

Cloud Endpoints: Qwik Start

GSP164



Overview

In this lab you will deploy a sample API with [Google Cloud Endpoints](#), which are a set of tools for generating APIs from within an App Engine application. The sample code will include:

- A REST API that you can query to find the name of an airport from its three-letter `IATA` code (for example, SFO, JFK, AMS).
 - A script that uploads the API configuration to Cloud Endpoints.
 - A script that deploys a Google App Engine flexible backend to host the sample API.
- After you send some requests to the sample API, you can view the Cloud Endpoints Activity Graphs and Logs. These are tools that allow you to monitor your APIs and gain insights into their usage.

Setup and Requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

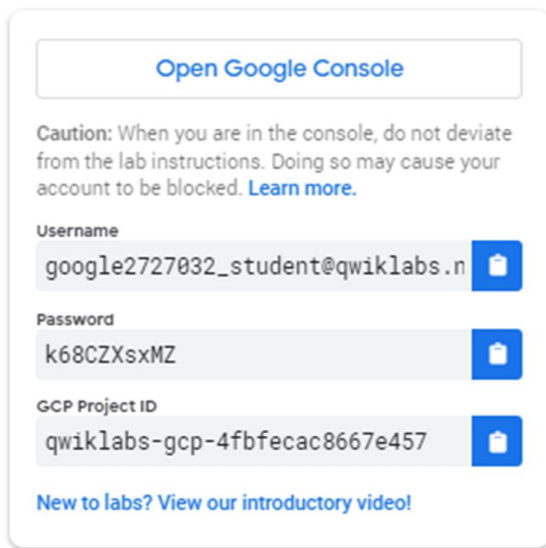
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.


How to start your lab and sign in to the Google Cloud Console


1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.




Open Google Console

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)

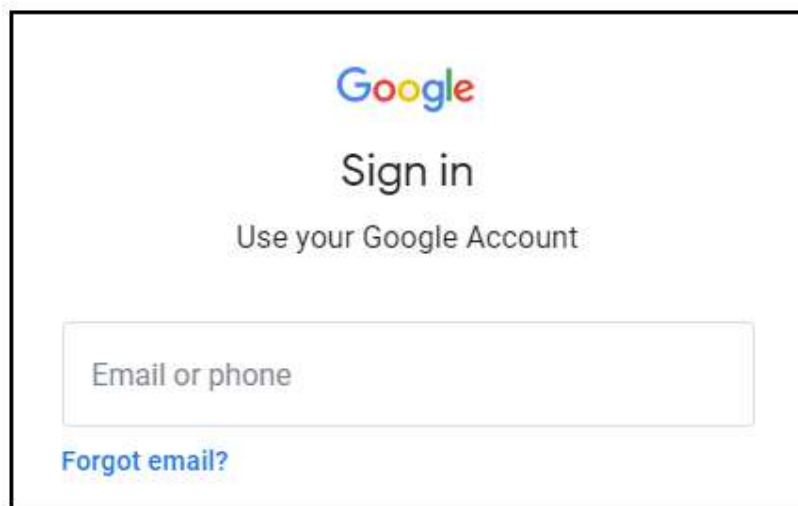
Username
google2727032_student@qwiklabs.n 

Password
k68CZXsxMZ 

GCP Project ID
qwiklabs-gcp-4fbfecac8667e457 

[New to labs? View our introductory video!](#)

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Google

Sign in

Use your Google Account

Email or phone

[Forgot email?](#)

Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another**



Account.

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

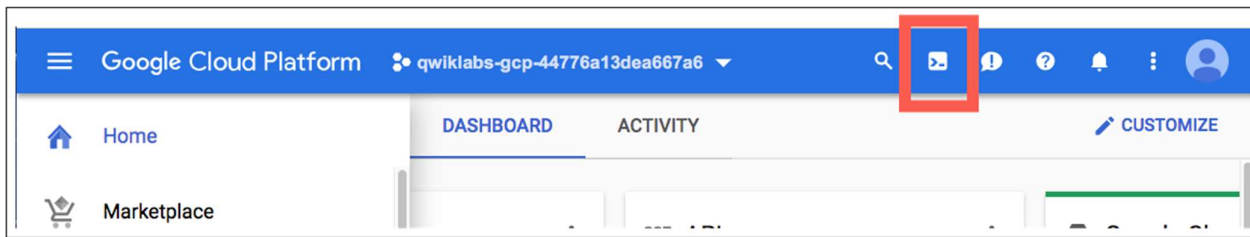
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



