

experiment Lab    schedule 1 hour 30 minutes    universal\_currency\_alt No cost

show\_chart Introductory

## Overview

An OpenAPI specification uses a standard format to describe a RESTful API. Written in either JSON or YAML format, an OpenAPI specification is machine readable, but is also easy for people to read and understand.

The specification describes elements of an API, including its base path, resource paths and verbs, operations, headers, query parameters, and responses. In addition, an OpenAPI specification is commonly used to generate API documentation.

In this lab, you will examine an OpenAPI specification for a retail backend service. You will then use this OpenAPI specification to create an API proxy that will be used to add features and security to the backend API.

## Objectives

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

- Explore an OpenAPI specification and understand the different components.
- Create an API proxy from an OpenAPI specification using the proxy wizard.
- Trace an API proxy.

# Setup

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.

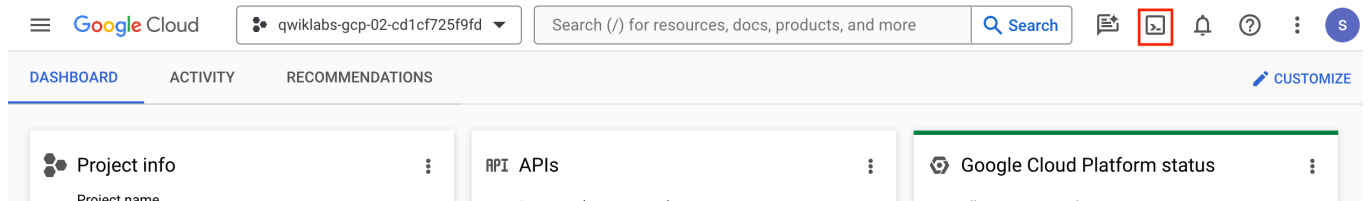
**Note:** Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

## 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.



## 2. 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:

```
...abs-gcp-44776a13dea667a6) x + v
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6) $
```

**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
```

content\_c

### Output:

```
Credentialed accounts:
- @.com (active)
```

### Example output:

```
Credentialed accounts:
- google1623327_student@qwiklabs.net
```

- You can list the project ID with this command:

content\_c

```
gcloud config list project
```

### Output:

```
[core]  
project =
```

### Example output:

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

**Note:** Full documentation of **gcloud** is available in the gcloud CLI overview guide .

## Task 1. Examine the OpenAPI specification for the backend service

In this task, you will explore the OpenAPI specification that has been created for a backend service that you will use in your API proxies.

### Download the OpenAPI specification

- In **Cloud Shell**, download the OpenAPI specification for the backend service using this curl command:

```
curl https://storage.googleapis.com/cloud-training-cors/developing-  
apis/specs/retail-backend.yaml?$(date +%s) --output ~/retail-
```

content\_co

