Overview

In this lab, you use the ResponseCache policy to cache entire responses and the LookupCache and PopulateCache policies to cache data.

Objectives

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

- Use the LookupCache and PopulateCache policies to cache data.
- Use the ResponseCache policy to cache entire 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.

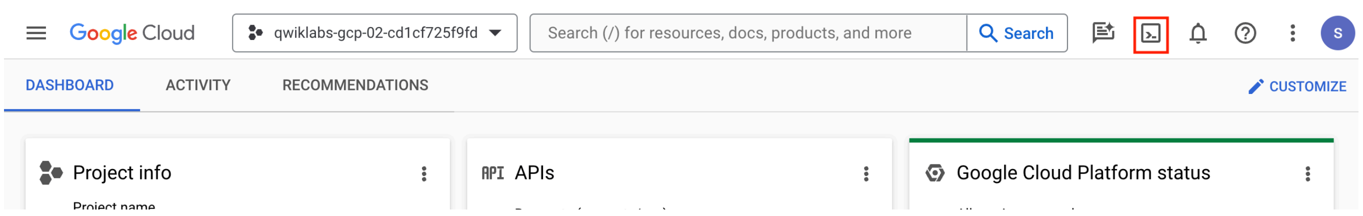
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

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.

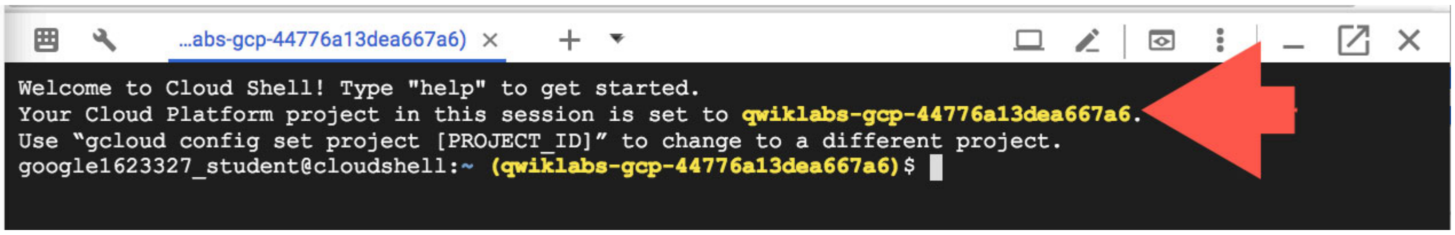
Google Cloud Shell provides command-line access to your Google Cloud resources.

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
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_co

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_co

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 backend-credentials shared flow (used by retail-v1)
- 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 backend-credentials)
- 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. Add a ResponseCache policy

In this task, you add a ResponseCache policy to cache **getProductById** responses.

Product catalogs are often a good use case for response caching, because the information is fairly static.

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 11.

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

5. On the **Request getProductById flow**, click **Add Policy Step (+)**.

6. In the **Add policy step** pane, select **Create new policy**, and then select **Traffic Management > Response Cache**.

7. Specify the following values:

Property	Value
Name	**RC-ProductsCache**
Display name	**RC-ProductsCache**

8. Click **Add**.

9. Click **Policies > RC-ProductsCache**.

10. Replace the ResponseCache configuration with:

```
<ResponseCache continueOnError="false" enabled="true" name="RC-
ProductsCache">
  <CacheResource>ProductsCache</CacheResource>
```

content_c

```
    <Scope>Exclusive</Scope>
    <CacheKey>
      <KeyFragment ref="proxy.pathsuffix" type="string"/>
    </CacheKey>
    <ExpirySettings>
      <TimeoutInSec>600</TimeoutInSec>
    </ExpirySettings>
    <SkipCacheLookup>request.verb != "GET"</SkipCacheLookup>
    <SkipCachePopulation>request.verb != "GET"
  </SkipCachePopulation>
  </ResponseCache>
```

There is a single key fragment, the **proxy.pathsuffix** variable. This variable includes the product ID. Typically, more cache key fragments will be necessary to ensure that the correct response is returned for a specific request.

The **TimeoutInSec** element overrides the expiration time for the **ProductsCache** cache by specifying 600 seconds, which is equal to 10 minutes. Responses that are cached will remain in the cache for 10 minutes.