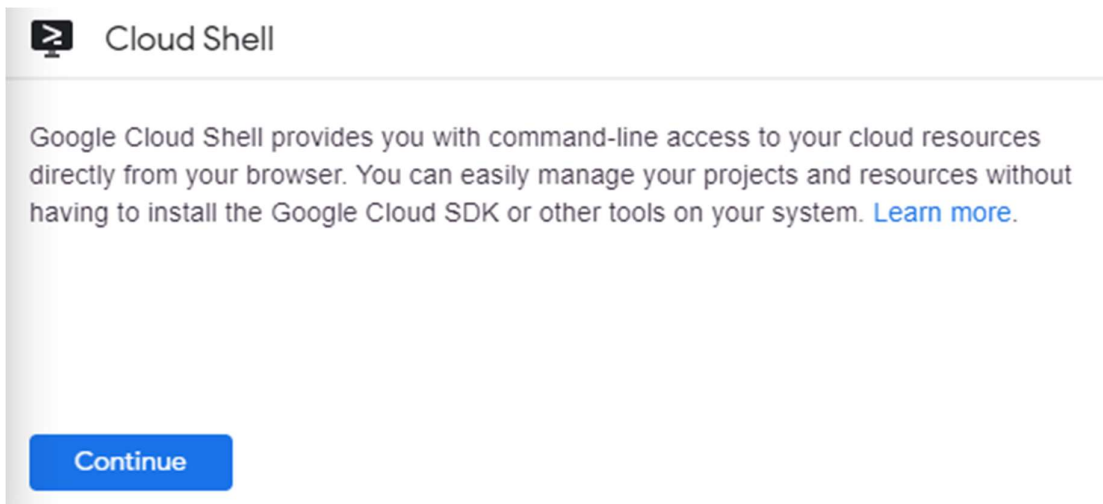
Activate Cloud Shell

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

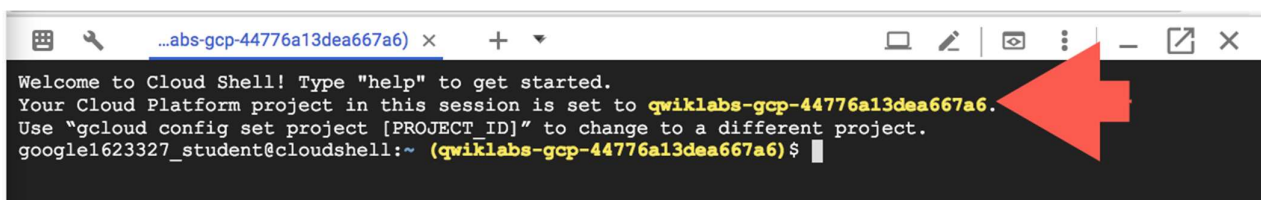
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



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
```

(Output)

```
Credentialed accounts:  
- <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:  
- google1623327 student@gwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]  
project = <project_ID>
```

(Example output)

```
[core]  
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the [gcloud command-line tool overview](#).

Getting the sample code

Enter the following command in Cloud Shell to get the sample API and scripts:

```
git clone https://github.com/GoogleCloudPlatform/endpoints-quickstart
```

Change to the directory that contains the sample code:

```
cd endpoints-quickstart
```

Deploying the Endpoints configuration

To publish a REST API to Endpoints, an OpenAPI configuration file that describes the API is required. The lab's sample API comes with a pre-configured OpenAPI file called `openapi.yaml`.

Endpoints uses Google `Service Management`, an infrastructure service of Google Cloud, to create and manage APIs and services. To use Endpoints to manage an API, you deploy the API's OpenAPI configuration to Service Management.

To deploy the Endpoints configuration.

In the `endpoints-qwikstart` directory, enter the following:

```
cd scripts
```

Run the following script, which is included in the sample:

```
./deploy_api.sh
```

Cloud Endpoints uses the `host` field in the OpenAPI configuration file to identify the service. The `deploy_api.sh` script sets the ID of your Cloud project as part of the name configured in the `host` field. (When you prepare an OpenAPI configuration file for your own service, you will need to do this manually.)

The script then deploys the OpenAPI configuration to Service Management using the command: `gcloud endpoints services deploy openapi.yaml`

As it is creating and configuring the service, Service Management outputs some information to the console. You can safely ignore the warnings about the paths in `openapi.yaml` not requiring an API key. On successful completion, you see a line like the following that displays the service configuration ID and the service name:

```
Service Configuration [2017-02-13-r2] uploaded for service [airports-  
api.endpoints.example-project.cloud.goog]
```

Deploying the API backend

So far you have deployed the OpenAPI configuration to Service Management, but you have not yet deployed the code that will serve the API backend. The `deploy_app.sh` script included in the lab sample creates an App Engine flexible environment to host the API backend, and then the script deploys the API to App Engine.

To deploy the API backend, make sure you are in the `endpoints-quickstart/scripts` directory. Then, run the following script:

```
./deploy_app.sh
```

The script runs the following command to create an App Engine flexible environment in the us-central region: `gcloud app create --region="$REGION"`

It takes a couple minutes to create the App Engine flexible backend. You'll see the following displayed in Cloud Shell after the App Engine is created:

```
Success! The app is now created. Please use `gcloud app deploy` to deploy your first app.
```

The script goes on to run the `gcloud app deploy` command to deploy the sample API to App Engine.

