

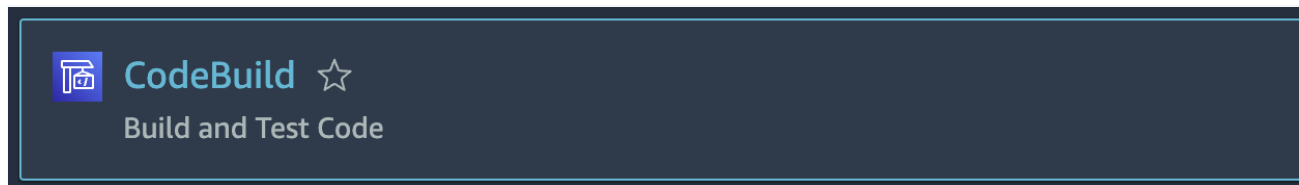
Introduction

AWS CodeBuild is a fully-managed build service. CodeBuild can compile code, run tests, and produce packages that you can deploy on any compatible resource. Using CodeBuild in this lab eliminates the need for you to create a build server or install any software on your local machine.

In this lab step, you will use CodeBuild to build the Docker images containing the two applications you need to complete the lab.

Instructions

1. In the search bar at the top, enter *CodeBuild*, and under **Services**, click the **CodeBuild** result:



The **Build projects** page of AWS Codebuild will load, and you will see one project listed named **ecslab-green**. This project builds a docker image from a zip file stored in an Amazon S3 bucket.

The green CodeBuild project has been created for you as a convenience. You will create a second project named **ecslab-blue-project** that is identical to the green project, except that it uses a different zip file. Both projects will produce Docker images, tagged **testblue** and **testgreen** respectively.

A blue/green deployment strategy is a method of updating an existing application by deploying a new version and switching over to it once the new deployment is ready. This approach allows you to perform functional testing of the new deployment to ensure the new version is working correctly. It also allows for easy rollbacks to the previous version.

2. To begin creating the blue CodeBuild project, click **Create build project**:

A rectangular button with an orange background and white text that reads "Create build project".

3. In the **Create build project** form, configure the following details, leaving the defaults for options not specified:

- **Project configuration:**
 - **Project name:** Enter *ecslab-blue-project*
- **Source:**
 - **Source provider:** Select **Amazon S3**
 - **Bucket:** Select bucket starting with **cloudacademylabs-caecslab-**
 - **S3 object key:** Enter *ecslabblue.zip*
- **Environment:**
 - **Environment image:** Ensure **Managed image** is selected
 - **Operating system:** Select **Ubuntu**
 - **Runtime:** Select **Standard**
 - **Image:** Select **aws/codebuild/standard:6.0**
 - **Image version:** Select **Always use the latest image for this runtime version**
 - **Privileged:** Checked
 - **Service role:** Select **Existing service role**
 - **Role ARN:** Select the **CodeBuildServiceRole** role
 - **Allow AWS CodeBuild to modify this service role...:** Unchecked
- **Buildspec:**
 - **Build specifications:** Ensure **Use a buildspec file** is selected
 - **Buildspec name:** Enter *buildspec.yml*
- **Artifacts:**
 - **Type:** Ensure **No artifacts** is selected

Project configuration

Project name

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Source

Source 1 - Primary

Source provider



Bucket



S3 object key or S3 folder

Source version - *optional* [Info](#)

Enter the version ID of the object that represents the build input ZIP file.

Environment


Environment image

☒ **Managed image**
Use an image managed by AWS CodeBuild

☐ **Custom image**
Specify a Docker image

Operating system

Ubuntu ▼

 The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See [Docker Images Provided by CodeBuild for details](#).

Runtime(s)

Standard ▼

Image

aws/codebuild/standard:6.0 ▼

Image version

Always use the latest image for this runtime version ▼

Environment type

Linux ▼

Privileged

☒ Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Service role

☐ **New service role**
Create a service role in your account

☒ **Existing service role**
Choose an existing service role from your account

Buildspec

Build specifications

☒ **Use a buildspec file**
Store build commands in a YAML-formatted buildspec file

☐ **Insert build commands**
Store build commands as build project configuration

Buildspec name - *optional*

By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

Batch configuration

You can run a group of builds as a single execution. Batch configuration is also available in advanced option when starting build.

☐ **Define batch configuration - *optional***
You can also define or override batch configuration when starting a build batch.