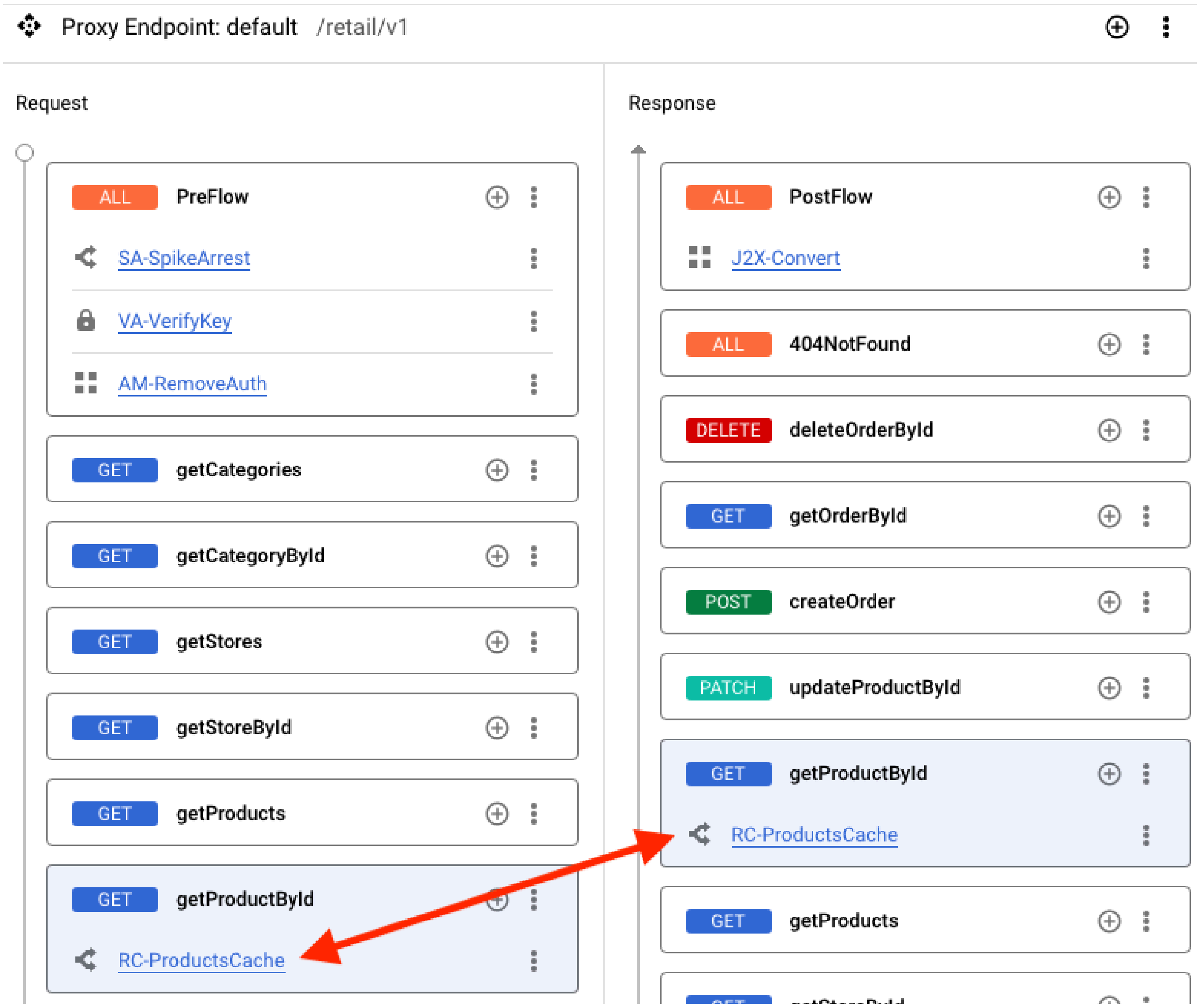
SkipCacheLookup and **SkipCachePopulation** are both set to skip when the request is not a **GET**. This is a best practice.

11. In the Navigator menu, click **Proxy endpoints > default > getProductById**.

12. On the **Response getProductById flow**, click **Add Policy Step (+)**.

13. In the **Add policy step** pane, for **Select existing policy**, select **Traffic Management > RC-ProductsCache**, and then click **Add**.

    The policy is now attached to both the Request and Response for getProductById. The policy must be attached twice: once in the request flow, and once in the response flow.



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

Task 2. Add LookupCache and PopulateCache policies

In this task, you use the LookupCache and PopulateCache policies to cache data from an external service.

1. In the Navigator menu, 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 **Traffic Management > Lookup Cache**.

4. Specify the following values:

Property	Value
Name	**LC-LookupAddress**
Display name	**LC-LookupAddress**

5. Click **Add**.

6. Click **Policies > LC-LookupAddress**.

7. Change the LookupCache configuration to:

content_c

```
<LookupCache continueOnError="false" enabled="true" name="LC-
LookupAddress">
  <CacheResource>AddressesCache</CacheResource>
  <Scope>Exclusive</Scope>
  <CacheKey>
    <KeyFragment ref="lat"/>
    <KeyFragment ref="lng"/>
  </CacheKey>
  <AssignTo>address</AssignTo>
</LookupCache>
```

This policy looks in the cache for an entry matching the latitude and longitude and assigns it to the variable address.

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

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

10. In the **Add policy step** pane, select **Create new policy**, and then select **Traffic Management > Populate Cache**.

