Module 2: The Modern API - Implementation

Overview

You have completed the design of the Omni Channel API and now you are ready to implement it. In this module, we will create the implementation of our API that will process the REST requests.

The Module is separated into four different labs.

- Lab 1: Create your API project in Studio
- Lab 2: Implement your API
- Lab 3: Deploy your API

Please proceed to Lab 1

Lab 1: Create your API project in Studio

Overview

In this first lab, you will use Anypoint Studio to create a Mule application where there will be one flow for each method of each resource (i.e. GET). Additionally you will use APIKit (as part of your Mule application) to process each REST request, transform them to messages to be handled and processed by each flow.

The implementation will consist of a few steps

1. Create a new Mule Project in Anypoint Studio from the Omni Channel API RAML
2. Run the Mule app from Anypoint Studio

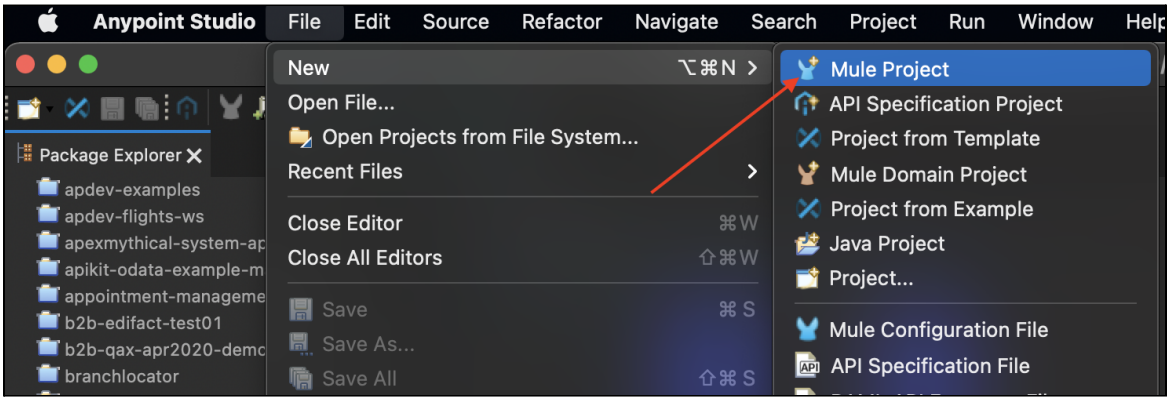
Then we will test this new API using the API Console.

Step 1: Create a new Mule Project from the RAML

In this step we will create a new Mule application in Anypoint Studio from the Omni Channel API RAML Definition. This will be the implementation of our REST API.

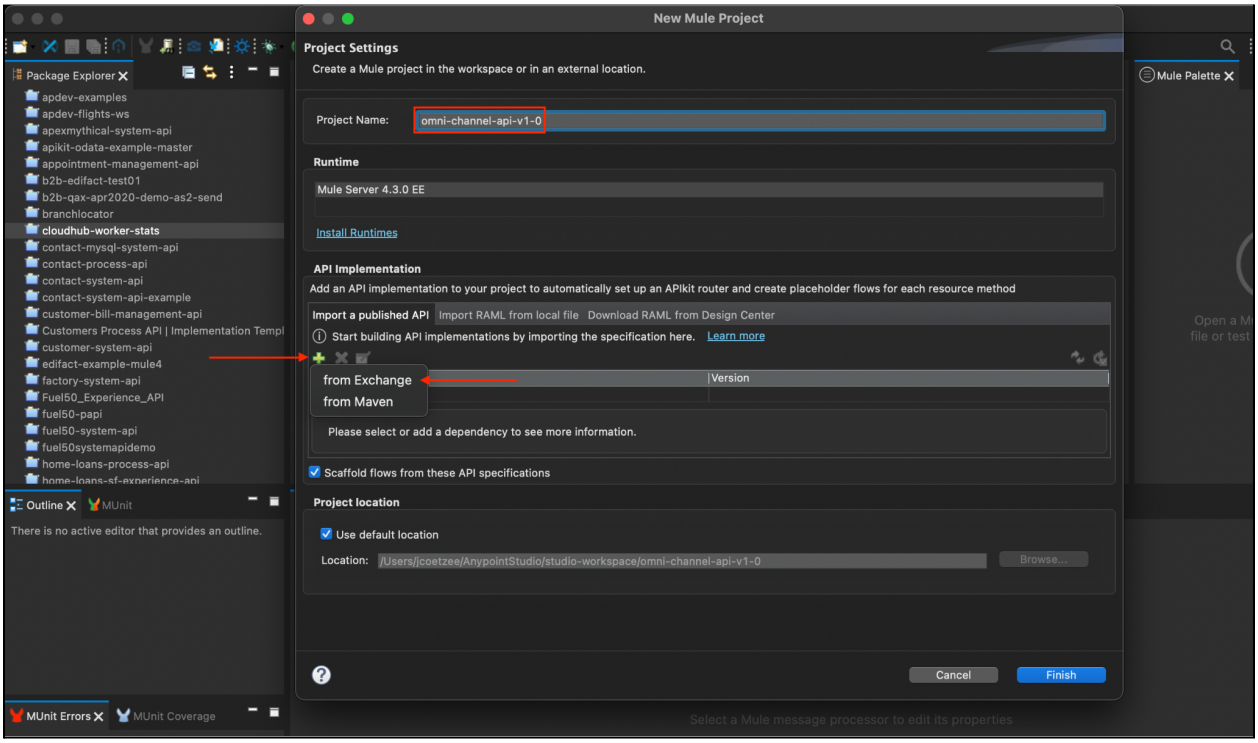
1. Start Anypoint Studio from the desktop icon.

2. You are presented with the Anypoint Studio 7 Welcome Screen.
3. Click through the "What's new" items and click **Go to Workspace**.
4. In the Workspace Launcher, use the default workspace directory location.
5. From Anypoint Studio's menu, select File > New > Mule Project to create a New Mule project. A Window will pop-up to define the details of this new application.

