**Overview**

In this lab you will build a Docker container image from provided code and a Dockerfile using Cloud Build. You will then upload the container to Container Registry.

**Objectives**

In this lab, you learn how to perform the following tasks:

* Use Cloud Build to build and push containers
* Use Container Registry to store and deploy containers

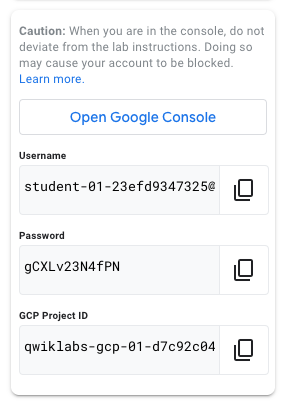
**Task 0. Lab Setup**

**Access Qwiklabs**

For each lab, you get a new GCP project and set of resources for a fixed time at no cost.

1. Make sure you signed into Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, img/time.png and make sure you can finish in that time block.

There is no pause feature. You can restart if needed, but you have to start at the beginning.

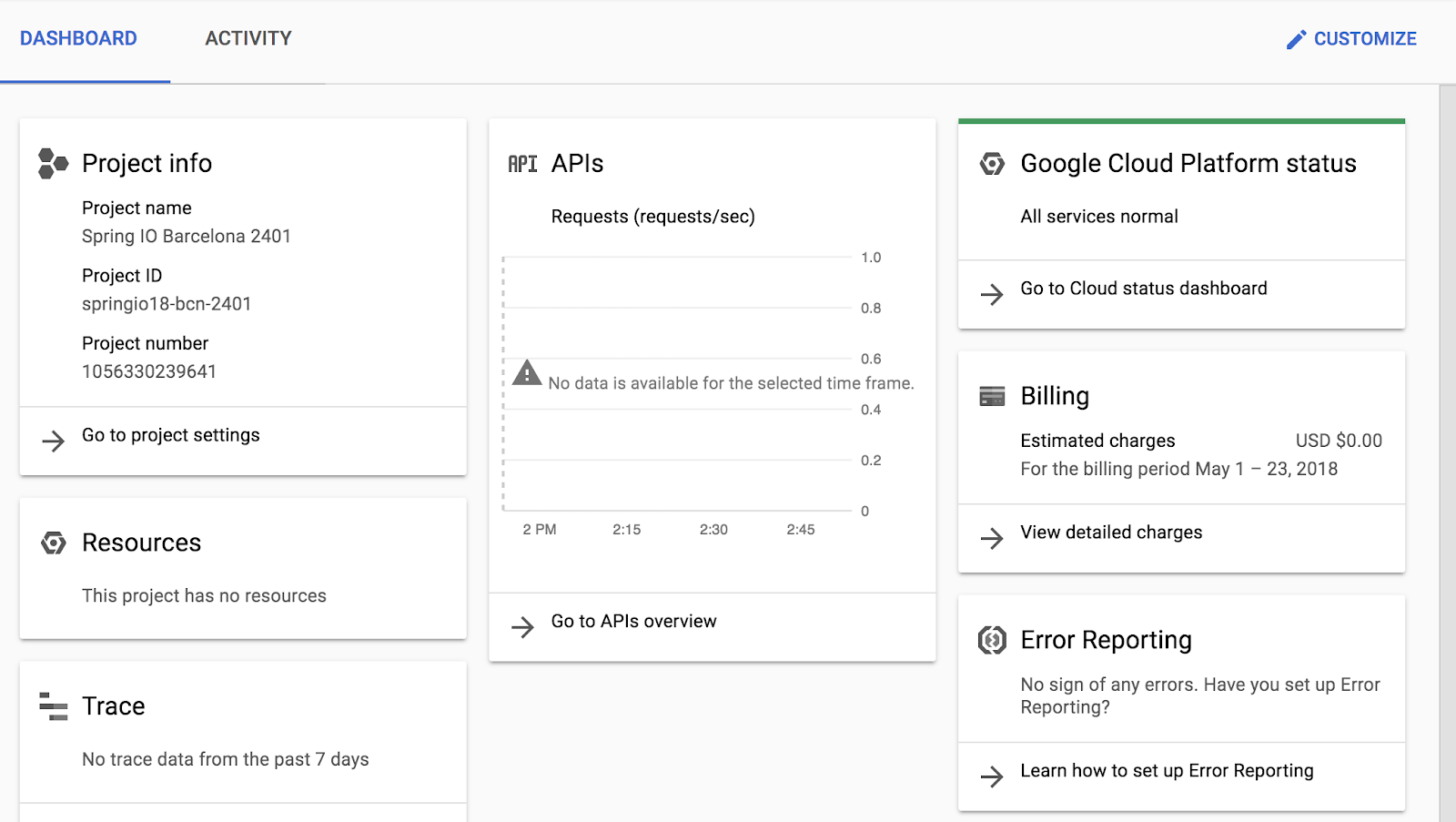
1. When ready, click img/start_lab.png.
2. Note your lab credentials. You will use them to sign in to Cloud Platform Console. 
3. Click **Open Google Console**.
4. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.

