Lab 5

Proxy Legacy Services with an HTTP Proxy (Optional)

Overview

API management is essential to an API-Led architecture as it provides a governance framework to your APIs in all three layers. You saw in Lab 1 how MuleSoft can proxy your published APIs in Exchange on an API Gateway, and in Lab 4 how you can directly govern a MuleSoft implemented API.

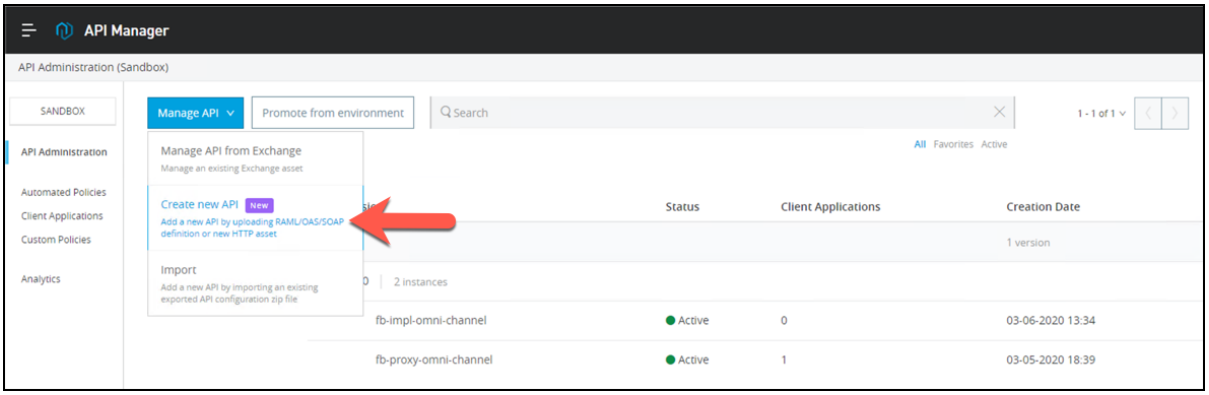
What if you have an existing/legacy service? MuleSoft can govern these as well. We govern these existing services through an HTTP proxy. Note that it is possible to govern REST-JSON services and SOAP Web Services this way!

The API we will use to demonstrate this is an already deployed mock implementation.

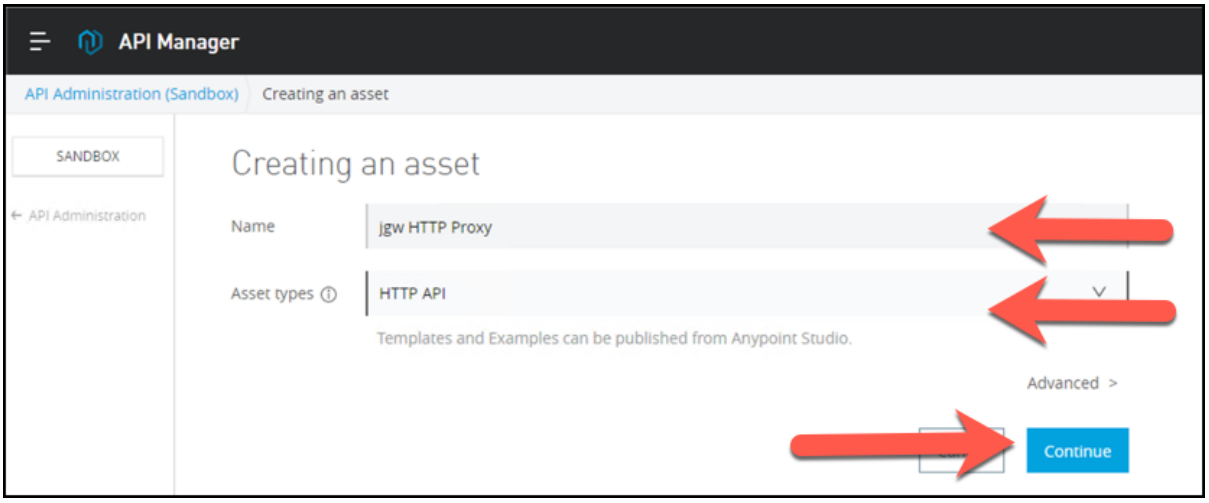
Clients will access the API through the API Gateway which will then forward the requests to the actual Omni Channel API Mock Implementation. Having the proxy deployed on the API gateway allows Anypoint Platform to manage, control access and monitor the usage of your legacy services, which we demonstrate in labs 2 and 3.

