

Qlog: Build and Manage APIs with Apigee: Challenge Lab

13 Sep 2020

In this article, we will go through the lab **GSP336 Build and Manage APIs with Apigee: Challenge Lab**, which is labeled as an expert-level exercise. You will practice the skills and knowledge in the Build and Manage APIs with Apigee.

Topics tested:

- Create an API Facade and add functionality
- Share the APIs with partners through a developer portal
- Route traffic to different backend implementations of the API

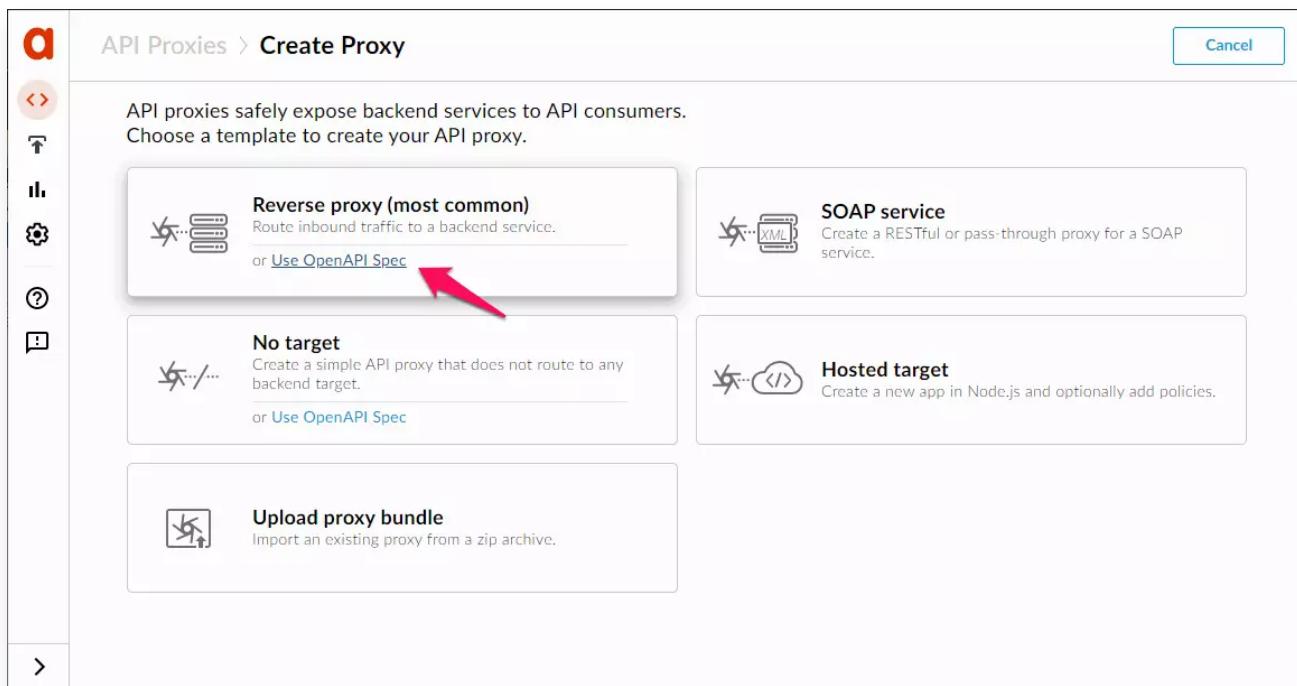
The challenge contains 4 required tasks:

Checkpoints	→
Upload API Proxy bundle to GCS	<button>Check my progress</button> / 20
Create a service account with permissions to write logs	<button>Check my progress</button> / 25
Add Authentication: API Key verification	<button>Check my progress</button> / 25
Route traffic to mock backend from real backend	<button>Check my progress</button> / 30

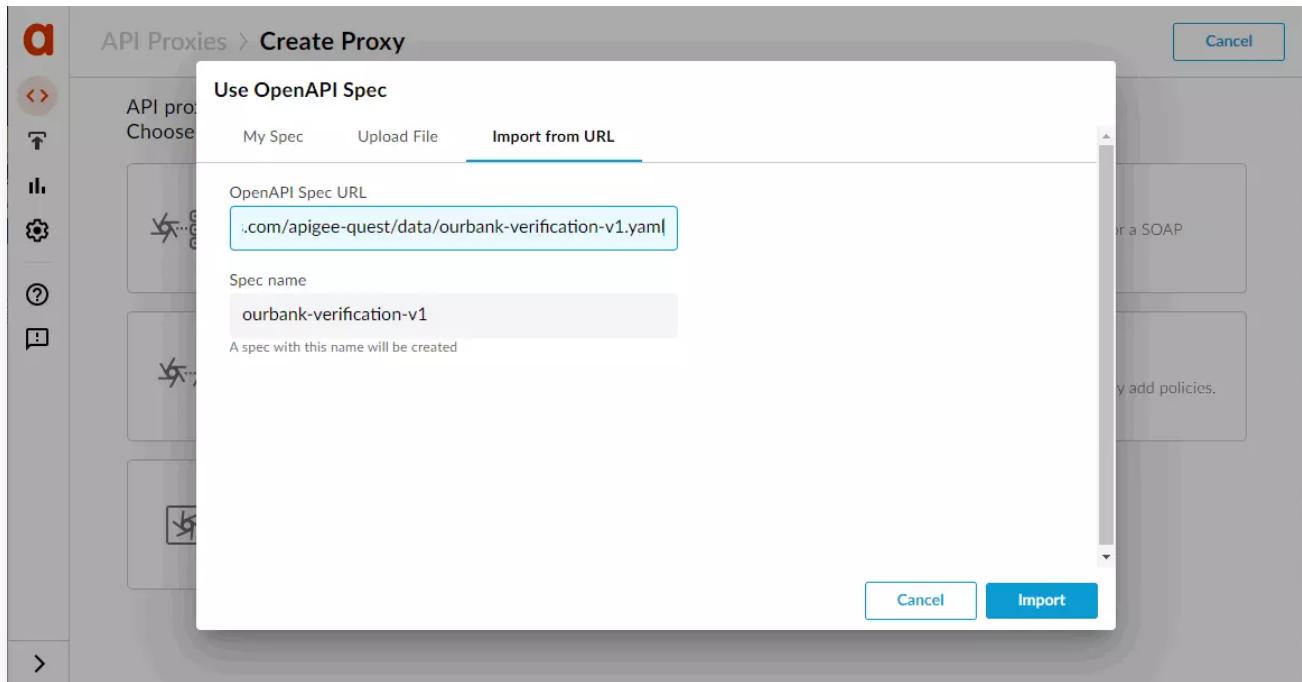
Task 1 - Create API Specification and Generate an API Proxy

Define a RESTful API in Apigee using an API specification

1. Login to your Apigee account (<https://login.apigee.com/login>)
2. In the Apigee console, select **Develop > API Proxies** from the left pane.
3. Click on the **+Proxy** button to create a new proxy.
4. Create a **Reverse proxy** by clicking “Use OpenAPI Spec” as shown in the picture below.



5. In the *Use OpenAPI Spec* dialog, select **Import from URL** and enter the following values:
 - **OpenAPI Spec URL:** <https://storage.googleapis.com/apigee-quest/data/ourbank-verification-v1.yaml>
 - **Spec name:** e.g. `ourbank-verification-v1`



6. Click **Import**.

7. Find the “View JSON response” endpoint URL of Mock Target API from [this Apigee Doc page](#), which is:

`https://mocktarget.apigee.net/json`

8. Copy the URL to the field **Target (Existing API)**, then click **Next**.

Proxy details

Name: Verification-API-v1 Available

Base path: /verification-api-v1

Description: This is OurBank's verification API. There are two operations available. You can verify credit card numbers and you can verify addresses.