If you use other credentials, you'll get errors or **incur charges**.

1. Accept the terms and skip the recovery resource page.

Do not click **End Lab** unless you are finished with the lab or want to restart it. This clears your work and removes the project.

After you complete the initial sign-in steps, the project dashboard appears.



**Task 1: Confirm that needed APIs are enabled**

1. Make a note of the name of your Google Cloud project. This value is shown in the top bar of the Google Cloud Console. It will be of the form qwiklabs-gcp- followed by hexadecimal numbers.
2. In the Google Cloud Console, on the **Navigation menu**(Navigation menu), click **APIs & Services**.
3. Click **Enable APIs and Services**.
4. In the **Search for APIs & Services** box, enter Cloud Build.
5. In the resulting card for the Cloud Build API, if you do not see a message confirming that the Cloud Build API is enabled, click the ENABLE button.
6. Use the Back button to return to the previous screen with a search box. In the search box, enter Container Registry.
7. In the resulting card for the Container Registry API, if you do not see a message confirming that the Container Registry API is enabled, click the ENABLE button.

**Task 2. Building Containers with DockerFile and Cloud Build**

You can write build configuration files to provide instructions to Cloud Build as to which tasks to perform when building a container. These build files can fetch dependencies, run unit tests, analyses and more. In this task, you'll create a DockerFile and use it as a build configuration script with Cloud Build. You will also create a simple shell script (quickstart.sh) which will represent an application inside the container.

1. On the Google Cloud Console title bar, click **Activate Cloud Shell**.
2. When prompted, click **Continue**.

Cloud Shell opens at the bottom of the Google Cloud Console window.

1. Create an empty quickstart.sh file using the nano text editor.

nano quickstart.sh

Copied!

content\_copy

1. Add the following lines in to the quickstart.sh file:

#!/bin/sh

echo "Hello, world! The time is $(date)."

Copied!

content\_copy

1. Save the file and close nano by pressing the **CTRL+X** key, then press **Y** and **Enter**.
2. Create an empty Dockerfile file using the nano text editor.

nano Dockerfile

Copied!

content\_copy

1. Add the following Dockerfile command:

FROM alpine

Copied!

content\_copy

This instructs the build to use the Alpine Linux base image.

1. Add the following Dockerfile command to the end of the Dockerfile:

COPY quickstart.sh /

Copied!

content\_copy

This adds the quickstart.sh script to the / directory in the image.

1. Add the following Dockerfile command to the end of the Dockerfile:

CMD ["/quickstart.sh"]

Copied!

content\_copy

This configures the image to execute the /quickstart.sh script when the associated container is created and run.

