Apigee Lab 8: Mashing Up Services

experiment Lab schedule 1 hour 30 minutes universal_currency_alt No cost

show_chart Introductory

Overview

Some API resource flows require calling more than one service and combining the responses into a single response. This is often called "mashing up" service calls.

In this lab, you call a third-party service, extract data, and add the data to the target's API response.

Objectives

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

- Use a ServiceCallout policy to call a service.
- Use custom JavaScript code to combine API responses.

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.

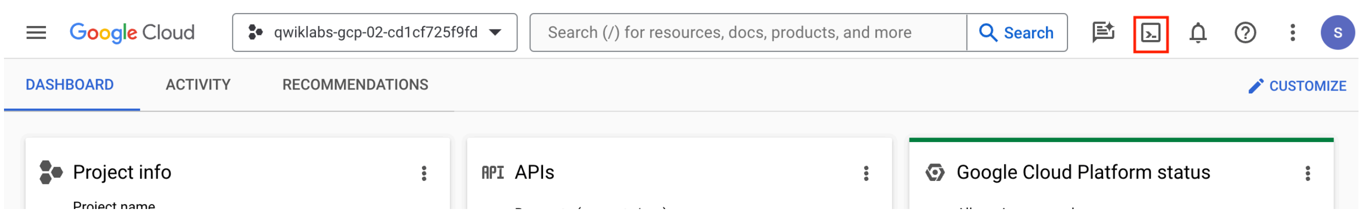
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

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.

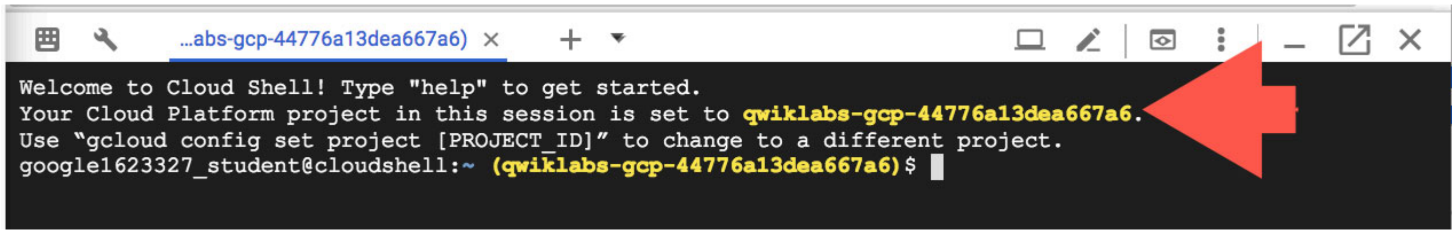
Google Cloud Shell provides command-line access to your Google Cloud resources.

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


Google Cloud
qwiklabs-gcp-02-cd1cf725f9fd
Search (/) for resources, docs, products, and more
Search
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 (used by retail-v1)
- The ProductsKVM key value map entries 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. Call a Google geocoding API

In this task, you use a ServiceCallout policy to call a Google Maps geocoding API to get a street address by latitude and longitude.

Understanding the stores resource

A list of stores may be retrieved by using **GET /stores**.

The beginning of the **GET /stores** response is:

```
{
    "benbrook": {
        "affiliate": "Walsample",
        "location": {
            "latitude": 32.677649,
            "longitude": -97.465539
        },
        "name": "Bensample",
        "phone": "682-233-6820",
        "site_type": "store"
    },
    "central-warehouse-1": {
        "affiliate": "Walsample",
        "location": {
            "latitude": 36.315127,
            "longitude": -94.12623
        },
        "name": "Central Warehouse-1",
        "phone": "479-633-0769",
        "site_type": "warehouse"
    },
    "mall": {
```

The first store has the ID **benbrook**. `GET /stores/benbrook` returns:

```
{
    "affiliate": "Walsample",
    "location": {
        "latitude": 32.677649,
        "longitude": -97.465539
    },
    "name": "Bensample",
    "phone": "682-233-6820",
    "site_type": "store"
}
```

This request is handled by the **getStoreById** conditional flow. The latitude and longitude must be extracted from the location object.