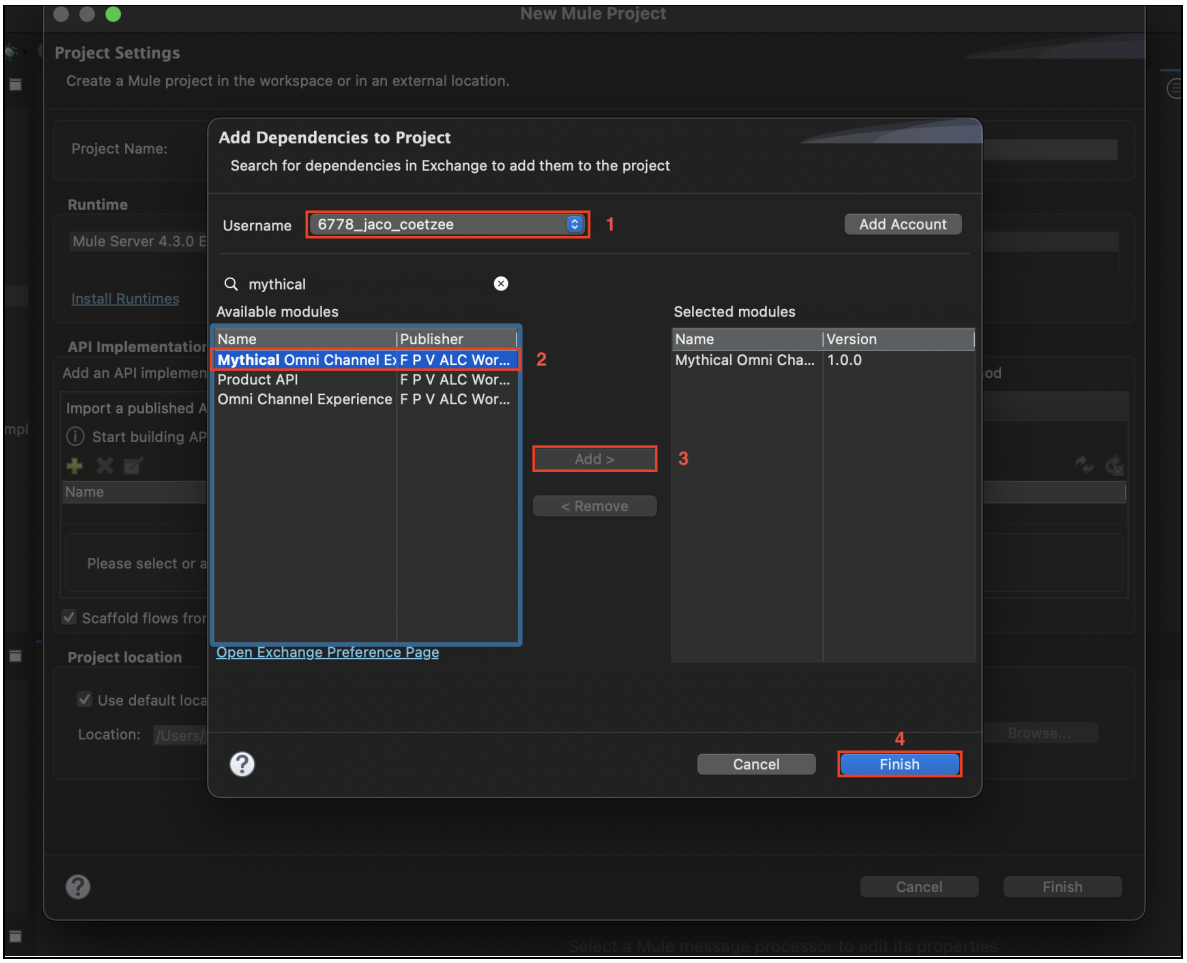


6. Give the project the name **omni-channel-api-v1-0**
7. Make sure the runtime is **Mule Server 4.3.0 EE**.

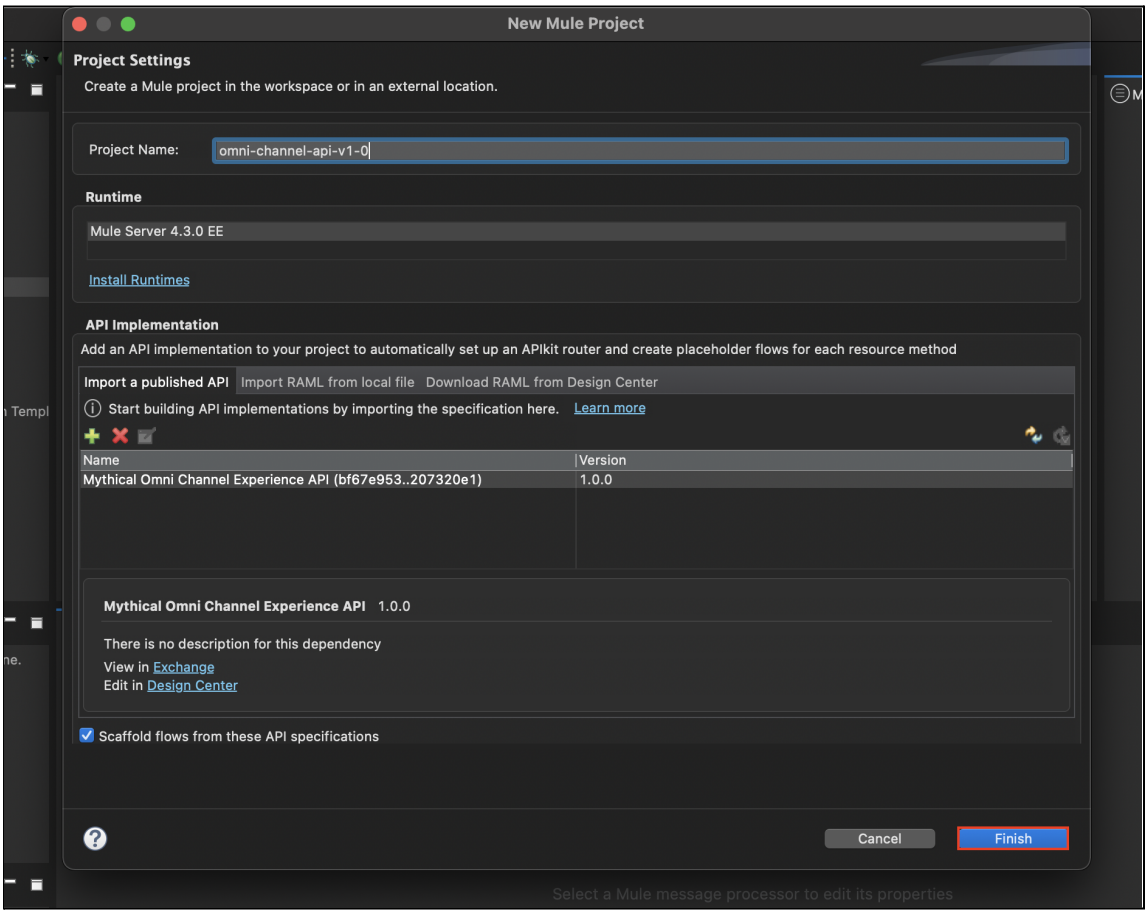
8. Under **API Implementation** settings, select the green + and choose **from Exchange**.



9. Once you click there, it should take you to the **Browse Exchange for APIs** window, if it asks you to login, use your Anypoint Platform account.
10. Select the workshop business group if not already selected.
11. **Search** for and select the **<username> - Omni Channel Experience API** that you created in Module 1 from the available modules and click **Add** to add it to the project then click **Finish**. If you can't find your API, click through the different Business Groups. You can also search for your API using the **type filter text** field. *(you may have more API Definitions showing in this window)*

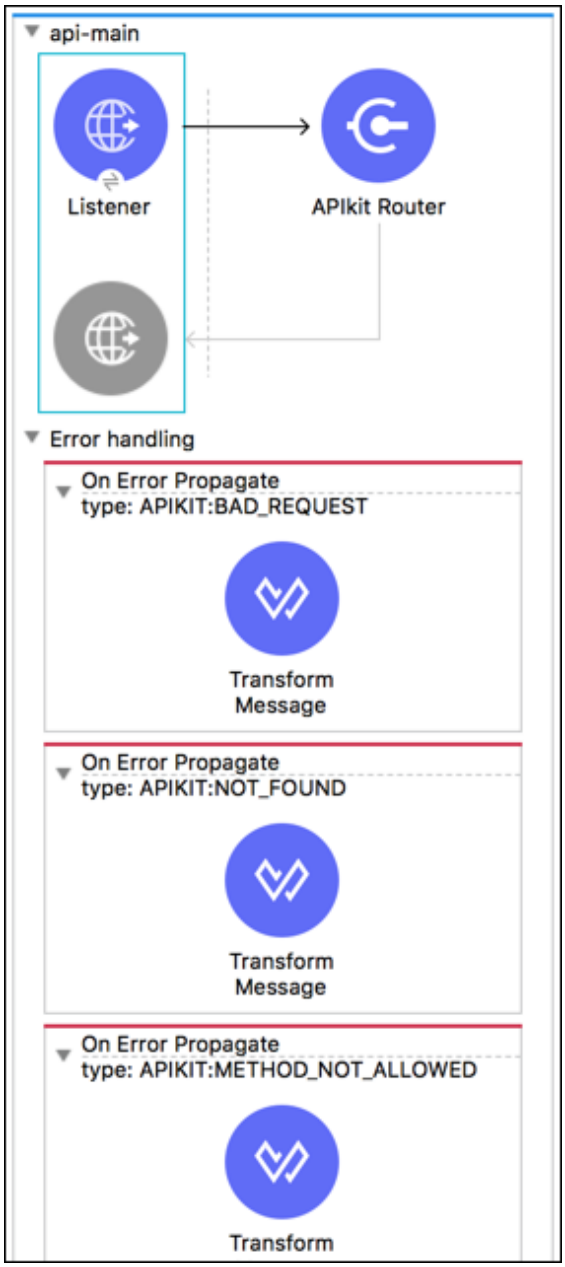

New Mule Project
Project Settings
Create a Mule project in the workspace or in an external location.
Project Name:
Runtime
Mule Server 4.3.0 E
Install Runtimes
API Implementation
Add an API implemen
Import a published A
Start building AP
Name
Please select or a
Scaffold flows fro
Project location
Use default loca
Location:
Cancel
Finish
Add Dependencies to Project
Search for dependencies in Exchange to add them to the project
Username
6778_jaco_coetzee
1
Add Account
mythical
Available modules
Name
Publisher
Mythical Omni Channel E
F P V ALC Wor...
2
Product API
F P V ALC Wor...
Omni Channel Experience
F P V ALC Wor...
Add >
3
< Remove
Selected modules
Name
Version
Mythical Omni Cha...
1.0.0
Open Exchange Preference Page
4
Cancel
Finish

12. Now click **Finish**. This will create a skeleton project that implements your API.



Let's explore what was automatically generated.

13. At the top of the diagram you will see a flow called **api-main** and **api-console** followed by flows, one for each method defined in the RAML.
    The flows below are defined by your API Design in the RAML file. Typically, there will be flows that look like this **get:\resource**, **post:\resource**, **put:\resource**, etc. Note that the name of the flow is very important for the APIkit router to be able to route the request to the appropriate flow - you don't want to change these.
    When APIkit detects example data in the response of a method in the RAML it inserts a **Dataweave Transform Message** into the flow which returns the static response specified by the example data reference.
    The static response returned by the auto-generated **Dataweave Transform Message** allows you to test the API as a stub immediately after generation. Obviously these flows can be enhanced to provide more advanced mock services as well as evolve them into full API implementation as we will see in the next lab.



api-main

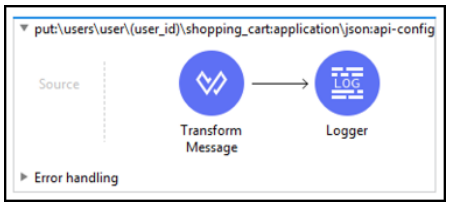
This is the main flow. It exposes an HTTP service and processes the requests using the APIKit Router.

The HTTP request will be converted to a mule message, and redirected to the requested flow by the APIKit router.

By selecting the http listener one can take a look at the HTTP configuration, you will see that its listening for requests on http://localhost:8081/api.

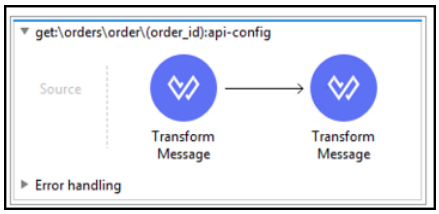
Inside the api-main flow also an Error-Handling block is included. In this block multiple **On Error Propagate** definitions are autogenerated for common API error responses such as

METHOD_NOT_ALLOWED, **BAD_REQUEST** etc.



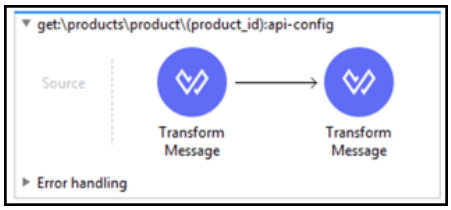
put:\users\user{user_id}\shopping_cart

This flow puts an item into a users shopping cart based on the user id. This **put** operation returns no response and therefore the **Dataweave Transform Message** processor is replaced automatically by a **Logger** message processor.



