1. Import and test the backend OpenAPI specification

Import the backend OpenAPI specification

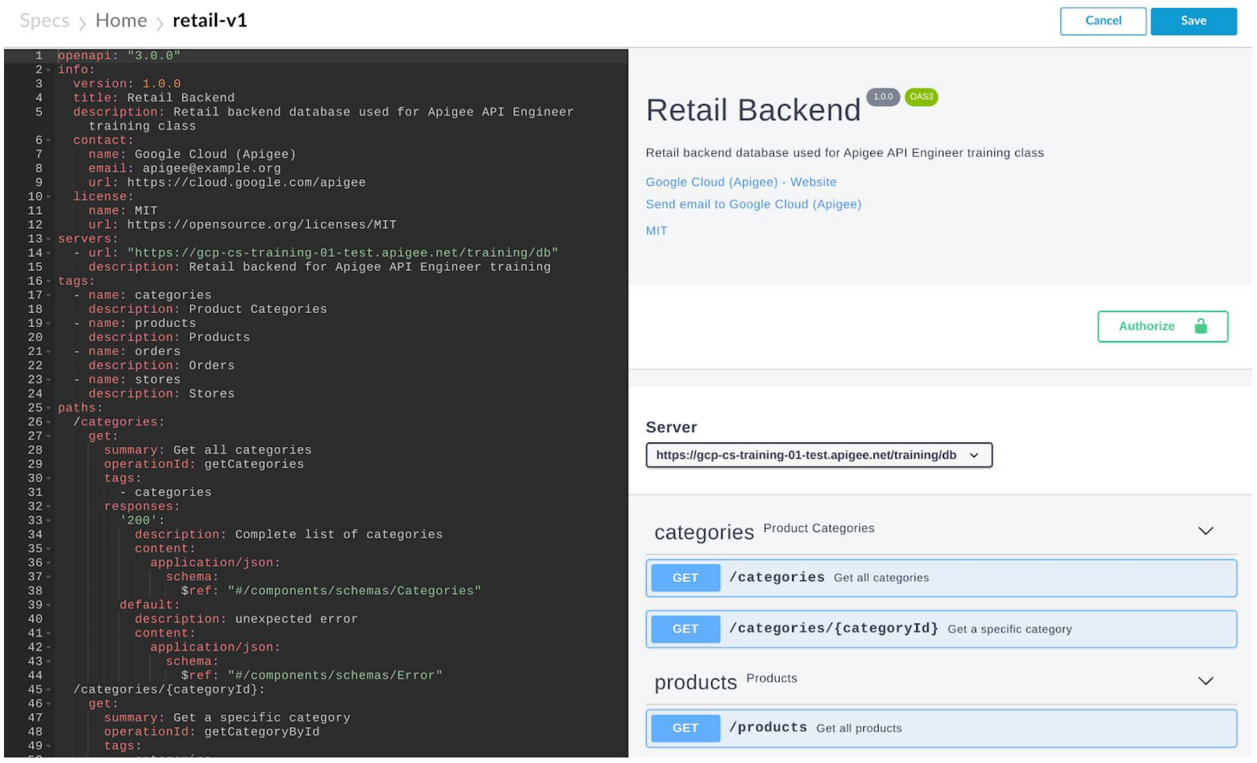
1. In a different browser tab, open the Apigee console.
2. On the left navigation menu, click **Develop > Specs**.
3. To create a new spec, on the **+Spec** menu, click **Import URL**.

   Specify the following:

Property	Value
Import name	**retail-v1**
Import URL	**https://storage.googleapis.com/cloud-training/developing-apis/specs/retail-backend.yaml**

4. Click **Import**.
5. Click on the specification name, **retail-v1**.

   When the specification editor opens, the YAML OpenAPI specification is on the left, and the live documentation is on the right:



```yaml
openapi: "3.0.0"
info:
  version: 1.0.0
  title: Retail Backend
  description: Retail backend database used for Apigee API Engineer training class
  contact:
    name: Google Cloud (Apigee)
    email: apigee@example.org
    url: https://cloud.google.com/apigee
  license:
    name: MIT
    url: https://opensource.org/licenses/MIT
servers:
  - url: "https://gcp-cs-training-01-test.apigee.net/training/db"
    description: Retail backend for Apigee API Engineer training
tags:
  - name: categories
    description: Product Categories
  - name: products
    description: Products
  - name: orders
    description: Orders
  - name: stores
    description: Stores
paths:
  /categories:
    get:
      summary: Get all categories
      operationId: getCategories
      tags:
        - categories
      responses:
        '200':
          description: Complete list of categories
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Categories"
        default:
          description: unexpected error
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Error"
  /categories/{categoryId}:
    get:
      summary: Get a specific category
      operationId: getCategoryById
      tags:
```