Artifacts

Add artifact

Artifact 1 - Primary

Type



You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

Warning: Ensure you have checked the **Privileged** checkbox before proceeding. If this is unchecked, the docker client will be unable to access the host's docker instance, and building the image will fail.

The project source files and Amazon S3 bucket are created for you by the Cloud Academy lab environment. The project uses Ubuntu Linux with Docker installed as the build environment to create the image. The build spec is a collection of build commands and settings that provide instructions on how to build the Docker image and where to push it once complete.

For your reference, if you would like to inspect the application code fetched from the Amazon S3 bucket, you can find a copy of the blue version of the application in the [Cloud Academy escdemo Github repository](#). This is a simple NodeJS application that serves a JSON message via HTTP.

For the purposes of differentiating the images and the applications they contain in this lab, the build spec also tags the images with distinct names. CodeBuild will tag your blue application image with **testblue** and your green application image with **testgreen** due to instructions in the source files.

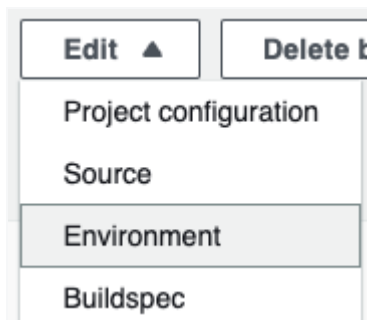
Be aware that AWS CodeBuild supports other sources than Amazon S3, including Git repositories provided by AWS CodeCommit, GitHub, and BitBucket.

4. To create your code build project, click **Create build project**:

A rectangular button with a solid orange background and white text that reads "Create build project". The button has a slight shadow effect.

You will see a green notification at the top of the page letting you know the project was created successfully.

5. To modify your project's environment, click on **Edit > Environment**:



6. Uncheck **Allows AWS CodeBuild to modify this service role...** if it has been selected:

Environment

Current environment image
aws/codebuild/standard:6.0

Override image

Service role
Choose an existing service role from your account

☐ Allow AWS CodeBuild to modify this service role so it can be used with this build project

AWS CodeBuild checks this option by default whenever you create a project or edit an existing project's properties. In this lab, due to security and limited permissions, it must be unchecked.

7. Expand **Additional configuration**, scroll down to **Environment variables**, and enter the following:

- **Name:** `AWS_ACCOUNT_ID`
- **Value:** `120530368474` (the student AWS Account ID)
- **Type:** Ensure **Plaintext** is selected

Environment variables		
Name	Value	Type
<input type="text" value="AWS_ACCOUNT_ID"/>	<input type="text" value="475614092062"/>	<input type="text" value="Plaintext"/> ▼

The Account ID is used to construct the ECR repository URI during the build process. The AWS region is also required to construct the URI. However, CodeBuild includes an [AWS_REGION environment variable in build environment by default](#).

8. At the bottom of the page, click **Update environment**:

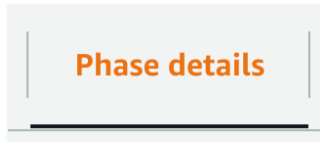
Update environment

9. To start building the blue project, click **Start build**:

Start build

The build information page will load. Once CodeBuild finishes building the Docker image, it executes a Docker push command that stores the image in the ECR repository. This behavior is defined in the `buildspec.yml` file in the deployment zip file.

10. To observe the build, click the **Phase Details** tab:



Observe the **Phase Details** section as each step of the build process proceeds and completes. The build will take up to two minutes to reach the file **COMPLETED** phase.

CodeBuild is building a Docker container image containing one of the custom applications (blue) made for this lab:

Build logs	Phase details	Environment variables	Build details
Name	Status	Context	Duration
SUBMITTED	✔ Succeeded	-	<1 sec
QUEUED	✔ Succeeded	-	1 sec
PROVISIONING	✔ Succeeded	-	19 secs
DOWNLOAD_SOURCE	✔ Succeeded	-	<1 sec
INSTALL	✔ Succeeded	-	<1 sec
PRE_BUILD	✔ Succeeded	-	4 secs
BUILD	✔ Succeeded	-	13 secs
POST_BUILD	✔ Succeeded	-	7 secs
UPLOAD_ARTIFACTS	✔ Succeeded	-	<1 sec
FINALIZING	✔ Succeeded	-	2 secs
COMPLETED	✔ Succeeded	-	-