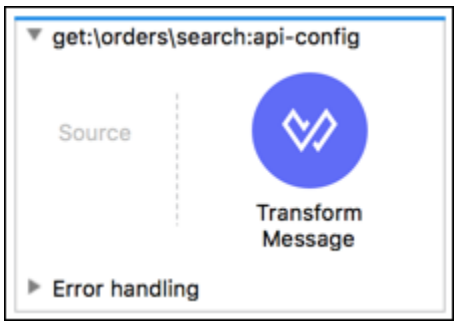
get:\orders\order\{order_id}

This flow gets an order based on an order id



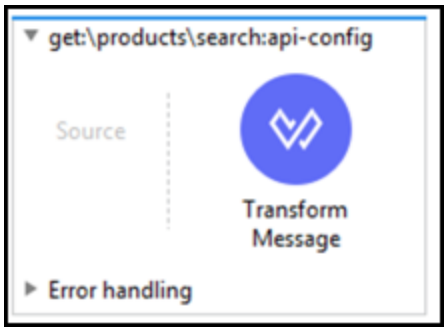
get:\products\product\{product_id}

This flow gets a product based on a product id



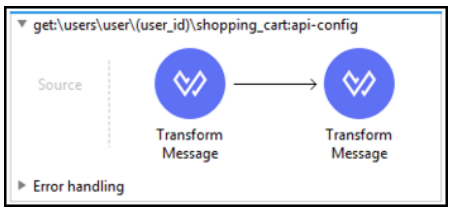
get:\orders\search

This flow returns orders based on search criteria



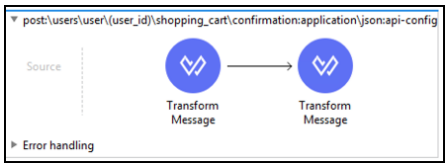
get:\products\search

This flow returns products based on a search criteria



get:\users\user\{user_id}\shopping_cart

This flow returns all the items in a users shopping cart based on a user id

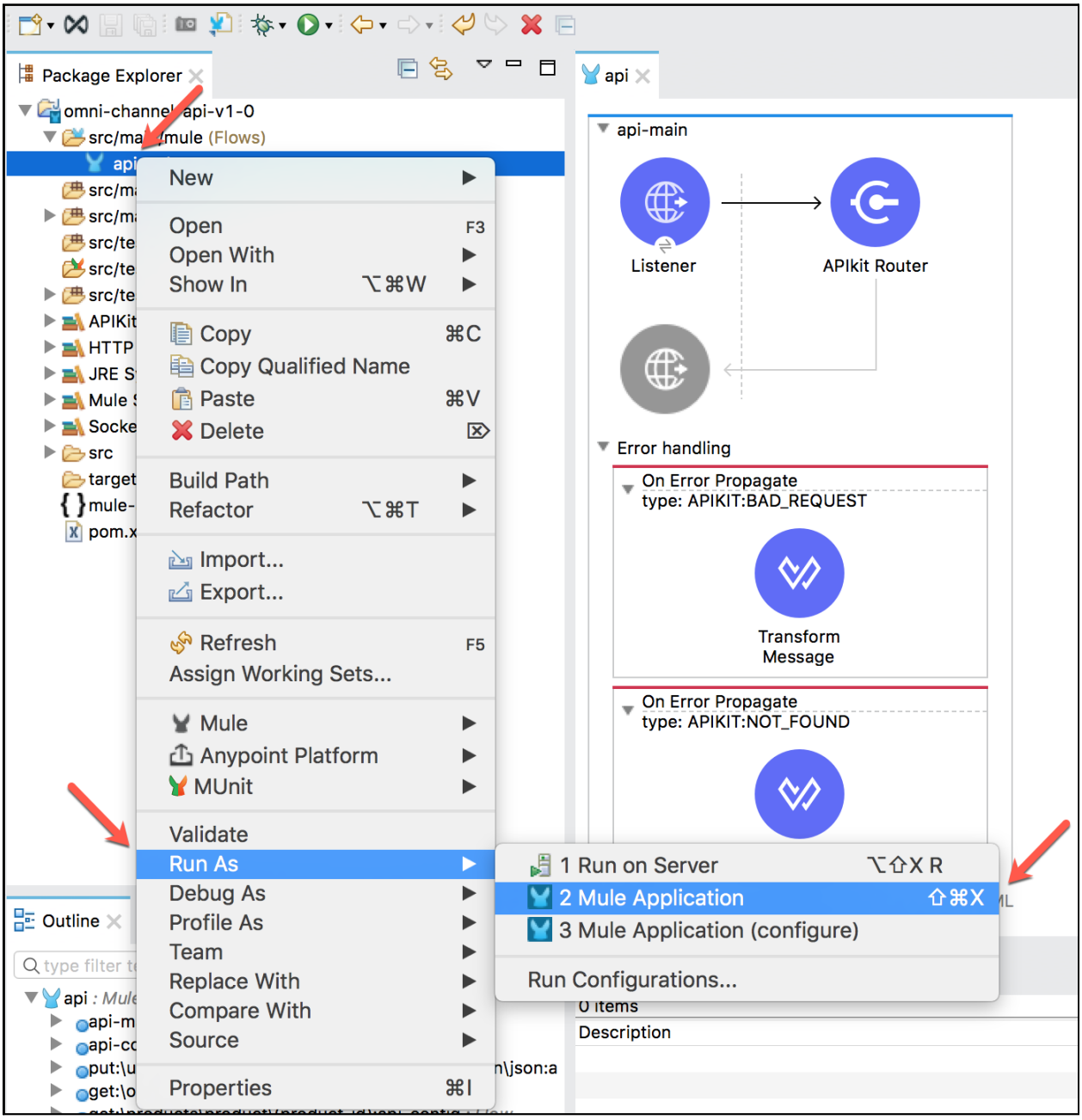


post:\users\user\{user_id}\shopping_cart\confirmation

This flow confirms the items to purchase in a users shopping cart.

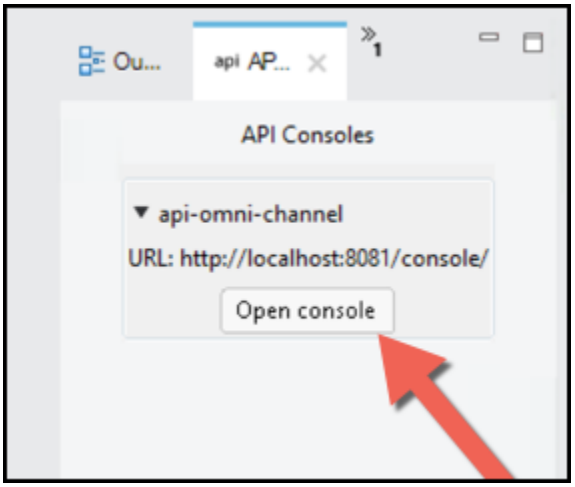
Step 2: Test the Generated API Project

1. To test the API, let's run it within Studio first. Right click the application.
2. Select **Run As > Mule Application**.



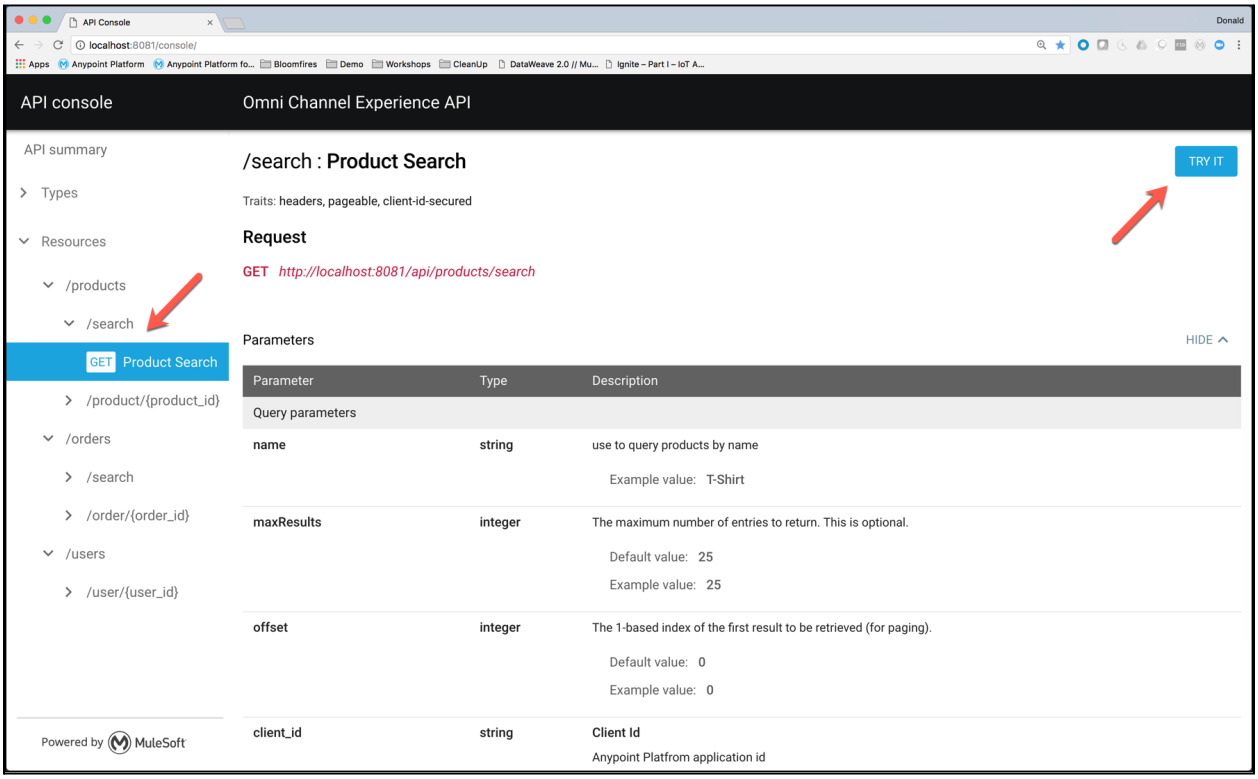
3. The application will start running, and the console will show the Mule Runtime logs

4. Test the application using the console. Click the button **Open Console** on the APIkit Console tab.

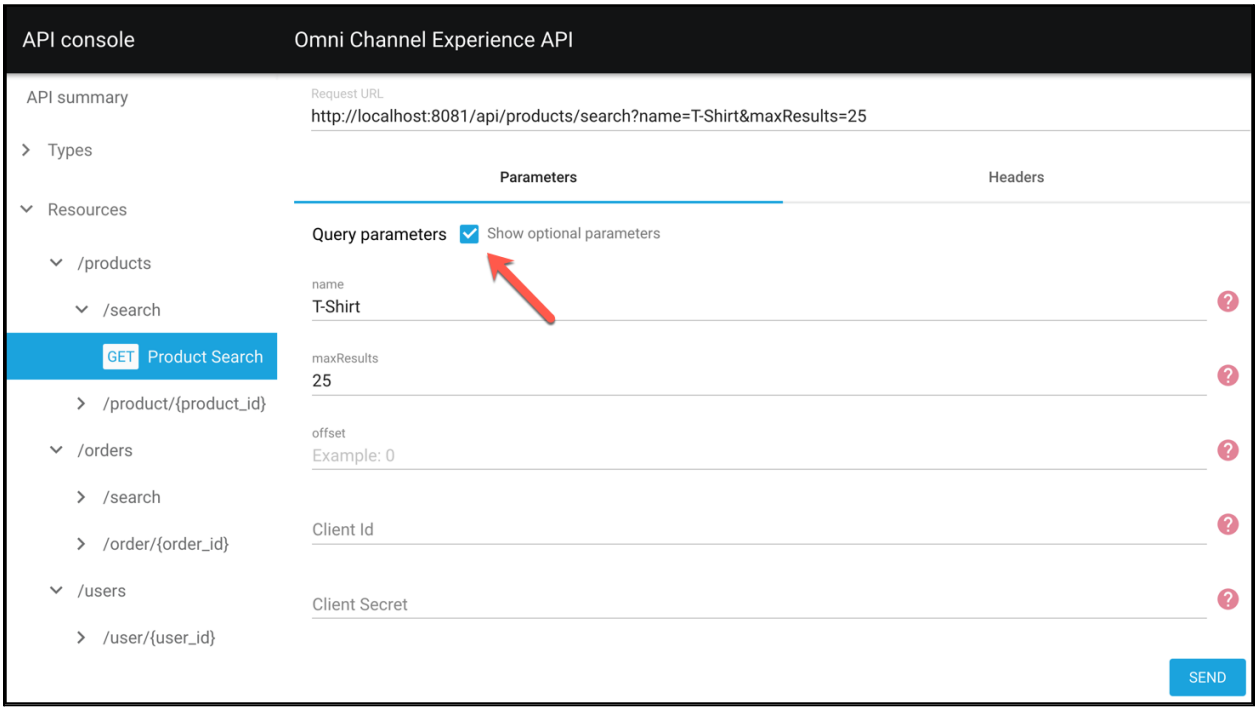


5. A browser window opens at: **http://localhost:8081/console/**.
6. Click open the **/products/search** resource on the left and click **GET**.

