Qlog: Build and Manage APIs with Apigee: Challenge Lab

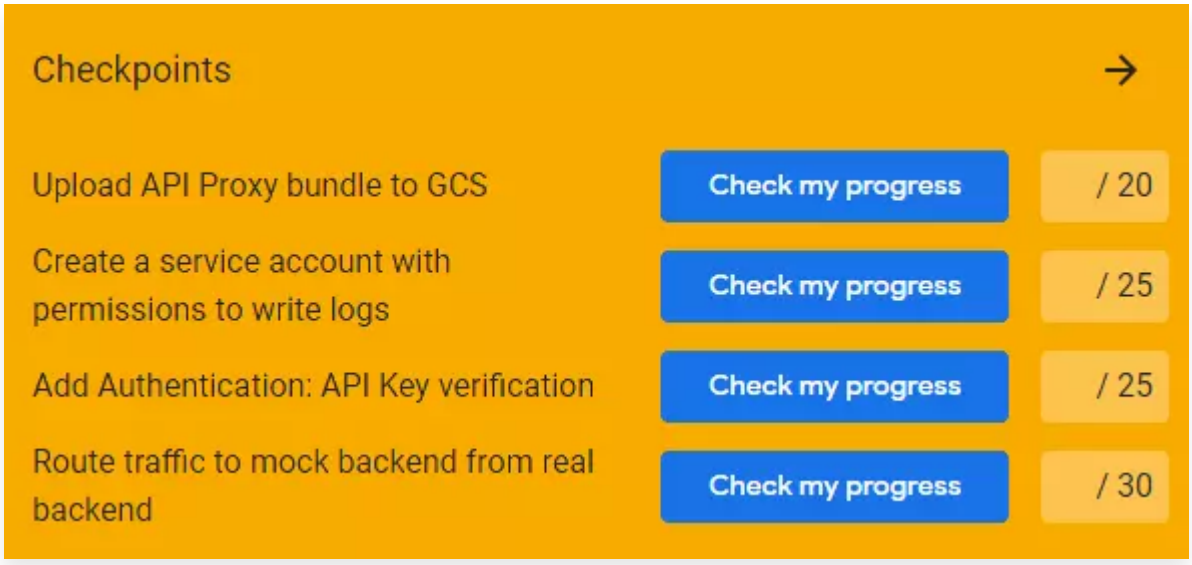
13 Sep 2020

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

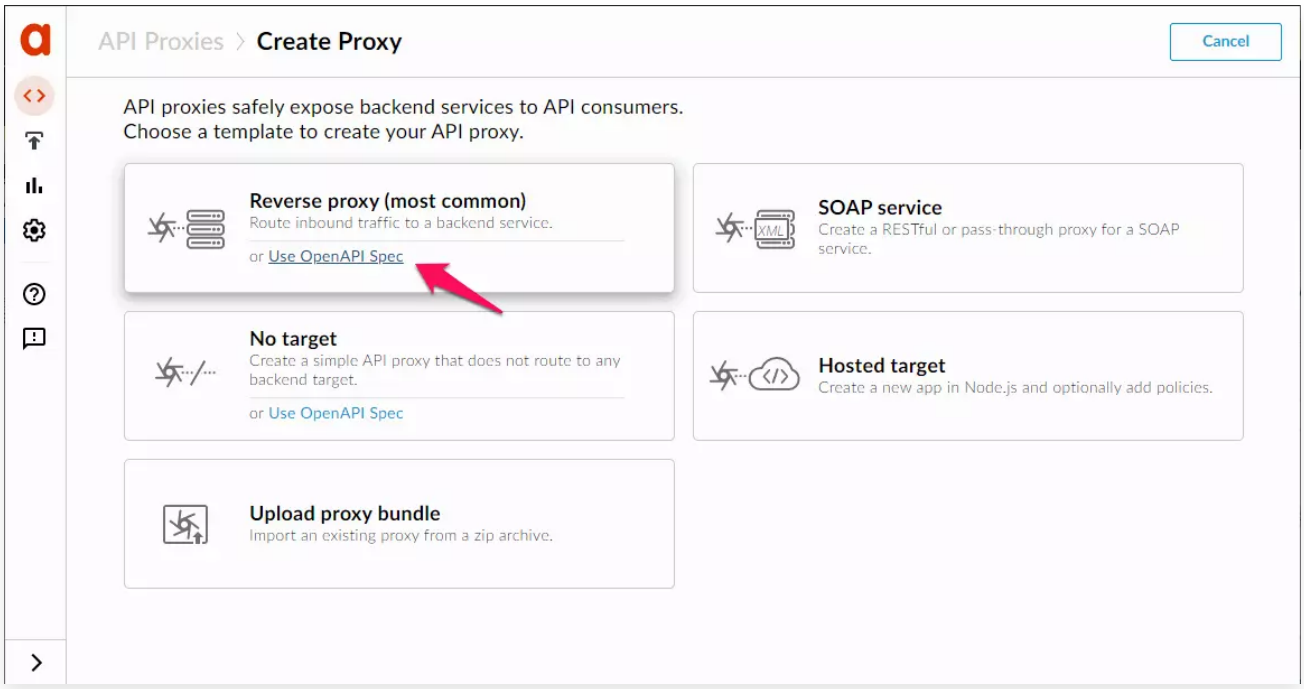
The challenge contains 4 required tasks:



Task 1 - Create API Specification and Generate an API Proxy

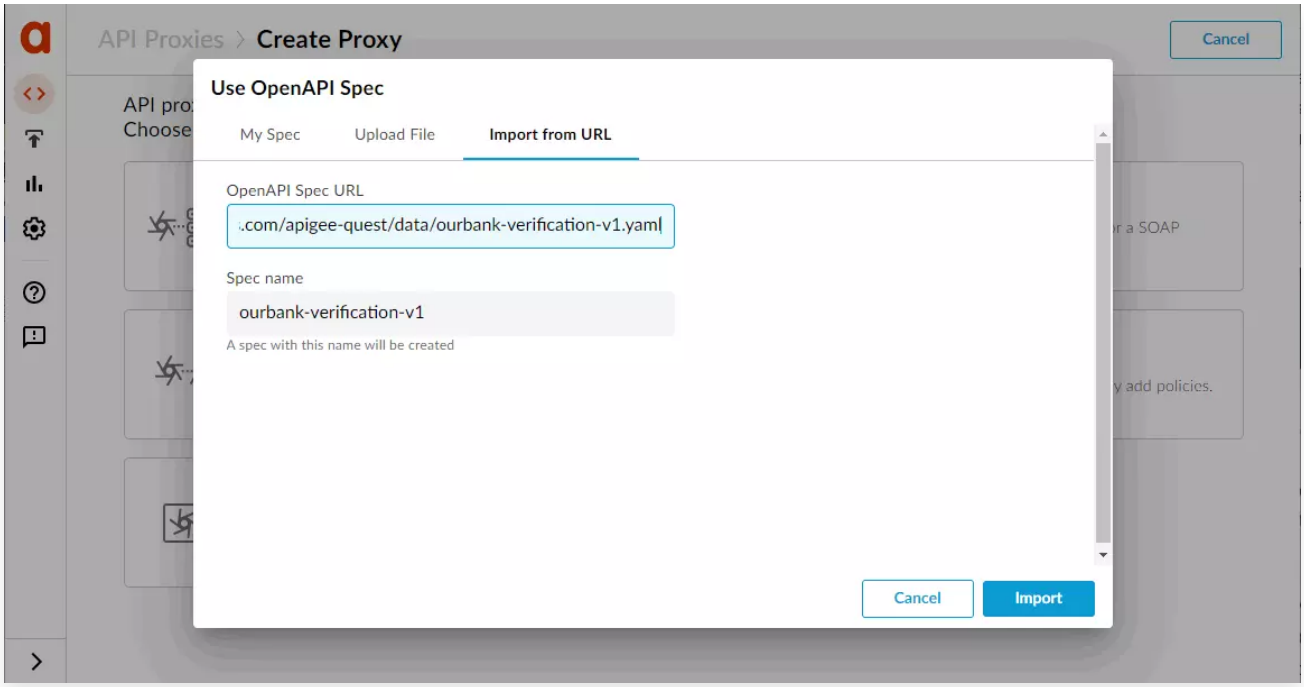
Define a RESTful API in Apigee using an API specification

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:

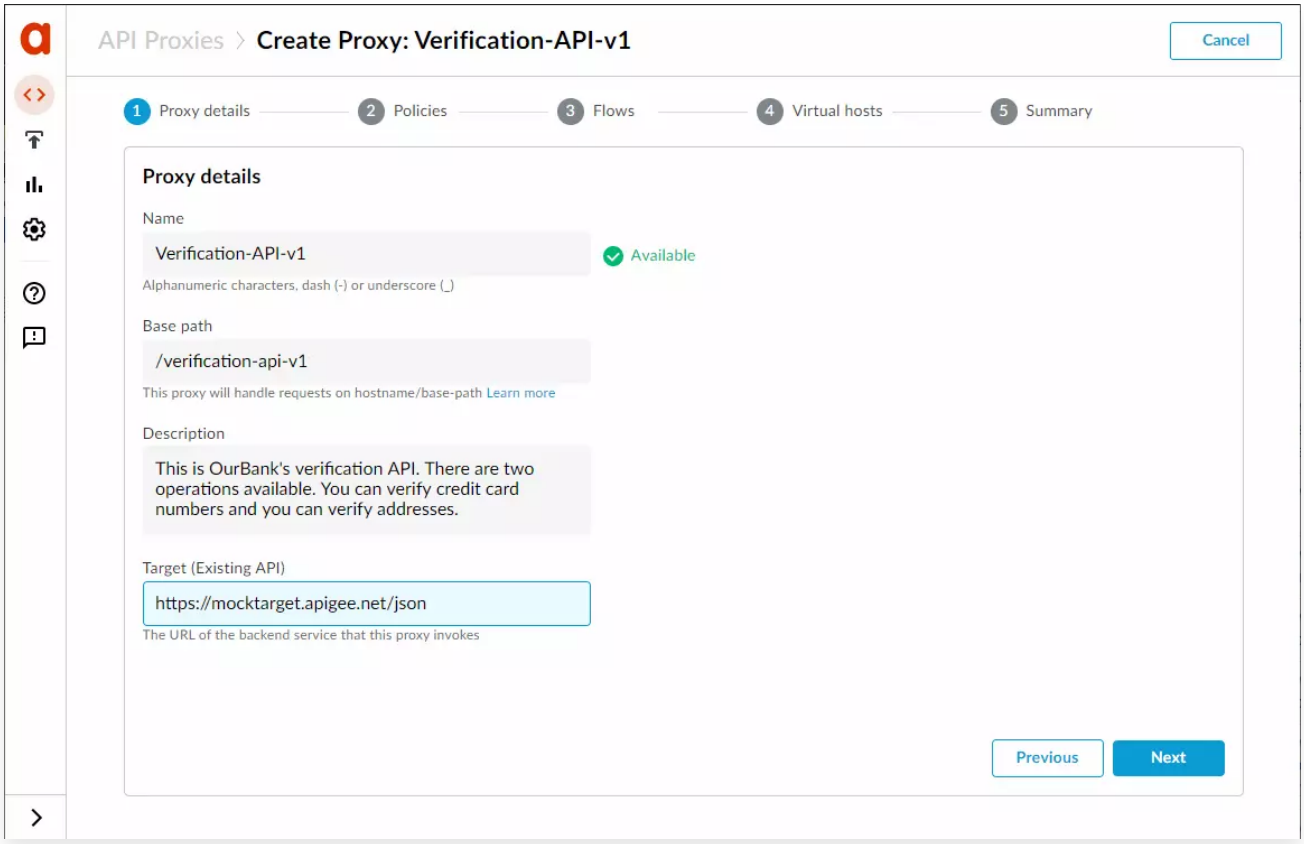
    - **OpenAPI Spec URL**: `https://storage.googleapis.com/apigee-quest/data/ourbank-verification-v1.yaml`
    - **Spec name**: *e.g.* `ourbank-verification-v1`