Step 1: Create an API-HTTP Proxy in Exchange

1. Navigate to **API Manager**
2. Select **Manage API**



3. Set the **Name** to **<your name> HTTP Proxy** and the **Asset Type** to **API-HTTP** and press **Publish**:



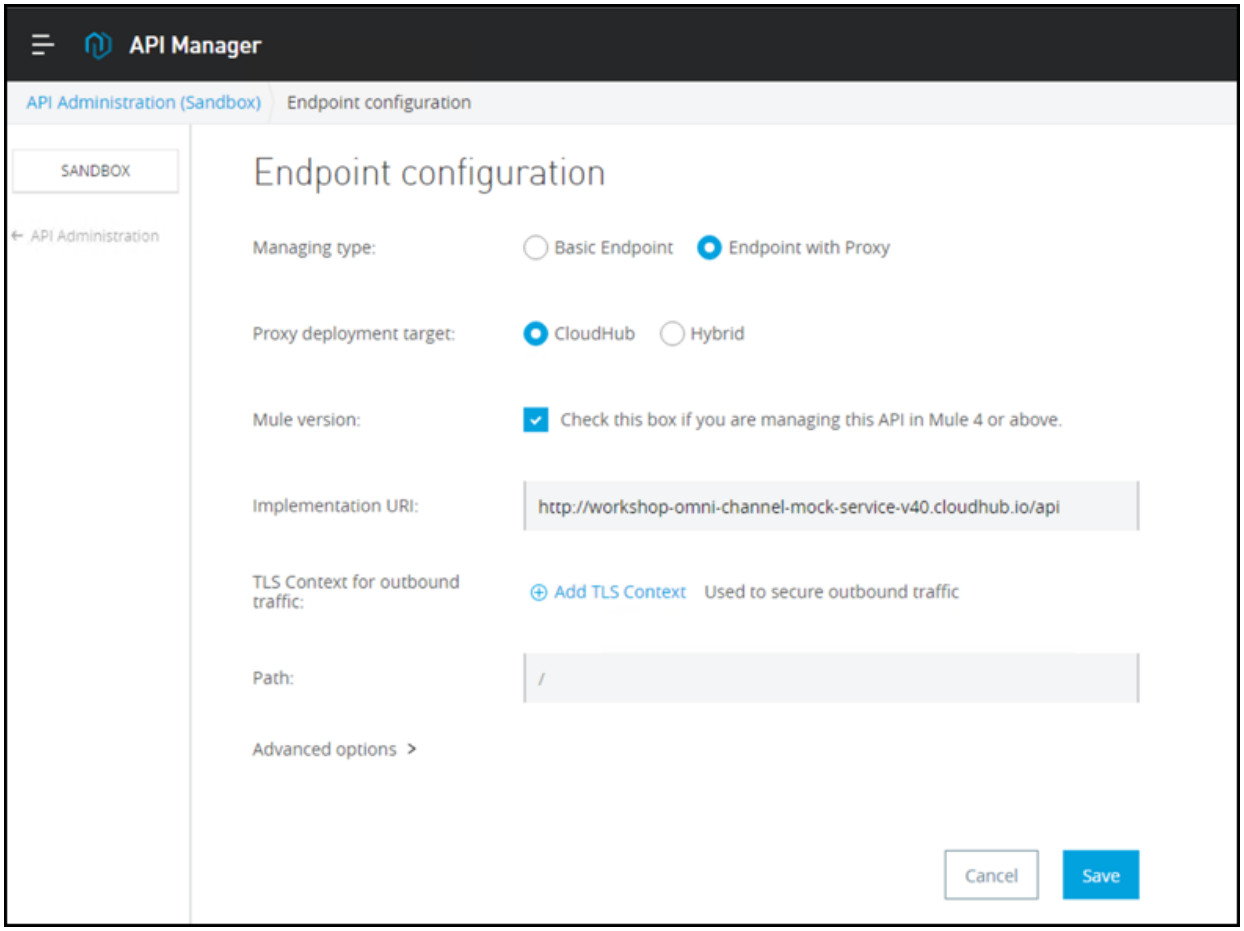
Step 2: Configure the API-HTTP Proxy in API Manager

For this lab, we are going to configure the API-HTTP 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 the products resource of this API that returns product information when an HTTP GET is issued.

