Lab 1

Proxy an existing Omni Channel API
Overview

API management is essential to an API-Led architecture as it provides a governance framework to your APIs in all three layers. For API Management to take place, we need to be able to host our APIs, both new and existing, on an API Gateway that will be used for enforcing policies and collecting data for analytics.

MuleSoft can apply governance directly to a MuleSoft implemented API or through a proxy gateway for these and other external existing APIs.

The APIs we will use are two versions of the Omni Channel API. You will create a proxy gateway to an already deployed mock implementation in this Lab 1.

In the optional Lab 4 we show how you can directly govern a MuleSoft service without a proxy, and the optional Lab 5 shows how you can proxy and govern existing services that were not developed in Design Center and published to Exchange.

In this lab, we will define an API that will act as a proxy for Mythical Corporation’s Omni Channel API. The proxy will be deployed to Anypoint Platform’s API Gateway, which is powered by the Mule runtime. The Mule run-time can run on-premise, on a private cloud or on the CloudHub service hosted by Anypoint Platform. In this scenario, we will be use the latter, CloudHub.

Clients will access the API through the API Gateway which will then forward the requests to the actual Omni Channel API Mock Implementation. Having the proxy deployed on the API gateway allows Anypoint Platform to manage, control access and monitor the usage of the API, which we will look at in the succeeding labs.

Step 1: Go to API Manager

1. Click the API Manager icon to start creating your API.
2. You can also click on the navigation button on the top left, then on API Manager.
3. select SANDBOX as the environment

Step 2: Configure an API Proxy

For this lab, we are going to configure the API as a proxy to Mythical Corporation's Omni Channel API Mock Implementation. The API is available as an HTTP Restful API accessible through the base URL http://workshop-omni-channel-mock-service-v40.cloudhub.io/api.

Let's take a look to the products resource of this API that returns product information when an HTTP GET is issued.

To see the all the resources of this API, you can view the API definition and Summary in Exchange
To create the proxy we are going to get the API Definition from Exchange.

1. Now let’s configure an API proxy gateway for this API. Back in the API Administration page, click on Manage API and select Manage API from Exchange.

2. Configure the API with the following information:
   a. API name: Omni Channel Experience API. Notice when you start to write the name, the field is autocompleted.
   b. API version: Select from the drop down list 1.0.0.
   c. Asset version: Select from the drop down list 1.0.0.
   d. Managing type: Select Endpoint with proxy.
   e. Implementation URI: http://workshop-omni-channel-mock-service-v40.cloudhub.io/api
   f. Proxy deployment target: CloudHub
   g. Path: /
3. Press **Advanced options** and configure:
   a. Check the box for managing an API in Mule4 and above
   b. Advanced options
      i. **API instance label**: `<username>-proxy-omni-channel`. This Parameter is going to be used to identify the API in the API Manager.
Since we are deploying to CloudHub, there is no need to specify the port. By default it’s port 80 or 443 depending if it’s http or https.

4. Press **Save**

After you press **Save** the deployment configuration section will appear to deploy the proxy.

5. Configure with the following information:
   a. **Runtime version**: Select **4.1.4**
   b. **Proxy application name**: `<username>-mythical-omni-channel-api-proxy`. This property will define the Proxy URL and the name of the Mule application in the **Runtime Manager**.

6. Press **Deploy Button**.

7. Click on **Click Here** to see the log data and monitor the progress.
8. A new browser tab will open that will show the CloudHub log for this application.

   **NOTE** If you are not directed to the Logs, you can get there by clicking on your application name, then click on Logs.

   You’ll see a blue circle next to the application name indicating the application is deploying. CloudHub is allocating a worker with 0.1 vCore to host your new proxy application. You will know it’s complete when a green icon appears. In the meantime, feel free to look around (view the Logs to see what’s going on).

   **NOTE** It can take anywhere from a few seconds to a few minutes to deploy the proxy. Good time to grab some coffee or explore the documentation.

   When you see **Your application is started** you can continue.

9. Go back to the previous browser tab where you were configuring the API. You should see the green Deploy Successful in the status bar.

10. Click Close.
11. Once it is deployed, at the top of the page you will see the status of the API. It should be green with a green ball next to it, as shown below. This indicates that your API was successfully deployed and is now being managed.

The proxy you just created is a Mule application/project. If you want to see what is going on under the covers just download the proxy and import it into Anypoint Studio. Unlike the blackbox proxies of the “pure-play” API Management tools like Mashery, Layer 7, apigee... Anypoint proxies are Mule applications which you can download and extend with all the capabilities of Mule that you will learn about through the remaining labs.

If needed, you could augment the generated flow with additional logic.

While you deployed this proxy to an API Gateway running on CloudHub, you could also deploy this proxy application to an on-premise API Gateway.

Step 3: Test the API Proxy

Your proxy API is now accessible through CloudHub.
You can get the exact URL of your hosted proxy under the API Status:

1. Right-click on the URL to copy it.
2. Open POSTMAN.
3. Paste what you copied into the field that says Enter request URL but do NOT hit enter yet.
4. Add “/products/search” to retrieve the product information using your proxy.

You should have a URL that looks like: http://{your_domain_name}/products/search

5. Go to the Headers tab
6. Add the following values:

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Accept</td>
<td>application/json</td>
</tr>
</tbody>
</table>

7. Now press Send and you should see the following:
NOTE You can also test the API by using any other REST Client. It’s important to set the headers after you do the request.

Summary

In this lab, we completed the following steps:

- **Overview**
- **Step 1: Go to API Manager**
- **Step 2: Configure an API Proxy**
- **Step 3: Test the API Proxy**

We saw how easy it is to proxy existing APIs and deploy it on a CloudHub-based API Gateway. This provides a **low friction** approach to manage existing APIs across your environment. You are able to leverage a **hybrid** API Gateway service offering which can be on-premise or on the cloud. In this lab, we saw how we can deploy our API Gateway to CloudHub which can significantly **speed up your deployment** without having to manage and maintain infrastructure.

For further reading on deploying an API gateway and proxying your API, please refer to the following documentation:
Congratulations! You have completed Lab 1.

Please proceed to Lab 2

Take me to the TOP