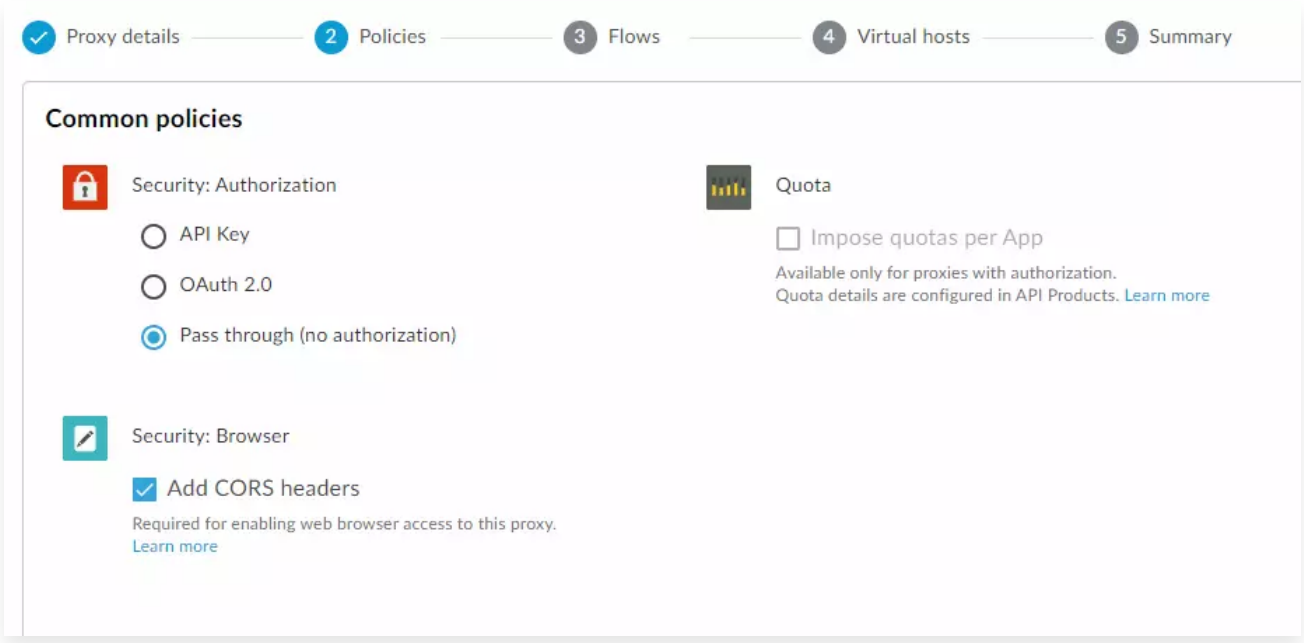
6. Click **Import**.
7. Find the "View JSON response" endpoint URL of Mock Target API from this Apigee Doc page, which is:

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

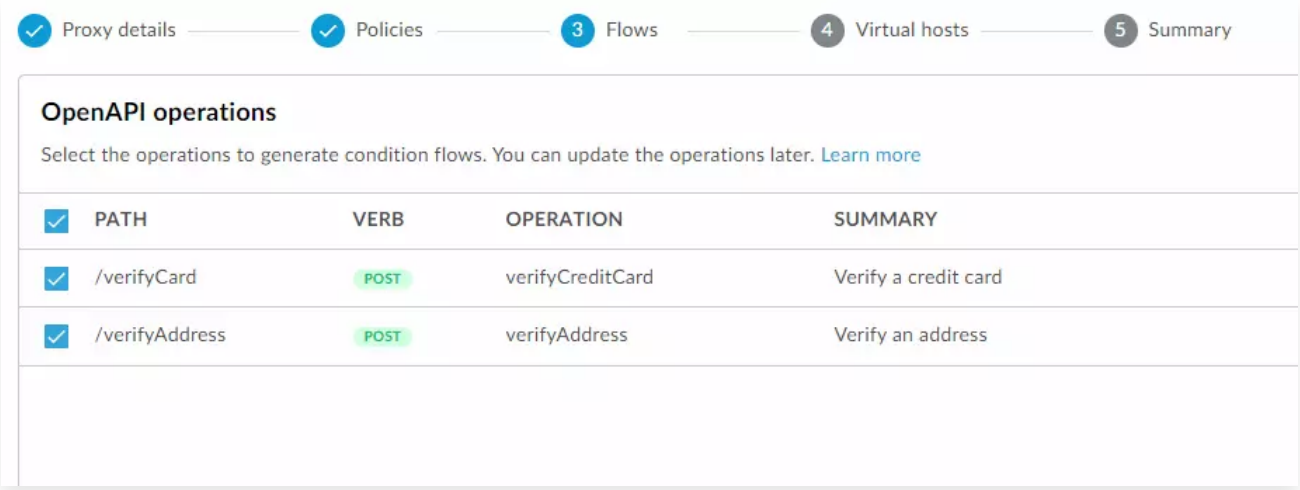
8. Copy the URL to the field **Target (Existing API)**, then click **Next**.



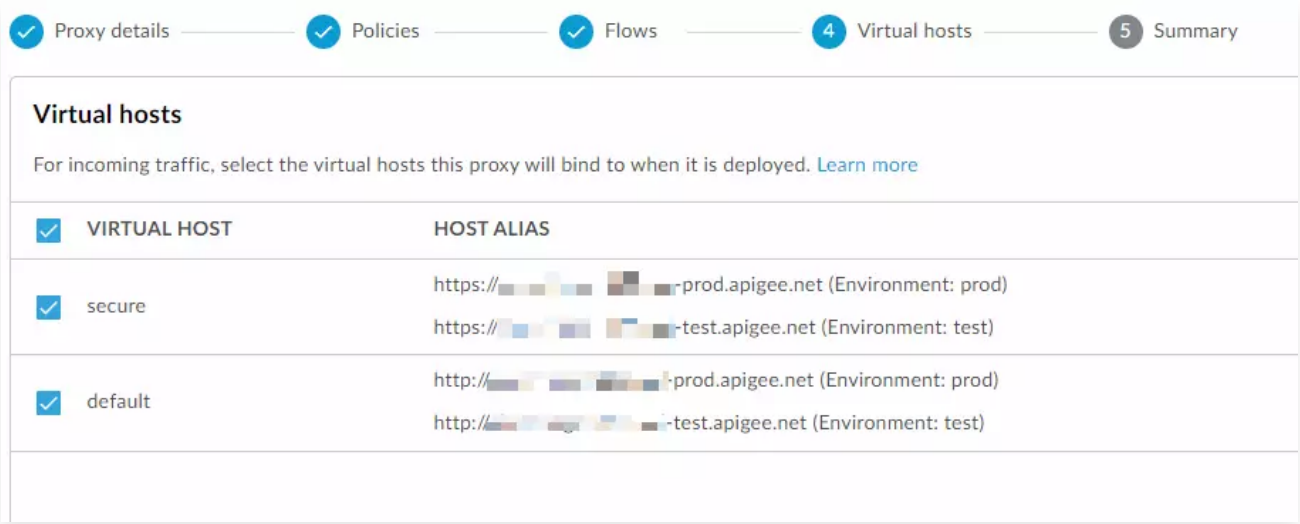
9. Check **Add CORS header** to enable CORS headers in Apigee, then click **Next**.



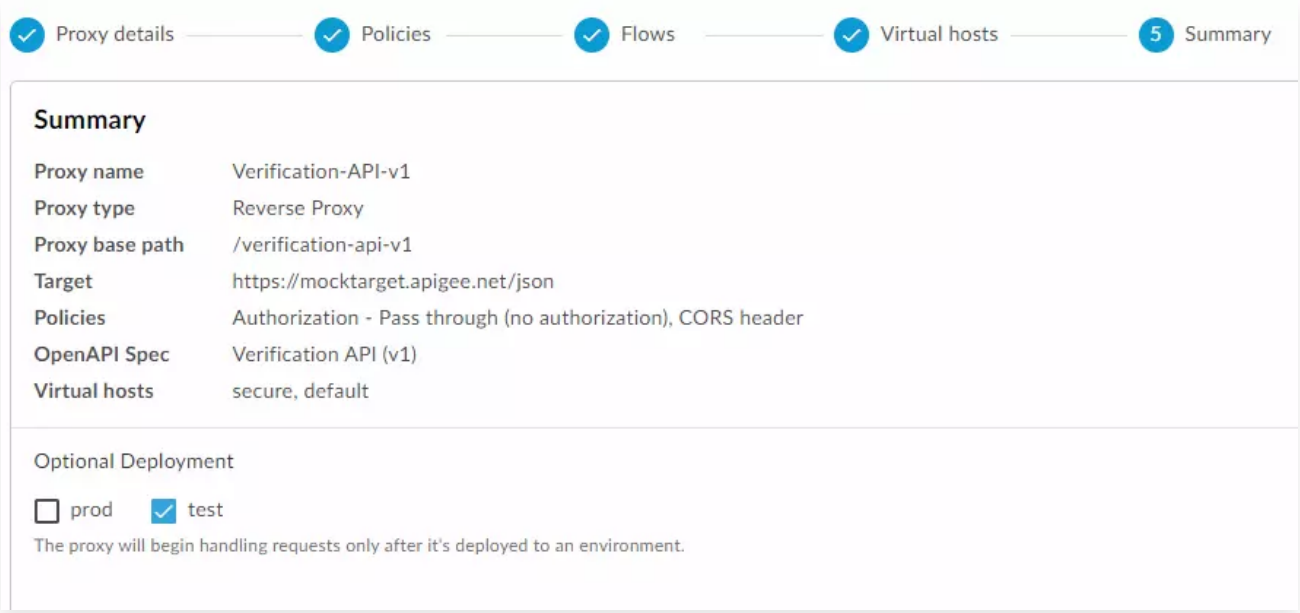
10. Click **Next** to continue.