You will modify the **GET /stores/{storeId}** API call to add the street address corresponding to the store's latitude and longitude.

Extract the latitude and longitude from backend response

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

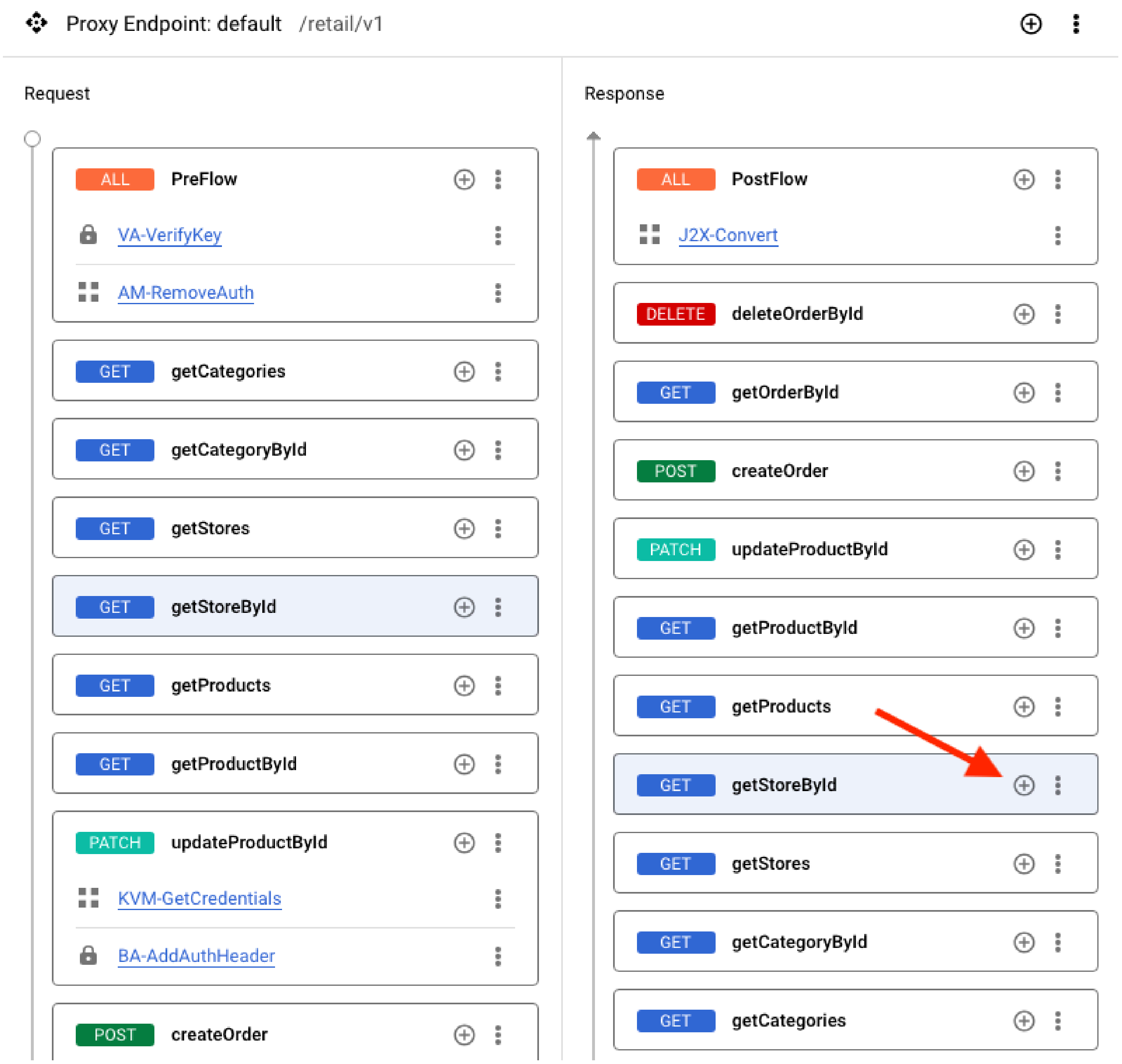
2. Select the **retail-v1** proxy.

