Using Shared Flows

experiment Lab schedule 1 hour 30 minutes universal_currency_alt No cost

show_chart Introductory

Overview

Shared flows can be used to encapsulate common tasks so they can be easily shared between proxies.

In this lab, you create a shared flow that adds the required Authorization header to a target request, and then call it from your retail proxy.

Objectives

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

- Create a shared flow.
- Call a shared flow from a proxy.

Setup and requirements

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.

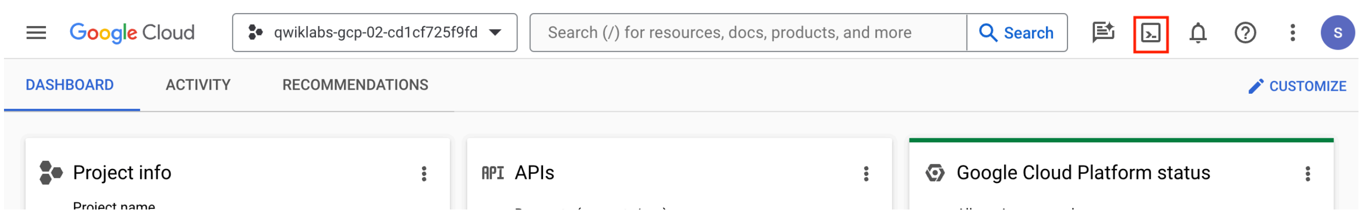
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

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.

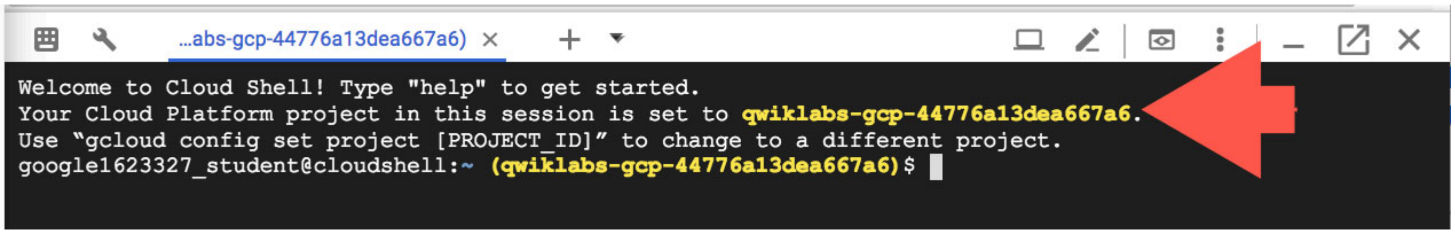
Google Cloud Shell provides command-line access to your Google Cloud resources.

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


Google Cloud
qwiklabs-gcp-02-cd1cf725f9fd
Search (/) for resources, docs, products, and more
Search
S
DASHBOARD
ACTIVITY
RECOMMENDATIONS
CUSTOMIZE
Project info
Project name
APIs
Google Cloud Platform status

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:



```
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:

```
gcloud config list project
```

content_c

Output:

```
[core]
project =
```

Example output:

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

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

Preloaded assets

These assets have already been added to the Apigee organization:

- The **retail-v1** API proxy
- The oauth-v1 API proxy (for generating OAuth tokens)
- The TS-Retail target server in the eval environment (used by retail-v1)

These assets will be added to the Apigee organization as soon as the runtime is available:

- The API products, developer, and **developer app** (used by retail-v1)
- The **ProductsKVM** key value map in the eval environment (currently used by retail-v1, and will be used by the shared flow that is created during this lab)
- The **ProductsKVM** key value map will be populated with backendId and backendSecret

The **highlighted** items are used during this lab.

Note: Revision 1 of the retail-v1 proxy is marked as deployed, and is immutable. If you ever make a mistake in your proxy code that you can't recover from, you can select revision 1 and restart editing from there.

Task 1. Create a new shared flow to build a basic auth header

In this task, you create a new shared flow that is used to build a basic auth header for calling your backend service.

Create the shared flow

1. In the Google Cloud console, on the **Navigation menu** (☰), select **Integration Services > Apigee > Proxy Development > Shared flows**.

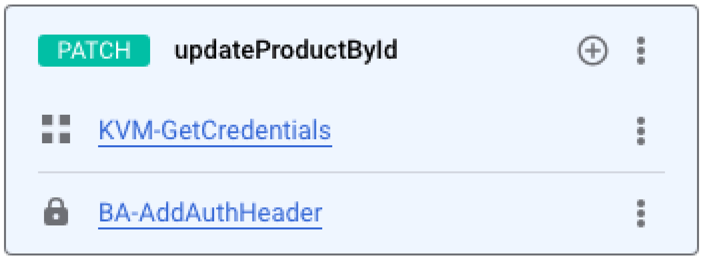
2. Click +**Create**.

3. For **Name**, enter `backend-credentials`.

4. Click **Create**.

Add the policies

The shared flow will be used to create the basic auth header created in the **updateProductById** flow of the **retail-v1** proxy:


PATCH updateProductById
KVM-GetCredentials
BA-AddAuthHeader

The policies will be identical to those used in the **retail-v1** proxy.

1. In the shared flow, click the **Develop** tab.

2. Click **Shared flows > default**.

You will add the key value map policy.

3. On the **default flow**, click **Add Policy Step (+)**.

4. In the **Add policy step** pane, select **Create new policy**, and then select **Mediation > Key Value Map Operations**.

5. Specify the following values:

Property	Value
Name	**KVM-GetCredentials**
Display name	**KVM-GetCredentials**

6. Click **Add**.

7. Click on **Policies > KVM-GetCredentials**.

8. Set the policy configuration to:

```
<KeyValueMapOperations continueOnError="false" enabled="true"
name="KVM-GetCredentials">
  <MapName>ProductsKVM</MapName>
  <ExpiryTimeInSecs>60</ExpiryTimeInSecs>
  <Get assignTo="private.backendId">
    <Key>
      <Parameter>backendId</Parameter>
    </Key>
  </Get>
  <Get assignTo="private.backendSecret">
    <Key>
      <Parameter>backendSecret</Parameter>
    </Key>
  </Get>
  <Scope>environment</Scope>
</KeyValueMapOperations>
```

content_c

This policy configuration is identical to the configuration of the KVM policy in the **retail-v1** proxy.

9. Click **Shared flows > default**.

   Now you will add the second key value map policy.

10. On the **Request updateProductById flow**, click **Add Policy Step (+)**.

11. In the **Add policy step** pane, select **Create new policy**, and then select **Security > Basic Authentication**.

12. Specify the following values:

Property	Value
Name	**BA-AddAuthHeader**
Display name	**BA-AddAuthHeader**

13. Click **Add**.

14. Click on **Policies > BA-AddAuthHeader**.

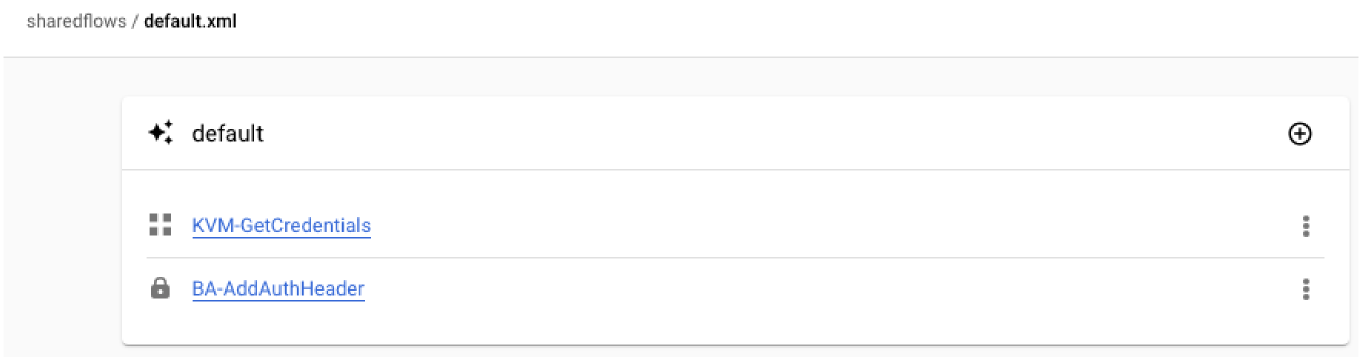
15. Set the policy configuration to:

content_c