7. Click the **TRY IT** button on the top right to submit the request.



8. Then tick the **Show optional parameters** and fill them in as below and click **SEND**.



The response will show mocked data that is defined in the imported RAML.

200 OK 16.96 ms DETAILS

```
[Array[2]
  -0:  {
      "id": 1,
      "rel_self": "http://alc.mulesoft.com/api/products/product/1",
      "name": "Imperial Mule T-shirt",
      "full_description": "What could be better than beeing an Imperial Mule?",
     -"pictures": [Array[1]
        0:  "https://s3.amazonaws.com/apiled-workshop-3.0/product-images/imperial_mule_grey_compact.jpg"
      ],
      "rating": 1.2,
      "available_stock": 4,
      "unit_price": 19,
      "unit_original_price": 19
   },
  -1:  {
      "id": 2,
      "rel_self": "http://alc.mulesoft.com/api/products/product/2",
      "name": "M-Shield Shirt Blue",
      "full_description": "Be part of the Mule family",
     -"pictures": [Array[1]
        0:  "https://s3.amazonaws.com/apiled-workshop-3.0/product-images/M_light_bluye_compact.jpg"
      ],
      "rating": 1.2,
      "available_stock": 4,
      "unit_price": 19,
      "unit_original_price": 19
   }
],
```


It is a live application deployed to the Mule runtime server but it is leveraging the RAML information to provide a “mocking” experience.

Let’s go ahead and stop the application to get ready for the next lab.

9. Go to the console tab and press the red squared button to stop the Mule runtime server.

Mule Properties | Problems | Console | APIkit Consoles (omni-channel-api-v1-0)

```
omni-channel-api-v1-0 [Mule Applications] Mule Server 4.0.0 EE
* Application libraries:                                          *
*  - omni-channel-api-v1-0                                        *
*******************************************************************
INFO  2017-12-22 12:21:59,413 [WrapperListener_start_runner] org.mule.runtime.module.deployment.internal.DeploymentDirectoryWatcher:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Mule is up and kicking (every 5000ms)                           +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
INFO  2017-12-22 12:21:59,416 [WrapperListener_start_runner] org.eclipse.jetty.server.ServerConnector: Started ServerConnector@3197063a{HTTP/1.1}{0.0.0.0:50510}
INFO  2017-12-22 12:21:59,418 [WrapperListener_start_runner] org.mule.runtime.module.deployment.internal.StartupSummaryDeploymentListener:
*******************************************************************
*              - - + DOMAIN + - -              * - - + STATUS + - - *
*******************************************************************
* default                                      * DEPLOYED          *
*******************************************************************

*****************************************************************************************************
*            - - + APPLICATION + - -           *       - - + DOMAIN + - -       * - - + STATUS + - - *
*****************************************************************************************************
* omni-channel-api-v1-0                        * default                        * DEPLOYED           *
*****************************************************************************************************

INFO  2017-12-22 12:27:01,658 [http.listener.02: http.listener @5298396e SelectorRunner] org.mule.service.http.impl.service.server.HttpListenerRegistry: No listener found fo
INFO  2017-12-22 12:27:01,663 [http.listener.02: http.listener @5298396e SelectorRunner] org.mule.service.http.impl.service.server.HttpListenerRegistry: Available listeners
```

Summary

In this lab, we

- Downloaded the API definition directly into Anypoint Studio.
- Used the RAML specification to generate the initial implementation skeleton for the API.
- Auto-generated a skeleton project to implement the API.

This lab shows how APIkit quickly allows developers to import the designed REST API (RAML and supporting artifacts) to enable rapid API logic implementation in the generated MuleSoft flows. The APIkit also deploys the interactive console with the MuleSoft flows. The lab shows the extent to which a developer can deploy an on-premise and interactive API from the API specification in RAML.

It is important to note that the implementation of an API is a real MuleSoft application and allowed us to test the end-to-end API. Building integration applications and exposing APIs do not require additional knowledge making it easy for MuleSoft developers to work on both integrations and APIs.

- See the APIkit documentation for more information.
- See the Anypoint Exchange documentation for more information.
- See the Error-Handling documentation for more information.

Congratulations! You have completed Lab 1.

Please proceed to Lab 2

Take me to the TOP

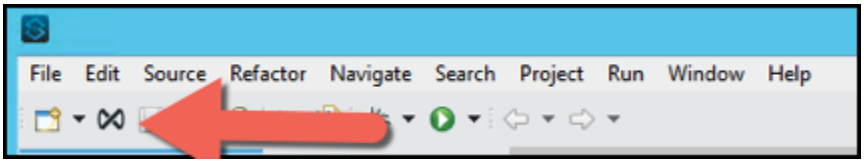
Lab 2: Implement your API

Overview

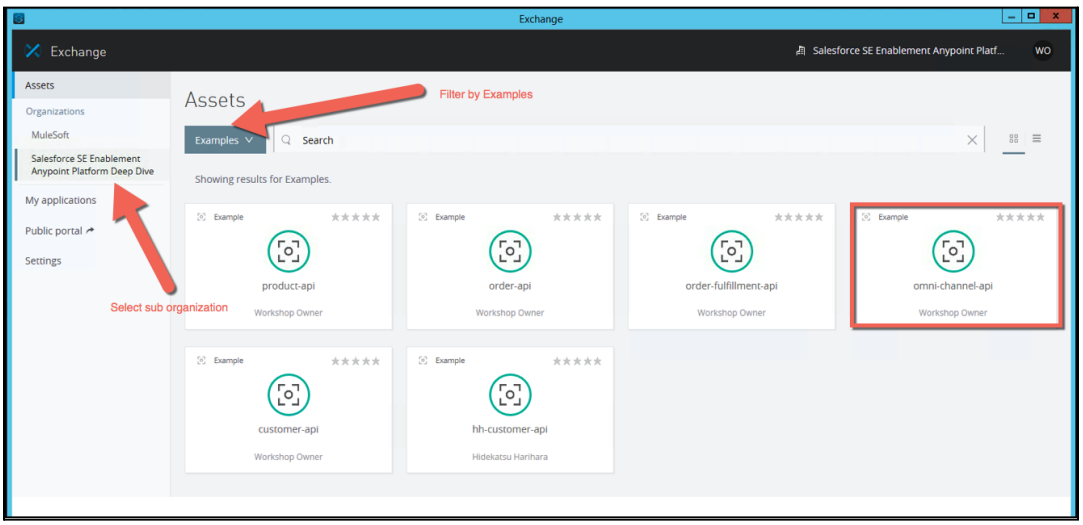
Now that we've seen how APIKit helps developers import the designed REST API to enable MuleSoft flow development, we'll download a fully implemented Omni Channel API project.

Step 1: Import the project from Anypoint Exchange

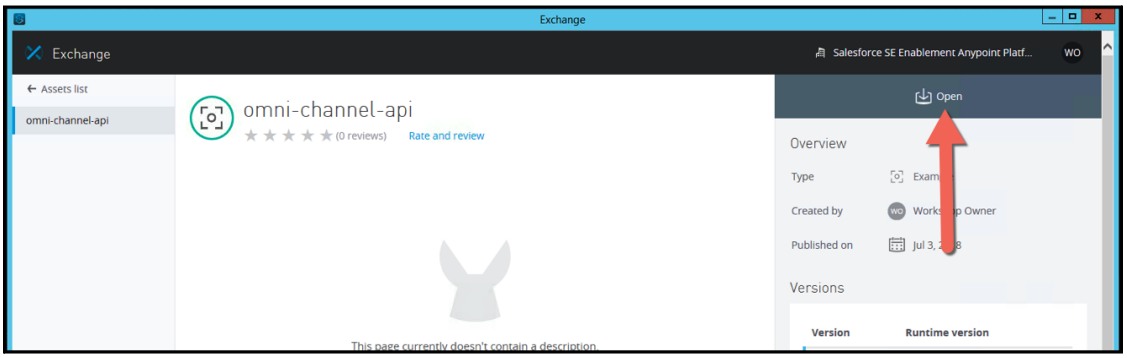
1. From Studio open exchange



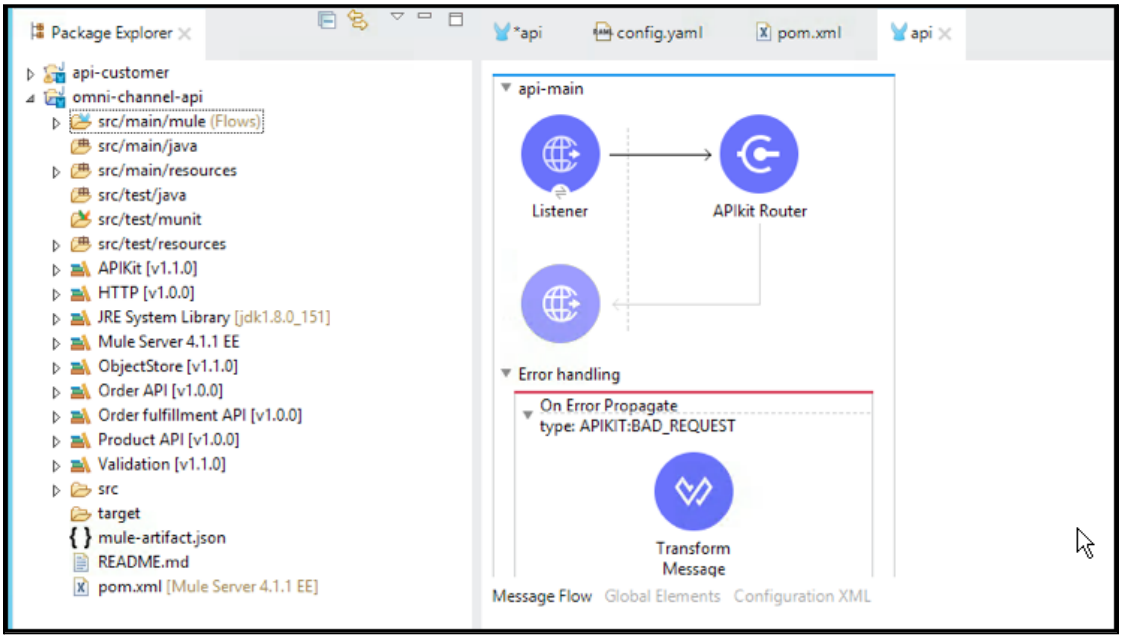
2. Filter by Examples
3. Select the sub organization
4. Click on the omni-channel-api



5. In the portal press **Open**

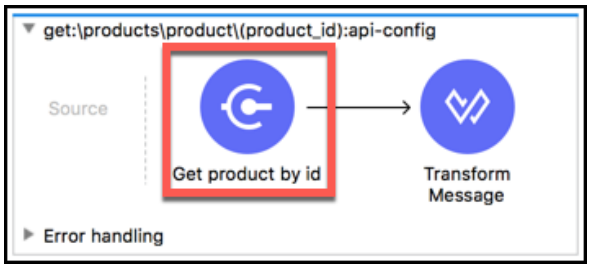


Once you open it, the project is going to be imported into Studio.