3. Click the **Develop** tab.

   You are modifying the version of the retail-v1 proxy that was created during Labs 1 through 7.

4. In the Navigator menu, click **Proxy endpoints > default > getStoreById**.

5. On the **Response getStoreById flow**, click **Add Policy Step (+)**.



Request:

- PreFlow
  - VA-VerifyKey
  - AM-RemoveAuth
- getCategories
- getCategoryById
- getStores
- getStoreById
- getProducts
- getProductById
- updateProductById
  - KVM-GetCredentials
  - BA-AddAuthHeader
- createOrder

Response:

- PostFlow
  - J2X-Convert
- deleteOrderById
- getOrderById
- createOrder
- updateProductById
- getProductById
- getProducts
- getStoreById
- getStores
- getCategoryById
- getCategories

6. In the **Add policy step** pane, select **Create new policy**, and then select **Mediation > Extract Variables**.

7. Specify the following values:

Property	**Value**
Name	**EV-ExtractLatLng**
Display name	**EV-ExtractLatLng**

8. Click **Add**.

9. Click **Policies > EV-ExtractLatLng**.

10. Set the policy configuration to extract the latitude and longitude into variables:

content_c

```
<ExtractVariables continueOnError="false" enabled="true"
name="EV-ExtractLatLng">
  <JSONPayload>
    <Variable name="lat">
      <JSONPath>$.location.latitude</JSONPath>
    </Variable>
    <Variable name="lng">
      <JSONPath>$.location.longitude</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">response</Source>
</ExtractVariables>
```

This uses JSONPath to extract the latitude and longitude from the location object and store them in the variables **lat** and **lng**.

Call the geocoding service

The service callout calls the Google Geocoding API, passing the latitude and longitude variables to the API.

Property	**Value**
Geocoding API Key	**API-ENG-TRAINING-GEOCODING-KEY**
Geocoding API URL	**https://gcp-cs-training-01-prod.apigee.net/googleapis/maps/api/geocode/json**

Note: This lab will call a proxied version of the Google Maps Geocoding API that only supports the latitudes and longitudes for the stores. It uses an API key that cannot be used directly with the Geocoding API.

If you want to create a Google Maps API key of your own (which requires a billing account) follow the directions on the Get Started documentation. You could then replace the key and the URL to call the Google Maps API directly.

1. Open a new browser tab and make a call directly to the API, using this URL for the **benbrook** location:

```
https://gcp-cs-training-01-prod.apigee.net/googleapis/maps/api/geoc
```

content_c

The **latlng** query parameter contains the latitude and longitude, separated by a comma. The API key is passed in the **key** query parameter.

The response field **formatted_address** will be accessed using JSONPath. The first entry of the results array will be used. The JSONPath location is:

```
$.results[0].formatted_address
```

content_c

2. Click **Proxy endpoints > default > getStoreById**.

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

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

5. Specify the following values:

Property	**Value**
Name	**SC-GoogleGeocode**
Display name	**SC-GoogleGeocode**

   Leave the **HTTP Target** unchanged.

6. Click **Add**.

7. Click **Policies > SC-GoogleGeocode**.

   The new policy should follow the **EV-ExtractLatLng** policy, which means that it will be executed after the ExtractVariables policy.