11. Specify the following values:

Property	Value
Name	**PC-PopulateAddress**
Display name	**PC-PopulateAddress**
Condition	**lookupcache.LC-LookupAddress.cachehit == false**

The condition skips cache population if the address was found in the cache.

12. Click **Add**.

13. Click **Policies** > **PC-PopulateAddress**.

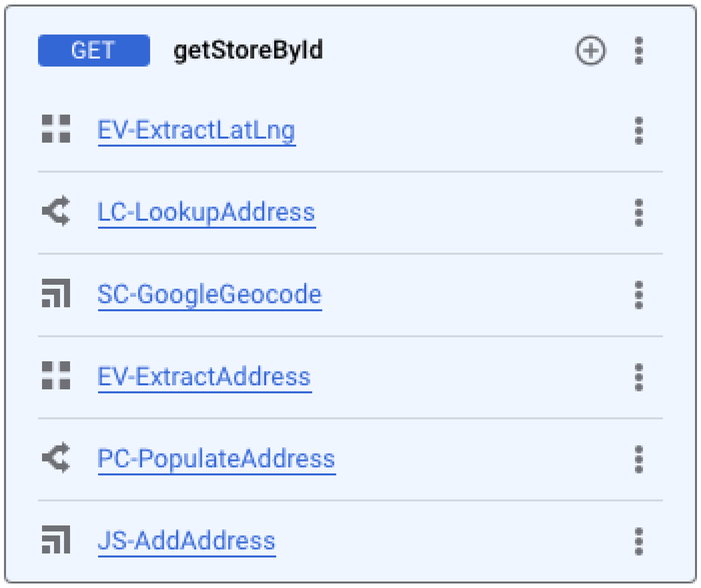
14. Change the PopulateCache configuration to:

content_c

```
<PopulateCache continueOnError="false" enabled="true" name="PC-
PopulateAddress">
  <CacheResource>AddressesCache</CacheResource>
  <Scope>Exclusive</Scope>
  <Source>address</Source>
  <CacheKey>
    <KeyFragment ref="lat"/>
    <KeyFragment ref="lng"/>
  </CacheKey>
  <ExpirySettings>
    <TimeoutInSec>3600</TimeoutInSec>
  </ExpirySettings>
</PopulateCache>
```

This policy populates the value stored in address into the cache with a key combining the latitude and longitude.

15. Move the **LC-LookupAddress** step to precede the **ServiceCallout** policy (SC-GoogleGeocode), and move the **PC-PopulateAddress** policy to precede the **JavaScript** policy (JS-AddAddress).



GET getStoreById

EV-ExtractLatLng

LC-LookupAddress

SC-GoogleGeocode

EV-ExtractAddress

PC-PopulateAddress

JS-AddAddress

After the latitude and longitude are extracted from the backend response, the LC-LookupAddress policy will look for a matching entry in the cache. If a matching entry does not exist, the API proxy should call the ServiceCallout and ExtractVariables steps to get the address from the geocoding service. If the address was not in the cache, the PC-PopulateCache policy will put the address in the cache so that it can be used for the next request of the same store.

If the address is found in the cache, the ServiceCallout, ExtractVariables, and PopulateCache steps should all be skipped in the proxy flow.

16. Add the following condition:

```
          <Condition>lookupcache.LC-LookupAddress.cachehit ==
false</Condition>
```

content_c

to the SC-GoogleGeocode and EV-ExtractAddress steps. You added it to the PC-PopulateAddress step when you created that policy. You can detach and reattach those policies, but it is easier to update the default proxy endpoint XML.

Make your getStoreById flow look like this:

```
    <Flow name="getStoreById">
      <Description>Get a specific store</Description>
      <Request/>
      <Response>
```

content_c

```
        <Step>
          <Name>EV-ExtractLatLng</Name>
        </Step>
        <Step>
          <Name>LC-LookupAddress</Name>
        </Step>
        <Step>
          <Condition>lookupcache.LC-LookupAddress.cachehit ==
false</Condition>
          <Name>SC-GoogleGeocode</Name>
        </Step>
        <Step>
          <Condition>lookupcache.LC-LookupAddress.cachehit ==
false</Condition>
          <Name>EV-ExtractAddress</Name>
        </Step>
        <Step>
          <Condition>lookupcache.LC-LookupAddress.cachehit ==
false</Condition>
          <Name>PC-PopulateAddress</Name>
        </Step>
        <Step>
          <Name>JS-AddAddress</Name>
        </Step>
      </Response>
      <Condition>(proxy.pathsuffix MatchesPath
"/stores/{storeId}") and (request.verb = "GET")</Condition>
    </Flow>
```

lookupcache.LC-LookupAddress.cachehit is a variable set by the LC-LookupAddress policy that specifies whether an entry was found in the cache. If the entry is found in the cache, the variable value is set to true, and the ServiceCallout, ExtractVariables, and PopulateCache steps will be skipped.