The Dockerfile should now look like:

FROM alpine

COPY quickstart.sh /

CMD ["/quickstart.sh"]

Copied!

content\_copy

1. Save the file and close nano by pressing the **CTRL+X** key, then press **Y** and **Enter**.
2. In Cloud Shell, run the following command to make the quickstart.sh script executable.

chmod +x quickstart.sh

Copied!

content\_copy

1. In Cloud Shell, run the following command to build the Docker container image in Cloud Build.

gcloud builds submit --tag gcr.io/${GOOGLE\_CLOUD\_PROJECT}/quickstart-image .

Copied!

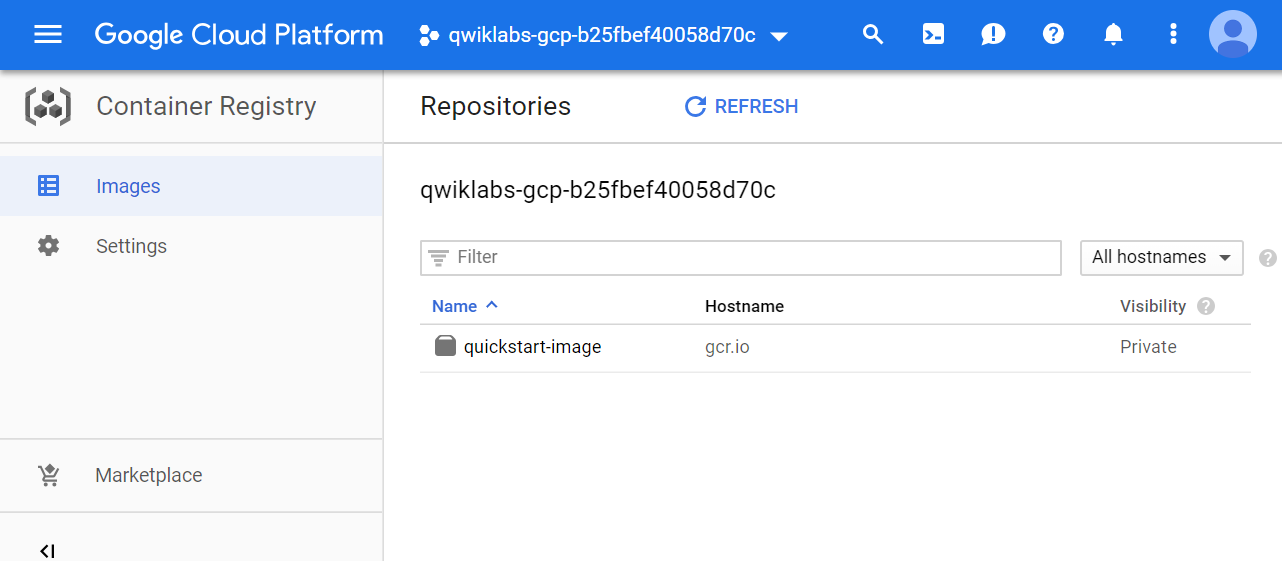
content\_copy

**Important**

Don't miss the dot (".") at the end of the command. The dot specifies that the source code is in the current working directory at build time.

When the build completes, your Docker image is built and pushed to Container Registry.

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu), click **Container Registry** > **Images**.



The quickstart-image Docker image appears in the list

**Task 3. Building Containers with a build configuration file and Cloud Build**

Cloud Build also supports custom build configuration files. In this task you will incorporate an existing Docker container using a custom YAML-formatted build file with Cloud Build.

1. In Cloud Shell enter the following command to clone the repository to the lab Cloud Shell.

git clone https://github.com/GoogleCloudPlatform/training-data-analyst

Copied!

content\_copy

1. Create a soft link as a shortcut to the working directory.

ln -s ~/training-data-analyst/courses/ak8s/v1.1 ~/ak8s

Copied!

content\_copy

1. Change to the directory that contains the sample files for this lab.

cd ~/ak8s/Cloud\_Build/a

Copied!

content\_copy

A sample custom cloud build configuration file called cloudbuild.yaml has been provided for you in this directory as well as copies of the Dockerfile and the quickstart.sh script you created in the first task.