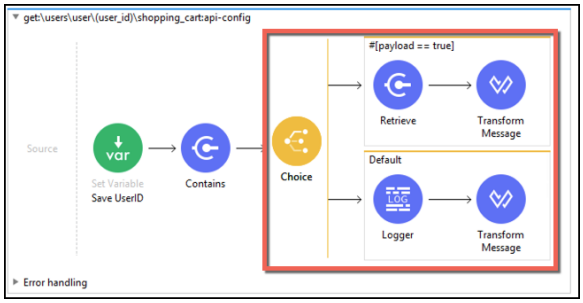
Step 2: Examine the different flows within the project

The previous lab already introduced the **APIKit**, **Error handling** and **Dataweave Transform Message** processors. In below flows we introduce **REST Connectors**, **Flow Control** components and **Scopes**. So let's first look at these in more detail.



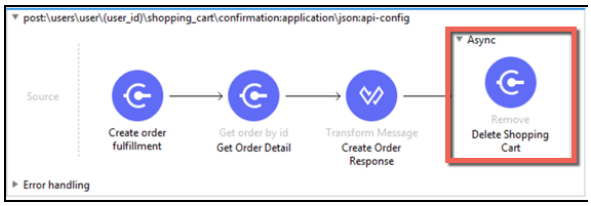
REST Connector

A REST Connector is a component that can be used in Mule applications that's been custom created via a specific framework. This framework allows you to create components entirely by using other existing Mule components. When an API Specification is published to exchange a REST Connector is automatically generated for it in Exchange by REST Connect Tool.



Flow Control

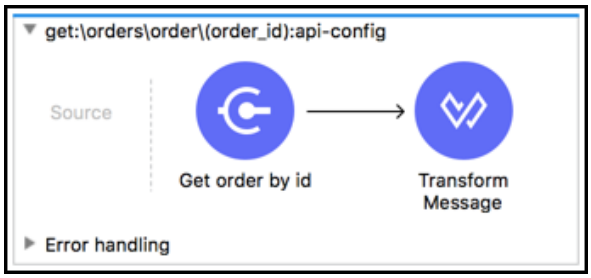
Flow Control Routers such as the **Choice Router** allow for controlling the message flow within a Mule flow.



Scopes

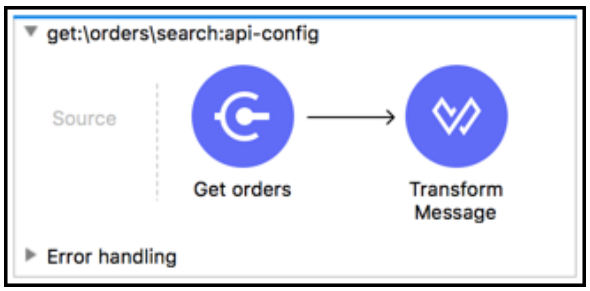
Sometimes referred to as "wrappers", the message processors known as **Scopes** appear as processing blocks when you first place them on the Message Flow canvas.

Now, let's look at all the flows within the project.



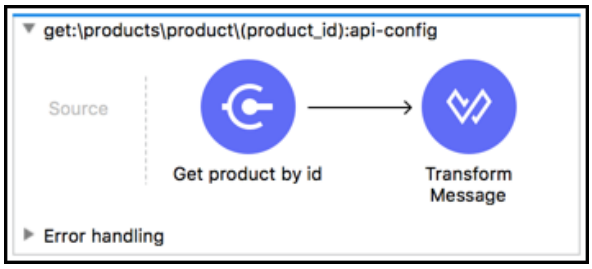
get:\orders\order\{order_id}

This flow calls the Order API to get an order from the database based on the order id



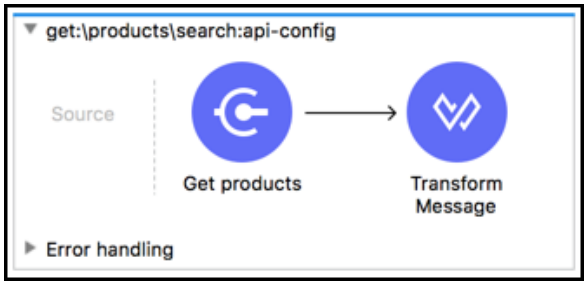
get:\orders\search

This flow calls the Order API to get orders from the database based on a search



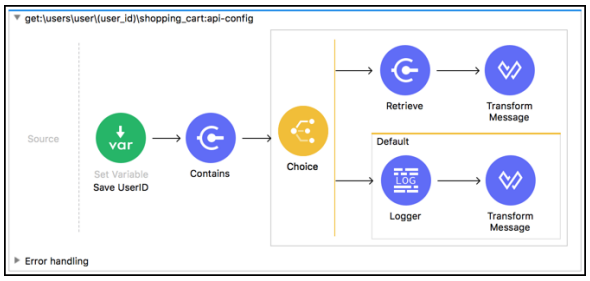
get:\products\product\{product_id}

This flow calls the Product API to get a product based on the product id



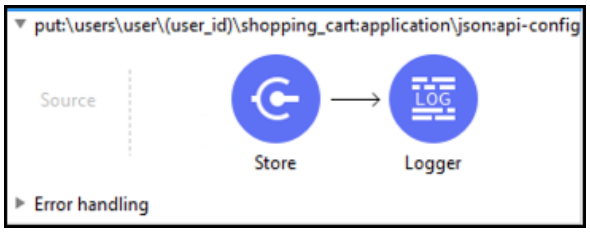
get:\products\search

This flow returns products from the database by calling the Product API



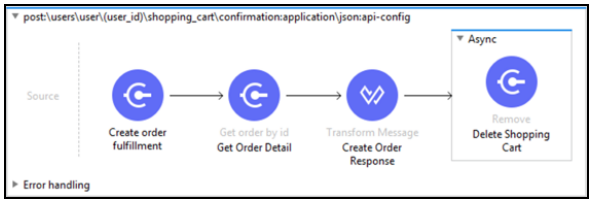
get:\users\user\{user_id}\shopping_cart

This flow returns all the items in a users shopping cart based on the user id



put:\users\user\{user_id}\shopping_cart

This flow puts an item into a users shopping cart based on the user id



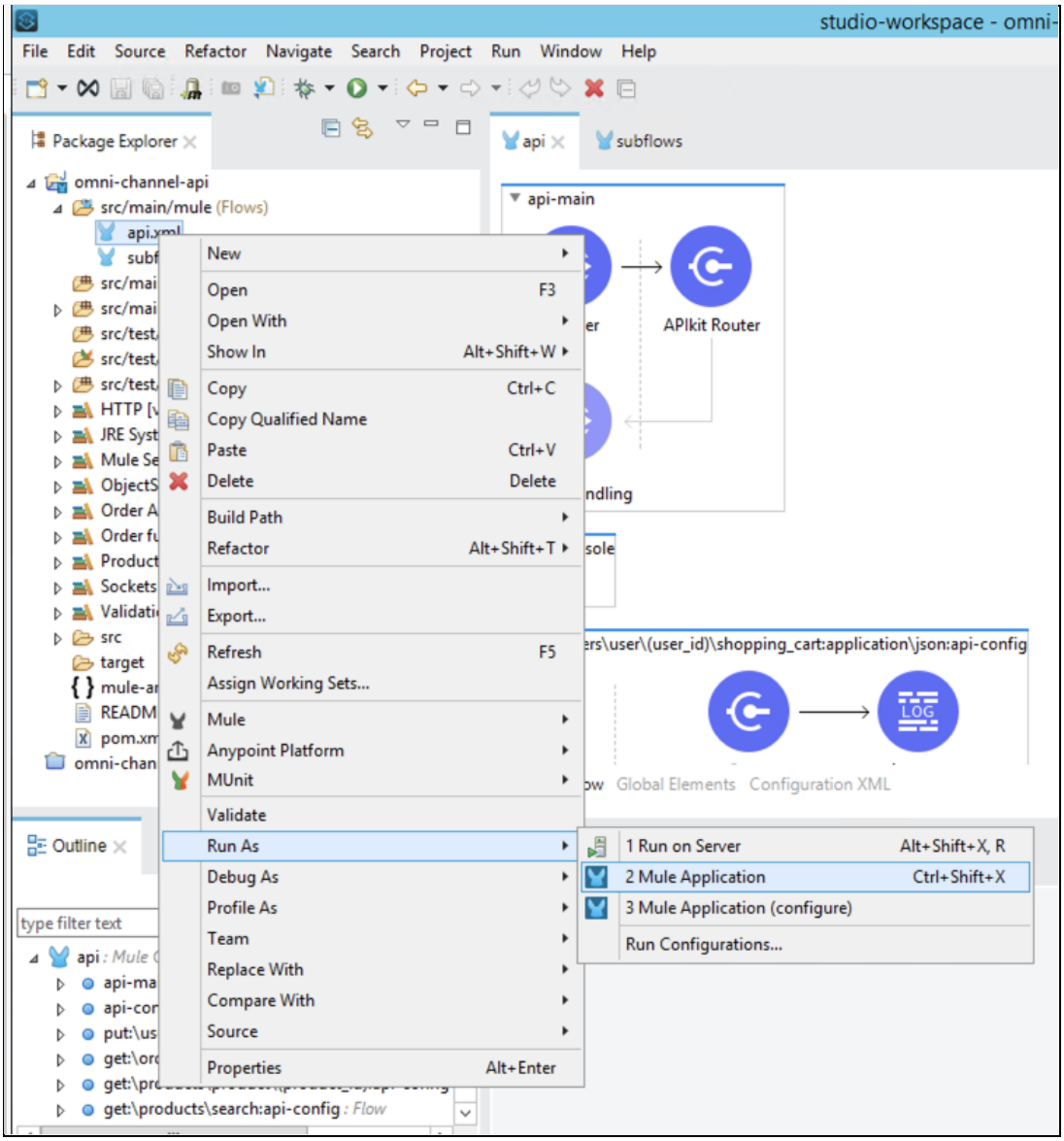
post:\users\user\{user_id}\shopping_cart\confirmation

This flow confirms the items to purchase in a users shopping cart.