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

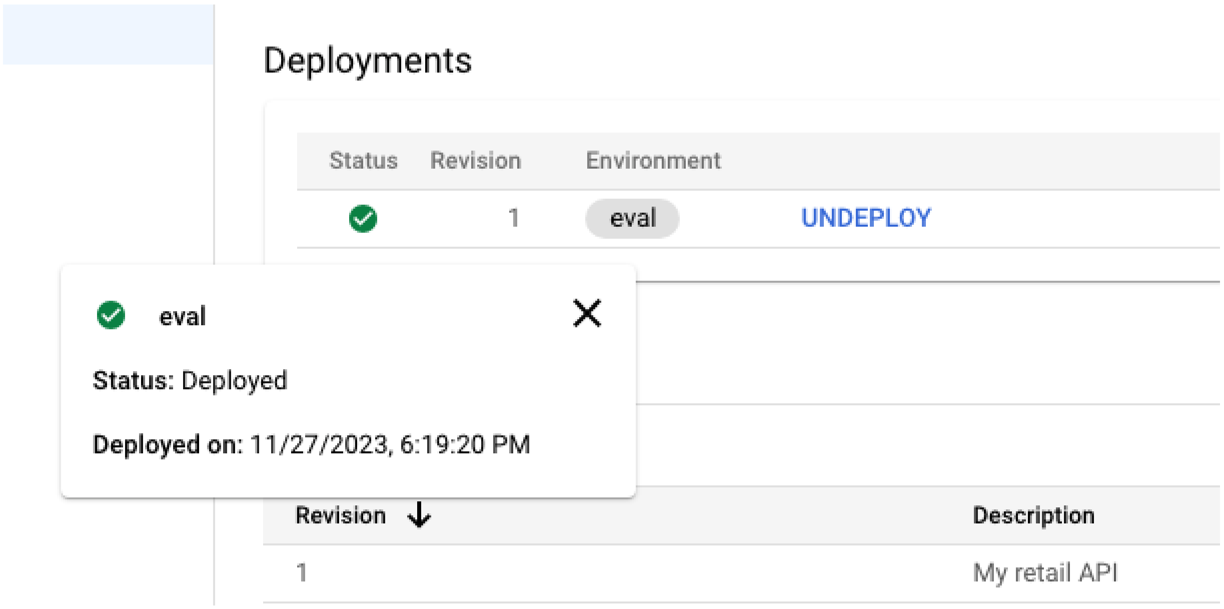
18. Click **Deploy**.

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**.

