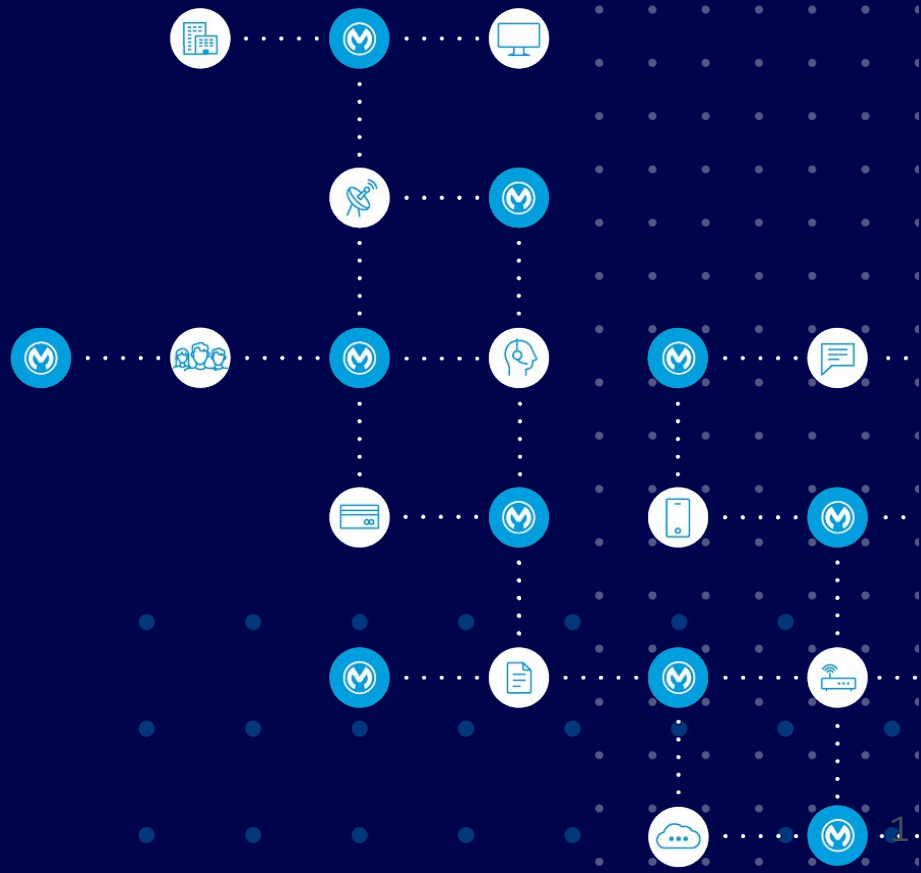
MuleSoft

Welcome!

Thank you for joining this API-led Connectivity Workshop for Direct to Consumer (DTC)

Module Lab Guide



Thank you!

MuleSoft

Module 0: Prerequisites

Source: DTC e-commerce: How consumer brands can get it right

Module 0: Prerequisites

Overview

Welcome to the MuleSoft API Led Connectivity Workshop, In this lab we will show you the prerequisites setup that are needed to start the workshop.

Following Workshop Instructions

Functions or actions to be performed by you (for example, menu selections, buttons to click, …) are for the most part in **bold**.

Text to be entered will be in a *italic* font.

The general format for the workshop docs is to provide you instructions, followed by a screen capture of what you are doing.

1. Click on some button
2. Enter some text
3. Click Save
4. …

Module 0: Prerequisites

Workshop Resources

1. **Guide Content:** The guide is also available at the following: https://muleyoscar.github.io/dtc-workshop/

2. **Lab Examples:** from code snippets to complete projects can be found in the Git repo: https://github.com/muleyoscar/dtc-workshop

3. If you can not access your virtual machine and would still like to follow along you can download the **Anypoint Studio** here: https://www.mulesoft.com/lp/dl/studio

MuleSoft

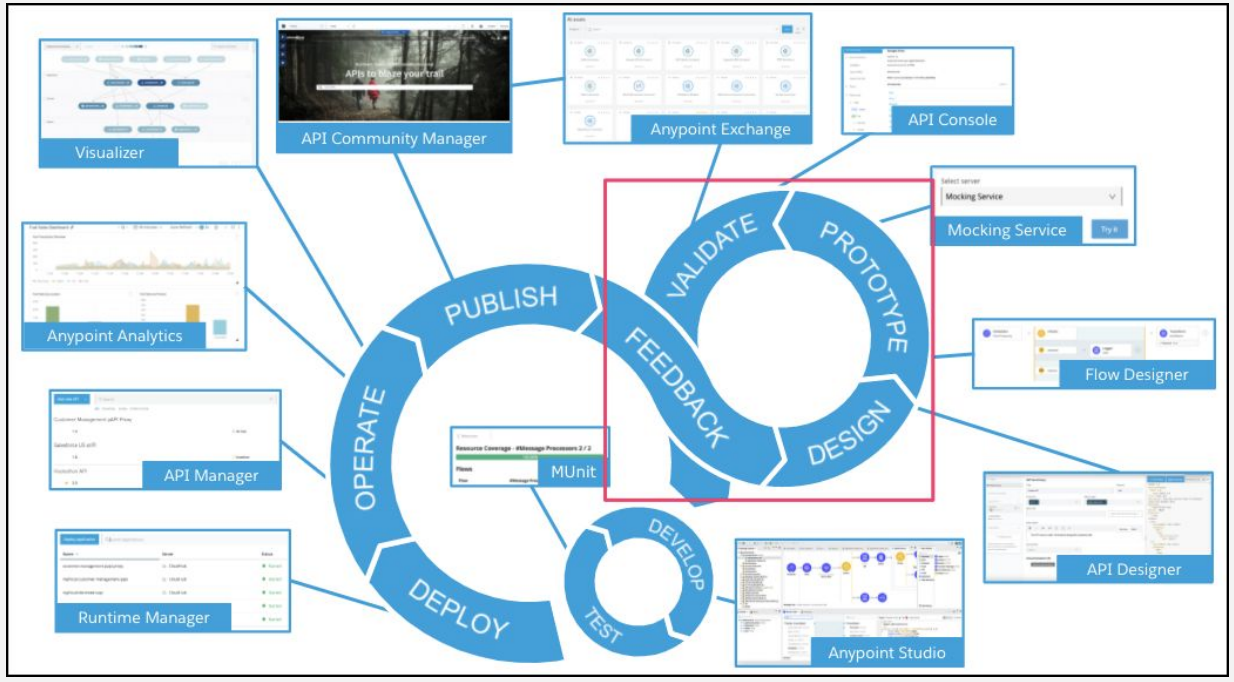
Module 1: Modern API Design: Overview

Source: DTC e-commerce: How consumer brands can get it right

Module 1: The Modern API - Design: Overview

Overview

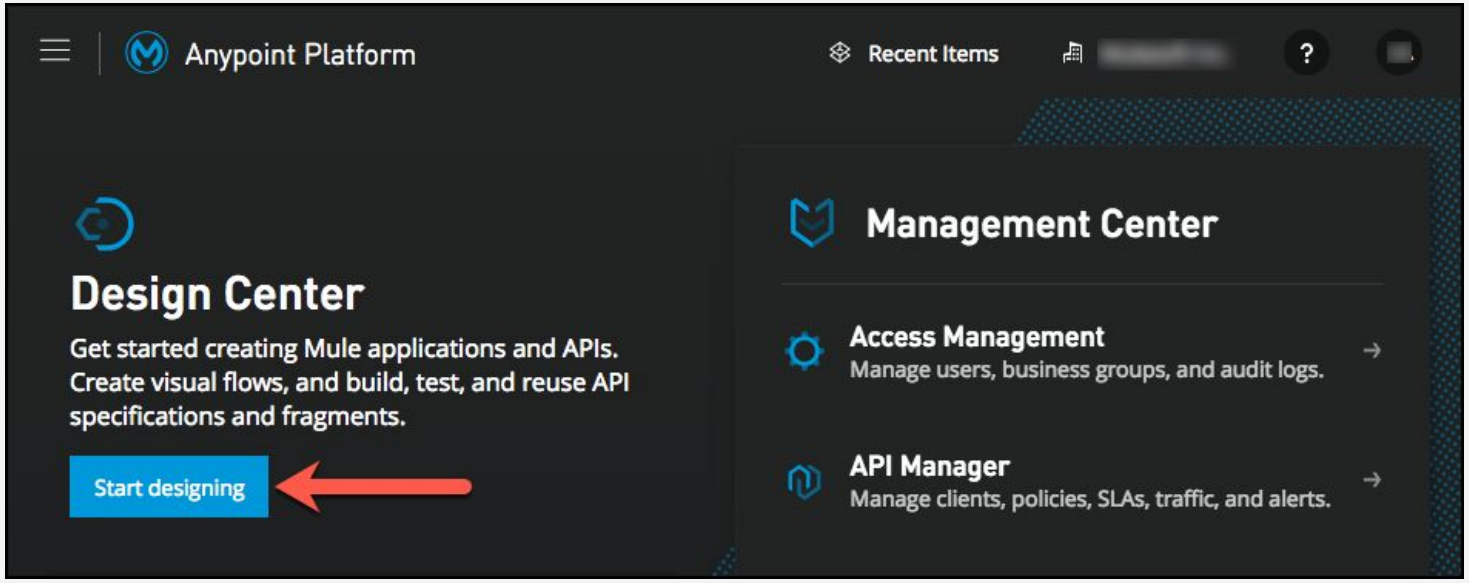
To get started with the workshop, let's take a look at how to discover, design, and publish an API specification using the **Anypoint Platform**. Mulesoft advocates for users to adopt a "design first" approach to creating API's. A "design first" approach is used to enable API consumers the ability to understand, interact, and solicit feedback on the proposed API contract prior to the development effort.



Module 1: The Modern API - Design: Overview

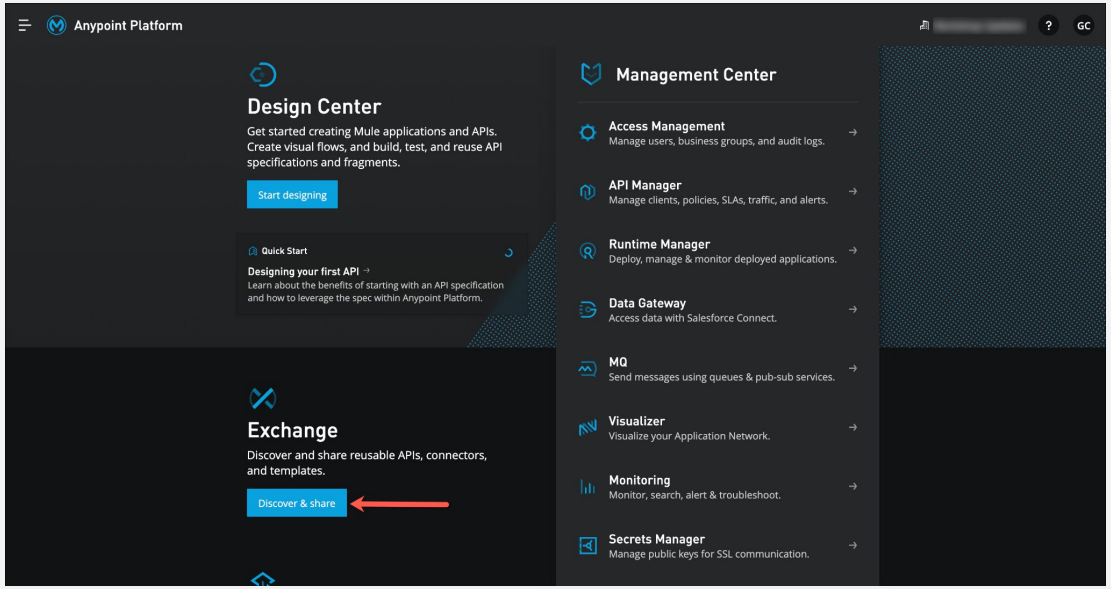
In this first module we will explore the capabilities of the **MuleSoft's Anypoint Platform** to design, publish, and discover API specifications.

Anypoint Design Center provides a set of interactive tools that enable API designers to create or modify the specifications for an API. These specifications form the underlying API contract between the provider and the consumer of the API. **Anypoint Design Center** also enables the development of API Portals, documentation used to communicate to developers how the API should be consumed.



Module 1: The Modern API - Design: Overview

API discovery and collaboration is provided by **Anypoint Exchange** and simplifies the way that reusable assets are discovered and consumed across the enterprise. We will explore the features of **Anypoint Exchange** in detail in this first module.



The first module of the workshop will focus on how to use **Anypoint Design Center** and **Anypoint Exchange** to design, document, publish, and discover the features of a modern API.

The 3 labs in this unit are:

- Lab 1: Search for an API in Exchange
- Lab 2: Design the Shopify API
- Lab 3: Publish the Shopify API to Exchange

Proceed to Lab 1

MuleSoft

Module 1: Lab 1 Search For an API in Exchange

Source: DTC e-commerce: How consumer brands can get it right

Module 1: The Modern API - Design: Lab 1 (1/14)

Lab 1: Search for an API in Exchange

Overview

API's are the reusable assets that simplify and accelerate the creation of modern software applications. As a Mulesoft developer you will need to consume API's created by other members of the organization and publish new API's for others to consume. This new consumption model is the foundation of a new approach for delivering software solutions where API's form the building blocks of the modern enterprise.

This first lab will focus on using **Anypoint Exchange** to search for API's and other assets published in the private exchange.

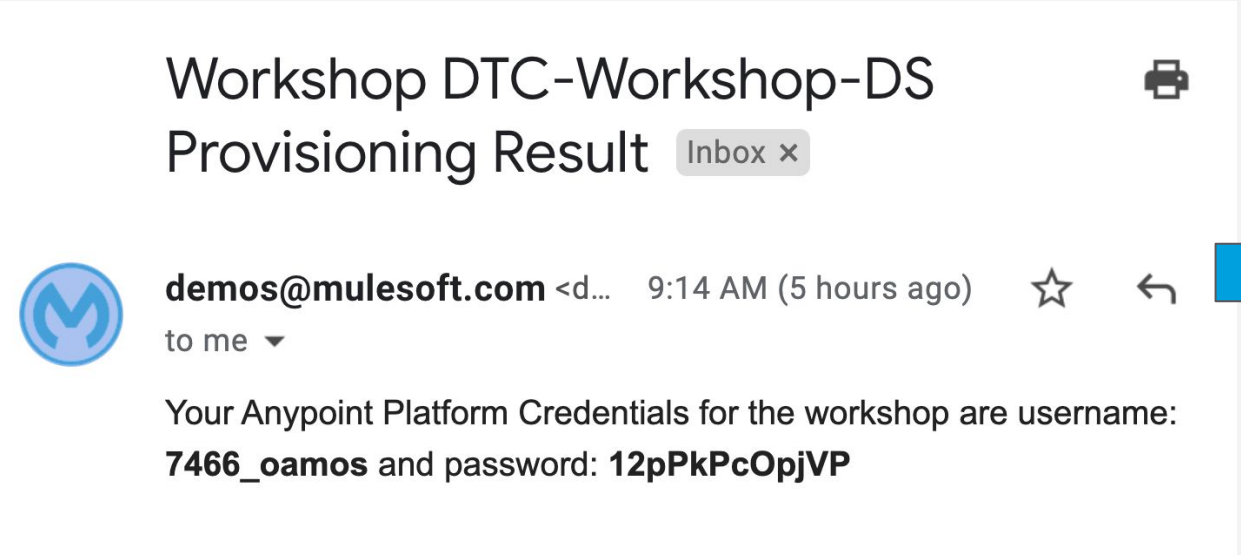
Module 1: The Modern API - Design: Lab 1 (2/14)

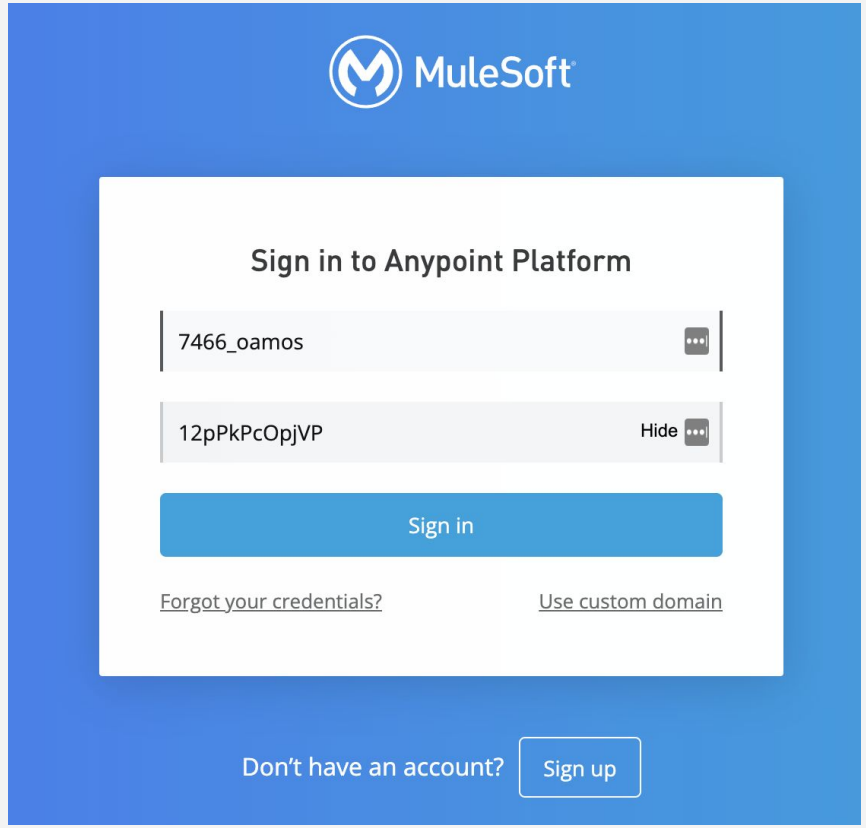
Step 1: Login to Anypoint Platform

1. Go to http://anypoint.mulesoft.com
2. Enter **your credentials** into the Username and Password fields.

   You should have received an email from demos@mulsoft.com

   with contents similar to the below:

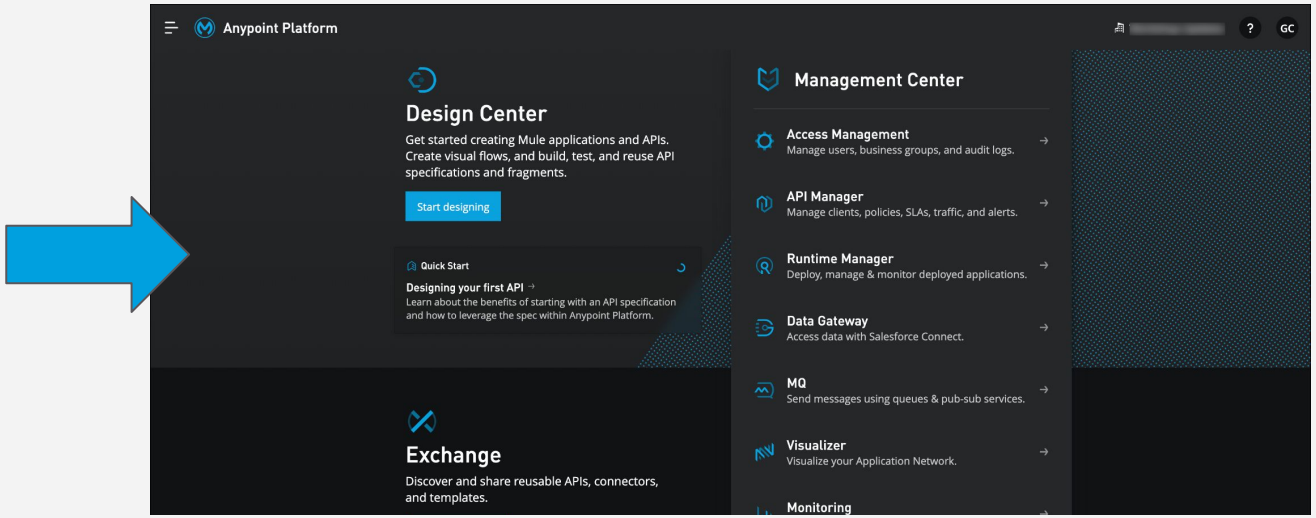




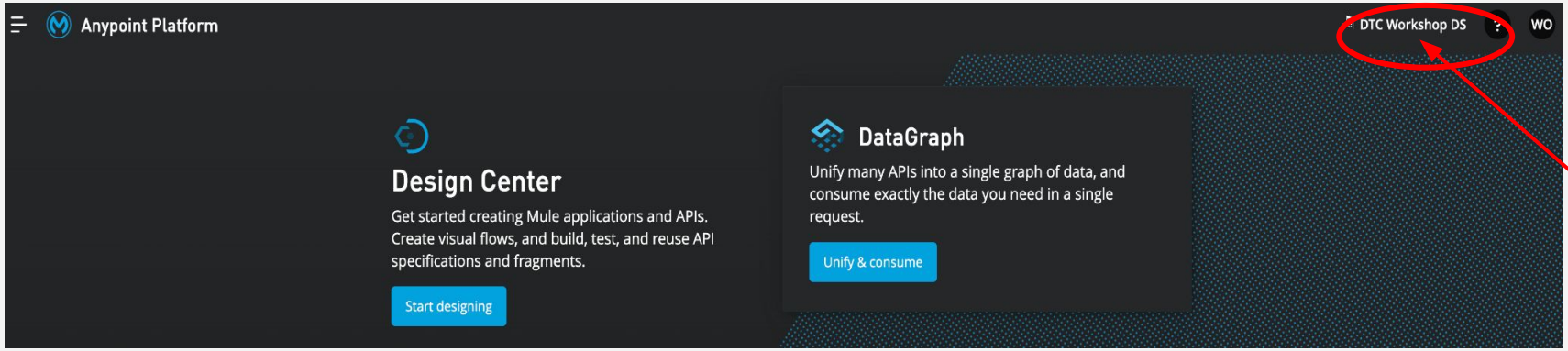
WARNING: If you already have an Anypoint login, they system will ask if you want to join the DTC Workshop with you existing account. DO NOT use you existing account select "Join with new account" option below the message.

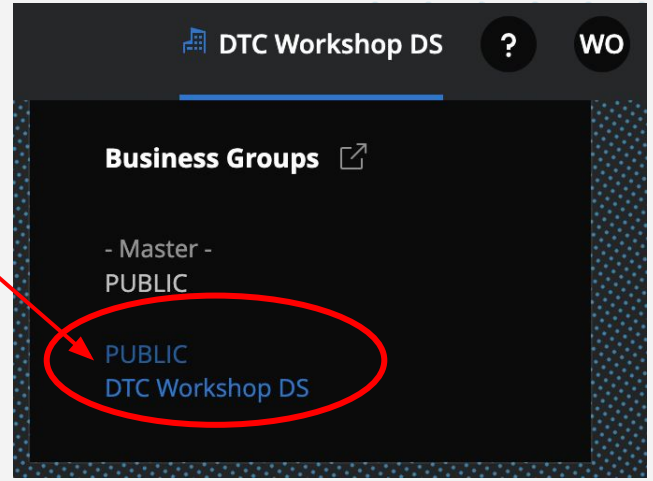
Module 1: The Modern API - Design: Lab 1 (3/14)

3. You should see the following landing page once you are logged in to Anypoint Platform.



4. Click on the Business Group that is on the right top corner of the screen. The name will vary with the class. Select sub group that has "**DTC Workshop…**" labelled.

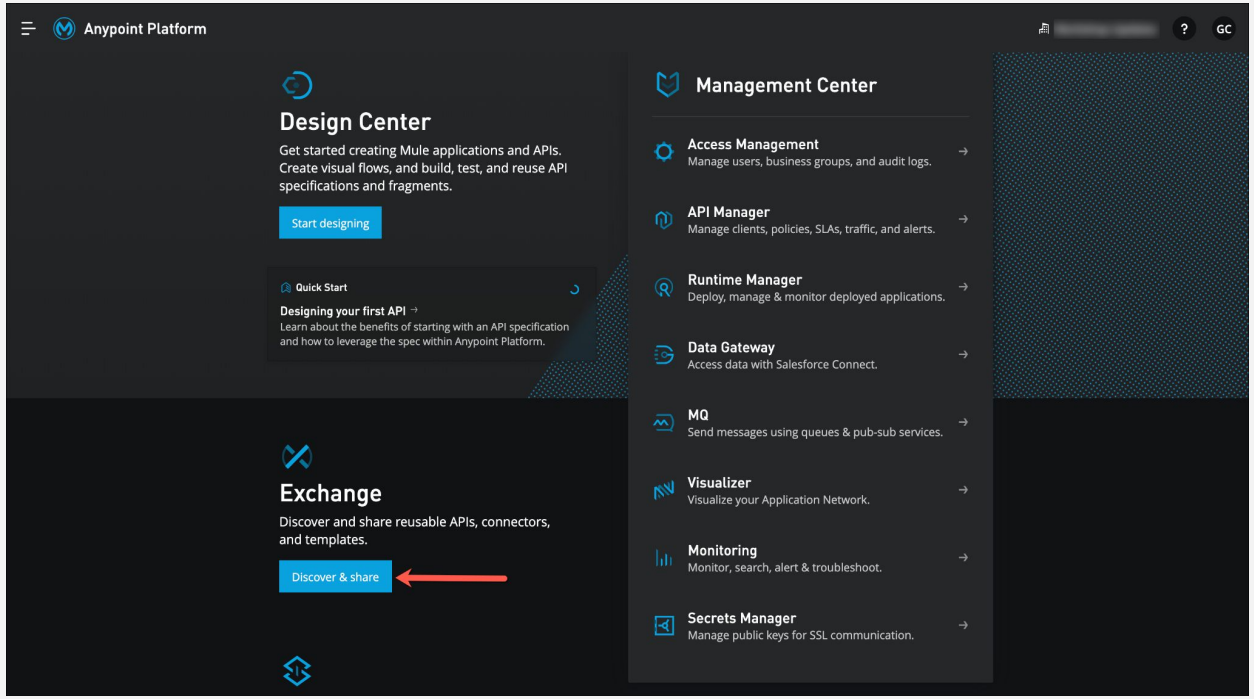




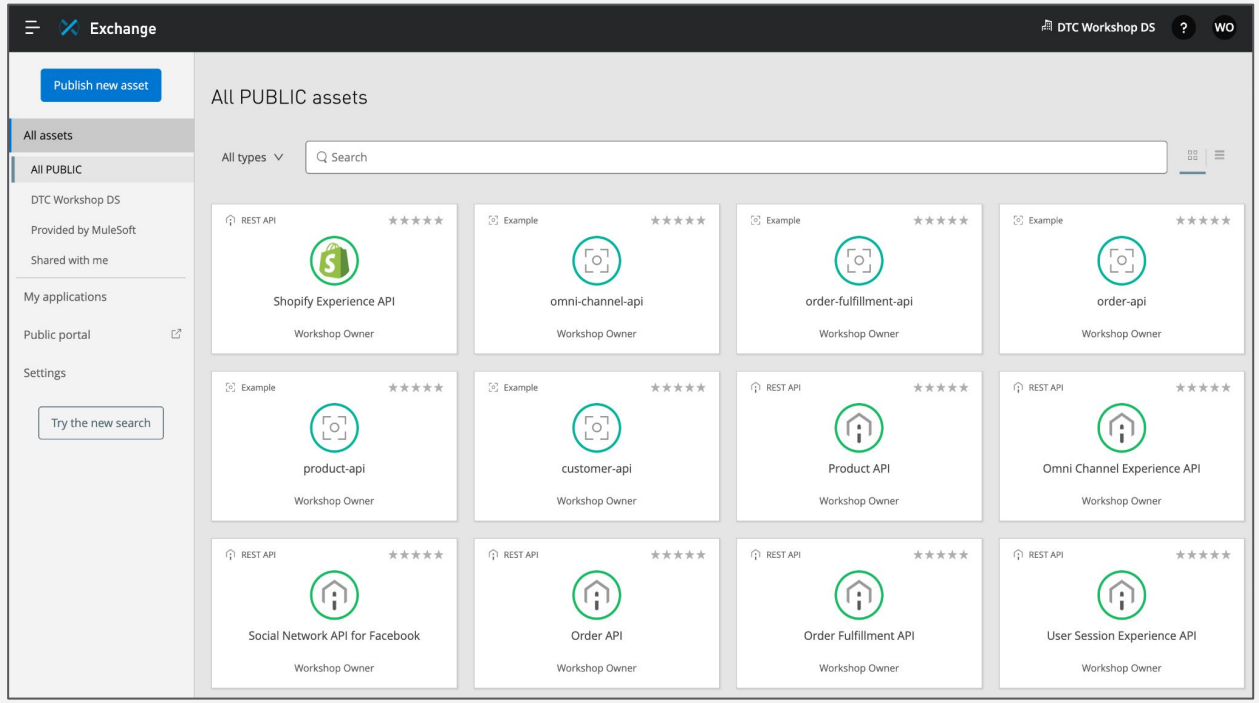
Module 1: The Modern API - Design: Lab 1 (4/14)

Step 2: Access Anypoint Exchange

1. Click on the icon labeled: **"Discover & share"**



2. You will now be presented the landing page for the **Anypoint Exchange** portal:

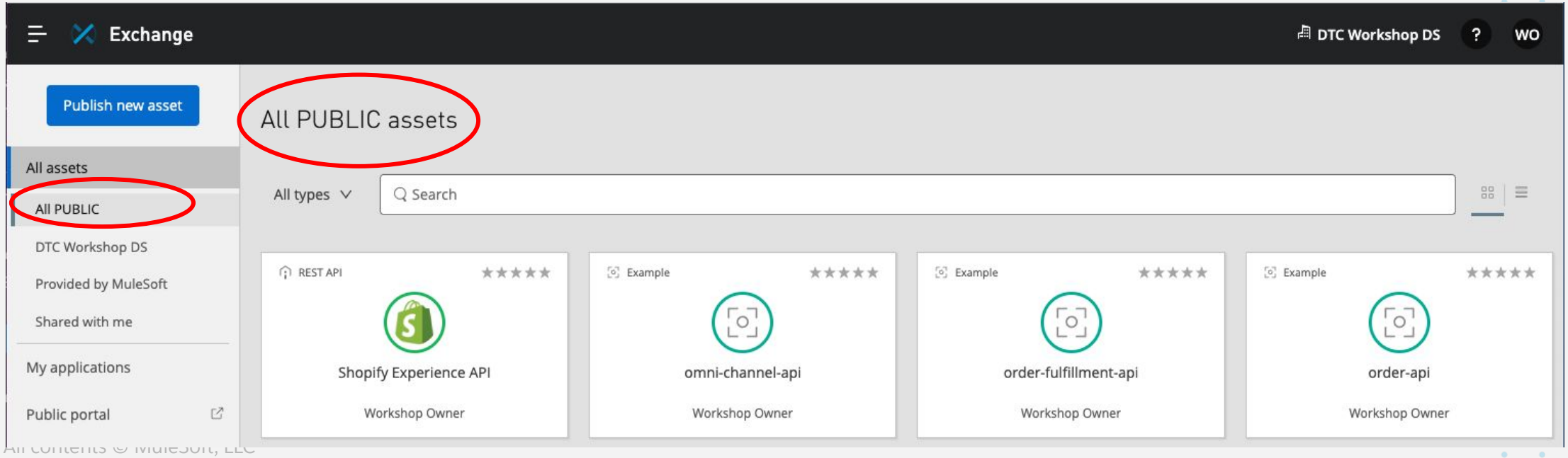


Anypoint Exchange should be your initial starting point for just about any project. If you are looking to reuse an API then **Anypoint Exchange** is obviously where you should start. But even if you are creating a new API to be reused by others, it makes sense to first look in Exchange to see if the API already exists. Sometimes you will find that someone else has already undertaken the task of creating the API, or has created a subset of what you need.

Module 1: The Modern API - Design: Lab 1 (5/14)

3. **Anypoint Exchange** allows for discovery on the *public* or *private* exchange. Every organization will automatically have a private exchange created for them.

Software assets stored in **Anypoint Exchange** can only be discovered by users that have permissions for the private exchange for that organization. Furthermore, organization administrators can also create independent *business groups* to provide even more restrictive discovery. **Anypoint Exchange** users can only search within their currently selected business group. The currently selected business group, "Public" in this example, is shown.

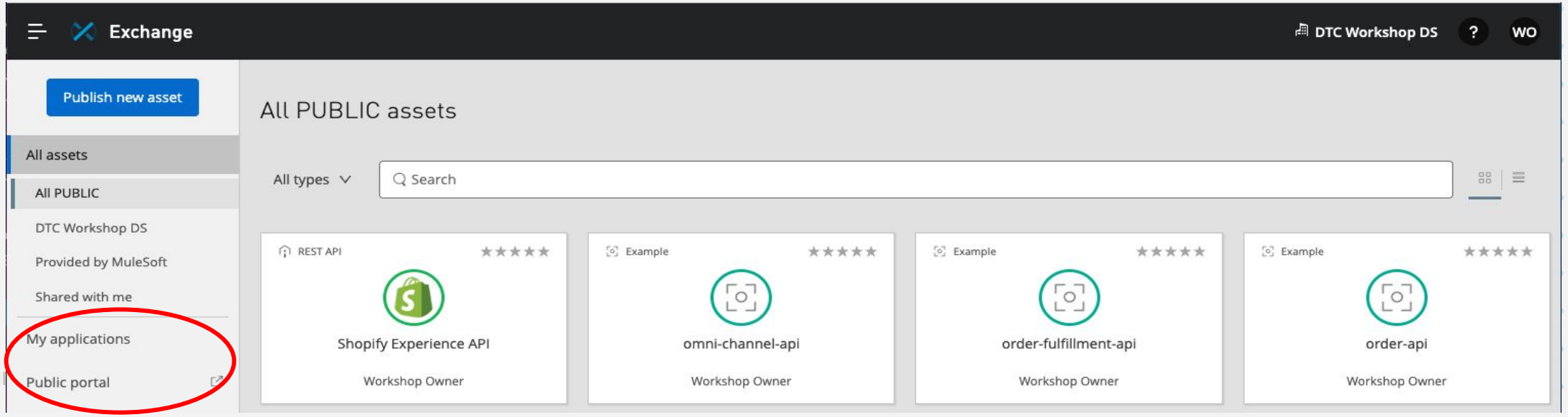


Module 1: The Modern API - Design: Lab 1 (6/14)

As you can see from the screen above, there are organizational search criteria on the left hand side of the page. By default searches will search the private exchange only, but you can restrict or expand your searches by selecting a different organizational level. Select **All assets** to select both MuleSoft public and your private repositories.

There are two more items behind the last organization **My applications** and **Public Portal**. We will describe them but they are covered in Module 3 and Lab 4.

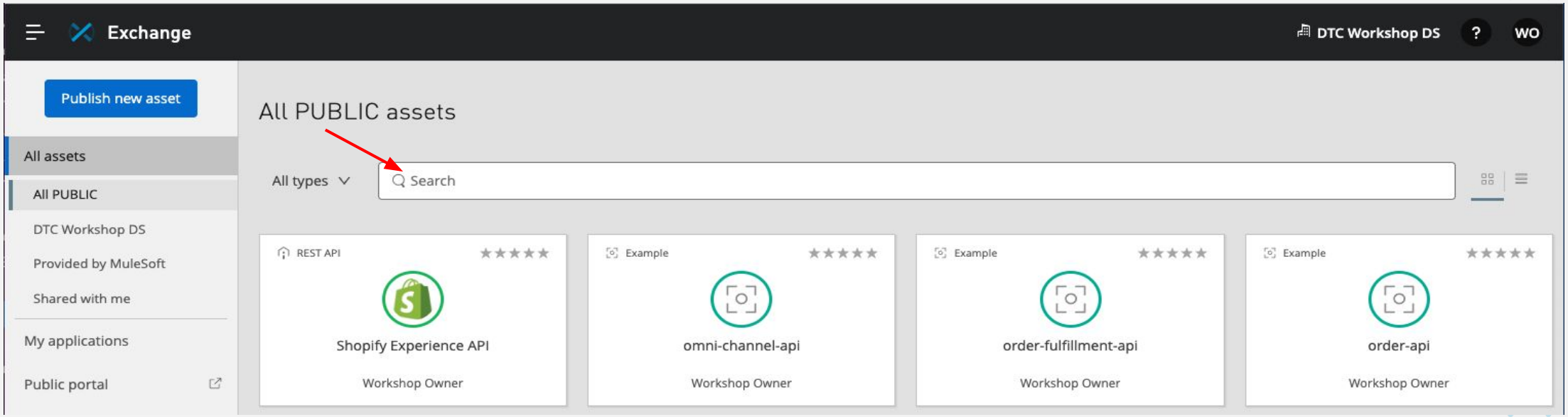
- **My applications**: These are the clients registered in the organization that are consuming the APIs that are deployed in the organization.
- **Public Portal**: You can define a public portal to let any user to consume any public org API that the organization may publish.



Module 1: The Modern API - Design: Lab 1 (7/14)

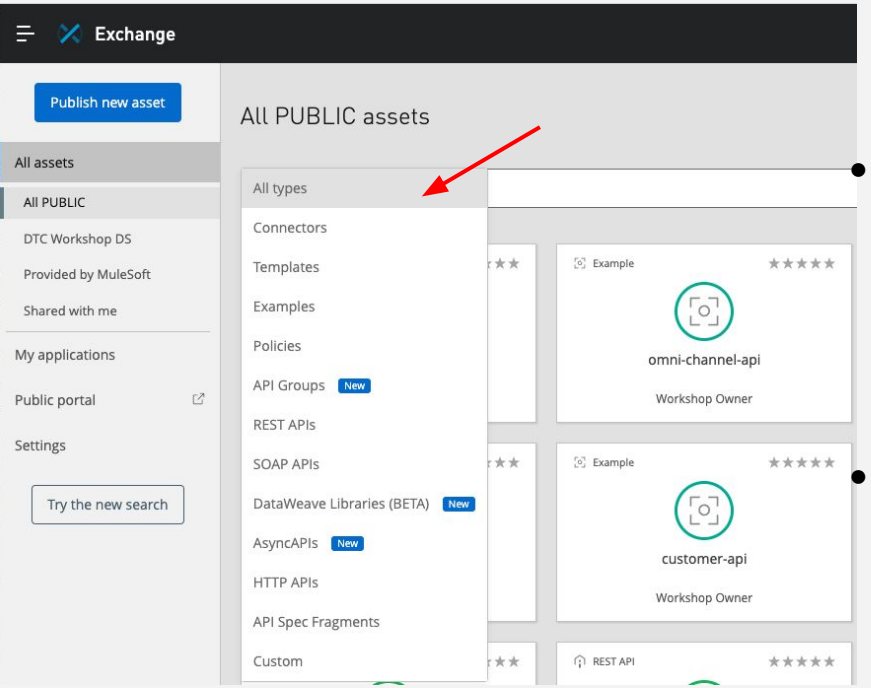
To search for an API, use the search bar at the top of the Anypoint Exchange portal:

Anypoint Exchange supports keyword searching to find the API you are looking for. However, **Anypoint Exchange** can be used as a discovery tool for more than just API's. **Anypoint Exchange** is the enterprise repository for the following types of reusable software assets:



Module 1: The Modern API - Design: Lab 1 (8/14)

4. Click on the **All types** drop down next to the search field to view the different search types:



- **Connectors** - Packaged connectivity to an endpoint developed and deployed on MuleSoft's Anypoint Platform with third-party APIs and standard integration protocols.
- **Templates** - Packaged integration patterns that address common use cases and are built on best practices. Applications to which you add your scenario-specific information to complete a use case or solution.
- **Examples** - Implementation projects that explain development elements within Anypoint Studio and how these can be leveraged to achieve specific API and integration objectives.
- **Policies** - Configuration modules to extend the functionality of an API and enforce capabilities such as security.
- **API Groups** - A set of APIs bundled into a single asset. Instead of requesting access to multiple APIs to satisfy a use case, a developer can access the group in one step.
- **REST APIs** - API descriptions in RAML format that make the consumption of REST API's faster and easier.
- **SOAP APIs** - API descriptions in WSDL format that make the consumption of a SOAP API's faster and easier.
- **HTTP APIs** - A placeholder for an endpoint for use by private Exchange users who want to manage the endpoint with API Manager
- **API Spec Fragments** - Shared fragments of RAML files that can be used to assemble new API's with common traits.
- **Custom** - A general category for sharing resources such as links, blogs, articles, videos and more.

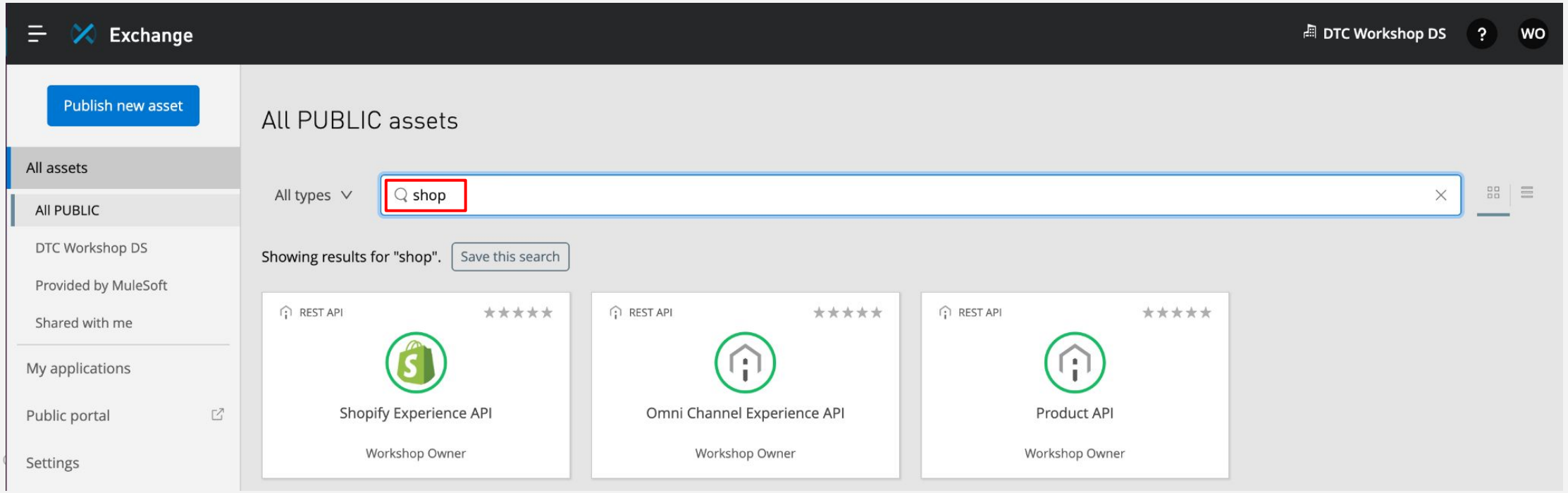
Module 1: The Modern API - Design: Lab 1 (9/14)

5. Click on the **All types** drop down again to collapse it.

6. Click on the **All Public** business group. The name can vary by class.

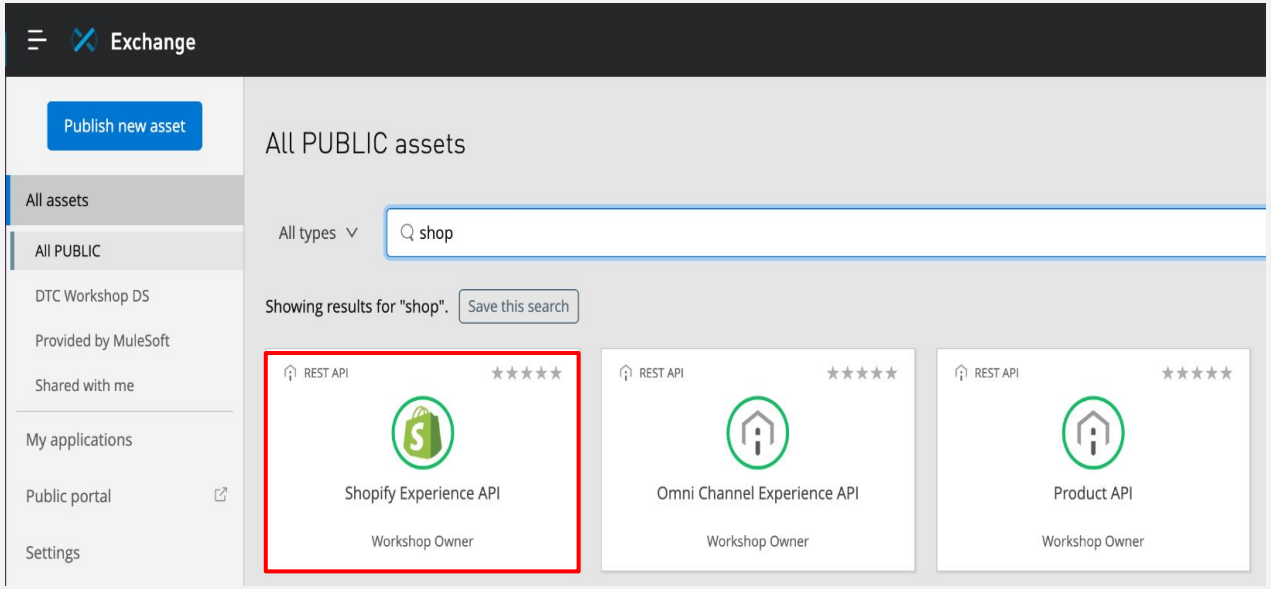
7. Click on the search field and type **"shop"**, then hit the **Enter Key**.

8. The list of Exchange assets should now be filtered to look something like this:

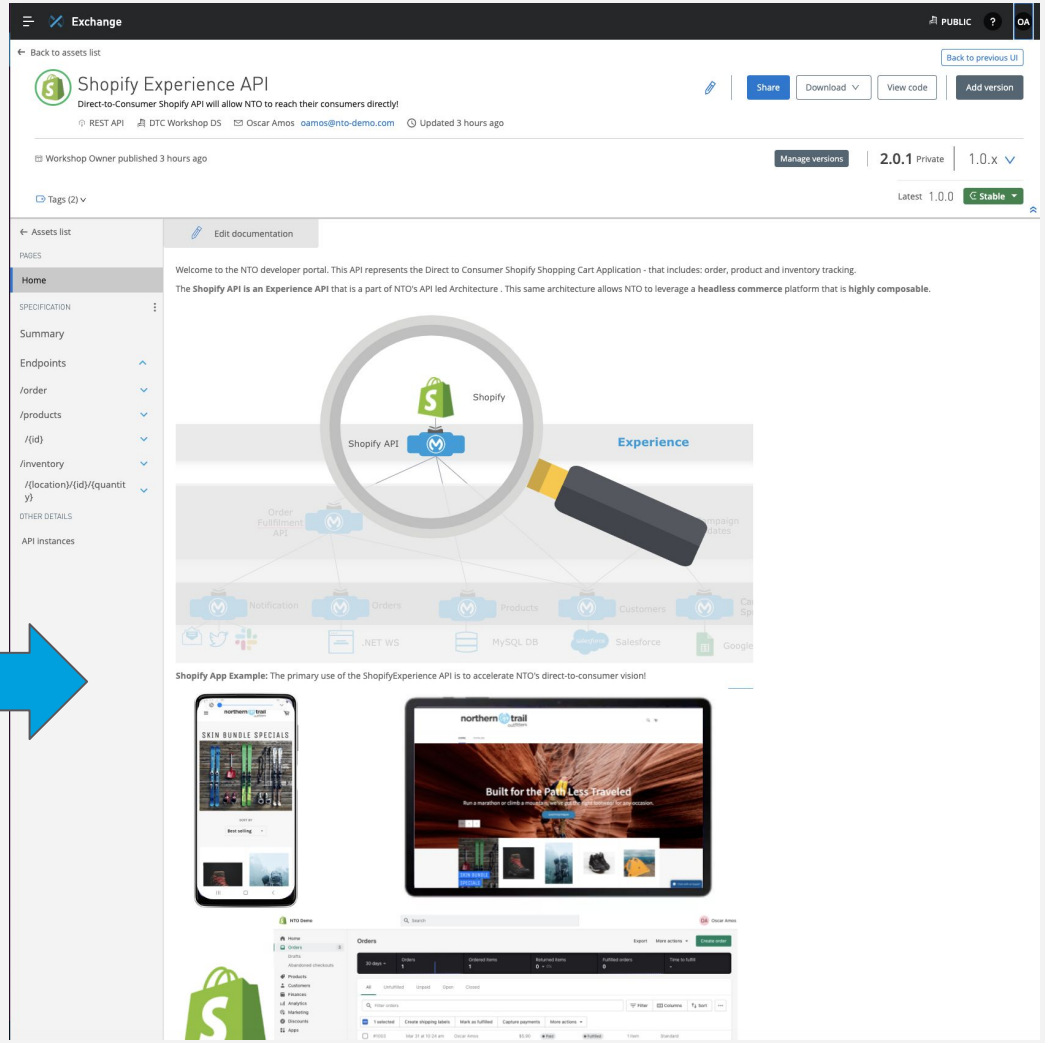


Module 1: The Modern API - Design: Lab 1 (10/14)

9. Click on the tile named **Shopify Experience API**.



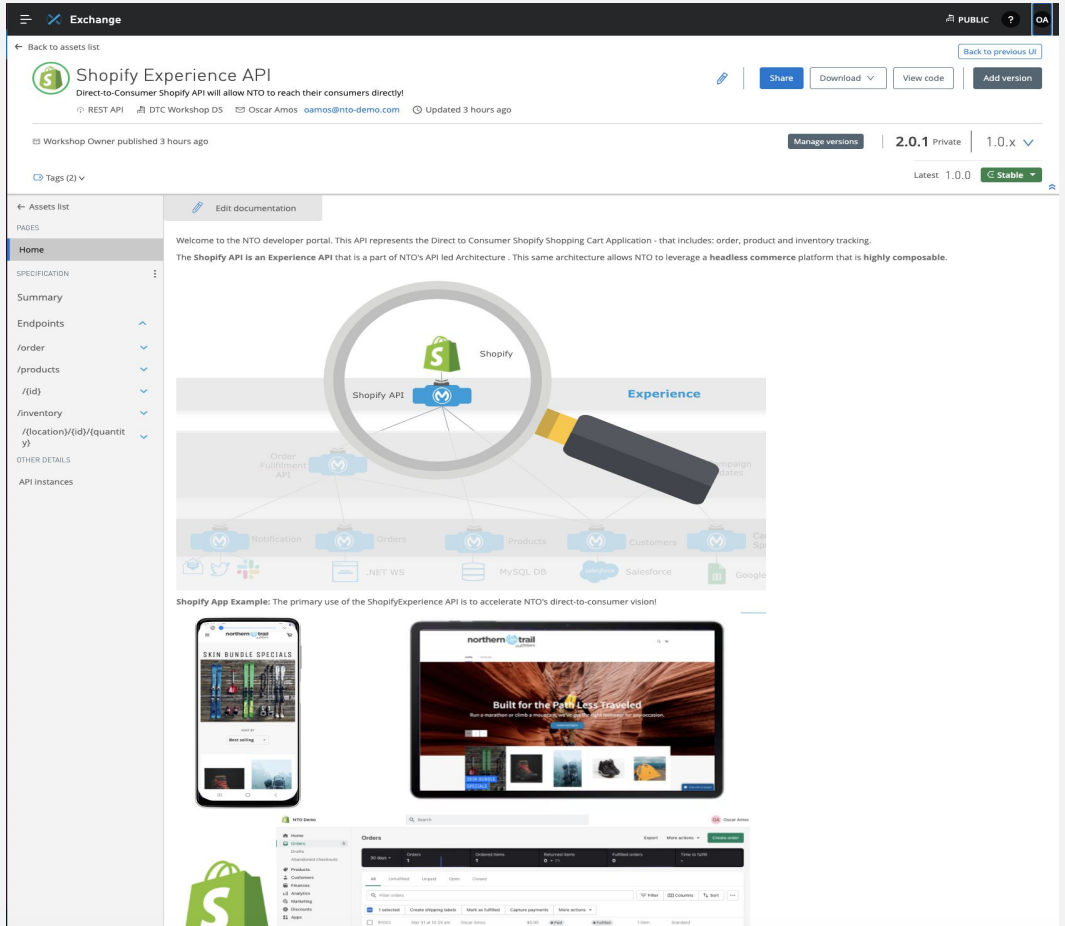
You should now see the page dedicated to the Shopify API definition:



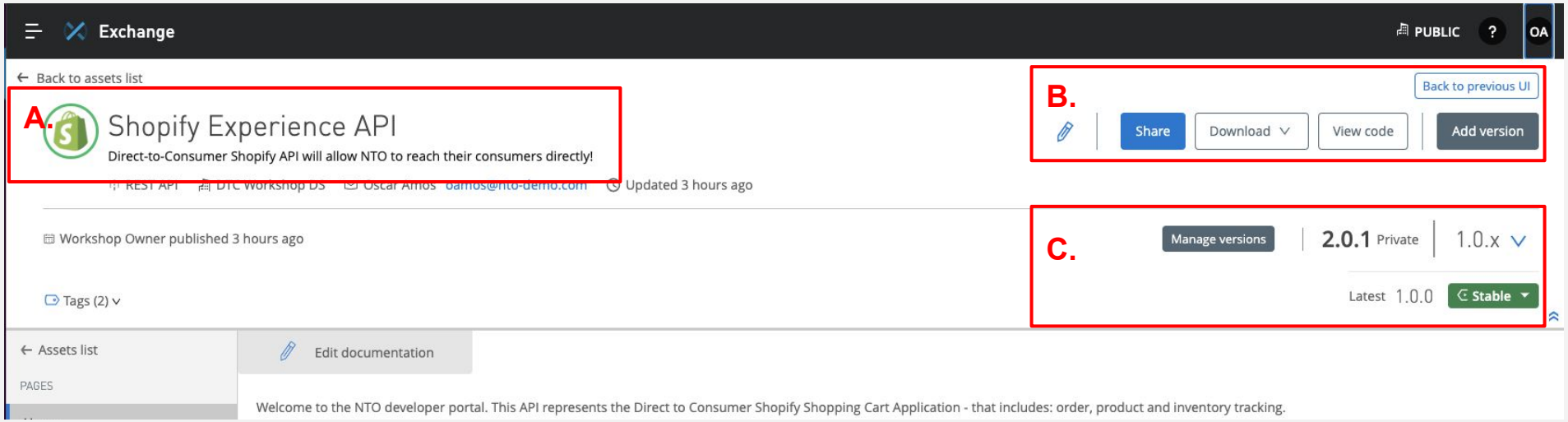
Module 1: The Modern API - Design: Lab 1 (11/14)

Exchange provides several new features to support and encourage discovery and collaboration.