Step 3: Test the application

To test the API, let's run it within Studio first.

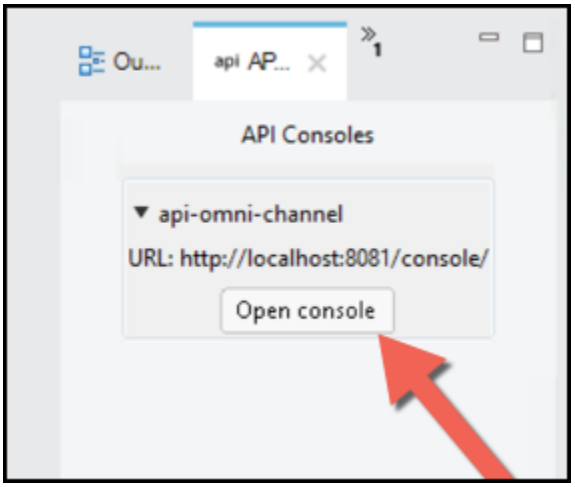
1. Right click the application.
2. Select **Run As > Mule Application**.



3. The application will start running, and the console will show the Mule Runtime logs

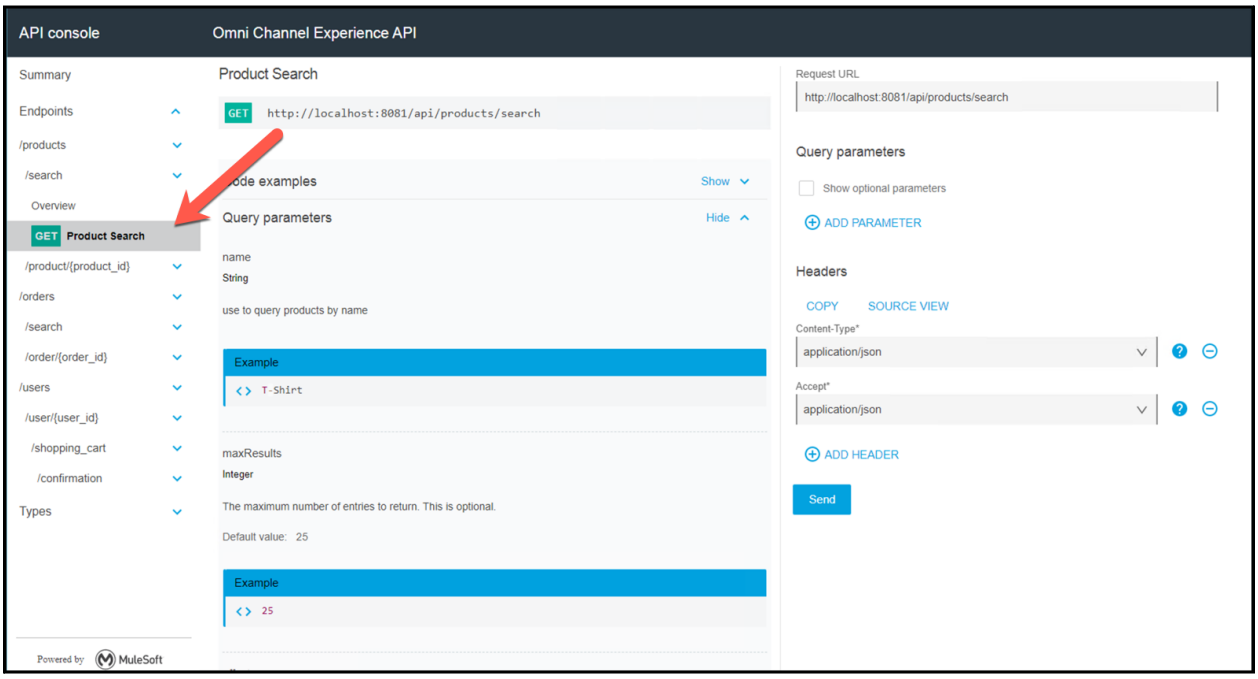
```
Mule Properties   Problems   Console   APIkit Consoles (eapi-onmi-channel-40)
eapi-onmi-channel-40 [Mule Applications] Mule Server 4.0.0 EE
*  - APIkit                                                          *
*  - Customer API                                                    *
*  - Sockets                                                         *
*  - HTTP                                                            *
*  - Order fulfillment API                                           *
*  - ObjectStore                                                     *
* Application libraries:                                             *
*  - eapi-onmi-channel-40                                            *
**********************************************************************
INFO  2017-12-25 18:03:43,181 [WrapperListener_start_runner] org.mule.runtime.module.deployment.internal.DeploymentDirectoryWatcher:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Mule is up and kicking (every 5000ms)                             +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
INFO  2017-12-25 18:03:43,184 [WrapperListener_start_runner] org.eclipse.jetty.server.ServerConnector: Started ServerConnector@3a8a10f9{HTTP/1.1}{0.0.0.0:56838}
INFO  2017-12-25 18:03:43,186 [WrapperListener_start_runner] org.mule.runtime.module.deployment.internal.StartupSummaryDeploymentListener:
**********************************************************************
*              - - + DOMAIN + - -               * - - + STATUS + - - *
**********************************************************************
* default                                       * DEPLOYED           *
**********************************************************************

*****************************************************************************************************
*            - - + APPLICATION + - -            *       - - + DOMAIN + - -       * - - + STATUS + - - *
*****************************************************************************************************
* eapi-onmi-channel-40                          * default                        * DEPLOYED           *
*****************************************************************************************************
```

4. Test the application using the console. Click the **Open Console** on the APIkit Console tab.

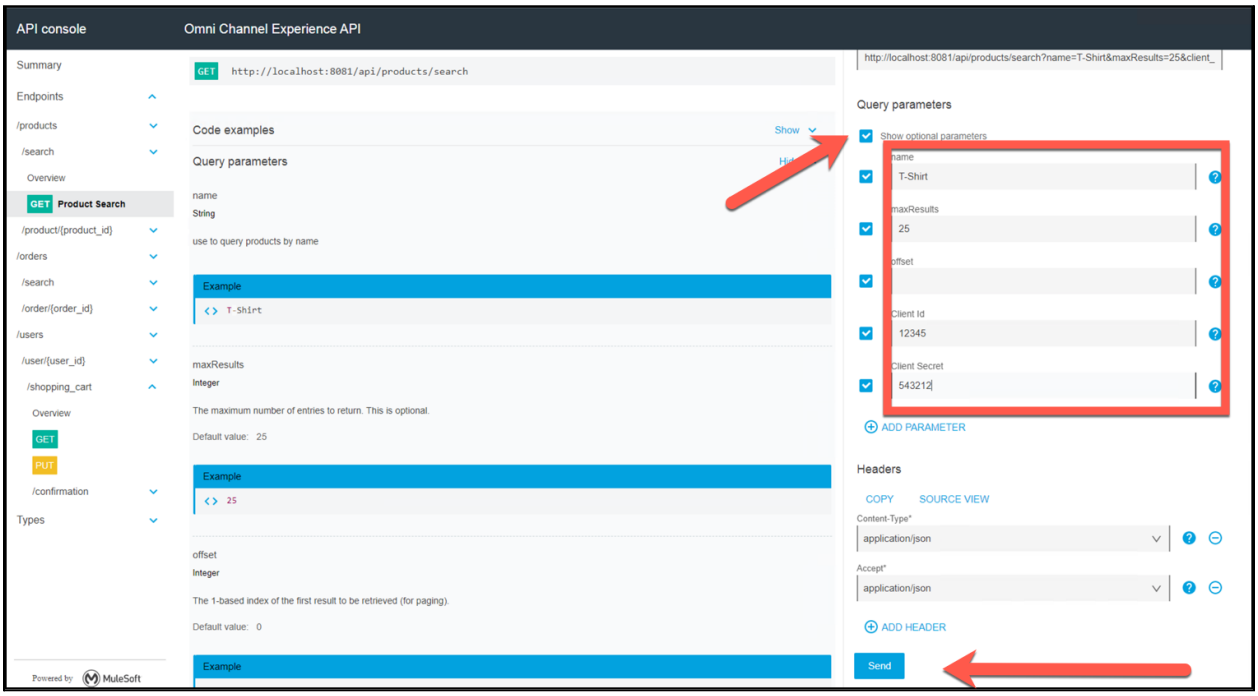


5. A browser window opens at: **http://localhost:8081/console/**.

6. Click open the **/products/search** resource on the left and click **GET**.



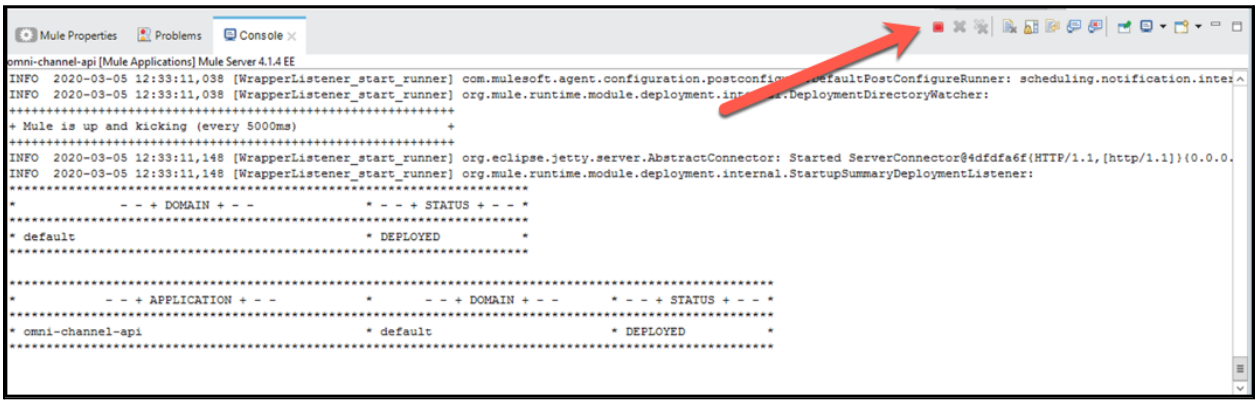
7. Then tick the **Show optional parameters** and fill them in as below and click **SEND**.