8. Change the configuration for the **SC-GoogleGeocode** policy to call the same location you called in the browser:

content_c

```
<ServiceCallout continueOnError="false" enabled="true" name="SC-
GoogleGeocode">
  <Request>
    <Set>
      <QueryParams>
        <QueryParam name="latlng">{lat},{lng}</QueryParam>
        <QueryParam name="key">API-ENG-TRAINING-GEOCODING-
KEY</QueryParam>
      </QueryParams>
      <Verb>GET</Verb>
    </Set>
  </Request>
  <Response>calloutResponse</Response>
  <HTTPTargetConnection>
    <URL>https://gcp-cs-training-01-
prod.apigee.net/googleapis/maps/api/geocode/json</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

The response from the API call is stored in an object named **calloutResponse**.

Extract the formatted address

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

2. On the **Response getStoreById flow**, click **Add Policy Step (+)**.

3. In the **Add policy step** pane, select **Create new policy**, and then select **Mediation > Extract Variables**.

4. Specify the following values:

Property	**Value**
Name	**EV-ExtractAddress**
Display name	**EV-ExtractAddress**

5. Click **Add**.

6. Click **Policies** > **EV-ExtractAddress**.

7. Change the configuration for **EV-ExtractAddress**:

   content_c

   ```
   <ExtractVariables continueOnError="false" enabled="true"
   name="EV-ExtractAddress">
     <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