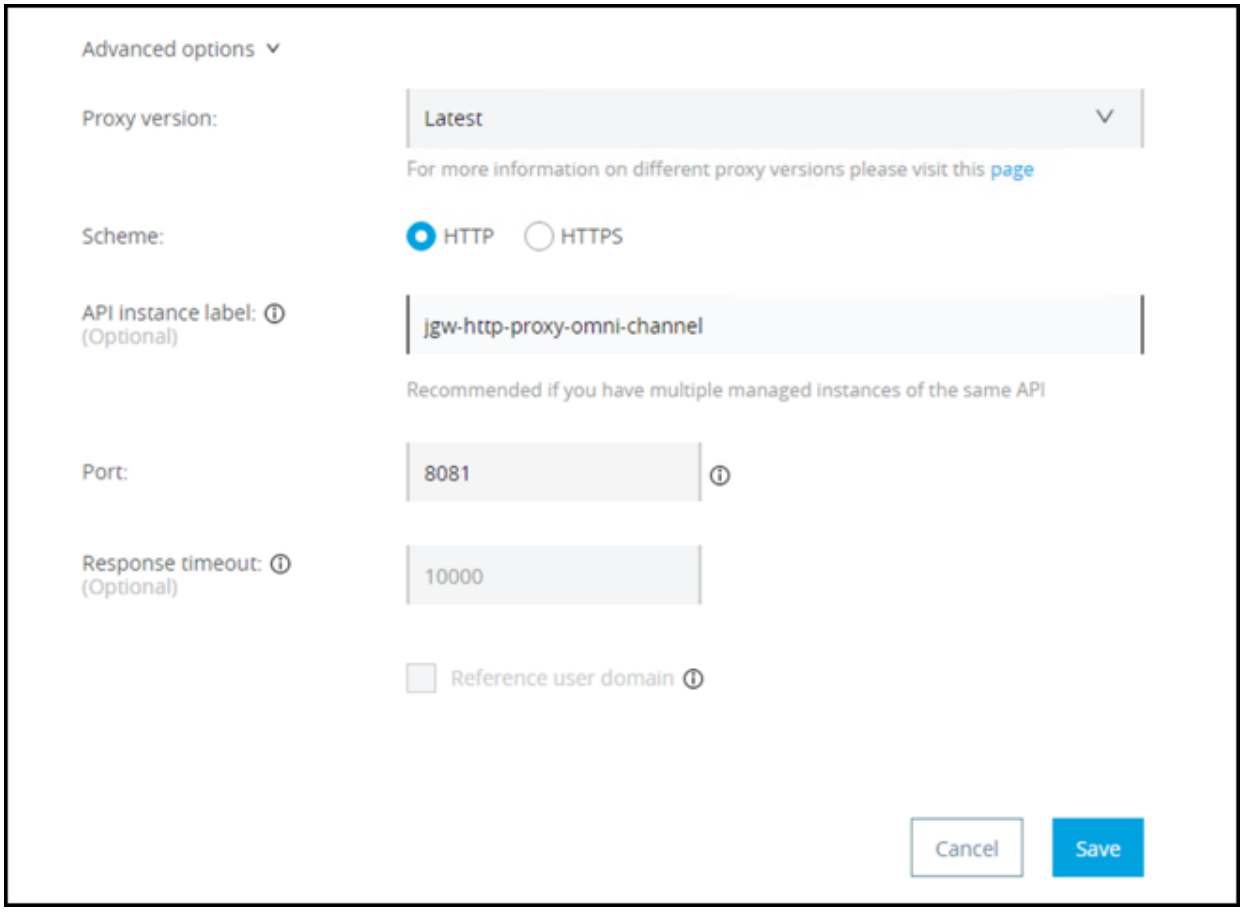
To see the all the resources of this API, you can view an API Console using the URL http://workshop-omni-channel-mock-service-v40.cloudhub.io/console/