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



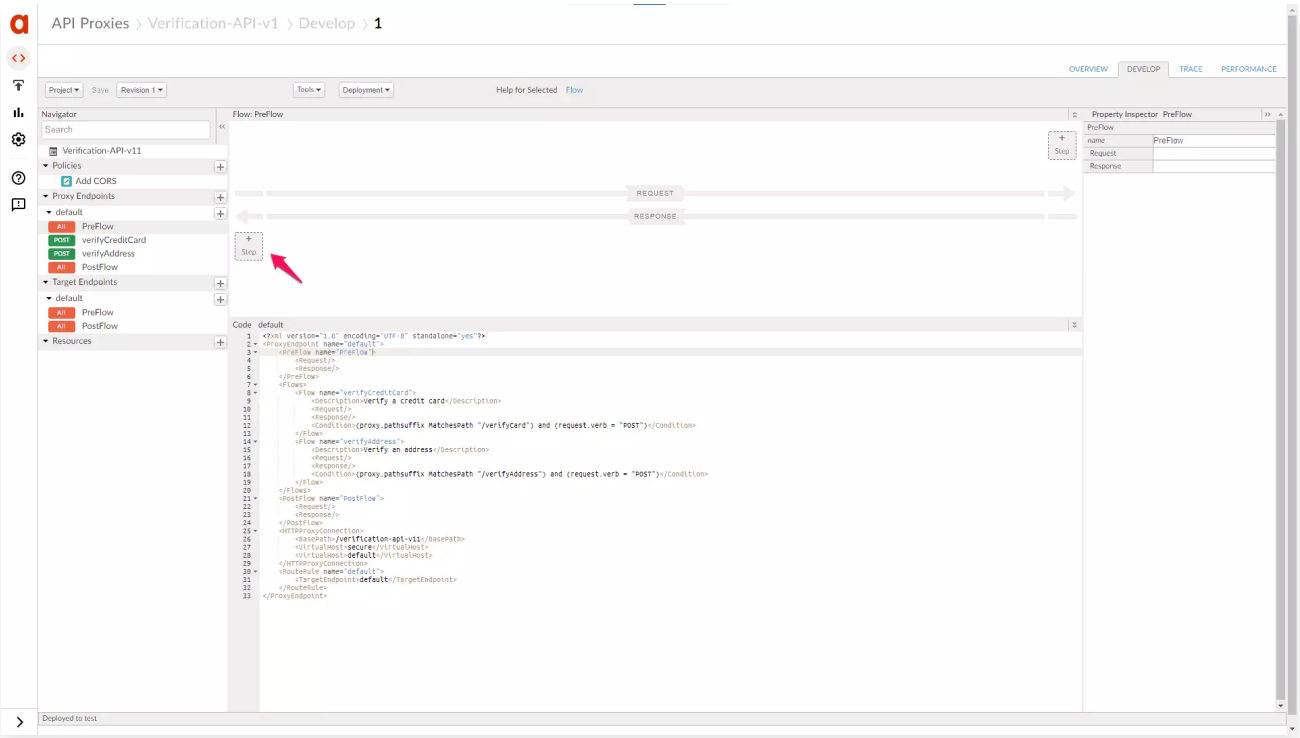
12. Check the box next to **default**, then click **Create and deploy**.



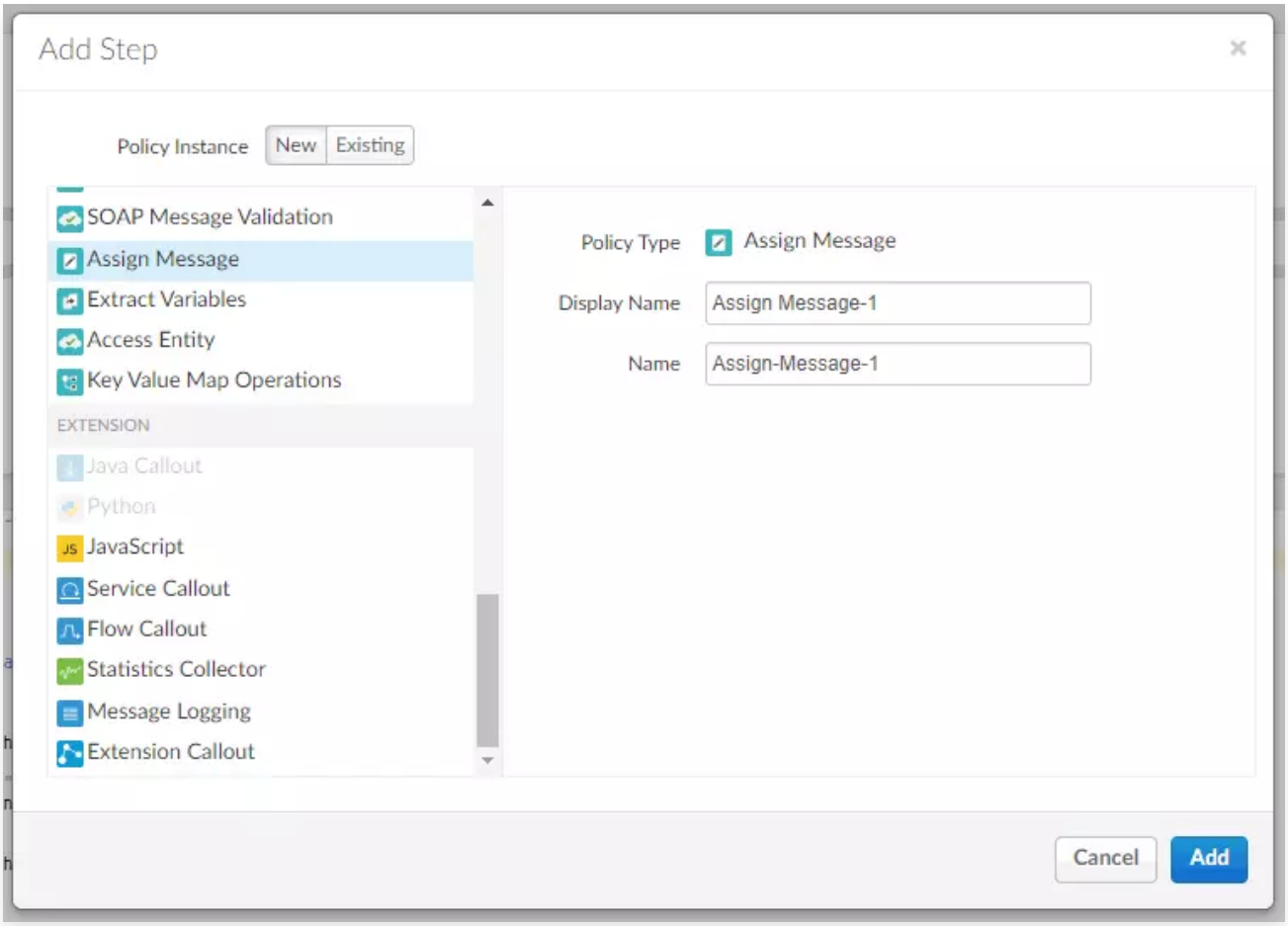
Apigee now deploys the API proxy into your test environment. Click **Edit proxy** to view the deployed proxy.

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.