Retail Backend 1.0.0 OAS3

Retail backend database used for Apigee API Engineer training class

Google Cloud (Apigee) - Website
Send email to Google Cloud (Apigee)
MIT

categories Product Categories

GET `/categories` Get all categories

GET `/categories/{categoryId}` Get a specific category

products Products

GET `/products` Get all products

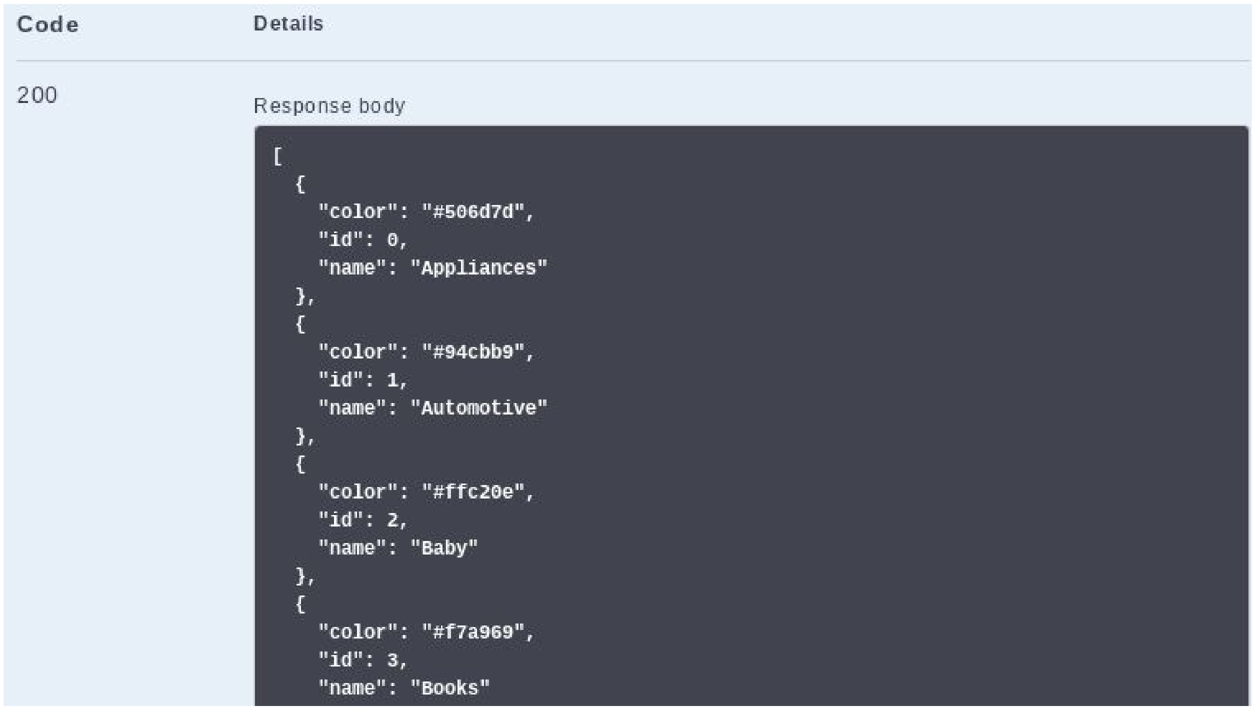
Changes made to the YAML specification on the left are immediately reflected in the documentation on the right.

Test the backend OpenAPI specification

1. In the live documentation, click on the **GET /categories** box to expand the panel.

2. Click **Try it out**, and then click **Execute**.

   You should see a successful response.



```
[
  {
    "color": "#506d7d",
    "id": 0,
    "name": "Appliances"
  },
  {
    "color": "#94cbb9",
    "id": 1,
    "name": "Automotive"
  },
  {
    "color": "#ffc20e",
    "id": 2,
    "name": "Baby"
  },
  {
    "color": "#f7a969",
    "id": 3,
    "name": "Books"
```

2. Generate the API proxy

In this task, you use the OpenAPI specification to generate an API proxy.

Generate an API proxy using the OpenAPI specification

1. On the left navigation menu, click **Develop > API Proxies**.
2. To start the proxy wizard, click **+Proxy**.
3. Click **Use OpenAPI Spec** in the **Reverse proxy** box.