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. **Managing type**: Select `Endpoint with proxy`.
    b. **Implementation URI**: `http://workshop-omni-channel-mock-service-v40.cloudhub.io/api`
    c. **Proxy deployment target**: `CloudHub`
    d. **Mule Version**: `Check the box`
    e. **Path**: `/`



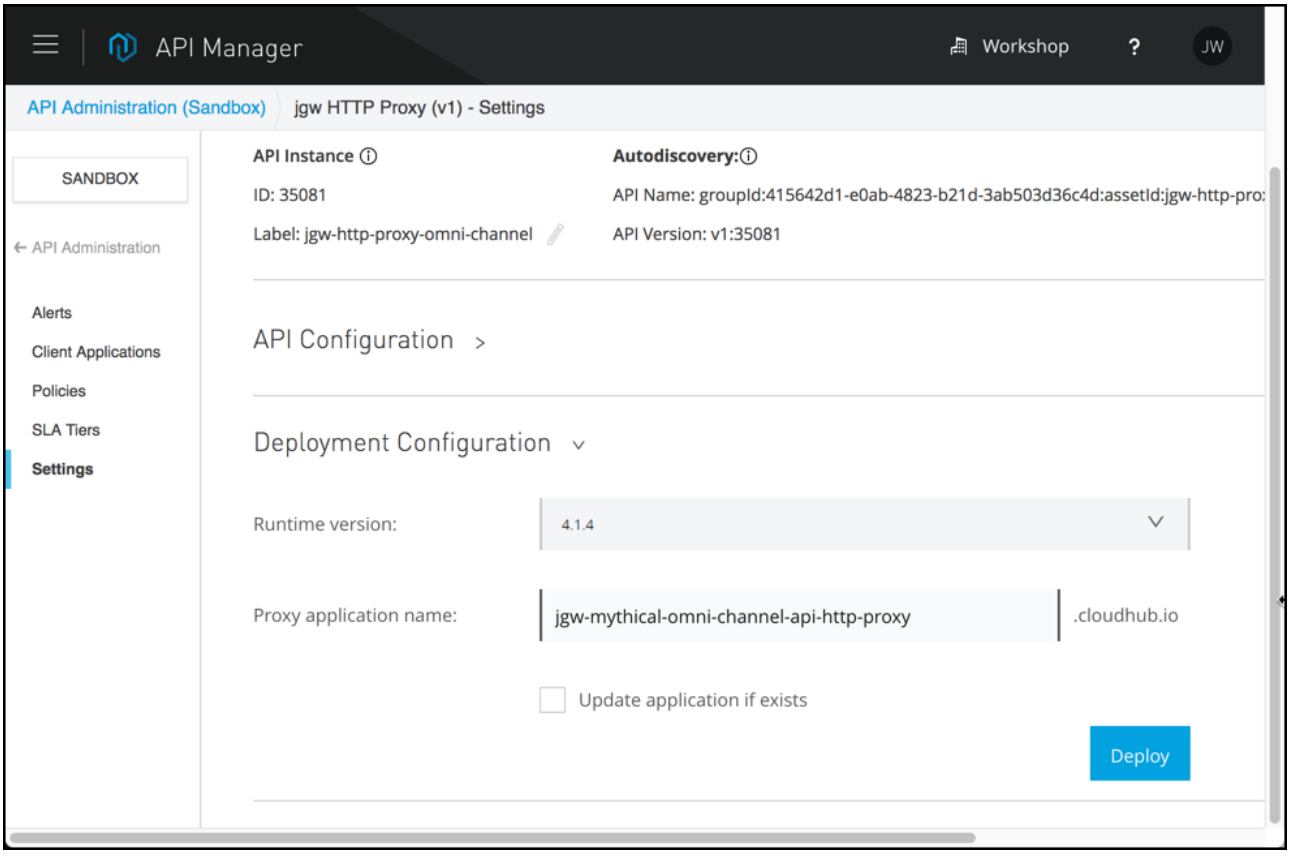
3. Click **Advanced options**
    a. **API instance label**: `<username>-http-proxy-omni-channel`. This Parameter is going to be use to identify the API in the API Manager.