Target (Existing API): https://mocktarget.apigee.net/json

Previous Next

9. Check **Add CORS header** to enable CORS headers in Apigee, then click **Next**.

Common policies

Security: Authorization

API Key

OAuth 2.0

Pass through (no authorization)

Quota

Impose quotas per App

Available only for proxies with authorization. Quota details are configured in API Products. [Learn more](#)

Security: Browser

Add CORS headers

Required for enabling web browser access to this proxy. [Learn more](#)

Previous Next

10. Click **Next** to continue.

Proxy details Policies Flows Virtual hosts Summary

OpenAPI operations

Select the operations to generate condition flows. You can update the operations later. [Learn more](#)

PATH	VERB	OPERATION	SUMMARY
/verifyCard	POST	verifyCreditCard	Verify a credit card
/verifyAddress	POST	verifyAddress	Verify an address

11. Check the box next to **default**, then click **Next**.

Proxy details Policies Flows Virtual hosts Summary

Virtual hosts

For incoming traffic, select the virtual hosts this proxy will bind to when it is deployed. [Learn more](#)

VIRTUAL HOST	HOST ALIAS
secure	https://[REDACTED].prod.apigee.net (Environment: prod) https://[REDACTED]-test.apigee.net (Environment: test)
default	http://[REDACTED].prod.apigee.net (Environment: prod) http://[REDACTED]-test.apigee.net (Environment: test)

12. Check the box next to **default**, then click **Create and deploy**.

The screenshot shows the 'Summary' tab of an API proxy configuration. At the top, there are five tabs: 'Proxy details' (with a checkmark), 'Policies' (with a checkmark), 'Flows' (with a checkmark), 'Virtual hosts' (with a checkmark), and 'Summary' (which is selected, indicated by a blue circle with the number '5'). Below the tabs, the 'Summary' section contains the following configuration details:

Proxy name	Verification-API-v1
Proxy type	Reverse Proxy
Proxy base path	/verification-api-v1
Target	https://mocktarget.apigee.net/json
Policies	Authorization - Pass through (no authorization), CORS header
OpenAPI Spec	Verification API (v1)
Virtual hosts	secure, default

Below the summary table is an 'Optional Deployment' section. It contains two checkboxes: 'prod' (unchecked) and 'test' (checked). A note below the checkboxes states: 'The proxy will begin handling requests only after it's deployed to an environment.'

Apigee now deploys the API proxy into your test environment. Click **Edit proxy** to view the deployed proxy.

Provision a Mock Response in Apigee

1. In the page of **API Proxies > Verification-API-v1**, click the **Develop** tab in the top right.
2. To add a policy to your proxy, click on **Proxy Endpoints → PreFlow** in the Navigator tab, then in the Response pipeline, click **+ Step** to add a step.

The screenshot shows the Apigee API Proxy Editor interface. On the left, the Navigator pane lists 'Verification-API-v11', 'Policies', 'Proxy Endpoints', 'Target Endpoints', and 'Resources'. Under 'Proxy Endpoints', 'default' is selected, showing 'PreFlow', 'verifyCreditCard', 'verifyAddress', and 'PostFlow'. The 'Flow: Preflow' tab is active, displaying a sequence of steps: REQUEST → Step → RESPONSE. The 'Property Inspector' on the right shows a step named 'Preflow' with 'name: Preflow'. The code editor displays the XML configuration for the Preflow steps.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<proxy version="1.0" encoding="UTF-8" standalone="yes">
  <!--> <proxyEndpoint name="default">
    <!--> <step name="Preflow">
      <!--> <description>Verify a credit card</description>
      <!--> <condition>(Proxy.pathsuffix MatchesPath "/verifyCard") and (request.verb = "POST")</condition>
      <!--> <flow name="verifyCreditCard">
        <!--> <description>Verify a credit card</description>
        <!--> <condition>Verify an address</description>
        <!--> <flow name="verifyAddress">
          <!--> <description>Verify an address</description>
          <!--> <condition>(Proxy.pathsuffix MatchesPath "/verifyAddress") and (request.verb = "POST")</condition>
          <!--> <flow name="PostFlow">
            <!--> <description>Post flow</description>
            <!--> <endPoint connection="virtualHost">
              <!--> <virtualHost default="virtualHost">
                <!--> <route name="PostRoute">
                  <!--> <targetEndpoints>default</targetEndpoints>
                <!--> </route>
              <!--> </virtualHost>
            <!--> </endPoint>
          <!--> </flow>
        <!--> </flow>
      <!--> </flow>
    <!--> </step>
  <!--> </proxyEndpoint>
</proxy>
  
```

3. Select **Assign Message** from the left menu, then click **Add**.

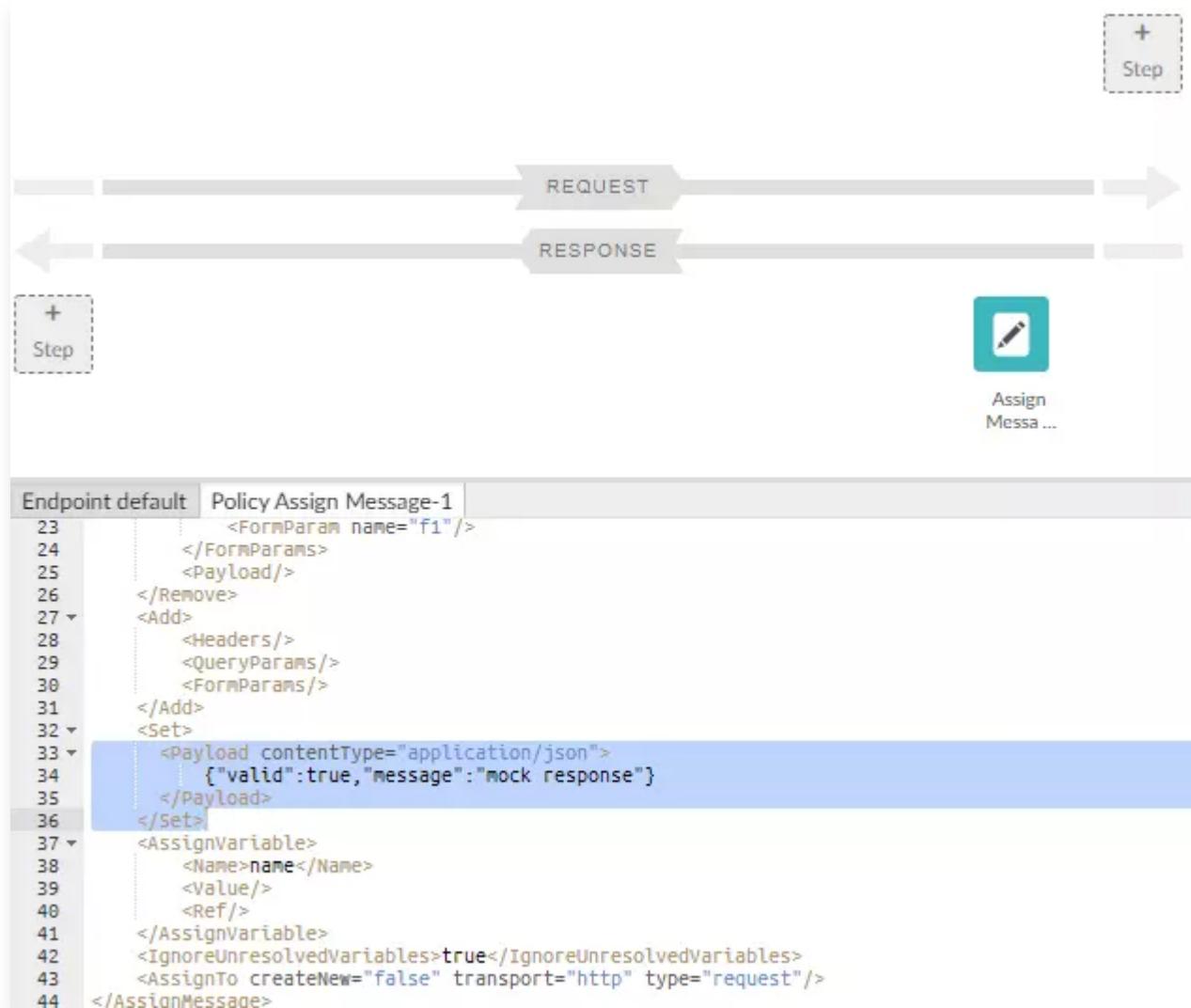
The screenshot shows the 'Add Step' dialog box. The 'Policy Instance' dropdown has 'New' selected. The left sidebar lists policies: 'SOAP Message Validation', 'Assign Message' (which is highlighted), 'Extract Variables', 'Access Entity', and 'Key Value Map Operations'. The right panel shows the 'Policy Type' as 'Assign Message' (with a checked checkbox). The 'Display Name' field contains 'Assign Message-1' and the 'Name' field contains 'Assign-Message-1'. At the bottom right are 'Cancel' and 'Add' buttons.

4. Modify Policy Assign Message-1:

- o Replace the **Set** element in the policy with the below.