First note the content in the portal to help you learn how to use the API. We will learn how to create this content latter. But for now, let's do a quick survey of the key features of the Exchange portal:



Module 1: The Modern API - Design: Lab 1 (12/14)



A. **Information Panel** - This provides general information about the API.

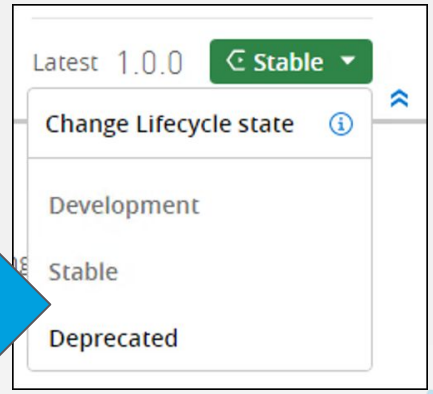
B. **Options Panel** - The options panel has three options.

- **Edit** (pencil icon): This option allows to edit the API portal.
- **Share**: This option allows you to share the API with users, users that have a role, or publish to the Public portal.
- **Download**: This option is to download the RAML Spec or Mule Plugin to use it as a Connector.
- **View Code**: Opens Design Center.
- **Add Version**: Lets you create a new version.

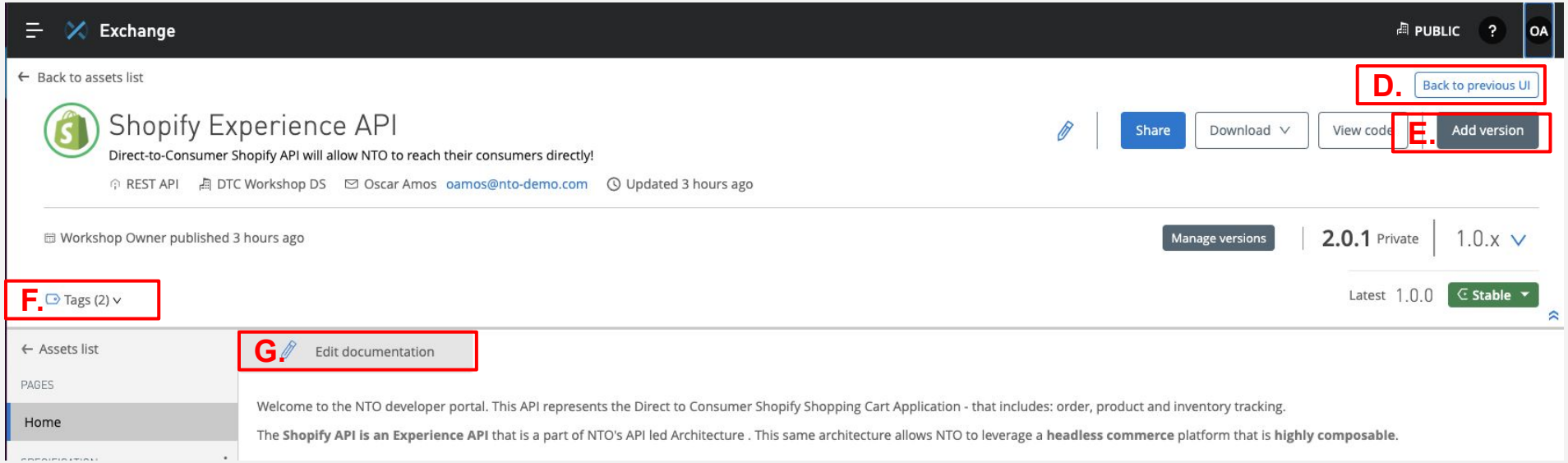
C. **Versions** It shows the different versions of the API.

Much of this information is automatically generated during the design and development of the API.

- **Change Lifecycle State**: Lets you change the state of the version



Module 1: The Modern API - Design: Lab 1 (13/14)



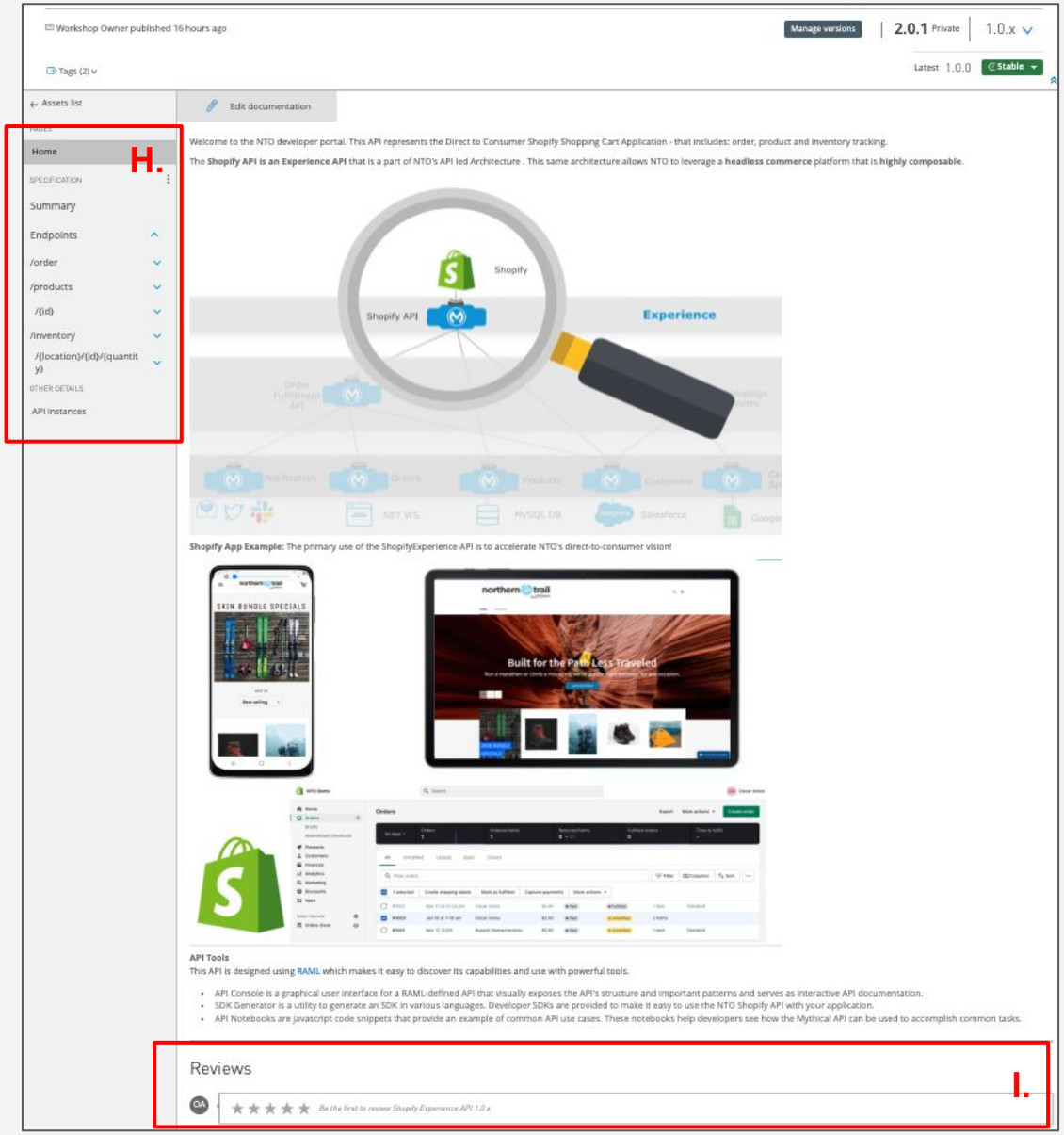
D. **Back to previous UI** allows you to toggle between our classic or newer Portal UI (defaulted at newer UI)

E. **Add Version** is where you can easily upload a newer version of the Shopify API specification via RAML or OAS file upload

F. **Add Tags** - *Tags* make it easier to search and discover API's. Rather than relying on the title or description of the API, tags allow for you to create categories of API's related to each other by *tags*.

G. **Edit Documentation** Lets you edit the documentation for your API.

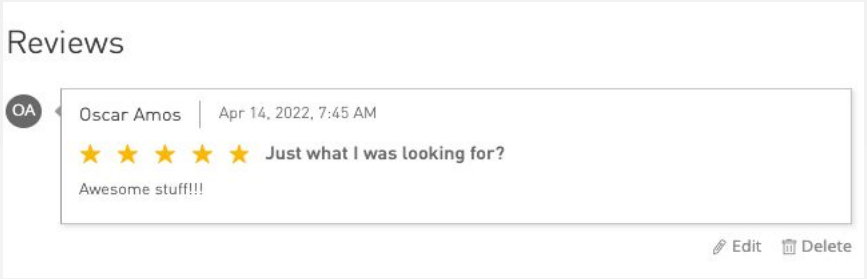
Module 1: The Modern API - Design: Lab 1 (14/14)



H. **API Summary** - This is the API specification. API resources can be seen here. You can click on each resource and view the documentation.

- You can expand the resource endpoints (e.g. orders) to view the operations that can be performed on that endpoint (e.g. GET, POST, DELETE):

I. **Ratings and Reviews** - New to Exchange, API users can review and comment on an API. This enables collaboration with the API owner and the ability to share experiences using the API that might help other API consumers.



Proceed to Lab 2

MuleSoft

Module 1: Lab 2 Design the Shopify API

Source: DTC e-commerce: How consumer brands can get it right

Module 1: The Modern API - Design: Lab 2 (1/20)

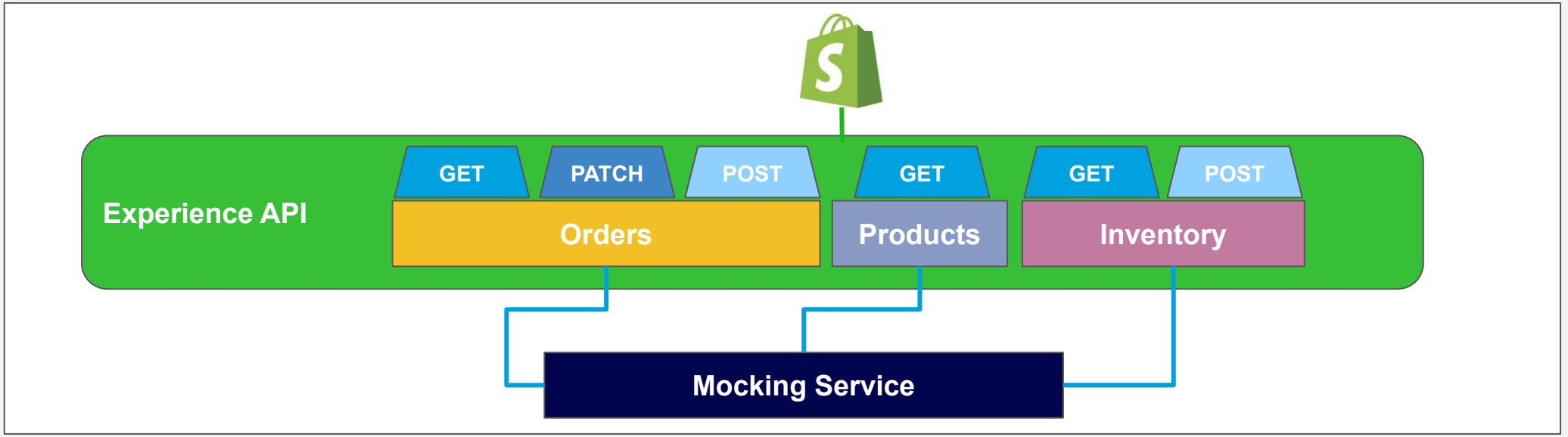
Overview

In the first lab of this module we learned how to discover the Shopify Experience API published in Anypoint Exchange. But of course, somebody had to design this API in the first place. For the purposes of this exercise, let's ignore the fact that the Shopify Experience API already exists. Instead, let's take on the role of the API designer and explore how to design the experience API using a **Design First** approach with MuleSoft's API-led Connectivity methodology. The goal of our *Design First* approach is to be able to design APIs that are easy to use for their intended target audience.

Anypoint Platform provides first-class tooling to support the needs of the API Designer. Anypoint Design Center provides a robust RAML editor for designing the specifications of the API and standing up a "mock" service to accelerate development efforts. It is also tightly integrated with Anypoint Exchange to simplify the process of discovering and referencing other API design-oriented assets like common data types, traits, schemas, and data examples.

For the purposes of this workshop we will design new API's using REST. RESTful API's are very popular right now and support a variety of patterns that help solve many reliability, scalability and integration challenges you might face. We will use an API-first design approach using the RESTful API Modeling Language (RAML) standard. We will follow REST best practices to promote adoption to your consumers. This lab will also introduce the main concepts of resource oriented design and will show how RAML is used to bring APIs to life. For more information on RAML, please see: www.raml.org.

Module 1: The Modern API - Design: Lab 2 (2/20)



In this lab, we will start designing a new REST API and test it before implementing it. We will use Anypoint Platform's **Design Center** to design the API.

We will then use **Anypoint Exchange** to search and find data definitions and examples published across the organization that enable us to standardize on common data elements.

Lastly, we will utilize the **Mocking Service** to unlock resource dependencies of your API design. This significantly cuts down the time spent building the omni-channel application by turning the RAML design over to the omni-channel developers immediately and adopting a rapid prototyping style. Developers can utilize the mocked up API capabilities to produce a "working" application that calls the Mocking Service.

Module 1: The Modern API - Design: Lab 2 (3/20)

Understanding Resource Oriented Design (ROD)

Designing REST interfaces is a bit of a paradigm change compared to SOAP services. SOAP services design was thinking in actions and methods whereas ROD wants you to think along the lines of your business entities that should be modeled as resources. And resources might be everything from a document, a video, your favorite device or an order process.

The quintessence on ROD is:

1. Use standard methods (e.g. HTTP GET, PUT, POST, DELETE, HEAD) and notice their specifications - e.g. GET, HEAD, DELETE and PUT are idempotent methods
2. Use standard response codes for success and error; use existing patterns e.g. not every successful call should return a 200 OK - often a 201 Created is appropriate after a POST /products or a 202 Accepted when a long running task was started by the server
3. Use standard header parameters
4. Design and craft resources in **your** business language - here is where RAML comes into play
5. **Interlink** as many of your resources as you can; e.g. when you return a collection of products, have every product in the result carry its own link so the client can easily access it

Module 1: The Modern API - Design: Lab 2 (4/20)

MuleSoft

Collection Resources

Typically resources are designed as either collection or instance resources. Collection resources hold a collection of resources that can be queried, ordered, extended and most importantly linked.

Let's take a typical example of a product resource. Most likely there are several of them and you want to be able to create new products also.

By designing a **/order** resource a list of products is available. Typically the methods

- GET (get the list)
- PATCH (can both create new order records or modify existing order record to specified target
- POST (will create order(s) at its intended target)

Module 1: The Modern API - Design: Lab 2 (5/20)

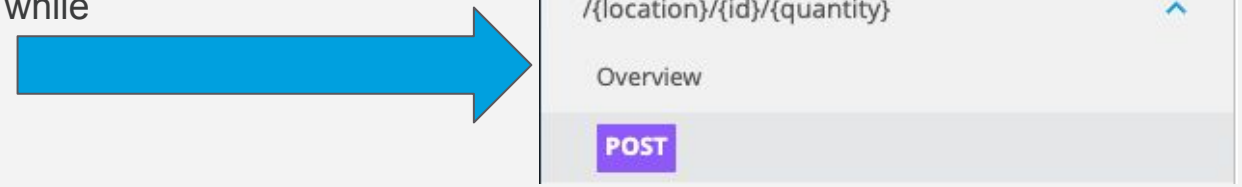
MuleSoft

Instance Resources

Instance resources contain the details of a specific resource, i.e. the instantiation of a resource. For our inventory example that might include a product id and id of the store. Highly valuable are then links to other parameters (e.g. quantity) that might be part of that inventory management process.

Instance resources must be uniquely identify-able and typically look like this: **/inventory/{location_id}/{product_id}/{quantity_value}**. Typically the methods

- GET (retrieve an inventory list of products)
- POST (update inventory data by location and specific product while specifying quantity to update)



are available on the inventory resource.

Now we understand resources let's explore how we can work with the resources.

Module 1: The Modern API - Design: Lab 2 (6/20)

Step 1: Review the E-Commerce Application Requirements

You have a new requirement for a new Shopify application. The NTO business team has the following requirements we are asked to fulfill. It is written in the language of the e-commerce developer and we will derive the RAML specification from that.

Functionality:

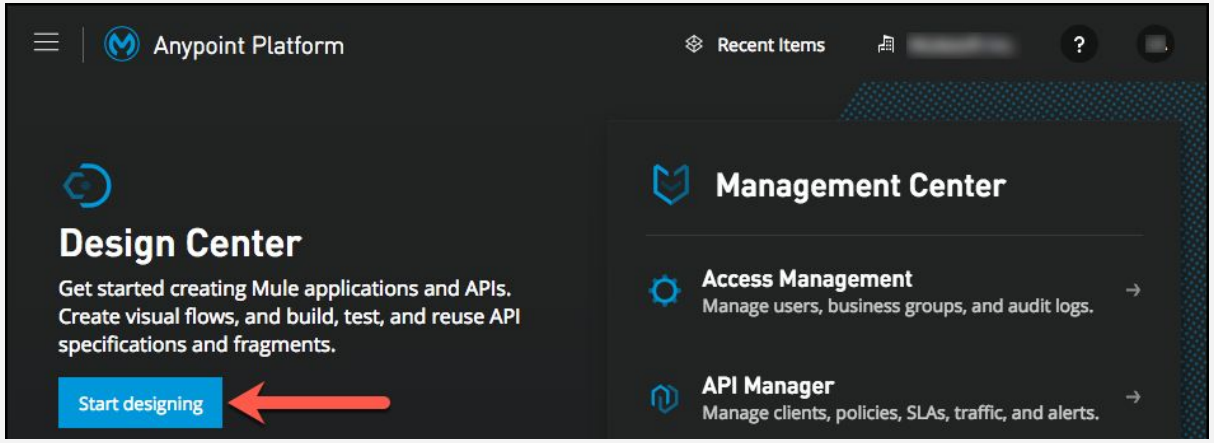
- Products:
  - List all the products available
    - Name
    - Description
    - Product Variants by SKU
    - Category/Brand
  - Get information about a specific product by identifier (ID)
- Orders:
  - List all consumer orders made in Shopify
    - Get information of an Order by its Order ID
  - Create new consumer order from other systems and post to Shopify for status updates via Shopify:
  - Bulk update and add new orders into Shopify
- Inventory:
  - List all product inventory across all NTO store locations
  - Update a specific product inventory level at certain store location

Module 1: The Modern API - Design: Lab 2 (7/20)

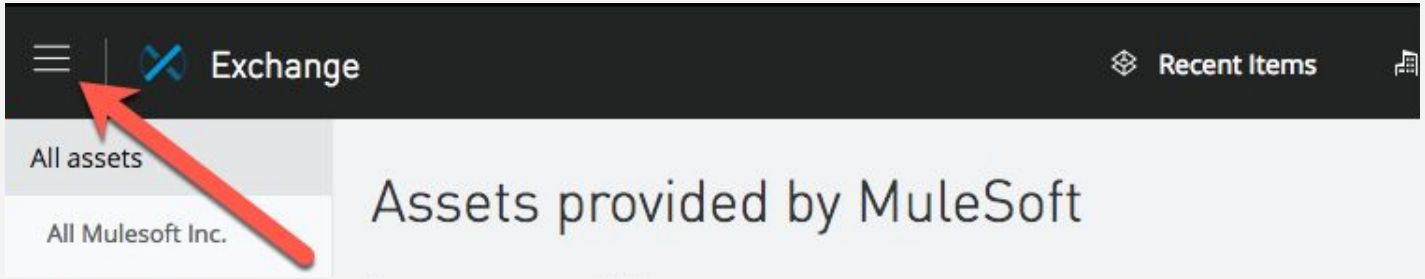
Step 2: Create the Shopify Experience API

In this step, you will create an API and design it using the Anypoint Design Center.

1. You can access Anypoint Design Center from the home page of Anypoint Platform:

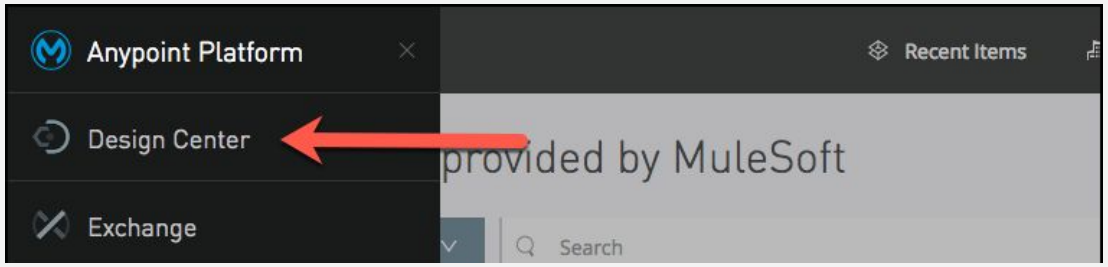


- Alternatively you can use the "hamburger menu" from any page within Anypoint Platform portal to navigate to Design Center:

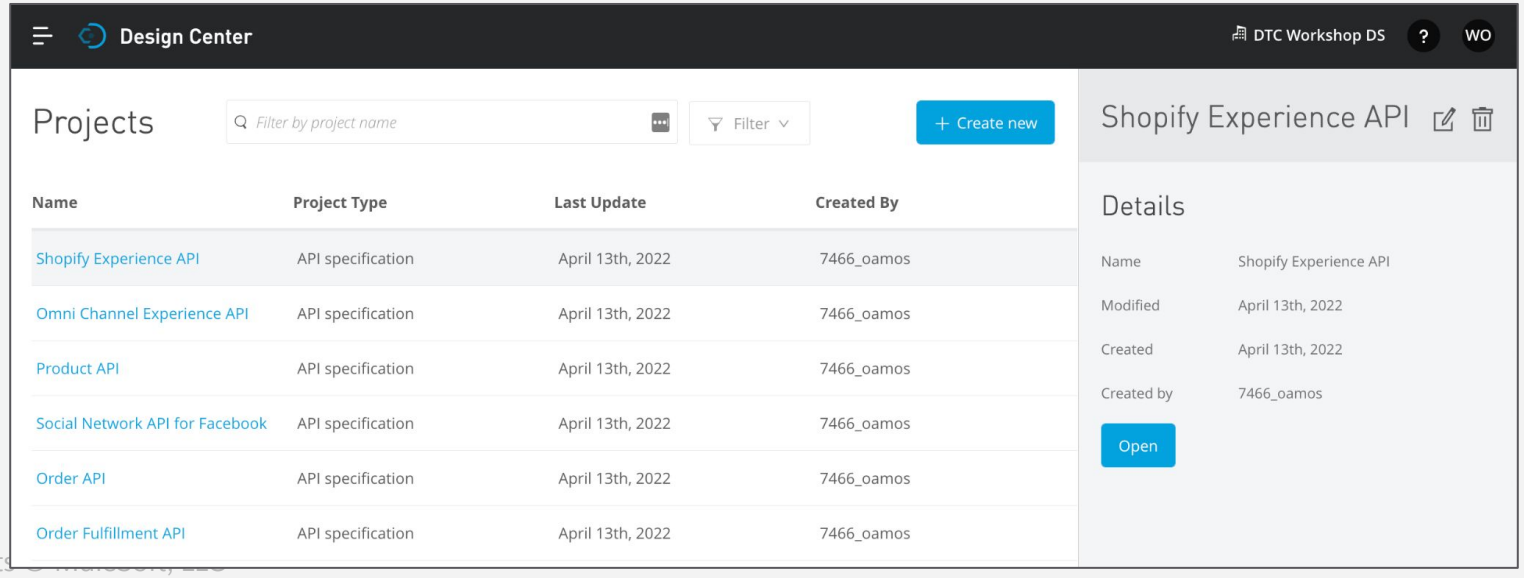


Module 1: The Modern API - Design: Lab 2 (8/20)

2. **Anypoint Design Center** is used to design your API's. When you arrive at the Design Center landing page you will see a list of API's that have already been designed by your organization.

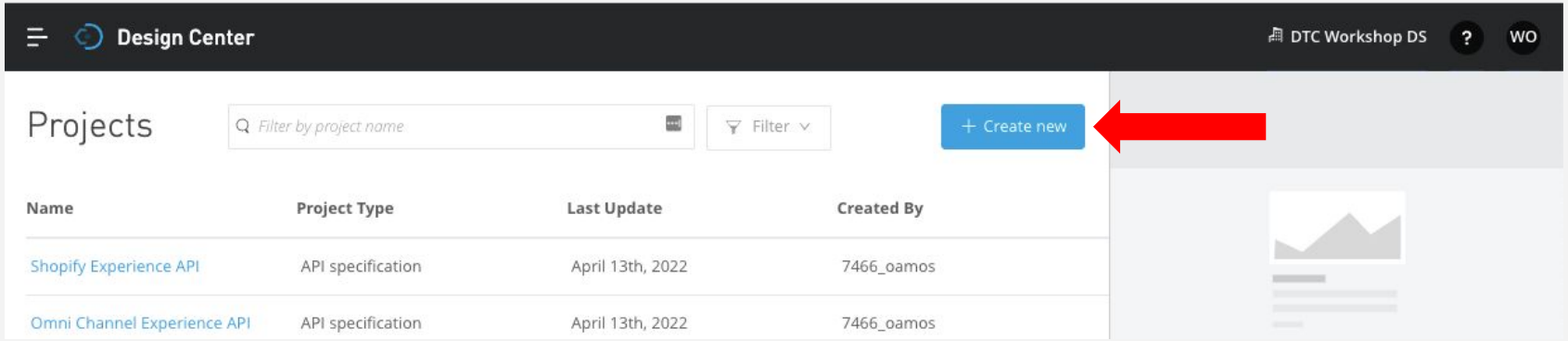


- **Anypoint Design Center** is used to design your API's. When you arrive at the Design Center landing page you will see a list of API's that have already been designed by your organization.

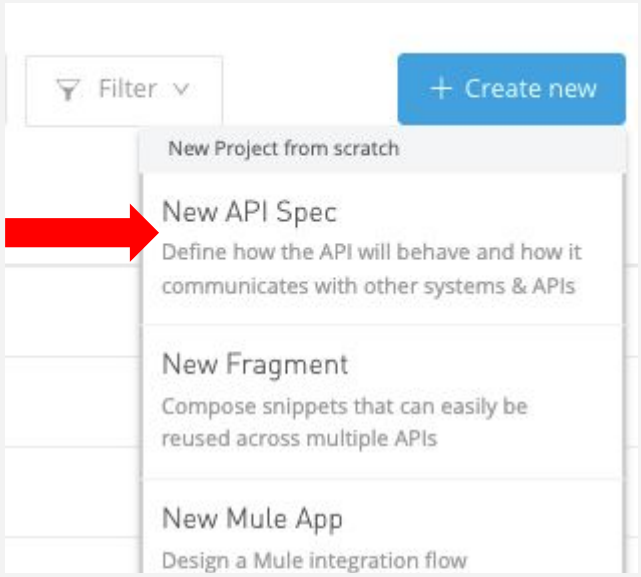


Module 1: The Modern API - Design: Lab 2 (9/20)

3. To create a new API, Click on the **Create new** button

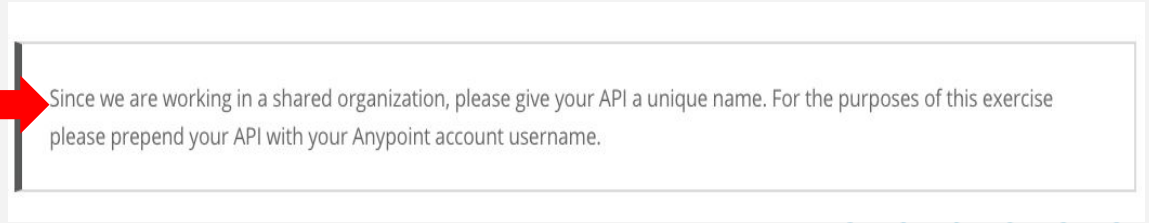


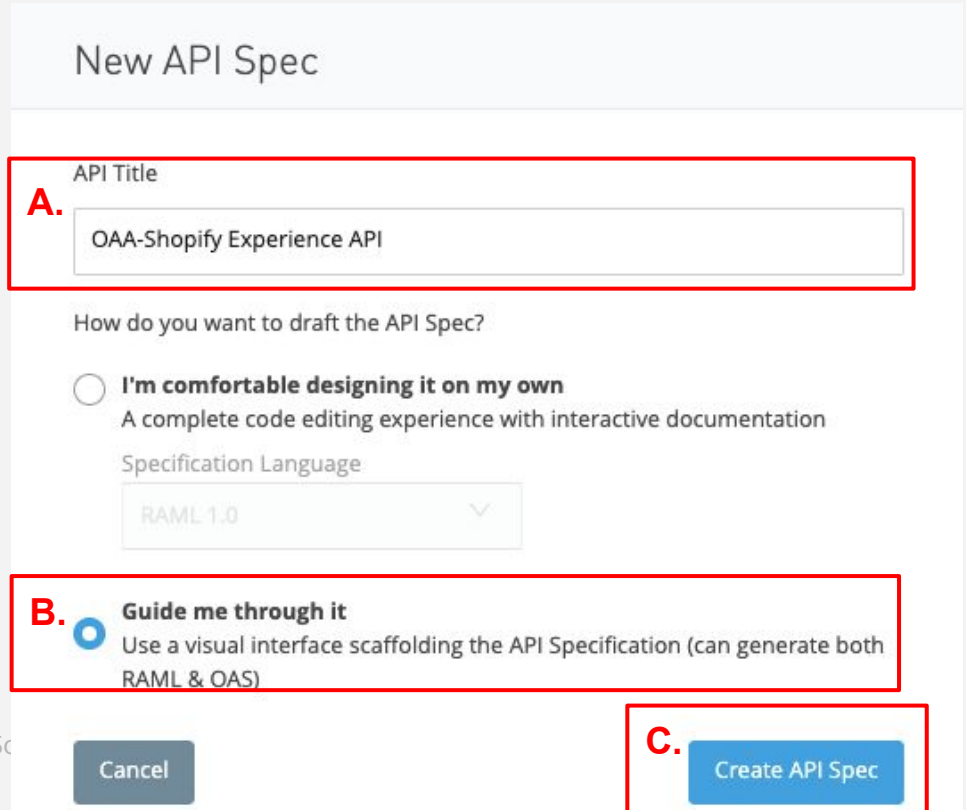
4. Select “New API Spec”:



Module 1: The Modern API - Design: Lab 2 (10/20)

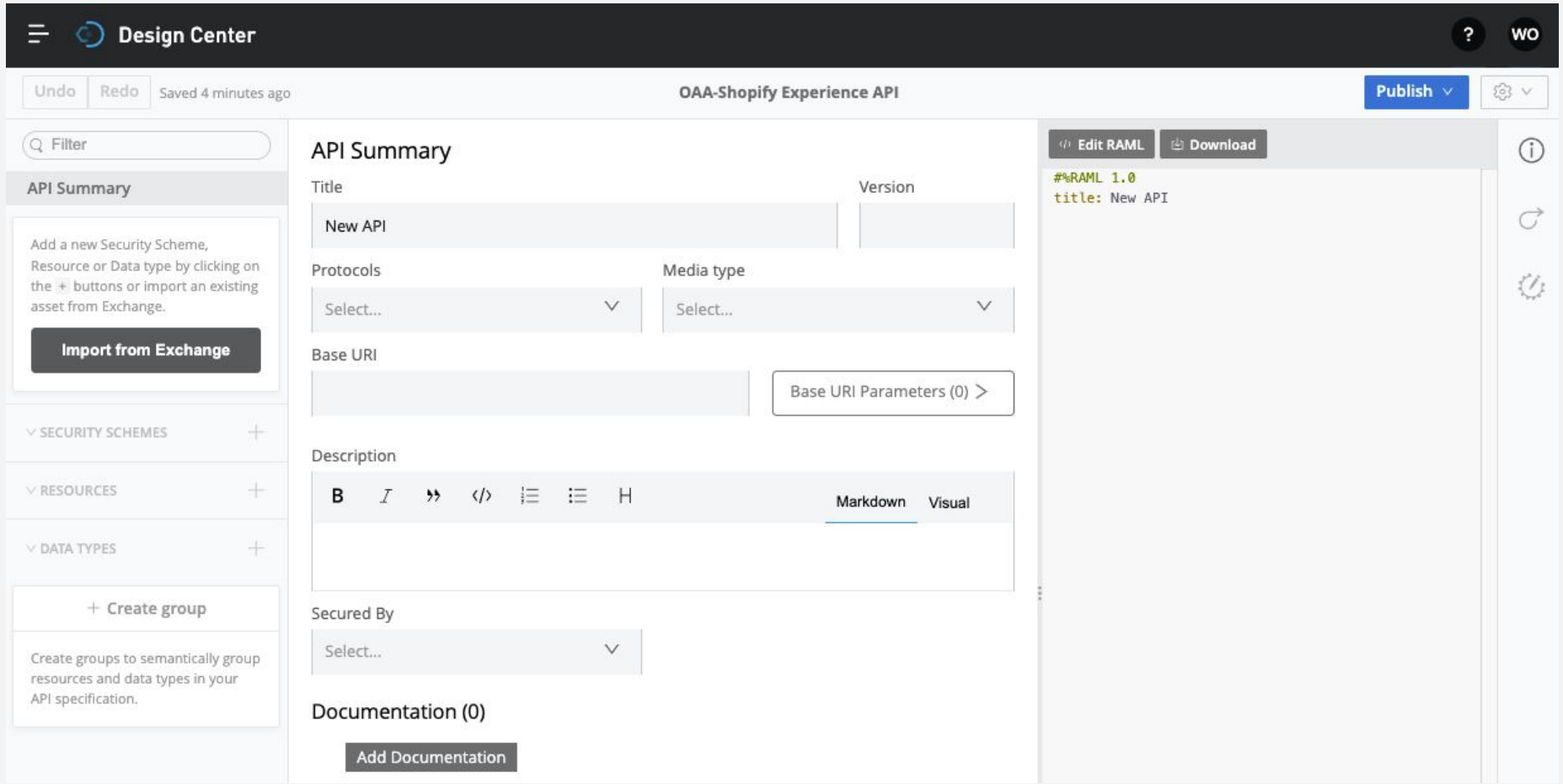
5. Set the following values in the **Add API** form:
   A. API name: **<your name or initials> - Shopify Experience API**.
   B. How do you want to draft the API Spec: Select **Guide me through it** radio button.
   C. Select **Create API Spec** button.





Module 1: The Modern API - Design: Lab 2 (11/20)

6. After clicking **Create API Spec**. This will create the basic structure of your RAML-based API specification. Note that for full RAML support, you should start with the first option (API Designer), which is the standard RAML-based API Designer.



Module 1: The Modern API - Design: Lab 2 (12/20)

Step 3: Create the Orders List Resource

In order to provide order handling functionality for the e-commerce application, we will design a list resource in our API using RAML.

Anypoint Design Center will provide a visual API editor to start with a step-by- step tutorial to guide you through designing your API visually.

1. Let's add some general details about the API.

   A. Name the API by setting the title to: "**<your name or initials>- Shopify Experience API**".

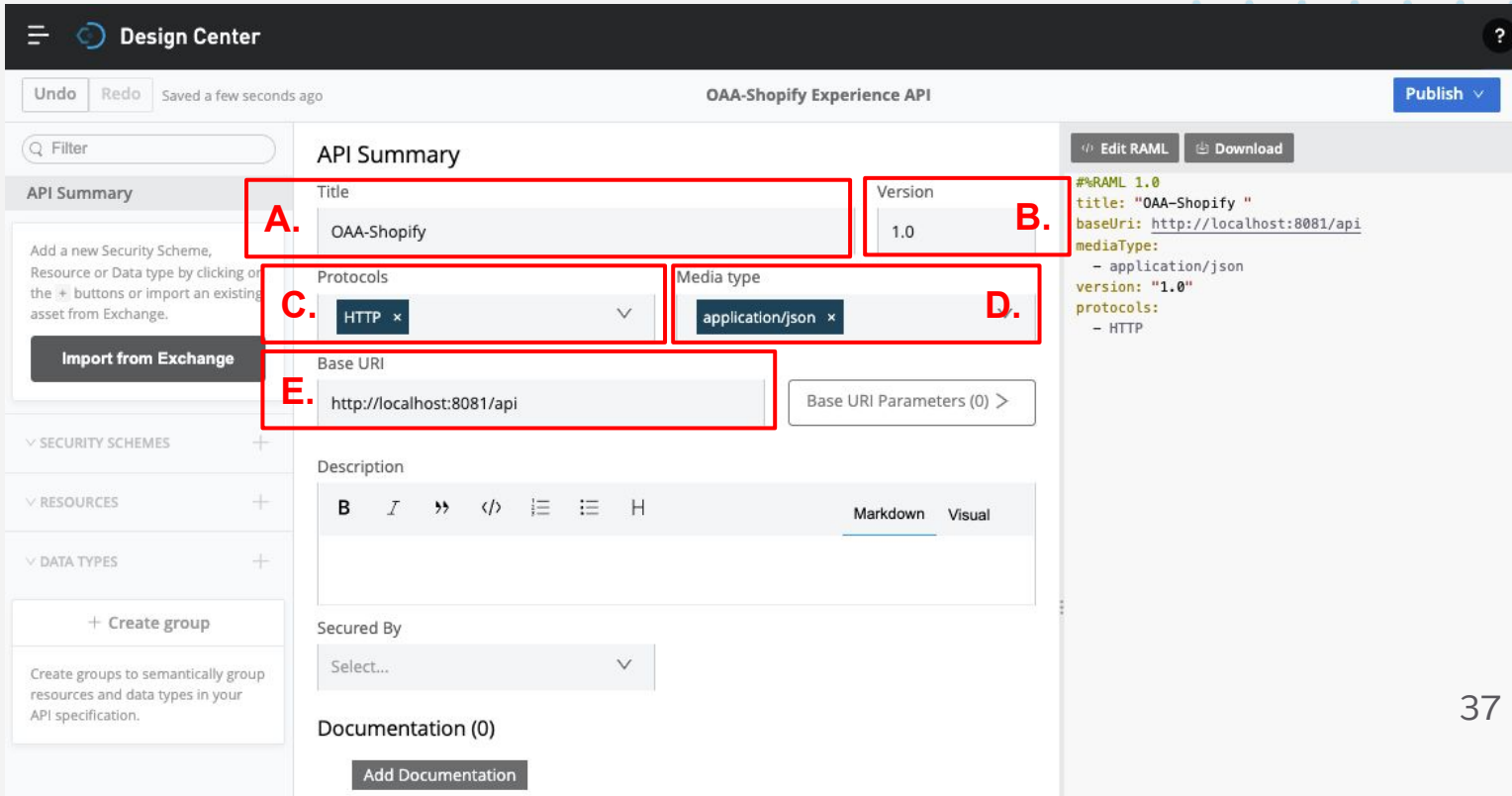
   B. Since this is our first design let's set API Version to "**1.0**"

   C. Select protocol as "**HTTP**"

   D. Set the media type to "**application/json**".

   E. Most API specifications will designate an API endpoint Base URI. This tells the API consumer where they will access the API. Set the URI to "**http://localhost:8081/api**"

You should now see the something like following in your API design:

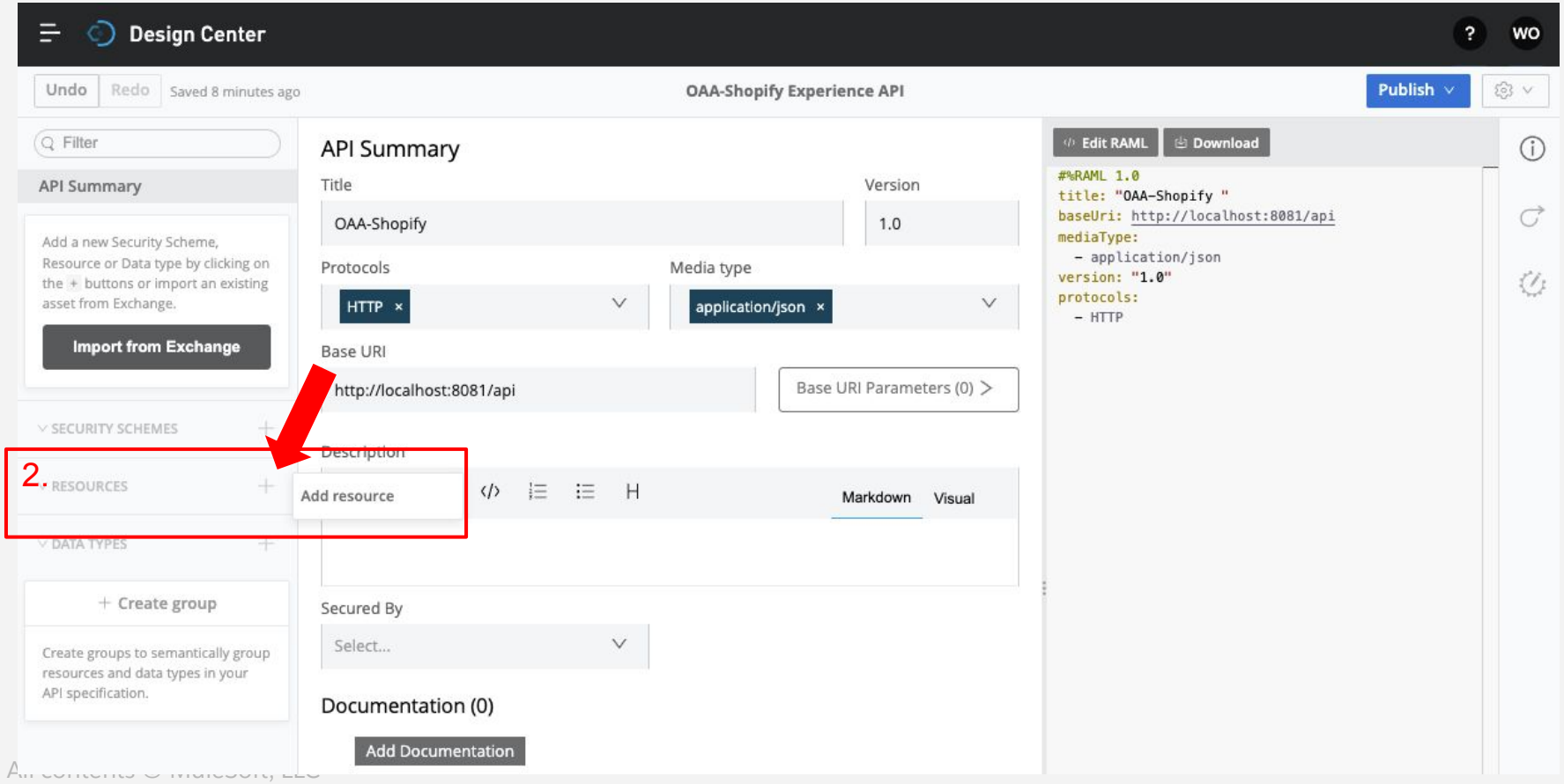


Module 1: The Modern API - Design: Lab 2 (13/20)

MuleSoft

Step 3: Create the Orders List Resource

2. Now you need to add a resource. A resource is the definition of an entity that your API will manage. Click the **+** to the right of **Resources**, then choose **Add resource**.



Module 1: The Modern API - Design: Lab 2 (14/20)

Step 3: Create the Orders List Resource

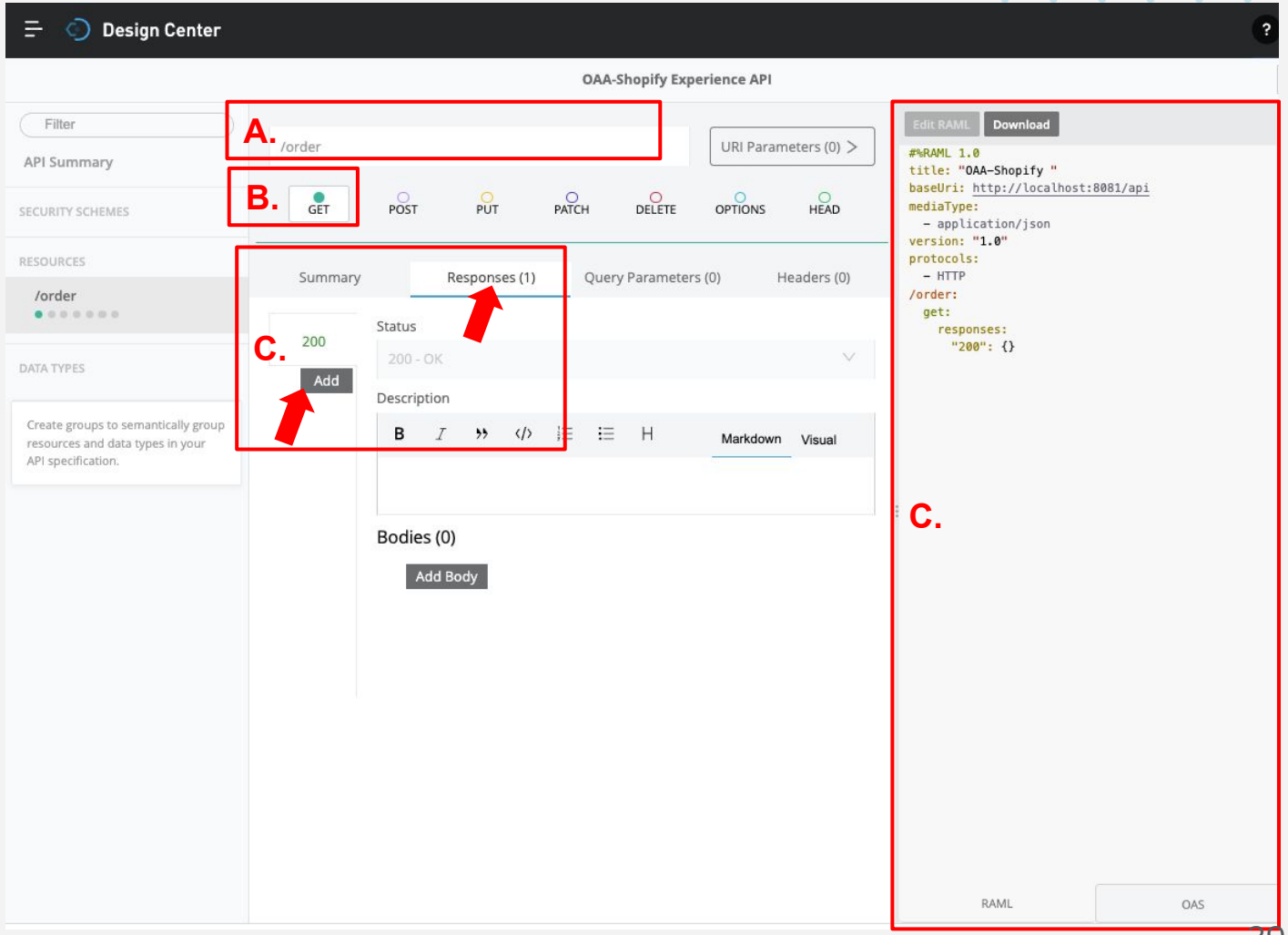
3. To create the order resource for the Shopify API we will need to do the follow:

   A. Rename the resource from **`/new-resource`** to **`/order`.**

   B. Select the **`GET`** option and under summary → description, give it a description

   C. Scroll to and select **`Responses`** tab below. It will have a default **200** Status (which signals successfully operation) preselected and hit **Add**

   D. Notice the generated RAML on the right.



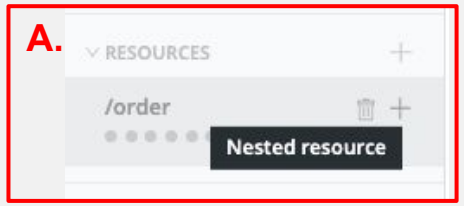
Module 1: The Modern API - Design: Lab 2 (15/20)

Step 4: Create the Order Instance Resource

In the previous step we created an Order collection resource and GET method to grab all Shopify Orders. But in most cases for NTO, information on a single order will be needed. We need to create a new resource underneath the Order instance and use the GET method.

