**Working with Cloud Build**

experimentLabschedule1 houruniversal\_currency\_alt5 Creditsshow\_chartIntroductory

**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 the Artifact Registry.

**Objectives**

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

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

**Lab setup**

Access the lab

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

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, 1:15:00), and make sure you can finish within that time.  
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.  
   If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

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

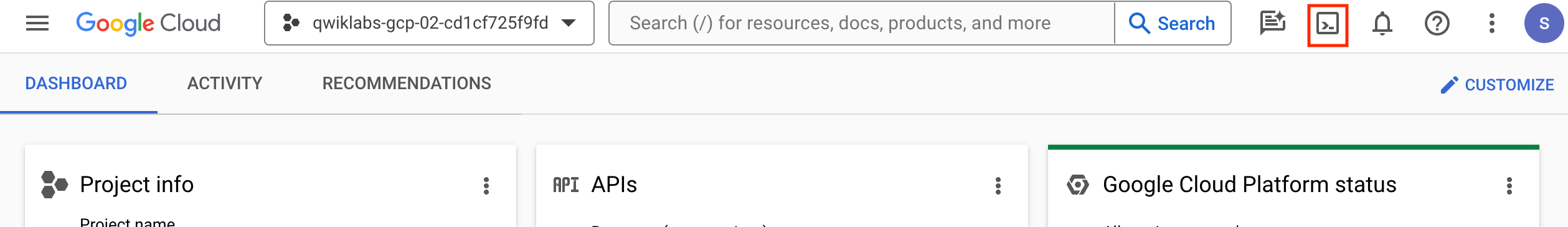
After you complete the initial sign-in steps, the project dashboard opens.

Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.



1. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT\_ID*. For example:



**gcloud** is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

* You can list the active account name with this command:

gcloud auth list

Copied!

content\_copy

**Output:**

Credentialed accounts:

- @.com (active)

**Example output:**

Credentialed accounts:

- google1623327\_student@qwiklabs.net

* You can list the project ID with this command:

gcloud config list project

Copied!

content\_copy

**Output:**

[core]

project =

**Example output:**

[core]

project = qwiklabs-gcp-44776a13dea667a6

**Note:**Full documentation of **gcloud** is available in the [gcloud CLI overview guide](https://cloud.google.com/sdk/gcloud" \t "_blank).

**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 icon), 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 **Artifact Registry**.
7. In the resulting card for the Google Artifact Registry API, if you do not see a message confirming that the Artifact 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 keys, 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 this:

FROM alpine

COPY quickstart.sh /

CMD ["/quickstart.sh"]

Copied!

content\_copy

1. Save the file and close nano by pressing the CTRL+X keys, 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. Create a new Docker repository named quickstart-docker-repo in the location europe-west4 with the description "Docker repository"

gcloud artifacts repositories create quickstart-docker-repo --repository-format=docker \

--location=europe-west4 --description="Docker repository"

Copied!

content\_copy

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

gcloud builds submit --tag europe-west4-docker.pkg.dev/${DEVSHELL\_PROJECT\_ID}/quickstart-docker-repo/quickstart-image:tag1

Copied!

content\_copy

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

1. In the Google Cloud console, in the **Search Bar** (Located at the top of the console window), Search for **Artifact Registry**.
2. Click the repository named quickstart-docker-repo.

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.

Let's create a sample custom cloud build configuration file called cloudbuild.yaml.

1. Create and open a file called cloudbuild.yaml with **nano** using the following command:

nano cloudbuild.yaml

Copied!

content\_copy

1. Once nano has opened, paste the following into the cloudbuild.yaml file:

steps:

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

args: [ 'build', '-t', 'YourRegionHere-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1', '.' ]

images:

- 'YourRegionHere-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1'

Copied!

content\_copy

1. Press **Ctrl+O**, and then press **Enter** to save your edited file.
2. Press **Ctrl+X** to exit the nano text editor.
3. Run the below command to set our region variable and insert that value into the yaml file.

export REGION=europe-west4

sed -i "s/YourRegionHere/$REGION/g" cloudbuild.yaml

Copied!

content\_copy

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', 'europe-west4-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1', '.' ]

images:

- 'europe-west4-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1'

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 Artifact 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 Artifact Registry.

1. In the Google Cloud console, in the **Search Bar** (Located at the top of the console window), Search for **Artifact Registry**.
2. Click the repository named quickstart-docker-repo.

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

Click **Check my progress** to verify the objective.

Assessment Completed!

Build two container images in Cloud Build

Check my progress

*Assessment Completed!*

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu icon), click **Cloud Build** > **History**. Two builds appear in the list.
2. 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 task, we will see a simple example, a build configuration file that tests the container it built and reports the result to its calling environment. The first step is to alter the quickstart.sh file.

1. In Cloud Shell, open quickstart.sh in nano.

nano quickstart.sh

Copied!

content\_copy

1. Replace the existing with the following:

#!/bin/sh

if [ -z "$1" ]

then

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

exit 0

else

exit 1

fi

Copied!

content\_copy

1. Press **Ctrl+O**, and then press **Enter** to save your edited file.
2. Press **Ctrl+X** to exit the nano text editor.

Let's create a new custom cloud build configuration file called cloudbuild2.yaml. This has been slightly modified to demonstrate Cloud Build's ability to test the containers it has built.

1. Create and open a file called cloudbuild2.yaml with **nano** using the following command:

nano cloudbuild2.yaml

Copied!

content\_copy

1. Once nano has opened, paste the following into the cloudbuild2.yaml file:

steps:

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

args: [ 'build', '-t', 'YourRegionHere-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1', '.' ]

- name: 'YourRegionHere-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1'

args: ['fail']

images:

- 'YourRegionHere-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1'

Copied!

content\_copy

1. Press **Ctrl+O**, and then press **Enter** to save your edited file.
2. Press **Ctrl+X** to exit the nano text editor.
3. Run the below command to insert our region value into the yaml file.

sed -i "s/YourRegionHere/$REGION/g" cloudbuild2.yaml

Copied!

content\_copy

1. In Cloud Shell, execute the following command to view the contents of cloudbuild2.yaml:

cat cloudbuild2.yaml

Copied!

content\_copy

You will see the following:

steps:

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

args: [ 'build', '-t', 'europe-west4-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1', '.' ]

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

args: ['fail']

images:

- 'europe-west4-docker.pkg.dev/$PROJECT\_ID/quickstart-docker-repo/quickstart-image:tag1'

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 cloudbuild2.yaml

Copied!

content\_copy

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

**Output**

BUILD FAILURE: Build step failure: build step 1 "us-east1-docker.pkg.dev/qwiklabs-gcp-02-1c7ba5c697a0/quickstart-docker-repo/quickstart-image:tag1" failed: starting step container failed: Error response from daemon: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: "fail": executable file not found in $PATH: unknown

ERROR: (gcloud.builds.submit) build 96c4a454-be06-4010-aa7c-da57c14165f4 completed with status "FAILURE"

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.

Assessment Completed!

Build and test containers with a build configuration file and Cloud Build

Check my progress

*Assessment Completed!*

**End your lab**

When you have completed your lab, click **End Lab**. Google Cloud Skills Boost 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 2022 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.