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

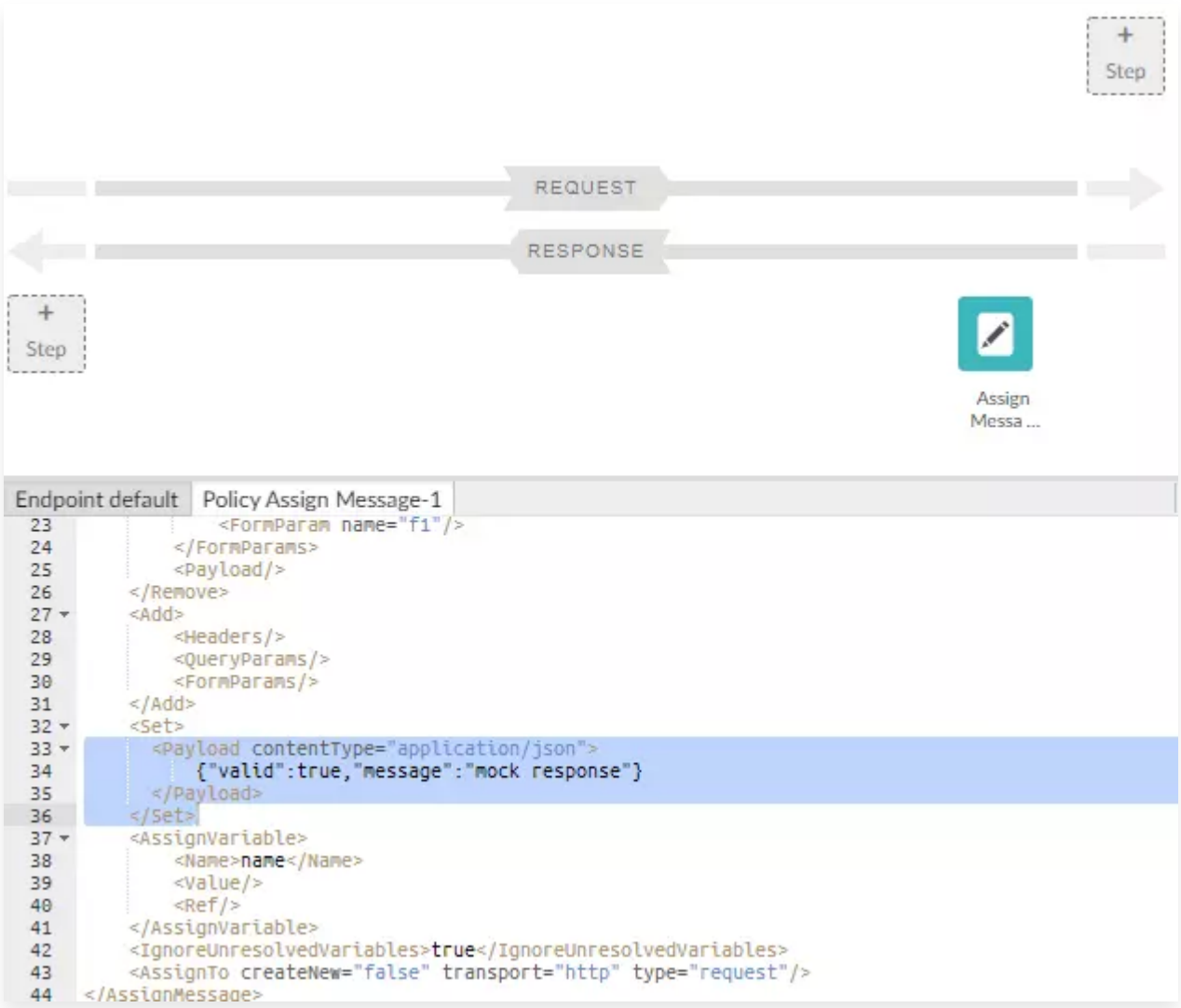


4. Modify Policy Assign Message-1:

- 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

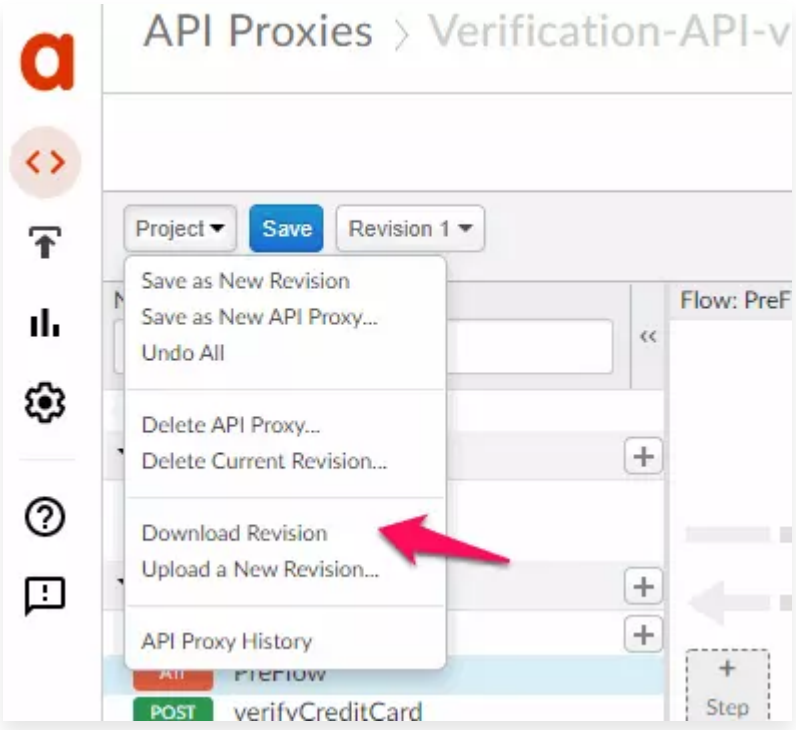
The deployment can be tested using the following curl statement in the Cloud Shell:

```
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.

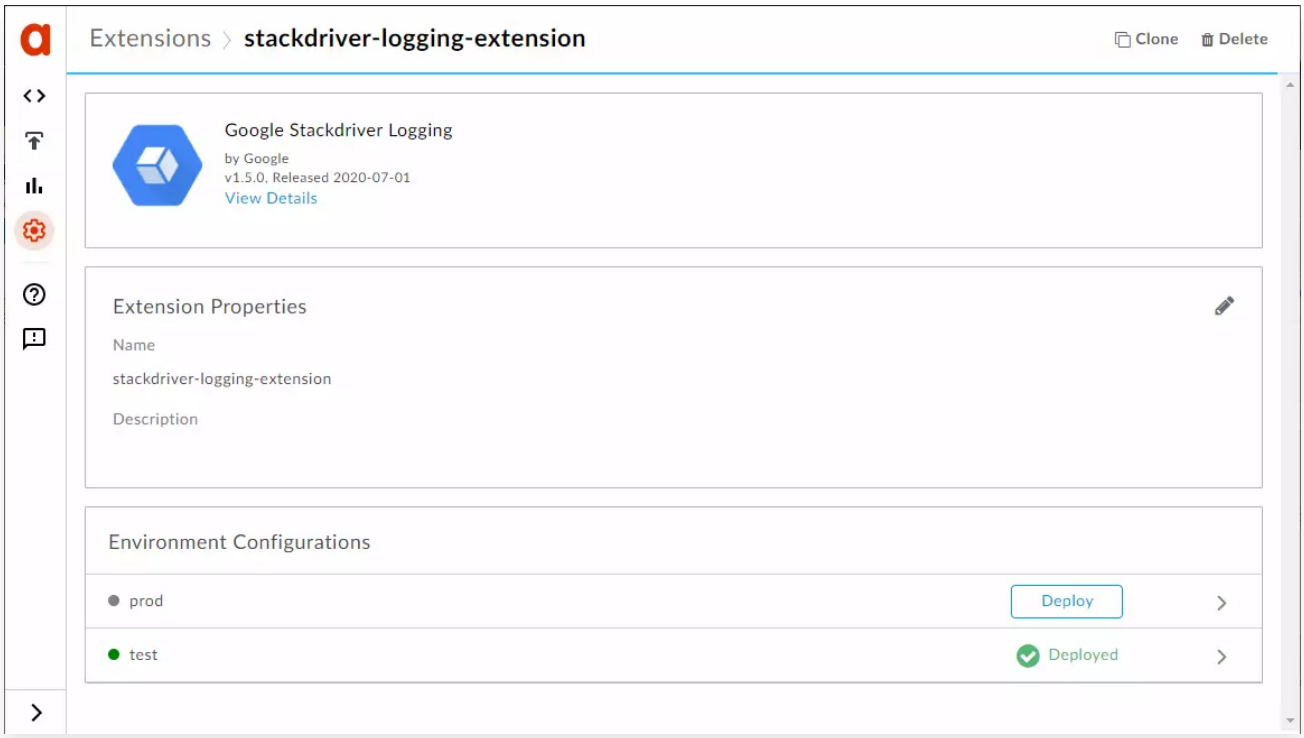
Create an extension and deploy it to test the environment using this service account

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.

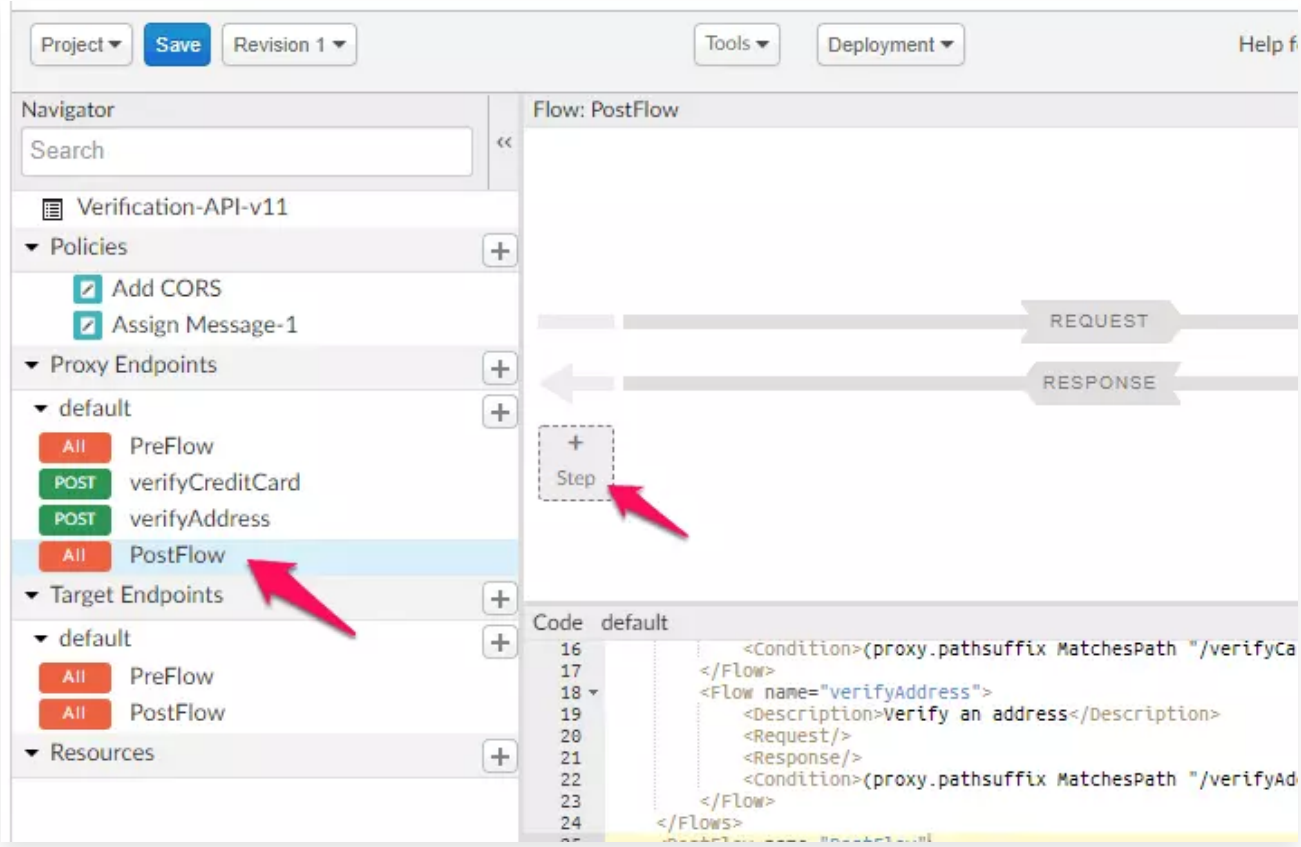
6. Click **Save**.
7. Once the configuration is saved, click on the **Deploy** button for the Test environment.