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

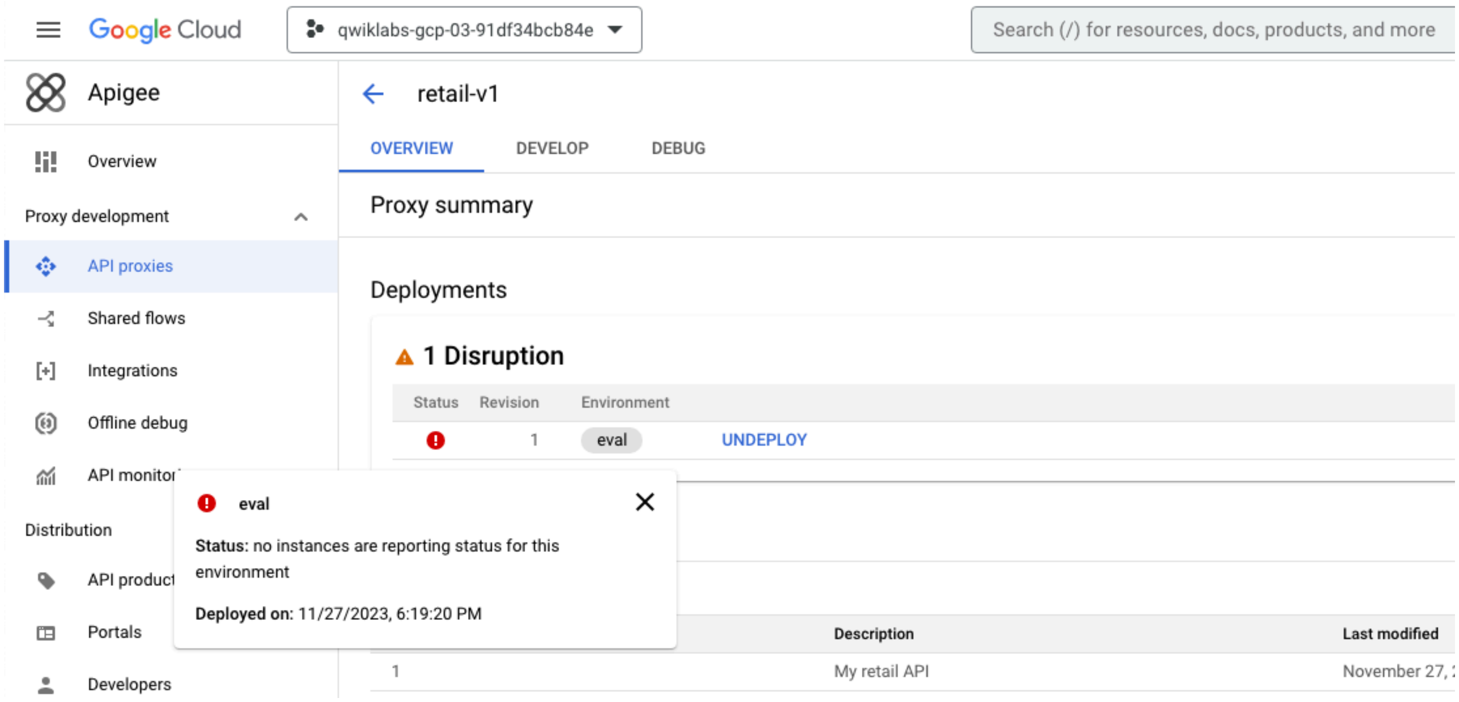
eval

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

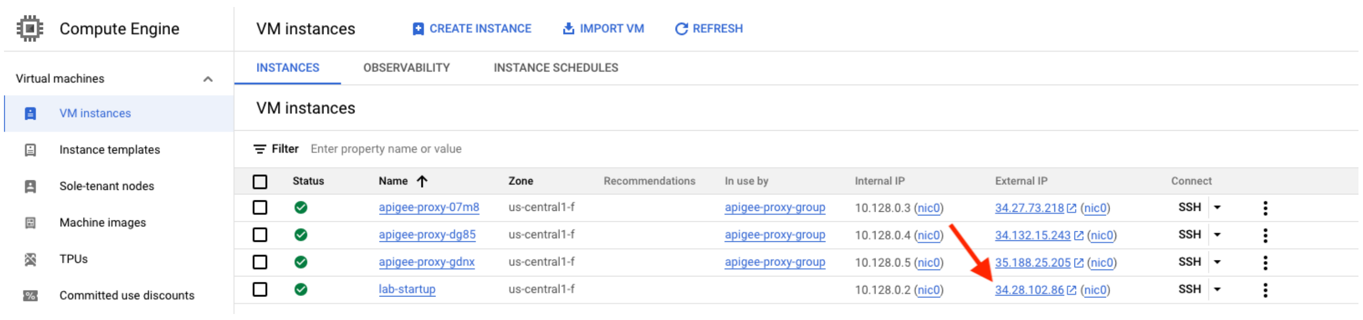
	Description	Last modified
1	My retail API	November 27,

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

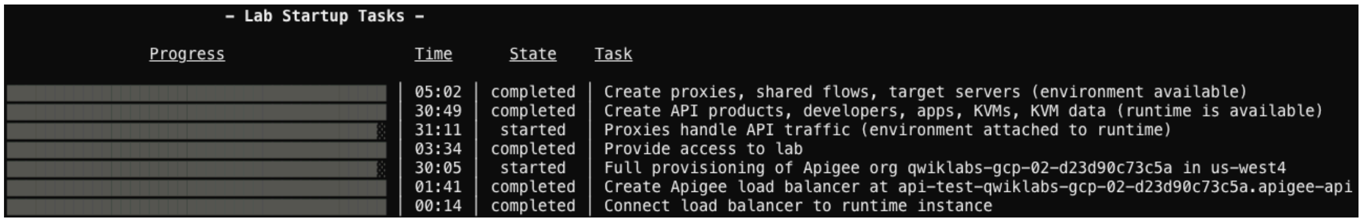
3. If you see a redirect notice page, click the link to the external IP address.

   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

- ResponseCache policy

- LookupCache policy
- PopulateCache policy
- Working with cache keys

Task 3. Test the API proxy

In this task, you use the debug tool to validate that the caching policies are working correctly in the API proxy.

