characters), the build fails.

**Possible cause:** Build environments provided by AWS CodeBuild have their default locale set to POSIX. POSIX localization settings are less compatible with AWS CodeBuild and file names that contain non-US English characters and can cause related builds to fail.

**Recommended solution:** Add the following commands to the `pre_build` section of your build specification. These commands make the build environment use US English UTF-8 for its localization settings, which is more compatible with AWS CodeBuild and file names that contain non-US English characters.

For build environments based on Ubuntu:

```sh
pre_build:
  commands:
    - export LC_ALL="en_US.UTF-8"
    - locale-gen en_US en_US.UTF-8
    - dpkg-reconfigure locales
```

For build environments based on Amazon Linux:

```sh
pre_build:
  commands:
```
Builds May Fail When Getting Parameters from Amazon EC2 Parameter Store

**Issue:** When a build tries to get the value of one or more parameters stored in Amazon EC2 Parameter Store, the build fails in the **DOWNLOAD_SOURCE** phase with the following error: "Parameter does not exist."

**Possible causes:** The service role the build project relies on does not have permission to call the `ssm:GetParameters` action or the build project uses a service role that is generated by AWS CodeBuild and allows calling the `ssm:GetParameters` action, but the parameters have names that do not start with `/CodeBuild/`.

**Recommended solutions:**

- If the service role was not generated by AWS CodeBuild, update its definition to allow AWS CodeBuild to call the `ssm:GetParameters` action. For example, the following policy statement allows calling the `ssm:GetParameters` action to get parameters with names starting with `/CodeBuild/`:

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Action": "ssm:GetParameters",
           "Effect": "Allow",
       }
   ]
}
```

- If the service role was generated by AWS CodeBuild, update its definition to allow AWS CodeBuild to access parameters in Amazon EC2 Parameter Store with names other than those starting with `/CodeBuild/`. For example, the following policy statement allows calling the `ssm:GetParameters` action to get parameters with the specified name:

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Action": "ssm:GetParameters",
           "Effect": "Allow",
       }
   ]
}
```

"Access denied" Error Message When Attempting to Download Cache

**Issue:** When attempting to download the cache on a build project that has cache enabled, you receive the following generic error: "Access denied".
Possible causes:

- You have just configured caching as part of your build project.
- The cache has recently been invalidated via the `InvalidateProjectCache` API.
- The service role being used by CodeBuild does not have `s3:GetObject` and `s3:PutObject` permissions to the Amazon S3 bucket that is holding the cache.

Recommended solutions: For first time use, it's normal to see this immediately after updating the cache configuration. If this error persists, then you should check to see if your service role has `s3:GetObject` and `s3:PutObject` permissions to the Amazon S3 bucket that is holding the cache. For more information, see Specifying S3 permissions.

Error: "Unable to download cache: RequestError: send request failed caused by: x509: failed to load system roots and no roots provided"

Issue: When you try to run a build project, the build fails with the error: "RequestError: send request failed caused by: x509: failed to load system roots and no roots provided."

Possible causes:

- You have configured caching as part of your build project and are using an older Docker image that includes an expired root certificate.

Recommended solutions:

- Update the Docker image that is being used in your AWS CodeBuild the project. For more information, see Docker Images Provided by AWS CodeBuild (p. 101).

Error: "Unable to download certificate from S3. AccessDenied"

Issue: When you try to run a build project, the build fails with the error "Unable to download certificate from S3. AccessDenied."

Possible causes:

- You have chosen the wrong S3 bucket for your certificate.
- You have entered the wrong object key for your certificate.

Recommended solutions:

- Edit your project. For **Bucket of certificate**, choose the S3 bucket where your SSL certificate is stored.
- Edit your project. For **Object key of certificate**, type the name of your S3 object key.
Error: "Git Clone Failed: unable to access 'your-repository-URL': SSL certificate problem: self signed certificate"

**Issue:** When you try to run a build project, the build fails with the error "Git Clone Failed: unable to access 'your-repository-URL': SSL certificate problem: self signed certificate."

**Possible causes:**
- Your source repository has a self-signed certificate, but you have not chosen to install the certificate from your S3 bucket as part of your build project.