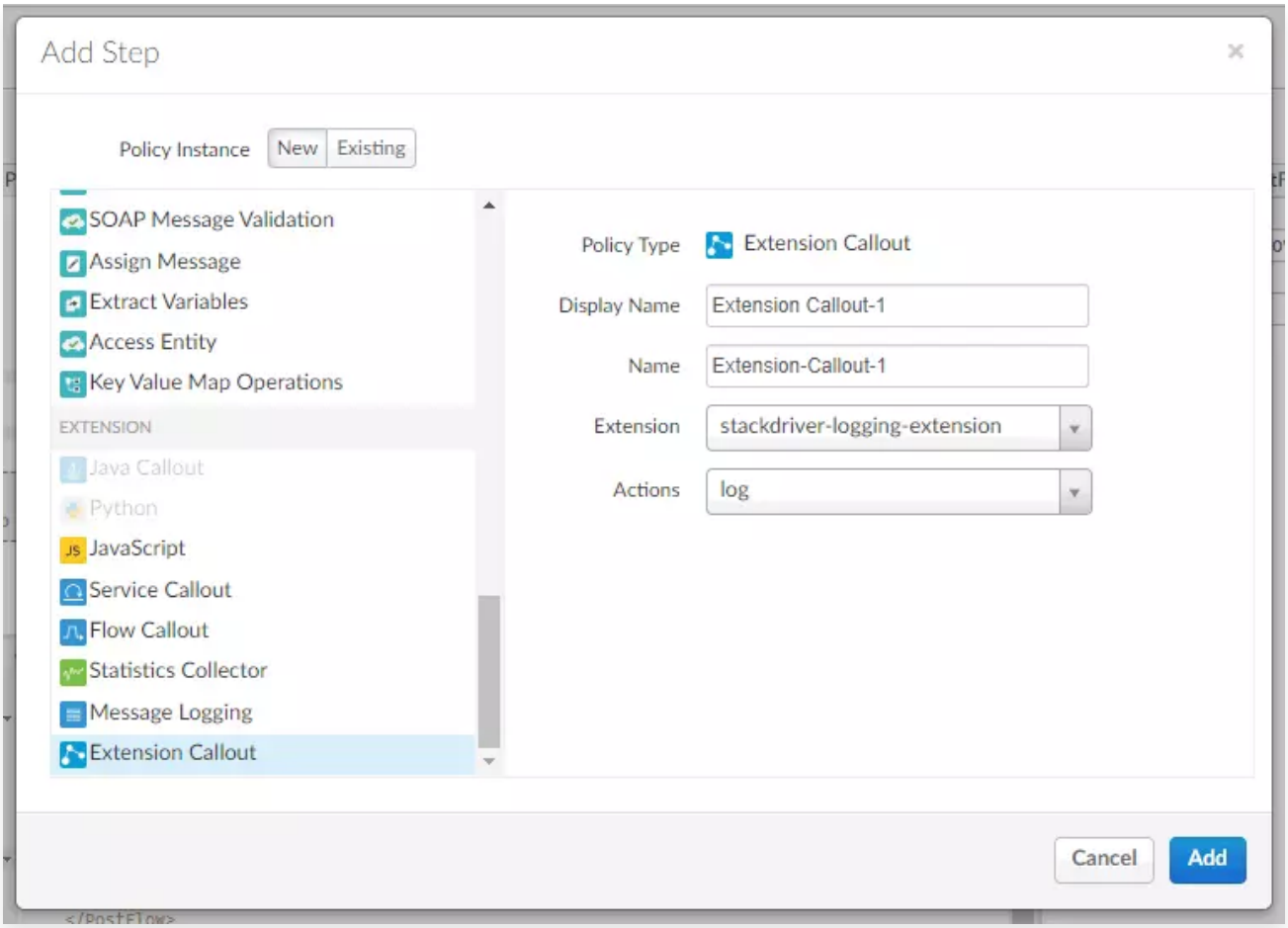
Create an Extension policy in the PostFlow response path

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.



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**.

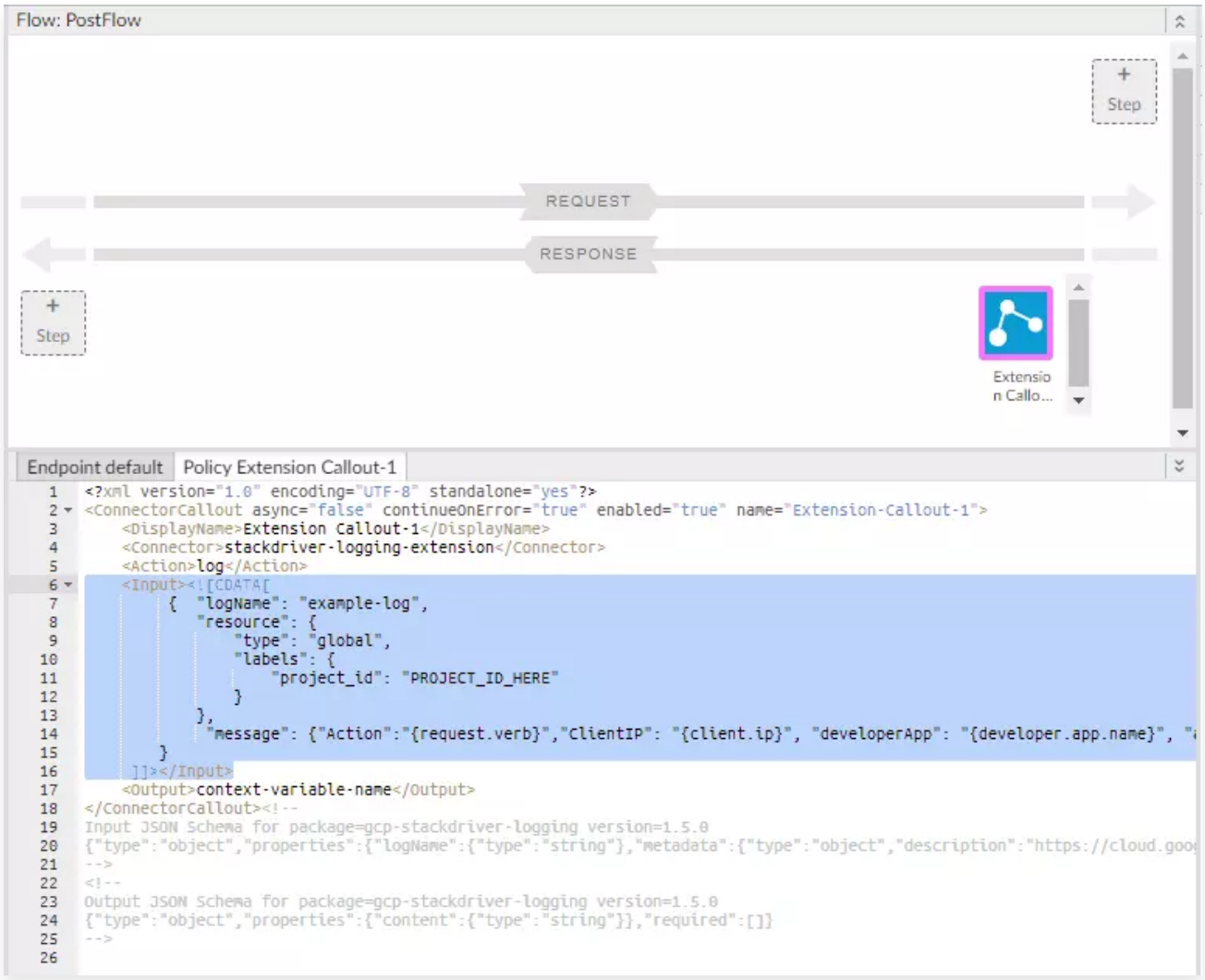


6. Modify Policy Extension Callout-1:

   - 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.

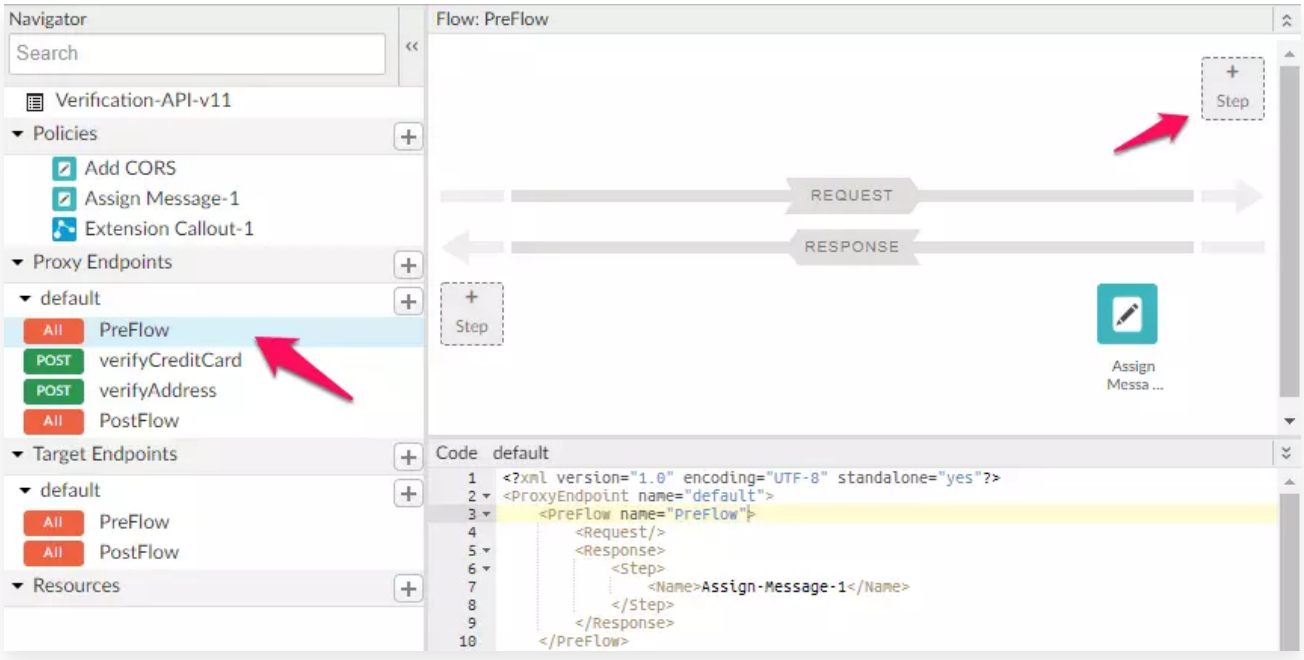
```
<Input><![CDATA[
   {  "logName": "example-log",
      "resource": {
          "type": "global",
          "labels": {
              "project_id": "PROJECT_ID_HERE"
          }
      },
       "message": {"Action":"{request.verb}","ClientIP": "
{client.ip}", "developerApp": "{developer.app.name}",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