4. Select the **retail-v1** specification, and then click **Select**.
5. Specify the following:

Property	Value
Name	**retail-v1**
Base path	**/retail/v1**
Description	**My retail API**

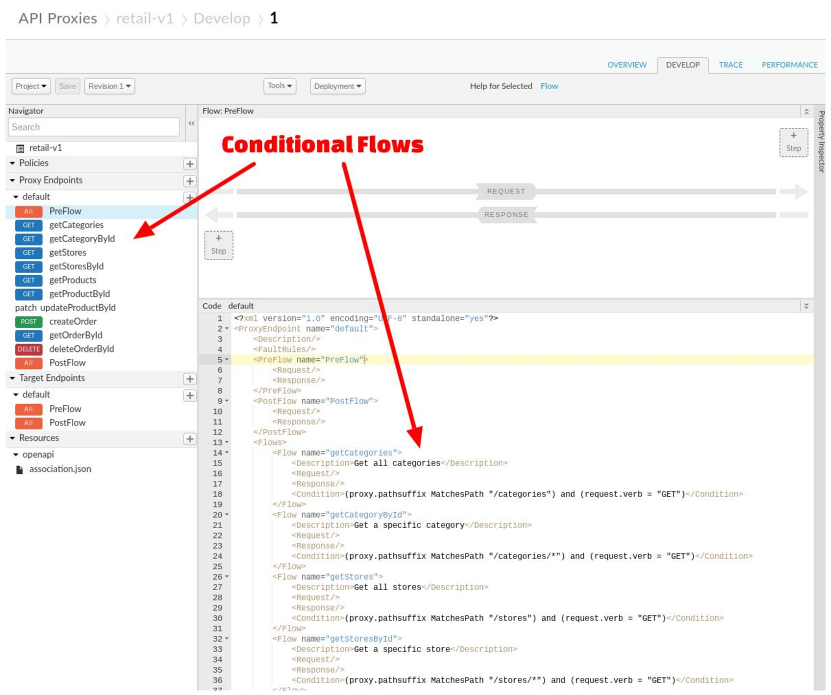
   Leave the Target unchanged.

6. Click **Next**.
7. Leave the Common Policies settings at their defaults. Click **Next**.
8. Operations that were found in the OpenAPI specification are listed. Confirm that all operations are selected, and click **Next**.
9. The secure virtual host should be selected, and the default virtual host should not be selected. Click **Next**.
10. On the Summary page, select the **test** environment, and then click **Create and deploy**.
    Your proxy will be generated and deployed.
11. When deployment is complete, click **Edit proxy**.