1. So we will do the following:

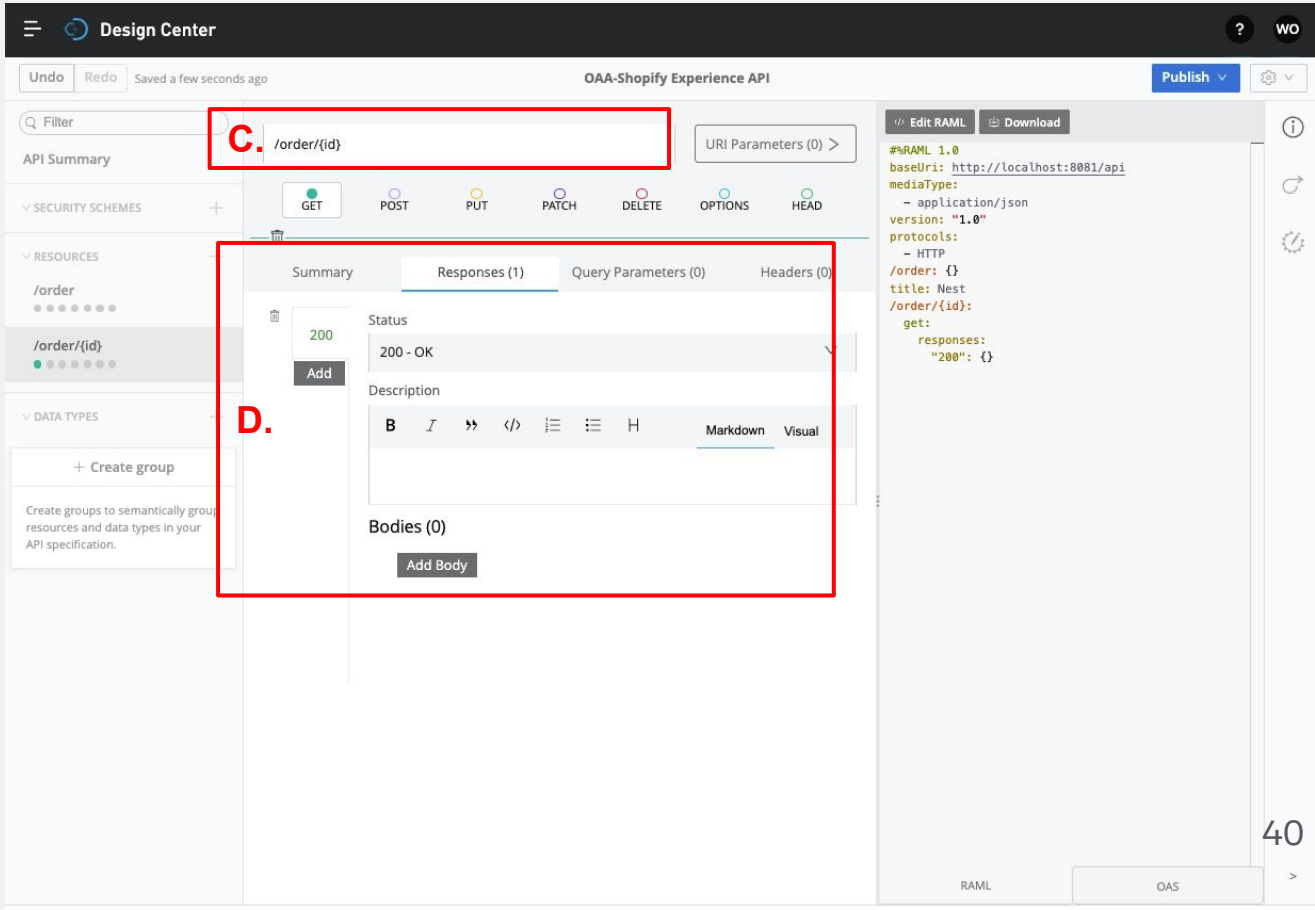
A. By the **order** resource hover your mouse on the resource and you should see a trash can and "**+**" icon. Under the "**+**" icon, **Nested resource** call out is visible to indicated we can nest child resource under **order**



B. Click the "+" icon and you will see that another resource, **/order/new-resource** has been added under **order**.

C. Change "**/order/new-resource"** to "**/order/{id}"** in the title pane

D. Scroll to and select `Responses` tab below. It will have a default **200** Status (which signals successfully operation) preselected and hit **Add**

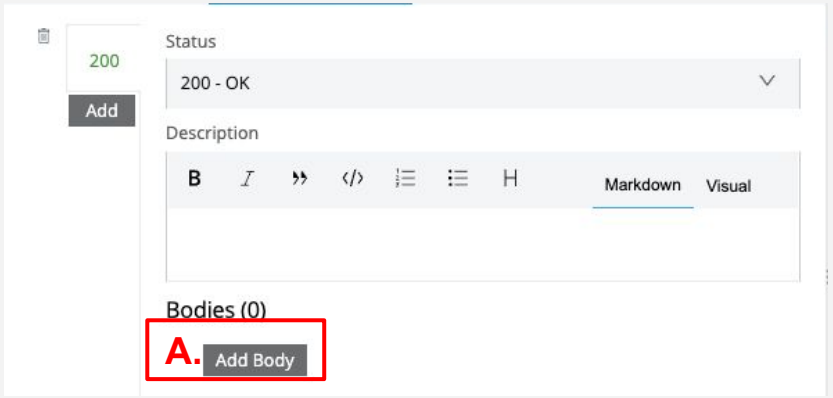


Module 1: The Modern API - Design: Lab 2 (16/20)

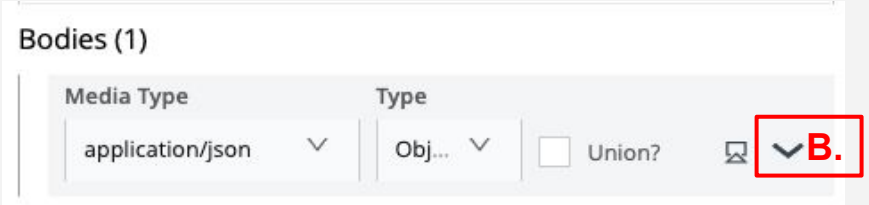
Step 4: Create the Order Instance Resource

2. Now let's add a specific body to the 200 response for **order/{id}** instance.

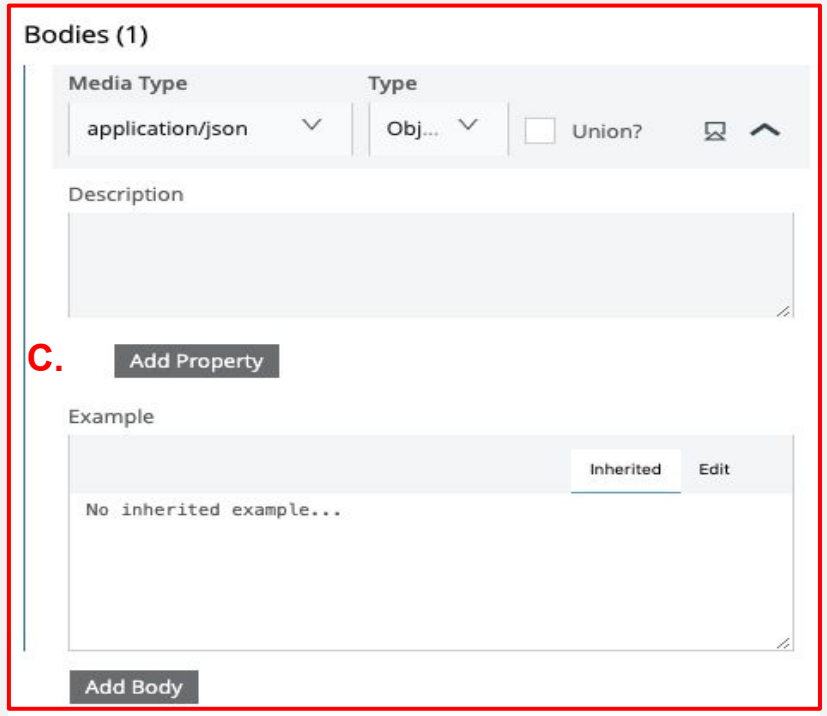
A. Select **Add Body**



B. Keep **application/json** as the Media Type and hover to the arrow icon at the right to expand options.



C. Once expanded it should look like this:

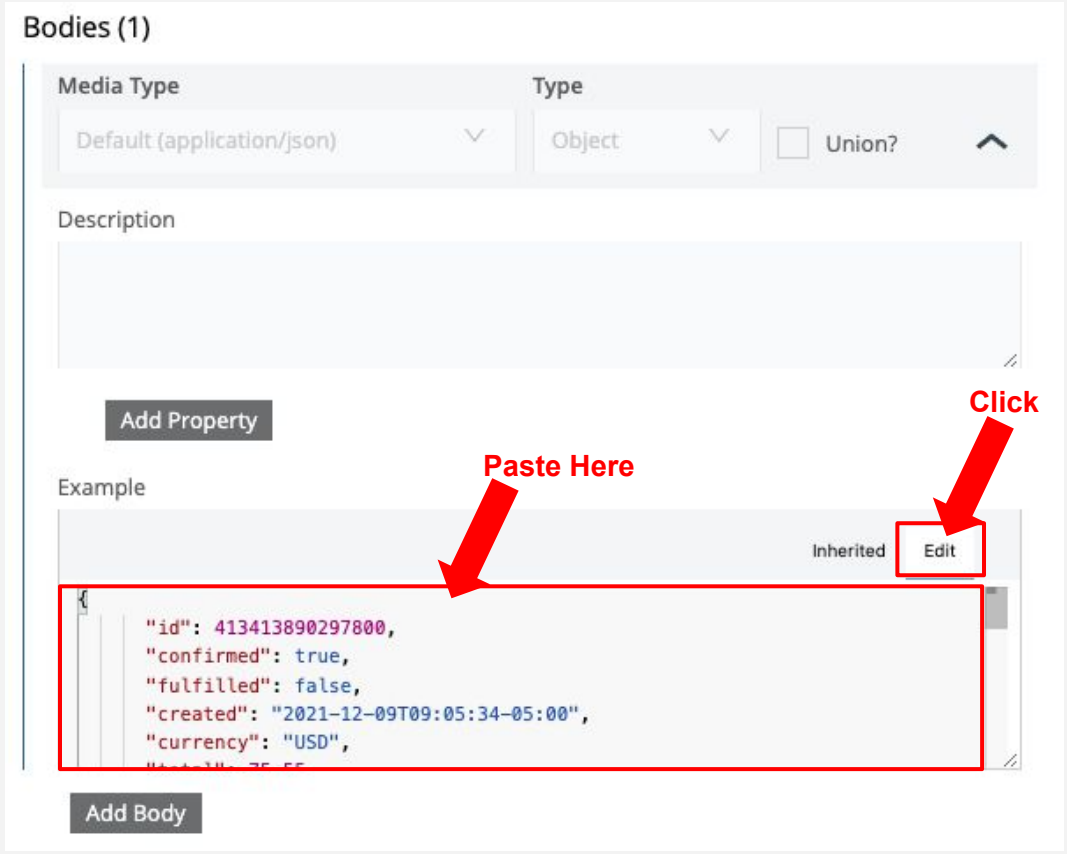


Module 1: The Modern API - Design: Lab 2 (17/20)

Step 4: Create the Order Instance Resource

3. Click **Edit**

4. Now let's copy and paste the following code into the body:




```
{
    "id": 413413890297800,
    "confirmed": true,
    "fulfilled": false,
    "created": "2021-12-09T09:05:34-05:00",
    "currency": "USD",
    "total": 75.55,
    "customer": {
      "id": 1616616,
      "email": "jane@gmail.com",
      "address1": "567 Main St",
      "city": "Stafford",
      "state": "VA",
      "country": "USA",
      "zip": "22554",
      "phone": "555-710-9325",
    "line_items": [
      {
        "id": 10417352802515,
        "sku": "841483107332",
        "name": "Waist Cincher - Soft Nude / L",
        "quantity": 1,
        "gift_card": false,
        "price": 75.55,
        "tax_code": "PC040100"
      }
    ]
  }
}
```

If you have problems copying the code you can find example here

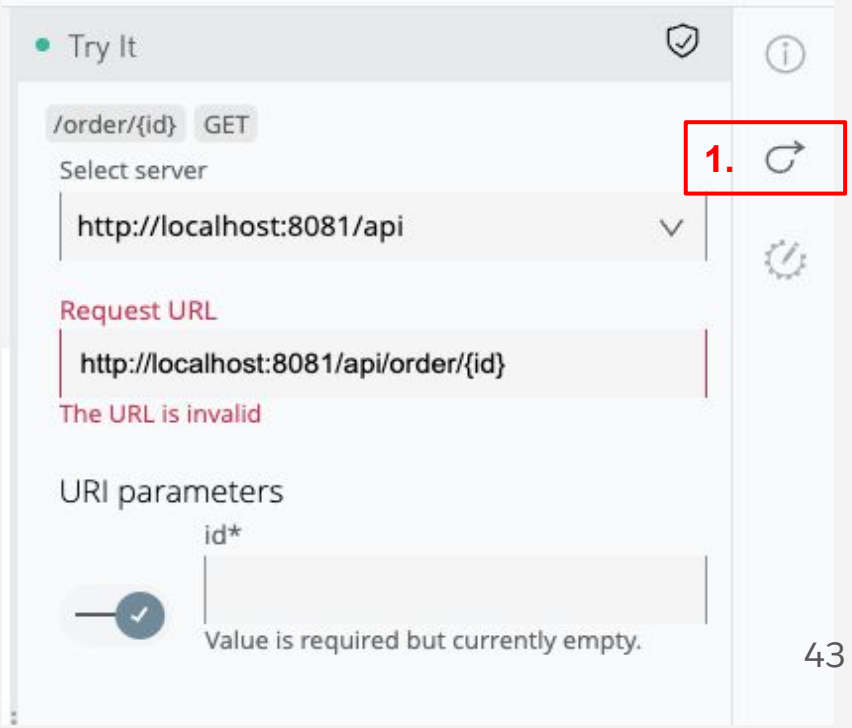
Module 1: The Modern API - Design: Lab 2 (18/20)

Step 5: Use Mocking Service

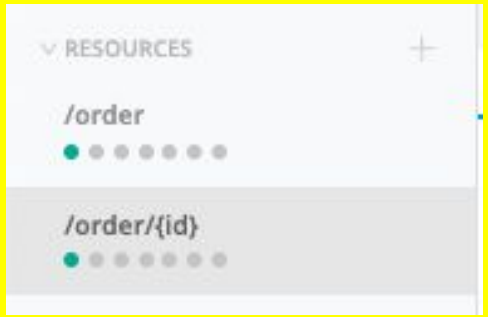
A very useful feature during the design phase is having the ability for your consumers to interact with your API without having to code anything. The mocking service is a feature of the Anypoint Platform. You can simulate calls to the API in API Designer before publishing the API specification to Exchange or in Exchange after publishing the API specification. The mocking service will read the RAML specification of your API, create the API/service and return example data responses. This service allows users to interact with the API as if it was built and deployed. This feature allows you to rapidly iterate the API design with its consumer to finalize the contract.

First we are going to enable the mocking service for internal use.

1. Click the **Circular Arrow** icon in the right-hand toolbar it should expand the pane like so:



If you see an empty pane check to make sure the order/{id} resource is highlighted in the left pane like so:



Module 1: The Modern API - Design: Lab 2 (19/20)

Step 5: Use Mocking Service

2. From **Select Server** dropdown change the selection from **http://localhost:8081/api** to **Mocking Service**

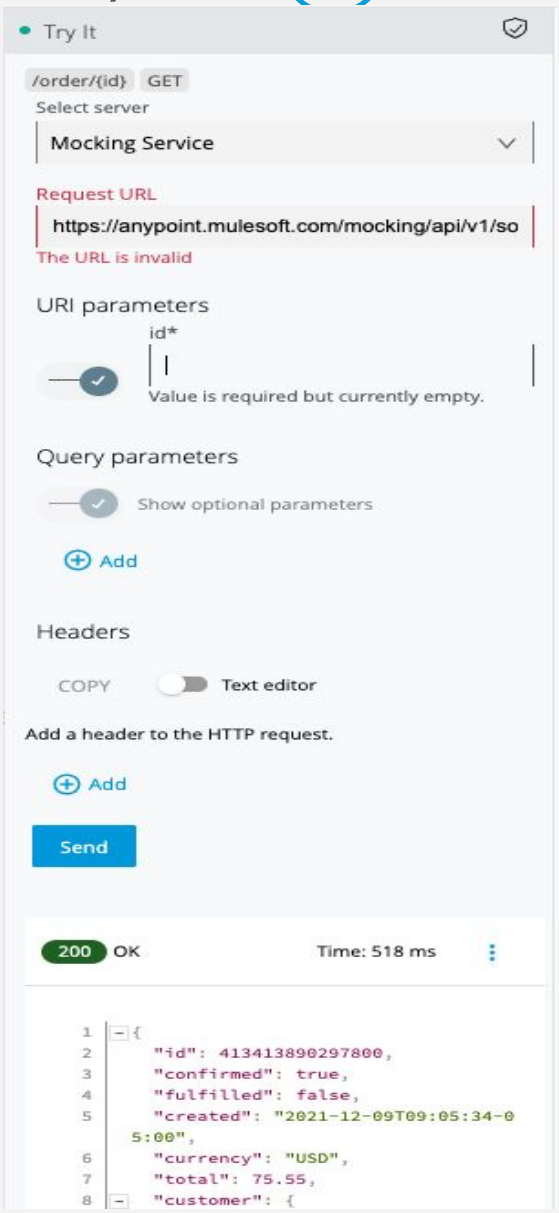
3. Under **URI parameters (id*)** you can add any number you would like to.

4. Then click the **Send** Button.

5. You will notice at the bottom the sample response with the sample JSON code we copy and pasted in the order/{id} resource response body is showing. The mock response also shows a **200 OK** and **Time** output to simulate a the request and reply process for Shopify API.

Use the mocking service and check the examples is updated too.

The API design is ready. In the next lab we are going to publish the API so it can be discoverable by anybody in the organization.



Module 1: The Modern API - Design: Lab 2 (20/20)

MuleSoft

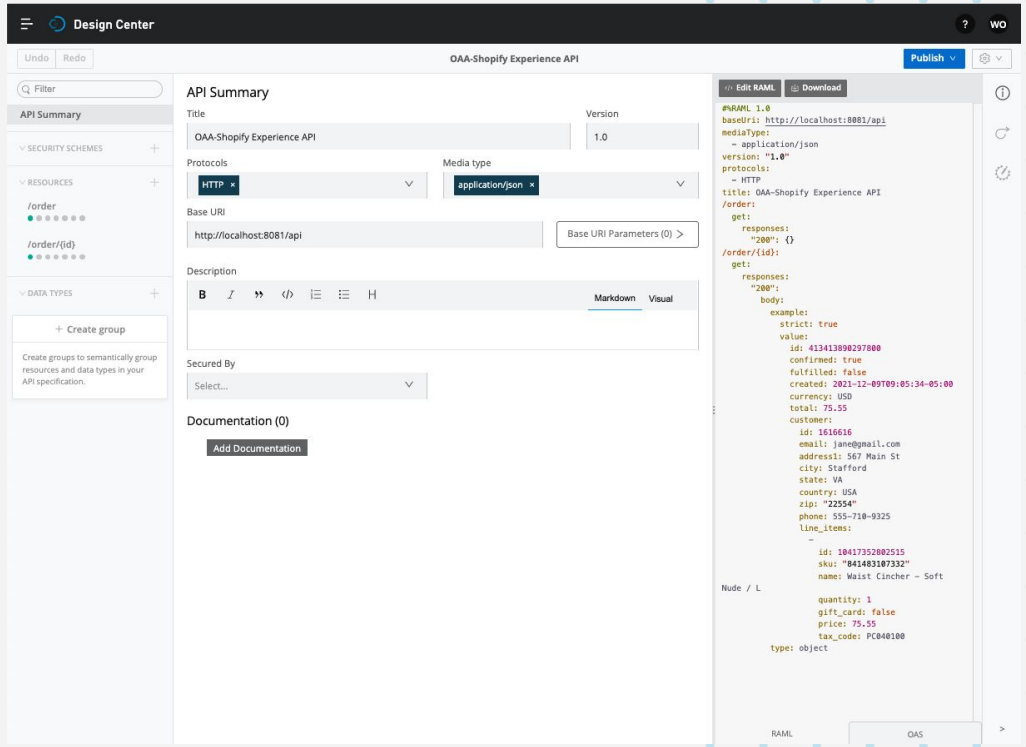
Summary

In this lab, you completed the following steps:

- Step 1: Review the Shopify Requirements
- Step 2: Create the Shopify Experience API
- Step 3: Create the Orders List Resource
- Step 4: Create the Order Instance Resource
- Step 5: Use the Mocking Service

We easily designed the framework for our Shopify API, providing the ability to get orders and check single order. We leveraged RAML for a **design first approach**.

We saw how the mock service can be utilized to provide application developers an API mock up they can build their applications on. This significantly **speeds up end to end development**.



To learn more about RAML follow this tutorial.

- See Designing Your API for more information on API design.
- See Design Mocking Service for more information on API design

Congratulations! You have completed Lab 2.

Proceed to Lab 3

MuleSoft

Module 1: Lab 3 Publish the Shopify API To Exchange

Source: DTC e-commerce: How consumer brands can get it right

Module 1: The Modern API - Design: Lab 3 (1/17)

Overview

In Lab 1 we examined how to use **Anypoint Exchange** to facilitate the API discovery process. In Lab 2 we learned how to use **Anypoint Design Center** to design the specifications for a modern RESTful API. In this lab we will learn how to publish our API to **Anypoint Exchange** so that API consumers and API developers within the organization are able to leverage the API through "self-service".

One of the biggest challenges encountered by early adopters of SOA was how to scale their development activities. SOA maintains many of the reuse principles that we have discussed in our "API-led" approach to software development. Unfortunately, most SOA efforts could not achieve their reusability objectives because of a few limiting factors including:

- Poorly documented services
- Lack of mature tools to facilitate service discovery
- Over-reliance on limited number of knowledgeable resources
- Burdensome processes for gaining access to services

As a result, many SOA-based initiatives were only able to partially achieve their reusability objectives because developers would adopt the path of least resistance and simply recreate services rather than reuse them. The **MuleSoft Anypoint Platform** addresses all of these issues by providing innovative tools like **Anypoint Exchange** that alleviate these challenges.

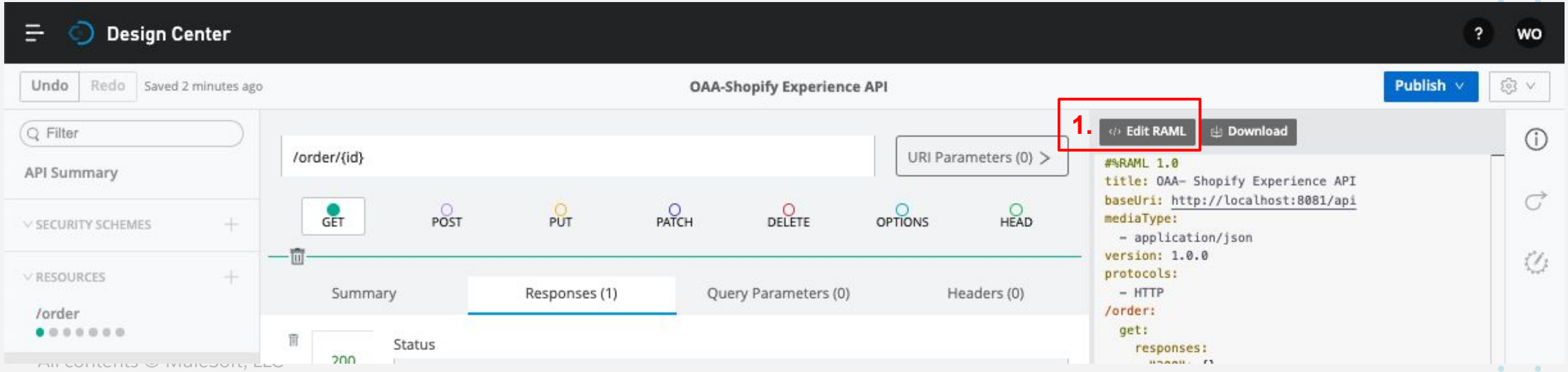
Let’s take a look at the process for publishing and documenting our reusable **Shopify Experience API**.

Module 1: The Modern API - Design: Lab 3 (2/17)

Step 1: Complete the Shopify Experience API Specification

Before we get started, make sure you are logged into the **Anypoint Platform** and viewing the **Anypoint Design Center** screen. You should have your **"<your name or initials> - Shopify Experience API"** open and visible. We have only defined a few resources and methods at this point and need to author the full API in order to complete the exercise. Rather than build it from scratch, we have the ability to import an API specification previously constructed for the workshop.

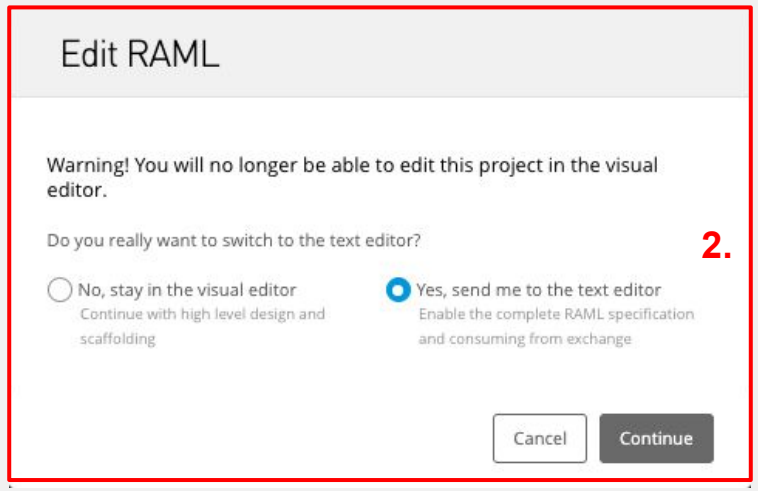
1. Go to and click **Edit RAML** located at the top right corner.



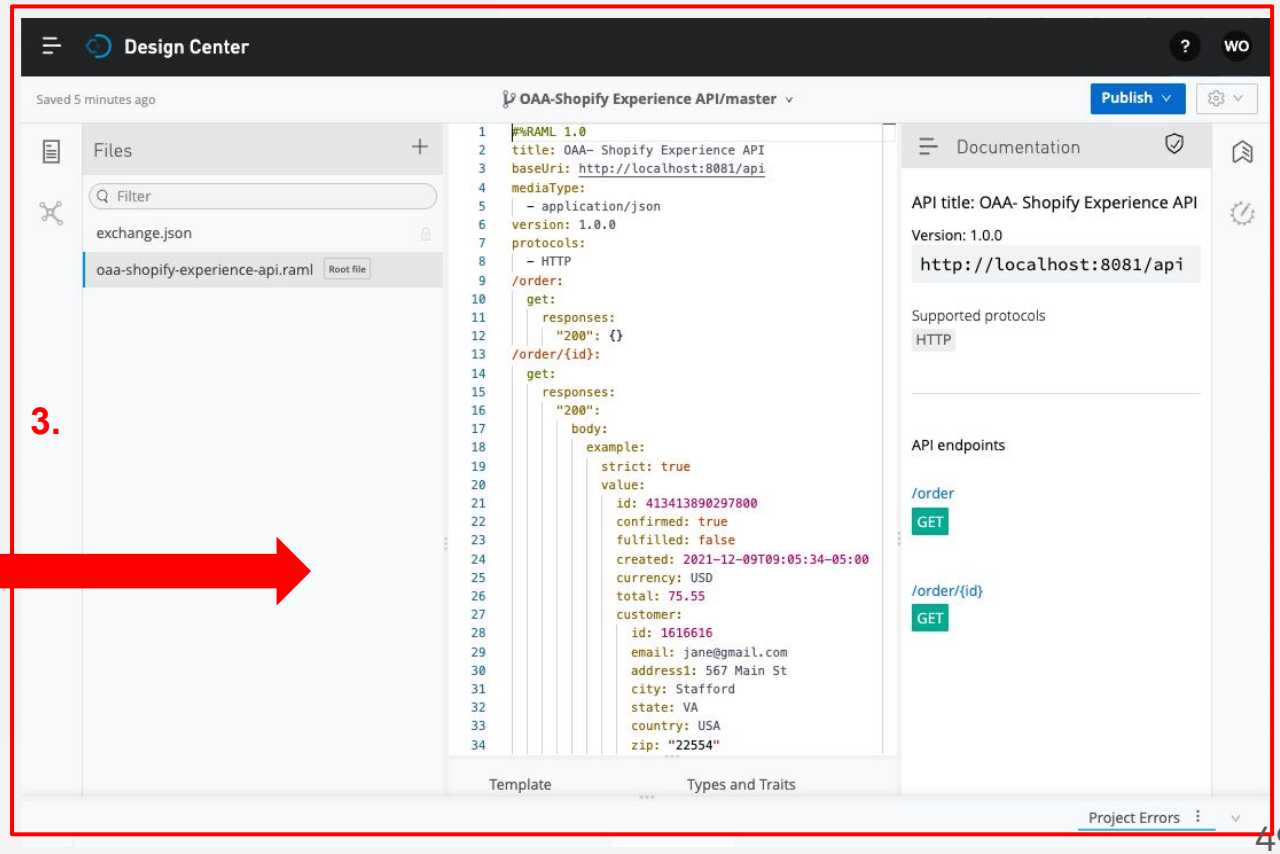
Module 1: The Modern API - Design: Lab 3 (3/17)

Step 1: Complete the Shopify Experience API Specification

2. An **Edit RAML** window should appear with a warning. Select the radio button **"Yes, send me to the text editor"** and then the **Continue** Button.



3. You will notice that we are no longer in the **Visual Editor,** but are now in **Text Editor:**

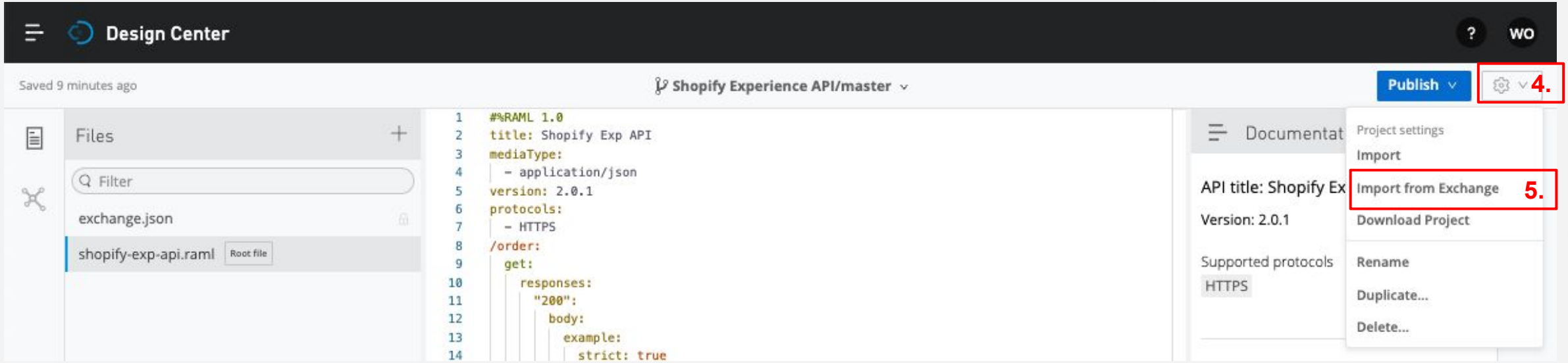


Module 1: The Modern API - Design: Lab 3 (4/17)

Step 1: Complete the Shopify Experience API Specification

4. Go to the Setup located at the top right corner.

5. In the drop down menu select **Import from Exchange**

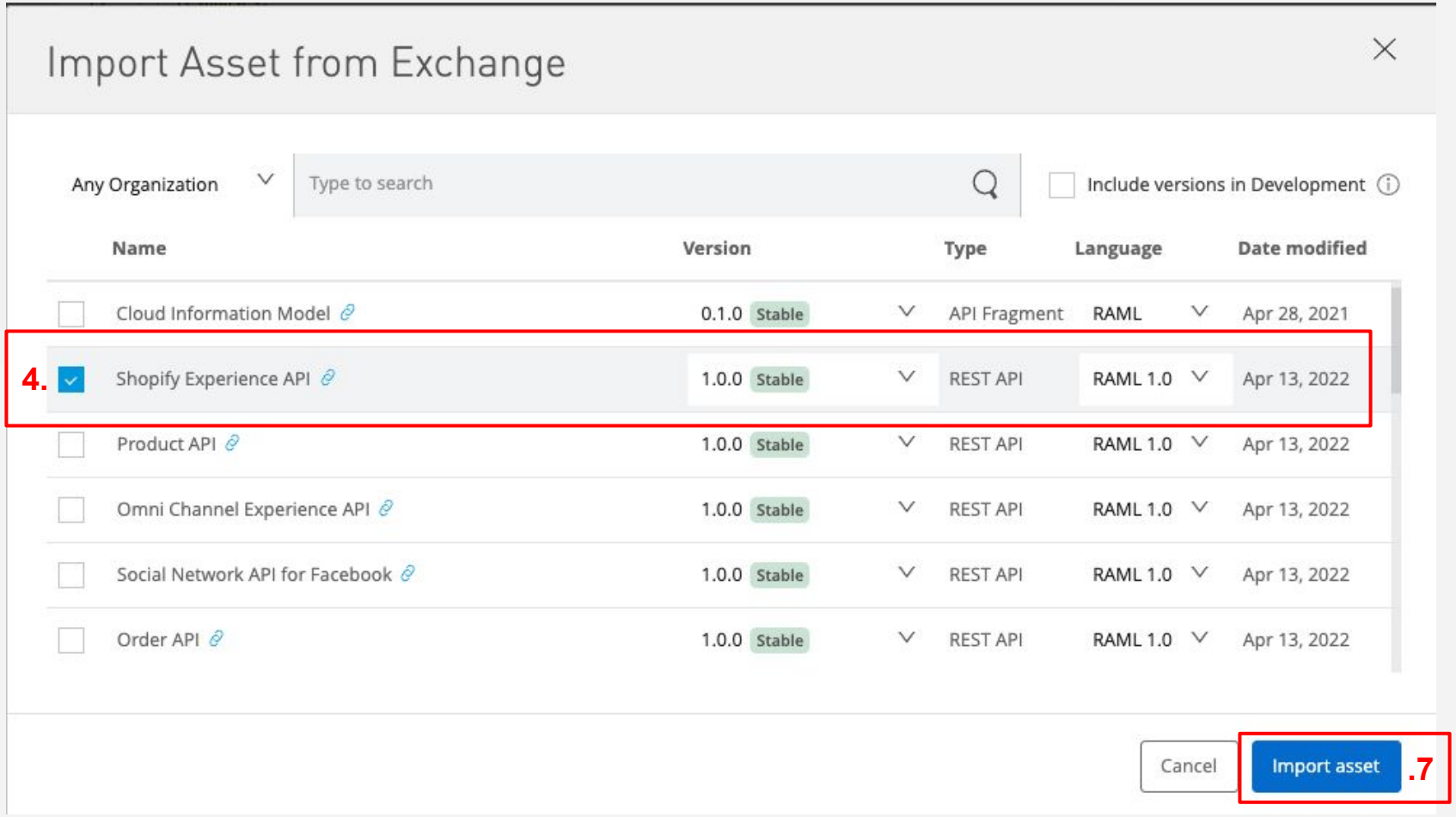


Module 1: The Modern API - Design: Lab 3 (5/17)

Step 1: Complete the Shopify Experience API Specification

6. Look for **Shopify Experience API** and Select it.

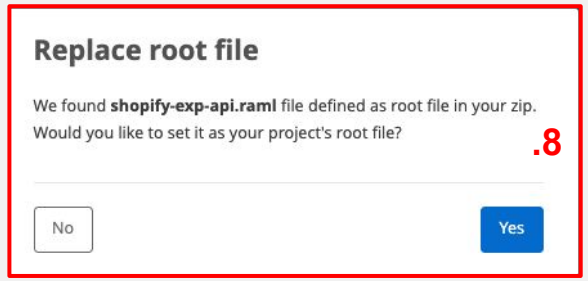
7. Press **Import asset**



Module 1: The Modern API - Design: Lab 3 (6/17)

Step 1: Complete the Shopify Experience API Specification

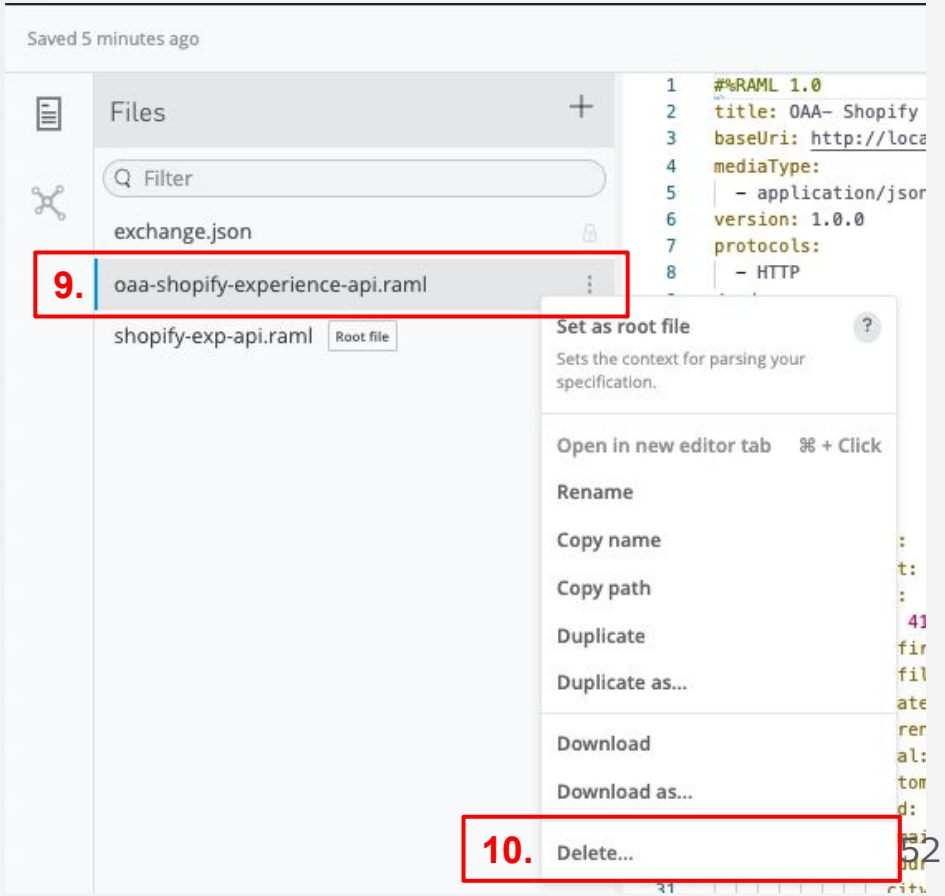
8. Choose **Yes** from the Window **Replace root file**.



9. On the left pane locate the **<you name or initials>-shopify-experience-api** and click on the three dots.

10. Select last option on the list, **Delete…**

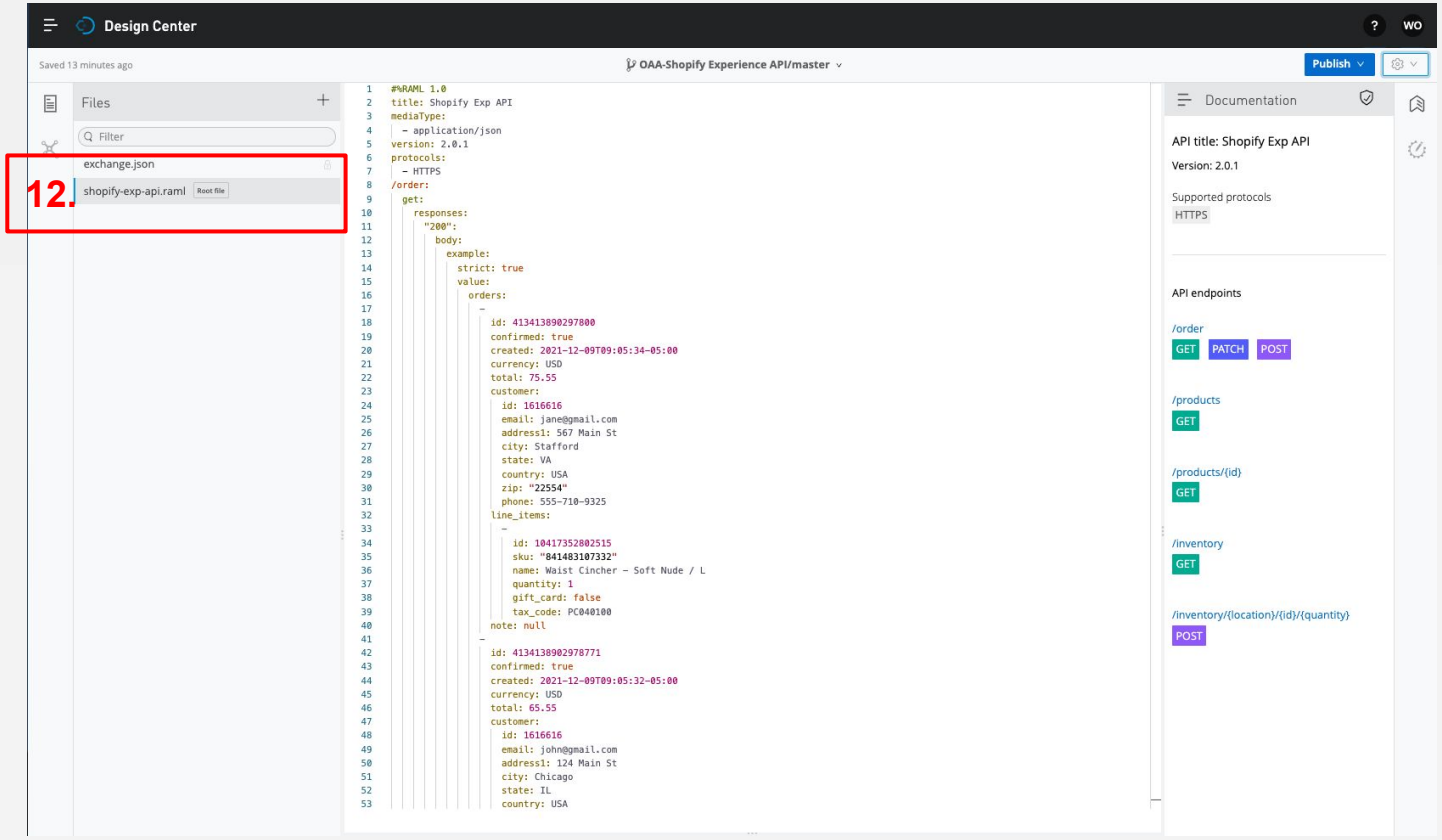
11. A **Delete File** Window will appear, Click **Delete**.



Module 1: The Modern API - Design: Lab 3 (7/17)

Step 1: Complete the Shopify Experience API Specification

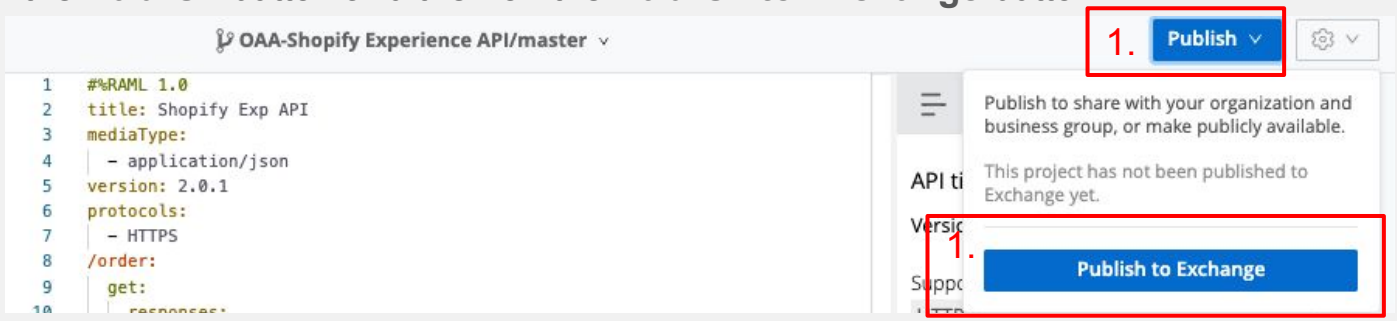
12. You should see the following complete API specification with only the **shopify-experience-api.raml** in the left pane:



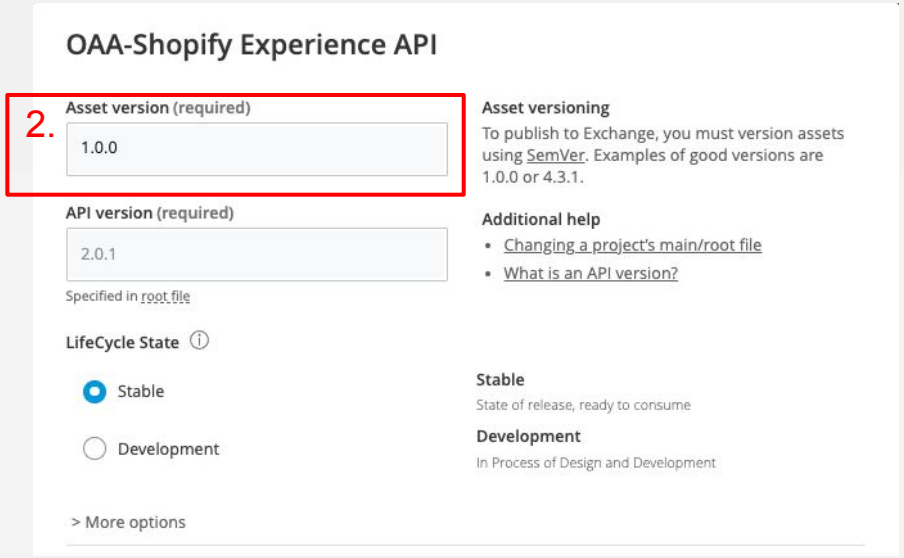
Module 1: The Modern API - Design: Lab 3 (8/17)

Step 2: Publish Your API to Anypoint Exchange

1. Click on the **Publish** button and then on the **Publish to Exchange** button:



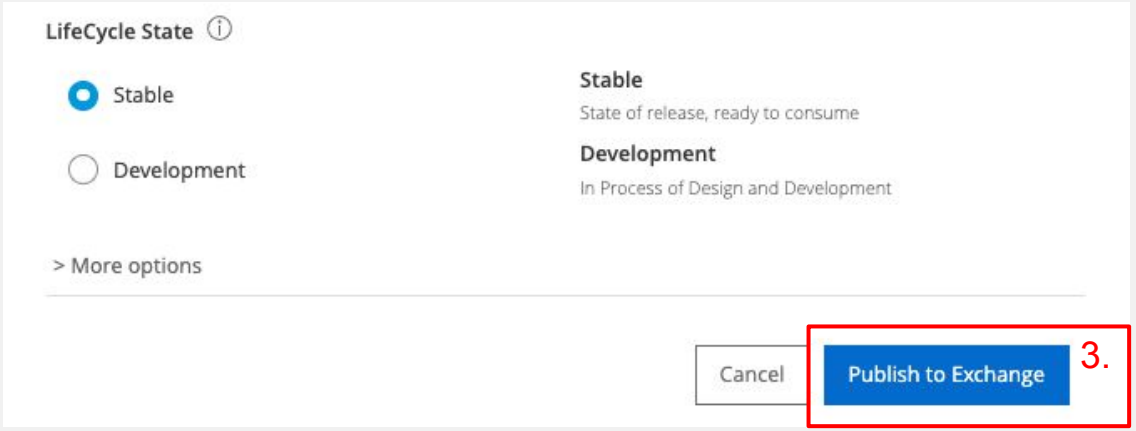
2. A popup window to capture the attributes that you want to publish to Exchange will appear. Most of these fields will be pre-populated and the RAML will be verified to ensure that there are no errors. Complete the **Asset version** field with "`1.0.0`."



Module 1: The Modern API - Design: Lab 3 (9/17)

Step 2: Publish Your API to Anypoint Exchange

3. Click the "Publish to Exchange" button when you have finished reviewing the API name and version information. We have now published our Shopify Experience API to **Anypoint Exchange** for other developers to discover and consume!