```
<BasicAuthentication continueOnError="false" enabled="true"
name="BA-AddAuthHeader">
  <Operation>Encode</Operation>
  <IgnoreUnresolvedVariables>false</IgnoreUnresolvedVariables>
  <User ref="private.backendId"/>
  <Password ref="private.backendSecret"/>
  <AssignTo
createNew="false">request.header.Authorization</AssignTo>
</BasicAuthentication>
```

16. Click **Shared flows > default**.

    Your shared flow should look like this:



17. Click **Save** to save the shared flow.

18. Click **Deploy**.

    A shared flow cannot be tested without including it in an API proxy, and it must be deployed to an environment before a proxy calling it can be deployed to that environment.

19. To specify that you want the new revision deployed to the eval environment, select **eval** as the **Environment**, and then click **Deploy**.

20. Click **Confirm**.

Task 2. Replace the policies in retail-v1 with a flow callout policy

In this task, you change your retail-v1 proxy to call the shared flow, replacing the KVM and BasicAuthentication policies with a FlowCallout policy.

1. On the Navigation menu, click **Proxy development > API proxies**.

2. Navigate to **Develop > API Proxies**.

3. Select the **retail-v1** proxy.

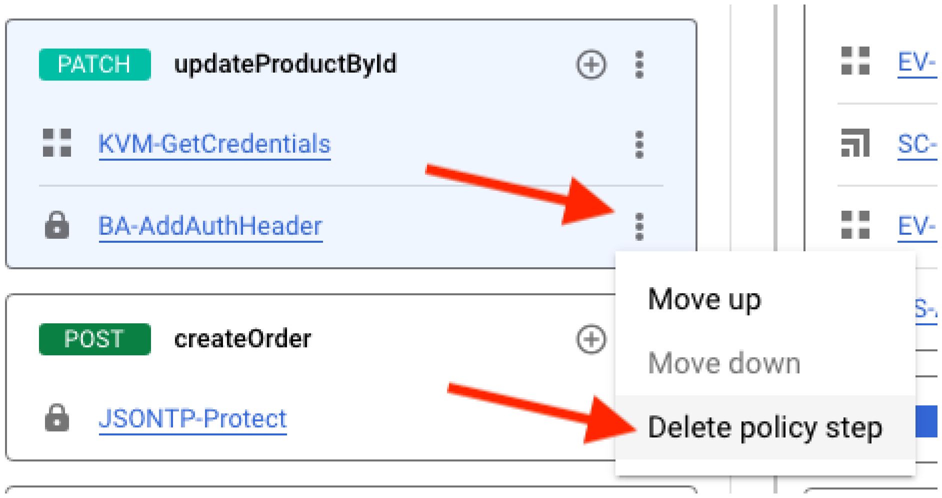
4. Click the **Develop** tab.

   You are modifying the version of the retail-v1 proxy that was created during Labs 1 through 8.

Detach the existing policies

1. Click **Proxy endpoints > default > updateProductById**.

2. To detach the **KVM-GetCredentials** and **BA-AddAuthHeader** policies, for each policy, click **Policy step actions** (⋮), and then click **Delete policy step**.



