

Introduction to APIs in Google Cloud

experiment Lab schedule 30 minutes universal_currency_alt No cost

show_chart Introductory

GSP294



Google Cloud Self-Paced Labs

Overview

APIs (Application Programming Interfaces) are software programs that give developers access to computing resources and data. Companies from many different fields offer publicly available APIs so that developers can integrate specialized tools, services, or libraries with their own applications and codebase.

This lab reviews the architecture and basic functioning of APIs. This also provides hands-on practice, where you configure and run Cloud Storage API methods in Google Cloud Shell. Taking this lab helps you understand the key principles of API communication, architecture, and authentication. You also gain practical experience with APIs, which you can apply to future labs or projects.

APIs - what and why

The ability to access data and computing resources greatly increases a developer's efficiency. It is much easier to use an API than to build every single program, method, or dataset from scratch. APIs are built on the principle of abstraction—you don't need to understand the inner workings or complexities of an API to use it in your own environment.

APIs are built with the developer in mind and often don't offer a graphical user interface (GUI). However, there are exceptions to this standard. Google has released a new tool called APIs Explorer, which allows you to explore various Google APIs interactively (be sure to check out the APIs Explorer: Qwik Start lab afterwards if you are interested in learning more.)

Cloud APIs

Google offers APIs that can be applied to many different fields and sectors. APIs are often used in web development, machine learning, data science, and system administration workflows. However, these are only a handful of use cases. By exploring AnyAPI, for example, you see that there are many APIs available.

When Qwiklabs provisions a new Google Cloud Project for a lab instance, it enables most APIs behind the scenes so you can work on the lab's tasks right away. If you create your own projects outside of Qwiklabs, you must enable certain APIs yourself.

As you gain proficiency as a Google Cloud user, you use more APIs in your workflow. Experienced users integrate and use Cloud APIs in their local environments almost exclusively, rarely using the Cloud Console to run tools and services. Dozens of Hands-on Labs are available that give you practice with various Google APIs in different languages. Here are two for example:

- Cloud Natural Language API: Qwik Start
- Entity and Sentiment Analysis with the Natural Language API

In this lab, you explore the API library to see what Google APIs are available.

Objectives

In this lab, you learn about:

- Google APIs
- API architecture
- HTTP protocol and methods
- Endpoints
- REST (Representational State Transfer) and RESTful APIs
- JSON (JavaScript Object Notation)
- API authentication services

Prerequisites

This is an **introductory level** lab. This assumes little to no prior knowledge of APIs or experience using Google APIs. Familiarity with shell environments and command line interface tools is recommended, but not required. Familiarity with the Cloud Console and Cloud Storage is recommended, so please at a minimum take the following labs before attempting this one:

- A Tour of Qwiklabs and Google Cloud
- Cloud Storage: Qwik Start - Cloud Console

Once you're ready, scroll down and follow the steps below to set up your lab environment.

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 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.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab---remember, once you start, you cannot pause a lab.

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

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 the **Lab Details** panel with the following:

- The **Open Google Cloud console** button
- Time remaining
- The temporary credentials that you must use for this lab
- Other information, if needed, to step through this lab

2. Click **Open Google Cloud console** (or right-click and select **Open Link in Incognito Window** if you are running the Chrome browser).

The lab spins up resources, and then opens another tab that shows the **Sign in** page.

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

Note: If you see the **Choose an account** dialog, click **Use Another Account**.

3. If necessary, copy the **Username** below and paste it into the **Sign in** dialog.

"Username"

content_co

You can also find the **Username** in the **Lab Details** panel.

4. Click **Next**.

5. Copy the **Password** below and paste it into the **Welcome** dialog.

"Password"

content_co

You can also find the **Password** in the **Lab Details** panel.

6. Click **Next**.

Important: You must use the credentials the lab provides you. Do not use your Google Cloud account credentials.

Note: Using your own Google Cloud account for this lab may incur extra charges.