Each time you publish an API Specification you will generate two assets:

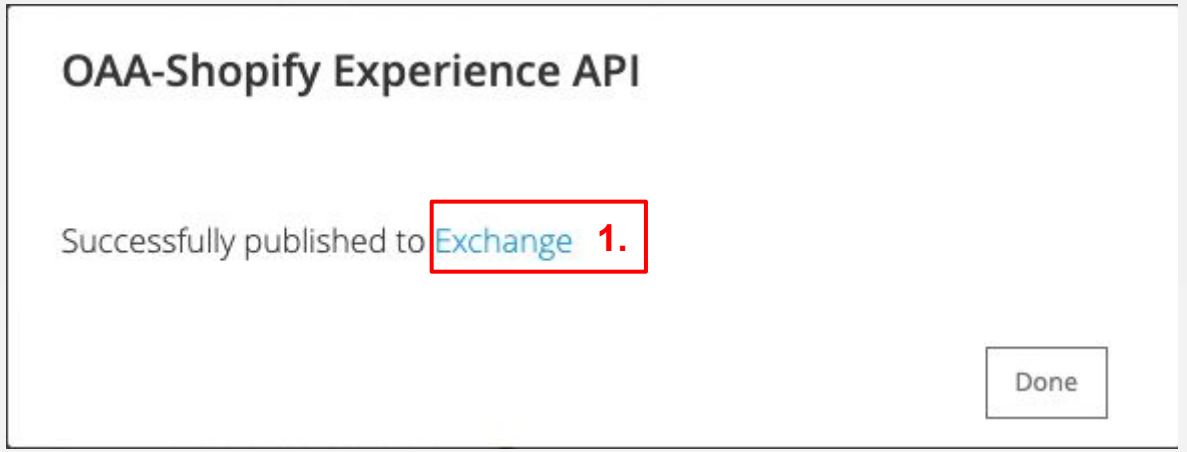
- RAML Spec: This is the RAML Specification. Everybody on the organization can discover the spec and documentation.
- Rest Connector: This is a connector automatically generated for the API you have just defined. We will cover this in more details on Module 2 and Module 9.

Module 1: The Modern API - Design: Lab 3 (10/17)

Step 3: Augment the Documentation for Your API

As demonstrated in Step 2, publishing your API to exchange is as easy as a few clicks. However, Exchange has really only just captured the basic attributes and documentation for your API at this point. Ensuring that your API is easy to find and "self-service ready" is your responsibility as an API designer. Let's find our API in Exchange and add more to our API documentation.

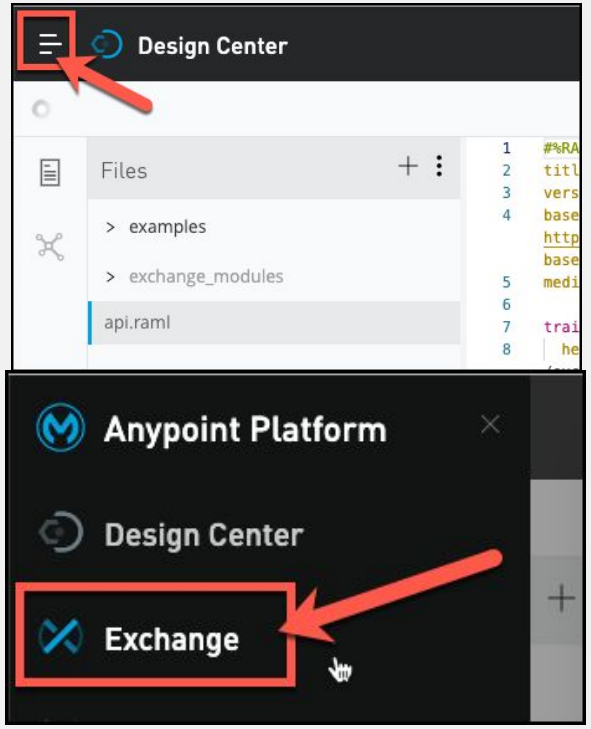
1a. After you click the **Publish to Exchange** the **Shopify Experience API** Specification will start loading into **Exchange**. Once loaded there will be a pop window letting you know that your **Shopify Experience API** is in **Exchange**. To go to the API specification you can simply click on the **"Exchange"** link in the window.



Module 1: The Modern API - Design: Lab 3 (11/17)

Step 3: Augment the Documentation for Your API

1b. **Alternatively,** if you had already closed out of the confirmation window you can Navigate to Exchange from the "hamburger menu" at the top left corner of **Design Center**.

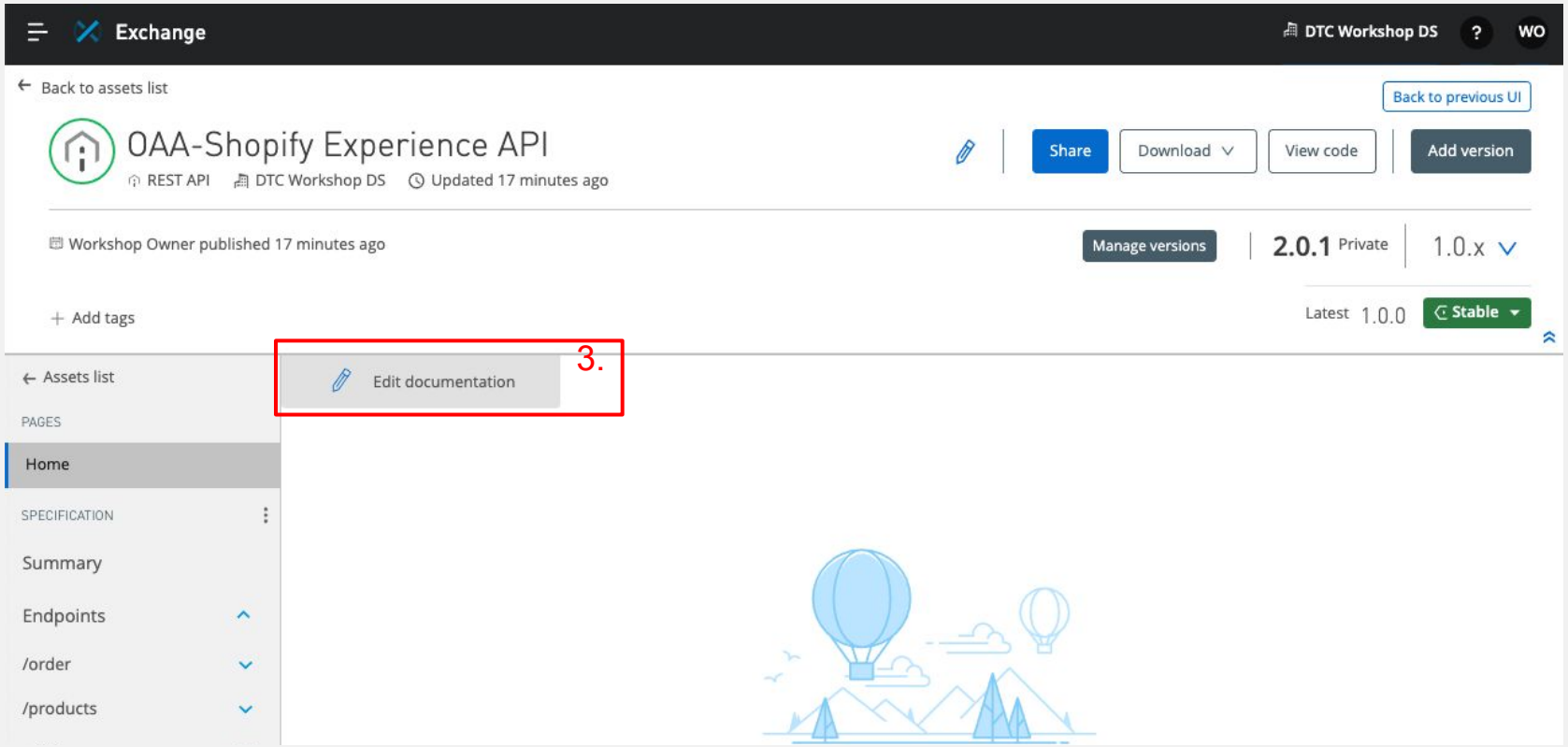


You can then quickly search for the **"<your name or initial>-Shopify Experience API"** in **Exchange** as we did **Lab 1** of this **Module**

Module 1: The Modern API - Design: Lab 3 (12/17)

Step 3: Augment the Documentation for Your API

2. Once you are in the **<your name or initial>-Shopify Experience API** you should see something that looks like this:



3. Let's start adding some content to the API by click on the **Edit documentation** icon

Module 1: The Modern API - Design: Lab 3 (13/17)

MuleSoft

Step 3: Augment the Documentation for Your API

4a. Exchange assets are documented using "markdown" language. Markdown language is a format-agnostic syntax used for creating documentation independent of how it will be rendered (i.e. HTML, PDF, text, etc). It is used by popular applications like GitHub as the standard way to create software documentation. For more information on markdown please refer to:

https://guides.github.com/features/mastering-markdown/

Create your own documentation for your API, or copy the following text into the editor:

```
Welcome to the **NTO developer portal**. This API represents the Direct to
Consumer Shopify Shopping Cart Application - that includes:

-order
-product
-inventory tracking

The **Shopify API is an Experience API** that is a part of NTO's API led
Architecture . This same architecture allows NTO to leverage a **headless
commerce** platform that is **highly composable** and **future proof!**
```

If you have problems copying the code you can find example here

Module 1: The Modern API - Design: Lab 3 (14/17)

MuleSoft

Step 3: Augment the Documentation for Your API

4b. **Alternatively**, if you are more comfortable with WYSIWYG editing you can click on the Visual button and copy an paste the following into the editor body:

Welcome to the **NTO developer portal.** This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes:

- Order
- Product
- inventory tracking

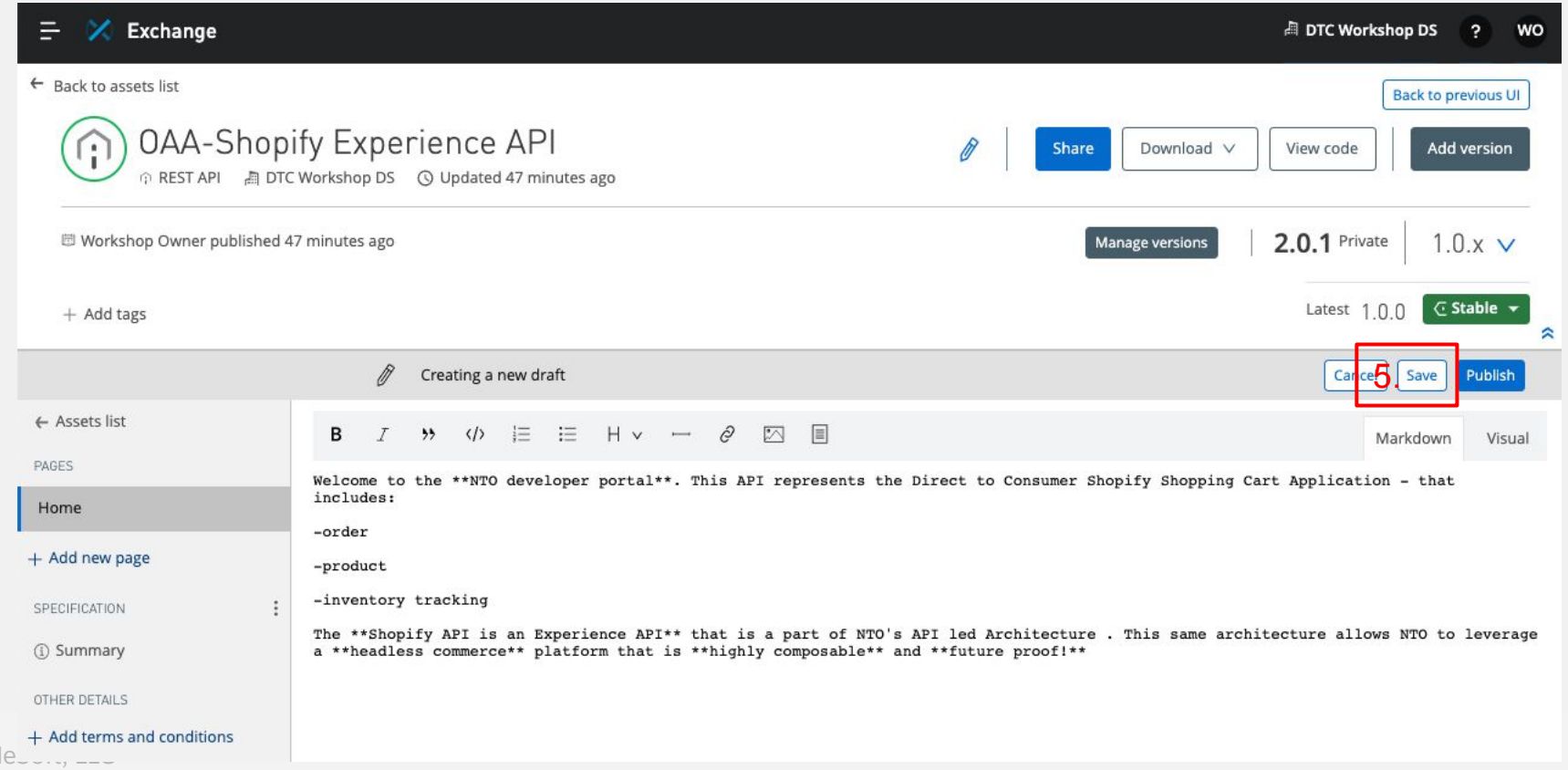
The **Shopify API is an Experience API** that is a part of NTO's API led Architecture . This same architecture allows NTO to leverage a **headless commerce** platform that is **highly composable** and **future proof!**

If you have problems copying the code you can find example here

Module 1: The Modern API - Design: Lab 3 (15/17)

Step 3: Augment the Documentation for Your API

5. Click on the **"Save"** button at bottom of screen. Your draft of the API documentation is saved but not published until you formally **"Publish"** the final updates to Exchange, so feel free to save multiple drafts as you create your documentation.



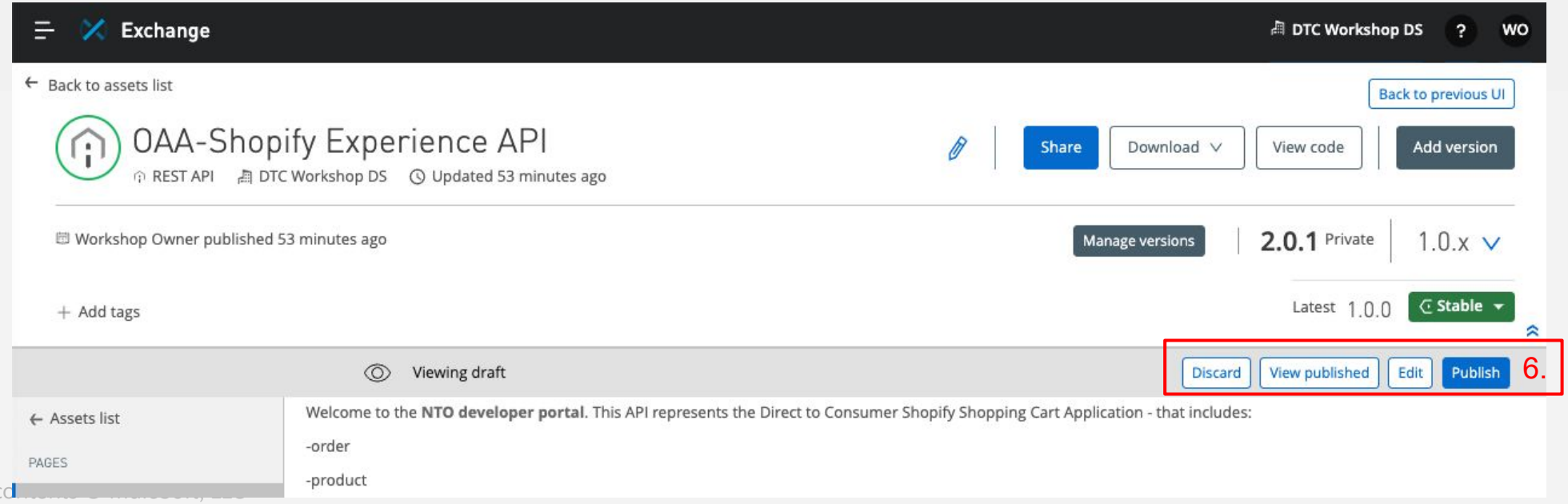
Module 1: The Modern API - Design: Lab 3 (16/17)

MuleSoft

Step 3: Augment the Documentation for Your API

6. You can press `Publish`, `Discard` , `Edit` or `View published`.

    - Discard - Throw away changes
    - View Published - View published documentation
    - Edit - Lets you edit the content
    - Publish - Publish documentation changes to exchange

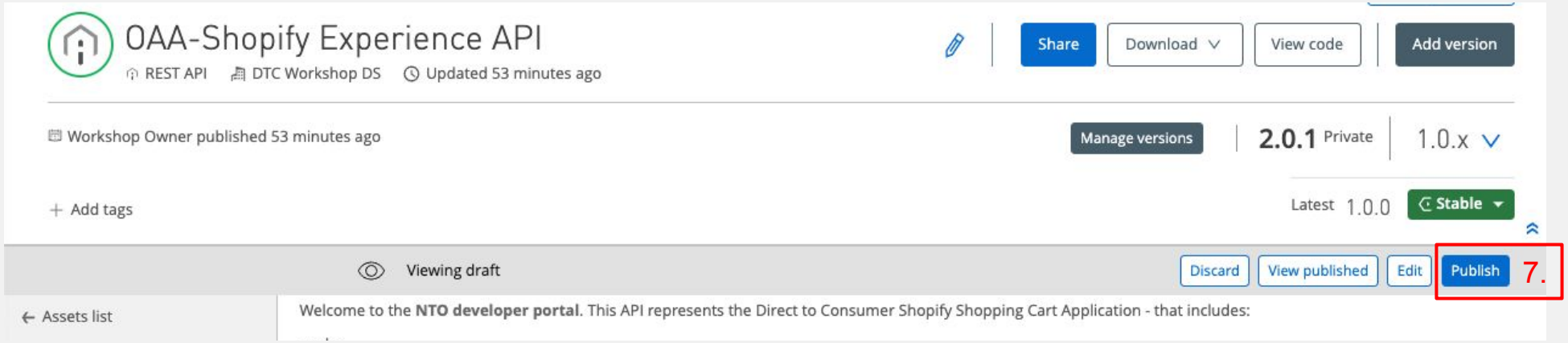


Module 1: The Modern API - Design: Lab 3 (17/17)

MuleSoft

Step 3: Augment the Documentation for Your API

7. Press the **`Publish`** button. Your documentation changes are now visible on Exchange!



Exchange documentation can be comprised of multiple pages. So far we have just created the home page for our API.

It is common practice to include links to other content like video tutorials in our Exchange documentation. This makes it easier for the interested developer to gain access to all of the documentation required to learn how to use the API.

Congratulations! You have completed all of the labs in Module 1!

Proceed to Module 2

MuleSoft

Module 2: Ubiquitous Connectivity

Source: DTC e-commerce: How consumer brands can get it right

Module 2: Ubiquitous Connectivity: Overview (1/2)

Overview

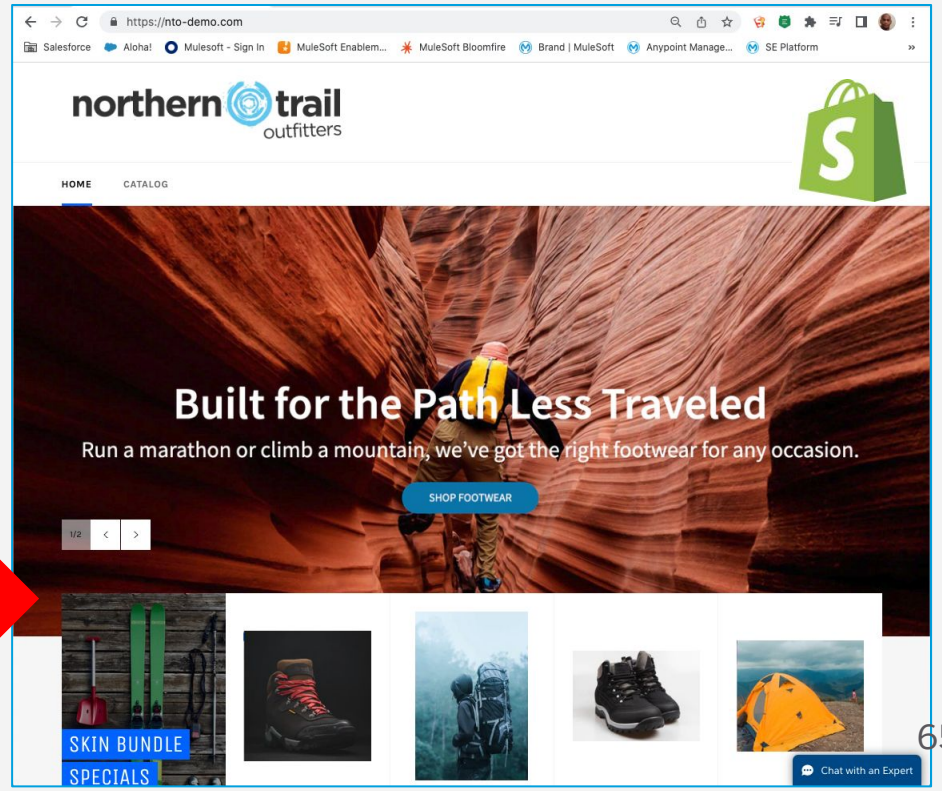
You have completed the design of the Shopify API and now you are ready to implement it. In this module, we will create the implementation of our API that will process the REST requests.

The Module is separated into three different labs.

- Lab 1: Create Shopify Project from API Specification
- Lab 2: Implement the Shopify API in Studio
- Lab 3: Deploy the Shopify API Implementation

To test the connectivity to the NTO Shopify shopping cart we will be testing a single use case of retrieving all Shopify orders. Since sourcing consumer orders from Shopify is critical for NTO's DTC goals we will make sure that works.

We will leverage the https://nto-demo.com/ Shopify site to do just this.



Module 2: Ubiquitous Connectivity: Overview (2/2)

Overview

This module will feature low code connectivity via what we call "**connectors".** Today Anypoint Studio has over 200 system specific connectors (e.g. Salesforce, Shopify, SAP, etc.) that makes connecting to some of our main source systems easy. Since NTO is using Shopify, we will be getting very familiar with the **Shopify Connector.**

Proceed to Lab 1

MuleSoft

Module 2: Lab 1 Create Shopify Project from API Specification

Source: DTC e-commerce: How consumer brands can get it right

Module 2: Ubiquitous Connectivity: Lab 1 (1/16)

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 requests, 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 Shopify API RAML
2. Review the Project with the Shopify API Scaffolding in place

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

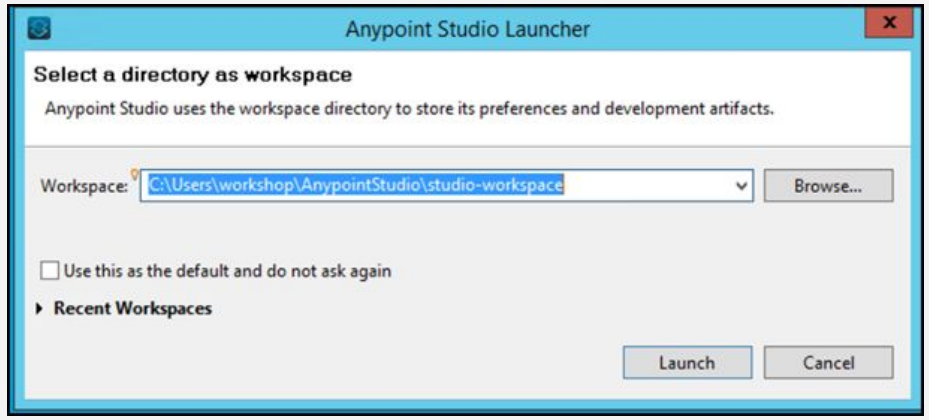
Module 2: Ubiquitous Connectivity: Lab 1 (2/16)

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. If you are using Remote Desktop look for the Start **Anypoint Studio** from the desktop icon.

2. When you open the Studio for the first time, you are going to be asked for the workspace where the projects are going to be saved. Select the one that comes by default.

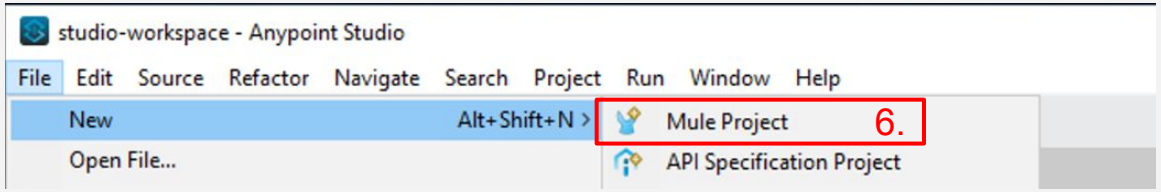


Module 2: Ubiquitous Connectivity: Lab 1 (3/16)

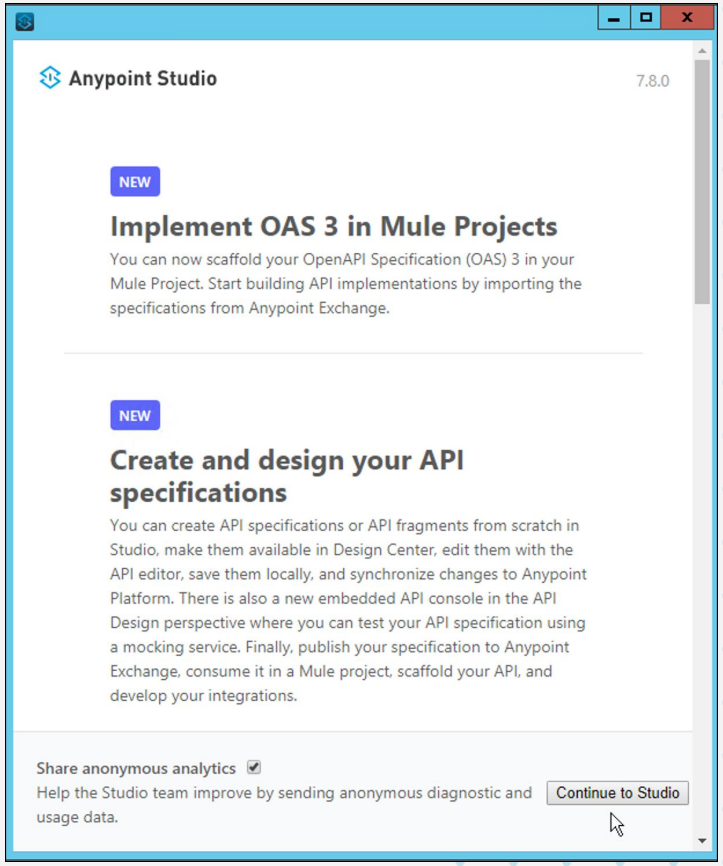
Step 1: Create a new Mule Project from the RAML

3. You are presented with the Anypoint Studio Welcome Screen.
4. Scroll to the bottom if you want to know more on Studio. After that, press **Continue to Studio**.
5. In the Workspace Launcher, use the default workspace directory location.
6. 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.



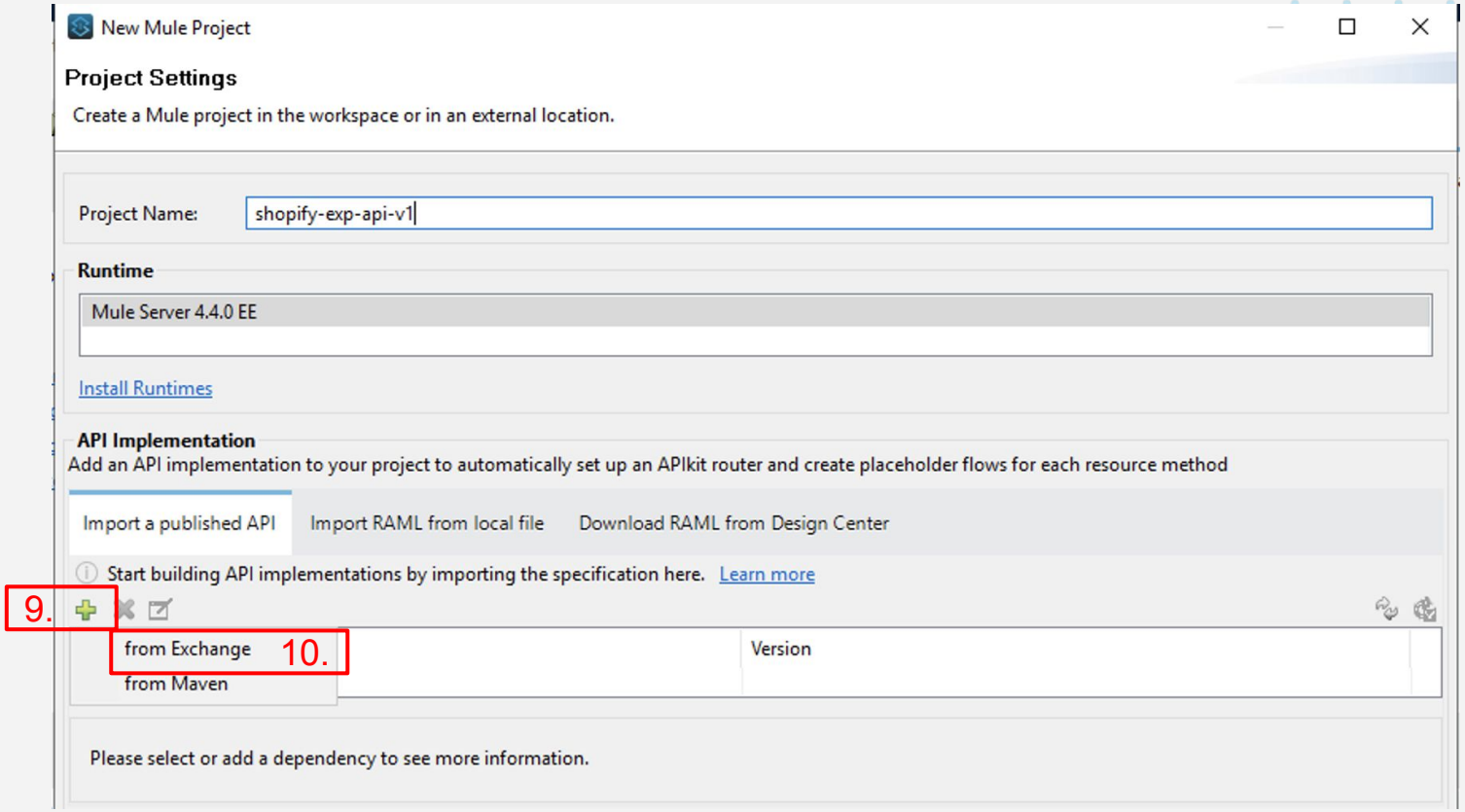
7. Give the project the name **shopify-exp-api-v1**
8. Select the **Mule Server 4.4.0 EE** (or latest version if present).



Module 2: Ubiquitous Connectivity: Lab 1 (4/16)

Step 1: Create a new Mule Project from the RAML

9. Under **Import a Published API** we are going to import the RAML Spec from Exchange. Click on **green +** this will present two options from Exchange or from Maven
10. Select from **Exchange.**



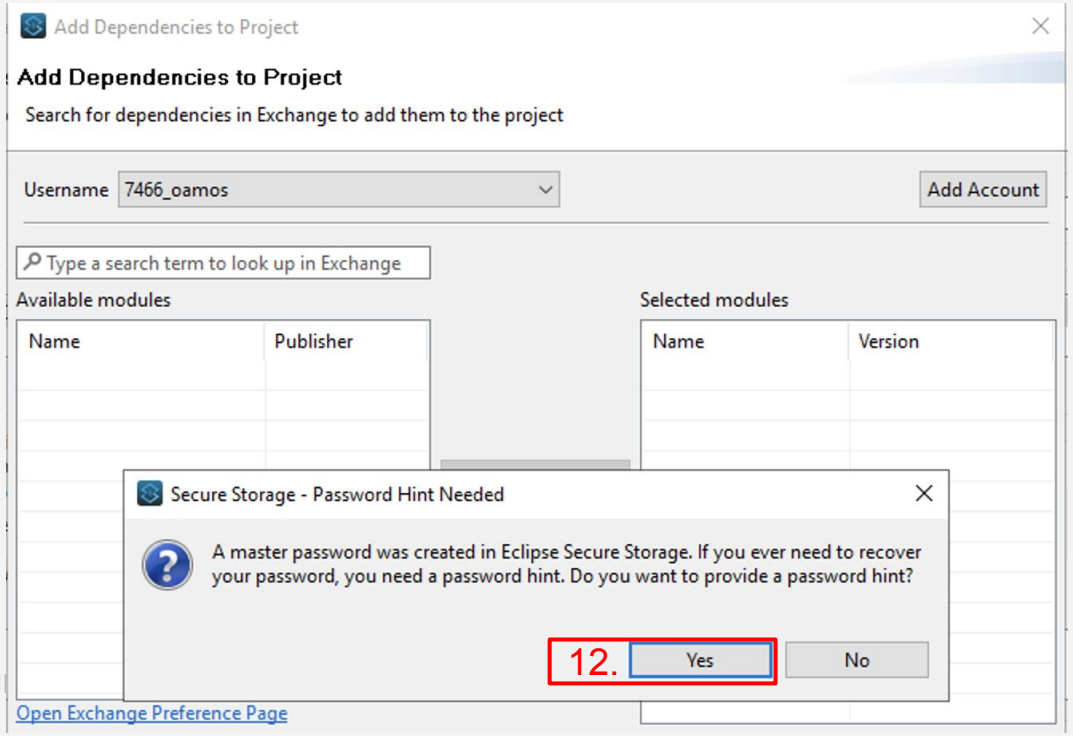
Module 2: Ubiquitous Connectivity: Lab 1 (5/16)

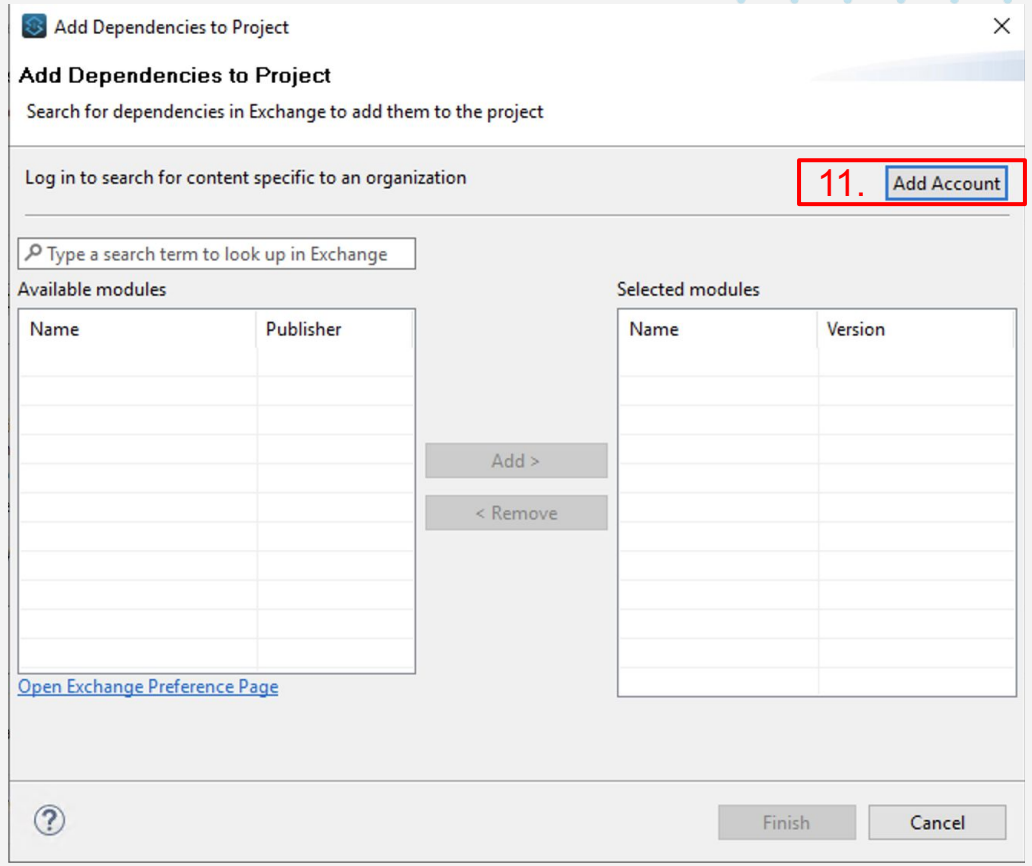
MuleSoft

Step 1: Create a new Mule Project from the RAML

11. Once you clicked there, you will see a new window. You need to login to the Anypoint Platform. So click on the **Add Account** button.

12. A **Secure Storage** prompt will appear click **Yes** to proceed.

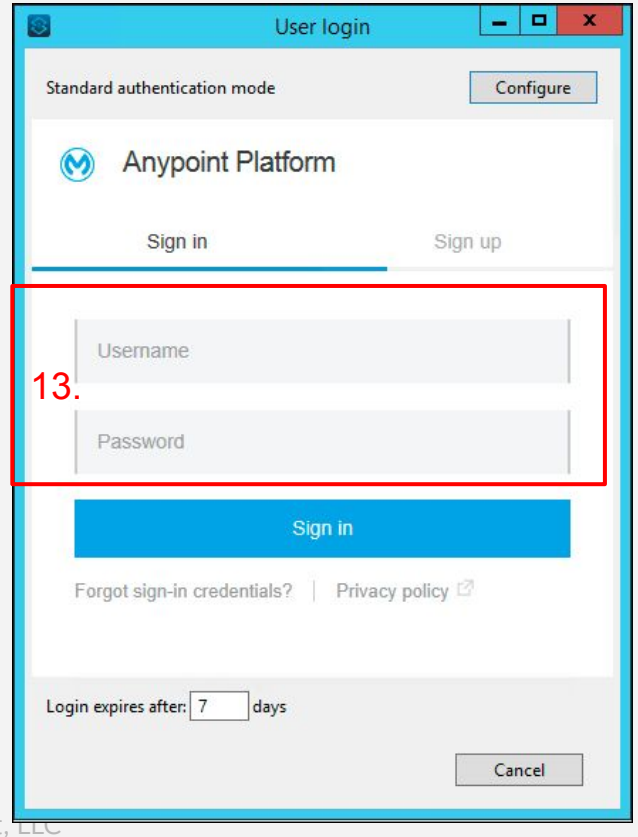




Module 2: Ubiquitous Connectivity: Lab 1 (6/16)

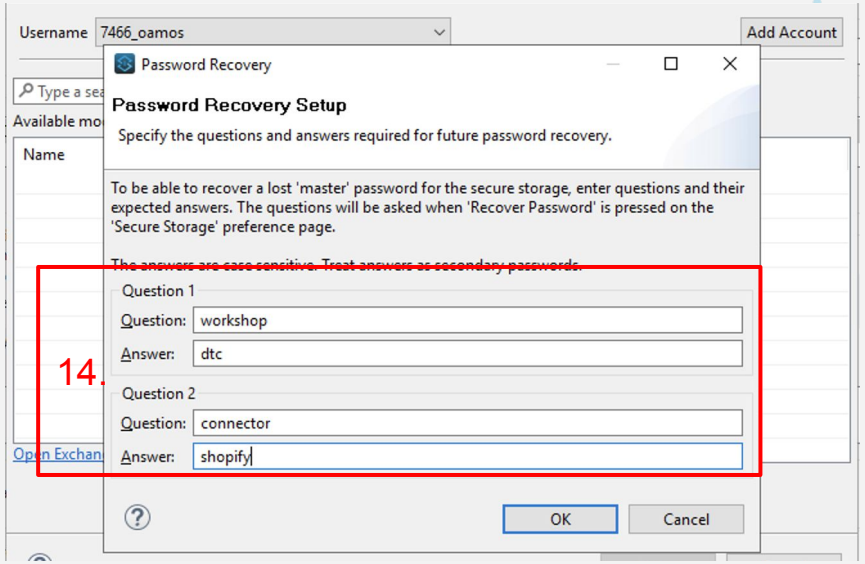
Step 1: Create a new Mule Project from the RAML

13. A new Login window will appear. Insert the credentials that were sent to your email earlier.

- 

14. You will be taken to a Password Recovery Setup window. From there add the following details:

- Question 1: Question: **workshop** Answer: **dtc**
- Question 2: Question: **connector** Answer: **shopify**



Module 2: Ubiquitous Connectivity: Lab 1 (7/16)

Step 1: Create a new Mule Project from the RAML

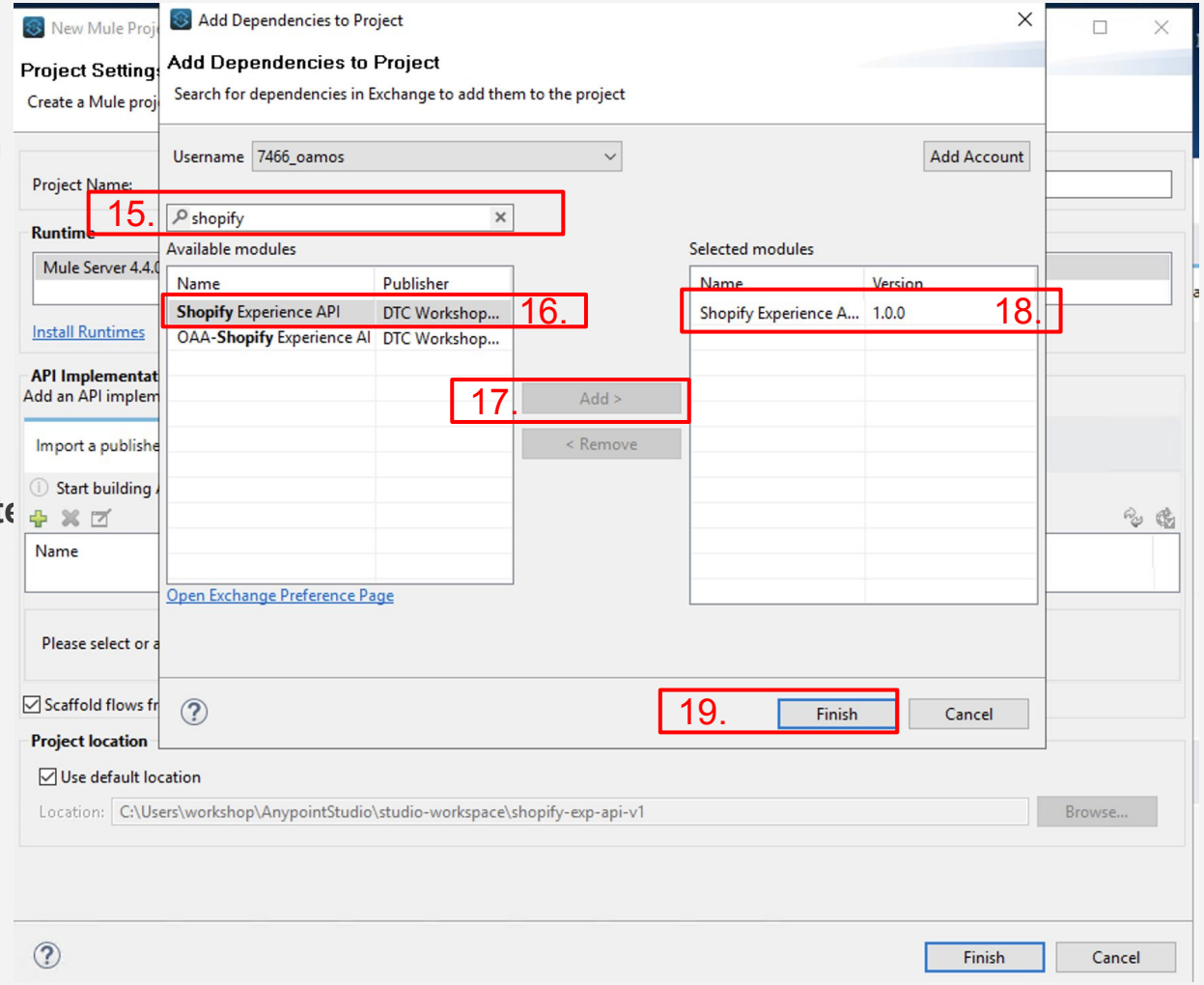
15. You will be taken to the Add Dependencies to Project window. From there type **“shopify”** in the search area.

16. You might see several options, select the **Shopify Experience API** that doesn’t have any names or initials in front of it.

17. Click the **Add** button.

18. The **Shopify Experience API** should be on the right pane of **Selecte modules.** Make sure that is the case before proceeding.

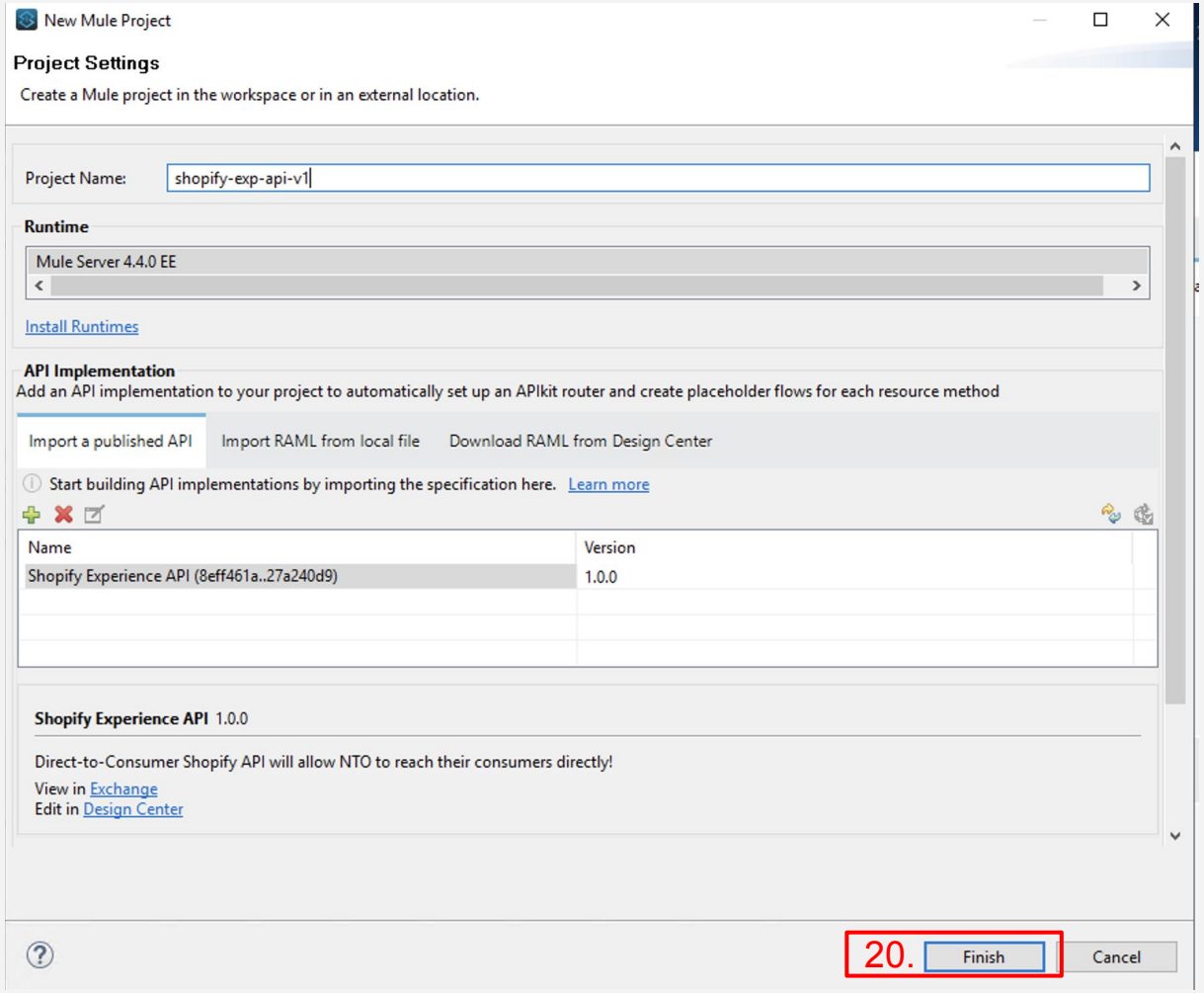
19. Click **Finish**.



Module 2: Ubiquitous Connectivity: Lab 1 (8/16)

Step 1: Create a new Mule Project from the RAML

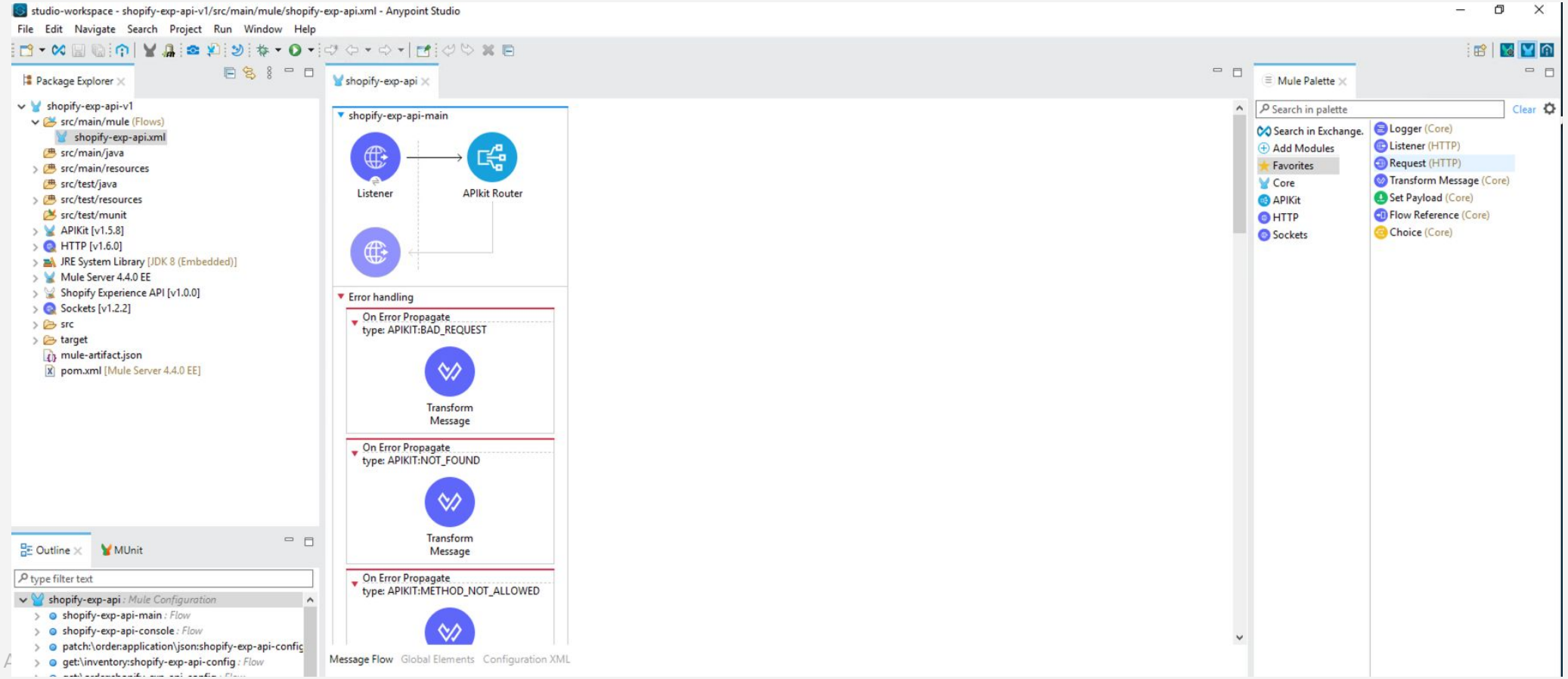
20. You should be back at the **Project Settings** Window. Click **Finish**.