```
backend.yaml
```

This curl command downloads a file named *retail-backend.yaml* and stores it in a file with the same name in the home directory. Later in the lab, you will use this same specification when creating an API proxy.

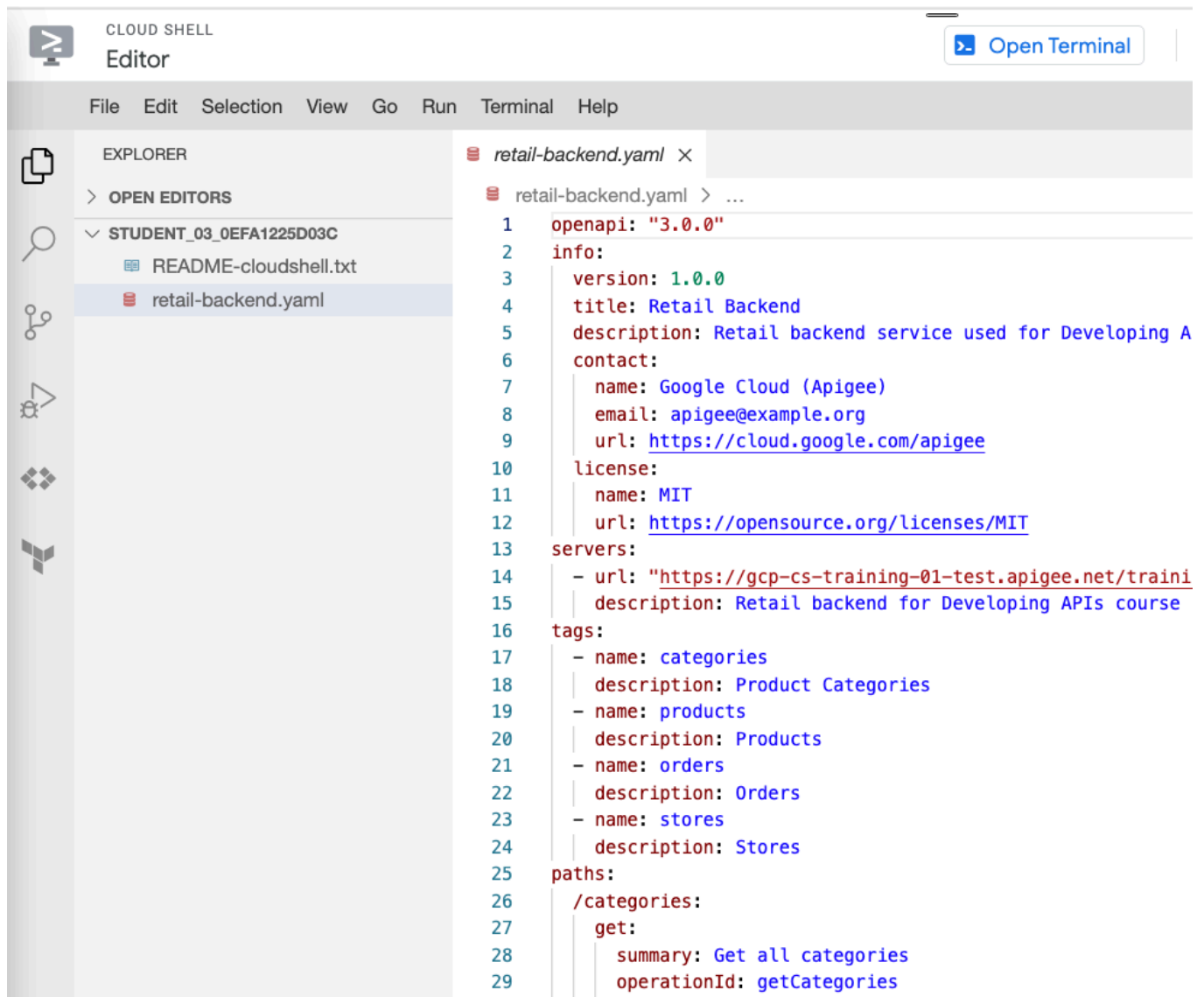
**Note:** "?\$(date +%s)" adds a query parameter to the URL that is a string representation of the current date/time. This dynamically changing variable changes the URL and forces curl to retrieve the latest version of a file, even if a previous version is cached.

## View the OpenAPI specification in Cloud Shell Editor

1. In Cloud Shell, click **Open Editor**.



2. In the editor, select the **retail-backend.yaml** file.



## Explore the sections of the spec

- Examine the OpenAPI specification.

This is the OpenAPI specification for the backend service that will be used in many of the course labs. Let's explore the sections of the OpenAPI spec.

The **openapi** field specifies the version of the OpenAPI specification. This is an OpenAPI version 3 spec, as indicated by the version number at the top of the file:

```
openapi: "3.0.0"
```

The **info** object provides metadata about the API itself. The version shown is the version of the Retail Backend specification:

```
info:
  version: 1.0.0
  title: Retail Backend
  description: Retail backend database used for Developing APIs course
  contact:
    name: Google Cloud (Apigee)
    email: apigee@example.org
    url: https://cloud.google.com/apigee
  license:
    name: MIT
    url: https://opensource.org/licenses/MIT
```

The **servers** array contains a list of server objects that specify connectivity information for target servers. This specification contains a single backend service, which your API proxy will call:

```
servers:
- url: "https://gcp-cs-training-01-test.apigee.net/training/db"
  description: Retail backend for Developing APIs course
```

The **tags** array adds metadata to tags used in the operations, which are shown below. Tags may be shared by multiple operations, and tags can be used to provide verbose descriptions or links to external documentation.