Explore and trace the API proxy

1. Click 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 a conditional flow, it appears in the Code pane on the right.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProxyEndpoint name="default">
    <Description/>
    <FaultRules/>
    <PreFlow name="PreFlow">
        <Request/>
        <Response/>
    </PreFlow>
    <PostFlow name="PostFlow">
        <Request/>
        <Response/>
    </PostFlow>
    <Flows>
        <Flow name="getCategories">
            <Description>Get all categories</Description>
            <Request/>
            <Response/>
            <Condition>(proxy.pathsuffix MatchesPath "/categories") and (request.verb = "GET")</Condition>
        </Flow>
        <Flow name="getCategoryById">
            <Description>Get a specific category</Description>
            <Request/>
            <Response/>
            <Condition>(proxy.pathsuffix MatchesPath "/categories/*") and (request.verb = "GET")</Condition>
        </Flow>
        <Flow name="getStores">
            <Description>Get all stores</Description>
            <Request/>
            <Response/>
            <Condition>(proxy.pathsuffix MatchesPath "/stores") and (request.verb = "GET")</Condition>
        </Flow>
        <Flow name="getStoresById">
            <Description>Get a specific store</Description>
            <Request/>
            <Response/>
            <Condition>(proxy.pathsuffix MatchesPath "/stores/*") and (request.verb = "GET")</Condition>
```

You will update these conditional flows in later labs.

2. Click the **Trace** tab.
3. **Deployment to Trace** should show the test environment. Click **Start Trace Session**.
4. To send a "get categories" request to your proxy, add `/categories` to the URL in the **Send Requests** pane.

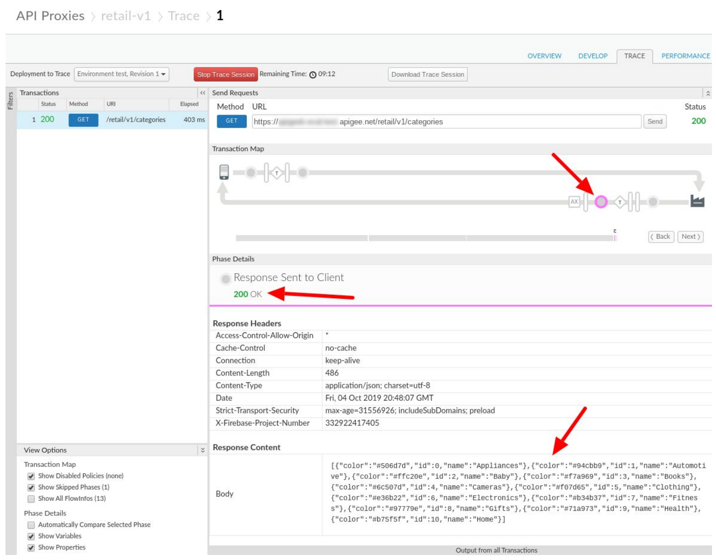
   Your URL should look similar to:

   ```
   https://myorg-eval-test.apigee.net/retail/v1/categories
   ```

   Instead of **myorg-eval**, your URL should include your org name, which can be found under your name in the upper left corner of the Apigee console.
5. Click **Send**.

   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 added **/categories** to the end of the Send Requests URL.

6. Click on the last circle as shown in the screenshot below. This shows the response that is being returned to the client.



Deployment to Trace: Environment test, Revision 1
Remaining Time: 09:12

Transactions

	Status	Method	URI	Elapsed
1	200	GET	/retail/v1/categories	403 ms

Send Requests

Method	URL	Status
GET	https://[redacted]apigee.net/retail/v1/categories	200

Phase Details

Response Sent to Client
200 OK

Response Headers

Access-Control-Allow-Origin	*
Cache-Control	no-cache
Connection	keep-alive
Content-Length	486
Content-Type	application/json; charset=utf-8
Date	Fri, 04 Oct 2019 20:48:07 GMT
Strict-Transport-Security	max-age=31556926; includeSubDomains; preload
X-Firebase-Project-Number	332922417405

Response Content

Body

