

experiment Lab **schedule** 1 hour 30 minutes **universal_currency_alt** No cost

show_chart Introductory

Overview

Credentials are often necessary when an API proxy communicates with backend or third-party services, and credentials should always be protected when stored in Apigee. Security breaches can occur when internal users get unauthorized access to sensitive data.

In this lab, you use a key value map (KVM) to store backend credentials. Values stored in a KVM are encrypted. You will then use the KeyValueTypeMapOperations policy to retrieve the credentials into private variables and use them to build a Basic Authentication header.

Objectives

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

- Create and populate a key value map (KVM).
- Use data from a KVM in your proxy.
- Build a Basic Authentication header.

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.

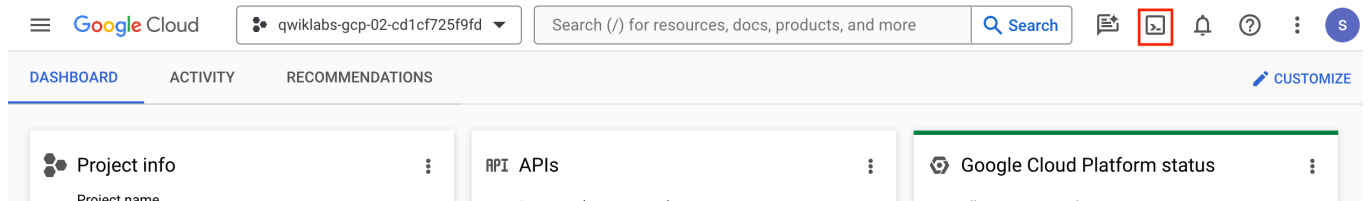
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

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:

```
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 .

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 **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. Modify the retail proxy to use the KVM credentials

In this task, you will use the KeyValueCollectionOperations policy to extract credentials from the key value map.

The **PATCH** `/products/{id}` call to the backend requires that a Basic Authentication header containing the backend credentials be added to the request. We will store these credentials in the KVM.

Note: A KVM cannot be created until the runtime is available. The Apigee organization you use for this lab takes a while to start up, and the runtime is not yet available when you start the lab. In a real-world scenario, you would typically create the KVM before you used it in a proxy.

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 5.

4. Select **Proxy endpoints > default > updateProductById**.
5. On the **Request updateProductById** flow, click **Add Policy Step (+)**.

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

7. Specify the following values:

Property	Value
Name	KVM-GetCredentials
Display name	KVM-GetCredentials

8. Click **Add**.

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

10. Set the policy configuration to:

```
<KeyValueMapOperations continueOnError="false" enabled="true" name=content_co
  <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>
```

This policy will extract the backendId and backendSecret values from the ProductsKVM into private variables.

Note: The ExpiryTimeInSecs element is set to 60, causing the policy to automatically cache the values of the KVM for 60 seconds. This helps with proxy performance. If you

modify an entry, you'll have to wait 60 seconds for cached values to expire before the change to the KVM entry will be detected.

The backend ID and secret are loaded into variables with a "**private.**" prefix. These variables will be masked in the debug tool. You must use private variables when retrieving data from a KVM.

Task 2. Add the Basic Authentication header

In this task, you use the BasicAuthentication policy to add a Basic Auth header, using the private variables from the KeyValueCollectionOperations policy.

1. Select **Proxy endpoints > default > updateProductById**.
2. On the **Request updateProductById** flow, click **Add Policy Step (+)**.
3. In the **Add policy step** pane, select **Create new policy**, and then select **Security > Basic Authentication**.
4. Specify the following values:

Property	Value
Name	BA-AddAuthHeader
Display name	BA-AddAuthHeader

5. Click **Add**.
6. Click on **Policies > BA-AddAuthHeader**.
7. Set the policy configuration to:

```
<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>
```

content_co

This policy will use the backend ID and secret to build a Basic Authentication header for the call to the backend service.

8. Select **Proxy endpoints > default > updateProductById**.

Your **updateProductById** flow should look like this:

Request	Response
<div> <div>ALL</div> <div>PreFlow</div> <div>+</div> <div>⋮</div> </div> <div> <div>VA-VerifyKey</div> <div>⋮</div> </div> <div> <div>AM-RemoveAuth</div> <div>⋮</div> </div>	<div> <div>ALL</div> <div>PostFlow</div> <div>+</div> <div>⋮</div> </div>
<div> <div>GET</div> <div>getCategories</div> <div>+</div> <div>⋮</div> </div>	<div> <div>DELETE</div> <div>deleteOrderById</div> <div>+</div> <div>⋮</div> </div>
<div> <div>GET</div> <div>getCategoryById</div> <div>+</div> <div>⋮</div> </div>	<div> <div>GET</div> <div>getOrderById</div> <div>+</div> <div>⋮</div> </div>
<div> <div>GET</div> <div>getStores</div> <div>+</div> <div>⋮</div> </div>	<div> <div>POST</div> <div>createOrder</div> <div>+</div> <div>⋮</div> </div>
<div> <div>GET</div> <div>getStoreById</div> <div>+</div> <div>⋮</div> </div>	<div> <div>PATCH</div> <div>updateProductById</div> <div>+</div> <div>⋮</div> </div>
<div> <div>GET</div> <div>getProducts</div> <div>+</div> <div>⋮</div> </div>	<div> <div>GET</div> <div>getProductById</div> <div>+</div> <div>⋮</div> </div>
<div> <div>GET</div> <div>getProductById</div> <div>+</div> <div>⋮</div> </div>	<div> <div>GET</div> <div>getProducts</div> <div>+</div> <div>⋮</div> </div>
<div> <div>PATCH</div> <div>updateProductById</div> <div>+</div> <div>⋮</div> </div> <div> <div>KVM-GetCredentials</div> <div>⋮</div> </div> <div> <div>BA-AddAuthHeader</div> <div>⋮</div> </div>	<div> <div>GET</div> <div>getStoreById</div> <div>+</div> <div>⋮</div> </div>
	<div> <div>GET</div> <div>getStores</div> <div>+</div> <div>⋮</div> </div>
	<div> <div>GET</div> <div>getCategoryById</div> <div>+</div> <div>⋮</div> </div>
	<div> <div>GET</div> <div>getCategories</div> <div>+</div> <div>⋮</div> </div>

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

10. Click **Deploy**.

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

Check runtime status

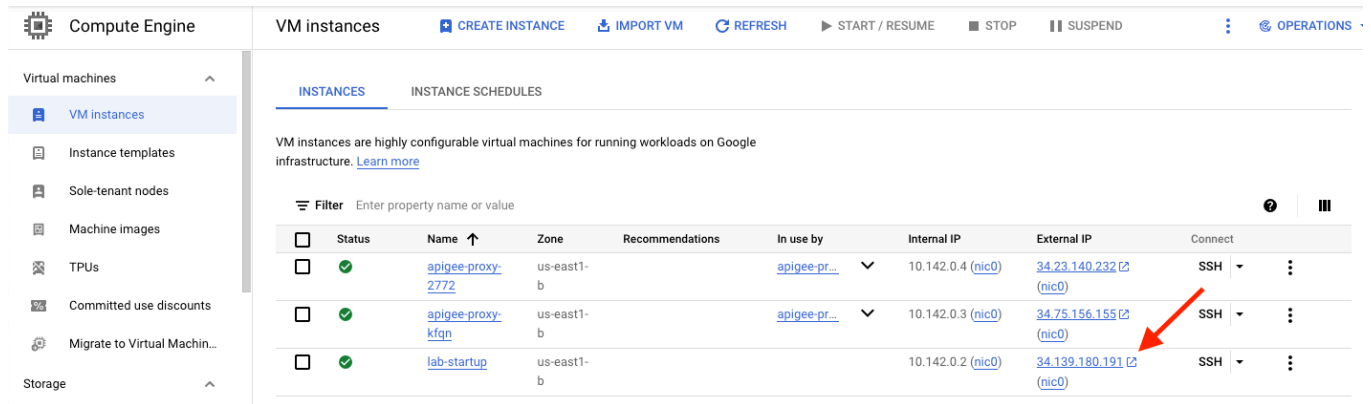
Certain assets, including API products, developers, developer apps, and KVMs, cannot be saved until the runtime is available.

For example, when navigating to the API products page, you might see an error message that reads "Products were not loaded successfully."

This is an error you should see when you are waiting for the runtime instance to be available. Once the runtime is available, refreshing the page will remove the error.

If you get this type of error, you can check the status of provisioning by navigating to the Lab Startup Tasks dashboard at the external IP address of the **lab-startup** VM instance.

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.



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	

**Before continuing, you need to wait for Create API products, developers, apps, KVMs, KVM data to complete.*

While you are waiting

While you wait for the new revision to deploy, review the following information:

- Reasons to use KVMs — Using key value maps
- KeyValueTypeMapOperations policy
- BasicAuthentication policy

Task 3. Create a new key value map

In this task, you create a key value map (KVM).

1. In the Google Cloud console, on the **Navigation menu** (≡), select **Integration Services > Apigee > Management > Environments**.
2. Click **eval**, and then click the **Key Value Maps** tab.

Note: If you see an error on the page, the runtime instance may not yet be available.

3. Click **+Create Key Value Map**:
4. For **Key value map name**, specify `ProductsKVM`, and then click **Create**.

Task 4. Populate the KVM

In this task, you use the Apigee API to populate your KVM.

KVM data is stored in the runtime database, and it can be populated by using the **KeyValueMapOperations** policy or using the Apigee API. For this lab, you will use the Apigee API.

The keys and values you will load are:

Key	Value
backendId	svcacct
backendSecret	UNdrDxeQ82

Load the KVM entries

1. Return to Cloud Shell and use this curl command to load the backendId:

```
curl -X POST
"https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PROJ}
-H "Authorization: Bearer $(gcloud auth print-access-token)" -H
"Content-Type: application/json" -d '{ "name": "backendId",
"value": "svcacct" }'
```

content_co

This command returns the key and value when they are created successfully.

2. Use this curl command to load the backendSecret with an *incorrect secret*:

```
curl -X POST
"https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PROJ}
-H "Authorization: Bearer $(gcloud auth print-access-token)" -H
"Content-Type: application/json" -d '{ "name": "backendSecret",
"value": "ABCD1234" }'
```

content_co

In the next task you will see that this incorrect secret causes your backend request to fail.

Task 5. Test the retail API

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

The only field in the product that you can update is the **overall_rating**. It must be updated to another decimal number.

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

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:

```
{  
  "18841": {  
    "category": "Clothing",  
    "image": "https://cdn.pixabay.com/photo/2016/03/20/13/48/zip-  
1268656_1280.jpg",  
    "name": "18841",  
    "overall_rating": 0,  
    "product_name": "Roll-Up Denim Bermuda Shorts (Wise) (Regular &  
Petite)",  
    "short_description": "An essential for kicked-back days, Bermuda-  
length denim shorts with rolled hems are detailed with whiskering for a  
comfy worn-in look.\\n15 1/2\\n regular inseam (size 8); 13\\n petite  
inseam (size 8P).\\nZip fly with button closure.\\nDark dye may transfer  
to lighter materials.\\n99% cotton, 1% spandex.\\nMachine wash cold,  
tumble dry low or lay flat to dry.\\nBy KUT from The Kloth;  
imported.\\nPoint of View."  
  },  
  "31001": {  
    "category": "Baby",  
    "image": "https://cdn.pixabay.com/photo/2017/09/11/16/11/ducks-  
2739503_480.jpg",  
    "name": "31001",  
    "overall_rating": 2.1,  

```

```
"product_name": "Munchkin 'White Hot' Duck Bath Toy",
"short_description": "Test the waters with America's #1 Safety Duck.
No need to worry that your baby's bath water is too hot to handle. This
adorable rubber ducky has our White Hot safety disc at the bottom that
tells you when the water is too hot, then lets you know that it's safe
to put your baby in."
},
"62003": {
```

The top-level keys are the IDs (18841, 31001, and 62003 are shown here). Choose any one of the IDs in the entire list.

2. Create an environment variable with the ID you have chosen, and replace "REPLACE" with the ID you have chosen:

```
export PRODUCT_ID=REPLACE
```

content_co

Replace "REPLACE" with the ID you have chosen.

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 might choose to change the rating to 4.5. Create an environment variable with the new rating you have chosen:

```
export NEW_RATING=REPLACE
```

content_co

Again, be sure to replace "REPLACE" with the new rating you have chosen.

4. Try to make the request using the incorrect backendSecret we loaded earlier by using this curl command:

```
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} }" | jq
```

content_co

You should get an error that looks like this:

```
{
  "error": "invalid_credentials",
  "error_description": "Credentials missing or incorrect."
}
```

5. Use these curl commands to delete the backendSecret and then add it with the correct value:

```
curl -X DELETE
"https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PROJ}
-H "Authorization: Bearer $(gcloud auth print-access-token)"
curl -X POST
"https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PROJ}
-H "Authorization: Bearer $(gcloud auth print-access-token)" -H
"Content-Type: application/json" -d '{ "name": "backendSecret",
"value": "UNdrDxeQ82" }'
```

content_co

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

content_co

The incorrect backendSecret may still be in the cache from the previous request. The KeyValueCollection policy specified an expiry time of 60 seconds. If you get an **invalid credentials** error for the first call, try your requests again until the request succeeds.

If successful, 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.

Task 6. Troubleshooting tips

If the response was not successful, read the code and debug the request to determine the issue. Here are some common issues:

- The KeyValueCollectionOperations or BasicAuthentication policy is in the wrong flow. The debug tool would show the policies not being called for the PATCH request.
- Your key value map was not successfully created or did not have the name **ProductsKVM**.
- The keys or values in the key value map were not set correctly. There could be leading or trailing spaces.
- The keys were not specified correctly in the KeyValueCollectionOperations policy. The private variables wouldn't be populated.

If you get a **401 Unauthorized** error that looks like this, your Basic Auth header is probably incorrect:

```
{
  "error": "invalid_credentials",
  "error_description": "Credentials missing or incorrect."
}
```

To debug an issue with the backendId or backendSecret, you need to assign the values to another variable to be able to see the values. You can create an **ExtractVariables** policy that follows the BasicAuthentication policy, and set its name to EV-DebugBasicAuth.

1. Replace the configuration with the following:

```
<ExtractVariables continueOnError="false" enabled="true"
name="EV-DebugBasicAuth">
  <DisplayName>EV-DebugBasicAuth</DisplayName>
  <Variable name="private.backendId">
    <Pattern>{retrievedBackendId}</Pattern>
  </Variable>
  <Variable name="private.backendSecret">
    <Pattern>{retrievedBackendSecret}</Pattern>
  </Variable>
  <Variable name="request.header.Authorization">
```

content_co

```
<Pattern>{builtAuthHeader}</Pattern>
</Variable>
<IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
</ExtractVariables>
```

2. Save the proxy and deploy the updated revision. When you debug the proxy, you can see the values that would otherwise be masked:

🔴 EV-DebugBasicAuth

Start Time: @105ms Timestamp: 2023-12-19 (09:12:08.254) -0800

Variables (4)

Name ↑	Value	Access
builtAuthHeader	Basic IHN2Y2FjY3Q6VU5kcckR4ZVE4Mg==	SET
request.header.Authorization	Basic IHN2Y2FjY3Q6VU5kcckR4ZVE4Mg==	GET
retrievedBackendId	svcacct	SET
retrievedBackendSecret	UNdrDxeQ82	SET

In the picture above, the backend ID and secret look correct. However, if you decode the Base64 value in the Authorization header, you'll see that there is a leading space in the backendId.

3. Command:

```
echo -n "IHN2Y2FjY3Q6VU5kcckR4ZVE4Mg==" | base64 --decode
```

content_c

The output has a leading space:

```
svcacct:UNdrDxeQ82
^-- space
```

Congratulations!

In this lab, you created a KVM and stored credentials in it. You retrieved the credentials from your retail-v1 proxy, and built a Basic Auth header that gave you access to the backend `updateProductById` resource.

End your lab