```
tags:
- name: categories
  description: Product Categories
- name: products
  description: Products
- name: orders
  description: Orders
- name: stores
  description: Stores
```

The **paths** object holds the relative paths to individual endpoints and their operations. One such path, **/categories/{categoryId}**, is used to specify a single category. Shown here is a **get** operation specified

to get a category by ID. The get object shows parameters and responses. For operations that contain a request body, like **PATCH** `/products/{productId}`, the request body will also be specified.

```
/categories/{categoryId}:
  get:
    summary: Get a specific category
    operationId: getCategoryById
    tags:
      - categories
    parameters:
      - name: categoryId
        in: path
        required: true
        description: category id
        schema:
          type: integer
    responses:
      '200':
        description: Selected category
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Category"
      '404':
        description: Category not found
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Error"
    default:
      description: unexpected error
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/Error"
```

The **components** object contains reusable objects for different portions of the OpenAPI spec.

The **securitySchemes** component object contains definitions of different types of security schemes used by the operations. This spec defines a single Basic Authentication scheme, which is referenced in the **PATCH** `/products/{productId}` operation. The **schemas** component object contains input and output data types.




Shown here is the **Category** object, which is returned when the **GET /categories/{categoryId}** operation returns successfully:

```
components:
  securitySchemes:
    basicAuth:
      type: http
      description: basic auth
      scheme: basic