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

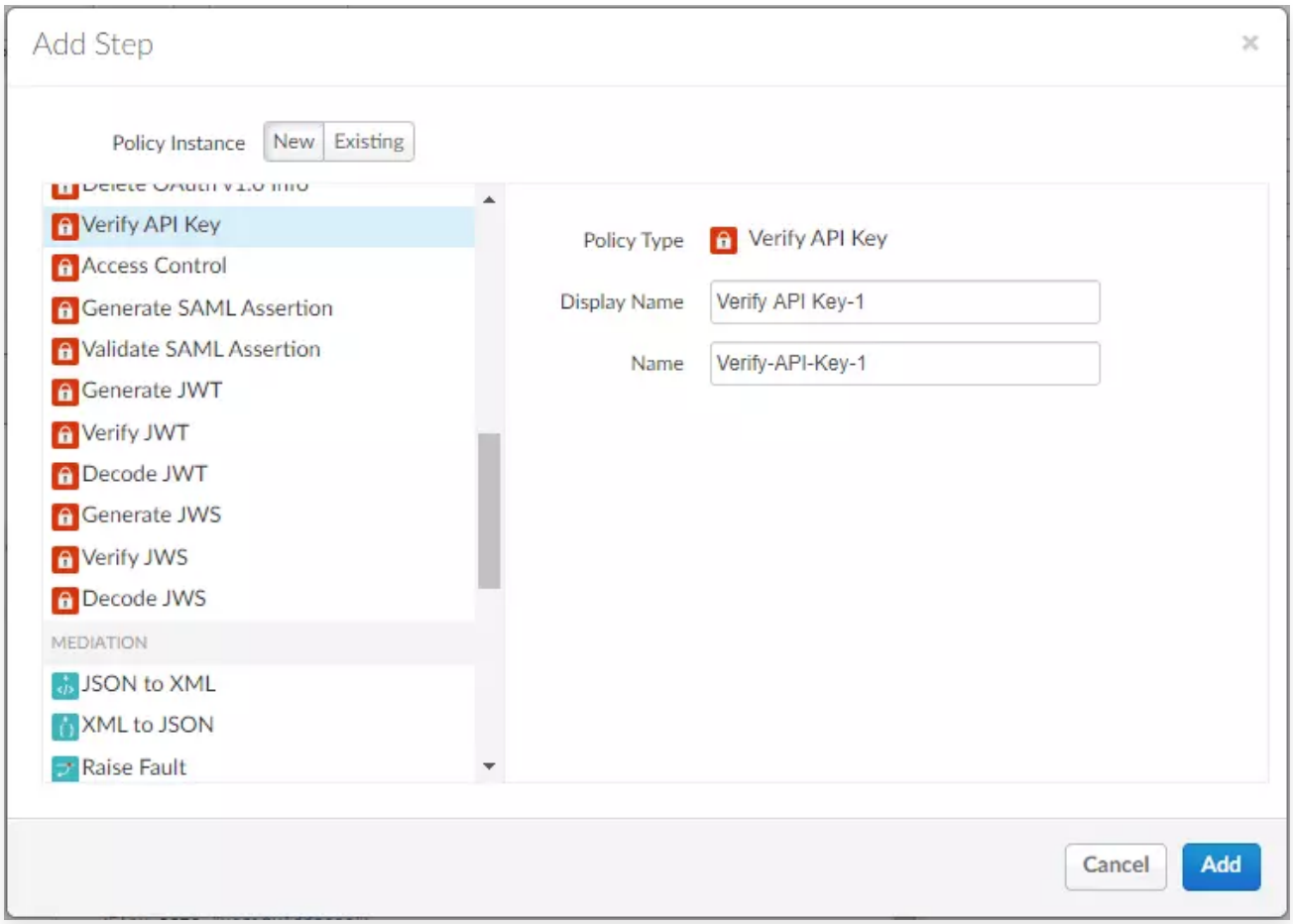


Add API Key verification

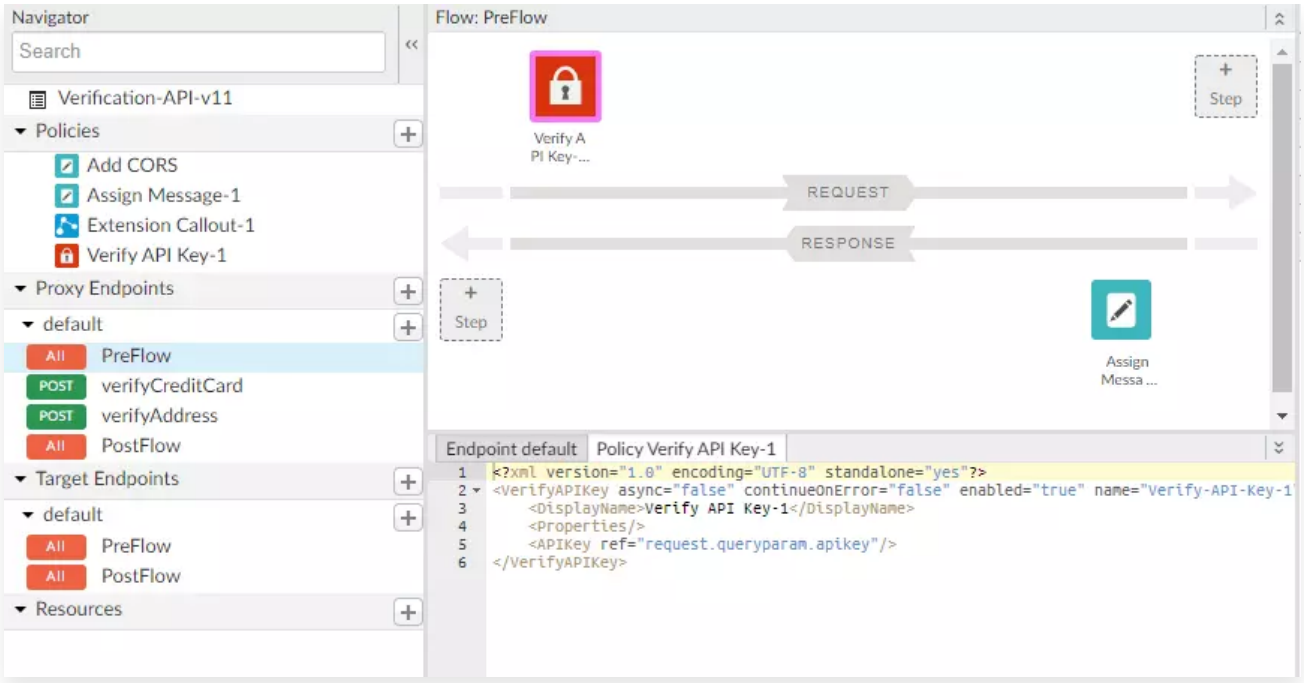
1. To add a policy to your proxy, click on **Proxy Endpoints** → **PreFlow** in the Navigator tab, then in the *Response* pipeline, click **+ Step** to add a step.



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.

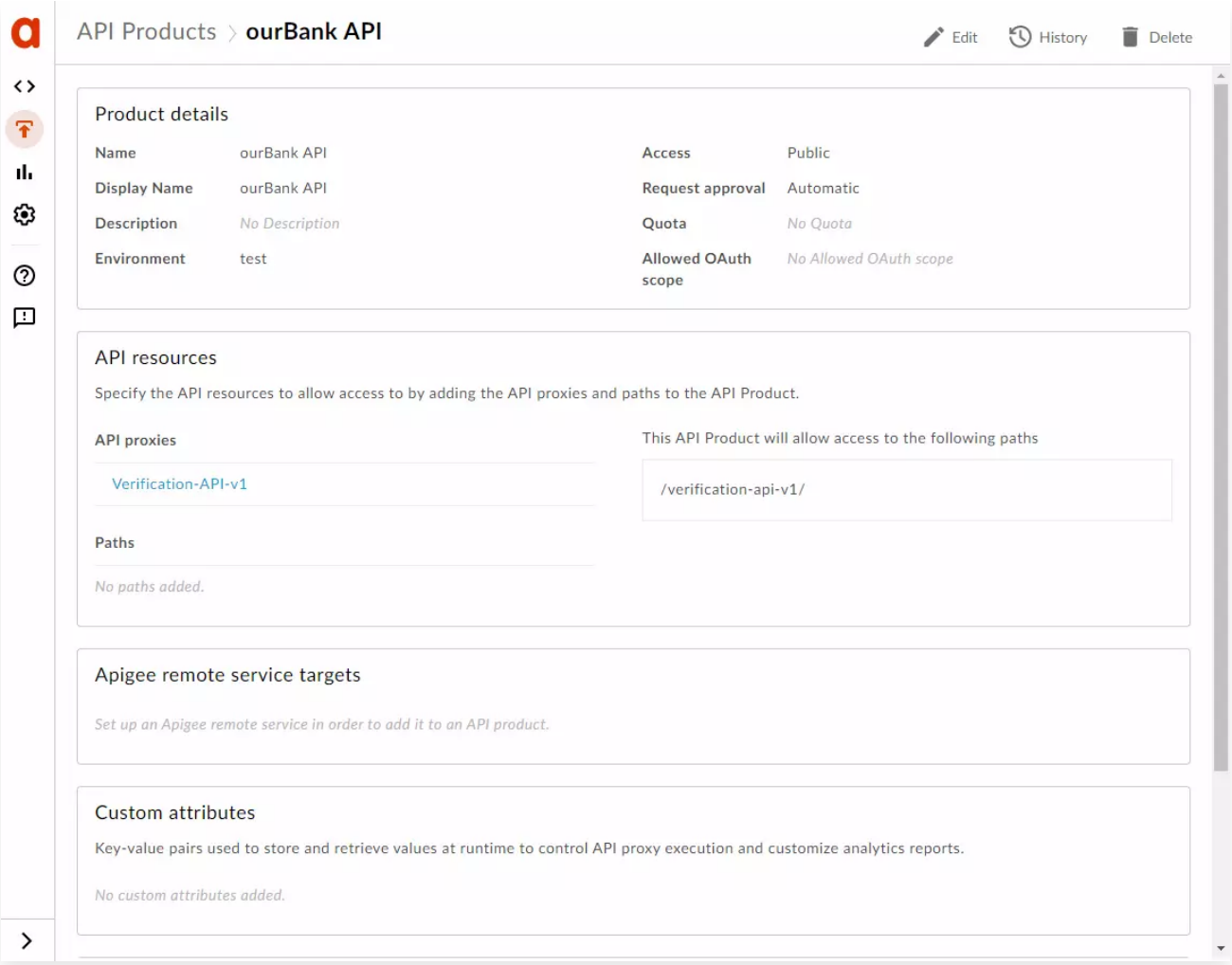


Create an API Product and an App

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)**.

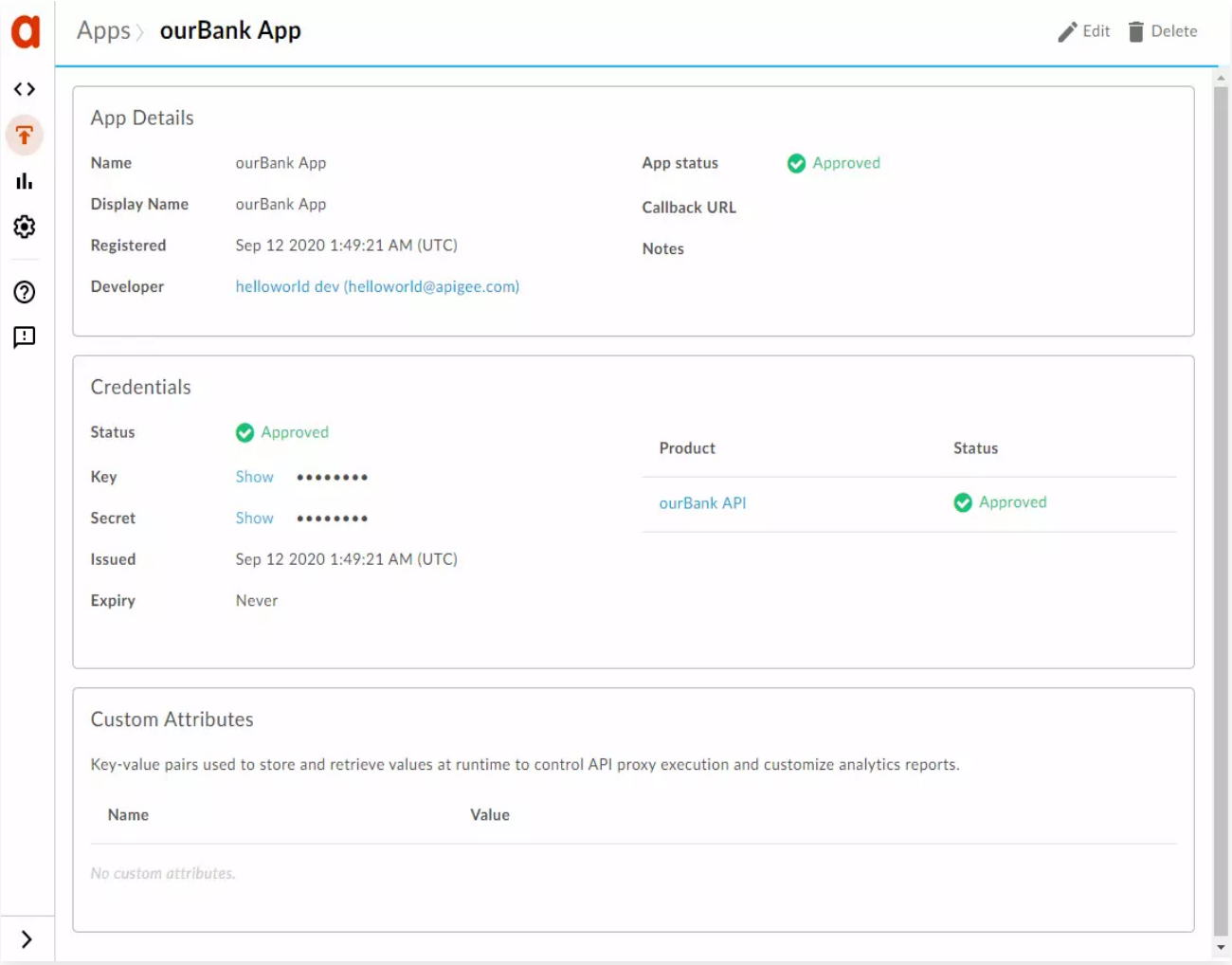
    The settings should look as follows:



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**.