You'll then see a line like the following in Cloud Shell:

```
Deploying ../app/app_template.yaml...You are about to deploy the following services:
```

It takes several minutes for the API to be deployed to App Engine. You'll see a line like the following when the API is successfully deployed to App Engine:

```
Deployed service [default] to [https://example-project.appspot.com]
```


Sending requests to the API

After deploying the sample API, you can send requests to it by running the following script:

```
./query_api.sh
```

The script echoes the `curl` command that it uses to send a request to the API, and then displays the result. You'll see something like the following in Cloud Shell:

```
curl "https://example-project.appspot.com/airportName?iataCode=SFO"  
San Francisco International Airport
```

The API expects one query parameter, `iataCode`, that is set to a valid IATA airport code such as SEA or JFK.

To test, run this example in Cloud Shell:

```
./query_api.sh JFK
```

You just deployed and tested an API in Cloud Endpoints!

Tracking API activity

With APIs deployed with Cloud Endpoints, you can monitor critical operations metrics in the Cloud Console and gain insight into your users and usage with Cloud Logging.

Run this traffic generation script in Cloud Shell to populate the graphs and logs.

```
./generate_traffic.sh
```

Note This script generates requests in a loop and automatically times out in 5 minutes. To end the script sooner, enter **Ctrl-C** in Cloud Shell.

In the Console, click on **Endpoints** in the Tools section to look at the activity graphs for your API. It may take a few moments for the requests to be reflected in the graphs. You can do this while you wait for data to be displayed:

- If the Permissions side panel is not open, click **+Permissions**. The Permissions panel allows you to control who has access to your API and the level of access.
- Click the **Deployment history** tab. This tab displays a history of your API deployments, including the deployment time and who deployed the change.
- Click the **Overview** tab. Here you'll see the traffic coming in. After the traffic generation script has been running for a minute, scroll down to see the three lines on the Total latency graph (50th, 95th, and 98th percentiles). This data provides a quick estimate of response times.

At the bottom of the Endpoints graphs, under Method, click the **View all logs** link for GET/airportName. The Logs Viewer page displays the request logs for the API.

Enter **Ctrl-C** in Cloud Shell to stop the script.

Add a quota to the API

NOTE: This is a beta release of Quotas. This feature might be changed in backward-incompatible ways and is not subject to any SLA or deprecation policy.

Cloud Endpoints lets you set quotas so you can control the rate at which applications can call your API. Quotas can be used to protect your API from excessive usage by a single client.

1. Deploy the Endpoints configuration that has a quota:

```
2. ./deploy_api.sh ../openapi_with_ratelimit.yaml
```

3. Redeploy your app to use the new Endpoints configuration (this may take a few minutes):

```
4. ./deploy_app.sh
```

5. In the Console, navigate to **Navigation menu > API & Services > Credentials**.

6. Click **Create credentials** and choose **API key**. A new API key is displayed on the screen.

7. Click the double rectangle icon to copy it to your clipboard.

8. In Cloud Shell, type the following. Replace YOUR-API-KEY with the API key you just created:

```
9. export API_KEY=YOUR-API-KEY
```

10. Send your API a request using the API key variable you just created:

```
11. ./query_api_with_key.sh $API_KEY
```

12. You'll see something like the following on the console:

```
13. curl -H 'x-api-key: AIzeSyDbdQdaSdhPMdiAuddd FALbY7JevoMzAB' "https://example-project.appspot.com/airportName?iataCode=SFO"
```

```
14. San Francisco International Airport
```

15. The API now has a limit of 5 requests per second. Run the following command to send traffic to the API and trigger the quota limit:

```
16. ./generate_traffic_with_key.sh $API_KEY
```

17. After running the script for 5-10 seconds, enter **Ctrl-c** in Cloud Shell to stop the script.

18. Send another authenticated request to the API:

```
19. ./query_api_with_key.sh $API_KEY
```

You'll see something like the following on the console:

```
{
  "code": 8,
  "message": "Insufficient tokens for quota 'airport_requests' and limit 'limit-on-airport-requests' of service 'example-project.appspot.com' for consumer 'api_key:AIzeSyDbdQdaSdhPMdiAuddd_FALbY7JevoMzAB'.",
  "details": [
    {
      "@type": "type.googleapis.com/google.rpc.DebugInfo",
      "stackEntries": [],
      "detail": "internal"
    }
  ]
}
```

If you get a different response, try running the `generate_traffic_with_key.sh` script again and retry.

Congratulations!

Congratulations! You've successfully rate-limited your API. You can also set varying limits on different API methods, create multiple kinds of quotas, and keep track of which consumers use which APIs.

Take your next lab

This lab is part of a series of labs called Qwik Starts. These labs are designed to give you a little taste of the many features available with Google Cloud. Search for “Qwik Starts” in the lab catalog to find the next lab you’d like to take!

Next Steps/Learn More

For more information about quotas, see the following:

- [Quotas for OpenAPI](#)
- [Quotas for Endpoints Frameworks](#)
- [Quotas for gRPC](#)

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual last updated: September 4, 2018

Lab last tested: June 28, 2018

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.