**Recommended solutions:**
- Edit your project. For **Certificate**, choose **Install certificate from S3**. For **Bucket of certificate**, choose the S3 bucket where your SSL certificate is stored. For **Object key of certificate**, type the name of your S3 object key.
- Edit your project. Select **Insecure SSL** to ignore SSL warnings while connecting to your GitHub Enterprise project repository.

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

Error: "The policy's default version was not created by enhanced zero click role creation or was not the most recent version created by enhanced zero click role creation."

**Issue:** When you try to update a project in the console, the update failed with the error: "The policy's default version was not created by enhanced zero click role creation or was not the most recent version created by enhanced zero click role creation."

**Possible causes:**
- You have manually updated the policies attached to the target AWS CodeBuild service role.
- You have selected a previous version of a policy attached to the target AWS CodeBuild service role.

**Recommended solutions:**
- Edit your AWS CodeBuild project, and deselect **Allow AWS CodeBuild to modify this service role so it can be used with this build project**. Manually update the target AWS CodeBuild service role to have sufficient permissions. For more information, see Create an AWS CodeBuild Service Role (p. 179).
- Edit your AWS CodeBuild project, and select **Create a role**.
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

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of build projects</td>
<td>1,000</td>
</tr>
<tr>
<td>Length of a build project name</td>
<td>2 to 255 characters, inclusive</td>
</tr>
<tr>
<td>Allowed characters in a build project name</td>
<td>The letters A–Z and a–z, the numbers 0–9, and the special characters – and _</td>
</tr>
<tr>
<td>Maximum length of a build project description</td>
<td>255 characters</td>
</tr>
<tr>
<td>Allowed characters in a build project description</td>
<td>Any</td>
</tr>
<tr>
<td>Maximum number of build projects you can request information about at any one time by using the AWS CLI, or AWS SDKs</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of tags you can associate with a build project</td>
<td>50</td>
</tr>
<tr>
<td>Number of minutes you can specify in a build project for the build timeout of all related builds</td>
<td>5 to 480 (8 hours)</td>
</tr>
<tr>
<td>Number of subnets you can add under VPC configuration</td>
<td>1 to 16</td>
</tr>
<tr>
<td>Number of security groups you can add under VPC configuration</td>
<td>1 to 5</td>
</tr>
</tbody>
</table>

Builds

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of concurrent running builds *</td>
<td>20</td>
</tr>
<tr>
<td>Maximum number of builds you can request information about at any one time by using the AWS CLI, AWS SDKs</td>
<td>100</td>
</tr>
<tr>
<td>Number of minutes you can specify for the build timeout of a single build</td>
<td>5 to 480 (8 hours)</td>
</tr>
</tbody>
</table>
* Limits for the maximum number of concurrent running builds vary, depending on the compute type. For some compute types, the default is 20. To request a higher concurrent build limit or if you get a "Cannot have more than X active builds for the account" error, contact AWS support.
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:** January 25, 2018