9. 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:

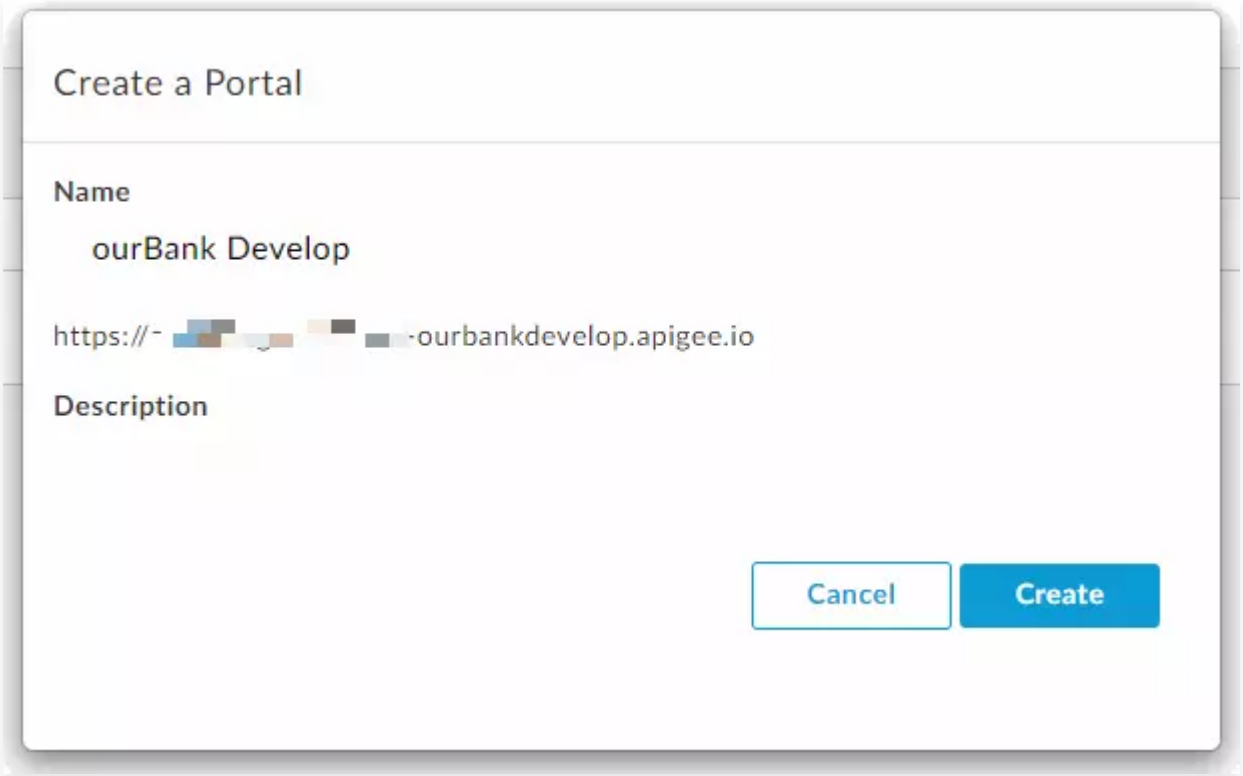
```
{"valid":true,"message":"mock response"}
```

Error response message:

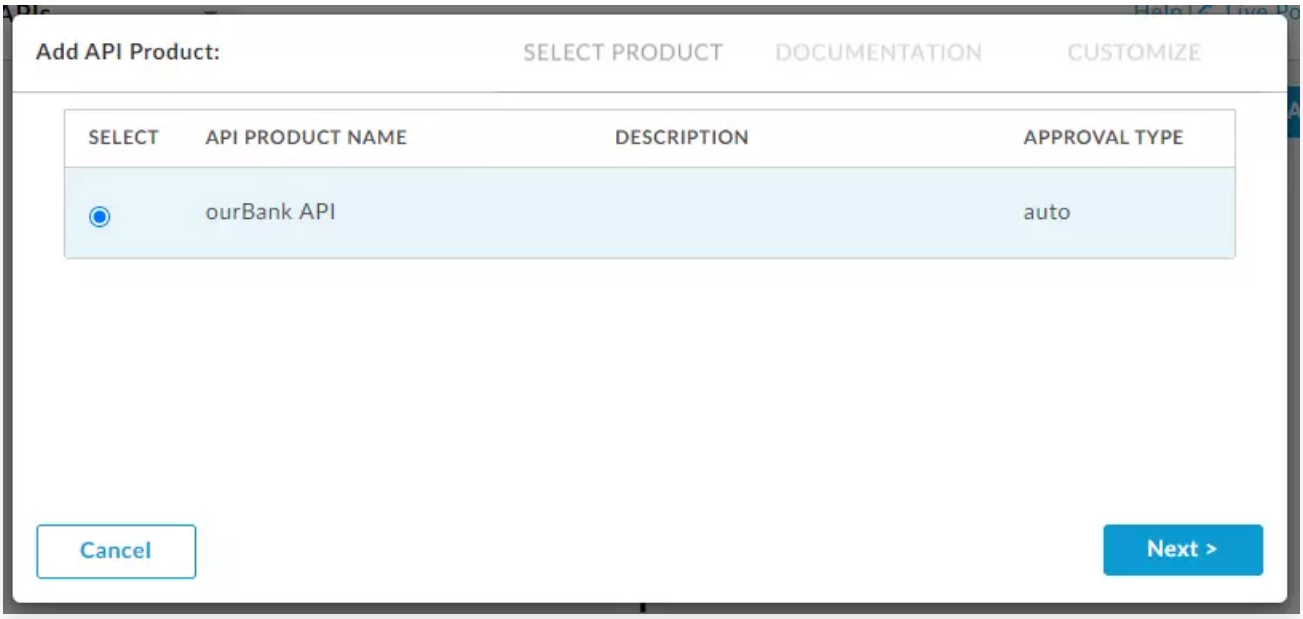
```
{"fault":{"faultstring":"Invalid ApiKey","detail":
{"errorcode":"oauth.v2.InvalidApiKey"}}}
```

Task 3 - Create a Developer Portal for consuming the APIs

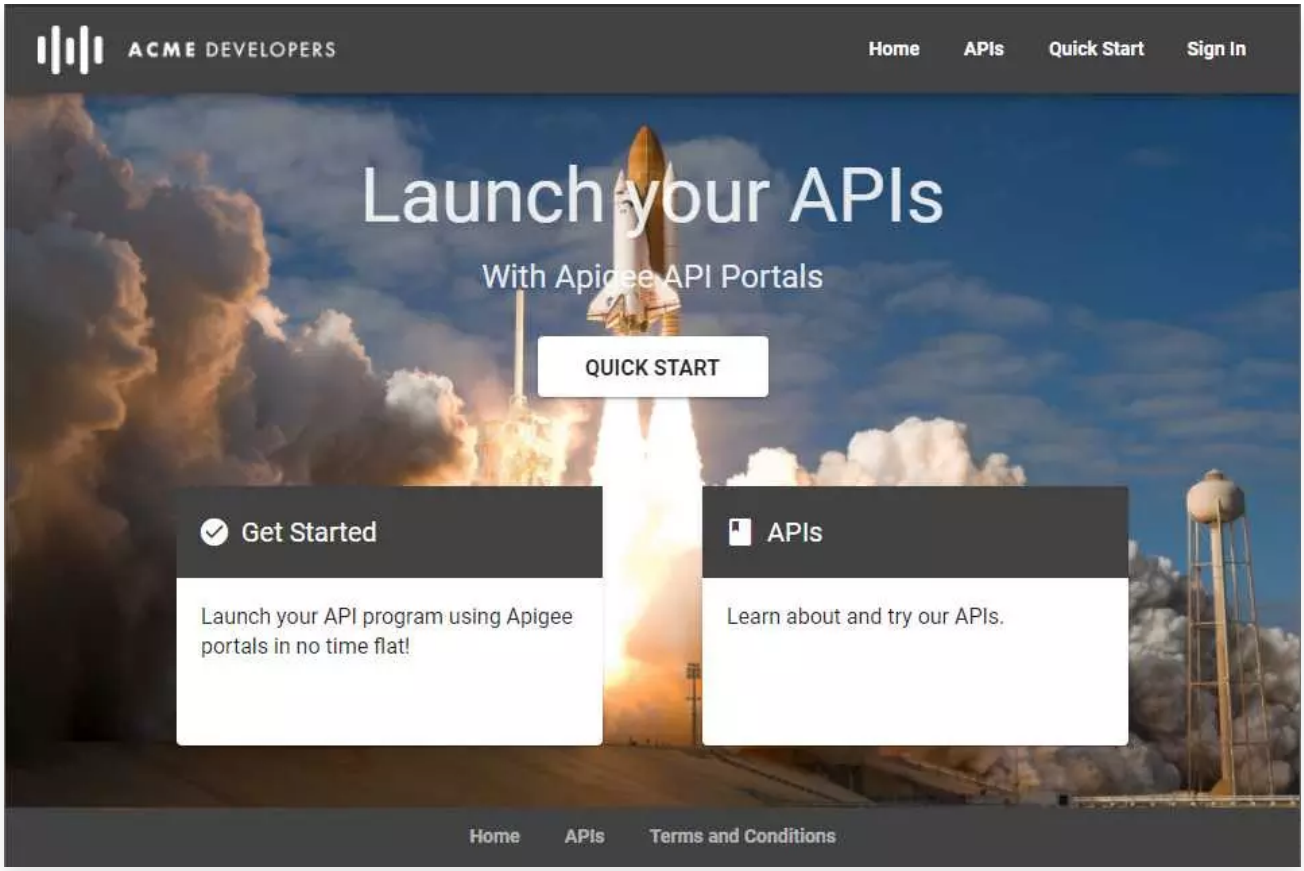
1. In Apigee, navigate to **Publish** > **Portals** from the left menu. Then Click **+ Portal**.
2. Enter a name, *e.g.* `ourBank Develop`