     <JSONPayload>
       <Variable name="address">
         <JSONPath>$.results[0].formatted_address</JSONPath>
       </Variable>
     </JSONPayload>
     <Source clearPayload="false">calloutResponse.content</Source>
   </ExtractVariables>
   ```

   The response payload from the geocoding API call is the source, and the address from the response is stored in a variable named **address**.

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

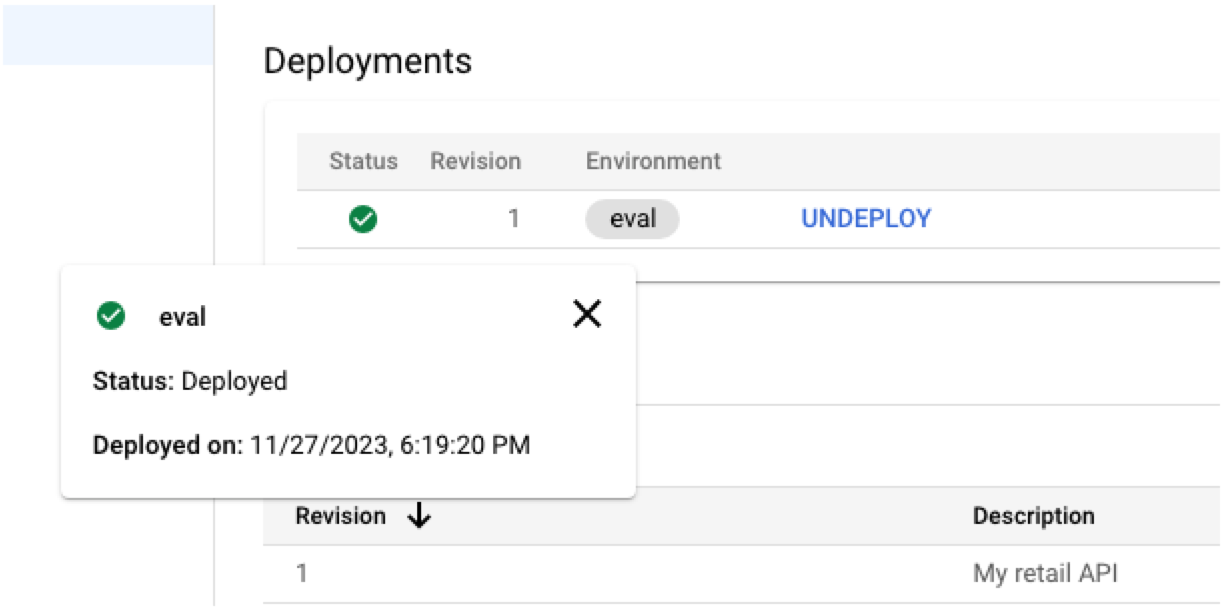
9. Click **Deploy**.

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

11. Click **Confirm**.

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

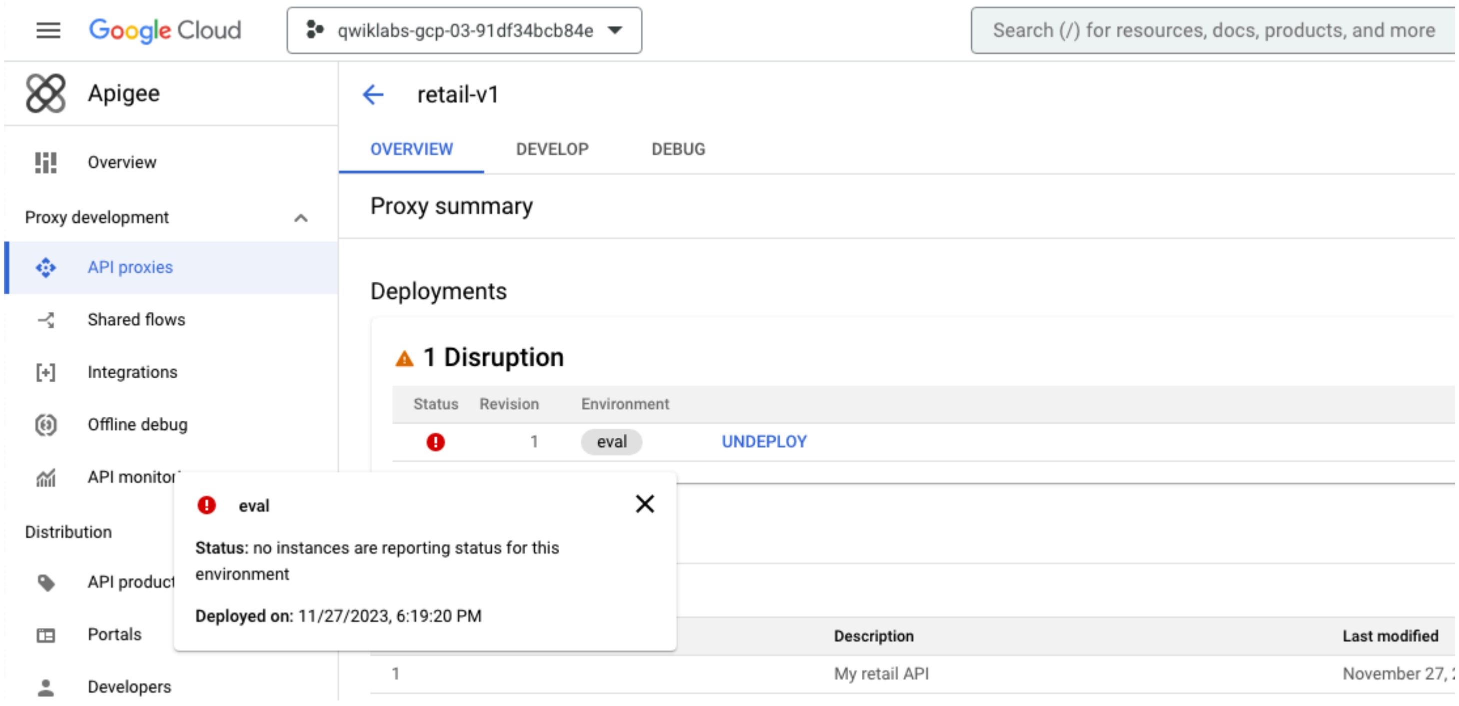
eval

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.



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

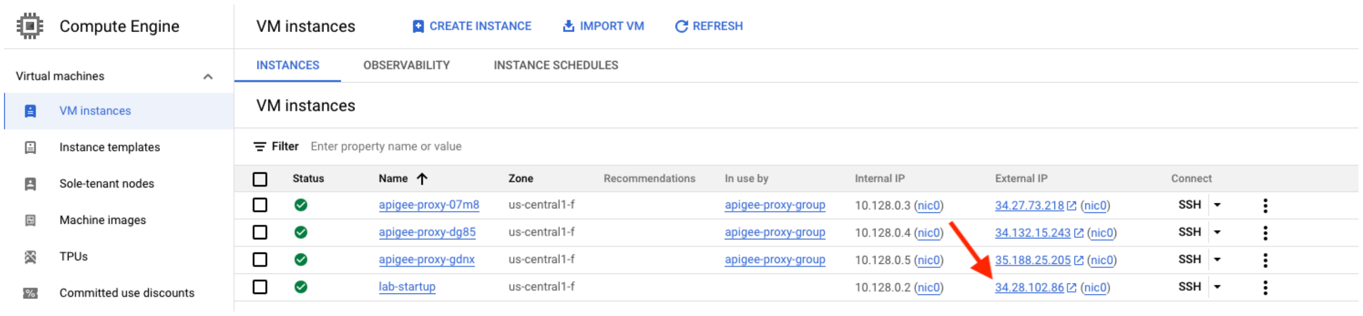
	Description	Last modified
1	My retail API	November 27,

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.

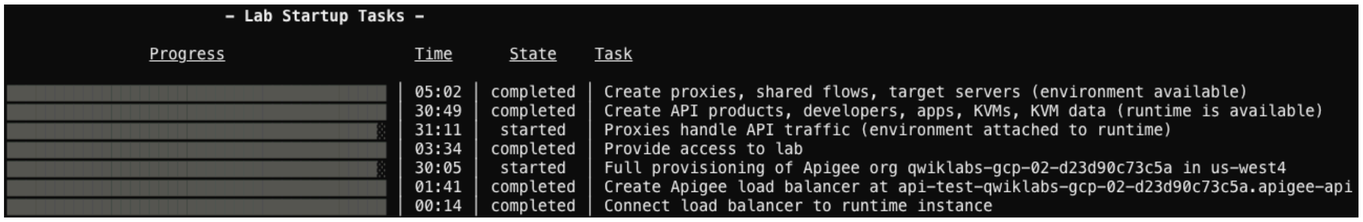


Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	apigee-proxy-07m8	us-central1-f		apigee-proxy-group	10.128.0.3 (nic0)	34.27.73.218 (nic0)	SSH
✓	apigee-proxy-dg85	us-central1-f		apigee-proxy-group	10.128.0.4 (nic0)	34.132.15.243 (nic0)	SSH
✓	apigee-proxy-gdnx	us-central1-f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

While you wait, explore the following:

- ServiceCallout policy
- JavaScript policy