1. In Cloud Shell, execute the following command to view the contents of cloudbuild.yaml.

cat cloudbuild.yaml

Copied!

content\_copy

You will see the following:

steps:

- name: 'gcr.io/cloud-builders/docker'

args: [ 'build', '-t', 'gcr.io/$PROJECT\_ID/quickstart-image', '.' ]

images:

- 'gcr.io/$PROJECT\_ID/quickstart-image'

Copied!

content\_copy

This file instructs Cloud Build to use Docker to build an image using the Dockerfile specification in the current local directory, tag it with gcr.io/$PROJECT\_ID/quickstart-image ($PROJECT\_ID is a substitution variable automatically populated by Cloud Build with the project ID of the associated project) and then push that image to Container Registry.

1. In Cloud Shell, execute the following command to start a Cloud Build using cloudbuild.yaml as the build configuration file:

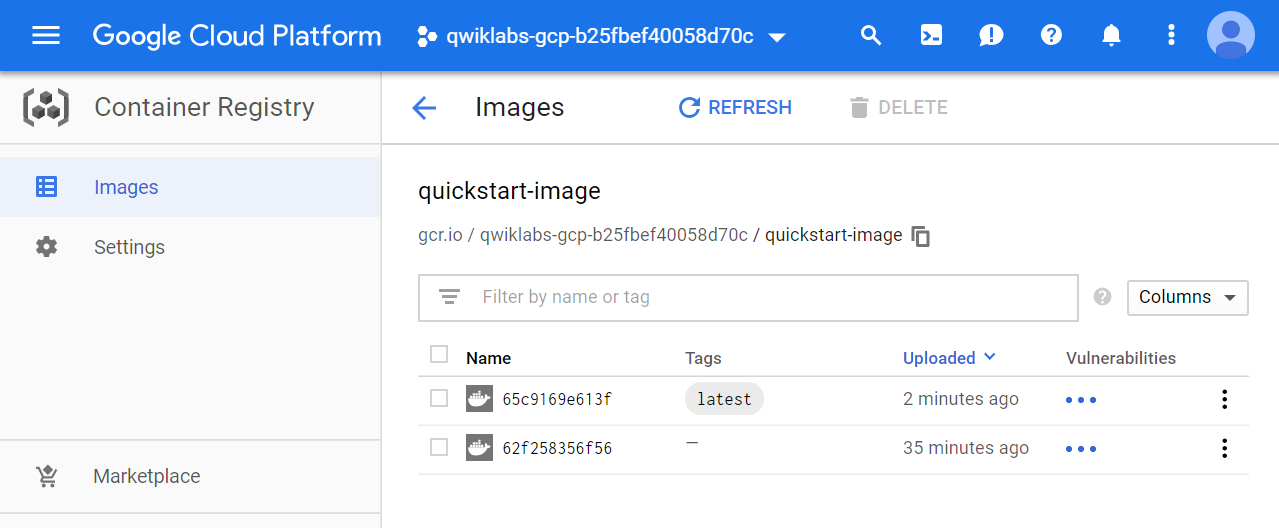
gcloud builds submit --config cloudbuild.yaml .

Copied!

content\_copy

The build output to Cloud Shell should be the same as before. When the build completes, a new version of the same image is pushed to Container Registry.

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu), click **Container Registry** > **Images** and then click quickstart-image.