```
[{"color":"#506d7d","id":0,"name":"Appliances"},{"color":"#94cbb9","id":1,"name":"Automoti
ve"},{"color":"#ffc20e","id":2,"name":"Baby"},{"color":"#f7a969","id":3,"name":"Books"},
{"color":"#6c507d","id":4,"name":"Cameras"},{"color":"#f07d65","id":5,"name":"Clothing"},
{"color":"#e36b22","id":6,"name":"Electronics"},{"color":"#b34b37","id":7,"name":"Fitnes
s"},{"color":"#97779e","id":8,"name":"Gifts"},{"color":"#71a973","id":9,"name":"Health"},
{"color":"#b75f5f","id":10,"name":"Home"}]
```

Output from all Transactions

View Options

Transaction Map
- [x] Show Disabled Policies (none)
- [x] Show Skipped Phases (1)
- [ ] Show All FlowInfos (13)

Phase Details
- [ ] Automatically Compare Selected Phase
- [x] Show Variables
- [x] Show Properties

The Send Requests pane is an easy way to create a GET request for your API proxy. However, you cannot add headers, change the verb in the request, or send a payload. You will need a REST client for later labs.

The Phase Details pane in your trace may look different from the screenshot, depending on the settings in the View Options pane in the lower left corner. The screenshots typically show the contents with the "Automatically Compare Selected Phase" box cleared. If you want your trace to more closely match screenshots in the labs, clear the "Automatically Compare Selected Phase" checkbox.

3. Modify the OpenAPI specification to reference your proxy

In this task, you modify the OpenAPI specification to reference your API proxy instead of the backend service.

Modify the OpenAPI specification

1. Keep the trace session open. In another tab, open the Apigee console.
2. On the left navigation menu, click **Develop > Specs**.
3. Click on the **retail-v1** specification.
4. Change the first 15 lines of the OpenAPI specification to look something like this:

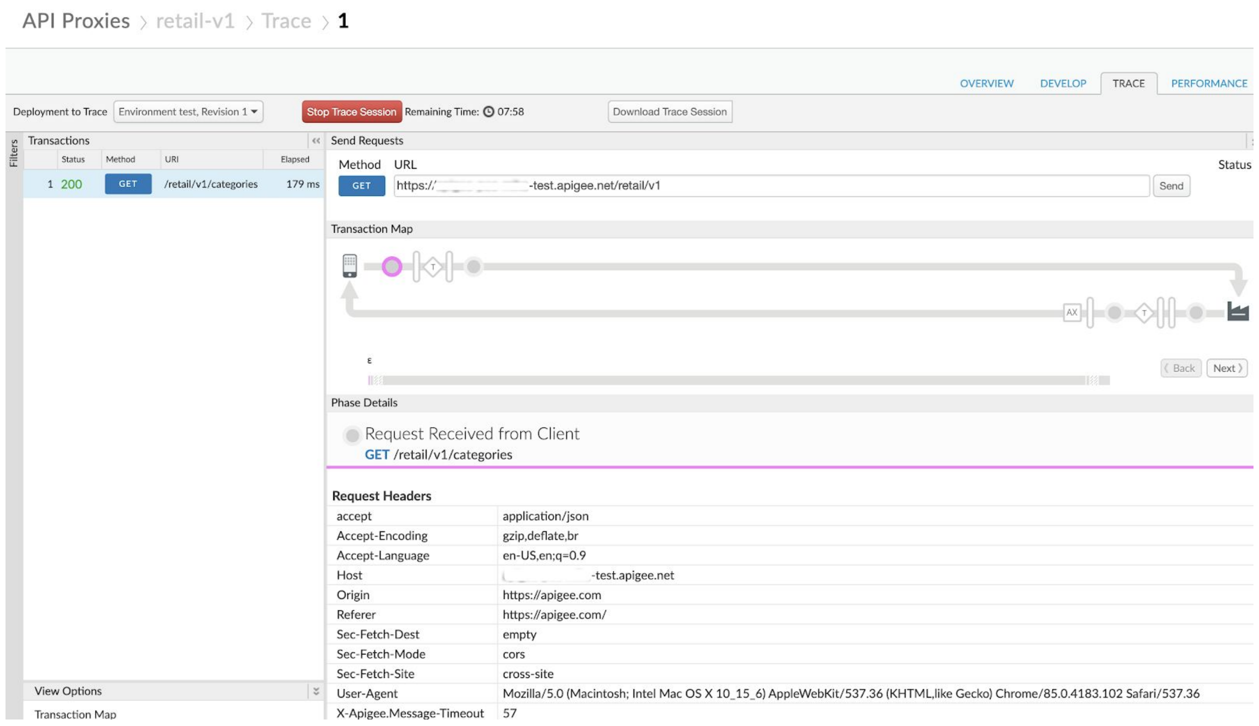
```
openapi: "3.0.0"
info:
version: 0.0.1
title: Retail API v1
description: Retail API
contact:
    name: Your Name
    email: youremail@example.org
    url: https://example.org
license:
    name: MIT
    url: https://opensource.org/licenses/MIT
servers:
- url: "https://YOURORG-test.apigee.net/retail/v1"
    description: Retail API v1
```

5. You must change the server URL to use your organization. Replace **YOURORG** with your organization name. Your org name typically ends with **-eval**. The URL should end with **/retail/v1**.
   If you have any problems, you can use the URL in the **Send Requests** pane of the trace tool. That URL, after you remove any resource like **/categories**, should match this URL.
6. You can optionally change the description and contact information.
7. Click **Save** to save the proxy.

Test the updated OpenAPI specification

1. Click **Start Trace Session** in the trace tab. If trace is already running, and the button is red, click the button to stop the trace, and click again to start the trace session.
2. On the right side of the Spec page, click the **GET /categories** box, and try the call by clicking **Try it out** and **Execute**. You should see the call in the trace tool.
   Click on the first circle in the trace to see the request. You may see headers that were automatically added by the browser, such as Origin,

User-Agent, and Referer.



	Status	Method	URI	Elapsed
1	200	GET	/retail/v1/categories	179 ms

Method	URL	Status
GET	https://[redacted]-test.apigee.net/retail/v1 Send	

Request Received from Client
GET /retail/v1/categories

Request Headers

accept	application/json
Accept-Encoding	gzip,deflate,br
Accept-Language	en-US,en;q=0.9
Host	[redacted]-test.apigee.net
Origin	https://apigee.com
Referer	https://apigee.com/
Sec-Fetch-Dest	empty
Sec-Fetch-Mode	cors
Sec-Fetch-Site	cross-site
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36 (KHTML,like Gecko) Chrome/85.0.4183.102 Safari/537.36
X-Apigee.Message-Timeout	57

If you receive a "TypeError: Failed to fetch" error when sending from the spec, your organization name might not have been specified correctly in the OpenAPI spec. Confirm the change you made to the first 15 lines of the OpenAPI spec, and confirm that your servers.url includes "/retail/v1" and your organization name, which includes "-eval" in most cases.