8. Unlike Lab 1, this project is a complete solution and is making calls to System and Process REST APIs. The response you see in the Console is coming from other sources.

Let's go ahead and stop the application to get ready for the next lab.

9. Go to the console tab and press the red squared button to stop the Mule runtime server.



Summary

In this lab, we

- Step 1: Import the project from Anypoint Exchange
- Step 2: Examine the different flows within the project
- Step 3: Test the application

This lab shows how quick and easy it is to enable your teams to get up and running quickly with pre-built project templates.

- See the REST Connectors documentation for more information.
- See the Flow Control Routers documentation for more information.
- See the Scopes documentation for more information.

Congratulations! You have completed Lab 2.

Please proceed to Lab 3

Take me to the TOP

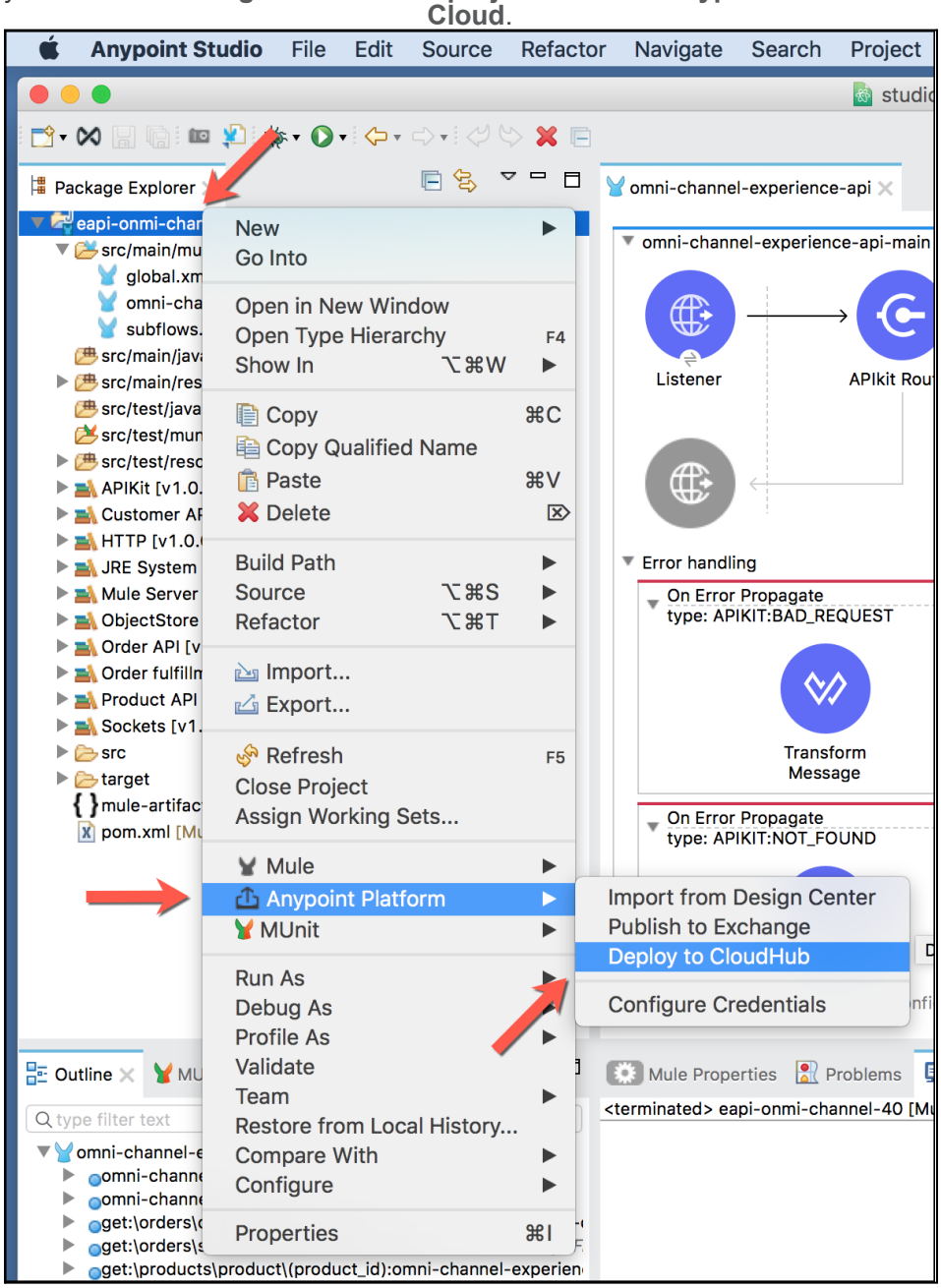
Lab 3: Deploy your API

Overview

In this lab we'll deploy the complete Omni Channel API from Lab 2 to the Cloudhub iPaaS Platform from Anypoint Studio itself the same way a developer would.

Step 1: Deploy the Complete Omni Channel API to Cloudhub

Let's deploy it to Cloudhub. **Right click on the project name → Anypoint Platform → Deploy to Cloud.**

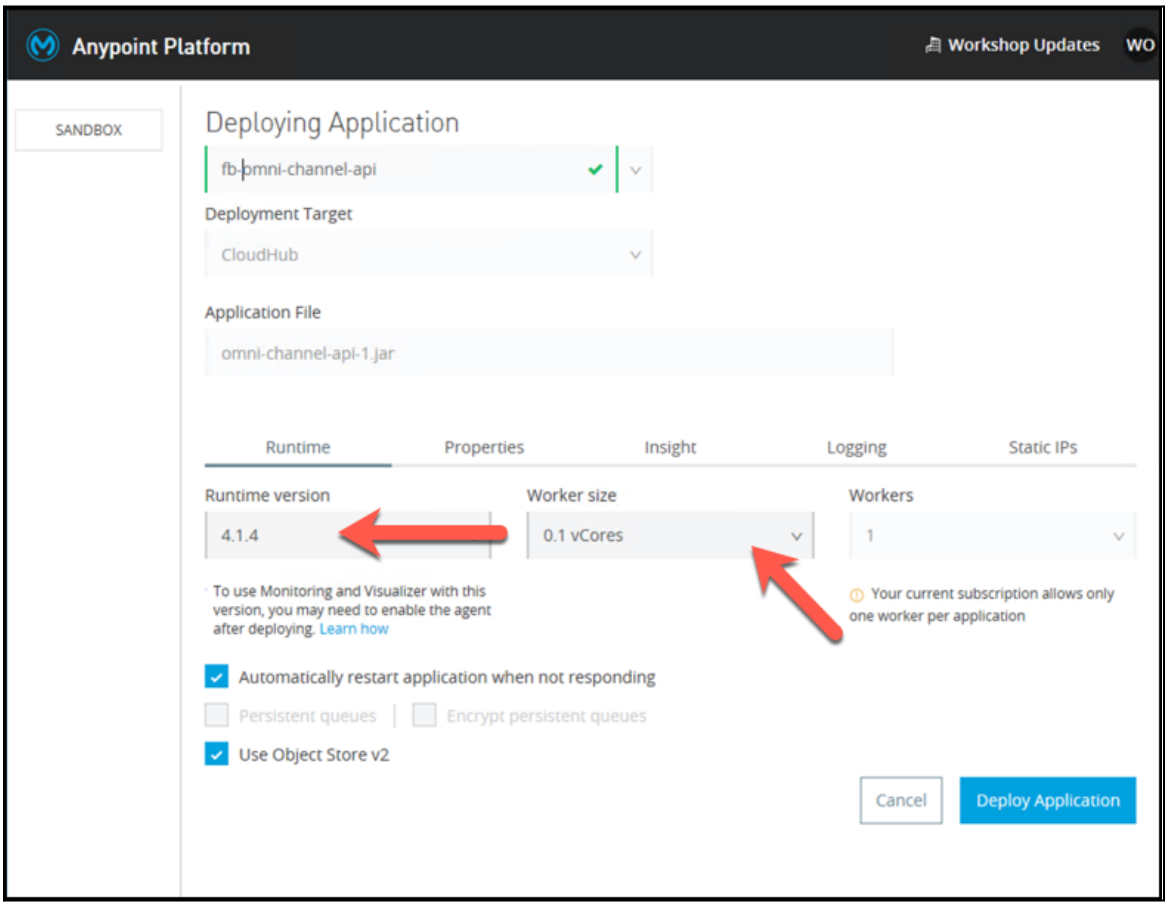


1. If this is your first time deploying in this way, a popup menu asks you to provide your login credentials for Anypoint Platform. Studio stores your credentials and uses them automatically the next time you deploy to CloudHub.
2. Once logged in to the Anypoint Platform one needs to select an **Environment** and **Business Group**. Choose the one **Business Group** you have access to and select as an environment

Sandbox.

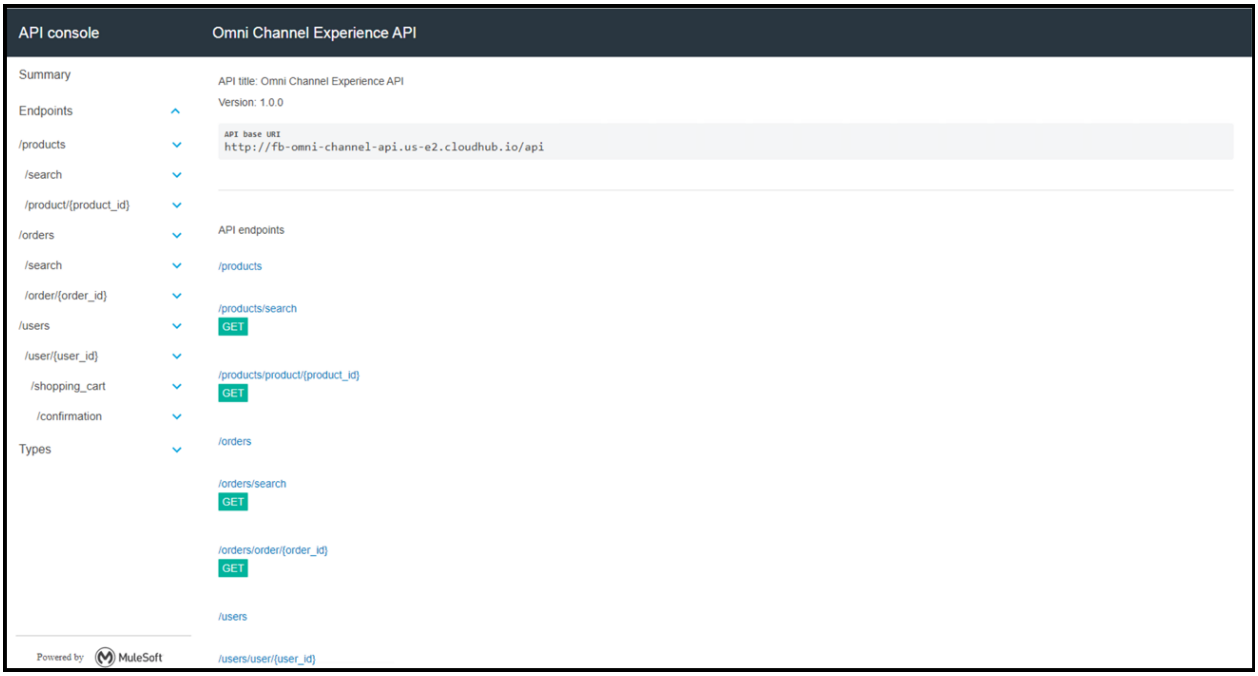
1. Now we need to specify application (worker) configuration for our API.
   a. On the application name put <<MyName-omni-channel-api>>. You should see a green check if the name is available. If not, please use another name
      i. Save this name as you will use it later on. An example would be **donald-omni-channel-api**
   b. On **Runtime version** select **4.3.0**

c. On **Worker Size** select **0.1 vCore** and **SANDBOX** as the environment



2. Once the configuration is done, click on the **Deploy Application** button.
3. Once the deployment is complete (it may take a few minutes), you can browse the API Console at http://<<username-omni-channel-api>>.<region>.cloudhub.io/console/.
4. In the previous example it would be something like

   http://donald-omni-channel-api.us-e2.cloudhub.io/console/.