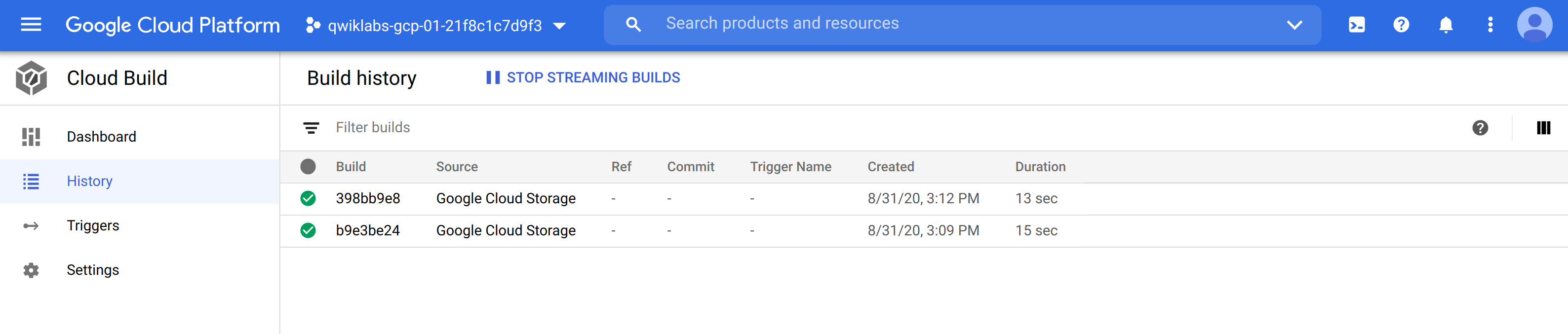
Two versions of quickstart-image are now in the list.

Click *Check my progress* to verify the objective.

Build two Container images in Cloud Build.