NOTE	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.



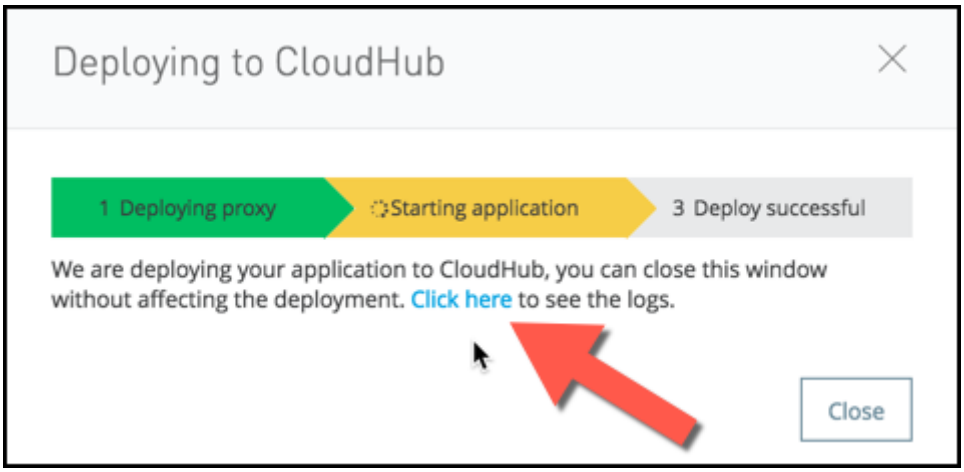
5. Configure with the following information:

a. **Runtime version**: Select `4.1.4`

b. **Proxy application name**: `<username>-mythical-omni-channel-api-http-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.



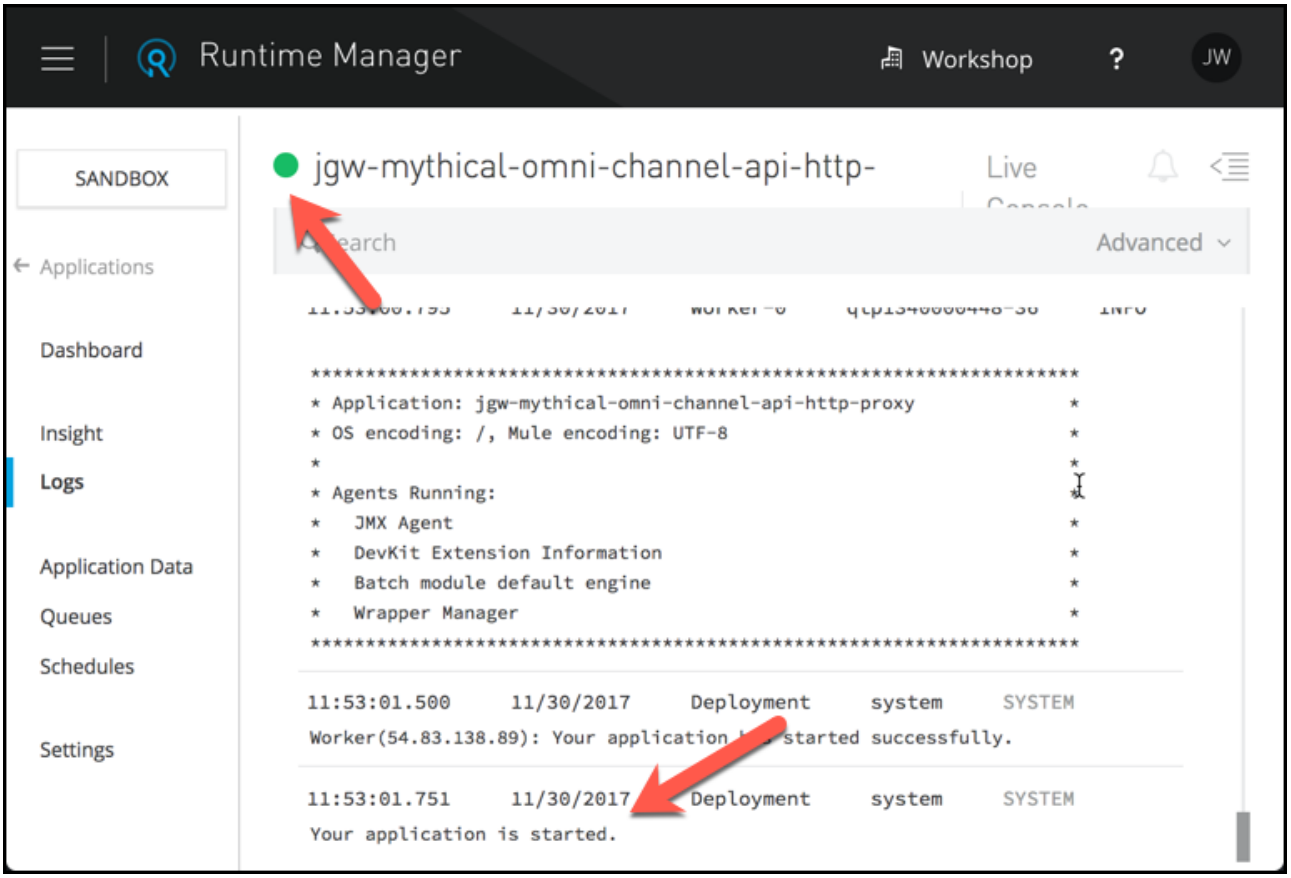
8. A new browser tab will open that will show the CloudHub log for this application.