  schemas:
    Category:
      type: object
      description: product category
      required:
        - color
        - id
        - name
      properties:
        color:
          description: use this color for displaying this category
          type: string
        id:
          description: integer id (used for access)
          type: integer
          format: int32
          minimum: 0
        name:
          description: category name
          type: string
```

Feel free to explore the specification or the OpenAPI specification documentation.

## Download the OpenAPI spec


1. Select Cloud Shell's *More* menu () , and then click **Download**.
2. Enter `retail-backend.yaml` , and then click **Download**.

This will download the file to your local machine.

## Task 2. Create an API proxy using the OpenAPI spec

In this task, you will create an API proxy using the OpenAPI specification for the backend service.

### Use the proxy wizard to create a proxy

1. In the Google Cloud console, on the **Navigation menu** () , select **Integration Services > Apigee > Proxy Development > API proxies**.
2. To start the proxy wizard, click **+Create**.
3. For **Proxy template**, select **OpenAPI spec template > Reverse proxy (Most common)**.
4. In **OpenAPI specs**, click **Browse**, select the `retail-backend.yaml` file you downloaded, and then click **Open**.
5. Click **Next**.
6. Specify the following for the **Proxy details**:

Property	Value
Proxy name	<b>retail-v1</b>
Base path	<b>/retail/v1</b>
Description	<b>My retail API</b>

The target was taken from the **servers** array in the OpenAPI spec. Leave the target unchanged.

**Note:** Confirm that you are using `"/retail/v1"` for the base path, and not `"/retail-v1"`.

7. Click **Next**.

Operations that were found in the OpenAPI specification are listed.

8. In Flows, on the header row, click **Select all rows**.

All flows should now be selected.

9. Click **Next**.

10. For **Deployment environments**, select the **eval** environment, and then click **OK**.

11. Click **Create**.

Your proxy will be generated and marked for deployment.

## Runtime availability may be delayed

The full provisioning of an Apigee organization typically takes 30 minutes or longer. Most of the time is spent provisioning the runtime cluster, runtime database, and services that are used to run your API proxies. When creating a long-lived Apigee organization, this delay to full provisioning is not an issue. You do not, however, want to wait half an hour before starting each lab.

You will sometimes find that the organization is already fully provisioned when you enter the lab. Sometimes the Apigee organization will only begin provisioning when you start the lab.

The management plane operations of the organization are available after a few minutes of the provisioning process. Instead of waiting for the runtime to be fully provisioned, these labs allow you to perform operations like proxy editing before the runtime is available. When you deploy a proxy to an environment before the runtime is available, it will not be able to take traffic until runtime provisioning is complete.

When you hold the pointer over the **Status** icon for a deployed proxy, as shown below, you might see that there are no instances reporting status. This is normal until the Apigee organization's runtime has been fully provisioned.

Google Cloud

qwiklabs-gcp-03-91df34bcb84e

Search (/) for resources, docs, products, and more

Apigee

Overview

Proxy development

API proxies

Shared flows

Integrations

Offline debug

API monitor

Distribution

API product

Portals

Developers


retail-v1

OVERVIEW DEVELOP DEBUG

Proxy summary

Deployments

1 Disruption

Status	Revision	Environment	
	1	eval	UNDEPLOY

eval

Status: no instances are reporting status for this environment


Deployed on: 11/27/2023, 6:19:20 PM

	Description	Last modified
1	My retail API	November 27, :

## Check deployment status

A proxy that is deployed and ready to take traffic will show a green status.

Deployments

Status	Revision	Environment	
	1	eval	UNDEPLOY

eval

Status: Deployed

Deployed on: 11/27/2023, 6:19:20 PM

Revision	Description
1	My retail API

When a proxy is marked as deployed but the runtime is not yet available and the environment is not yet attached, you may see a red warning sign. Hold the pointer over the **Status** icon to see the current status.

Google Cloud

qwiklabs-gcp-03-91df34bcb84e

Search (/) for resources, docs, products, and more

Apigee

Overview

Proxy development

API proxies

Shared flows

Integrations

Offline debug

API monitor

Distribution

API product

Portals

Developers

retail-v1

OVERVIEW DEVELOP DEBUG

Proxy summary

Deployments

1 Disruption

Status Revision Environment

! 1 eval UNDEPLOY

eval

Status: no instances are reporting status for this environment

Deployed on: 11/27/2023, 6:19:20 PM

Description Last modified

1 My retail API November 27, 2023

If the proxy is deployed and shows as green, your proxy is ready for API traffic. If your proxy is not deployed because there are no runtime pods, you can check the status of provisioning.

## Check provisioning dashboard

1. In the Google Cloud Console, navigate to **Compute Engine > VM instances**.
2. To open the Lab Startup Tasks dashboard, click on the **External IP** for the **lab-startup** VM.

Compute Engine

VM instances

CREATE INSTANCE IMPORT VM REFRESH

INSTANCES OBSERVABILITY INSTANCE SCHEDULES

VM instances

Filter Enter property name or value

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	<a href="#">apigee-proxy-07m8</a>	us-central1-f		<a href="#">apigee-proxy-group</a>	10.128.0.3 (nic0)	<a href="#">34.27.73.218</a> (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	<a href="#">apigee-proxy-dp85</a>	us-central1-f		<a href="#">apigee-proxy-group</a>	10.128.0.4 (nic0)	<a href="#">34.132.15.243</a> (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	<a href="#">apigee-proxy-gdix</a>	us-central1-f		<a href="#">apigee-proxy-group</a>	10.128.0.5 (nic0)	<a href="#">35.188.25.205</a> (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	<a href="#">lab-startup</a>	us-central1-f			10.128.0.2 (nic0)	<a href="#">34.28.102.86</a> (nic0)	SSH ▾ ⋮

3. If you see a redirect notice page, click the link to the external IP address.

A new browser window will open. Lab startup tasks are shown with their progress.

- *Create proxies, shared flows, target servers* should be complete when you first enter the lab, allowing you to use the Apigee console for tasks like proxy editing.

- *Create API products, developers, apps, KVMs, KVM data* indicates when the runtime is available and those assets may be saved.
- *Proxies handle API traffic* indicates when the eval environment has been attached to the runtime and the deployed proxies can take runtime traffic.

- Lab Startup Tasks -			
Progress	Time	State	Task
<div></div>	05:02	completed	Create proxies, shared flows, target servers (environment available)
<div></div>	30:49	completed	Create API products, developers, apps, KVMs, KVM data (runtime is available)
<div></div>	31:11	started	Proxies handle API traffic (environment attached to runtime)
<div></div>	03:34	completed	Provide access to lab
<div></div>	30:05	started	Full provisioning of Apigee org qwiklabs-gcp-02-d23d90c73c5a in us-west4
<div></div>	01:41	completed	Create Apigee load balancer at api-test-qwiklabs-gcp-02-d23d90c73c5a.apigee-api
<div></div>	00:14	completed	Connect load balancer to runtime instance

In this case, you need to wait for *Proxies handle API traffic* to complete.

## While you are waiting

- Web API Design: The Missing Link — an e-book on API design principles
- OpenAPI specification

## Task 3. Explore and trace the API proxy

1. Select the **Develop** tab.

This tab is used to edit the proxy that has been generated. Conditional flows have been created for each of the operations in the OpenAPI specification. These conditional flows appear in the proxy endpoint in the Navigator on the left. When you click on a conditional flow, its request and response sections are selected in the visual editor pane. The `default.xml` code shown below is the code representation of the proxy endpoint flows.

The screenshot displays the 'retail-v1' proxy configuration interface. On the left, a sidebar shows the project structure with 'Proxy endpoints' expanded, highlighting the 'default' endpoint and its 'PreFlow' configuration. The main area shows the 'Proxy Endpoint: default /retail/v1' configuration. It features a 'Request' and 'Response' flow diagram. The 'Request' flow starts with 'ALL PreFlow', followed by a series of GET requests (getCategories, getCategoryById, getStores, getStoreById, getProducts, getProductById, updateProductById, createOrder, getOrderById, deleteOrderById) and ends with 'ALL PostFlow'. The 'Response' flow starts with 'ALL PostFlow', followed by a series of DELETE, GET, and PATCH requests (deleteOrderById, getOrderById, createOrder, updateProductById, getProductById, getProducts, getStoreById, getStores, getCategoryById, getCategories) and ends with 'ALL PreFlow'. Red arrows point from the 'PreFlow' label in the sidebar to the 'ALL PreFlow' box in the request flow, and from the 'PostFlow' label in the sidebar to the 'ALL PostFlow' box in the response flow. At the bottom, the 'default.xml' file is shown with the XML configuration. A red arrow points to the '<PreFlow name="PreFlow">' line in the XML.

retail-v1

OVERVIEW DEVELOP DEBUG

REVISION: 1 Updated 2 hours ago

DEPLOY SAVE

Search proxy

proxy-endpoints / default.xml / PreFlow-1

API Client

Proxy Endpoint: default /retail/v1

Request

Response

default.xml

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <Description/>
4   <FaultRules/>
5   <PreFlow name="PreFlow">
6     <Request/>
7     <Response/>
8   </PreFlow>
9   <PostFlow name="PostFlow">
10    <Request/>
11    <Response/>
12  </PostFlow>
13 </ProxyEndpoint>