```
<Set>
  <Payload contentType="application/json">
    {"valid":true,"message":"mock response"}
  </Payload>
</Set>
```

After the above procedure, the **Proxy Endpoints → PreFlow** should look like the picture below.



Testing

The deployment can be tested using the following curl statement in the Cloud Shell:

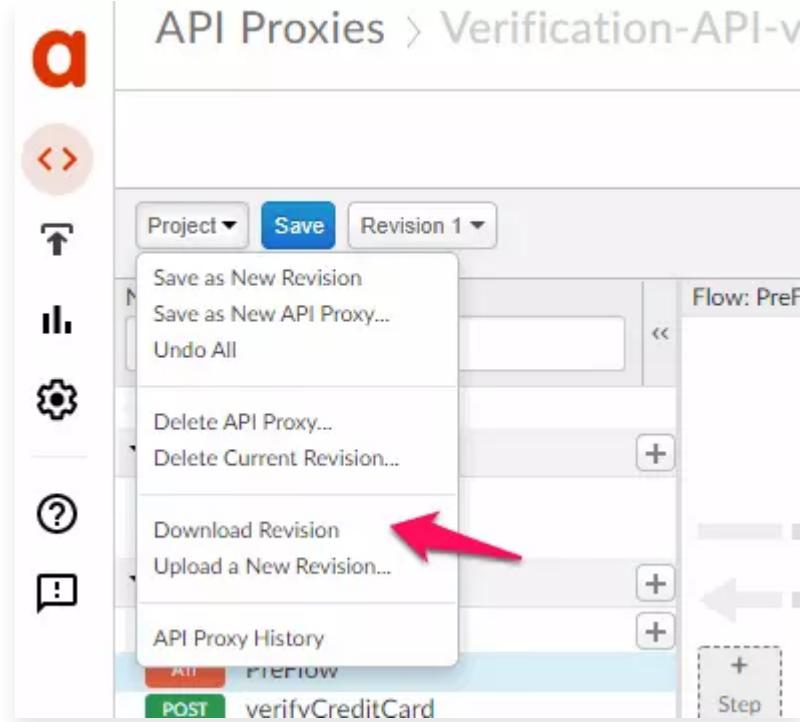
```
export APIGEE_ORG=<YOUR_APIGEE_ORG_NAME>