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

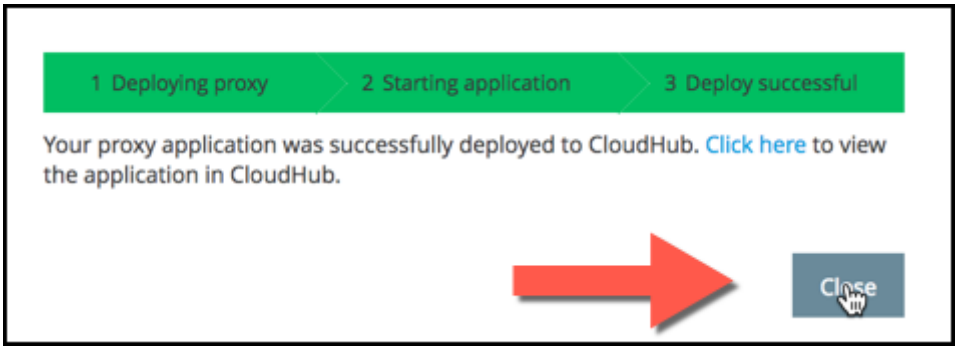
You'll see a blue circle next to the application name indicating the application is deploying. CloudHub is allocating a worker with 0.1 vCore to host your new proxy application. You will know it's complete when a green icon appears. In the meantime, feel free to look around (view the Logs to see what's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

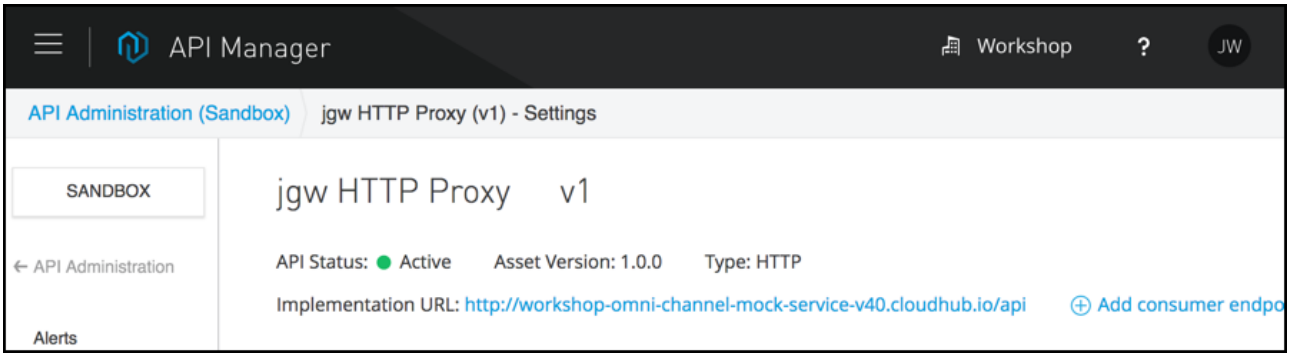
When you see **Your application is started** you can continue.