Add the flow callout policy

1. In the **updateProductById request flow**, click **Add Policy Step (+)**.

2. In the **Add policy step** pane, select **Create new policy**, and then select **Extension > Flow Callout**.

3. Specify the following values:

Property	**Value**
Name	**FC-BackendCredentials**
Display name	**FC-BackendCredentials**
Sharedflow	*select* **backend-credentials**

4. Click **Add**.

   The policy configuration simply specifies the shared flow to call in the **SharedFlowBundle** element. The policy configuration does not need to be changed.

5. Click **Save**, and then click **Save as New Revision**.

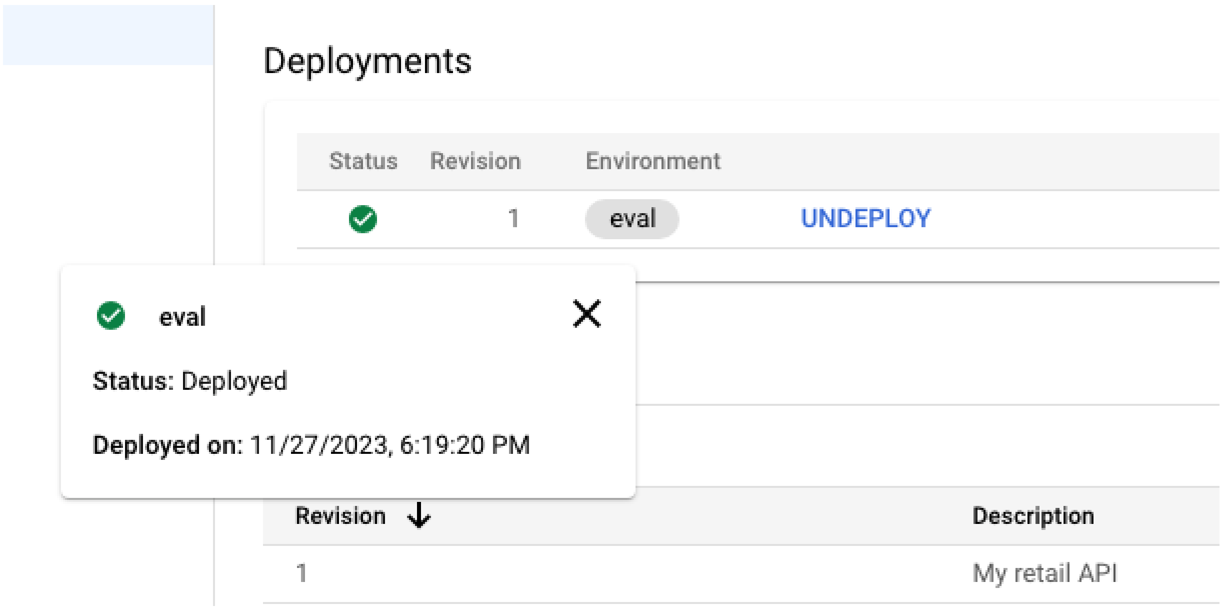
6. Click **Deploy**.

7. To specify that you want the new revision deployed to the eval environment, click **Deploy**.

8. Click **Confirm**.

Check deployment status

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



Status	Revision	Environment
✓	1	eval

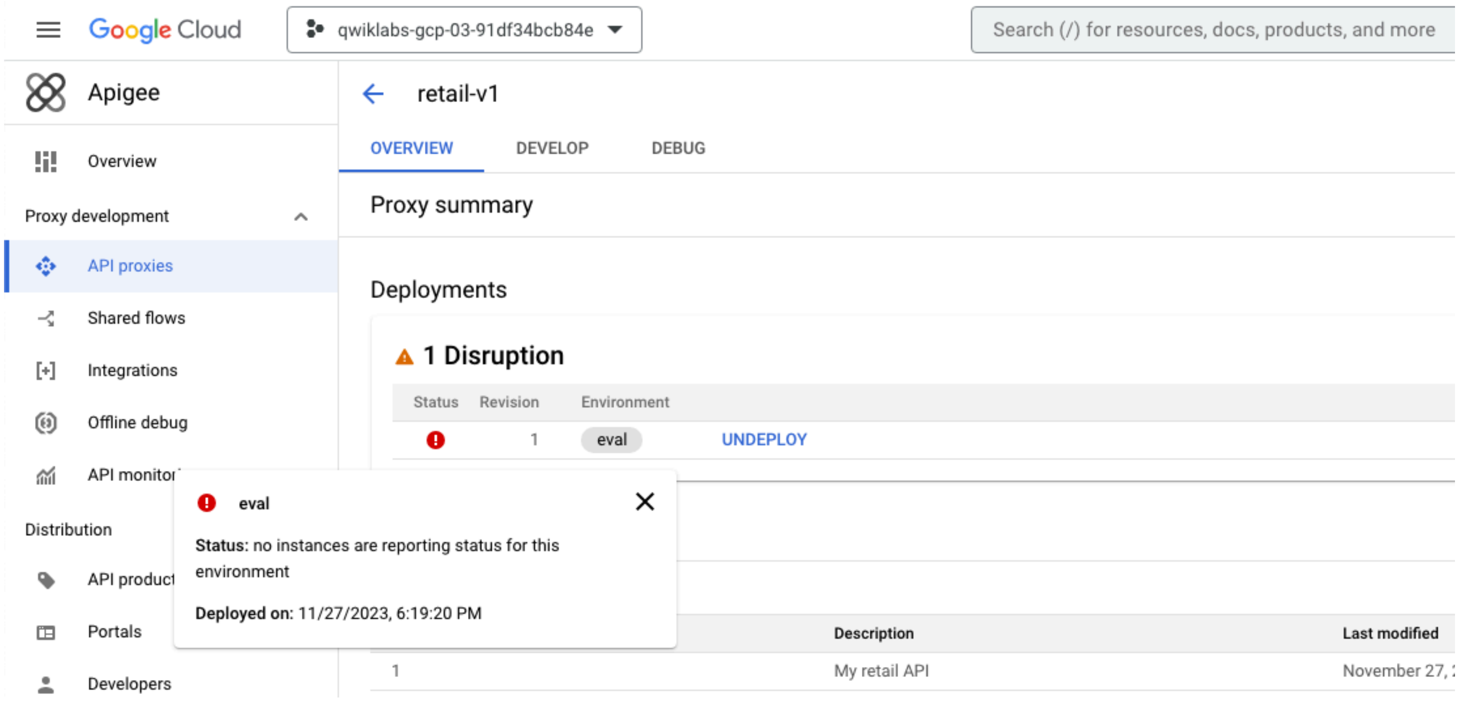
eval

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.



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

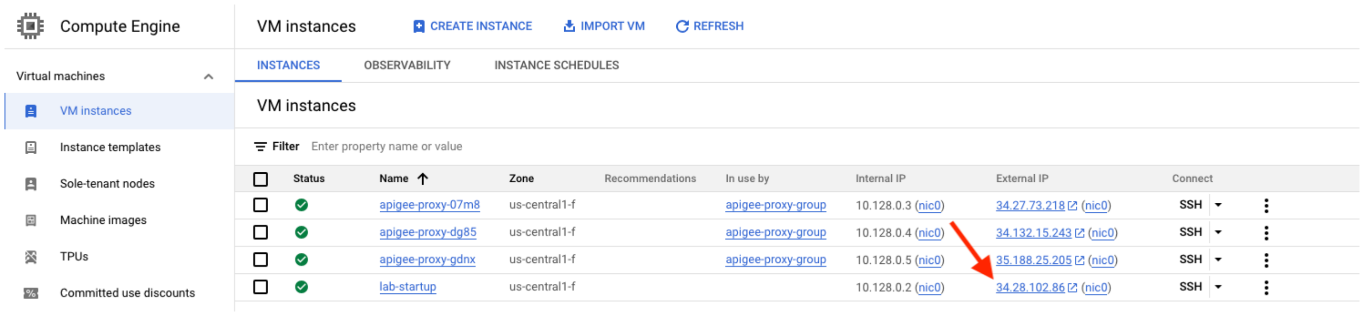
	Description	Last modified
1	My retail API	November 27,

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.



Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	apigee-proxy-07m8	us-central1-f		apigee-proxy-group	10.128.0.3 (nic0)	34.27.73.218 (nic0)	SSH
✓	apigee-proxy-dg85	us-central1-f		apigee-proxy-group	10.128.0.4 (nic0)	34.132.15.243 (nic0)	SSH
✓	apigee-proxy-gdnx	us-central1-f		apigee-proxy-group	10.128.0.5 (nic0)	35.188.25.205 (nic0)	SSH