curl -X POST \
  https://${APIGEE_ORG}-test.apigee.net/verification-api-v1/verifyCard \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: 89236919-eabe-4357-e4c4-079f20ecd798' \
  -d '{
    "number": "2221005276762844",
    "cvv": "345",
    "expiration": "10/2025"
}'
```

Replace <YOUR_APIGEE_ORG_NAME> with your Apigee organization name.

Upload API Proxy bundle to GCS

1. Go back to Apigee, click on **Project > Download Revision** at the top-right of the Deploy tab.



2. Extract the downloaded zip file to your local storage.
3. In the Cloud Console, click on **Navigation Menu > Storage**.
4. Create a new bucket.

5. Upload the `apiproxy` folder to the bucket.

Task 2 - Add Policies to the API Proxy

Create a service account with permissions to write logs

1. In the Cloud Console, click on **Navigation Menu > IAM & admin > Service accounts**.
2. Click **Create Service Account**, then enter the following:
 - Service account name: `apigee-stackdriver`
 - Service account description: `Service account for Apigee Stackdriver integration`
3. Click **Create** to continue.
4. Click into **Select a Role** field and choose **Logging > Logs Writer** permission.
Click **Continue** then click **Done**.
5. After creating service account ‘apigee-stackdriver’, click on three dots at the right corner and click **Create Key** in the dropdown, then **Create** to download your JSON output.

Create an extension and deploy it to test the environment using this service account

1. Go back to Apigee, naviagte to **Admin > Extensions** from the left navigation menu, then **+ Add Extension** to create a new extension.
2. On the new Extension Properties page, click on the **Google Stackdriver Logging** extension.

3. Enter a name (e.g. `stackdriver-logging-extension`) and an optional description for the Extension instance, then click **Create**.
4. On the Extension detail page, click the arrow (>) for the test environment to configure the instance for the Apigee environment.
5. In the configuration dialog, enter the following information:
 - Select the latest extension version from the Version drop-down list.
 - Add your GCP Project ID (which you can get from the Home Console).
 - Open the downloaded JSON file and copy/paste the contents into the Credential field in the Apigee UI.
6. Click **Save**.
7. Once the configuration is saved, click on the **Deploy** button for the Test environment.

The screenshot shows the Apigee Extension detail page for 'stackdriver-logging-extension'. The top navigation bar includes 'Extensions > stackdriver-logging-extension', 'Clone', and 'Delete'. The left sidebar has icons for back, forward, up, down, settings, and help. The main content area is divided into sections:

- Extension Properties:** Shows the extension name 'stackdriver-logging-extension' and a 'View Details' link.
- Environment Configurations:** Shows two environments: 'prod' (selected) and 'test'. The 'test' environment has a green checkmark and the word 'Deployed'.

Create an Extension policy in the PostFlow response path

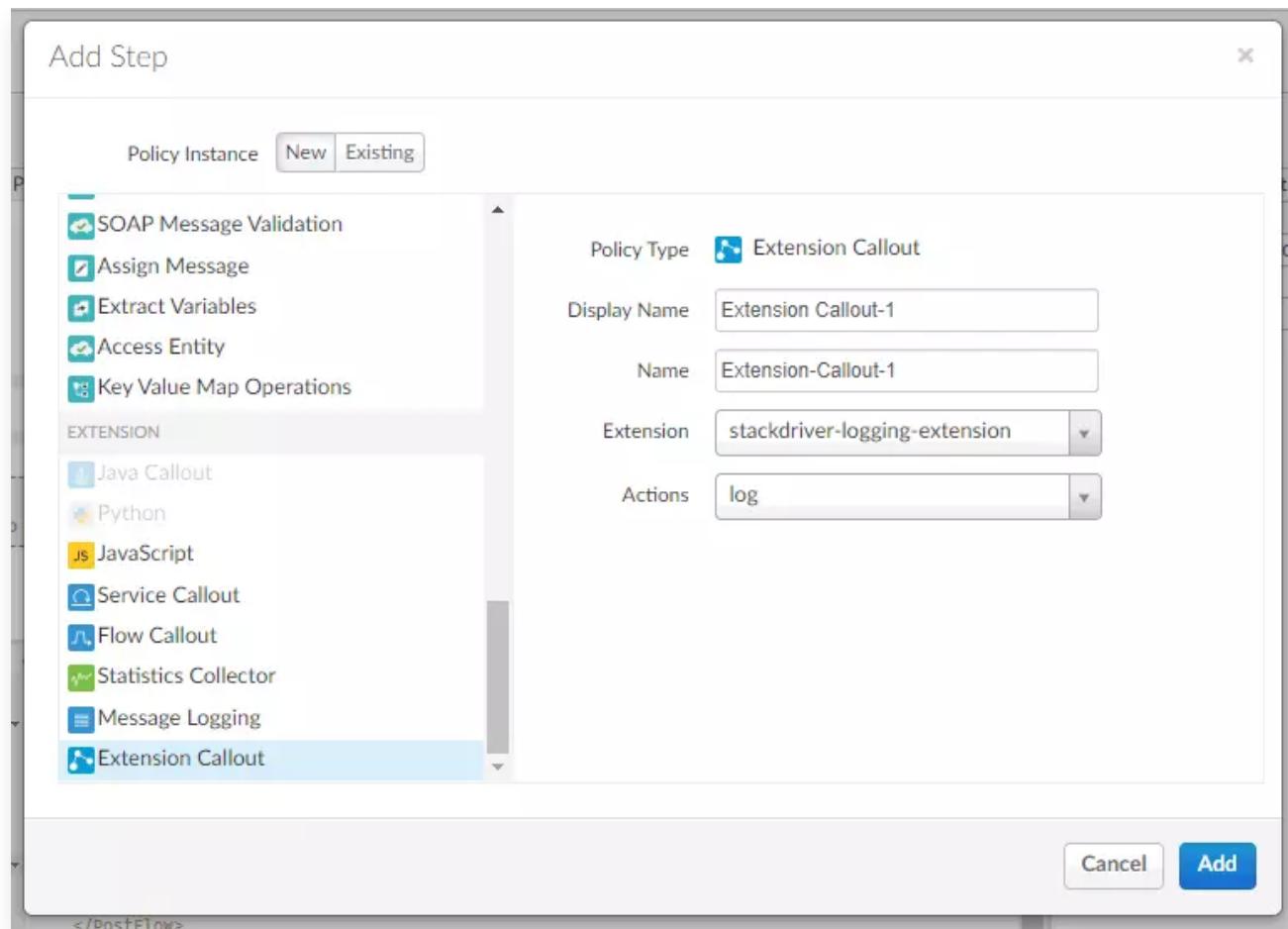
1. In Apigee, navigate to **Develop > API Proxies** from the left menu.
2. Click to open **Verification-API-v1** from the proxy list.

3. Click the **Develop** tab in the top right.
4. To add a policy to your proxy, click on **Proxy Endpoints** → **PostFlow** in the Navigator tab, then in the *Response* pipeline, click **+ Step** to add a step.

The screenshot shows the Apigee API Publisher interface. The top navigation bar includes 'Project', 'Save', 'Revision 1', 'Tools', 'Deployment', and 'Help'. The left sidebar, titled 'Navigator', contains sections for 'Verification-API-v11', 'Policies' (with 'Add CORS' and 'Assign Message-1' checked), 'Proxy Endpoints' (selected), 'Target Endpoints' (selected), and 'Resources'. Under 'Proxy Endpoints', 'default' is expanded, showing 'All PreFlow', 'POST verifyCreditCard', 'POST verifyAddress', and 'All PostFlow' (which is selected and highlighted in blue). Under 'Target Endpoints', 'default' is expanded, showing 'All PreFlow' and 'All PostFlow'. The main panel, titled 'Flow: PostFlow', shows a flow diagram with 'REQUEST' and 'RESPONSE' arrows. In the 'Response' pipeline, there is a dashed box labeled '+ Step' with a red arrow pointing to it. Below the pipeline, a code editor shows XML code for the PostFlow steps, with lines 16 through 24 visible.