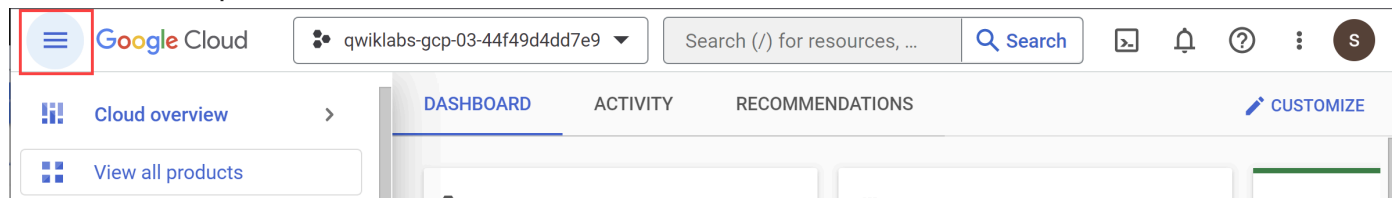
7. Click through the subsequent pages:

- 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 Google Cloud console opens in this tab.

Note: To view a menu with a list of Google Cloud products and services, click 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.

1. Click **Activate Cloud Shell**  at the top of the Google Cloud console.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `PROJECT_ID`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to "PROJECT_ID"
```

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

2. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

content_c

3. Click **Authorize**.

Output:

```
ACTIVE: *  
ACCOUNT: "ACCOUNT"
```

To **set** the active account, run:
`$ gcloud config set account `ACCOUNT``

4. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

content_co

Output:

```
[core]  
project = "PROJECT_ID"
```

Note: For full documentation of `gcloud`, in Google Cloud, refer to the [gcloud CLI overview guide](#).

Set the region for your project

Run the following command to set the region for your project:

```
gcloud config set compute/region Region
```

content_co

How do APIs work?

API architecture

APIs (Application Programming Interfaces) are a set of methods that allow programs to communicate with each other. To communicate effectively, programs must adhere to a clear protocol that governs the transfer and interpretation of data. The internet is the standard communication channel that APIs use to transmit requests and responses between programs. Web-based APIs use the client-server model as the underlying architecture for exchanging information. The client is a computing device that makes a request for some computing resource or data, and the server has data and/or computing resources stored on it, which interprets and fulfills the client's request.

HTTP protocol and methods

Since APIs use the web as a communication channel, many of them adhere to the HTTP protocol, which specifies rules and methods for data exchange between clients and servers over the internet. APIs that utilize the HTTP protocol use HTTP request methods (also known as "HTTP verbs") for transmitting client requests to servers. The most commonly used HTTP request methods are GET, POST, PUT, and DELETE. GET is used by a client to fetch data from a server, PUT replaces existing data or creates data if it does not exist, POST is used primarily to create new resources, and DELETE removes data or resources specified by the client on a server.

Endpoints

APIs use HTTP methods to interact with data or computing services hosted on a server. These methods are useless if there isn't a way to access specific resources with consistency. APIs utilize communication channels called endpoints so that clients can access the resources they need without complication or irregularity.

Endpoints are access points to data or computing resources hosted on a server and they take the form of an HTTP URI. Endpoints are added to an API's base URL to create a path to a specific resource or container of resources. Additionally, query strings can be added to endpoints to pass in variables that may be needed to complete an API's request.

REST and RESTful APIs

APIs that utilize the HTTP protocol, request methods, and endpoints are referred to as RESTful APIs. RESTful APIs live on the server, acting as an implementer for client requests. This model defines a framework of endpoints (nouns) that HTTP methods (verbs) act on, and APIs use this framework to fulfill requests. To summarize, RESTful APIs utilize the client-server model, adhere to the HTTP protocol, utilize HTTP request methods, and utilize endpoints to access specific resources.

Test your understanding

Answer the following multiple choice questions to reinforce your understanding of the concepts covered so far.



A computing device (e.g. a smartphone, laptop, etc.) that makes a request for some computing resource or data.

- ☐ Server
- ☐ Web-based API
- ☐ Client
- ☐ Client-server model

Submit



An HTTP method commonly used for creating new resources.

- ☐ PUT
- ☐ GET
- ☐ DELETE
- ☐ POST

Task 1. Using the API library

In this section, you explore the API library and enable an API. The API library is a centralized location for all Google APIs. You can use the API library to enable, disable, and manage APIs across your projects.

1. Open the **Navigation menu** and select **APIs & Services > Library**.

The **API library** offers quick access, documentation, and configuration options for 200+ Google APIs. Even though it's housed in the Console, it's important to note that the library offers access to *all Google APIs* — not only Google Cloud centric ones. This highlights an important theme: APIs are fundamental to all Google services, and Cloud APIs don't all fall under the Google Cloud category.

Time for some hands-on practice enabling an API in the API library. Assume that you are a mobile developer for a fitness site and you want to use the Google Fitness API to build your application.

2. Type `Fitness API` and press **ENTER** into the **Search for APIs and Service** search bar,.
3. Click the **Fitness API** from the result list. Then, click **Enable**.

The **Fitness API** page opens and the API **Status** is **Enabled**.

The API library provides links to tutorials and documentation, terms of service, and interactive methods offered in the APIs Explorer. To see metric and usage information, use the **APIs & Services** dashboard. In this page, you view and request quotas, control access to resources and data, and view metrics.

4. To see one of these features in action, select the **QUOTAS & SYSTEM LIMITS** tab.
5. This shows you how many queries this API allows per day and per minute.

You've now provisioned a non-Cloud API. In the next sections, you learn about the architecture and basic functioning of APIs. You then practice using the Cloud Storage API.

Test your understanding

Answer the following multiple choice questions to reinforce your understanding of the concepts covered so far.



Offers quick access, documentation, and configuration options for 200+ Google APIs.

- ☐ API Library
- ☐ APIs & Services
- ☐ Google Cloud Project
- ☐ Dialogflow

Submit



Details your project's usage of specific APIs, including traffic levels, error rates, and even latencies.

- ☐ Google Cloud Project

- ☐ Dialogflow
- ☐ API Library
- ☐ APIs & Services

Submit

Task 2. Creating a JSON File in the Cloud Console

To apply what you've learned, you make Cloud Storage REST/JSON API calls in Cloud Shell to create buckets and upload content.

1. In a new tab, open Google Cloud Storage JSON API to ensure that the Cloud Storage API is enabled.
Notice the API is enabled.
2. In Cloud Shell, run the following command to create and edit a file called `values.json`:

```
nano values.json
```

content_co

3. Inside the `nano` text editor copy and paste the following. Since the bucket must have a unique bucket name, the **Project ID** is used in the bucket name:

```
{  "name": "Project_ID-bucket",  
  "location": "us",  
  "storageClass": "multi_regional"  
}
```

content_co

Note: You can use your Project ID as your bucket name since it is globally unique.

4. Once you have, exit out of the `nano` text editor by pressing **CTRL+X**, **Y**, and then **ENTER**.

You just created a JSON file that contains an object that has three key-value pairs: `name`, `location`, and `storageClass`. These are the same values that are required when you make a bucket with the `gsutil` command line tool or in the console.

Before a bucket can be created with the Cloud Storage REST/JSON API, you need to get the proper authentication and authorization policies in place.

Task 3. Authenticate and authorize the Cloud Storage JSON/REST API

Authentication and authorization

Authentication and **authorization** are two terms that are often used interchangeably, but they are not the same thing.

- *Authentication* refers to the process of determining a client's identity.
- *Authorization* refers to the process of determining what permissions an authenticated client has for a set of resources.

Authentication identifies who you are, and authorization determines what you can do.

There are three types of authentication/authorization services that Google APIs use. These are "API Keys", "Service accounts", and "OAuth". An API uses one of these authentication services depending on the resources it requests and from where the API is called from.

API keys

API keys are secret tokens that usually come in the form of an encrypted string. API keys are quick to generate and use. Oftentimes, APIs that use public data or methods and want to get developers up and running use API keys to quickly authenticate users.

In Google Cloud terms, API keys identify the calling project making the call to an API. By identifying the calling project, API keys enable usage information to be associated with that project, and they can reject calls from projects that haven't been granted access or enabled by the API.

OAuth

OAuth tokens are similar to API keys in their format, but they are more secure and can be linked to user accounts or identities. These tokens are used primarily when APIs give a developer the means to access user data.

While API keys give developers access to all of an API's functionality, OAuth client IDs are all based on scope; different privileges are granted to different identities.

Service Accounts

A **service account** is a special type of Google account that belongs to your application or a virtual machine (VM) instead of to an individual end user. Your application assumes the identity of the service account to call Google APIs, so that the users aren't directly involved.

You can use a service account by providing its private key to your application, or by using the built-in service accounts available when running on Cloud Functions, Google App Engine, Compute Engine, or Google Kubernetes Engine.

For a lab specifically dealing with service accounts and roles, refer to: [Service Accounts and Roles: Fundamentals](#).

Since Cloud Storage is a platform that hosts and provides access to user data, you need to generate an OAuth token before you use its services.

1. Open the OAuth 2.0 playground in a new tab. This is a service that allows you to generate OAuth tokens with ease.
2. Scroll down and select **Cloud Storage API V1**.
3. Then select the **https://www.googleapis.com/auth/devstorage.full_control** scope.
4. Click on the blue box that says **Authorize APIs**. This opens the Google Sign-in page.
5. Select your username and then click **Allow** when prompted for permissions.

OAuth 2.0 Playground opens, notice that Step 2 has an authorization code generated.

6. Click on **Exchange authorization code for tokens**. If you get moved to Step 3, click on the Step 2 panel.
7. **Copy** the access token to use in the next step.

Task 4. Create a bucket with the Cloud Storage JSON/REST API

1. Return to your Cloud Shell session. At the CLI prompt, type in `ls` and hit enter. You should see the `values.json` file that you created before and a `README-cloudshell.txt` file:

Output:

```
README-cloudshell.txt  values.json
```

2. Run the following command to set your OAuth2 token as an environment variable, replacing `<YOUR_TOKEN>` with the access token you generated:

```
export OAUTH2_TOKEN=<YOUR_TOKEN>
```

content_co

3. Run the following command to set your Project ID as an environment variable:

```
export PROJECT_ID=$(gcloud config get-value project)
```

content_co

4. Now run the following command to create a Cloud Storage bucket:

```
curl -X POST --data-binary @values.json \  
-H "Authorization: Bearer $OAUTH2_TOKEN" \  
-H "Content-Type: application/json" \  
"https://www.googleapis.com/storage/v1/b?project=$PROJECT_ID"
```

content_co

5. You should receive a similar output:

```
{  
  "kind": "storage#bucket",  
  "selfLink": "https://www.googleapis.com/storage/v1/b/qwiklabs-gcp-02-  
5d551758b5a7",  
  "id": "qwiklabs-gcp-02-5d551758b5a7",  
  "name": "qwiklabs-gcp-02-5d551758b5a7",  
  "projectNumber": "670840659006",  
  "metageneration": "1",  
  "location": "US",  
  "storageClass": "MULTI_REGIONAL",  
  "etag": "CAE=",  
  "timeCreated": "2020-11-11T06:41:40.901Z",  
  "updated": "2020-11-11T06:41:40.901Z",  
  "iamConfiguration": {  
    "bucketPolicyOnly": {  
      "enabled": false  
    },  
    "uniformBucketLevelAccess": {  
      "enabled": false  
    }  
  },  
  "locationType": "multi-region"  
}
```

Note: If you received an error message like "Use of this bucket name is restricted" or "Sorry, that name is not available", it means that there is a conflict with the universal bucket naming

convention. Edit the `values.json` file and replace the bucket name.

This request is the culmination of everything you've learned about so far. You used the curl CLI tool to make an HTTP POST method request. You passed in the `values.json` file into the request body. You passed the OAuth token and a JSON specification as request headers. This request was routed to the Cloud Storage endpoint, which contains a query string parameter set to your Project ID.

View your newly created Cloud Storage bucket

- To see your newly created bucket, from the **Navigation menu** select **Cloud Storage > Buckets**.

Test completed task

Click **Check my progress** to verify your performed task. The assessment score updates if you've successfully created a bucket with the Cloud Storage JSON/REST API.



Create a bucket with the Cloud Storage JSON/REST API

Check my progress

Task 5. Upload a file using the Cloud Storage JSON/REST API

You can use the Cloud Storage JSON/REST API to upload files to buckets.

1. Save the following image to your computer and name it **demo-image.png**:



2. In your Cloud Shell session, click on the three-dotted menu icon in the top-right corner.
Click **Upload** > **Choose File**. Select and upload `demo-image.png` file. This adds the image to your directory.
3. Run the following command to get the path to the image file:

```
realpath demo-image.png
```

content_c

You should receive a similar output:

```
/home/gcpstaging25084_student/demo-image.png
```

4. Set the file path as an environment variable by running the following command, replacing `<DEMO_IMAGE_PATH>` with your output from the previous command:

```
export OBJECT=<DEMO_IMAGE_PATH>
```

content_co

5. Set your bucket name as an environment variable by running the following command:

```
export BUCKET_NAME=Project_ID-bucket
```

content_co

6. Now run the following command to upload the demo image to your Cloud Storage bucket:

```
curl -X POST --data-binary @$OBJECT \  
-H "Authorization: Bearer $OAUTH2_TOKEN" \  
-H "Content-Type: image/png" \  
"https://www.googleapis.com/upload/storage/v1/b/$BUCKET_NAME/o?uploadType=media&name=demo-image"
```


content_co

You should receive a similar output:

```
{  
  "kind": "storage#object",  
  "id": "qwiklabs-gcp-02-5d551758b5a7/demo-image/1605077118178936",  
  "selfLink": "https://www.googleapis.com/storage/v1/b/qwiklabs-gcp-02-5d551758b5a7/o/demo-image",  
  "mediaLink": "https://www.googleapis.com/download/storage/v1/b/qwiklabs-gcp-02-5d551758b5a7/o/demo-image?generation=1605077118178936&alt=media",  
  "name": "demo-image",  
  "bucket": "qwiklabs-gcp-02-5d551758b5a7",  
  "generation": "1605077118178936",  
  "metageneration": "1",  
  "contentType": "image/png",  
  "storageClass": "MULTI_REGIONAL",  
  "size": "401951",  
  "md5Hash": "LbpHpwhnApQKQx9IEXjTsQ==",  
  "crc32c": "j5oPrg==",  
  "etag": "CPis3Zvy+ewCEAE=",  
  "timeCreated": "2020-11-11T06:45:18.178Z",  
  "updated": "2020-11-11T06:45:18.178Z",  
  "timeStorageClassUpdated": "2020-11-11T06:45:18.178Z"  
}
```

7. To see the image that was added to your bucket, open the navigation menu and select **Cloud Storage > Buckets**.

8. Click on the name of your bucket to see that `demo-image` has been added:

Buckets > qwiklabs-gcp-03-7ed510a4c11b 

UPLOAD FILES

UPLOAD FOLDER



CREATE FOLDER

MANAGE HOLDS

DOWNLOAD

DELETE

 Filter Filter by object or folder name prefix

<input type="checkbox"/>	Name	Size	Type	Created time 	Storage class
<input type="checkbox"/>	 demo-image	392.5 KB	image/png	Jan 7, 2021, 5:12:17 PM	Multi-regional

9. Click on the image name to open the **Object details** page.

Test completed task

Click **Check my progress** to verify your performed task. The assessment score updates ff you have successfully uploaded a file using the Cloud Storage JSON/REST API



Upload a file using the Cloud Storage JSON/REST API

Check my progress

Congratulations!