11. Move to the **Build logs** section to view information about the build process.

Build logs

This screen shows up to the last 10000 lines of the build log. The lines in the following image show part of the POST_BUILD stage where the layers of the image are pushed to the repository before finalizing the build:

```
148 The push refers to repository [124643851959.dkr.ecr.us-west-2.amazonaws.com/ca-container-registry]
149 166c7748c6cd: Preparing
150 4c0555044d30: Preparing
151 9992c1a300f8: Preparing
152 689391dc1c39: Preparing
153 9ac73e86770e: Preparing
154 49b4d76320d3: Preparing
155 5e17a81d7b97: Preparing
156 0adf834b7dd3: Preparing
157 797894c6233f: Preparing
158 1b3ee35aacca: Preparing
159 0adf834b7dd3: Waiting
160 797894c6233f: Waiting
161 1b3ee35aacca: Waiting
162 49b4d76320d3: Waiting
163 5e17a81d7b97: Waiting
164 4c0555044d30: Pushed
165 9992c1a300f8: Pushed
166 166c7748c6cd: Pushed
167 9ac73e86770e: Pushed
168 49b4d76320d3: Pushed
169 689391dc1c39: Pushed
170 5e17a81d7b97: Pushed
171 0adf834b7dd3: Pushed
172 1b3ee35aacca: Pushed
173 797894c6233f: Pushed
174 testblue: digest: sha256:9ebc2fbc3dce8f09b6455e15c2d1625be9dfc5e449f9de2e569bf6c57f00b7da size: 2408
```

12. Scroll up to the top of the build logs and click **View entire log** to go to CloudWatch logs and view detailed information about every step in the build process:

Showing the last 176 lines of the build log. [View entire log](#)

CloudWatch Logs will open in a new browser tab and you will see the build's log entries. Examining and retaining these logs can be helpful when troubleshooting.

13. Return to your AWS CodeBuild browser tab, and go to the build projects page by clicking **Build projects** in the breadcrumb navigation at the top:


> [Build projects](#)

You will now see your **ecslab-blue-project** listed alongside the existing **ecslab-green-project**.

14. To navigate to the project page for the green project, click **ecslab-green-project** in the list:

[ecslab-green-project](#)

Optional: Feel free to click on the **Build details** tab and briefly observe the configuration of the green project. The only configuration difference between it and your blue project is the zip file used as the source from the Amazon S3 bucket. For the green project, the source file is named **ecslabgreen.zip**:



Repository	Source version
cloudacademylabs-caecslab-3u8zoy7shr8j/ecslabgreen.zip 	cloudacademylabs-caecslab-3u8zoy7shr8j/ecslabgreen.zip

15. To begin building the green project, in the top-right, click **Start build**:



The green project will start building, it will take a couple of minutes to complete and upload the Docker image built from the ecslabgreen.zip file. Feel free to click on the **Phase details** to observe its progress.

16. Return to the **Build projects** page and observe that the **Last build status** for both blue and green projects reports **Succeeded**:

Latest build status
 Succeeded
 Succeeded

Note: If one of the projects has the **Latest build status** of **In progress**, wait thirty seconds and click the refresh icon until the status changes:



You have built two Docker images, a blue and green version of the same application.

Be aware that whilst you have manually triggered builds for the projects using the **Start build** button, you are doing so in this lab for the purposes of learning. Builds can be triggered automatically when changes to the source are detected. In a non-lab environment, triggering a build when a Git repository is updated is commonly how new builds are scheduled.

17. [Navigate to the Amazon Elastic Container Registry](#), and click on the repository named **ca-container-registry**:

Repository name
ca-container-registry

You will see two images listed, named **testgreen** and **testblue**:

Image tag
testgreen
testblue

The application buildspec.yml files tag the container images with **testgreen** or **testblue**, depending on the version of the application they contain.

Note: If you do not see two images, it's likely the green project hasn't finished building. Wait a minute or so and click the refresh icon:



18. Copy the URI of each image somewhere you can easily retrieve it later.

You will use these URIs later in the lab.

Cloud Academy recommends opening a draft email and using it to store notes temporarily for the duration of the lab. Alternatively, you can leave this page open and fetch the URIs using the **Copy URI** icon when you need to.

Summary

In this lab step, you used CodeBuild to create two Docker container images with two different applications. You added an environment variable and examined the build phases and logs. You also viewed your Docker images and details in your ECR repository.

VALIDATION CHECKS

1 Checks

Check again 



Created a CodeBuild Project

Check if the AWS CodeBuild project has been created

AWS CodeBuild