Module 2: Ubiquitous Connectivity: Lab 1 (9/16)

Step 2: Review Scaffolded Flows in the Project

Done! Now let's explore what was automatically generated for our project.



Module 2: Ubiquitous Connectivity: Lab 1 (10/16)

Step 2: Review Scaffolded Flows in the Project

If you click your mouse cursor to the middle pane under the tab called **shopify-exp-api,** you will see our first **"flow"** called **shopify-exp-api-main.** When you scroll your mouse down you will start seeing other flows (e.g. **shopify-exp-api-console**, and so on).

All the flows are defined by your API Design in the RAML file and have created a scaffolding for the Shopify API project. 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.

Module 2: Ubiquitous Connectivity: Lab 1 (11/16)

Step 2: Review Scaffolded Flows in the Project

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. The placeholder Transform Message will have sample payload outputs like the one you supplied for your version of the Shopify API in Module 1 Lab 2. Obviously these flows can be enhanced to evolve them into full API implementation as we will see in the next lab.

Module 2: Ubiquitous Connectivity: Lab 1 (12/16)

Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

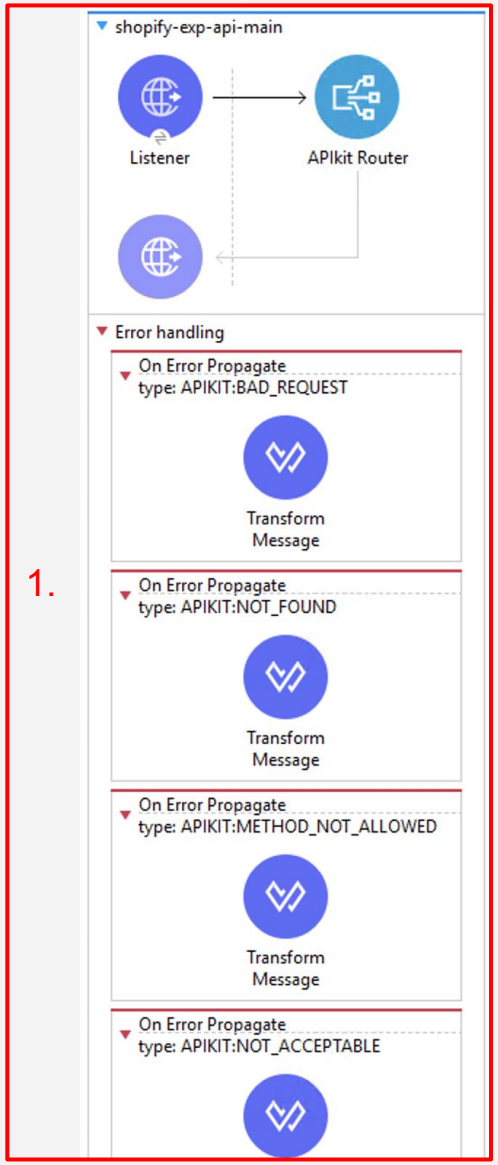
1. **shopify-exp-api-main**

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, there you will see the Protocol, Host and Port attributes, all that information means that the listener is in fact 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.



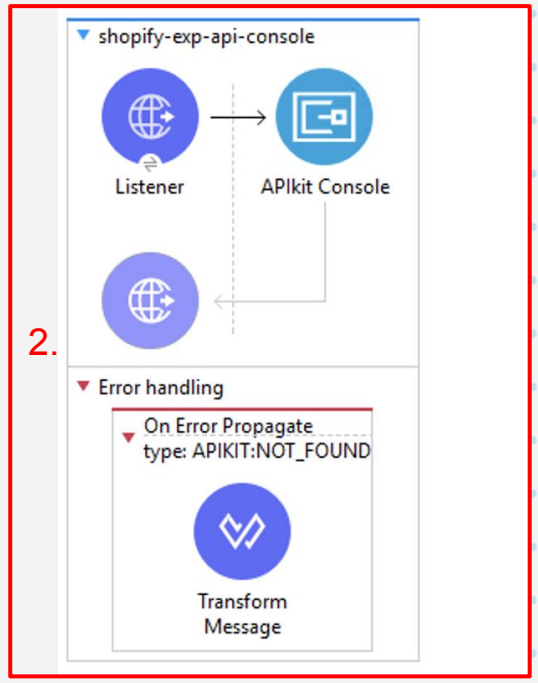
Module 2: Ubiquitous Connectivity: Lab 1 (13/16)

Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

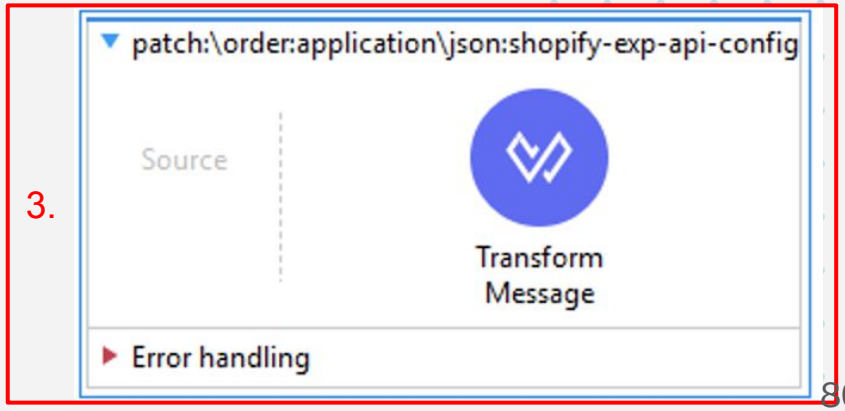
2. **shopify-exp-api-console**

The console flow is not mandatory. It provides an auto generated interactive HTML console for interacting with your api for testing etc. It is auto generated from your RAMl/api spec. Where as to test an API you will be able to test from the similar UI as when where Mocking Services previously in Module 1.



3. **patch:\order:application\json:shopify-exp-api-config**

Is our first flow that represents a resource endpoint. This flow is the framework for the PATCH operator in our order resource of our Shopify API. Patch command will add new consumer orders in NTO's Shopify and can update existing orders from the same call.



Module 2: Ubiquitous Connectivity: Lab 1 (14/16)

Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

4. **get\inventory:shopify-exp-api-config**

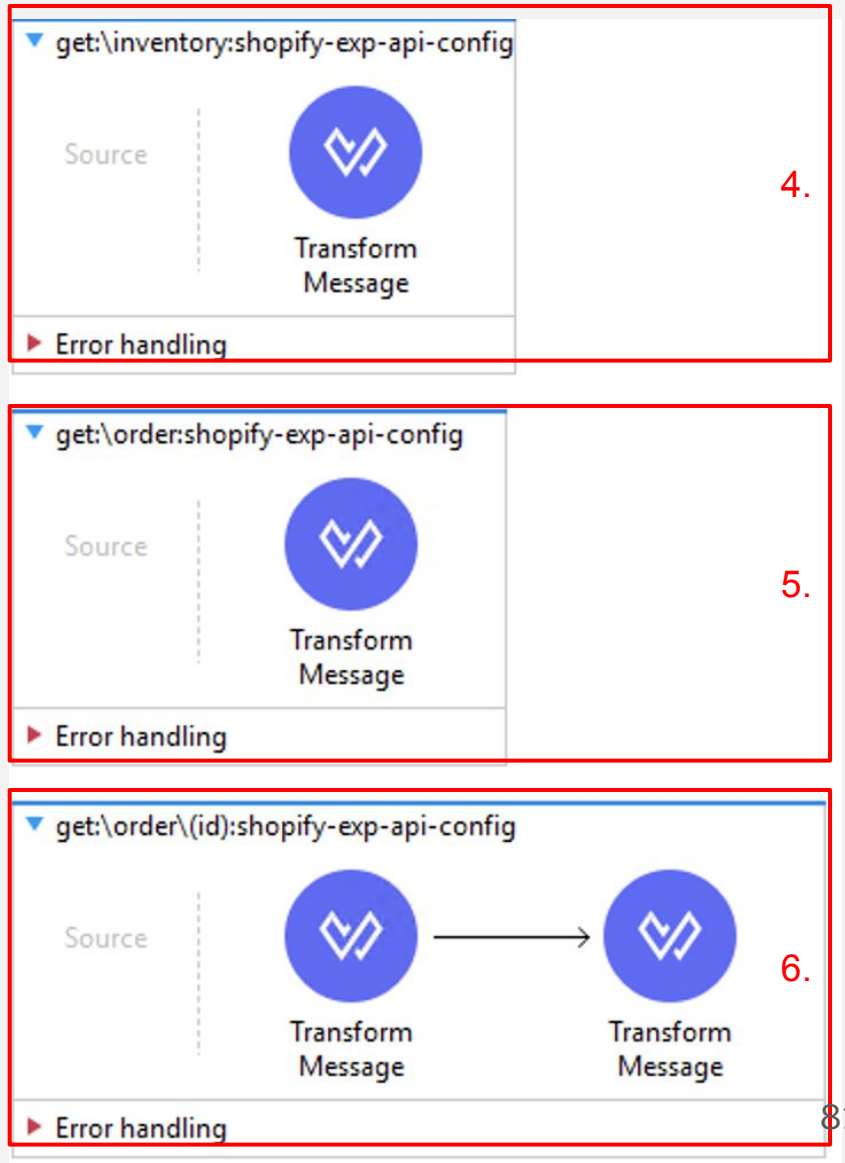
This flow will be the end point to allow the full list retrieval of items and inventory within NTO's Shopify application.

5. **get\order:shopify-exp-api-config**

This flow will get the all consumer orders that reside in NTO's Shopify site. **This will be the flow we will be working on in our next lab, Lab 2.**

6. **get\order\(id):shopify-exp-api-config**

This flow is getting an instance or a single order by ID.



Module 2: Ubiquitous Connectivity: Lab 1 (15/16)

Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

7. **get\products:shopify-exp-api-config**

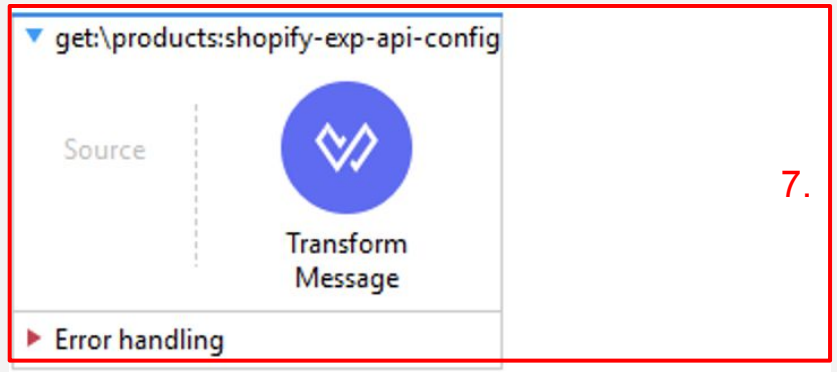
This flow will retrieve full list retrieval of products and product data within NTO's Shopify Site.

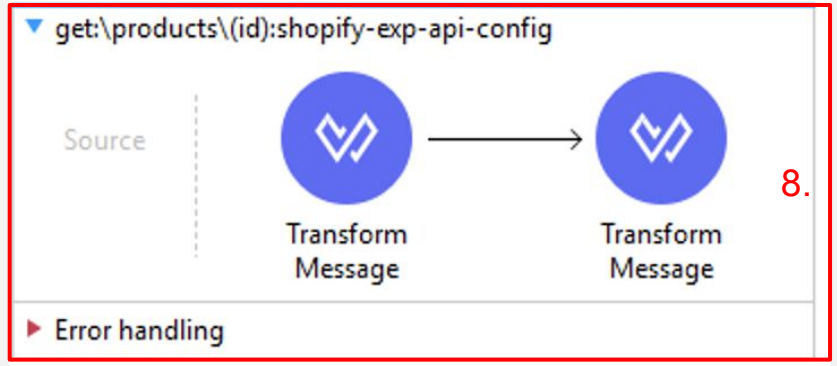
8. **get\products\(id):shopify-exp-api-config**

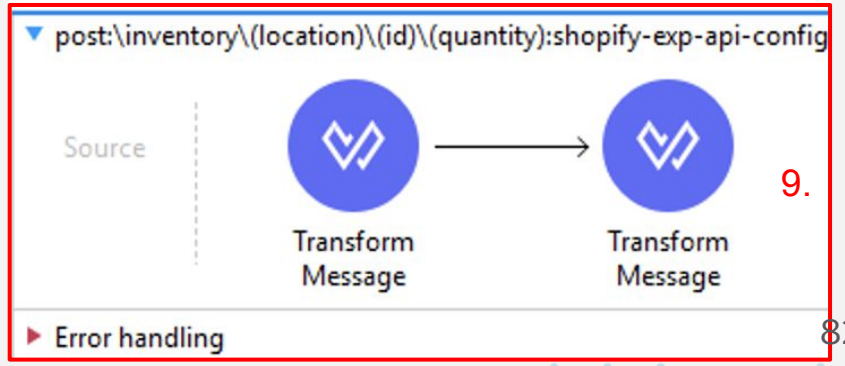
This flow will get product information from a single product.

9. **post:\inventory\(location)\(id)\(quantity):shopify-exp-api-config**

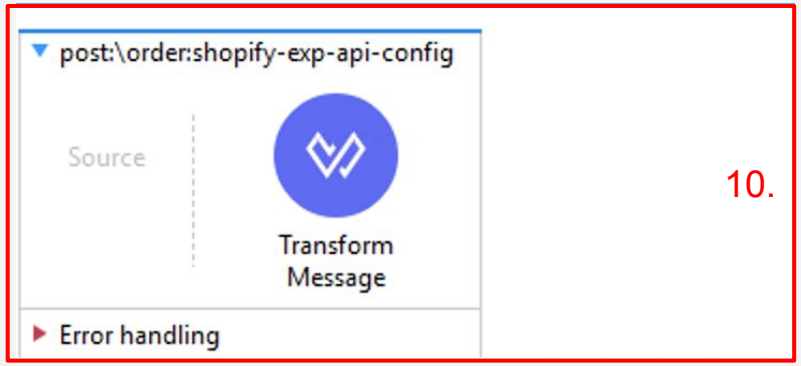
This flow is will allow to update inventory levels within NTO's Shopify Site while passing three needed attributes (Product ID, location ID, Updated Quantity). This way of executing the end point offers a lightweight POST command that doesn't require a BODY in the POST request.







Module 2: Ubiquitous Connectivity: Lab 1 (16/16)



Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

10. **post\order:shopify-exp-api-config**
Lastly, we have a POST orders to allow for consumer orders that were not created in Shopify to be in the Shopify order details as away of allowing for consistent consumer experience across channels.

Now that we have familiarized ourselves with the scaffolding of our Shopify API we can now proceed to building the connectivity to the NTO Shopify Site.

Proceed to Lab 2

MuleSoft

Module 2: Lab 2 Implement the Shopify API In Studio

Source: DTC e-commerce: How consumer brands can get it right

Module 2: Ubiquitous Connectivity: Lab 2 (1/23)

Overview

Since you understand how to start a project by importing the Shopify RAML Specification. You have also seen how the project scaffolding works with all the required resource endpoints. Now we need to add connectivity to Shopify and we will be using a Shopify Connector that will allow us to quickly create connectivity.

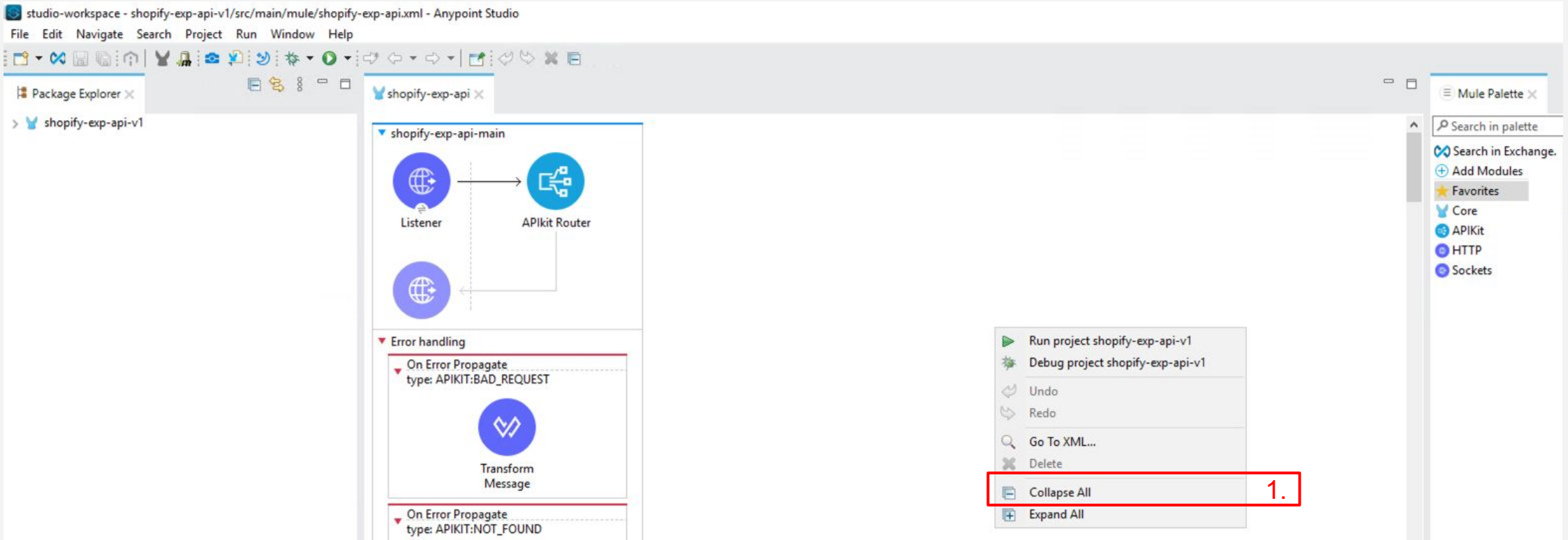
The implementation will consist of a few steps

1. Find the Shopify Connector via Anypoint Studio (through Exchange)
2. Create Configuration Properties YAML to externalize needed Shopify credentials
3. Add our Shopify Connector operator in the flow we will be working on
4. Update the Transform Message in the flow we are working on
5. Test the implementation by running the Shopify project

Module 2: Ubiquitous Connectivity: Lab 2 (2/23)

Step 1: Setup up working on your flow

We will start to add implementation connectivity of Shopify and will test out on flow endpoint resource **get\order:shopify-exp-api-config**. To make working on this endpoint a bit easier let's collapse all of the flows in the Project. We do so by right clicking in the body (any blank area will do) and right-mouse click to see the **Collapse All** option and select it.



Module 2: Ubiquitous Connectivity: Lab 2 (3/23)

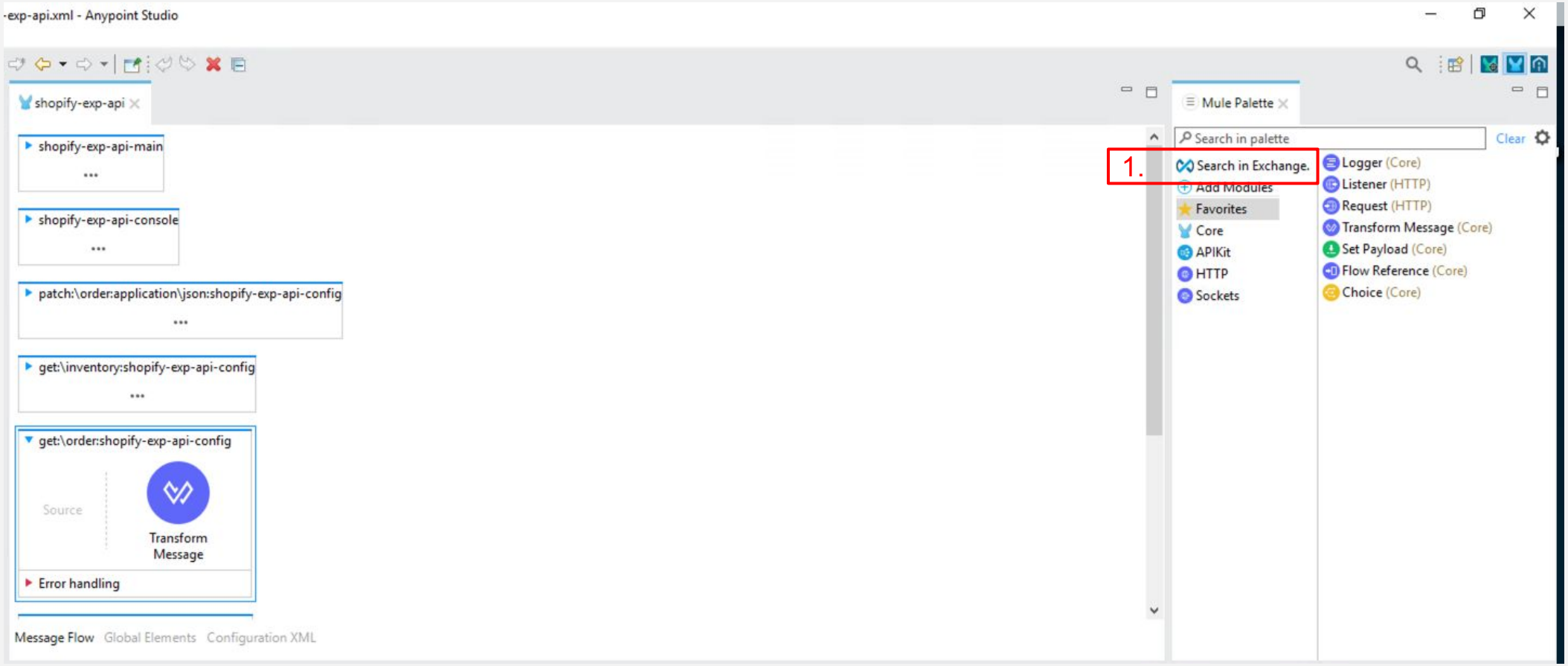
Step 1: Setup up working on your flow

2. Now find the **get\order:shopify-exp-api-config** flow and click on the blue arrow in the top left to expand the flow like this:

Module 2: Ubiquitous Connectivity: Lab 2 (4/23)

Step 2: Find and load the Shopify Connector

1. Put your cursor to the far right pane called Mule Palette. And click on the **Search in Exchange** option.



Module 2: Ubiquitous Connectivity: Lab 2 (5/23)

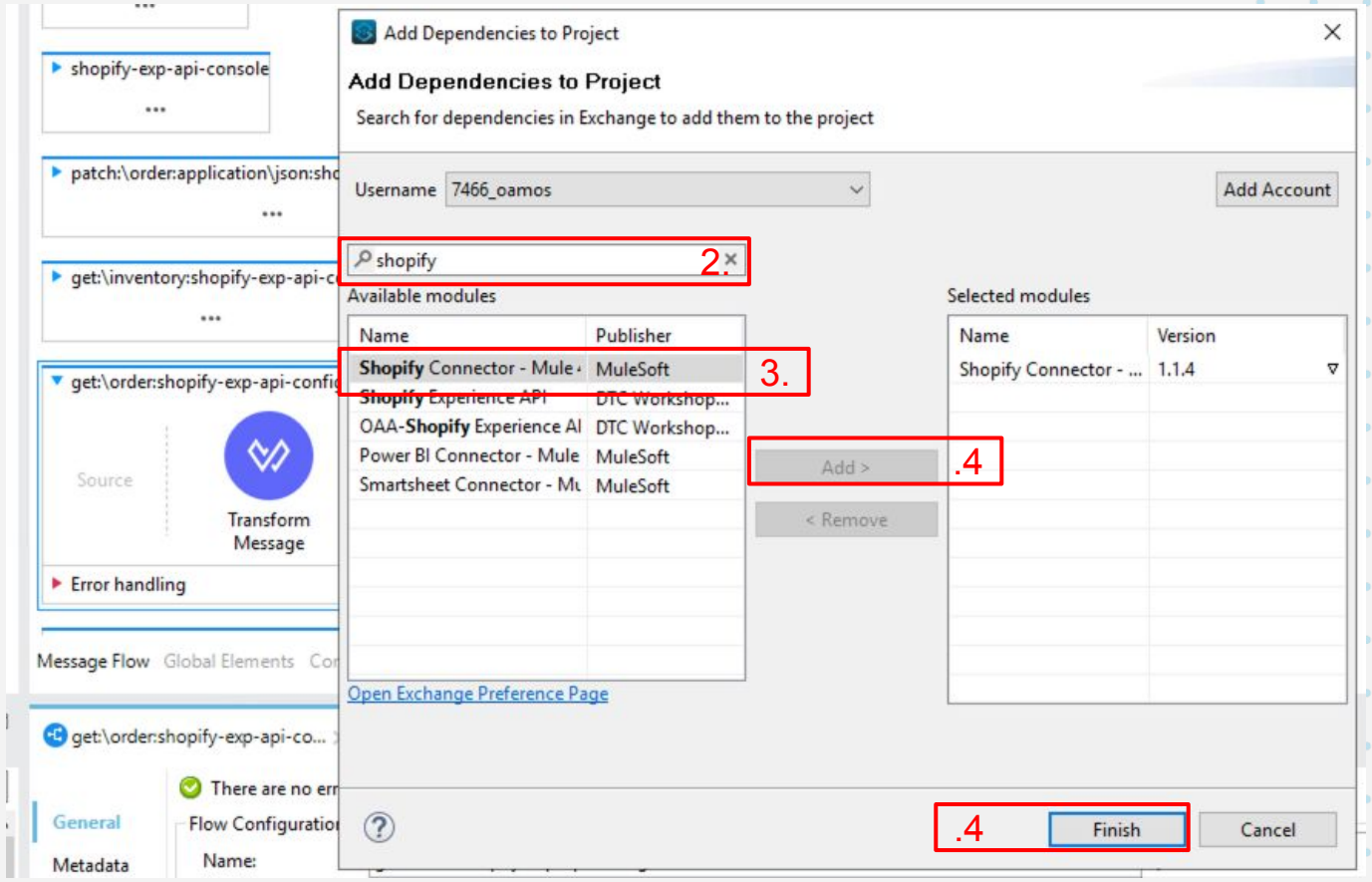
Step 2: Find and load the Shopify Connector

2. The **Add Dependencies to Project** Window will appear. In the search area type **“shopify”.** The **Shopify Connector - “MuleSoft”**should be the first option to appear.

3. Select the connector to highlight it.

3. Select the connector and click the **Add** button.

4. Then Select **Finish.**

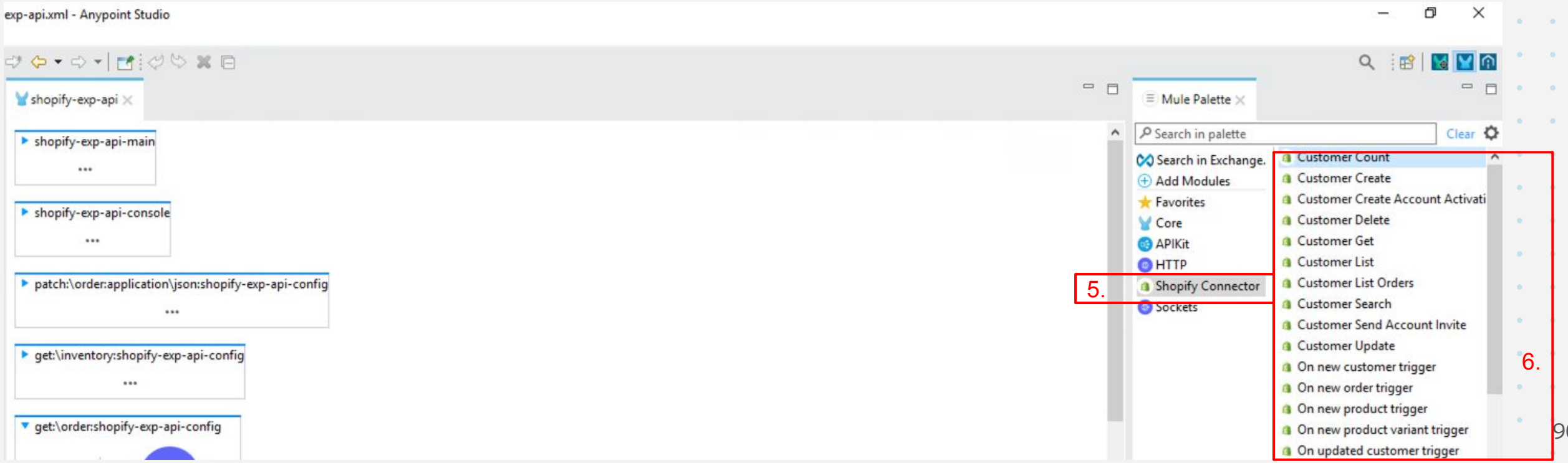


Module 2: Ubiquitous Connectivity: Lab 2 (6/23)

Step 2: Find and load the Shopify Connector

5. The Mule Palette in the right pane should have updated with the Addition of the **Shopify Connector**.

6. On the furthest right sub pane there will be an extensive list of Shopify Operators.



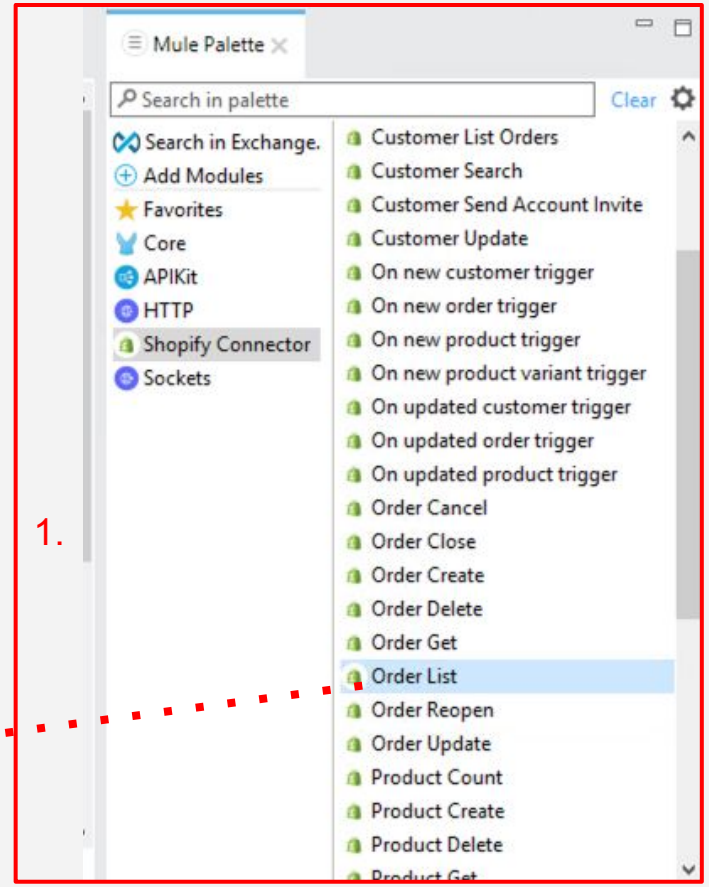
Module 2: Ubiquitous Connectivity: Lab 2 (7/23)

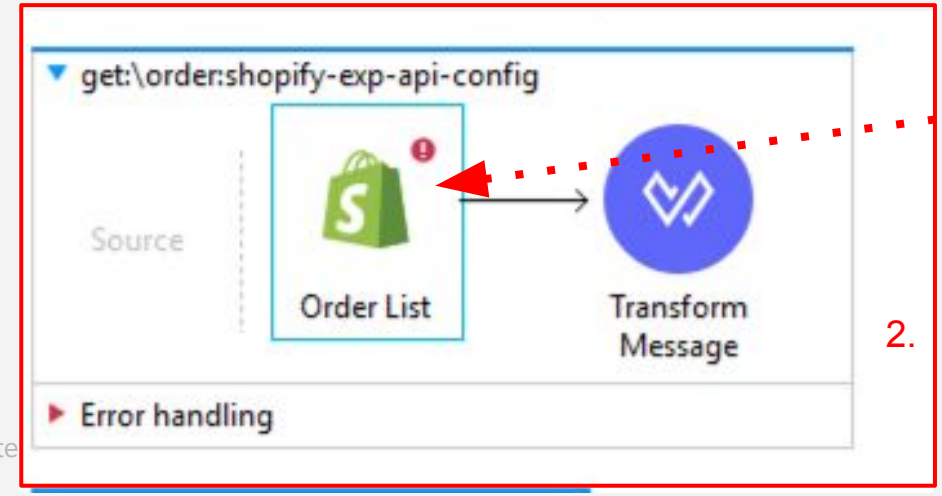
MuleSoft

Step 3: Configure and Use Connector Operation

1. In the operator sub palette Find and Select **Order List** This operator is perfect for what we are looking to do, which is retrieve all orders from within the **NTO Shopify Site**.

2. Highlight and drag **Order List** to the **get:\order:shopify-exp-api-config** flow. Try to drag **Order List** next to the **Transform Message** like so:



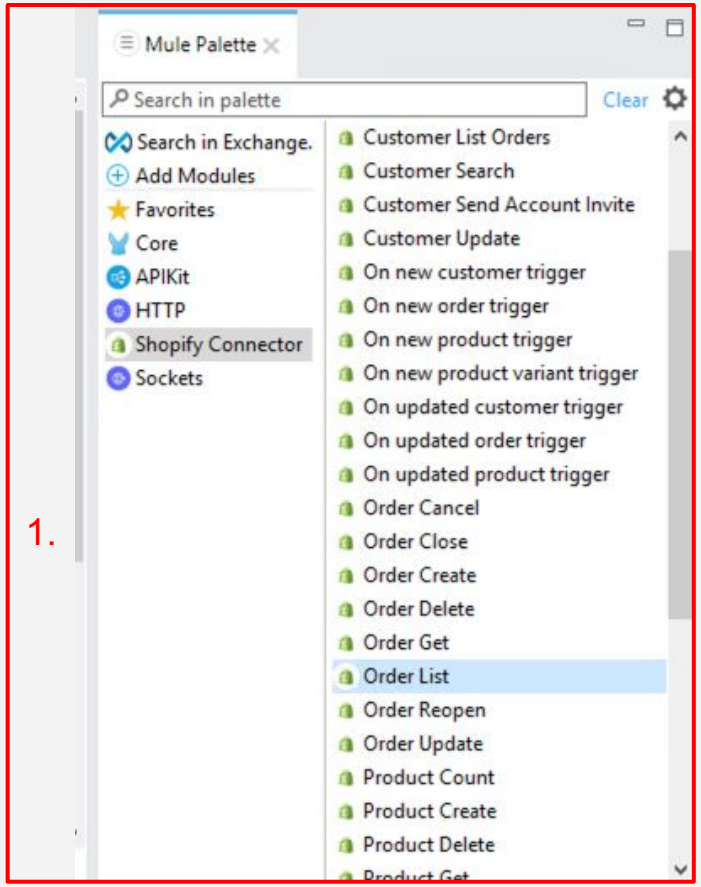


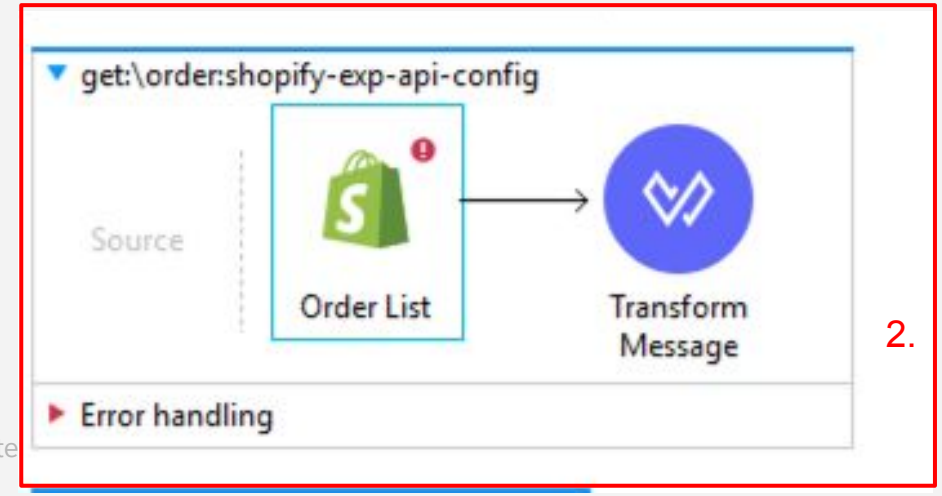
Module 2: Ubiquitous Connectivity: Lab 2 (7/23)

Step 3: Configure and Use Connector Operation

1. In the operator sub palette Find and Select **Order List** This operator is perfect for what we are looking to do, which is retrieve all orders from within the **NTO Shopify Site**.

2. Highlight and drag **Order List** to the **get:\order:shopify-exp-api-config** flow. Try to drag **Order List** next to the **Transform Message** like so:

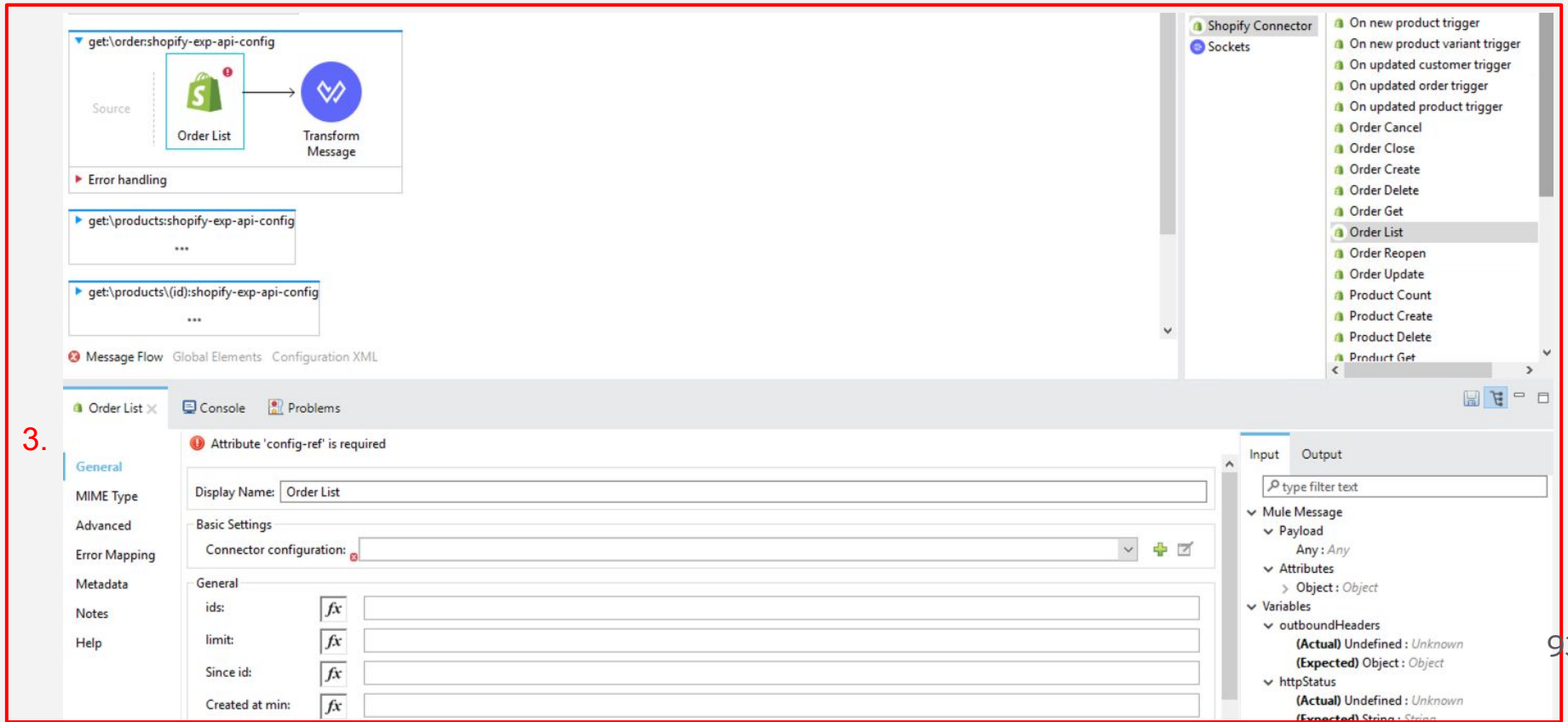




Module 2: Ubiquitous Connectivity: Lab 2 (8/23)

Step 3: Configure and Use Connector Operation

3. Double click on the **Order List in** the **get:\order:shopify-exp-api-config** flow. Look at the lower pane just below the flow and you will see configuration settings with an **Attribute 'config-ref' is required** Warning. We need to configure the connector.

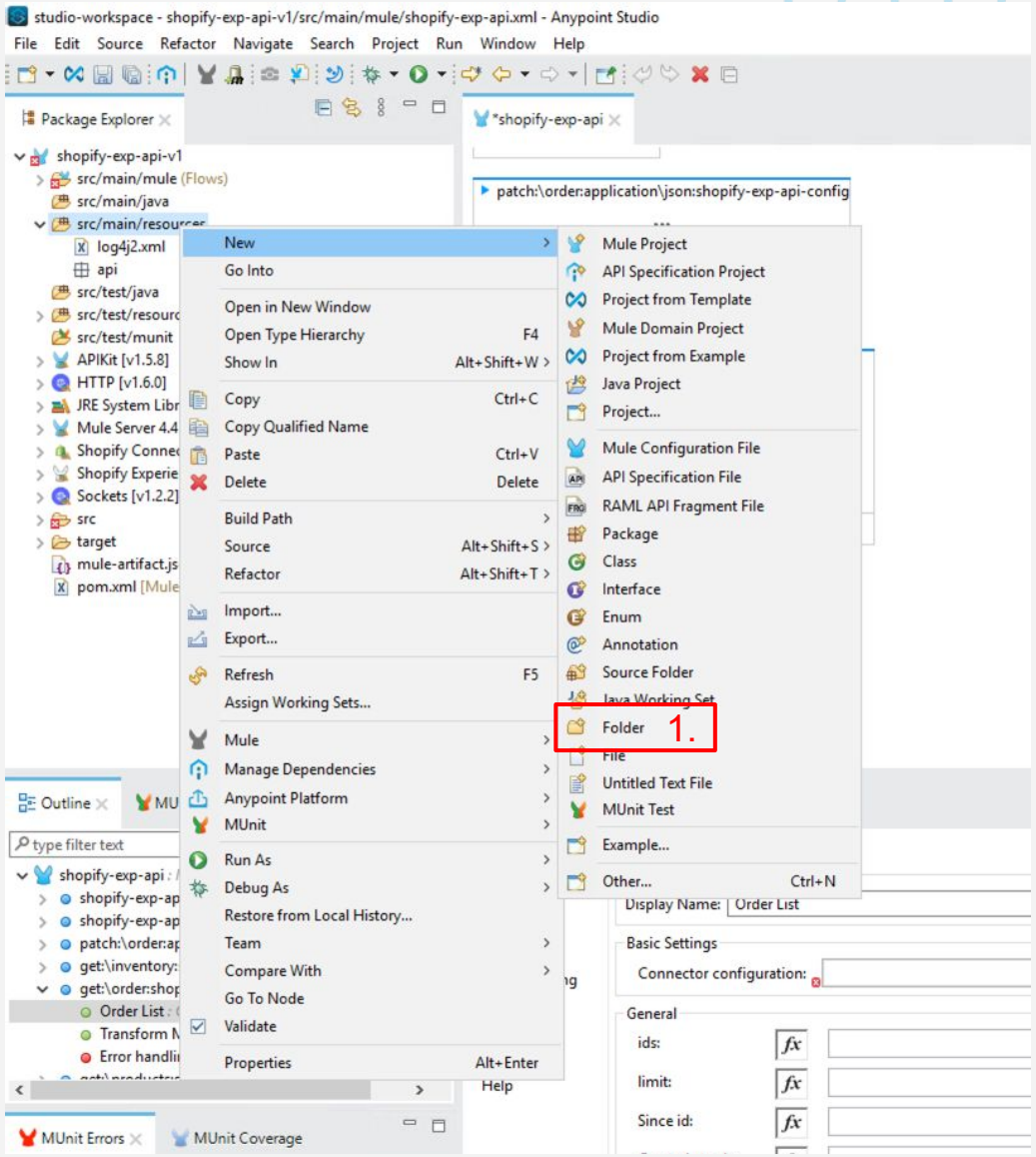


Module 2: Ubiquitous Connectivity: Lab 2 (9/23)

Step 4: Create a Property File

We could enter our Shopify configuration details directly in the below area. But it's a good practice to create a configuration file to hold all the parameterized properties in the project. We are going to create one and configure the connection credentials for our Shopify API implementation. First, we are going to create a folder and then create the config file.