9. Go back to the previous browser tab where you were configuring the API. If the "Deploying" pop-up is still open you should see the green Deploy Successful in the status bar.
10. Click **Close**.



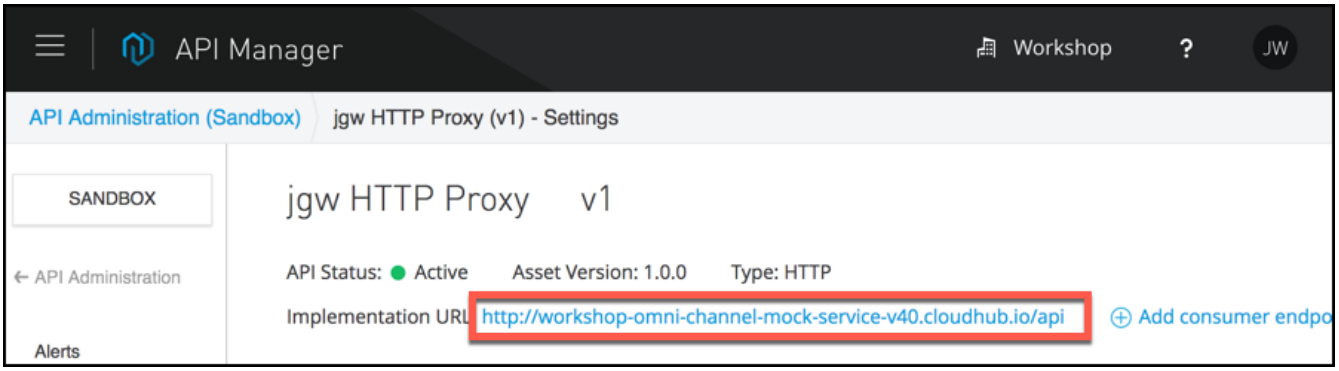
11. Once it is deployed, at the top of the page you will see the status of the API. It should be green with a green ball next to it, as shown below. This indicates that your API was successfully deployed and is now being managed (**you may need to refresh your browser**).



NOTE	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.
NOTE	If needed, you could augment the generated flow with additional logic.
NOTE	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-HTTP Proxy

Your proxy API is now accessible through CloudHub.