```

You will update many of these conditional flows in later labs.

2. Select the **Debug** tab.

The debug tool is used to trace API requests that are handled by the proxy.

3. Click **Start Debug Session**.

4. In the **Start debug session** pane, on the Environment dropdown, select **eval**.

The deployed revision number will also show in the dropdown.

5. Click **Start**.

The debug session will run for 10 minutes.

6. To send a "get categories" request to your proxy, in Cloud Shell, run the following command:

```
curl https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/content_co
```



Note that the `GOOGLE_CLOUD_PROJECT` variable contains the project ID, so this is automatically added into the URL. You will be provided curl calls that are formatted like this in future labs.

A transaction for this request should appear in the Transactions pane on the left. When a transaction is selected, you'll see a trace of the request and response through Apigee. You should see a **200 status code** if your backend URL was correctly set and you correctly updated the Send Requests URL.

**Note:** Apigee debug session traffic is retrieved by asynchronously polling for new API calls, so there will be a delay between when an API request is completed, and when it shows in the debug tool.

7. Click the **Back** and **Next** buttons to navigate through the steps of the transaction.

The request was `GET /retail/v1/categories`. This request was sent to the backend, which responded with a JSON array containing the categories.

You can use a REST client for later labs, or you can use curl in Cloud Shell. The labs will specify the curl command that you can use in Cloud Shell.

## Congratulations!

In this lab, you learned about OpenAPI specifications, and explored some of the features in OpenAPI specs. You used an OpenAPI specification for a retail backend service to create an API proxy, and you traced calls through



that proxy.

**End your lab**