```
16 <Condition>(proxy.pathsuffix MatchesPath "/verifyCa
17 </Flow>
18 <Flow name="verifyAddress">
19 <Description>Verify an address</Description>
20 <Request/>
21 <Response/>
22 <Condition>(proxy.pathsuffix MatchesPath "/verifyAd
23 </Flow>
24 </Flows>
```

5. In the left menu scroll down to the end, then select **Extension Callout**. Select your extension name from the **Extension** dropdown menu. Select **Log** from the **Actions** dropdown. Click **Add**.

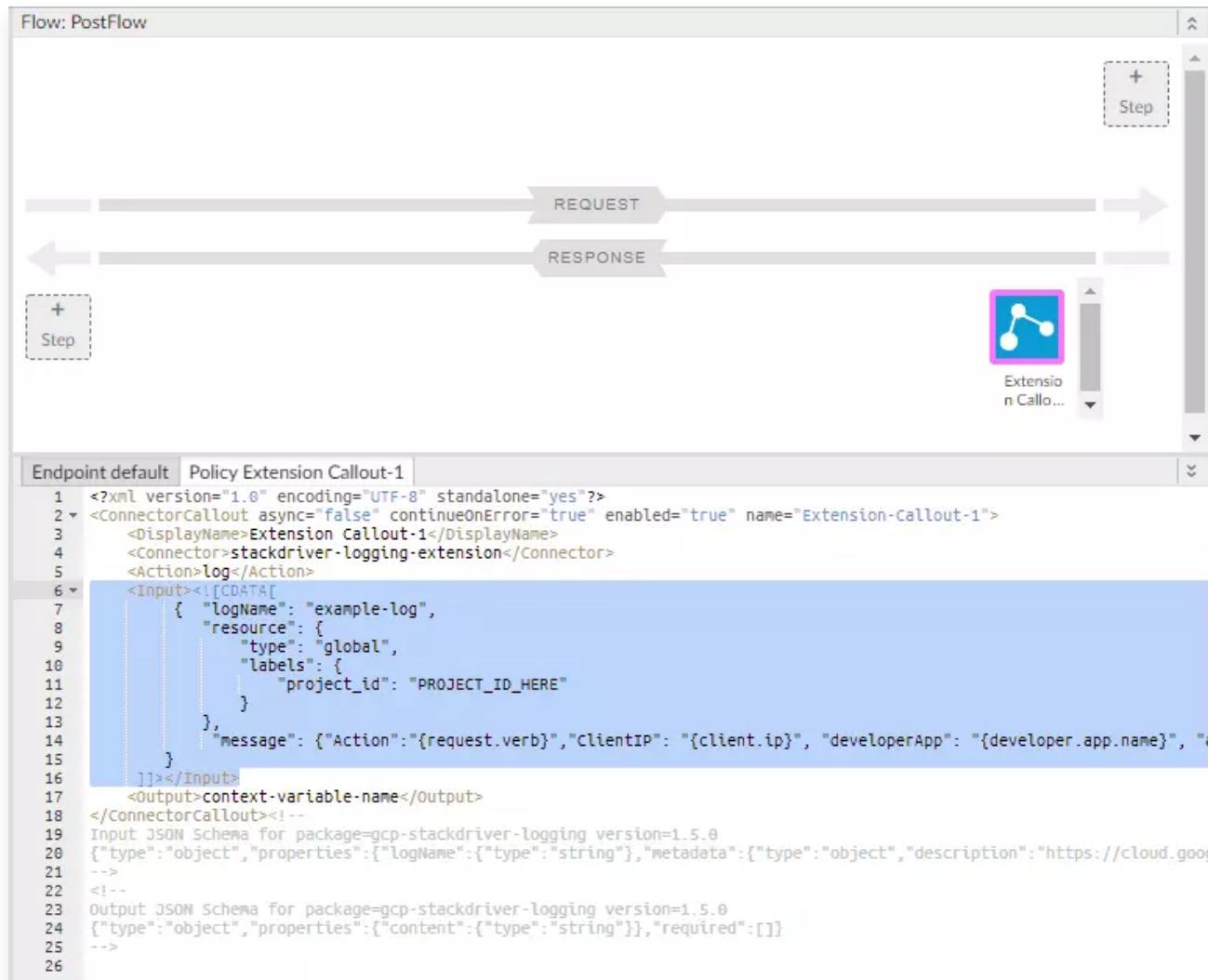


6. Modify Policy Extension Callout-1:

- Replace the **Input** element in the policy with the below (make sure indentation is correct)
- Replace **PROJECT_ID_HERE** with GCP Project ID for this lab.

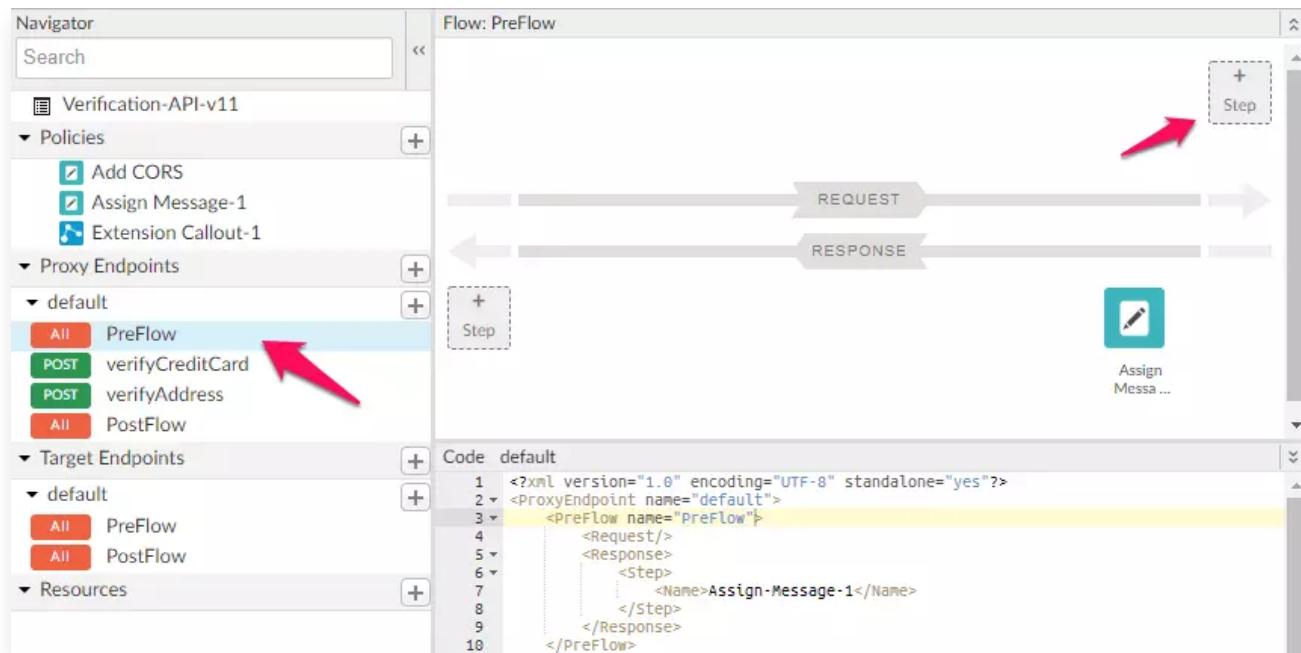
```
<Input><![CDATA[
  {
    "logName": "example-log",
    "resource": {
      "type": "global",
      "labels": {
        "project_id": "PROJECT_ID_HERE"
      }
    },
    "message": {"Action": "{request.verb}", "clientIP": "client.ip", "developerApp": "developer.app.name", "apiKeyParam": "{request.queryparam.apikey}", "responsePayload": {response.content}}
  }
]></Input>
```

After the above procedure, the **Proxy Endpoints → PostFlow** should look like the picture below.

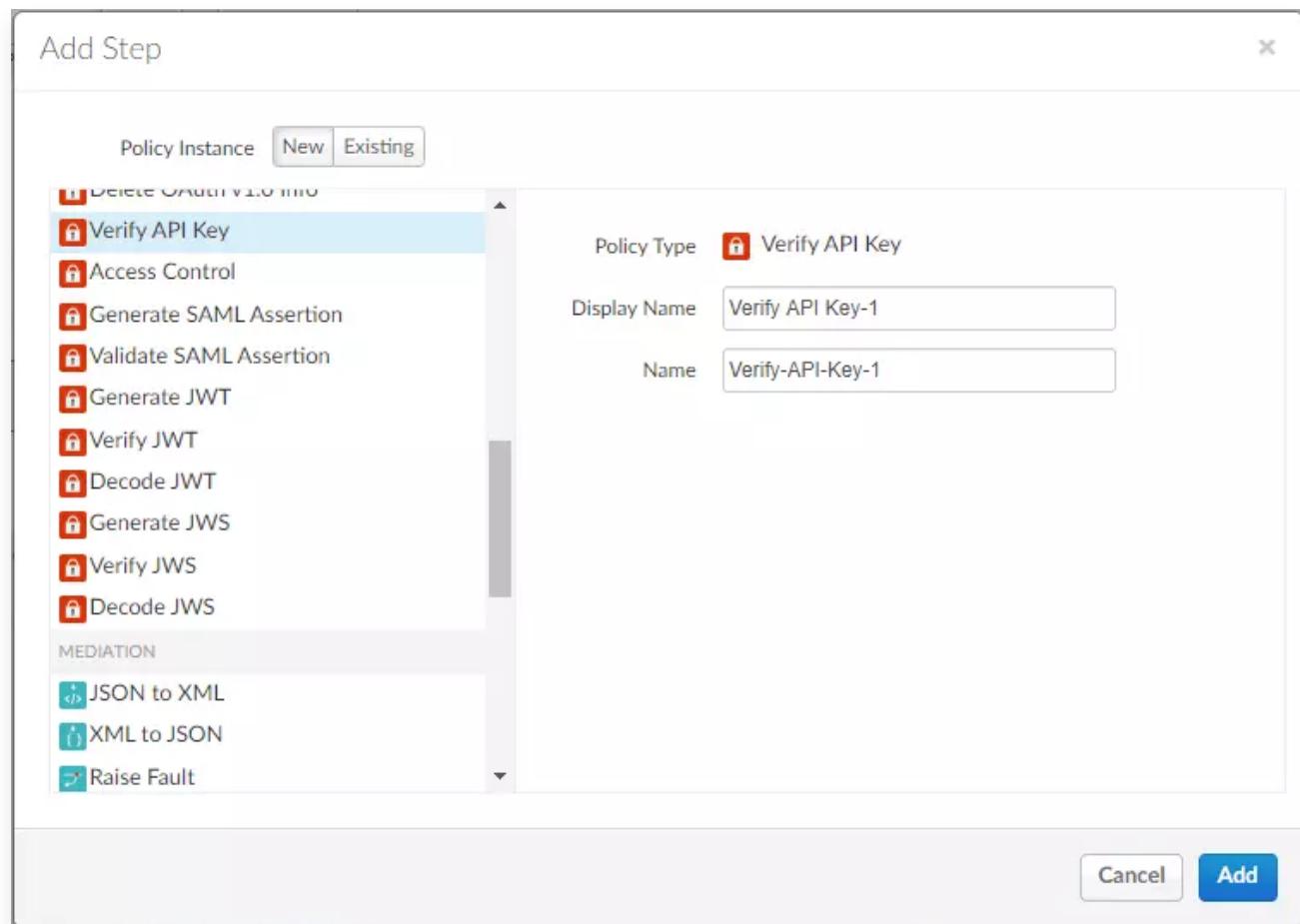


Add API Key verification

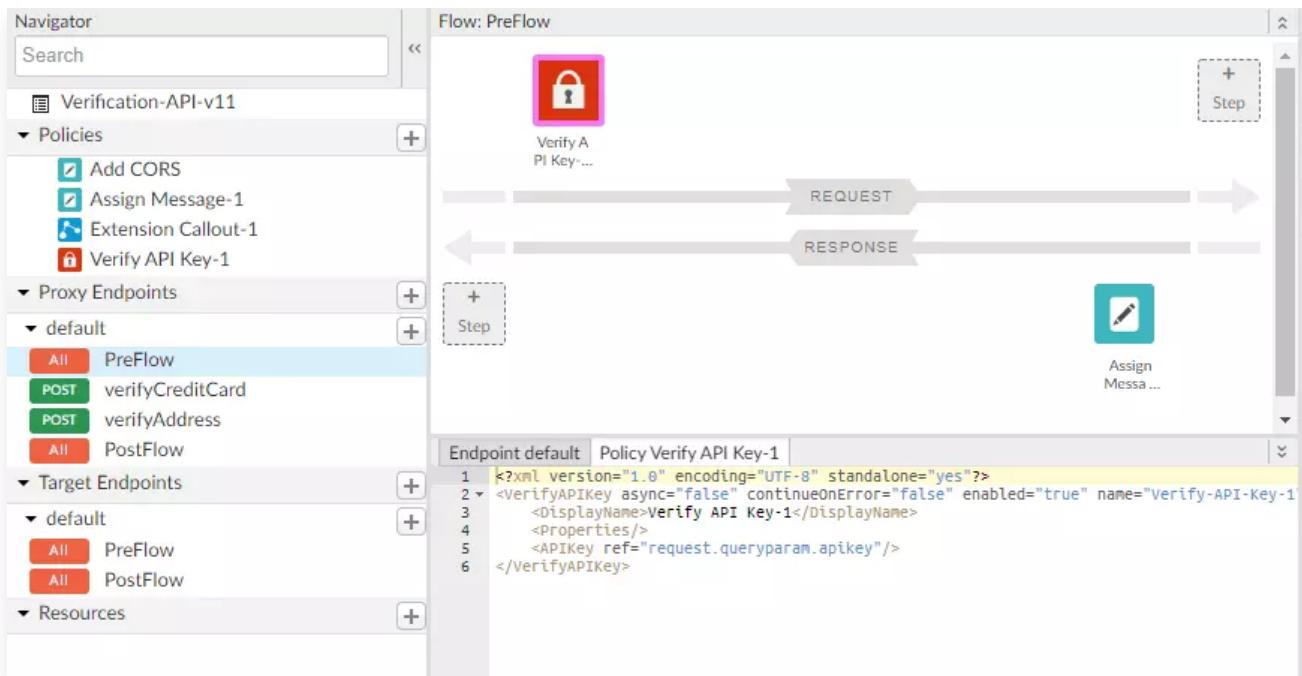
1. To add a policy to your proxy, click on **Proxy Endpoints → PreFlow** in the Navigator tab, then in the *Response* pipeline, click **+ Step** to add a step.



2. Select **Verify API Key** from the left menu, then click **Add**.



After the above procedure, the **Proxy Endpoints → PreFlow** should look like the picture below.



Create an API Product and an App

1. In Apigee, navigate to **Publish > API Products** from the left menu. Then Click **+ API Product**.
2. Create a new API product with the following details:
 - **Name:** e.g. `ourBank API`
 - **Display Name:** e.g. `ourBank API`
 - **Environment:** Check `test`
 - **Access:** Select `Public` from the dropdown menu
 - In the API resources section, **API proxies** list, click **Add a proxy** and select the **Verification-API-v1 API proxy**. Click **Add(1)**.

The settings should look as follows:

Product details