- Google Geocoding Service
- JSONPath syntax

- JSONPath Online Evaluator

Task 2. Verify that the address variable is populated

In this task, you use the debug tool to verify that the formatted address is retrieved from the geocoding API and extracted into the address variable.

1. Click the **Debug** tab, and then click **Start Debug Session**.

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

3. Click **Start**.

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.

Debug the API

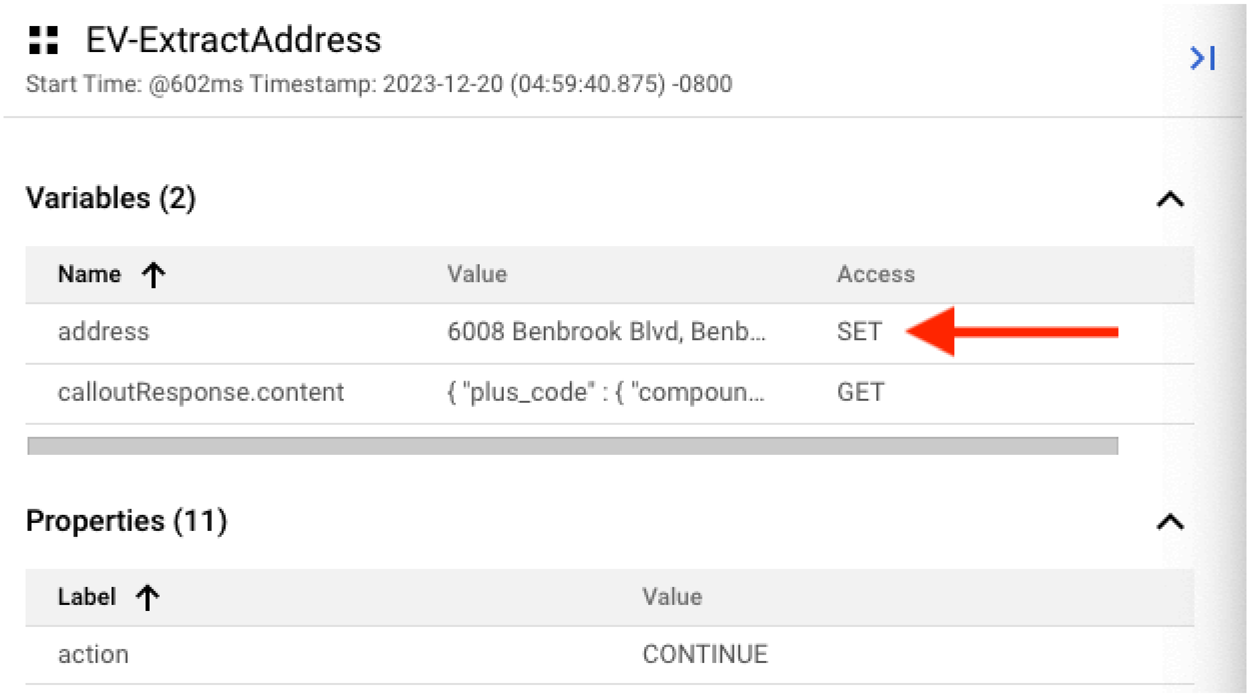
1. To get the store by ID, run this curl command in Cloud Shell:

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

content_c

2. In **Debug**, click on the GET request, and then click **EV-ExtractAddress**.

   If you have successfully extracted the latitude and longitude, called the geocoding API, and extracted the formatted address, the address variable will be populated.



EV-ExtractAddress

Start Time: @602ms Timestamp: 2023-12-20 (04:59:40.875) -0800

Variables (2)

Name	Value	Access
address	6008 Benbrook Blvd, Benb...	SET
calloutResponse.content	{ "plus_code" : { "compoun...	GET

Properties (11)

Label	Value
action	CONTINUE

Task 3. Add the address to the target response

In this task, you use a JavaScript policy to insert the address into the target response.

Begin by adding a **JavaScript** policy that is executed after **EV-ExtractAddress** in the **getStoreById** flow.