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GitHub Enterprise support</td>
<td>AWS CodeBuild can now build from source code stored in a GitHub Enterprise repository. For more information, see [GitHub Enterprise Sample](p. 30).</td>
<td>January, 25, 2018</td>
</tr>
<tr>
<td>Git clone depth support</td>
<td>AWS CodeBuild now supports the creation of a shallow clone with a history truncated to the specified number of commits. For more information, see [Create a Build Project](p. 141).</td>
<td>January, 25, 2018</td>
</tr>
<tr>
<td>VPC support</td>
<td>VPC-enabled builds are now able to access resources inside your VPC. For more information, see [VPC Support](p. 112).</td>
<td>November, 27, 2017</td>
</tr>
<tr>
<td>Dependency caching support</td>
<td>AWS CodeBuild now supports the dependency caching. This allows AWS CodeBuild to save certain reusable pieces of the build environment in the cache and use this across builds.</td>
<td>November, 27, 2017</td>
</tr>
<tr>
<td>Build badges support</td>
<td>AWS CodeBuild now supports the use of build badges, which provide an embeddable, dynamically generated image (badge) that displays the status of the latest build for a project. For more information, see [Build Badges Sample](p. 39).</td>
<td>November 27, 2017</td>
</tr>
<tr>
<td>AWS Config integration</td>
<td>AWS Config now supports AWS CodeBuild as an AWS resource, which means the service can track your AWS CodeBuild projects. For more information about AWS Config, see [Use AWS Config with AWS CodeBuild Sample](p. 37).</td>
<td>October 20, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Automatically rebuild updated source code in GitHub repositories</td>
<td>If your source code is stored in a GitHub repository, you can enable AWS CodeBuild to rebuild your source code whenever a code change is pushed to the repository. For more information, see GitHub Pull Request Sample (p. 35).</td>
<td>September 21, 2017</td>
</tr>
<tr>
<td>New ways for storing and retrieving sensitive or large environment variables in Amazon EC2 Systems Manager Parameter Store</td>
<td>You can now use the AWS CodeBuild console or the AWS CLI to retrieve sensitive or large environment variables stored in Amazon EC2 Systems Manager Parameter Store. You can also now use the AWS CodeBuild console to store these types of environment variables in Amazon EC2 Systems Manager Parameter Store. Previously, you could only retrieve these types of environment variables by including them in a build spec or by running build commands to automate the AWS CLI. You could only store these types of environment variables by using the Amazon EC2 Systems Manager Parameter Store console. For more information, see Create a Build Project (p. 141), Change a Build Project's Settings (p. 156), and Run a Build (p. 161).</td>
<td>September 14, 2017</td>
</tr>
<tr>
<td>Build deletion support</td>
<td>You can now delete builds in AWS CodeBuild. For more information, see Delete Builds (p. 173).</td>
<td>August 31, 2017</td>
</tr>
<tr>
<td>Updated way to retrieve sensitive or large environment variables stored in Amazon EC2 Systems Manager Parameter Store by using a build spec</td>
<td>AWS CodeBuild now makes it easier to use a build spec to retrieve sensitive or large environment variables stored in Amazon EC2 Systems Manager Parameter Store. Previously, you could only retrieve these types of environment variables by running build commands to automate the AWS CLI. For more information, see the parameter-store mapping in Build Spec Syntax (p. 94).</td>
<td>August 10, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>AWS CodeBuild supports Bitbucket</td>
<td>AWS CodeBuild can now build from source code stored in a Bitbucket repository. For more information, see Create a Build Project (p. 141) and Run a Build (p. 161).</td>
<td>August 10, 2017</td>
</tr>
<tr>
<td>AWS CodeBuild available in US West (N. California), EU (London), and Canada (Central)</td>
<td>AWS CodeBuild is now available in the US West (N. California), EU (London), and Canada (Central) regions. For more information, see the AWS CodeBuild section of the &quot;AWS Regions and Endpoints&quot; topic in the Amazon Web Services General Reference.</td>
<td>June 29, 2017</td>
</tr>
<tr>
<td>.NET Core in Linux sample</td>
<td>A sample showing how to use AWS CodeBuild and .NET Core to build an executable file out of code written in C# has been added. For more information, see the .NET Core in Linux Sample (p. 89).</td>
<td>June 29, 2017</td>
</tr>
<tr>
<td>Alternate build spec file names and locations supported</td>
<td>You can now specify an alternate file name or location of a build spec file to use for a build project, instead of a default build spec file named buildspec.yml at the root of the source code. For more information, see Build Spec File Name and Storage Location (p. 93).</td>
<td>June 27, 2017</td>
</tr>
<tr>
<td>Updated build notifications sample</td>
<td>AWS CodeBuild now provides built-in support for build notifications through Amazon CloudWatch Events and Amazon Simple Notification Service (Amazon SNS). The previous Build Notifications Sample (p. 41) has been updated to demonstrate this new behavior.</td>
<td>June 22, 2017</td>
</tr>
<tr>
<td>Docker in custom image sample added</td>