1. In the left pane labeled **Package Explore**, Click on **src/main/resources** in the **Package Explorer** view, right click and create a **New** → **Folder**, name it **configuration** and click **Finish**



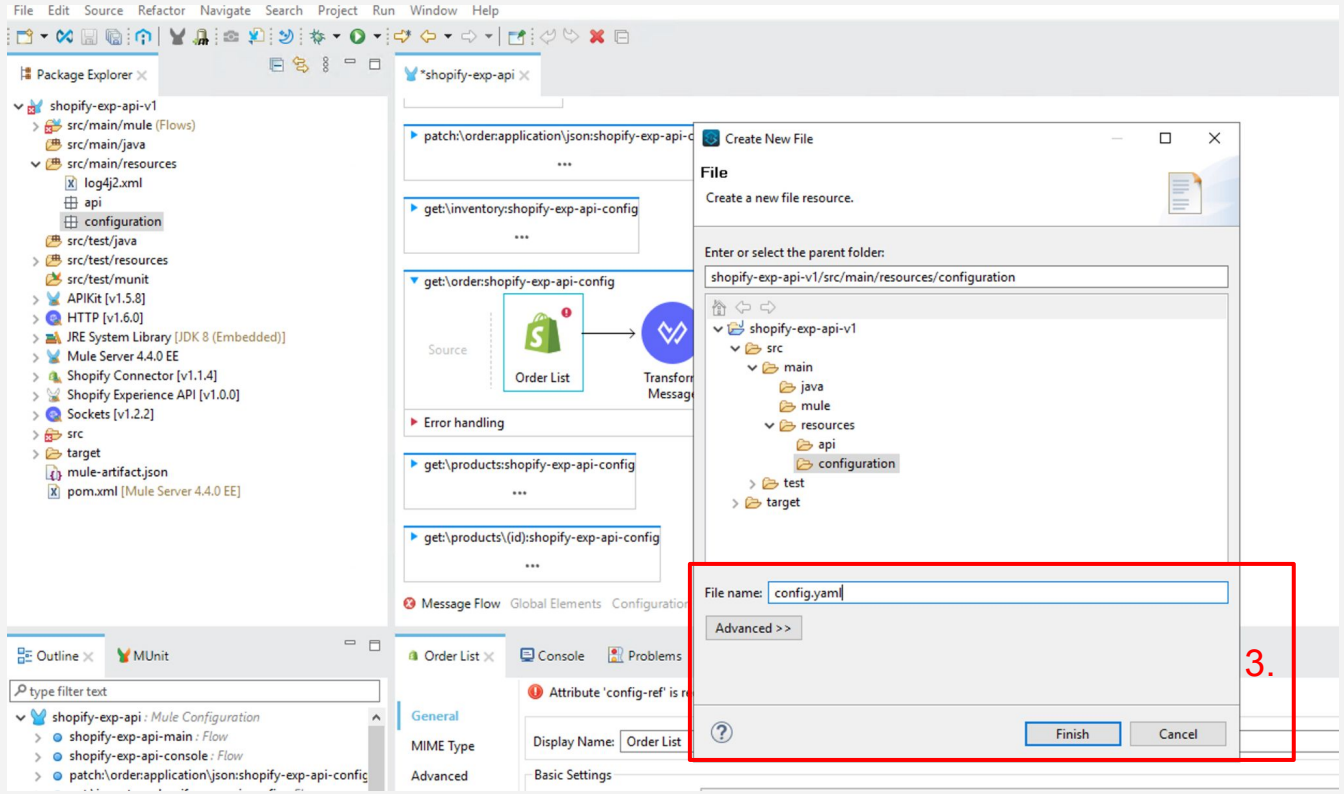
Module 2: Ubiquitous Connectivity: Lab 2 (10/23)

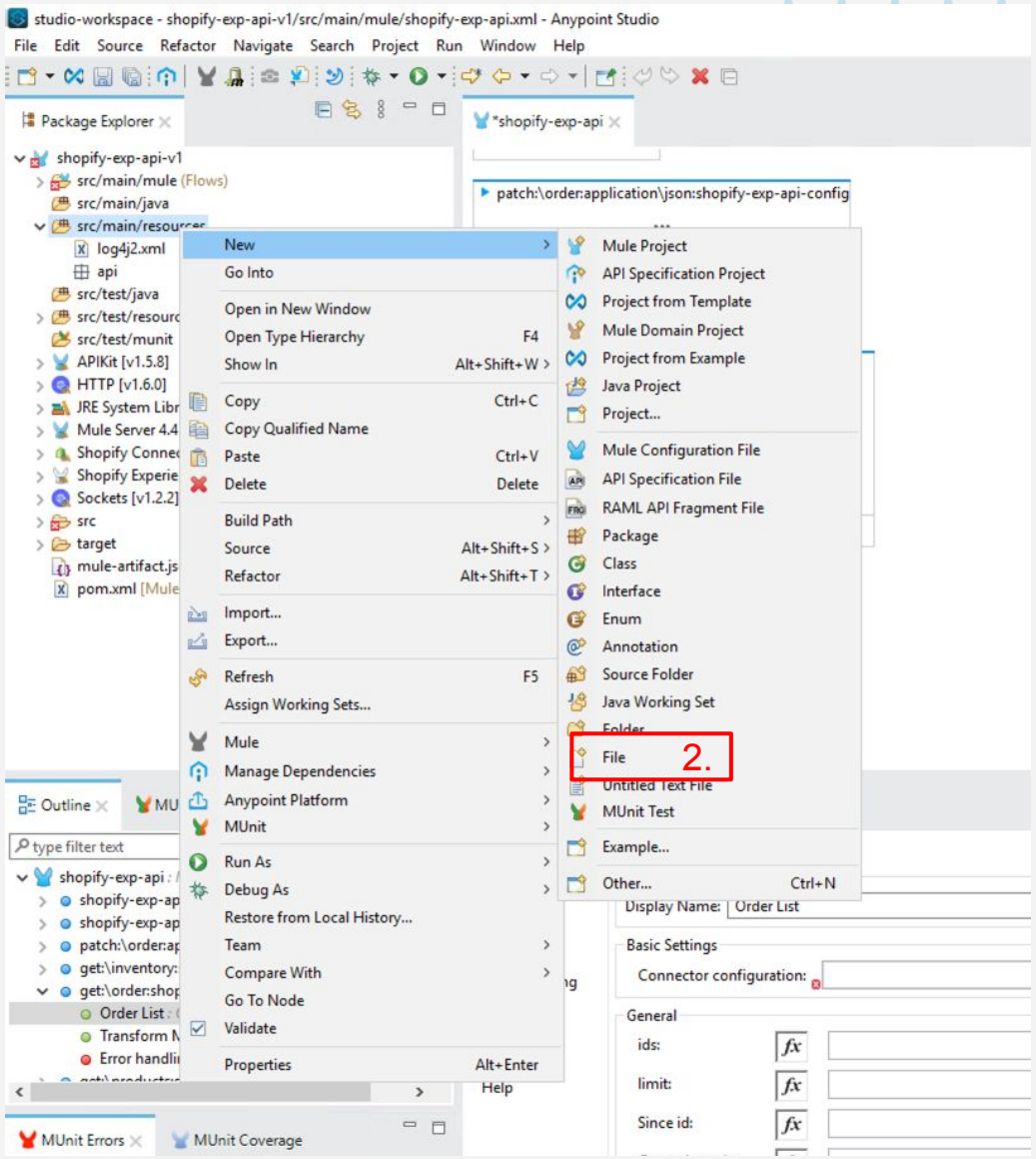
MuleSoft

Step 4: Create a Property File

2. Click on **src/main/resources/configuration** in the **Package Explorer** view, right click and create a **New → File.**

3. Name it **config.yaml** and click **Finish**.





Module 2: Ubiquitous Connectivity: Lab 2 (11/23)

MuleSoft

Step 4: Create a Property File

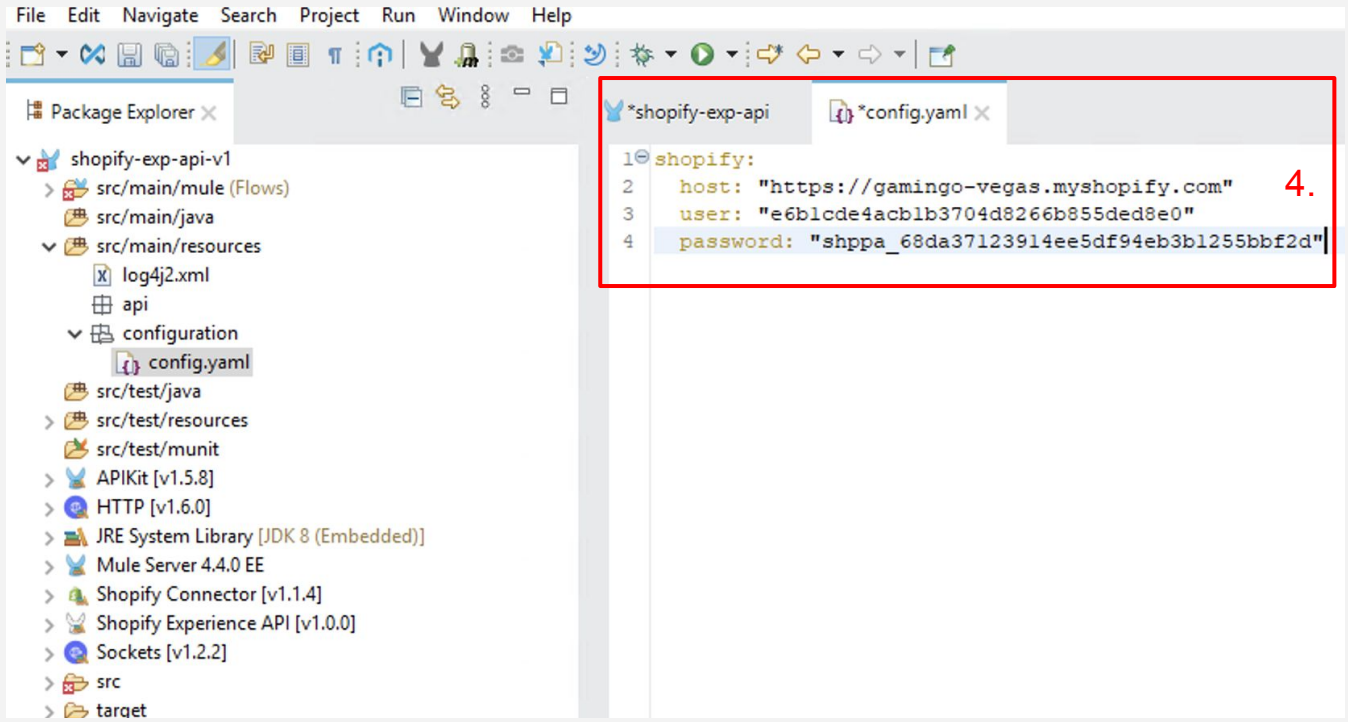
4. Copy the following contents into **config.yaml**. These are the connection credentials your API will use for the Shopify connector.

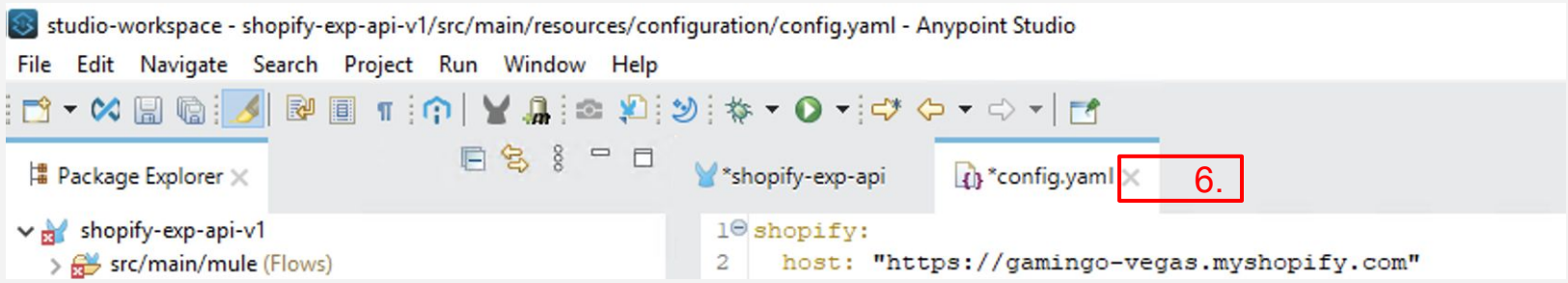
```
shopify:
  host: "https://gamingo-vegas.myshopify.com"
  user: "e6b1cde4acb1b3704d8266b855ded8e0"
  password: "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"
```

5. Click the **Save All icon** on the top Navigation Menu.

6. Close **config.yaml** file tab.

If you have problems copying the code you can find example here

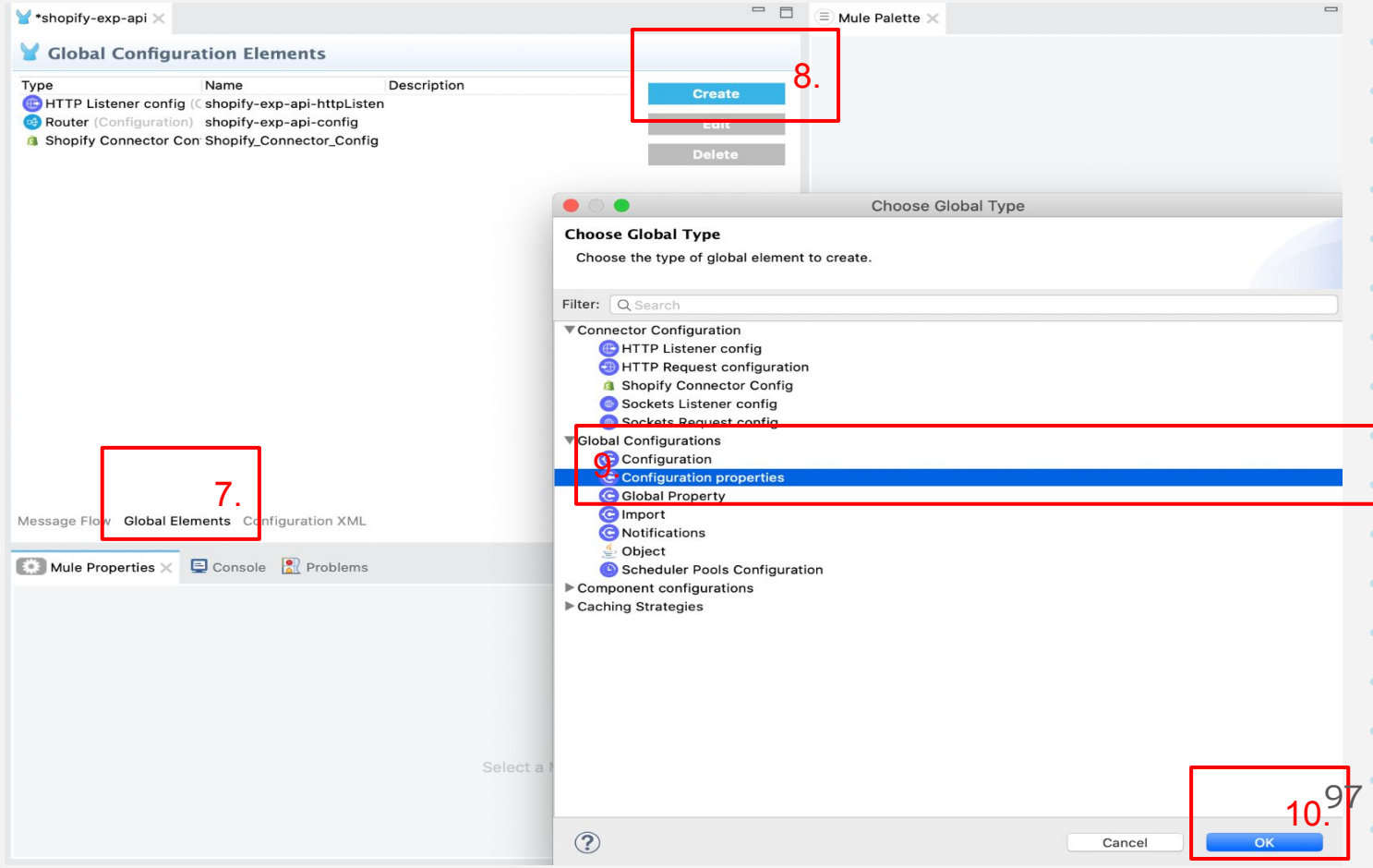




Module 2: Ubiquitous Connectivity: Lab 2 (12/23)

Step 4: Create a Property File

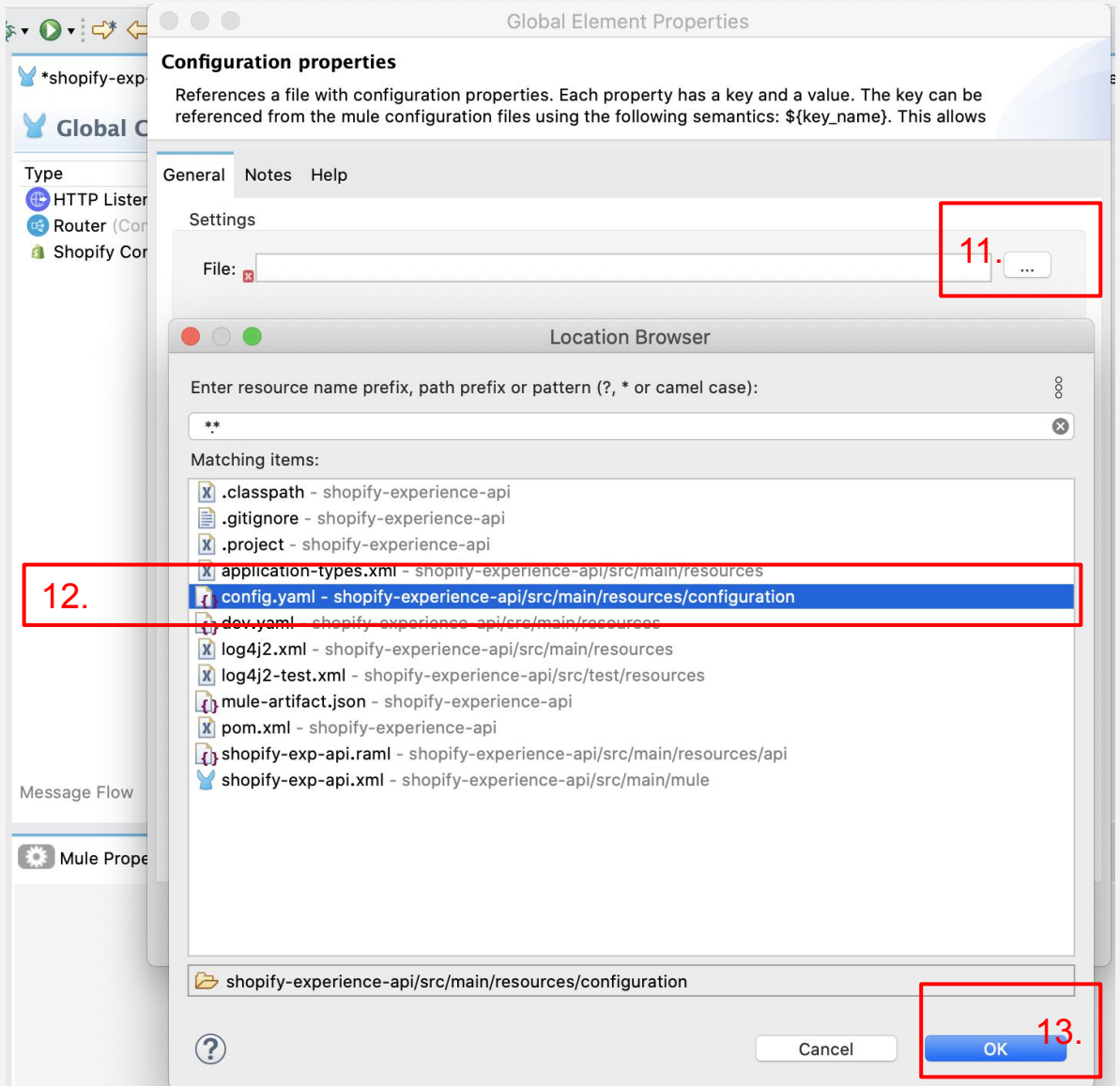
7. To finish the configuration of the properties file for your API, click on the **shopify-exp-api** view and click on the **Global Elements** tab at the bottom of the pane.
8. Click the **Create** Option.
9. Go to Global **Configurations> Configuration Properties.**
10. Click **OK.**



Module 2: Ubiquitous Connectivity: Lab 2 (13/23)

Step 4: Create a Property File

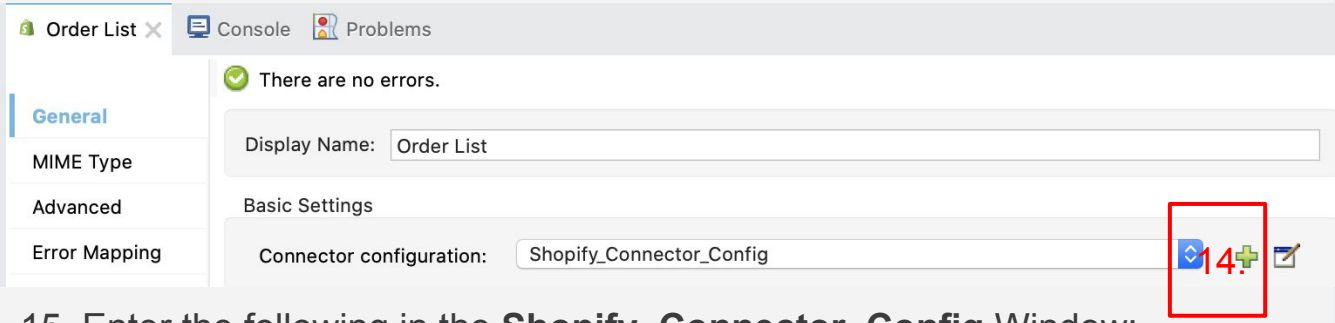
11. By **File** Click the **…**
12. Select the **config.yaml** file you just created
13. Click **OK.**



Module 2: Ubiquitous Connectivity: Lab 2 (13/23)

Step 4: Create a Property File

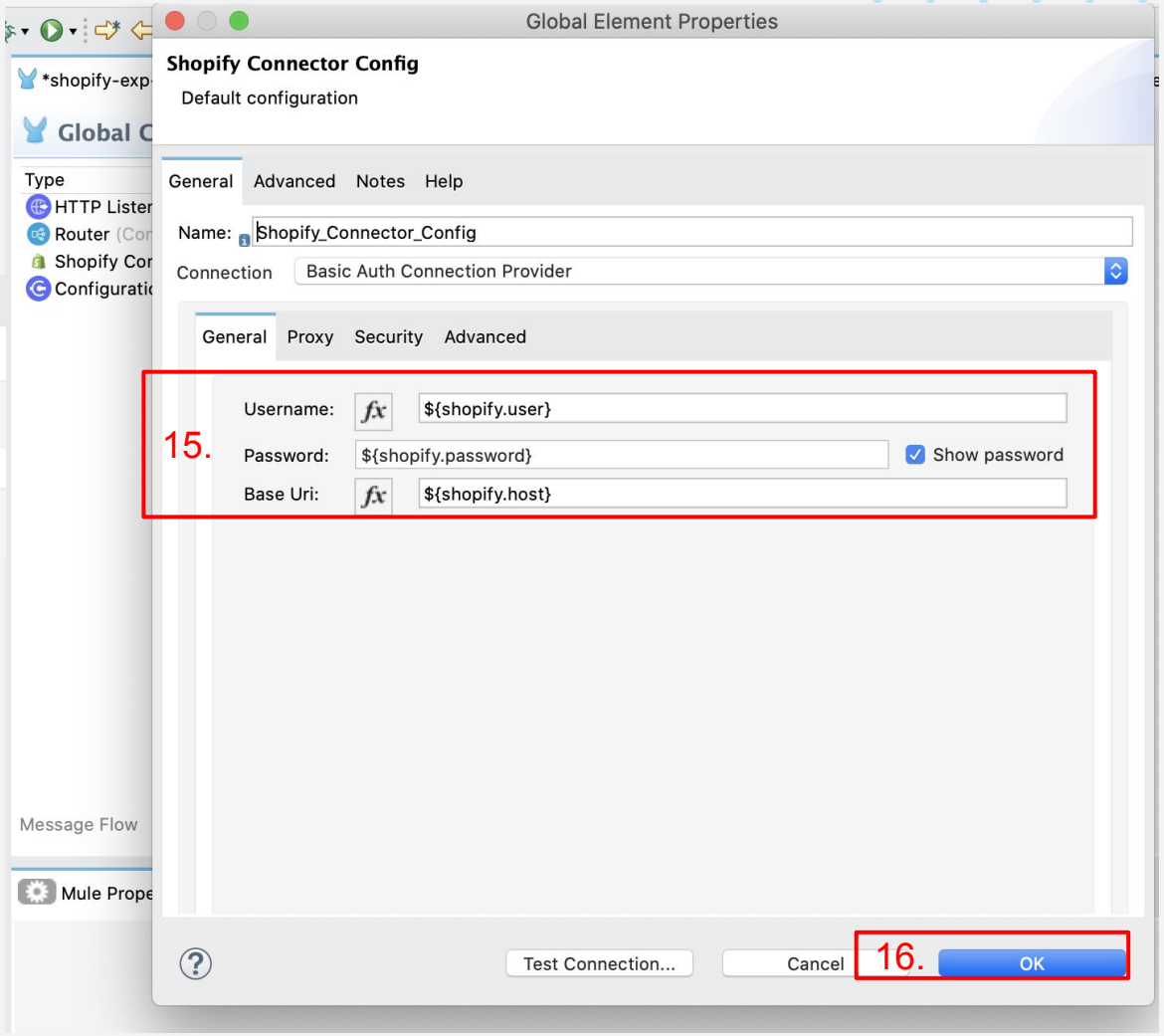
14. Navigate down to the lower configuration panel and click on the green "+" by **Connector configuration: Shopify_Connector_Config**



15. Enter the following in the **Shopify_Connector_Config** Window:

- **Username: ${shopify.user}**
- **Password: ${shopify.password}**
- **Base Uri: ${shopify.host}**

16. Click **OK.**



Module 2: Ubiquitous Connectivity: Lab 2 (14/23)

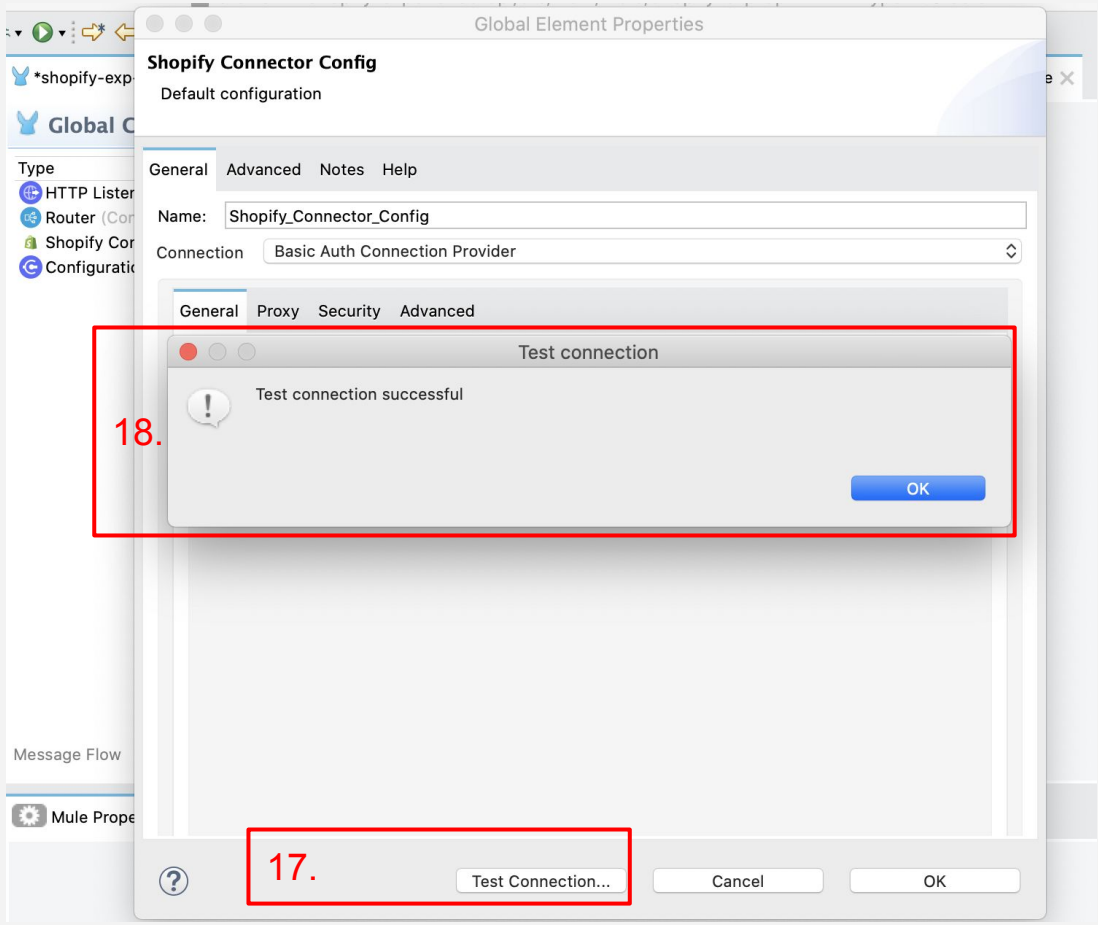
Step 4: Create a Property File

17. Click **Test Connection...** Anypoint Studio will now attempt connecting to the NTO Shopify. **There are several different was to authenticate to Shopify, but for the sake of the workshop we are using this Basic Authentication method.**

18. You should receive a "**Test connection successful"** Message. Press **OK** to Close the message.

19. Click OK again in the **Shopify_Connector_Config** Window.

Now our Shopify API is connected to **NTO's Shopify Account**!



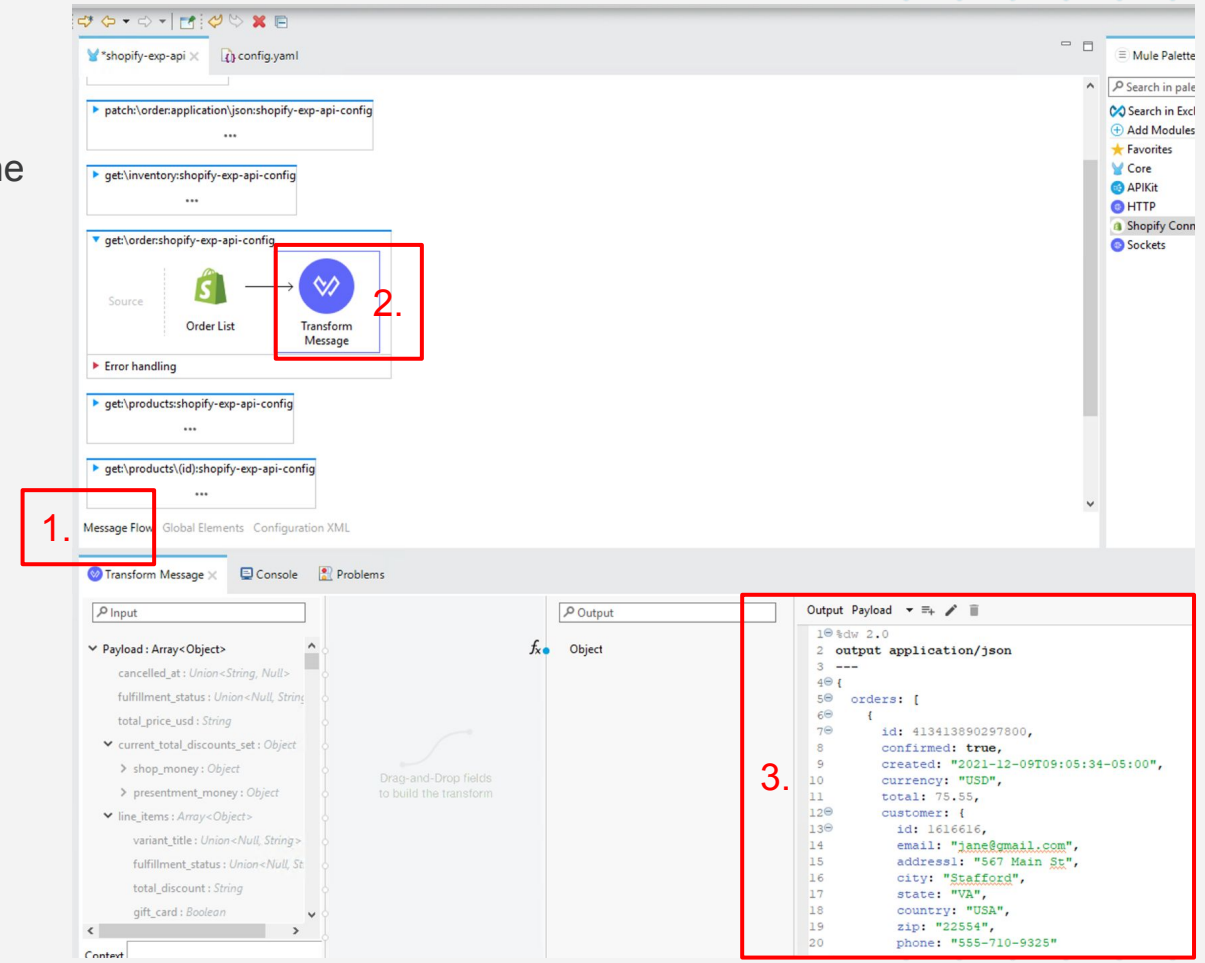
Module 2: Ubiquitous Connectivity: Lab 2 (15/23)

Step 5: Transform Message

1. Click **Message Flow** tab to switch out of the Global Element Pane back to the Message Flow pane where we need to update our **get:\order:shopify-exp-api-config** flow.

2. Move your mouse to the Transform Message in the **get:\order:shopify-exp-api-config** flow and double click it.

3. In the below **Transform Message** pane you will see some code in the right pane called **Output Payload**. This is example response body that we created when we designed the **Shopify API** in **Module 1**.



Module 2: Ubiquitous Connectivity: Lab 2 (16/23)

Step 5: Transform Message

1. Delete the sample code under the Output Payload pane and replace with the following:

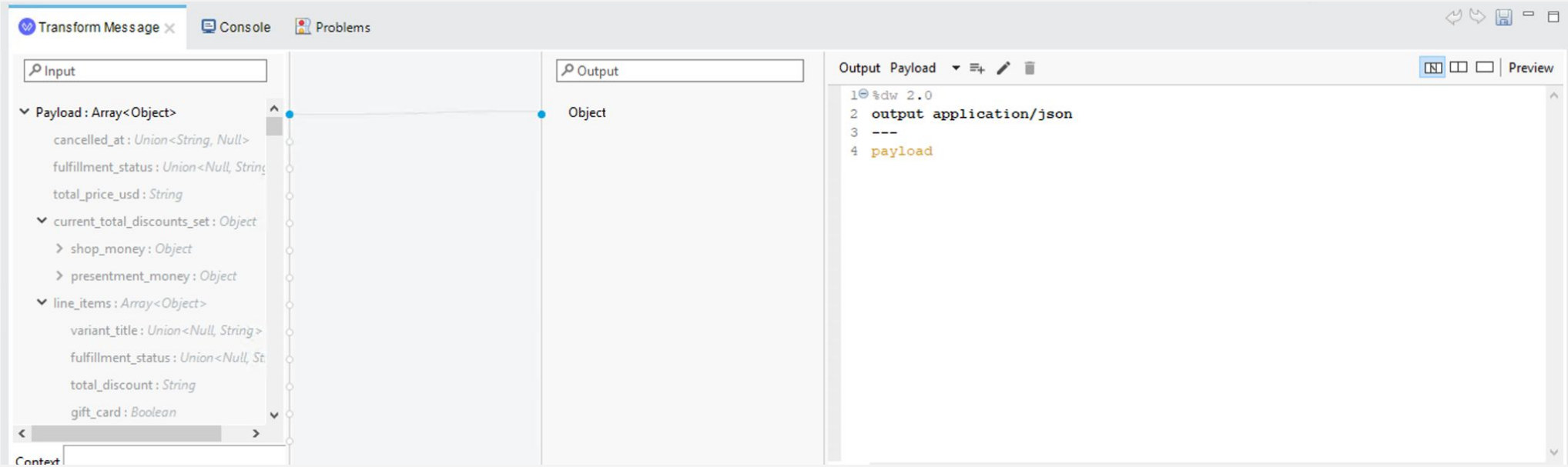
.

```
%dw 2.0
output application/json
---
payload
```

2. Click the **Save icon** at the top right corner of the **Transform Message** Pane. To save you changes.

We have now made the necessary changes to the **get:\order:shopify-exp-api-config** flow and we are now ready to test it!

.



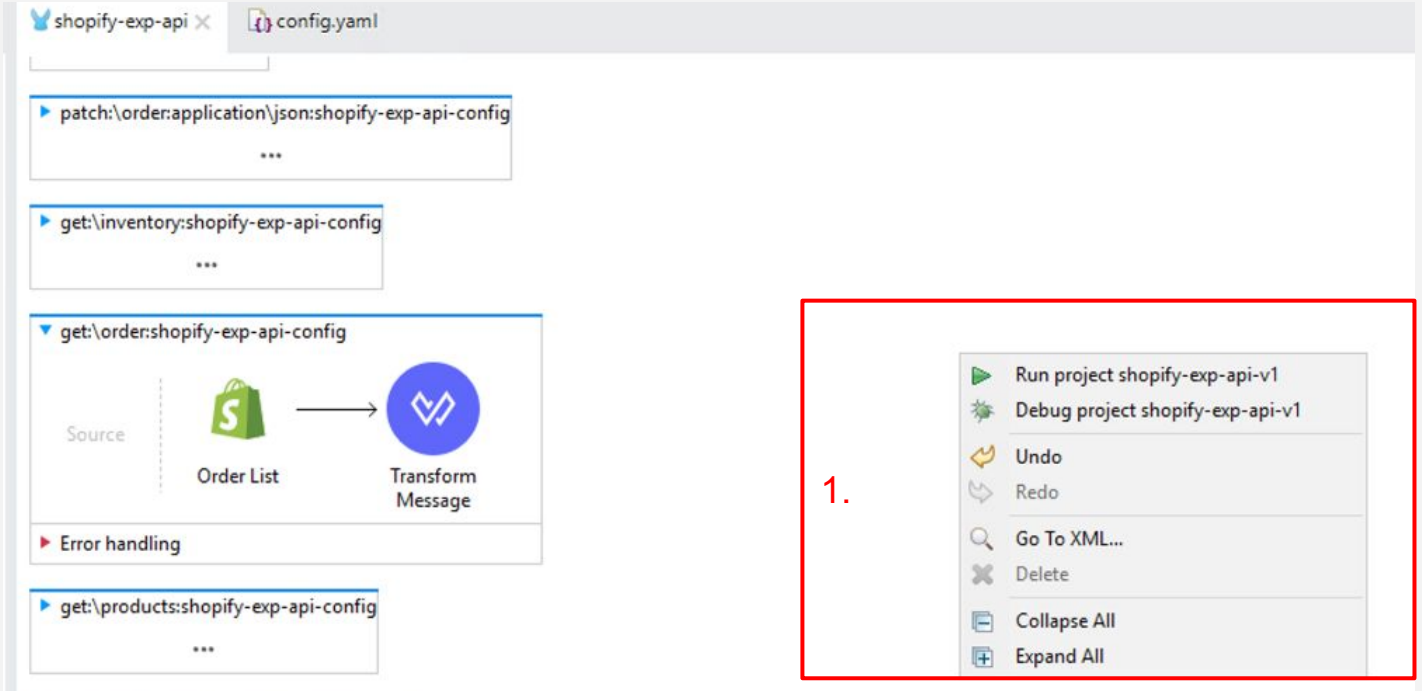
Module 2: Ubiquitous Connectivity: Lab 2 (17/23)

Step 6: Testing the Shopify API Connectivity to NTO's Shopify

In order to test the API we will need to run our project on our local machine.

1. To run the project all you need to do is Right Click in any blank area in the **shopify-exp-api** Message Flow and Select **Run project shopify-experience-api.**

.



Module 2: Ubiquitous Connectivity: Lab 2 (18/23)

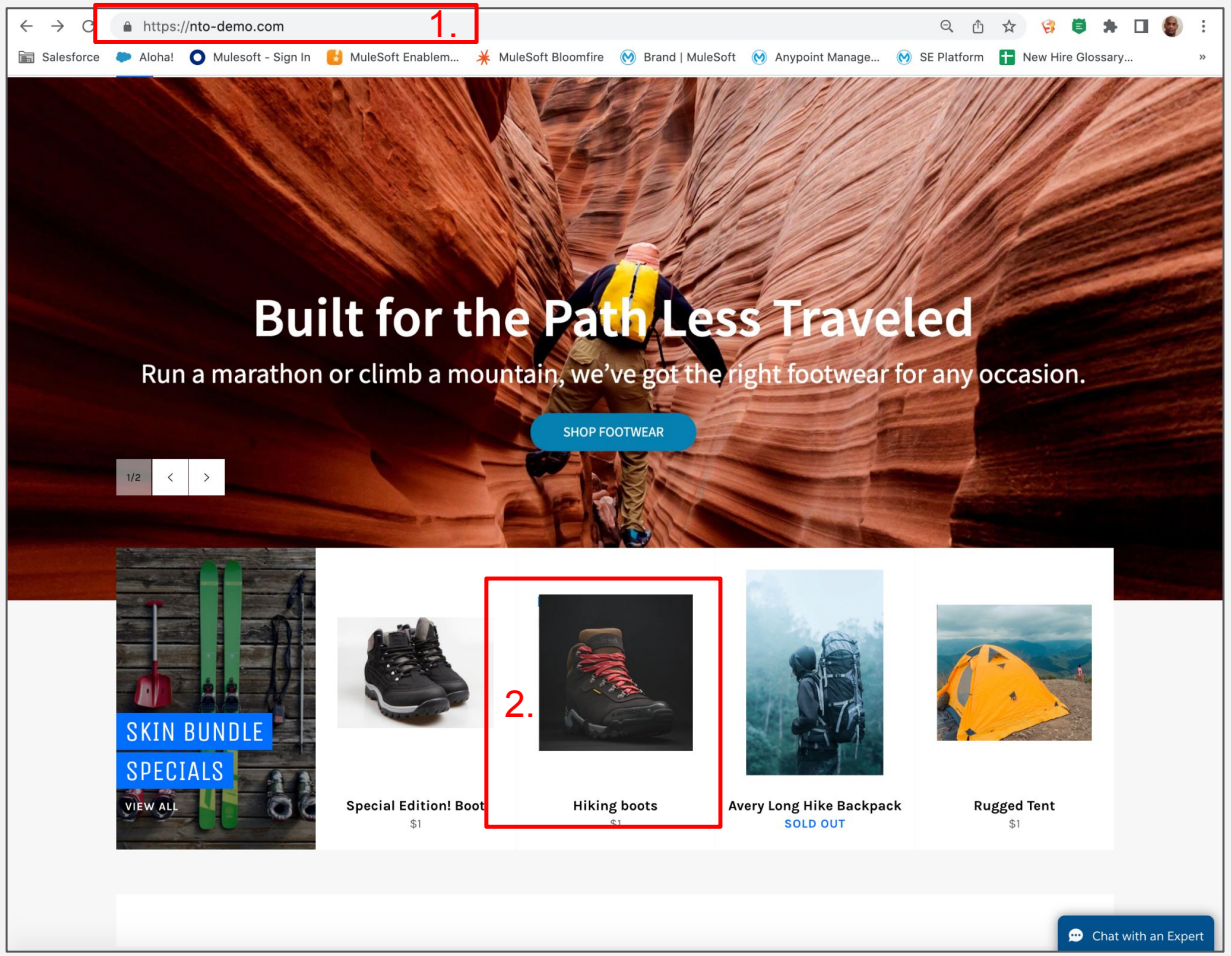
Step 6: Testing the Shopify API Connectivity to NTO's Shopify

While we wait for the project to run let's add our own Shopify order date in the **NTO Shopify Site**!

1. Go to https://nto-demo.com/

2. Let's buy the "**Hiking boots",** by selecting that item.

.



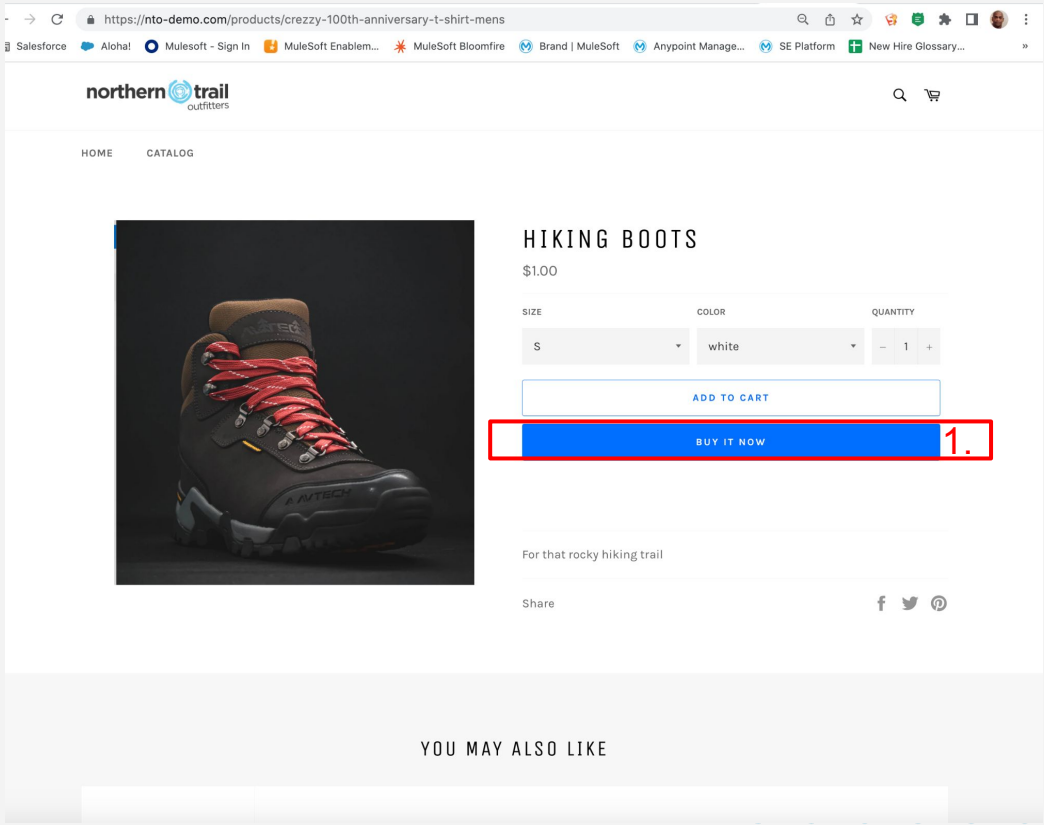
Module 2: Ubiquitous Connectivity: Lab 2 (18/23)

Step 7: Testing the Shopify API Create Order

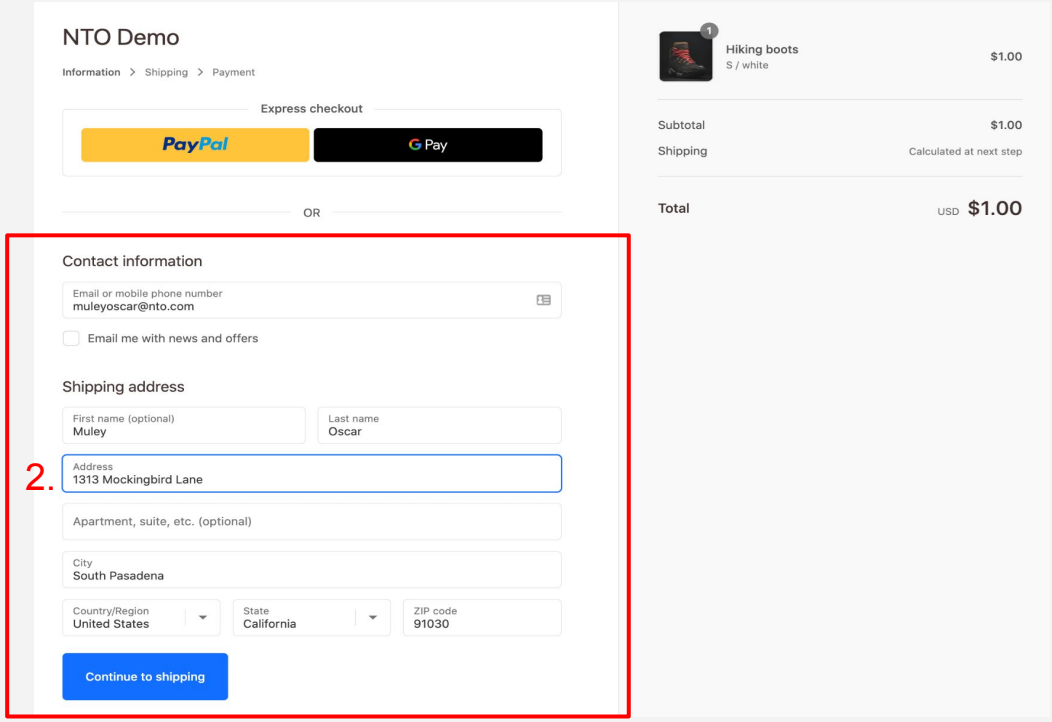
1. Click **Buy It Now**

2. Add some fake contact details like so and Click **Continue to Shipping**.

Keep record of the fake email you used it will come in handy in subsequent steps.



.



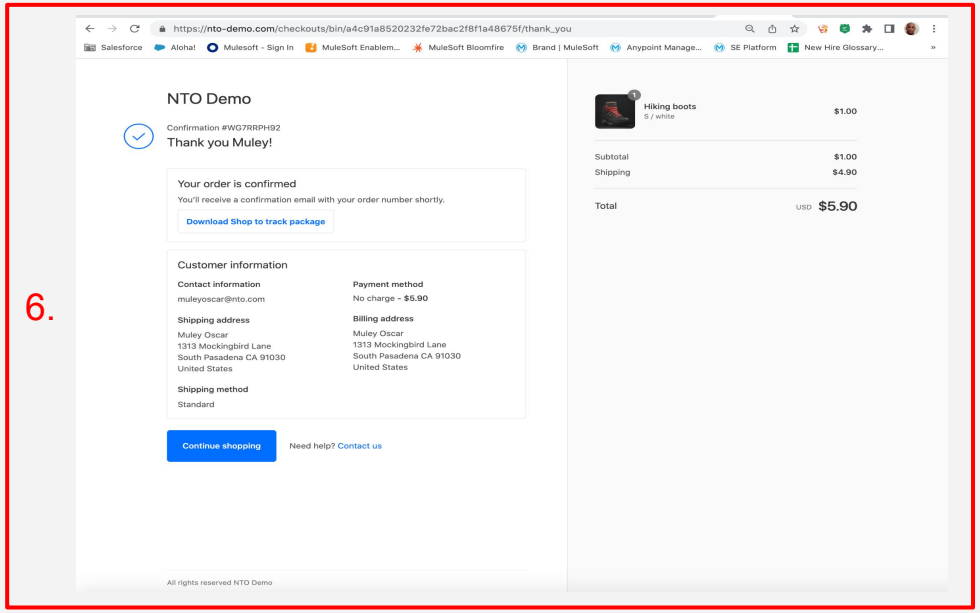
DO NOT USE YOURS OR ANYONE'S PERSONAL INFORMATION!!! MAKE SOMETHING UP.

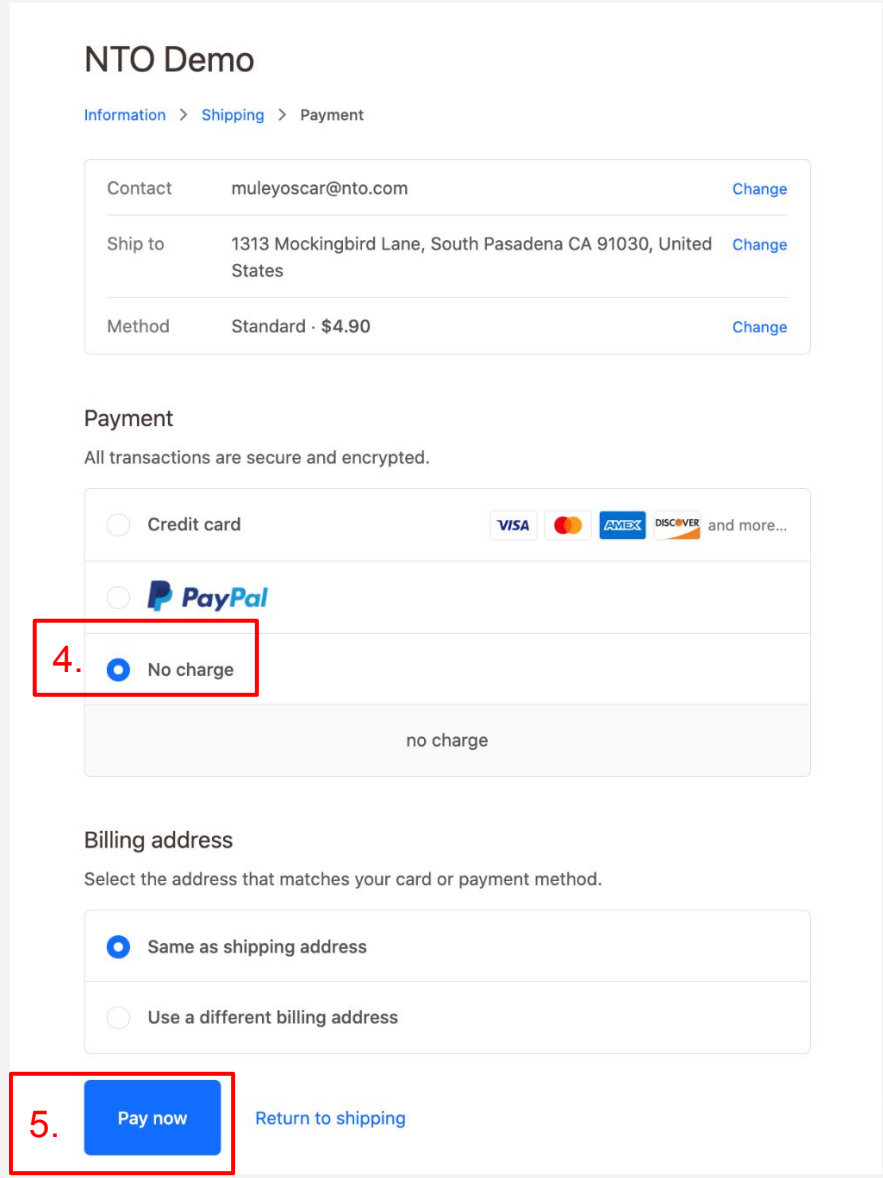
Module 2: Ubiquitous Connectivity: Lab 2 (19/23)

Step 7: Testing the Shopify API Create Order

4. Select the **No Charge** Radio Button.
5. Click **Pay Now**.
6. You should see an Order Confirmation. Close out of https://nto-demo.com/.