✓	lab-startup	us-central1-f			10.128.0.2 (nic0)	34.28.102.86 (nic0)	SSH

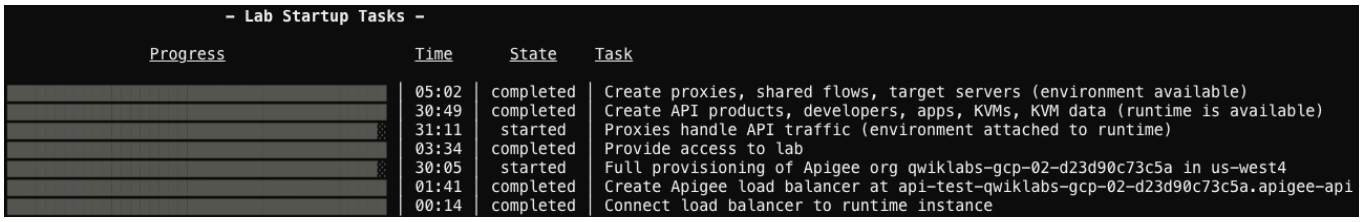
3. If you see a redirect notice page, click the link to the external IP address.

   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

                                   | 05:02 | completed | Create proxies, shared flows, target servers (environment available)
                                   | 30:49 | completed | Create API products, developers, apps, KVMs, KVM data (runtime is available)
                                   | 31:11 |  started  | Proxies handle API traffic (environment attached to runtime)
                                   | 03:34 | completed | Provide access to lab
                                   | 30:05 |  started  | Full provisioning of Apigee org qwiklabs-gcp-02-d23d90c73c5a in us-west4
                                   | 01:41 | completed | Create Apigee load balancer at api-test-qwiklabs-gcp-02-d23d90c73c5a.apigee-api
                                   | 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

Read:

- Creating reusable shared flows
- FlowCallout policy
- Using flow hooks for shared flows

Task 3. Test the retail API

In this task, you validate that the retail API and shared flow present the credentials to the backend service and the product overall rating is updated.

Store the app's key in a shell variable

The API key may be retrieved directly from the app accessible on the **Publish > Apps** page. It can also be retrieved via Apigee API call.

- In **Cloud Shell**, run the following command:

```
export API_KEY=$(curl -q -s -H "Authorization: Bearer $(gcloud auth
print-access-token)" -X GET
"https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PROJECT
app" | jq --raw-output '.credentials[0].consumerKey'); echo "export
API_KEY=${API_KEY}" >> ~/.profile; echo "API_KEY=${API_KEY}"
```

content_c

This command retrieves a Google Cloud access token for the logged-in user, sending it as a Bearer token to the Apigee API call. It retrieves the **retail-app** app details as a JSON response, which is parsed by **jq** to retrieve the app's key. That key is then put into the **API_KEY** environment variable, and the export command is concatenated onto the **.profile** file which runs automatically when starting a Cloud Shell tab.

> **Note:** If you run the command and it shows API_KEY=null, the runtime instance is probably not yet available.

Get the list of products

1. Use this curl command to get a list of products:

```
curl -H "apikey: ${API_KEY}" -X GET "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/products"
| json_pp
```

content_c

The response should be a JSON list of products that resembles this:

```
{
  "00621094000P" : {
     "category" : "Fitness",
     "image" : "https://cdn.pixabay.com/photo/2016/04/01/09/30/boy-1299405_
```

```
    "name" : "00621094000P",
    "overall_rating" : 0.2,
    "product_name" : "AFG 7.1AT Treadmill",
    "short_description" : "The right treadmill makes a difference, especia
  },
  "00624932000P" : {
    "category" : "Fitness",
    "image" : "https://cdn.pixabay.com/photo/2016/04/01/09/30/boy-1299405_
    "name" : "00624932000P",
    "overall_rating" : 4.3,
    "product_name" : "NordicTrack Elite 3700 Treadmill",
    "short_description" : "Amp it Up with the NordicTrack Elite 3700 Inter
  },
```

The top-level keys are the IDs (00621094000P and 00624932000P are shown here). Choose any one of the IDs in the entire list.

2. Create an environment variable, replacing `REPLACE` with the chosen ID:

```
export PRODUCT_ID=REPLACE
```

content_c

3. Look at the current **overall_rating** for the product, and choose a different positive decimal number. For example, 2.1 is the overall_rating for product 31001 shown above. You could change the rating to 4.5. Create an environment variable, replacing `REPLACE` with the chosen rating:

```
export NEW_RATING=REPLACE
```

content_c

4. Use this command to update the overall_rating, and then retrieve the product to make sure that the overall_rating has changed:

```
curl -H "apikey: ${API_KEY}" -X PATCH "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/products/${P
-H "Content-Type: application/json" -d "{ \"overall_rating\":
${NEW_RATING} }" | json_pp
curl -H "apikey: ${API_KEY}" -X GET "https://api-
```

content_c

```
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/products/${P
| json_pp
```

The first curl command will return the same overall_rating that you used to update it. The second curl command will return the entire product, including the updated overall_rating.

The shared flow has the same functionality in the API proxy.

Congratulations!