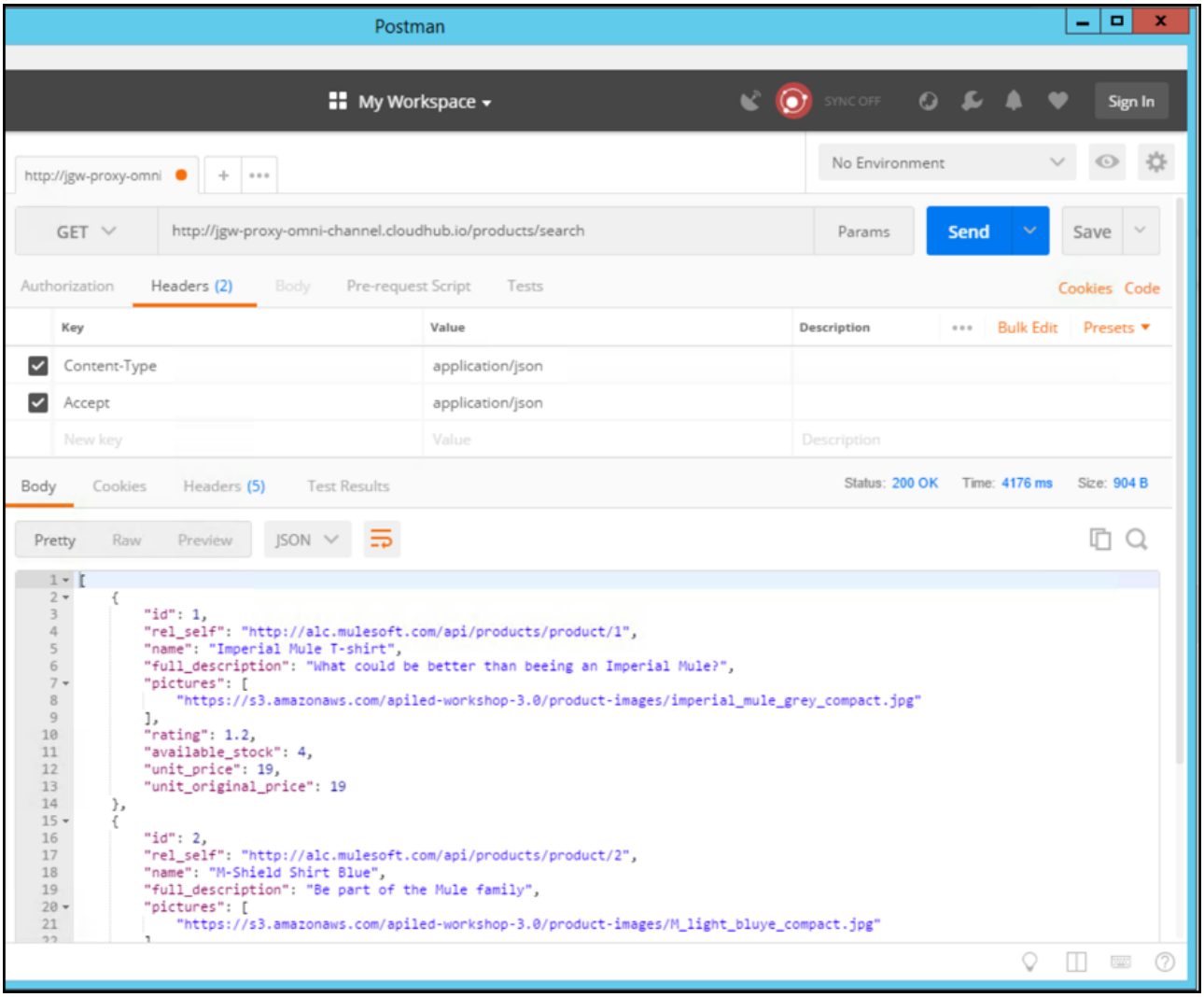


You can get the exact URL of your hosted proxy under the API Status:

1. Right-click on the URL to copy it.
2. Open another tab in your browser.
3. Paste what you copied into the new tab 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. Now press Enter and you should see a response similar to the following:



6. You can also test the API by using a REST Client like PostMan or ARC (Advanced Rest Client).

Summary

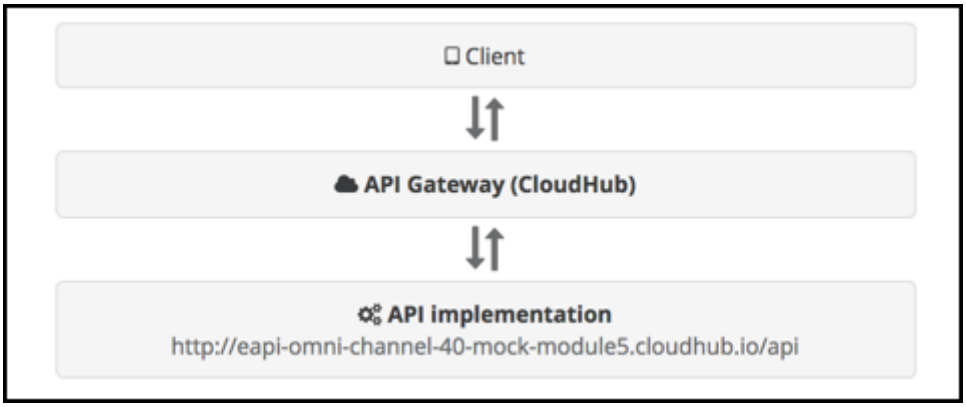
In this lab, we completed the following steps:

- Overview
- Step 1: Create an API-HTTP Proxy in Exchange
- Step 2: Configure the API-HTTP Proxy in API Manager
- Step 3: Test the API-HTTP 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:

- Proxying Your API



Congratulations! You have completed Lab 5.

Please proceed to Module 6

Take me to the TOP