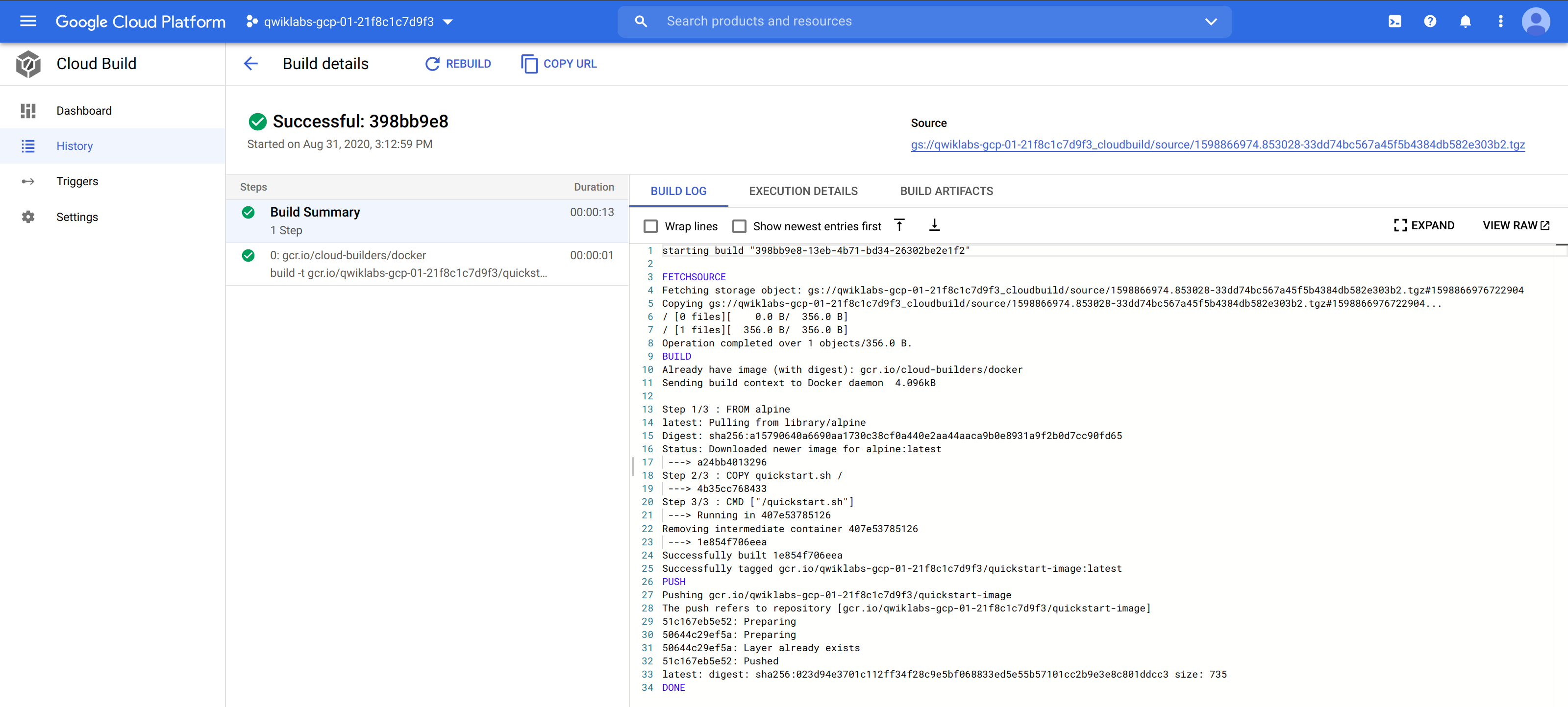
Check my progress

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu), click **Cloud Build** > **History**.



Two builds appear in the list.

1. Click the build ID for the build at the top of the list.



The details of the build, including the build log, are displayed.

**Task 4. Building and Testing Containers with a build configuration file and Cloud Build**

The true power of custom build configuration files is their ability to perform other actions, in parallel or in sequence, in addition to simply building containers: running tests on your newly built containers, pushing them to various destinations, and even deploying them to Kubernetes Engine. In this lab, we will see a simple example: a build configuration file that tests the container it built and reports the result to its calling environment.

1. In Cloud Shell, change to the directory that contains the sample files for this lab.

cd ~/ak8s/Cloud\_Build/b

Copied!

content\_copy

As before, the quickstart.sh script and the a sample custom cloud build configuration file called cloudbuild.yaml has been provided for you in this directory. These have been slightly modified to demonstrate Cloud Build's ability to test the containers it has build. There is also a Dockerfile present, which is identical to the one used for the previous task.

1. In Cloud Shell, execute the following command to view the contents of cloudbuild.yaml.

cat cloudbuild.yaml

Copied!

content\_copy

You will see the following:

steps:

- name: 'gcr.io/cloud-builders/docker'

args: [ 'build', '-t', 'gcr.io/$PROJECT\_ID/quickstart-image', '.' ]

- name: 'gcr.io/$PROJECT\_ID/quickstart-image'

args: ['fail']

images:

- 'gcr.io/$PROJECT\_ID/quickstart-image

Copied!

content\_copy

In addition to its previous actions, this build configuration file runs the quickstart-image it has created. In this task, the quickstart.sh script has been modified so that it simulates a test failure when an argument ['fail'] is passed to it.

1. In Cloud Shell, execute the following command to start a Cloud Build using cloudbuild.yaml as the build configuration file:

gcloud builds submit --config cloudbuild.yaml .

Copied!

content\_copy

You will see output from the command that ends with text like this:

**Output (do not copy)**

Finished Step #1

ERROR

ERROR: build step 1 "gcr.io/ivil-charmer-227922klabs-gcp-49ab2930eea05/quickstart-image" failed: exit status 127

----------------------------------------------------------------------------------------------------------------------------------------------------------------

ERROR: (gcloud.builds.submit) build f3e94c28-fba4-4012-a419-48e90fca7491 completed with status "FAILURE"

Copied!

content\_copy

1. Confirm that your command shell knows that the build failed:

echo $?

Copied!

content\_copy

The command will reply with a non-zero value. If you had embedded this build in a script, your script would be able to act up on the build's failure.

Click *Check my progress* to verify the objective.

Build and Test Containers with a build configuration file and Cloud Build

Check my progress

**End your lab**

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you’ve used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

* 1 star = Very dissatisfied
* 2 stars = Dissatisfied
* 3 stars = Neutral
* 4 stars = Satisfied
* 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.