Name	ourBank API	Access	Public
Display Name	ourBank API	Request approval	Automatic
Description	No Description	Quota	No Quota
Environment	test	Allowed OAuth scope	No Allowed OAuth scope

API resources

Specify the API resources to allow access to by adding the API proxies and paths to the API Product.

API proxies

This API Product will allow access to the following paths

- /verification-api-v1/

Paths

No paths added.

Apigee remote service targets

Set up an Apigee remote service in order to add it to an API product.

Custom attributes

Key-value pairs used to store and retrieve values at runtime to control API proxy execution and customize analytics reports.

No custom attributes added.

3. Click **Save** to create the new API product.

4. Navigate to **Publish > Apps** from the left menu. Then Click **+ App**.

5. Enter the following:

- **Name:** e.g. ourBank App
- **Display Name:** e.g. ourBank App
- **Company / Developer:** Developer
- **Developer:** Use the helloworld@apigee.com developer account (which has been pre-populated for the Apigee trial environment).

6. Within the Credentials section, click the **Add product** button.

7. Select **ourBank API** from the list and click **Add (1)**.

8. Review that the configuration looks as below, and click **Create**.

The screenshot shows the Apigee interface for managing APIs. On the left is a sidebar with icons for Home, Apps, API, Product, Analytics, Metrics, and Help. The main area is titled 'Apps > ourBank App'. The page is divided into three sections: 'App Details', 'Credentials', and 'Custom Attributes'.

App Details:

Name	ourBank App	App status	Approved
Display Name	ourBank App	Callback URL	
Registered	Sep 12 2020 1:49:21 AM (UTC)	Notes	
Developer	helloworld dev (helloworld@apigee.com)		

Credentials:

Status	Approved	Product	Status
Key	Show ······	ourBank API	Approved
Secret	Show ······		
Issued	Sep 12 2020 1:49:21 AM (UTC)		
Expiry	Never		

Custom Attributes:

No custom attributes.

- Once the app is created, the page displays details about the app credentials. Click **Show** on the Key row then copy the API key.

Testing

Replace `<APP_API_KEY>` with the key from the app credentials. Then, run the following curl statement in the Cloud Shell:

```
export APIKEY=<APP_API_KEY>

curl -X POST \
  https://${APIEE_ORG}-test.apigee.net/verification-api-v1/verifyCard?
apikey=${APIKEY} \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: 89236919-eabe-4357-e4c4-079f20ecd798' \
  -d '{
    "number": "2221005276762844",
```

```
"cvv": "345",
"expiration": "10/2025"
}'
```

Successful response message:

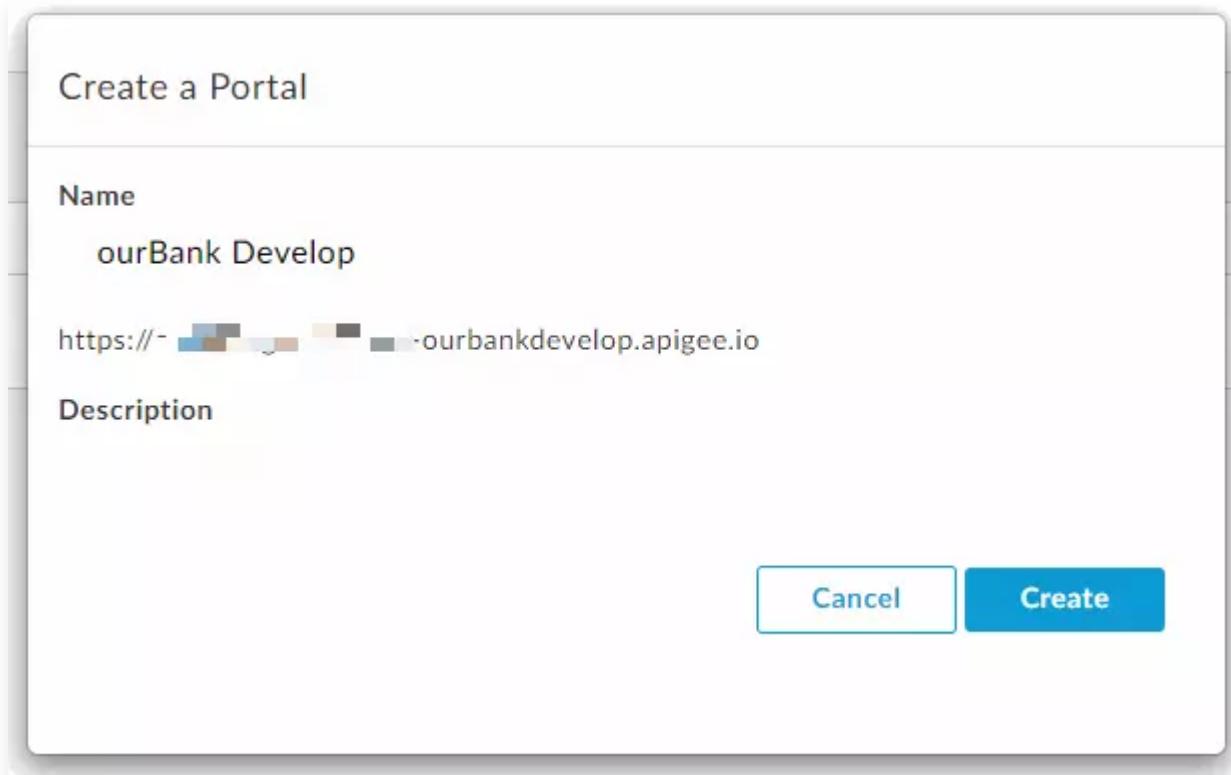
```
{"valid":true, "message":"mock response"}
```

Error response message:

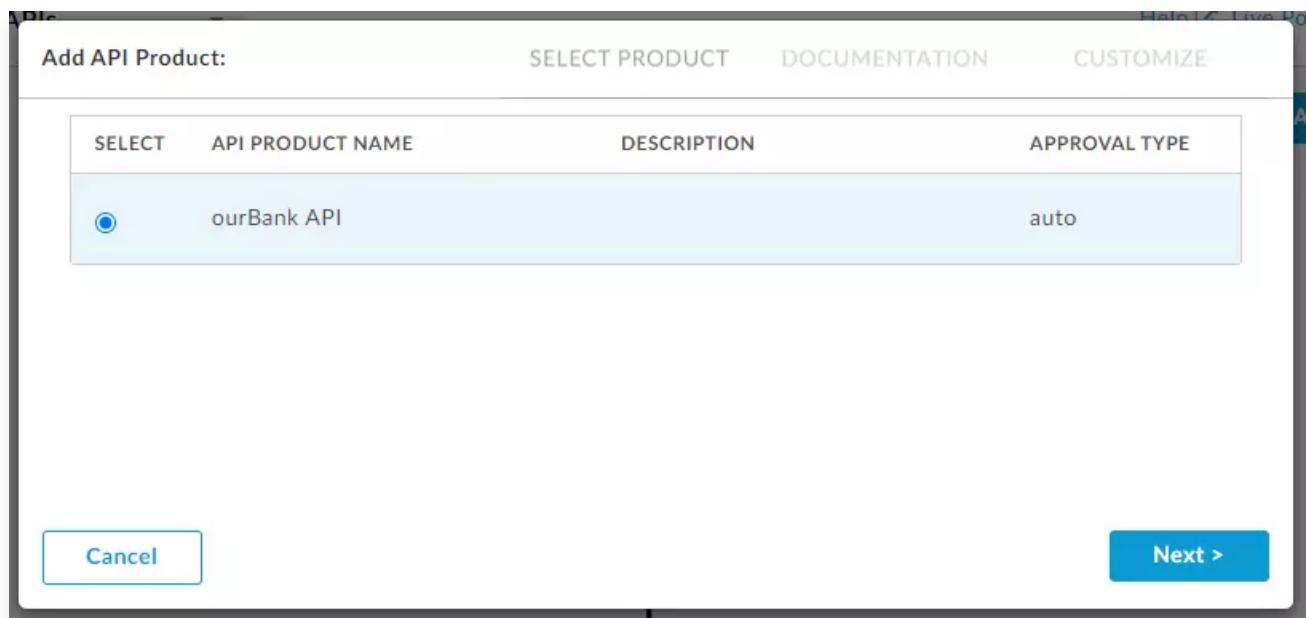
```
{"fault":{"faultstring":"Invalid ApiKey", "detail":
{"errorcode": "oauth.v2.InvalidApiKey"}}}
```

Task 3 - Create a Developer Portal for consuming the APIs

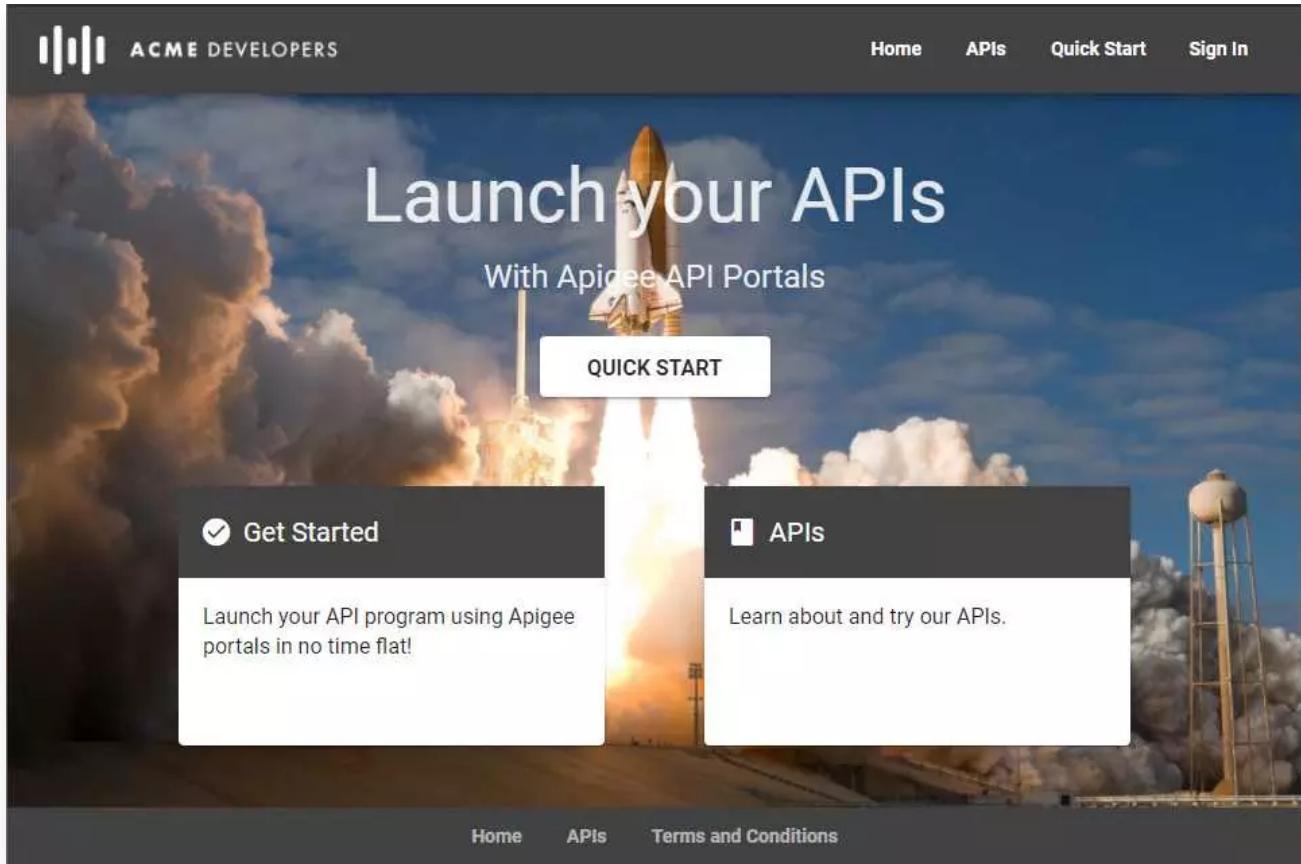
1. In Apigee, navigate to **Publish > Portals** from the left menu. Then Click **+ Portal**.
2. Enter a name, e.g. **ourBank Develop**



3. Select **APIs** from the dropdown menu at the top.
4. In the “Publish an API product” page, click on **Get started**.
5. Select the API Product (*ourBank API*) and click **Next**.



6. Click **Next >** **Next >** **Finish**.
7. Click **Live Portal** at the top-right corner to test the new developer portal.



Task 4 - Route traffic from mock response to real backend

Deploy the real backend on Cloud Function

Go the Cloud Shell, run the commands given in the lab.