.





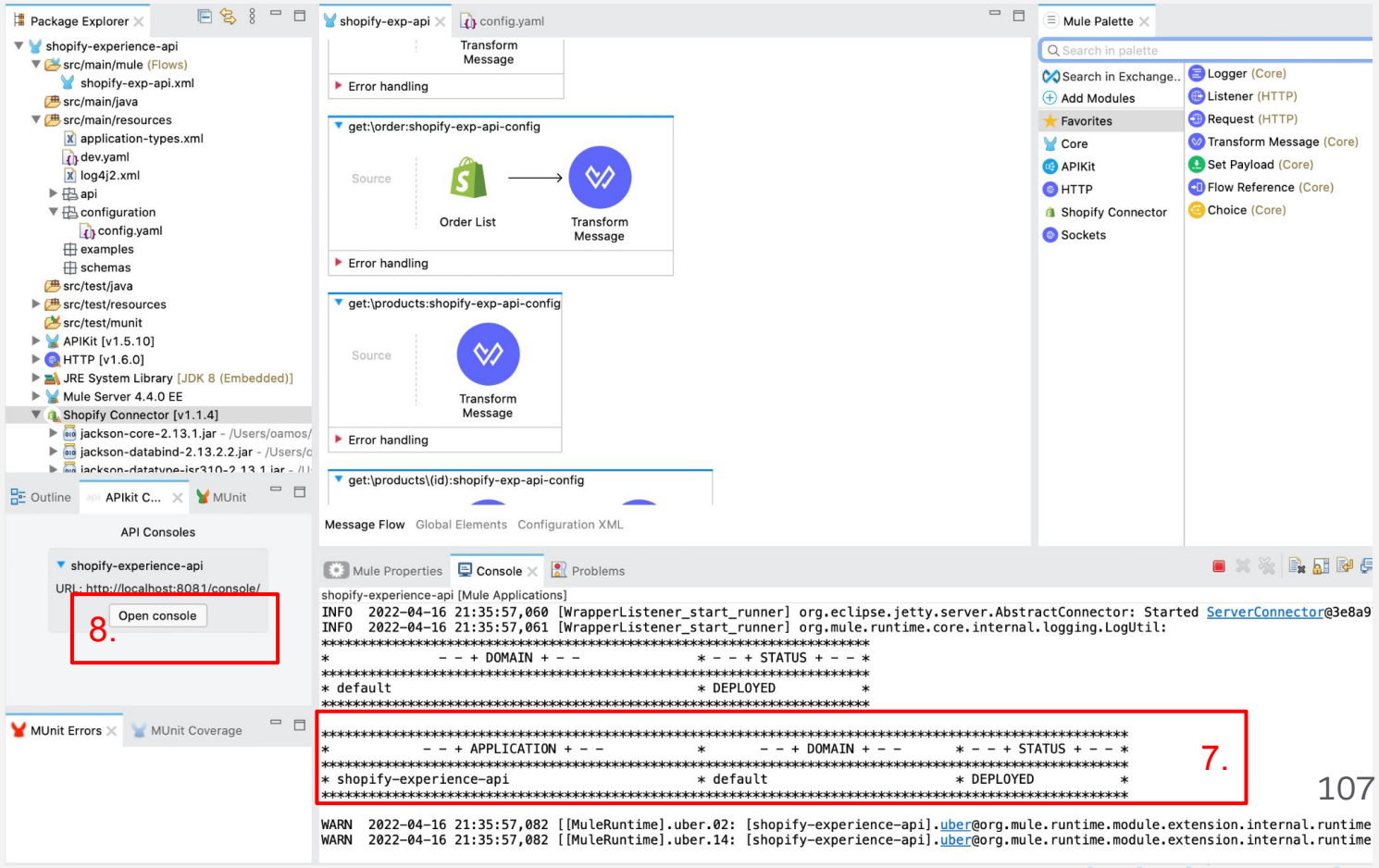
Module 2: Ubiquitous Connectivity: Lab 2 (20/23)

Step 7: Testing the Shopify API Create Order

7. Switch back to **Anypoint Studio** and by now your project should be in a **DEPLOYED** status

8. Click on **Open console** in the left lower pane labeled **API Consoles**

.

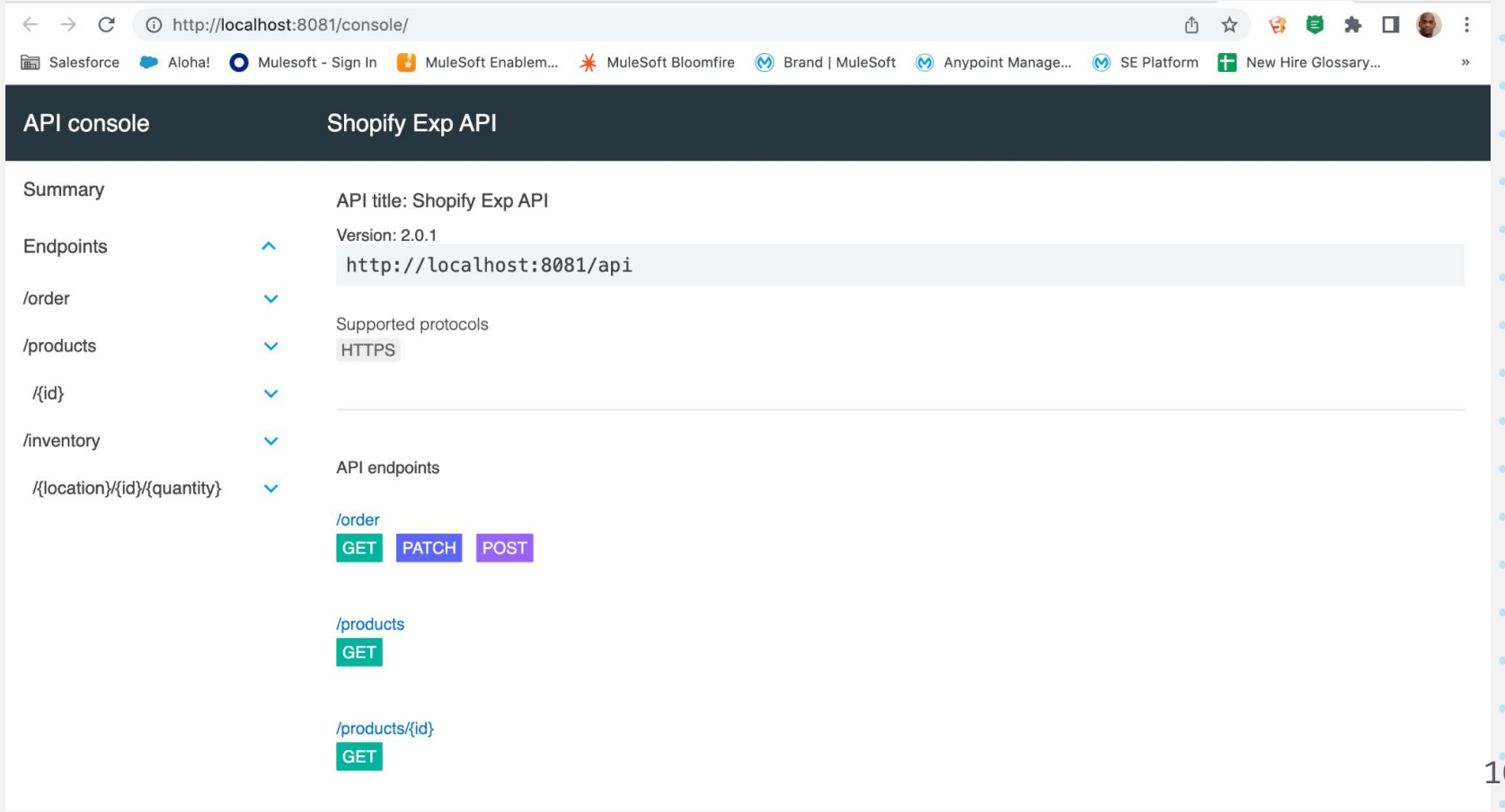


Module 2: Ubiquitous Connectivity: Lab 2 (21/23)

Step 7: Testing the Shopify API Create Order

9. A Web Browser tab will get launch to show the **API Console** Web UI. Click **/order GET** button to test our flow out.

.

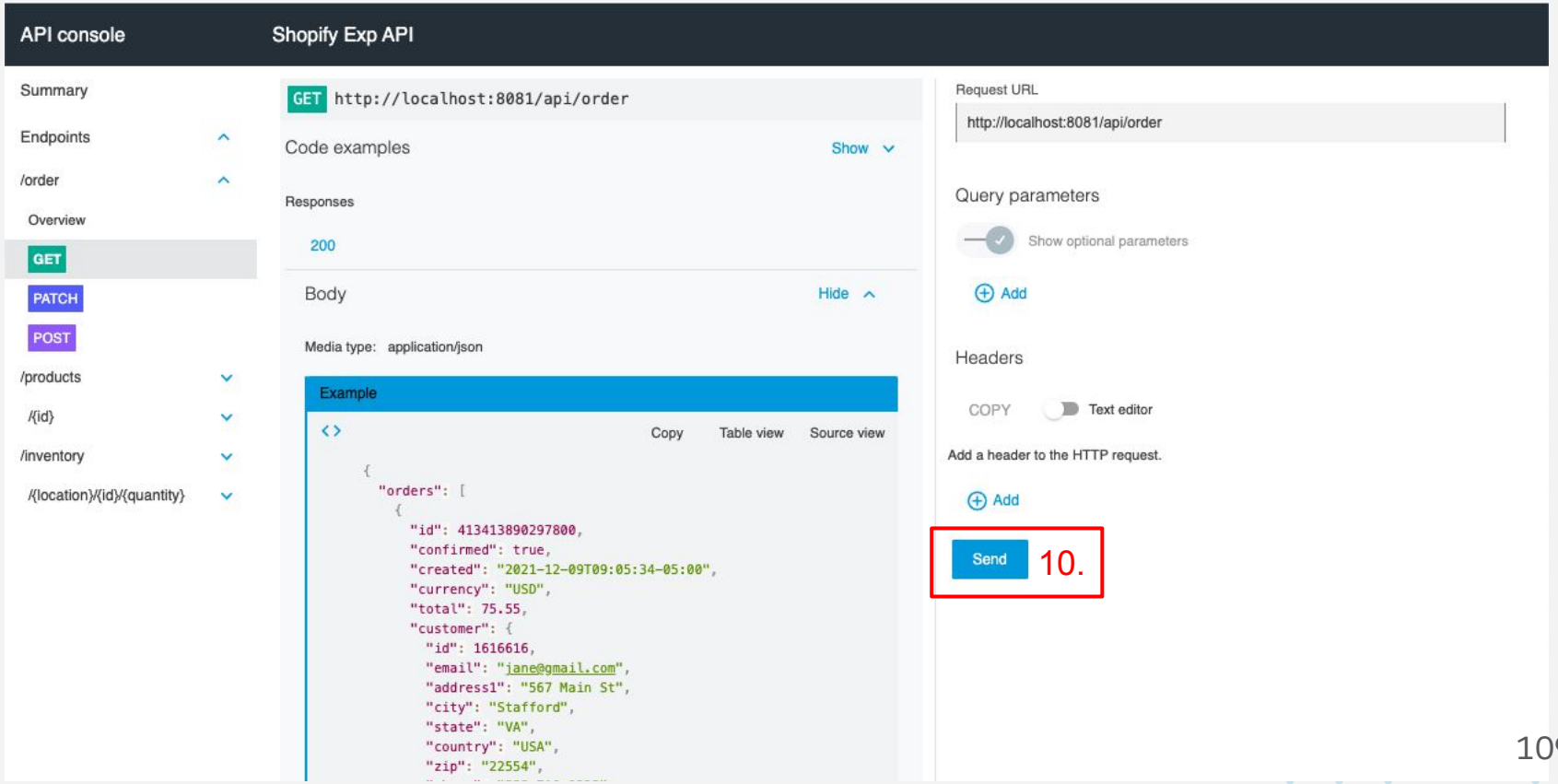


Module 2: Ubiquitous Connectivity: Lab 2 (22/23)

Step 7: Testing the Shopify API Create Order

10. You will be taken to a view for the **GET /order** resource endpoint. Click the send button **Send**.

.



Module 2: Ubiquitous Connectivity: Lab 2 (23/23)

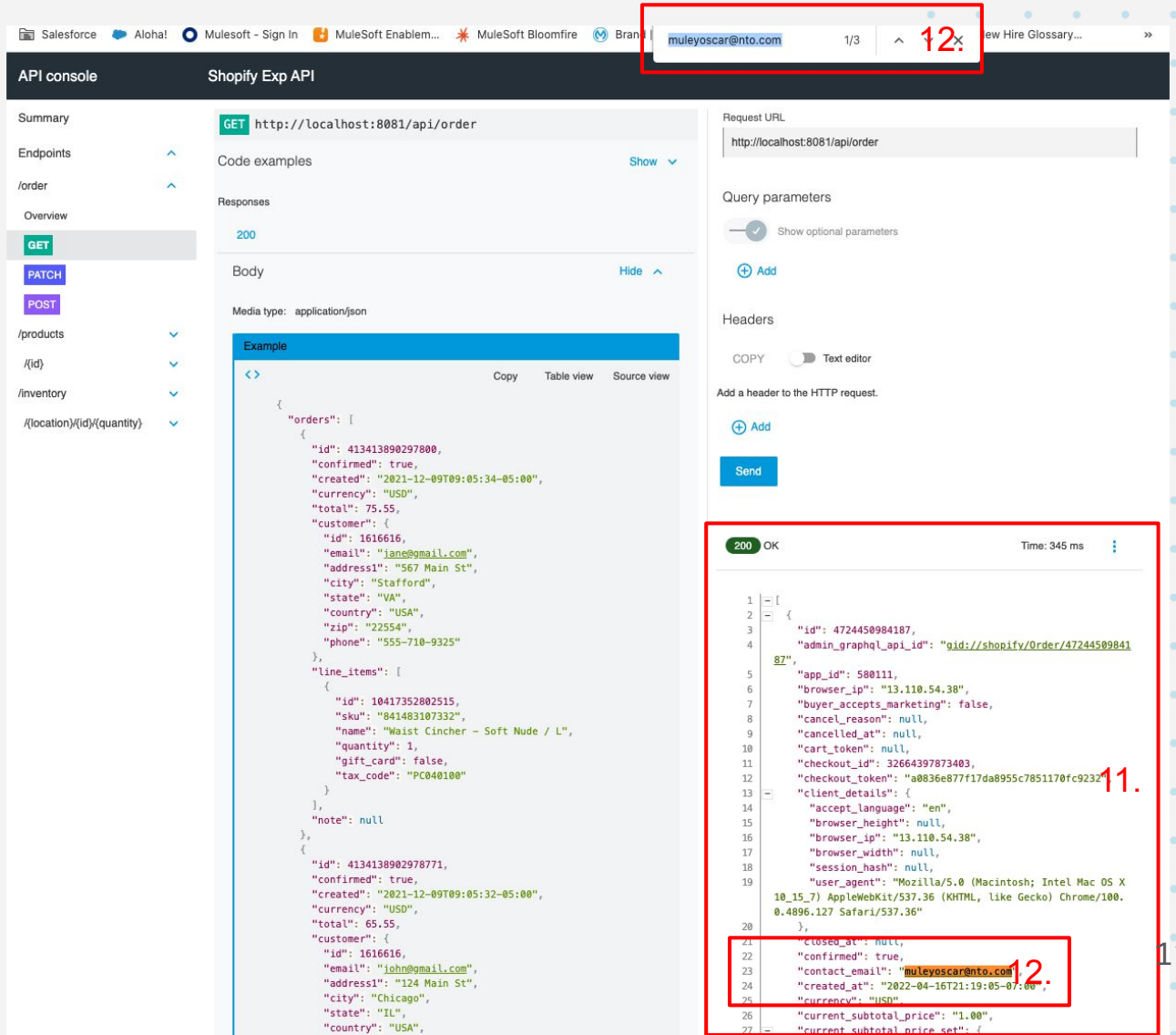
Step 7: Testing the Shopify API Create Order

11. Response will show in the lower right pane.with a large payload of an array of orders.

12. Do a **MAC: Command + F**, **PC: Ctrl +F**, and search for the fake email you used for your order.

The Shopify response is a very comprehensive payload take a look at the structure. **In future Lab updates we will show you how to make this payload tailored and more readable.**

Great work! We successfully tested our Implementation now lets deploy it.



Proceed to Lab 3

MuleSoft

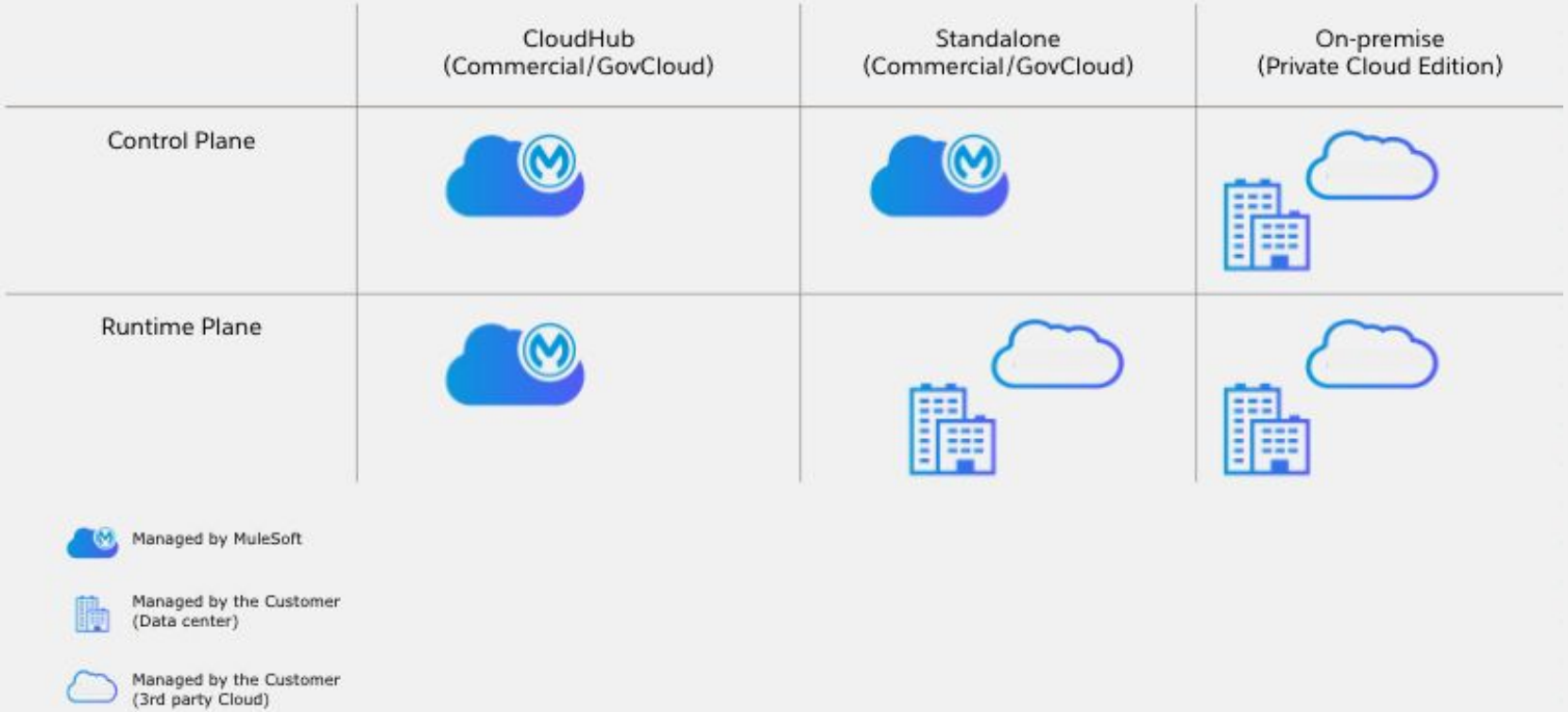
Module 2: Lab 3 Deploy the Shopify API Implementation

Source: DTC e-commerce: How consumer brands can get it right

Module 2: API Deployment : Overview

Overview

Anypoint projects can be deployed in a number of ways (via our Cloud, On Premise Data Center, 3rd Party (AWS, Google,Azure), or a mix of all. This workshop will show how deployment can be done in our Cloud managed offering, **CloudHub**.

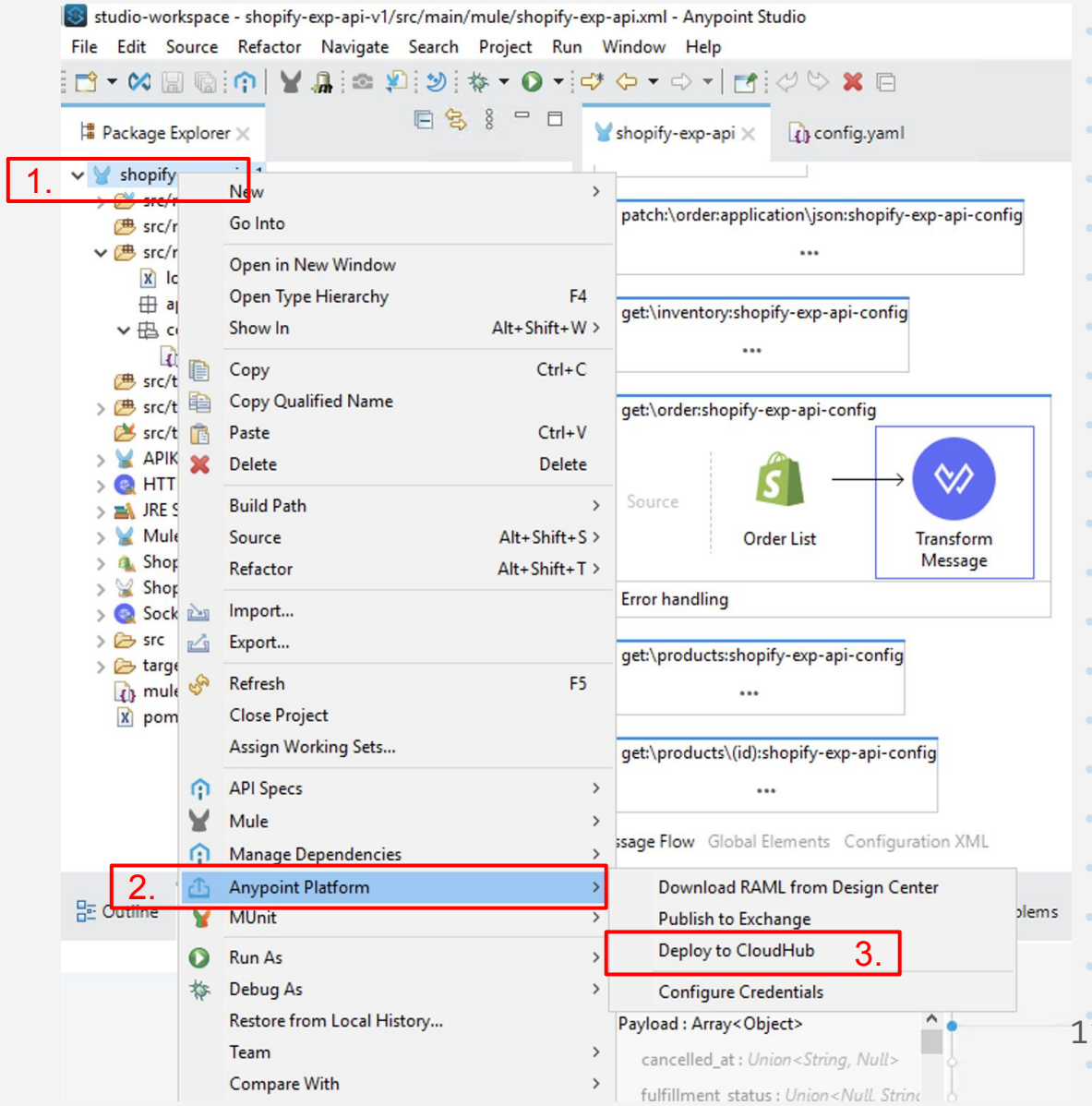


- **Footnote: In some of our more advanced workshops we also show complete CI/CD process of deployment. Reach out to you Account Executive if you like to learn more about MuleSoft's CI/CD capabilities.**

Module 2: Ubiquitous Connectivity: Lab 3 (2/10)

Step 1: Deploy Shopify Project from Anypoint Studio

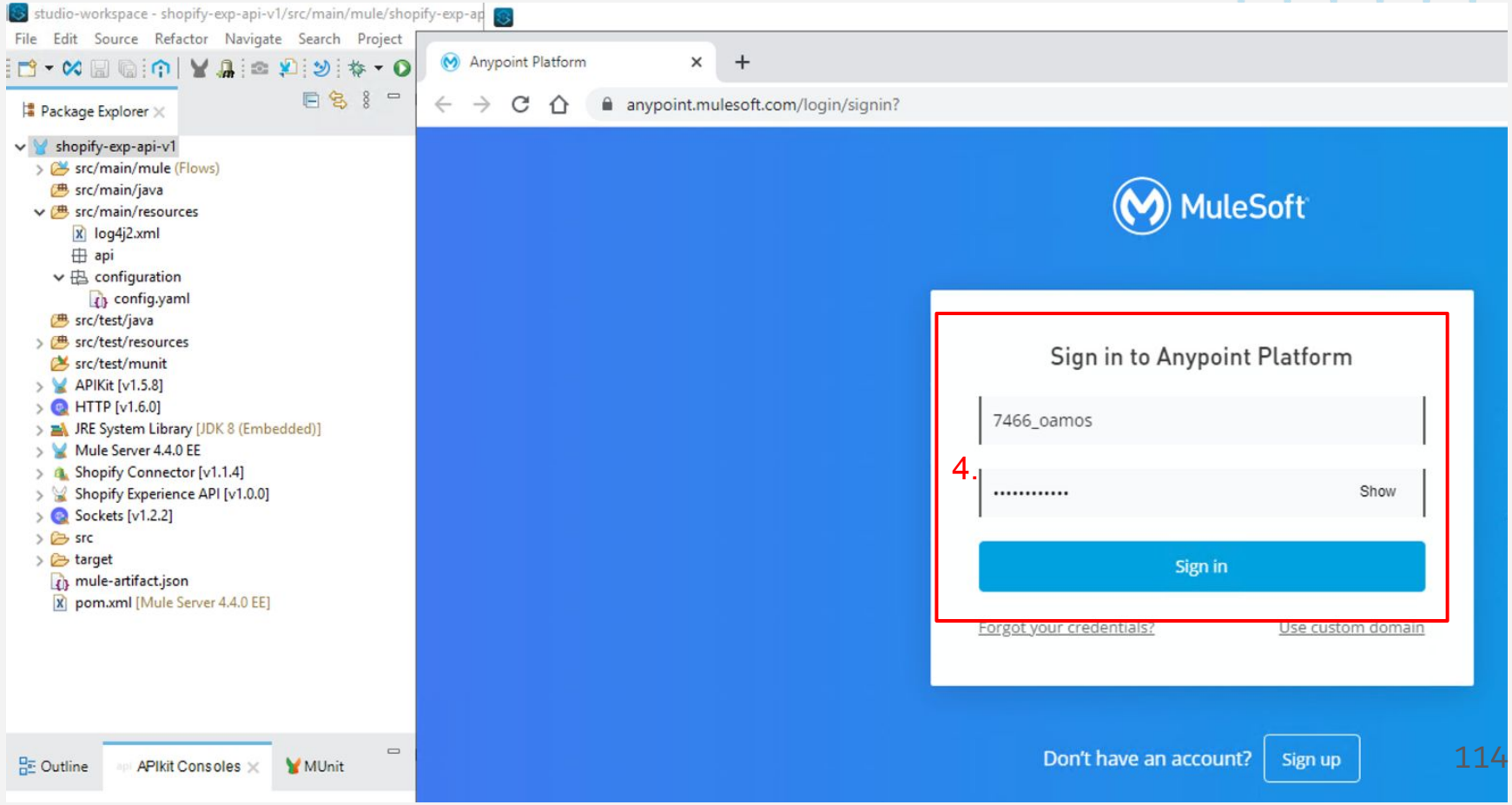
1. With your mouse select the **shopify-experience-api** under Package Explorer in the left pane and **Right Mouse Click.**
2. Menu option should appear scroll down and select **Anypoint Platform.**
3. Sub menu will appear to the right. Select **Deploy to Cloud.**



Module 2: Ubiquitous Connectivity: Lab 3 (3/10)

Step 1: Deploy Shopify Project from Anypoint Studio

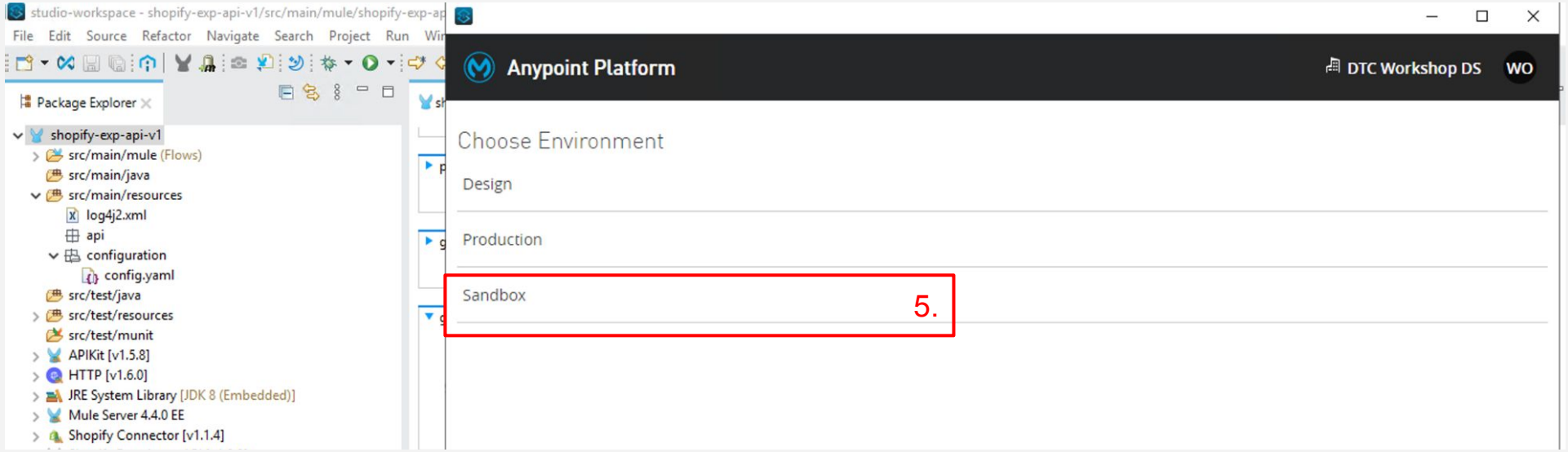
4. Anypoint login window will appear. Enter your Anypoint credentials and click **Sign In.**



Module 2: Ubiquitous Connectivity: Lab 3 (4/10)

Step 1: Deploy Shopify Project from Anypoint Studio

5. Under **Choose Environment** Select **Sandbox**.

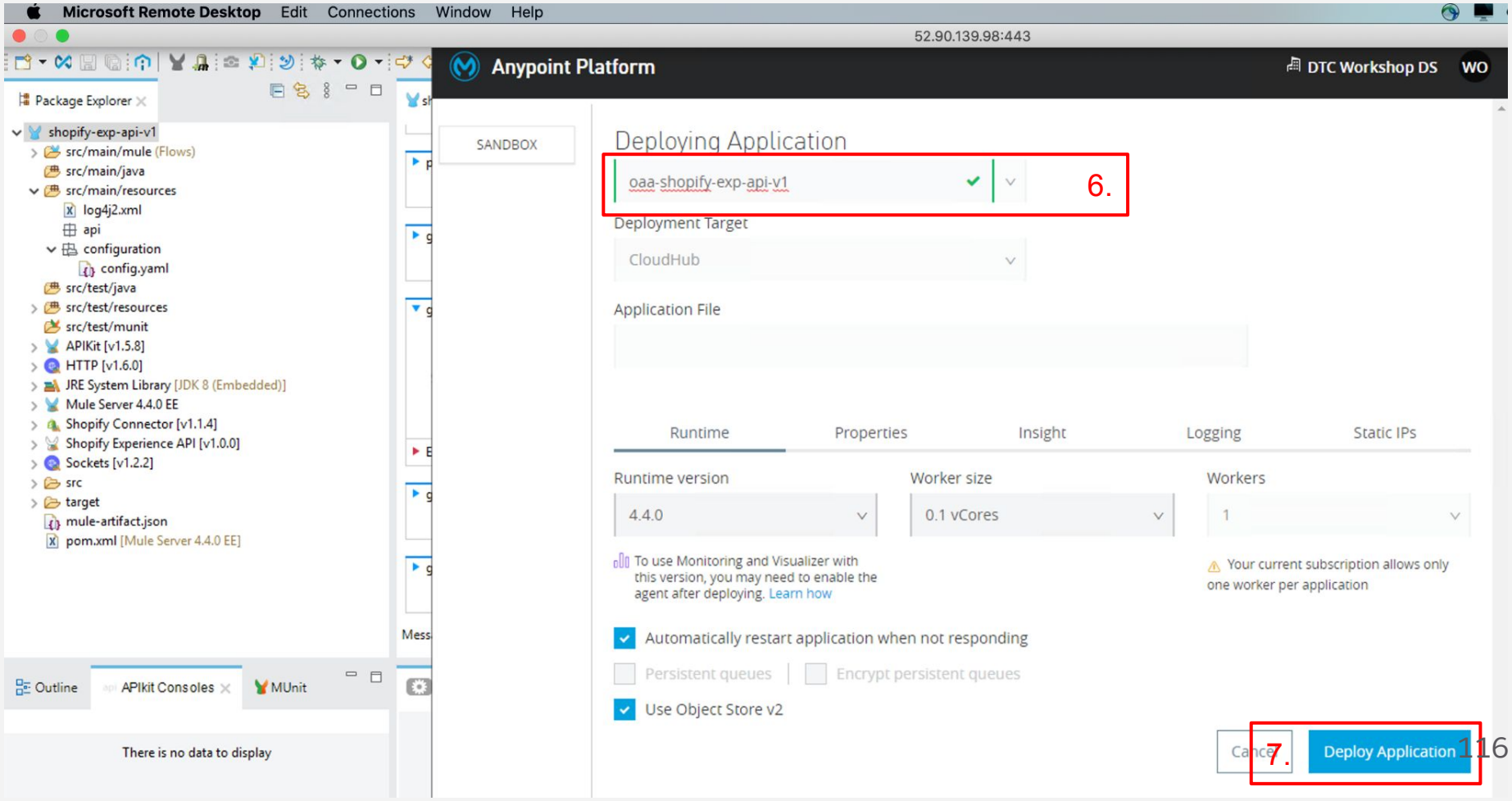


Module 2: Ubiquitous Connectivity: Lab 3 (5/10)

Step 1: Deploy Shopify Project from Anypoint Studio

6. Under Deployment Application name you deployment "**<first name or initials>-shopify-exp-api-v1"**.

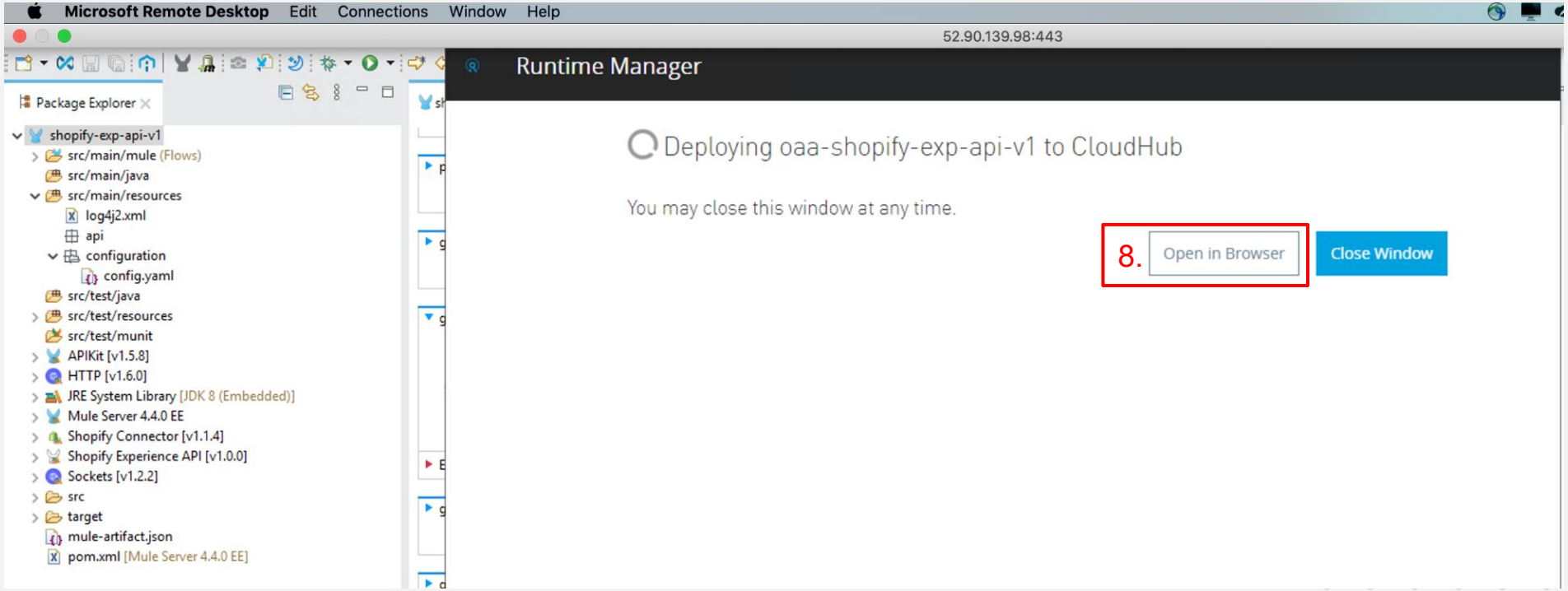
7. Click **Deploy Application.**



Module 2: Ubiquitous Connectivity: Lab 3 (6/10)

Step 1: Deploy Shopify Project from Anypoint Studio

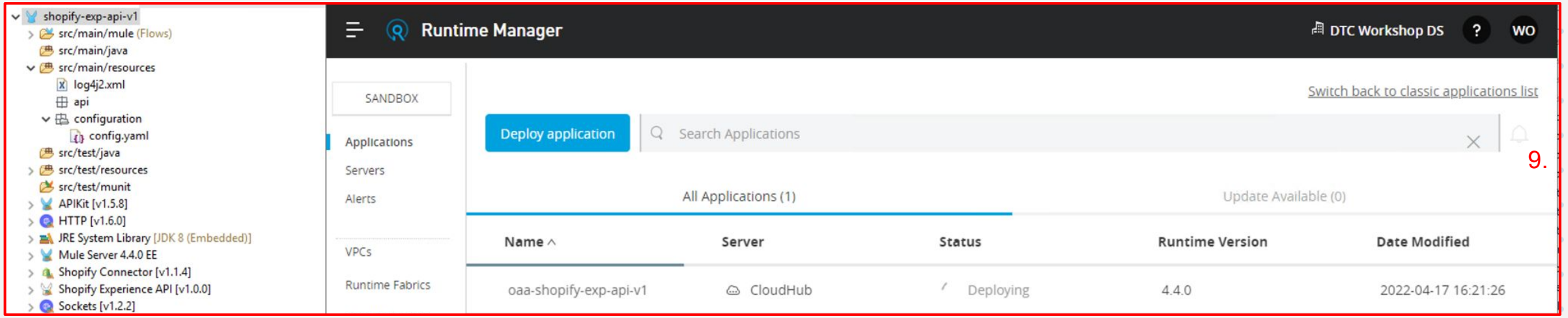
8. Screen will show that your application is **Deploying**. Click **Open in Browser.**



Module 2: Ubiquitous Connectivity: Lab 3 (7/10)

Step 1: Deploy Shopify Project from Anypoint Studio

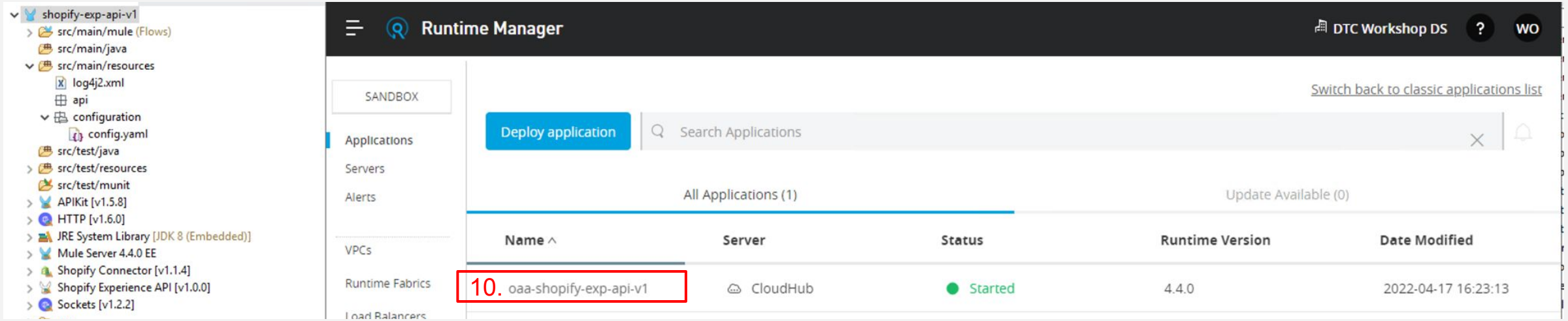
9. Screen will show that your application is **Deploying**.



Module 2: Ubiquitous Connectivity: Lab 3 (8/10)

Step 1: Deploy Shopify Project from Anypoint Studio

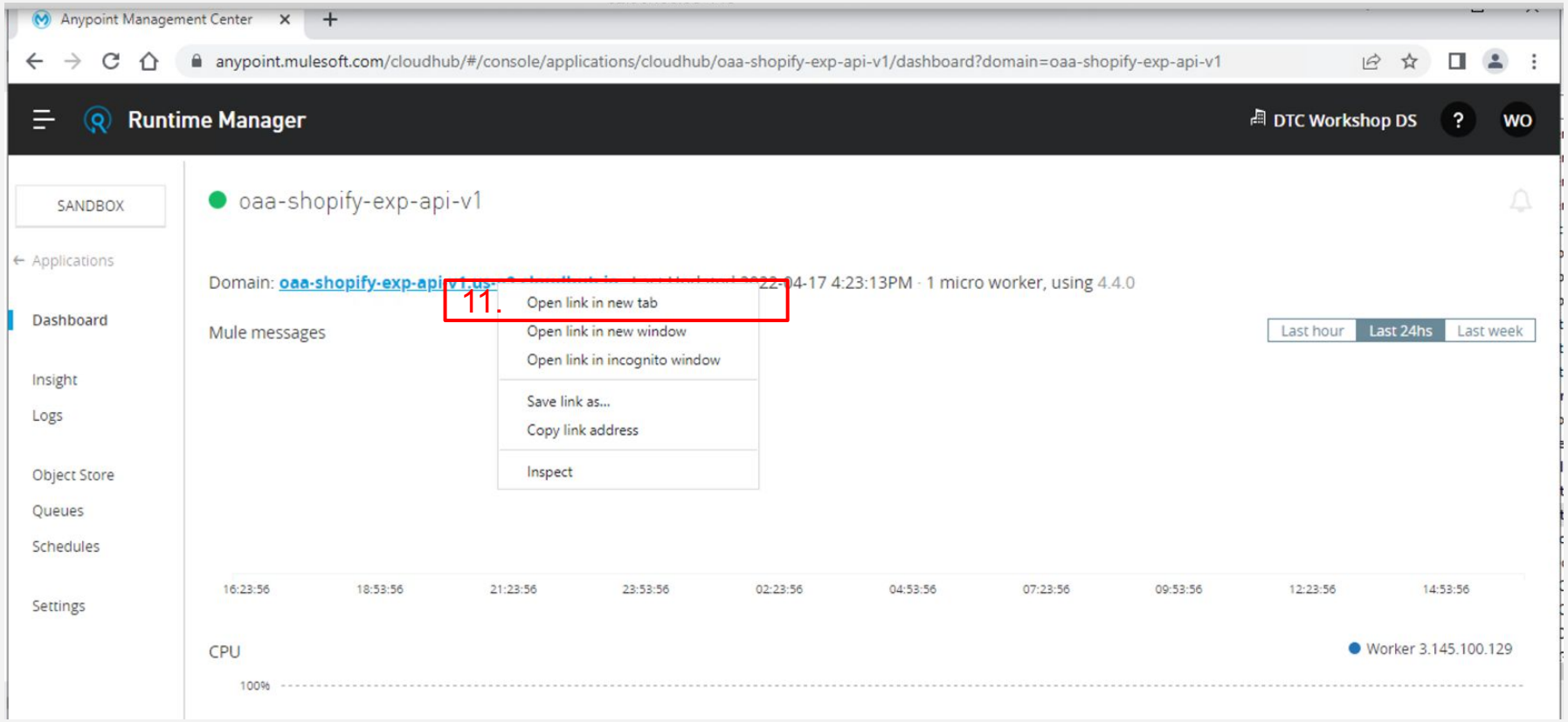
10. Wait until you see that status has changed to **Started.** Click on the **<first name or initials>shopify-exp-api-v1.**



Module 2: Ubiquitous Connectivity: Lab 3 (9/10)

Step 1: Deploy Shopify Project from Anypoint Studio

11. You will be taken to a Dashboard screen **Right Click** on the **<first name or initials>shopify-exp-api-v1 and select Open link in new Tab.**



Module 2: Ubiquitous Connectivity: Lab 3 (10/10)

MuleSoft

Step 1: Deploy Shopify Project from Anypoint Studio

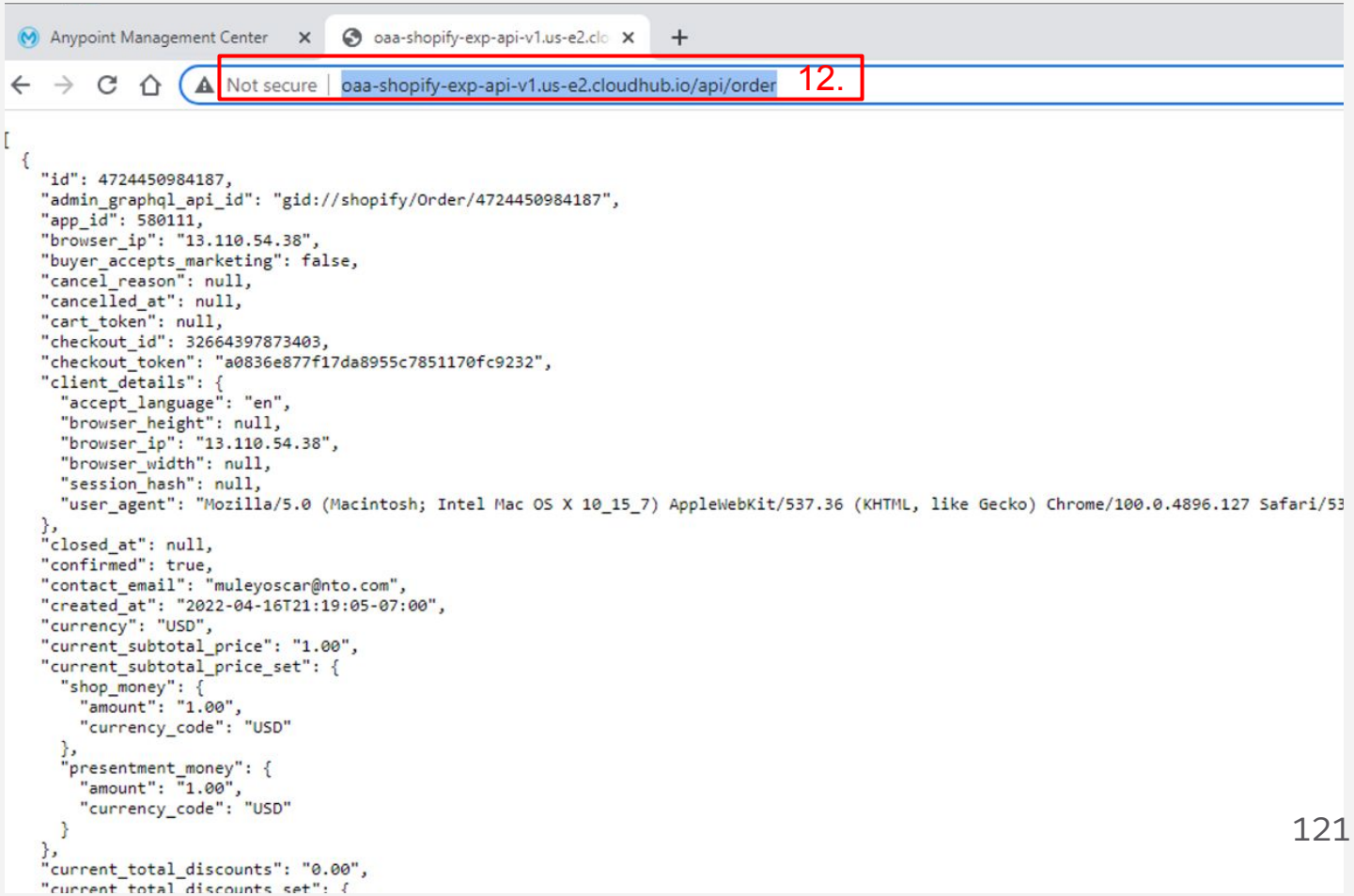
12. Replace browser URL with: **http://<your first name or initial>-shopify-exp-api-v1.us-e2.cloudhub.io/api/order** and hit **Enter/Return.**

You should get a payload response like so.

Now we have successfully deployed our Shopify API!