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

2. On the **Response getStoreById flow**, click **Add Policy Step (+)**.

3. In the **Add policy step** pane, select **Create new policy**, and then select **Extension > Javascript**.

4. Specify the following values:

Property	**Value**
Name	**JS-AddAddress**
Display name	**JS-AddAddress**

5. For **Javascript file**, select **Create New Resource**.

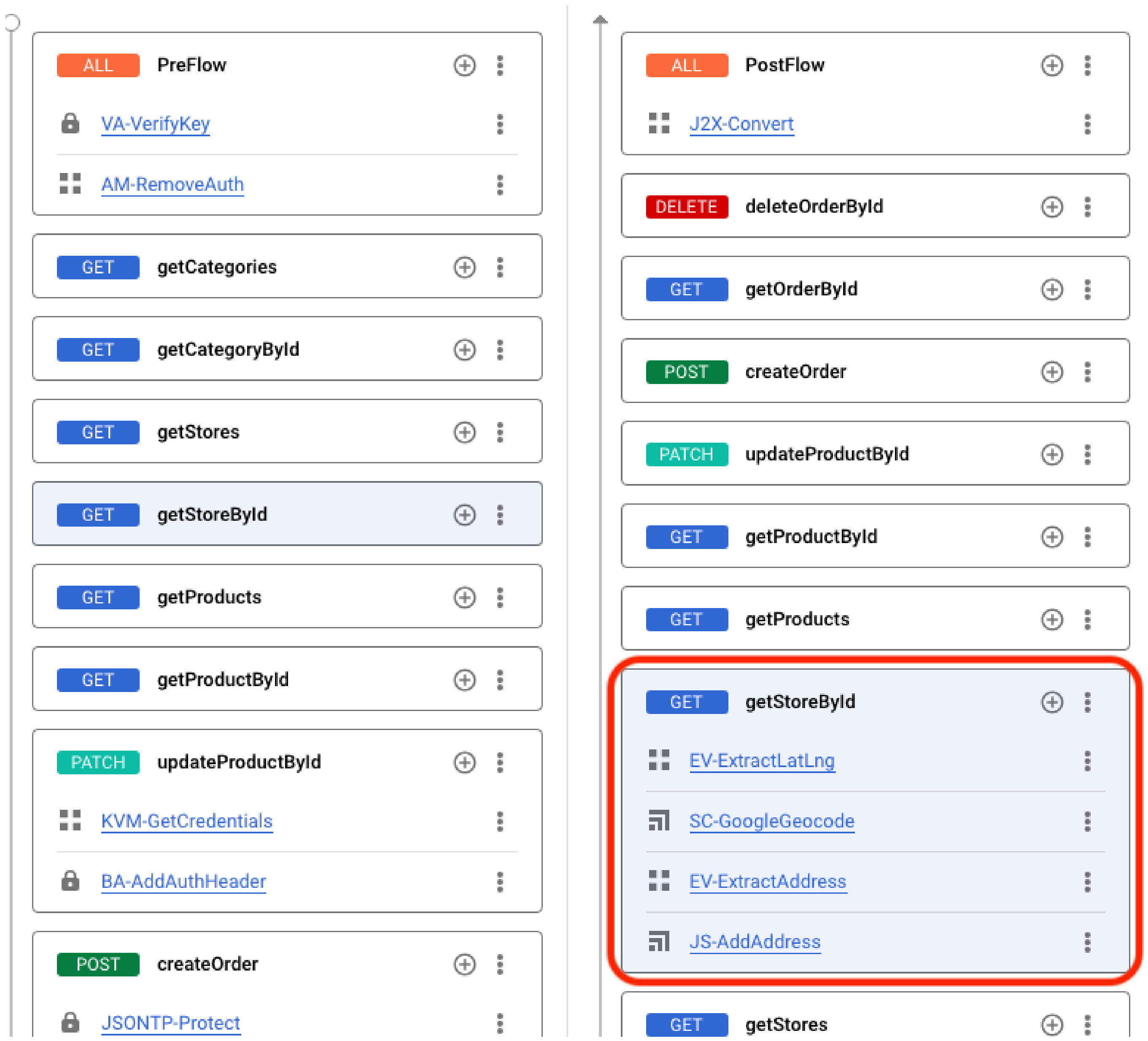
6. For **Resource name**, specify `addAddress.js`.

7. Click **Add**.

8. For **Javascript file**, select **addAddress.js**.

9. Click **Add**.

   The four policies in the **getStoreById** response flow should look like this:


ALL PreFlow
VA-VerifyKey
AM-RemoveAuth
GET getCategories
GET getCategoryById
GET getStores
GET getStoreById
GET getProducts
GET getProductById
PATCH updateProductById
KVM-GetCredentials
BA-AddAuthHeader
POST createOrder
JSONTP-Protect

ALL PostFlow
J2X-Convert
DELETE deleteOrderById
GET getOrderById
POST createOrder
PATCH updateProductById
GET getProductById
GET getProducts
GET getStoreById
EV-ExtractLatLng
SC-GoogleGeocode
EV-ExtractAddress
JS-AddAddress
GET getStores

The JavaScript policy configuration does not need to be changed.

10. In the Navigator pane, click **Resources** > **jsc** > **addAddress.js**.

    The code pane contains the addAddress.js Javascript code. It is currently empty.

11. To combine the responses, use this Javascript code:

```
// get the flow variable 'address'
var address = context.getVariable('address');

// parse the response payload into the responsePayload object
var responsePayload =
```

content_c

```
JSON.parse(context.getVariable('response.content'));
try {
  // add address to the response
  responsePayload.address = address;

  // convert the response object back into JSON
  context.setVariable('response.content',
JSON.stringify(responsePayload));

  context.setVariable('mashupAddressSuccess', true);

} catch(e) {
  // catch any exception
  print('Error occurred when trying to add the address to the
response.');
  context.setVariable('mashupAddressSuccess', false);
}
```

This code parses the JSON response payload into an object, adds an address field to the object, and converts the object back into a JSON string, storing it in the response.

try/catch is used so that an exception isn't thrown out of the JavaScript policy. A fault is raised if an exception isn't caught.

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

13. Click **Deploy**.

14. To specify that you want the new revision deployed to the eval environment, click **Deploy**, and then click **Confirm**.

Task 4. Verify that the address is returned in the response

In this task, you verify that the API call returns the address in the response payload.

- Get the store by ID:

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

content_cc

The response now should include the address:

```
{
   "address" : "6008 Benbrook Blvd, Benbrook, TX 76126, USA",
   "affiliate" : "Walsample",
   "location" : {
      "latitude" : 32.677649,
      "longitude" : -97.465539
   },
   "name" : "Bensample",
   "phone" : "682-233-6820",
   "site_type" : "store"
}
```

If the address field is not in the response, use the debug tool to determine what is not working.

Congratulations!