```
# Make subdirectories for the code
mkdir bank-verification-service; cd bank-verification-service

# Download the code
wget https://storage.googleapis.com/apigee-quest/code/index.js
wget https://storage.googleapis.com/apigee-quest/code/package.json

# Deploy card verification function
gcloud functions deploy verifyCard --runtime nodejs10 --trigger-http --
allow-unauthenticated
```

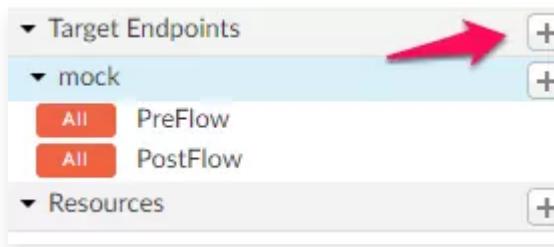
```
# Deploy address verification function
gcloud functions deploy verifyAddress --runtime nodejs10 --trigger-http --allow-unauthenticated
```

Route traffic to these backends based on the incoming requests

1. Go back to Apigee, navigate to **API Proxies > Verification-API-v1**, click the **Develop** tab in the top right.
2. To rename the mock API endpoint, click on **Target Endpoints → default** in the Navigator tab. Replace `<TargetEndpoint name="default">` to .

```
Code default
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <TargetEndpoint name="mock">
3    <PreFlow name="PreFlow">
4      <Request/>
5      <Response>
6        <Step>
7          <Name>add-cors</Name>
8        </Step>
9      </Response>
10     </PreFlow>
11   <Flows/>
12   <PostFlow name="PostFlow">
13     <Request/>
14     <Response/>
15   </PostFlow>
16   <HTTPTargetConnection>
17     <URL>https://mocktarget.apigee.net/json</URL>
18   </HTTPTargetConnection>
19 </TargetEndpoint>
```

3. To add a new target endpoint, click on the plus sign (+) next to **Target Endpoints** in the Navigator tab.



4. Enter the following:
 - **Target Endpoint Name:** e.g. `cloud`
 - **HTTP Target:** _The Cloud Function Endpoint (e.g. `https://-.cloudfunctions.net_`)

New Target Endpoint

Target Endpoint Name:

HTTP Proxy Chaining Path Chaining NodeJS

HTTP Target:

5. Click on **Proxy Endpoints** → **default** in the Navigator tab. Remove the **Assign Message-1** from the Response pipeline.
6. Click on **Target Endpoints** → **mock** → **PreFlow** in the Navigator tab. Remove the **Assign Message-1** from the Response pipeline. Drag the **Assign Message-1** from **Policies** in the Navigator tab to the Response pipeline.

The screenshot shows the Apigee API Designer interface. On the left, the Navigator tab displays the API structure under 'Verification-API-v11'. It includes sections for Policies (Add CORS, Assign Message-1, Extension Callout-1, Verify API Key-1), Proxy Endpoints (default, cloud, mock), Target Endpoints (cloud, mock), and Resources. In the Flow tab, a sequence of REQUEST and RESPONSE steps is shown. A red arrow highlights the 'Add CORS' policy from the Navigator being dragged into the Response pipeline.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TargetEndpoint name="mock">
<PreFlow name="PreFlow">
<Step>
<Name>add-cors</Name>
</Step>
<Step>
<Name>Assign-Message-1</Name>
</Step>
</Response>
</PreFlow>
<Flows/>
<PostFlow name="PostFlow">

```

7. Click on **Target Endpoints** → **cloud** → **PreFlow** in the Navigator tab. Drag the **Add CORS** from **Policies** in the Navigator tab to the Response pipeline.
8. Click on **Proxy Endpoints** → **default** → **PreFlow** in the Navigator tab. Replace the **RouteRule** tag with the following code,

```

<RouteRule name="mock">
<Condition>request.queryparam.mock = "true"</Condition>
<TargetEndpoint>mock</TargetEndpoint>
</RouteRule>
<RouteRule name="default">
<TargetEndpoint>cloud</TargetEndpoint>
</RouteRule>

```

The screenshot shows the XML code editor with the following code:

```

<VirtualHost>
<ProxyEndpoint>
<VirtualHost>secure</VirtualHost>
<VirtualHost>default</VirtualHost>
</HTTPProxyConnection>
<RouteRule name="mock">
<Condition>request.queryparam.mock = "true"</Condition>
<TargetEndpoint>mock</TargetEndpoint>
</RouteRule>
<RouteRule name="default">
<TargetEndpoint>cloud</TargetEndpoint>
</RouteRule>
</ProxyEndpoint>

```

The RouteRule tag for the 'mock' endpoint is highlighted with a blue selection bar.

9. Click **Save**.

Testing

Test the real backend routing using the `curl` statement below.

```
curl -X POST \
  https://${APIEE_ORG}-test.apigee.net/verification-api-v1/verifyCard?
apikey=${APIKEY} \
-H 'cache-control: no-cache' \
-H 'content-type: application/json' \
-H 'postman-token: 89236919-eabe-4357-e4c4-079f20ecd798' \
-d '{
  "number": "2221005276762844",
  "cvv": "345",
  "expiration": "10/2025"
}'
```

Expected output:

```
{"valid":true, "message":"credit card is valid"}
```

Test the mock response routing using the `curl` statement below.

```
curl -X POST \
  'https://${APIEE_ORG}-test.apigee.net/verification-api-v1/verifyCard?
mock=true&apikey=${APIKEY}' \
-H 'cache-control: no-cache' \
-H 'content-type: application/json' \
-H 'postman-token: 89236919-eabe-4357-e4c4-079f20ecd798' \
-d '{
  "number": "2221005276762844",
  "cvv": "345",
  "expiration": "10/2025"
}'
```

Expected output:

```
{"valid":true, "message":"mock response"}
```

Congratulations!



Chris F. [Follow](#) Author of this blog, M.Phil.