Now you have a fully functional **Omni Channel API** of your own.

Step 2: Test the API from a Mobile Application

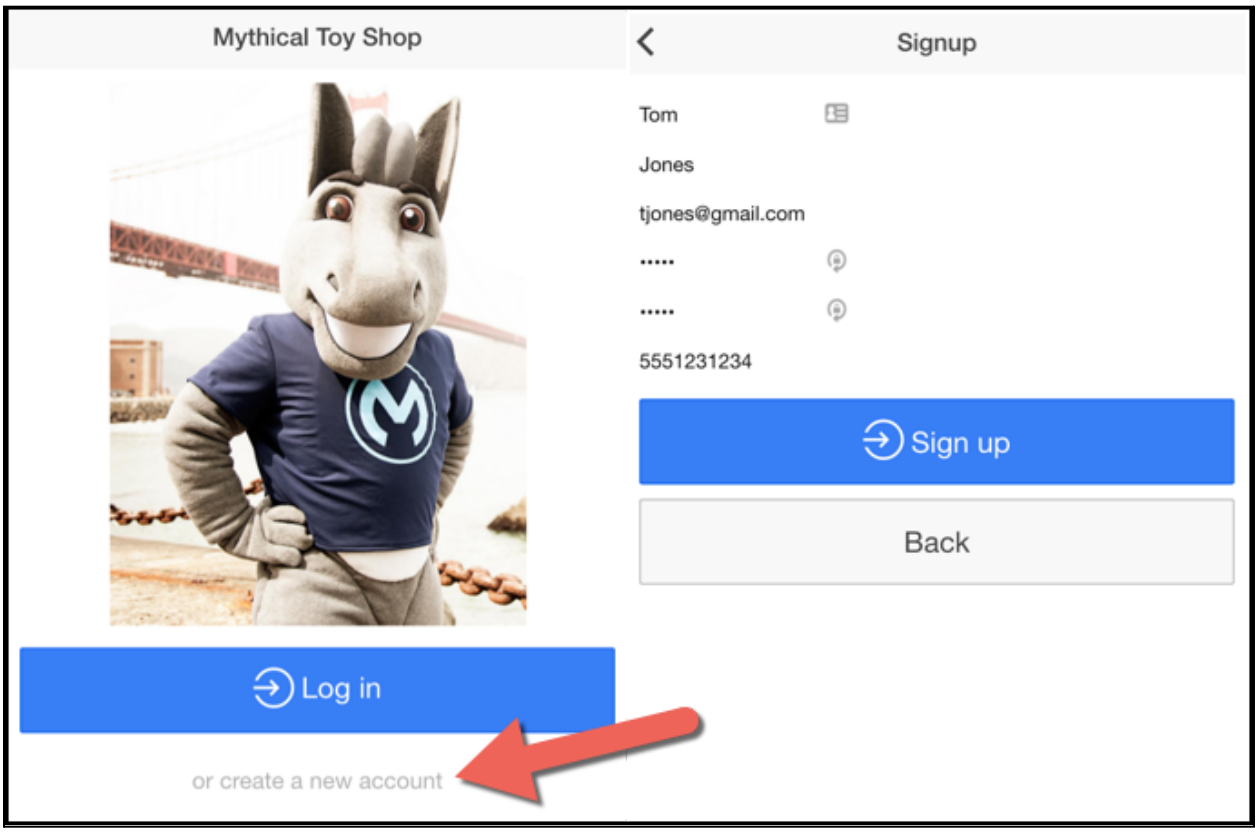
Your API application is running on CloudHub, lets try to hit it using a mobile application. We have developed a mobile application to test our APIs. This mobile application will invoke your REST API using the CloudHub URL.

You can access the mobile application through:

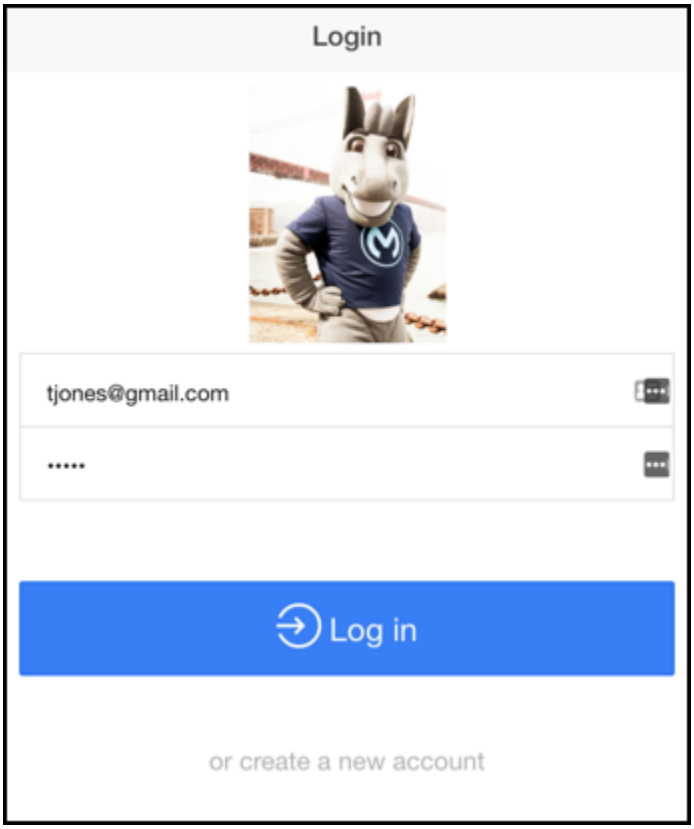
http://services.mythicalcorp.com/mobile

Follow the instructions there to access an emulator of the mobile app or download the mobile app to your phone.

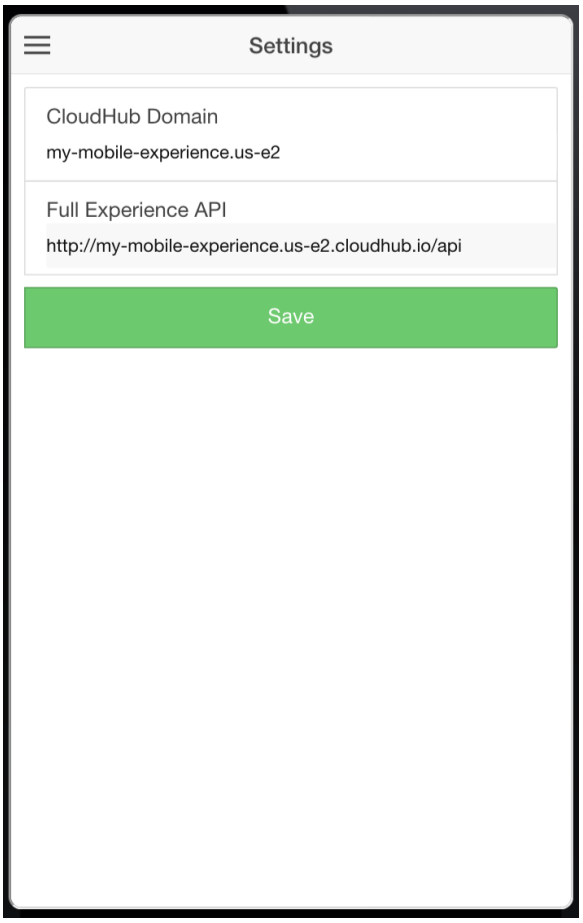
1. Click on **create a new account** and create an account to access the Mythical Mobile Commerce App



2. Login using your email address. The mobile app is not enforcing any authentication yet, so the password can be anything.

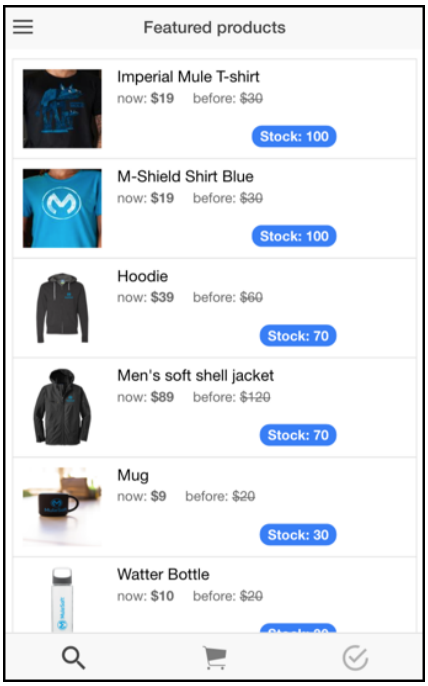


3. To connect this to your Mobile Experience API, go to Settings and add the CloudHub URL for your Mobile Experience API with the corresponding region.



The mobile app will communicate with your Experience API to bring your design to life with API

Led Connectivity!



Summary

In this lab, we:

- Deployed our Omni Channel API application to Anypoint Platform's CloudHub directly from Studio. CloudHub provides you a way to manage your applications in the cloud or servers located on-premises.
- Test the application through the Mobile App.

Go Further:

- See the CloudHub doc for more information.
- See the Deploy to CloudHub doc for more information.

Congratulations! You have completed Module 2.

Please proceed to Module 3

Take me to the TOP