<td>A sample showing how to use AWS CodeBuild and a custom Docker build image to build and run a Docker image has been added. For more information, see the Docker in Custom Image Sample (p. 58).</td>
<td>June 7, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Fetch source code for GitHub pull requests</td>
<td>When you run a build with AWS CodeBuild that relies on source code stored in a GitHub repository, you can now specify a GitHub pull request ID to build. You can also specify a commit ID, a branch name, or a tag name instead. For more information, see the Source version value in Run a Build (Console) (p. 161) or the sourceVersion value in Run a Build (AWS CLI) (p. 163).</td>
<td>June 6, 2017</td>
</tr>
<tr>
<td>Build specification version updated</td>
<td>A new version of the build spec format has been released. Version 0.2 addresses the issue of AWS CodeBuild running each build command in a separate instance of the default shell. Also in version 0.2, environment_variables is renamed to env, and plaintext is renamed to variables. For more information, see Build Specification Reference for AWS CodeBuild (p. 93).</td>
<td>May 9, 2017</td>
</tr>
<tr>
<td>Dockerfiles for build images available in GitHub</td>
<td>Definitions for many of the build images provided by AWS CodeBuild are available as Dockerfiles in GitHub. For more information, see the &quot;Definition&quot; column of the table in Docker Images Provided by AWS CodeBuild (p. 101).</td>
<td>May 2, 2017</td>
</tr>
<tr>
<td>AWS CodeBuild available in EU (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Sydney), and Asia Pacific (Tokyo)</td>
<td>AWS CodeBuild is now available in the EU (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Sydney), and Asia Pacific (Tokyo) regions. For more information, see the AWS CodeBuild section of the &quot;AWS Regions and Endpoints&quot; topic in the Amazon Web Services General Reference.</td>
<td>March 21, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>AWS CodePipeline test action support for AWS CodeBuild</td>
<td>You can now add to a pipeline in AWS CodePipeline a test action that uses AWS CodeBuild. For more information, see Add an AWS CodeBuild Test Action to a Pipeline (AWS CodePipeline Console) (p. 136).</td>
<td>March 8, 2017</td>
</tr>
<tr>
<td>Build specs support fetching build output from within selected top-level directories</td>
<td>Build specs now enable you to specify individual top-level directories whose contents you can instruct AWS CodeBuild to include in build output artifacts. You do this by using the base-directory mapping. For more information, see Build Spec Syntax (p. 94).</td>
<td>February 8, 2017</td>
</tr>
<tr>
<td>Built-in environment variables</td>
<td>AWS CodeBuild provides additional built-in environment variables for your builds to use. These include environment variables describing the entity that started the build, the URL to the source code repository, the source code's version ID, and more. For more information, see Environment Variables in Build Environments (p. 109).</td>
<td>January 30, 2017</td>
</tr>
<tr>
<td>AWS CodeBuild available in US East (Ohio)</td>
<td>AWS CodeBuild is now available in the US East (Ohio) region. For more information, see the AWS CodeBuild section of the &quot;AWS Regions and Endpoints&quot; topic in the Amazon Web Services General Reference.</td>
<td>January 19, 2017</td>
</tr>
<tr>
<td>AWS Lambda sample</td>
<td>A reference was added to a sample showing how to use AWS CodeBuild along with Lambda, AWS CloudFormation, and AWS CodePipeline to build and deploy a serverless application that follows the AWS Serverless Application Model (AWS SAM) standard. For more information, see the AWS Lambda Sample (p. 63).</td>
<td>December 20, 2016</td>
</tr>
<tr>
<td>C++ and Go samples</td>
<td>Samples showing how to use AWS CodeBuild to build C++ and Go output artifacts have been added. For more information, see the C++ Sample (p. 71) and the Go Sample (p. 73).</td>
<td>December 9, 2016</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Shell and command behaviors info</td>
<td>AWS CodeBuild runs each command you specify in a separate instance of a build environment's default shell. This default behavior can produce some unexpected side effects for your commands. We recommend some approaches to work around this default behavior if needed. For more information, see Shells and Commands in Build Environments (p. 108).</td>
<td>December 9, 2016</td>
</tr>
<tr>
<td>Environment variables info</td>
<td>AWS CodeBuild provides several environment variables that you can use in your build commands. You can also define your own environment variables. For more information, see Environment Variables in Build Environments (p. 109).</td>
<td>December 7, 2016</td>
</tr>
<tr>
<td>Troubleshooting topic</td>
<td>Troubleshooting information is now available. For more information, see Troubleshooting Your VPC Setup (p. 114).</td>
<td>December 5, 2016</td>
</tr>
<tr>
<td>Jenkins plugin initial release</td>
<td>This is the initial release of the AWS CodeBuild Jenkins Plugin. For more information, see Use AWS CodeBuild with Jenkins (p. 140).</td>
<td>December 5, 2016</td>
</tr>
<tr>
<td>User Guide initial release</td>
<td>This is the initial release of the AWS CodeBuild User Guide.</td>
<td>December 1, 2016</td>
</tr>
</tbody>
</table>
AWS Glossary

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