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

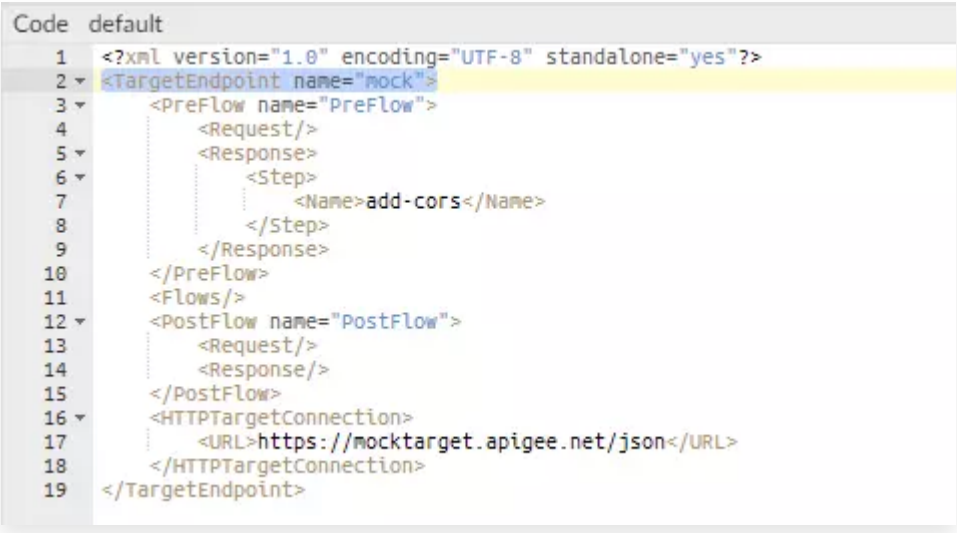
Go the Cloud Shell, run the commands given in the lab.

```
   # 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
```

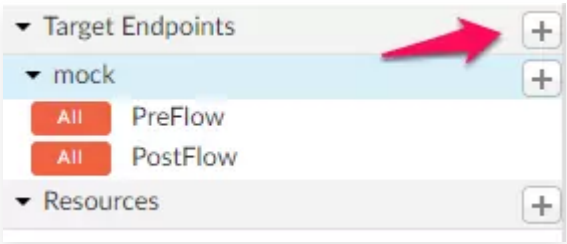
```
  # Deploy address verification function
  gcloud functions deploy verifyAddress --runtime nodejs10 --trigger-http --
allow-unauthenticated
```

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 .

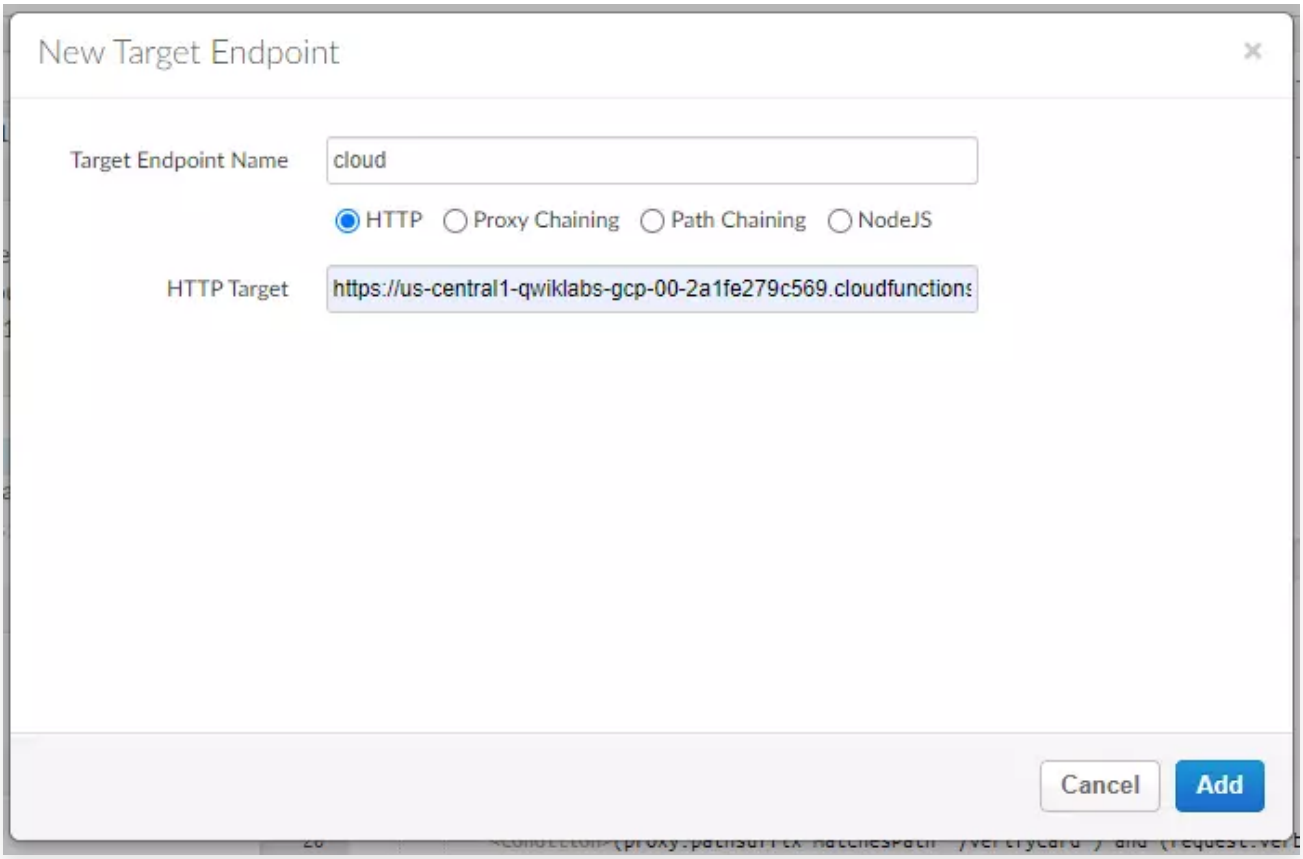


3. To add a new target endpoint, click on the plus sign (**+**) next to **Target Endpoints** in the Navigator tab.

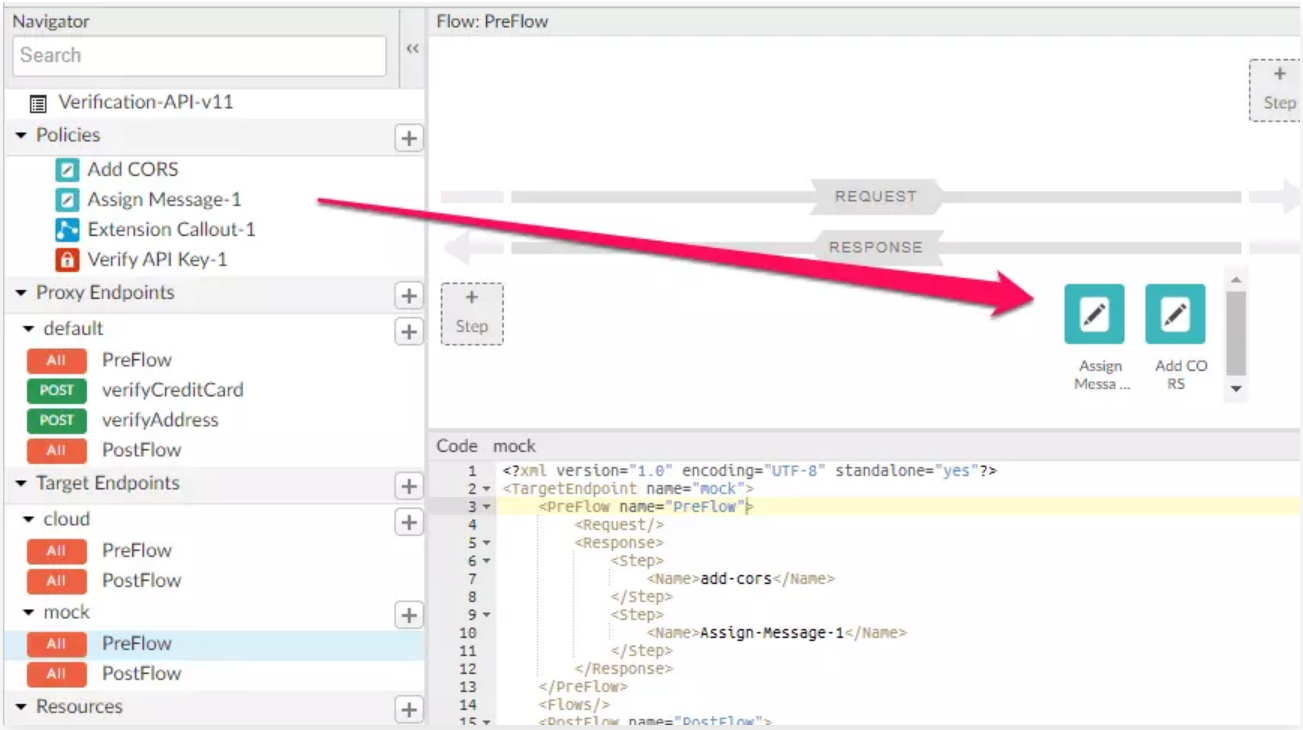


4. Enter the following:

    - **Target Endpoint Name**: *e.g.* `cloud`
    - **HTTP Target**: _The Cloud Function Endpoint (e.g. https://-.cloudfunctions.net)_

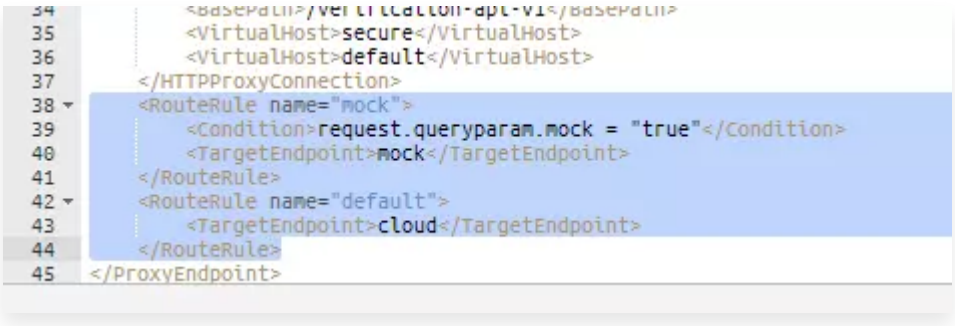


5. Click on **Proxy Endpoints** → **default** in the Navigator tab. Remove the **Assign Message-1** from the Response pipeline.
6. Click on **Target Endpoints** → **mock** → **PreFlow** in the Navigator tab. Remove the **Assign Message-1** from the Response pipeline. Drag the **Assign Message-1** from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



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>
```



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"}
```

Apigee | Google Cloud | Logbook | Qwiklabs