Start a debug session

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
```

content_c

```
app" | jq --raw-output '.credentials[0].consumerKey'); echo "export
API_KEY=${API_KEY}" >> ~/.profile; echo "API_KEY=${API_KEY}"
```

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.

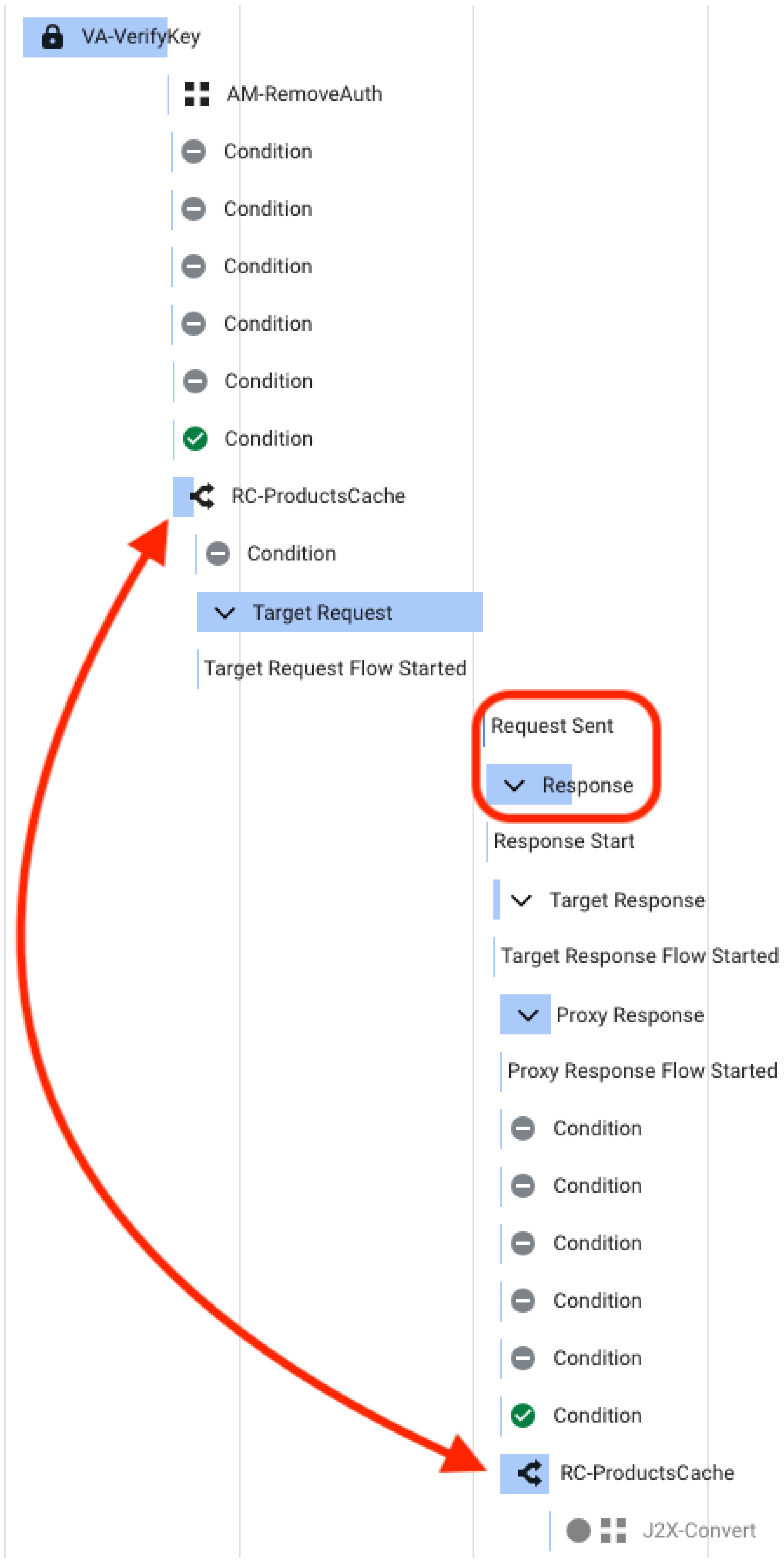
Test ResponseCache

1. In Cloud Shell, to retrieve a specific product by ID, execute this curl command:

```
curl -H "apikey: ${API_KEY}" -X GET "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/products/DV3
```

content_c

The debug tool will show that there was a cache miss and that the backend target was called:



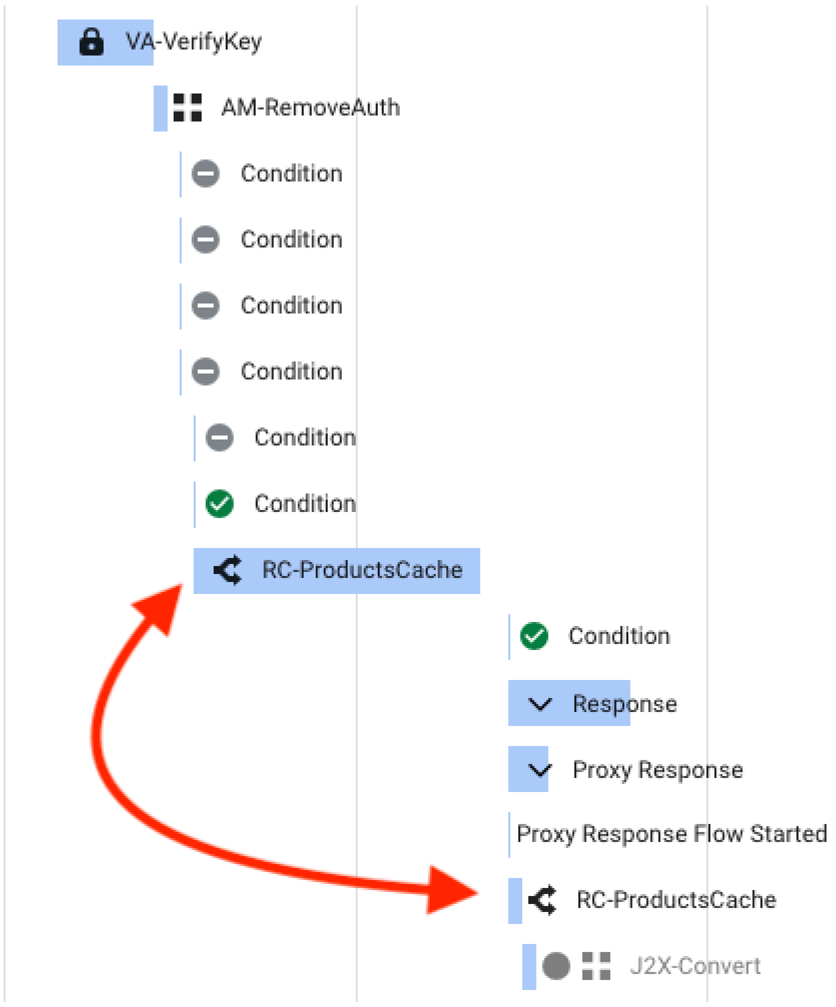
VA-VerifyKey
AM-RemoveAuth
Condition
Condition
Condition
Condition
Condition
Condition
RC-ProductsCache
Condition
Target Request
Target Request Flow Started
Request Sent
Response
Response Start
Target Response
Target Response Flow Started
Proxy Response
Proxy Response Flow Started
Condition
Condition
Condition
Condition
Condition
Condition
RC-ProductsCache
J2X-Convert

2. Make the same request again:

```
curl -H "apikey: ${API_KEY}" -X GET "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/products/DV3
```

content_c

This time, the debug tool will show that there was a cache hit and that the backend target was not called:



You can see that the elapsed time for the cache hit call is significantly shorter than that for the cache miss, because the backend service does not need to be called.

Note: In this implementation, products are cached for 10 minutes. If you want to replace the cached entry for a specific product, you can change the SkipCacheLookup condition to look for a specific header and skip the cache lookup if the header exists. If SkipCachePopulation is still false, the updated response for the product would be cached.

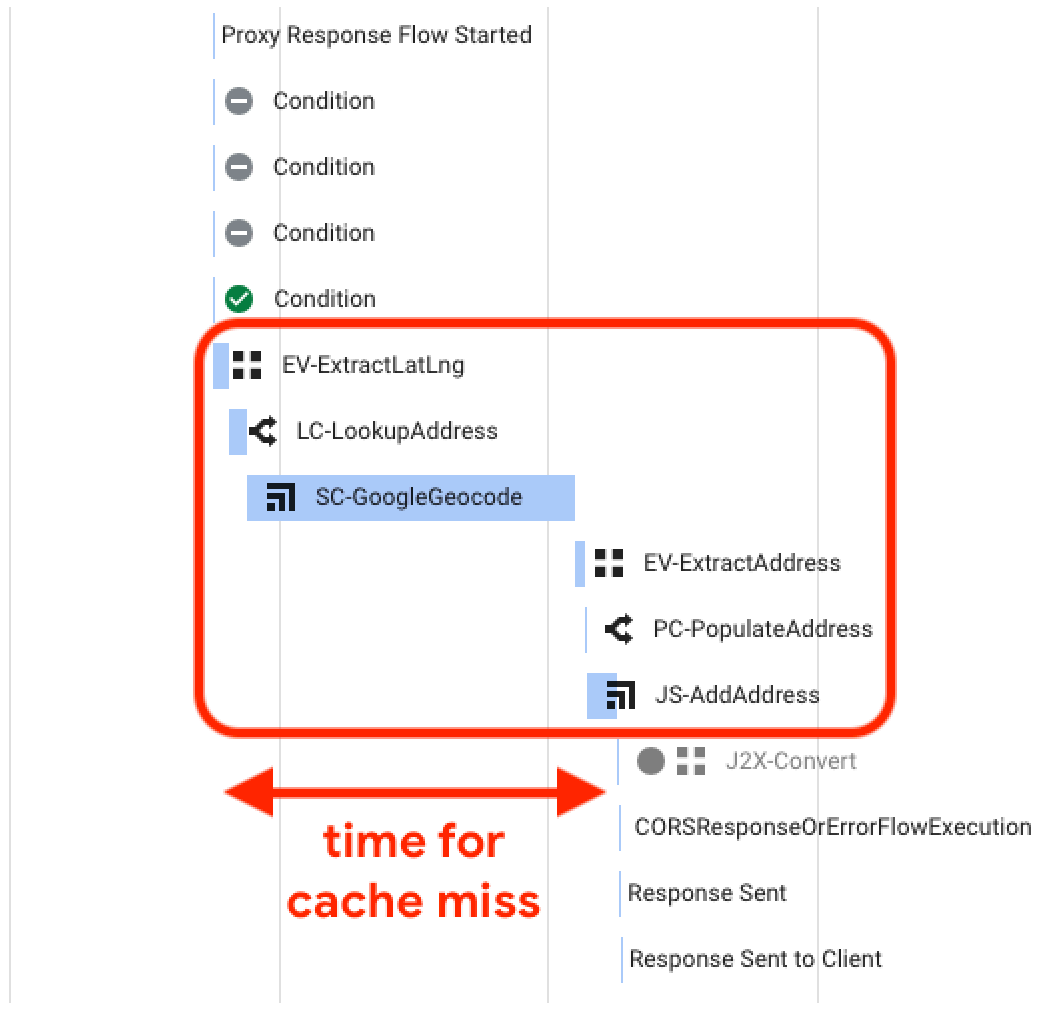
Test LookupCache and PopulateCache

1. In Cloud Shell, to retrieve a specific store by ID, execute this command:

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

content_c

The debug tool will show that all of the steps to call the geocoding service were called, because the store address was not found in the cache.

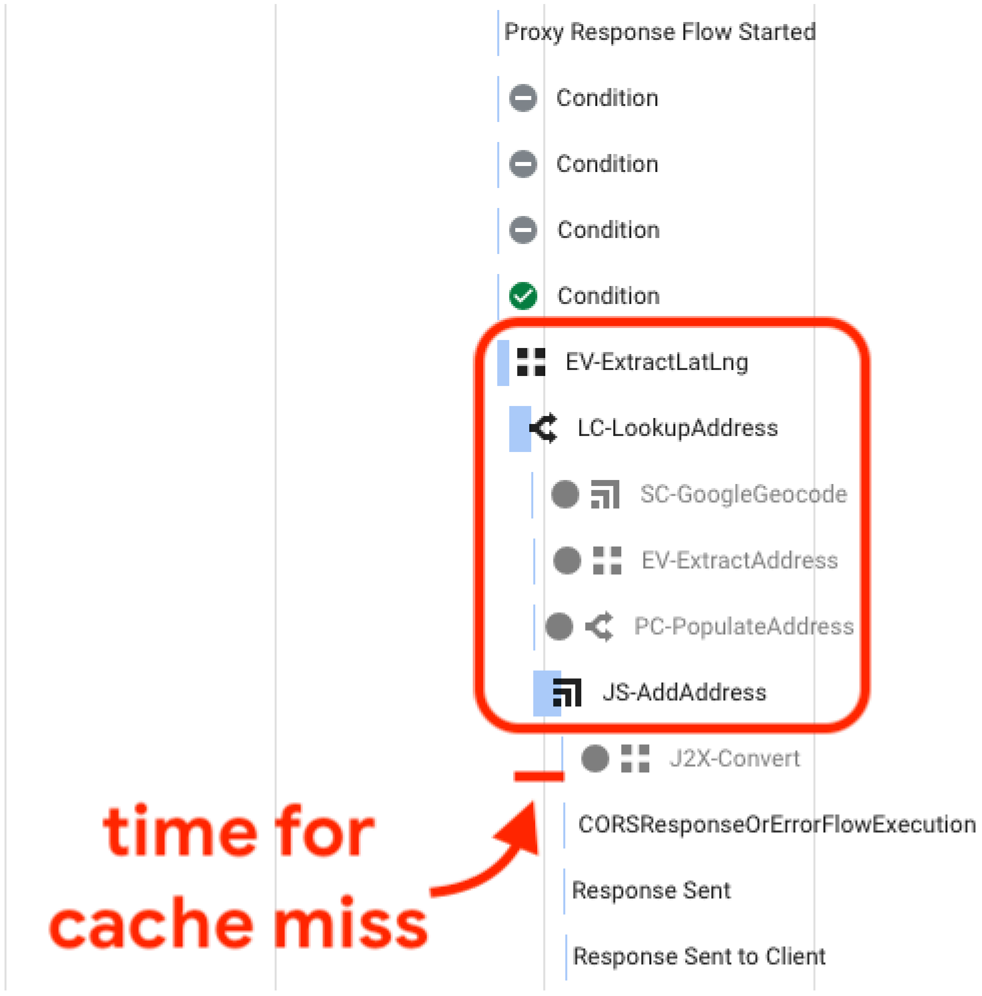


2. Make the same request again:

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

content_c

This time, the debug tool will show that there was a cache hit and that the ServiceCallout, ExtractVariables, and PopulateCache steps did not need to be called.



Congratulations!

In this lab, you learned how to use the ResponseCache policy to cache responses and how to use the LookupCache and PopulateCache policies to cache other data.

End your lab