Congratulations! You have completed Module 2!

Proceed to Module 3



MuleSoft

Module 3: Overview

Source: DTC e-commerce: How consumer brands can get it right

Module 3: Apply Policy to Shopify API: Overview

Overview

In these labs you will **configure policies and SLAs** that will allow you to manage security, quality of service to the **Shopify API**. In addition, you will **monitor your API through API analytics**.

We will proxy an existing API in Lab 1 and You can:

- Manage security and enforce policies
- Manage contracts associated with that API
- Monitor the API with analytics and KPI metrics in a dashboard view

This makes it easy for the **API manager** to understand how the API is performing, how it's being used and by whom, and for the API Administrator to identify potential issues before they arise.

The 1 lab in this unit is:

- Lab 1: Proxy an existing Shopify API & Apply Policies to Shopify API

Proceed to Lab 1

MuleSoft

Module 3: Lab 1

Source: DTC e-commerce: How consumer brands can get it right

Module 3: Apply Policy to Shopify API: Lab 1 (1/23)

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.

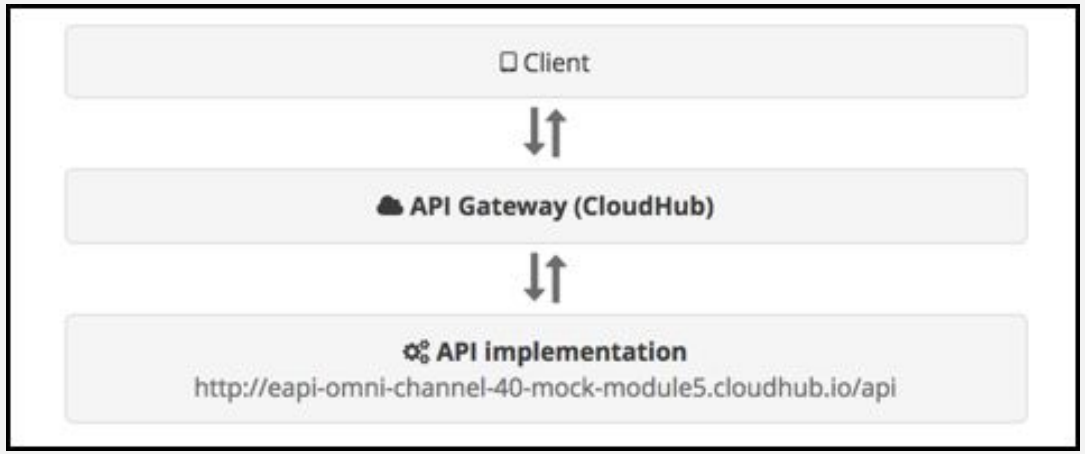
MuleSoft can apply governance directly to a MuleSoft implemented API or through a proxy gateway for these and other external existing APIs.

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

In this lab, we will define an API that will act as a proxy for NTO's Shopify API. The proxy will be deployed to Anypoint Platform's API Gateway, which is powered by the Mule runtime.

Module 3: Apply Policy to Shopify API: Lab 1 (2/23)

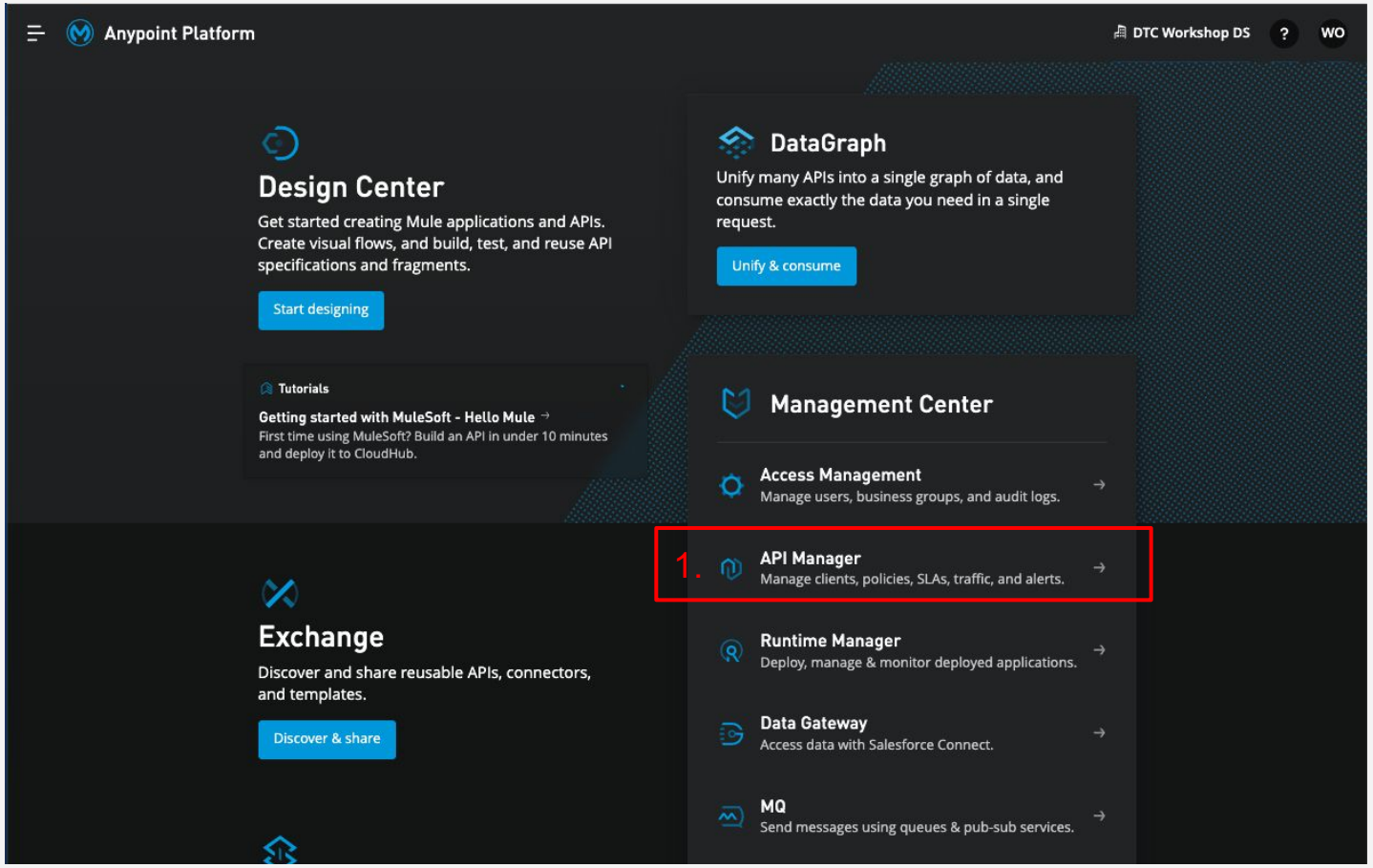
Clients will access the API through the API Gateway which will then forward the requests to the actual Omni Channel API Mock Implementation. Having the proxy deployed on the API gateway allows Anypoint Platform to manage, control access and monitor the usage of the API, which we will look at in the succeeding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



Module 3: Apply Policy to Shopify API: Lab 1 (3/23)

Step 1: Go to API Manager

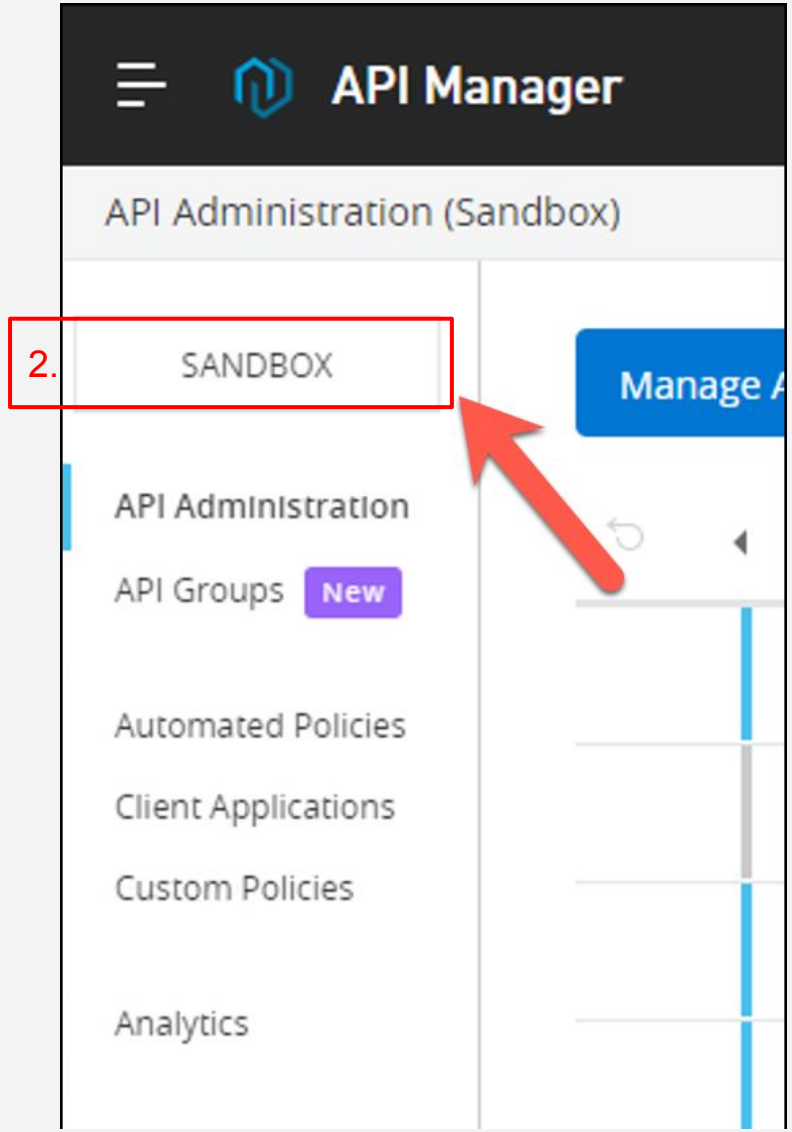
1. In Anypoint Platform (https://anypoint.mulesoft.com), click the **API Manager** icon to start creating your API.



Module 3: Apply Policy to Shopify API: Lab 1 (4/23)

Step 1: Go to API Manager

2. Select **SANDBOX** as the environment



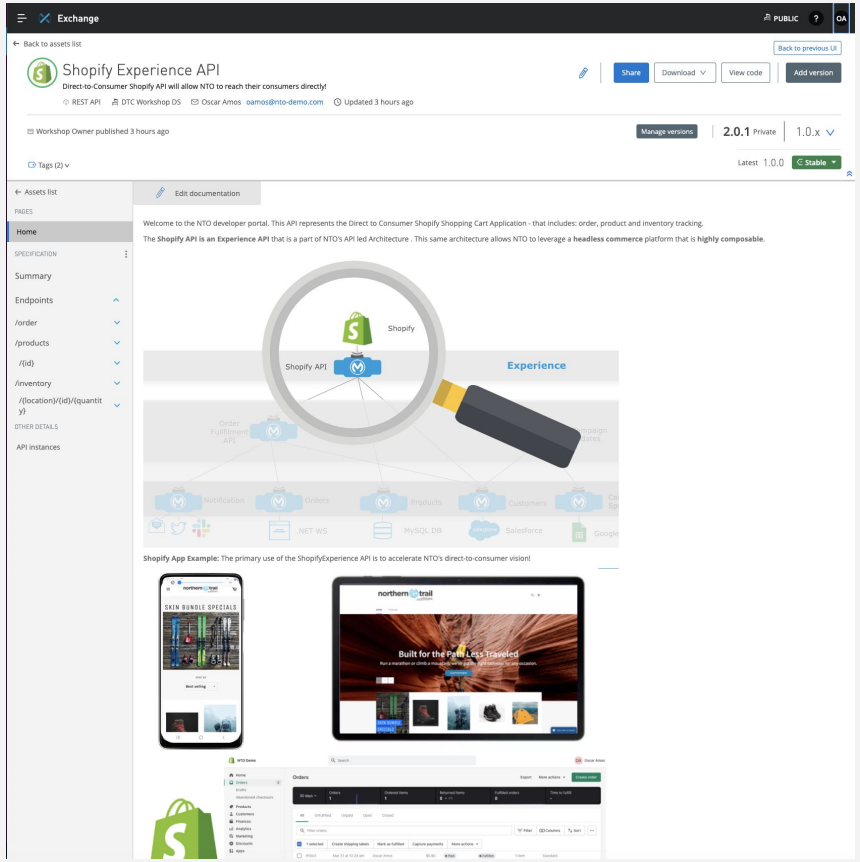
Module 3: Apply Policy to Shopify API: Lab 1 (5/23)

Step 2: Configure an API Proxy

For this lab, we are going to configure the API as a proxy to NTO's Shopify API Implementation. The API is available as an HTTP Restful API accessible through the base URL http://shopify-exp-api-v1.us-e2.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

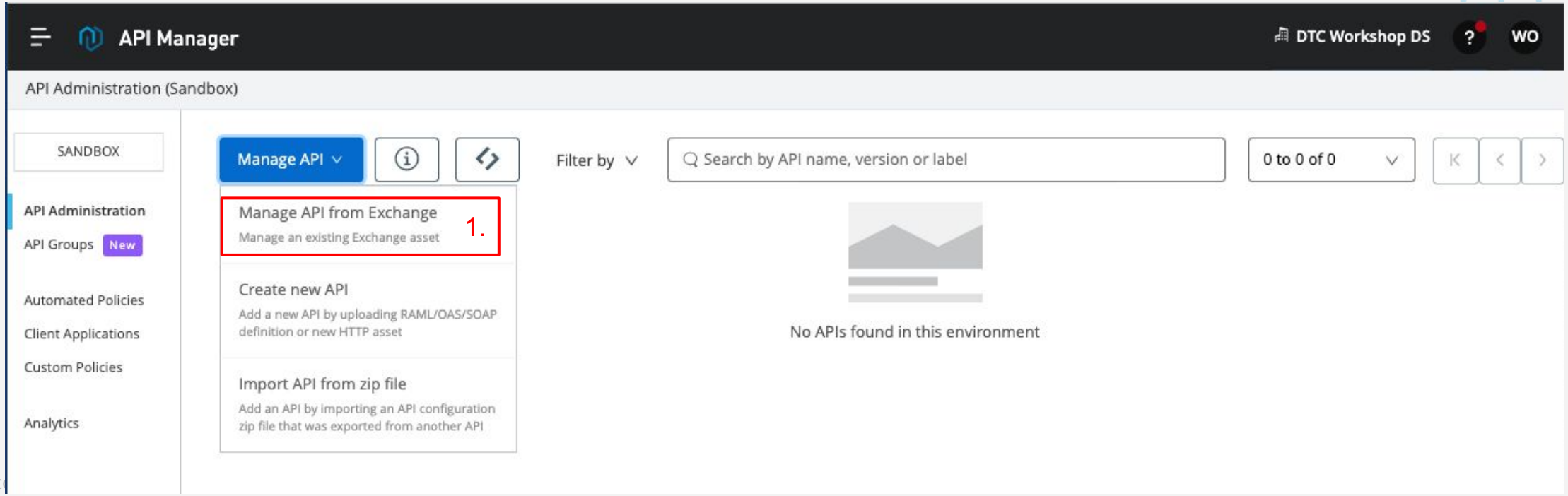


Module 3: Apply Policy to Shopify API: Lab 1 (6/23)

Step 2: Configure an API Proxy

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. Navigate to the API Manager page, click on **Manage API** and select **Manage API from Exchange**.



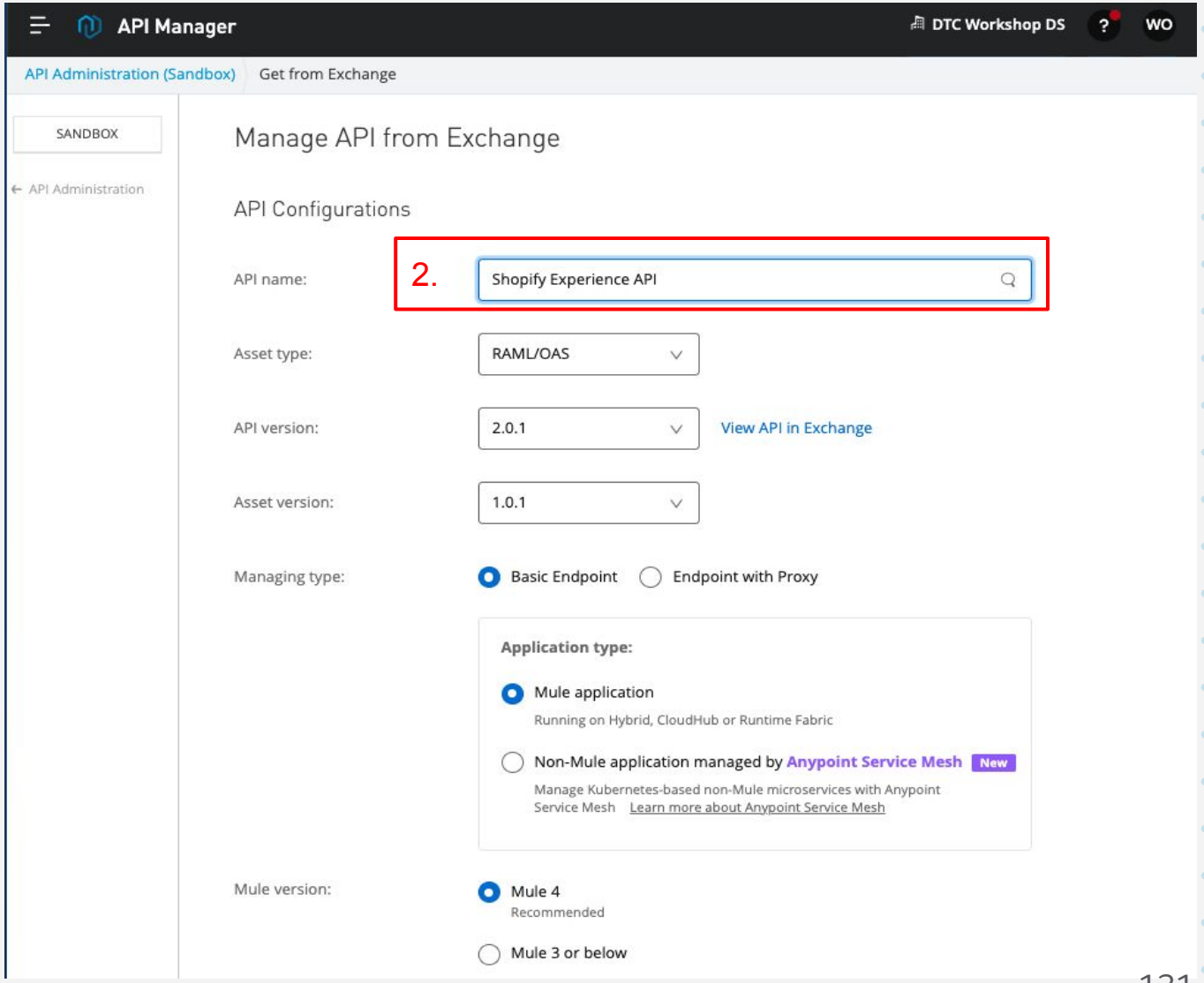
Module 3: Apply Policy to Shopify API: Lab 1 (7/23)

Step 2: Configure an API Proxy

Configure the API with the following information:

2. Under **API name**: Type "**Shopify Experience API**". Notice when you start to write the name, the field is auto completed:

a. **Asset type**: Select from the drop down list `RAML/OAS`.
b. **API version**: Select from the drop down list `1.0.0`.
c. **Asset version**: Select from the drop down list `1.0.1`.
d. **Managing type**: Select `Endpoint with proxy`.
e. **Proxy Deployment Target**: Choose `Cloudhub`
f. **Mule Version**: Check the Mule 4 option.



Module 3: Apply Policy to Shopify API: Lab 1 (8/23)

MuleSoft

Step 2: Configure an API Proxy

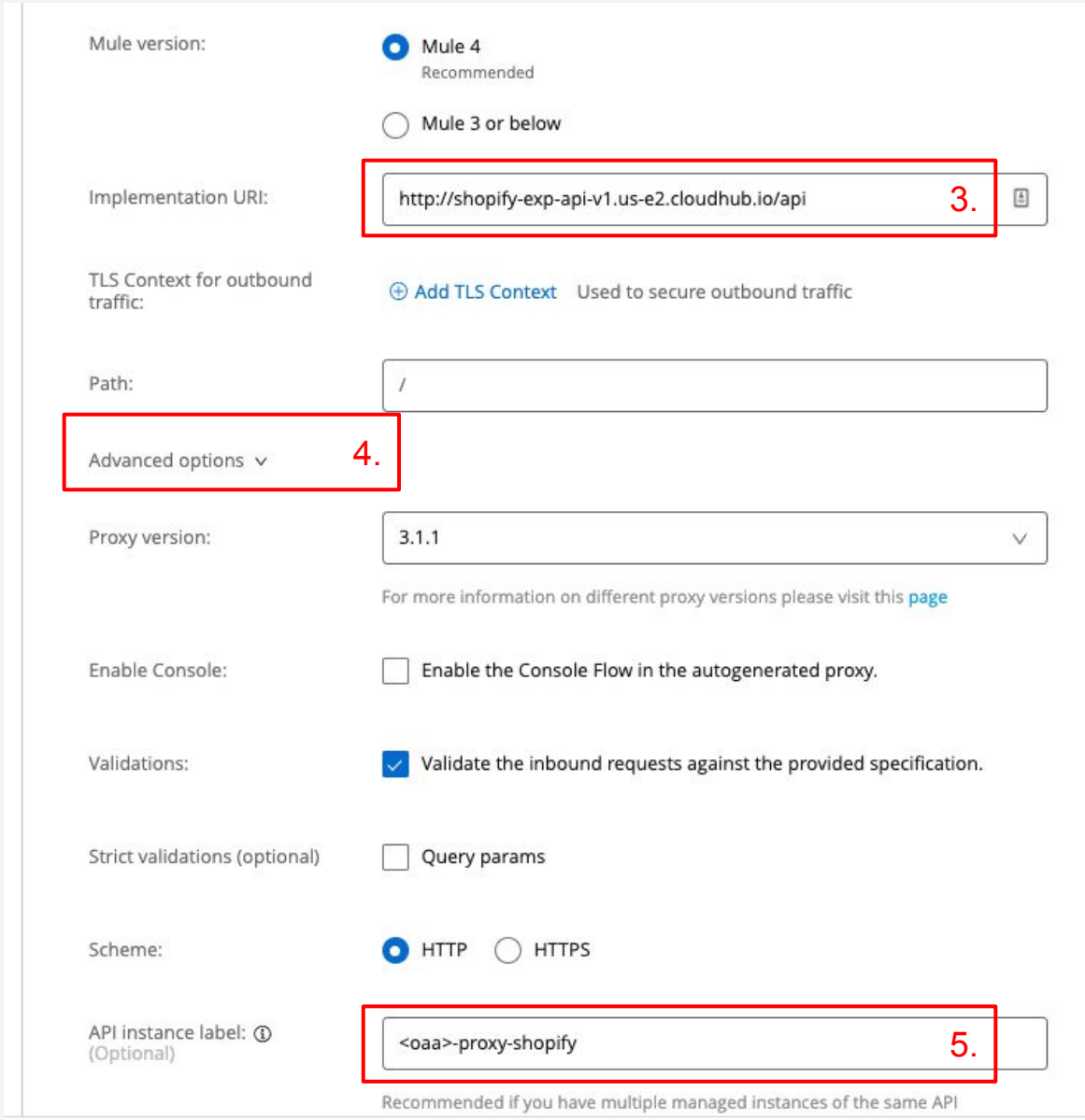
Configure the API with the following information:

3. Type: http://shopify-exp-api-v1.us-e2.cloudhub.io/api in the Section labeled **Implementation URI**:

4. Click **Advanced options** to expand to as follows.

5. Add **<first name or initials>-proxy-shopify** in the field, **API Instance label.**

6. And Click **Save.**



Module 3: Apply Policy to Shopify API: Lab 1 (9/23)

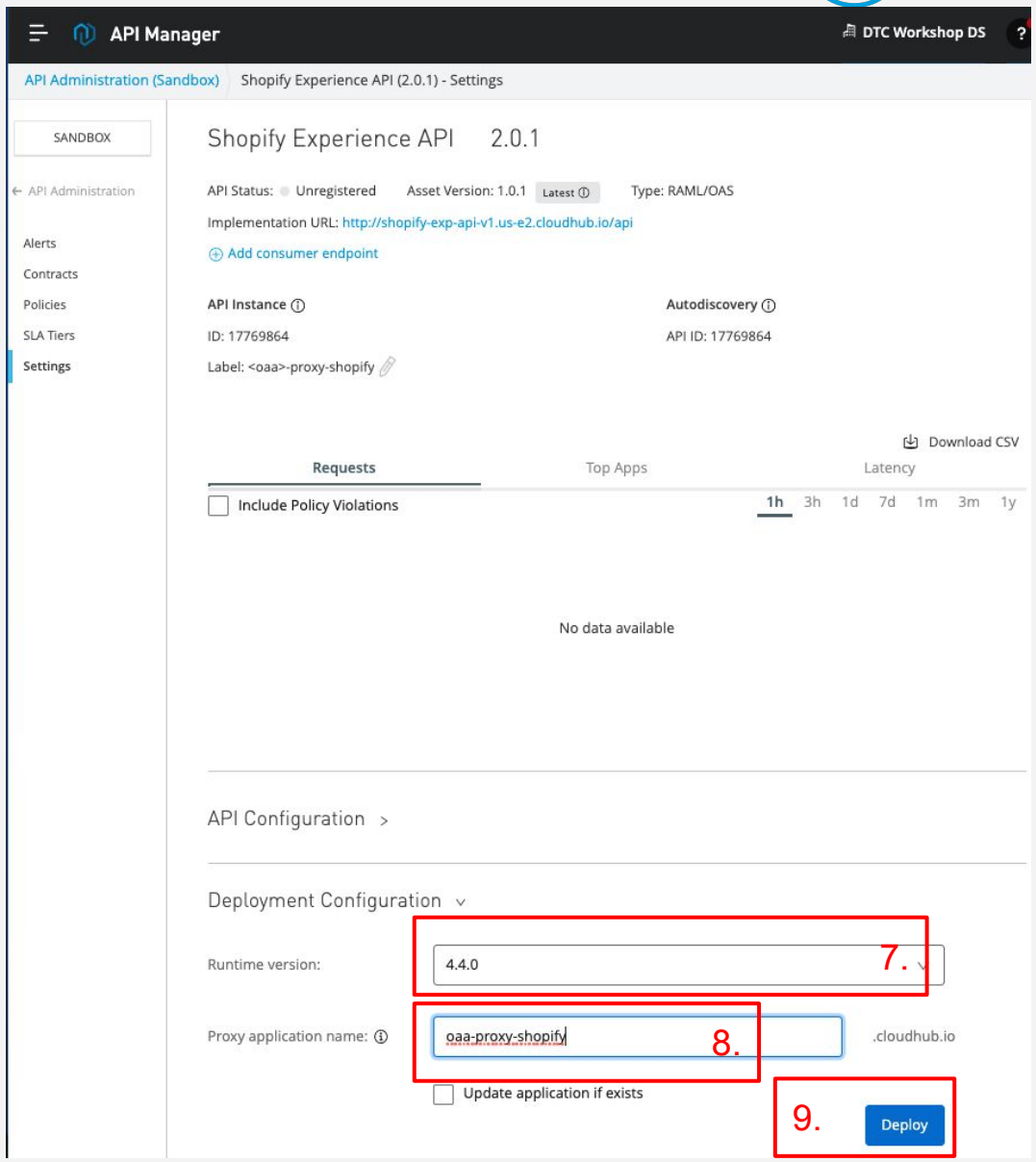
Step 2: Configure an API Proxy

Configure the API with the following information:

7. Set the field Runtime version to **4.4.0.**

8. In Proxy application field type:
<first name or initials)-proxy-shopify

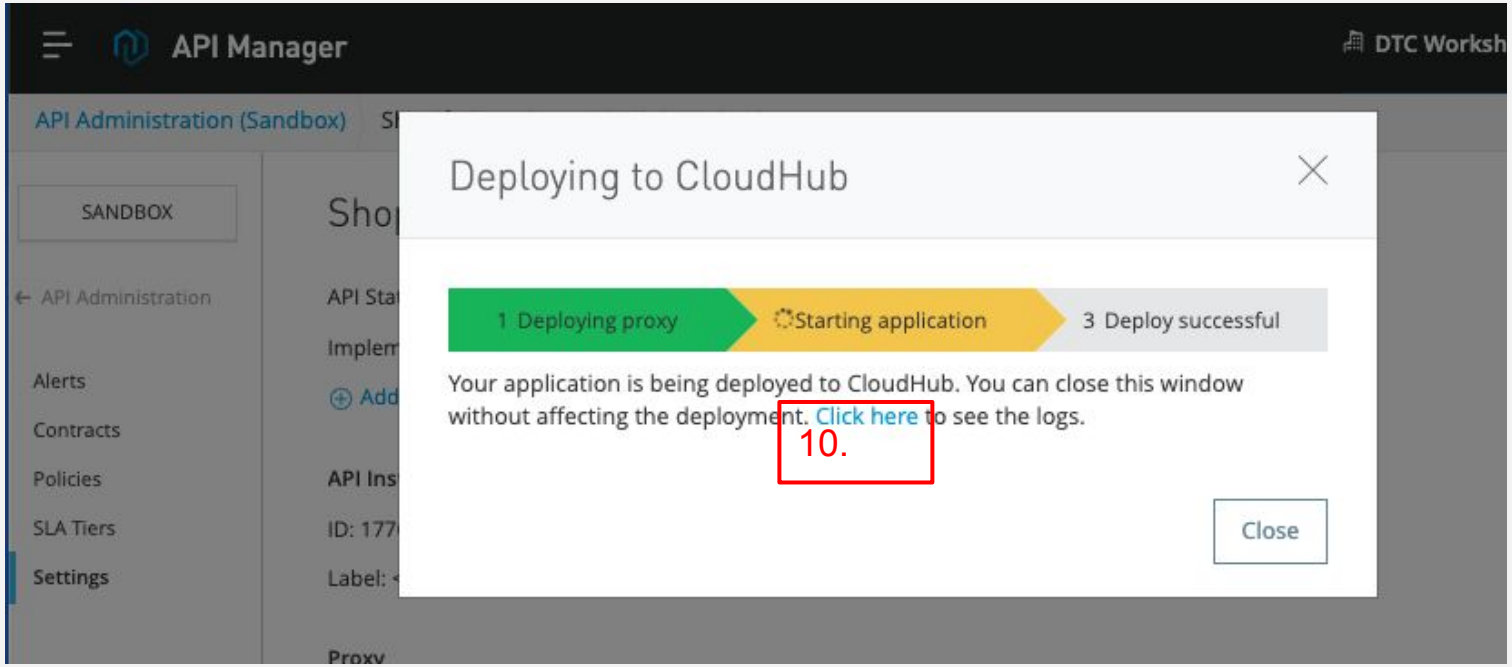
9. Then click **Deploy**.



Module 3: Apply Policy to Shopify API: Lab 1 (10/23)

Step 2: Configure an API Proxy

10. Click on **Click Here** to see the log data and monitor the progress.

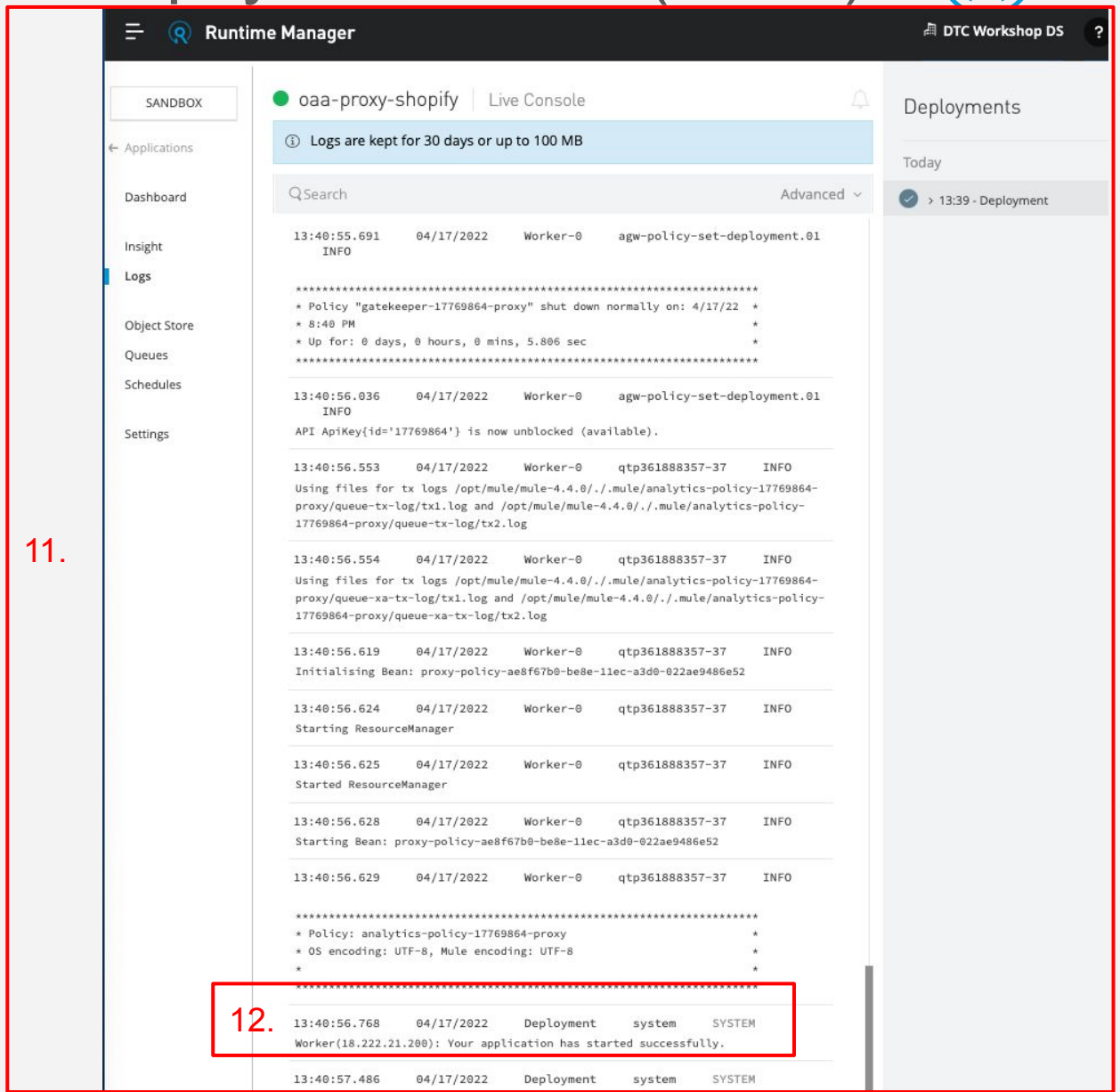


Module 3: Apply Policy to Shopify API: Lab 1 (11/23)

Step 2: Configure an API Proxy

11. A new browser tab will open that will show the CloudHub log for this application in **Runtime Manager**.

12. Wait until you see "**Your application is started** " and you can close this browser tab and go to back to the previous tab.

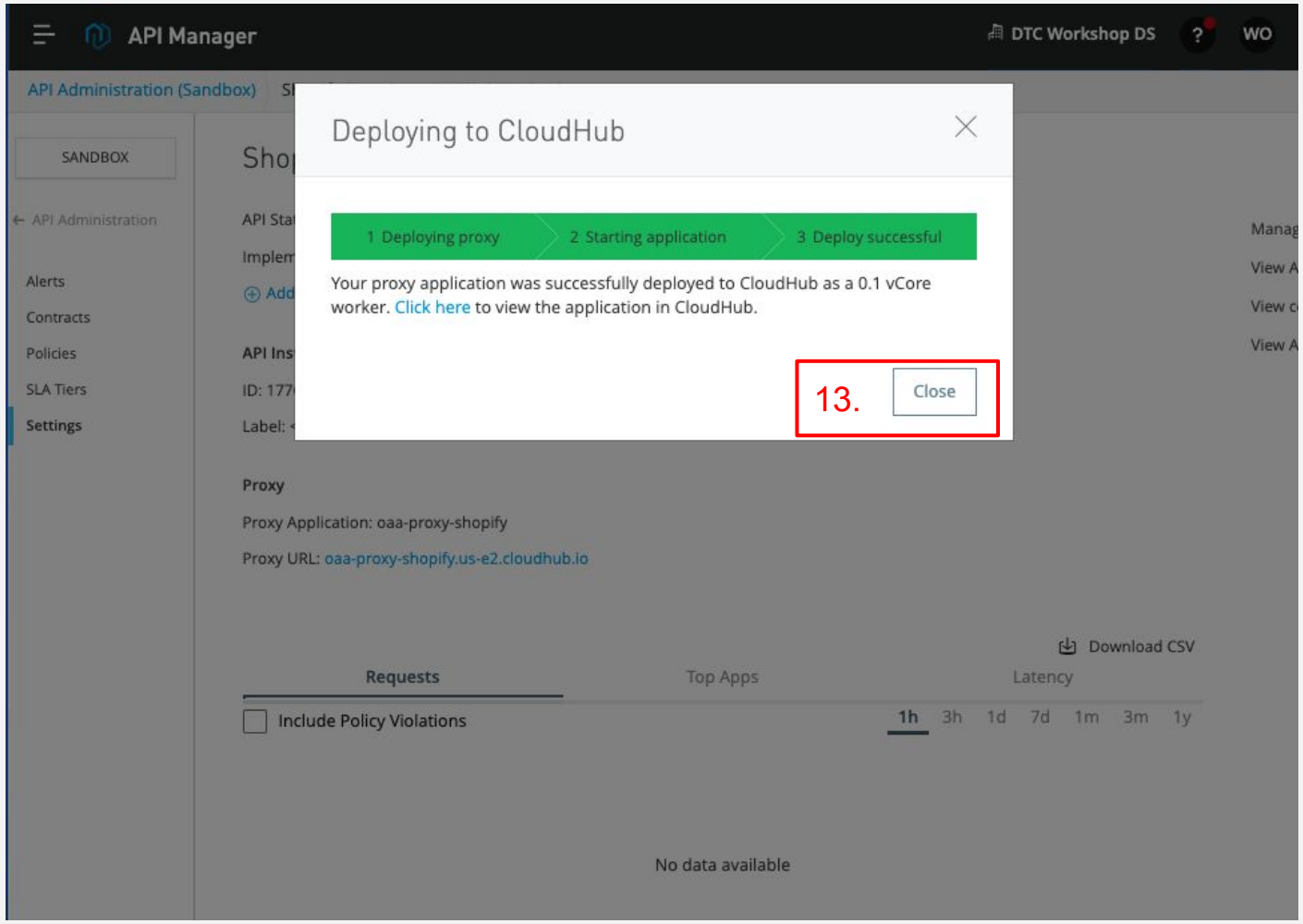


Module 3: Apply Policy to Shopify API: Lab 1 (12/23)

MuleSoft

Step 2: Configure an API Proxy

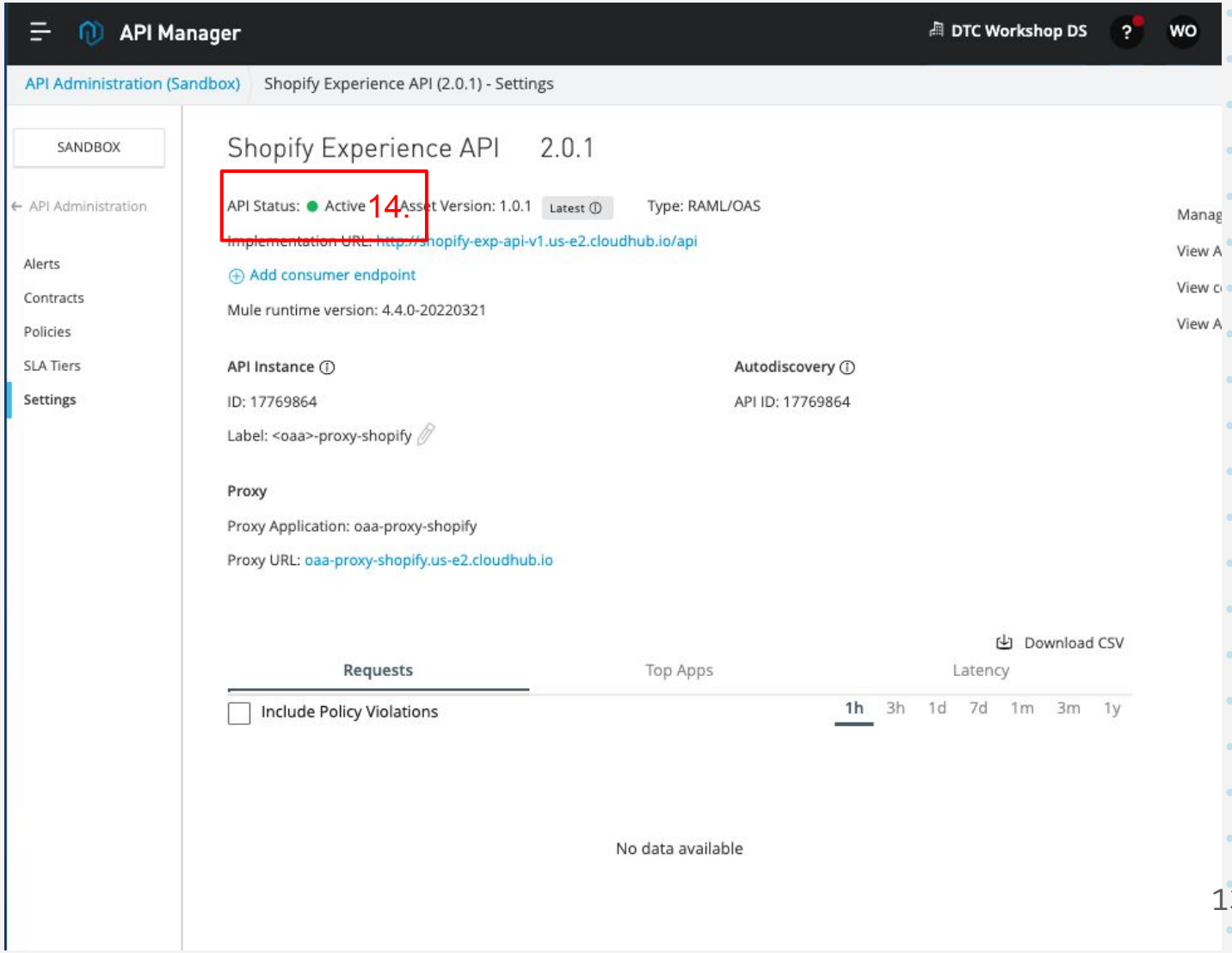
13. Go back to the previous browser tab where you were configuring the API. You should see the green Deploy Successful in the status bar. Click **Close.**



Module 3: Apply Policy to Shopify API: Lab 1 (13/23)

Step 2: Configure an API Proxy

14. 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 dot next to it, as shown below. This indicates that your API was successfully deployed with its corresponding Proxy and is now being managed.

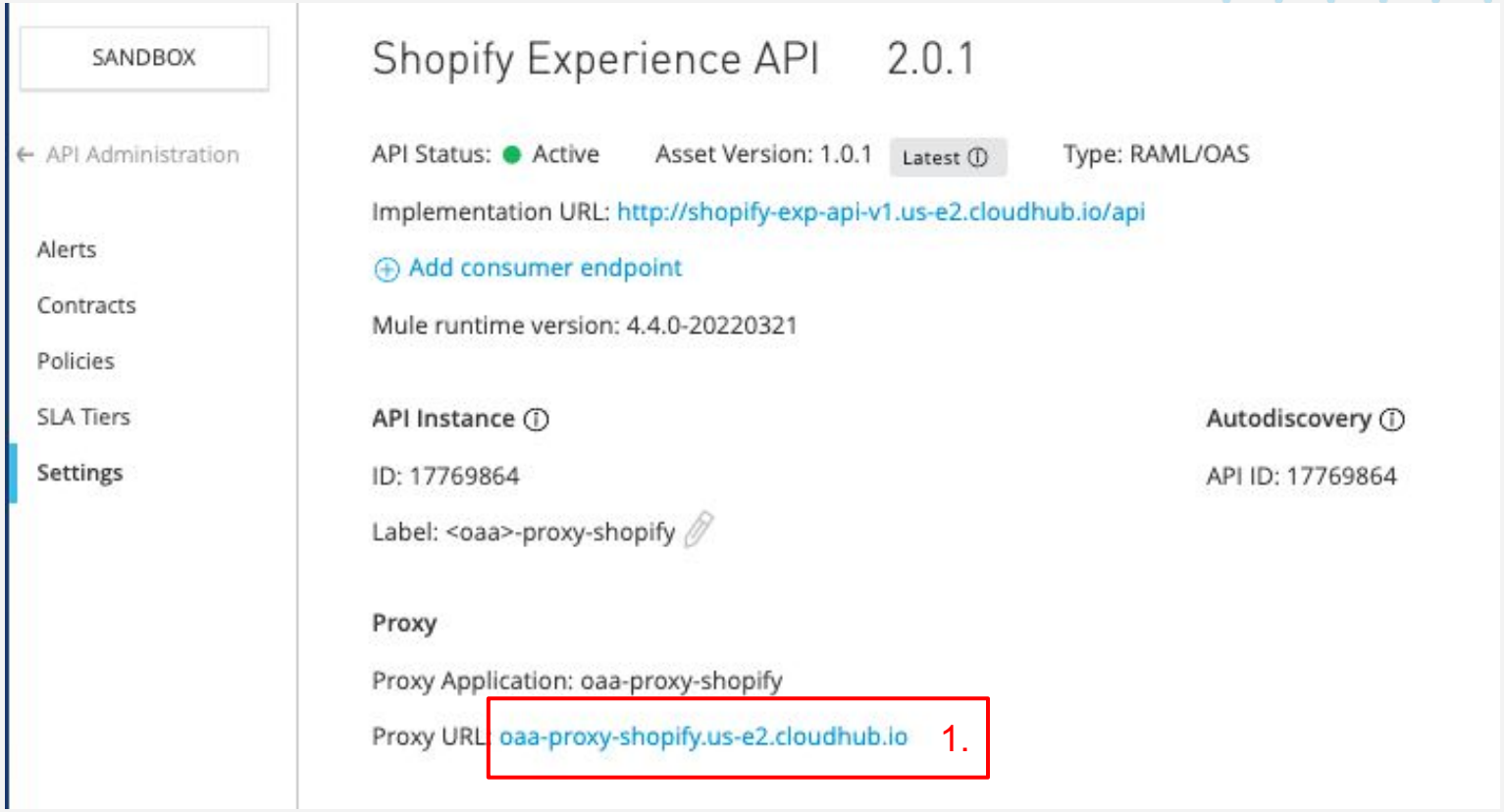


Module 3: Apply Policy to Shopify API: Lab 1 (14/23)

Step 3: Test the API Proxy

Your proxy API is now accessible via CloudHub.

1. At the Proxy URL, Right Click the URL and Select **Open Link in New Tab.** This will open a new Browser Tab.



Module 3: Apply Policy to Shopify API: Lab 1 (15/23)

Step 3: Test the API Proxy

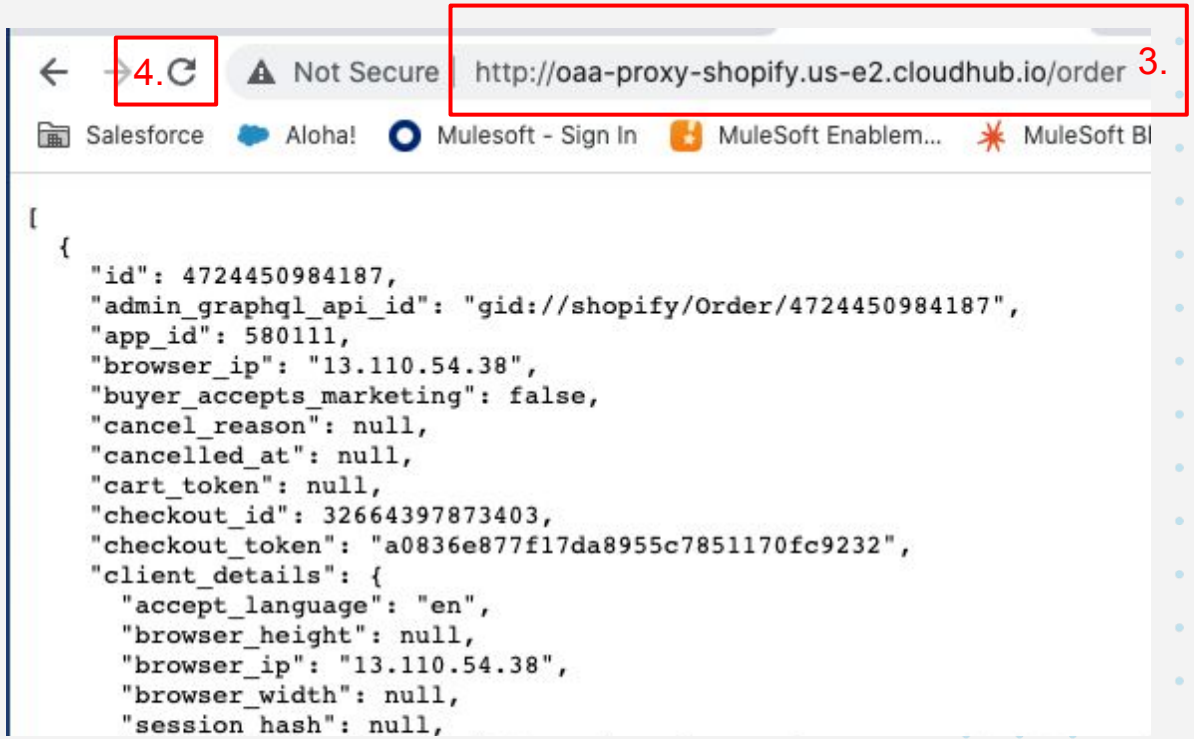
Your proxy API is now accessible via CloudHub.

2. You will see a page with a error message stating: **“resource_not_found”**

3. Add **“/order”** to the end of the proxy URL in the Browser and

4. **Refresh** the Page.

You should see the Shopify GET orders payload response. Good the proxy is working! Now to our next Step.



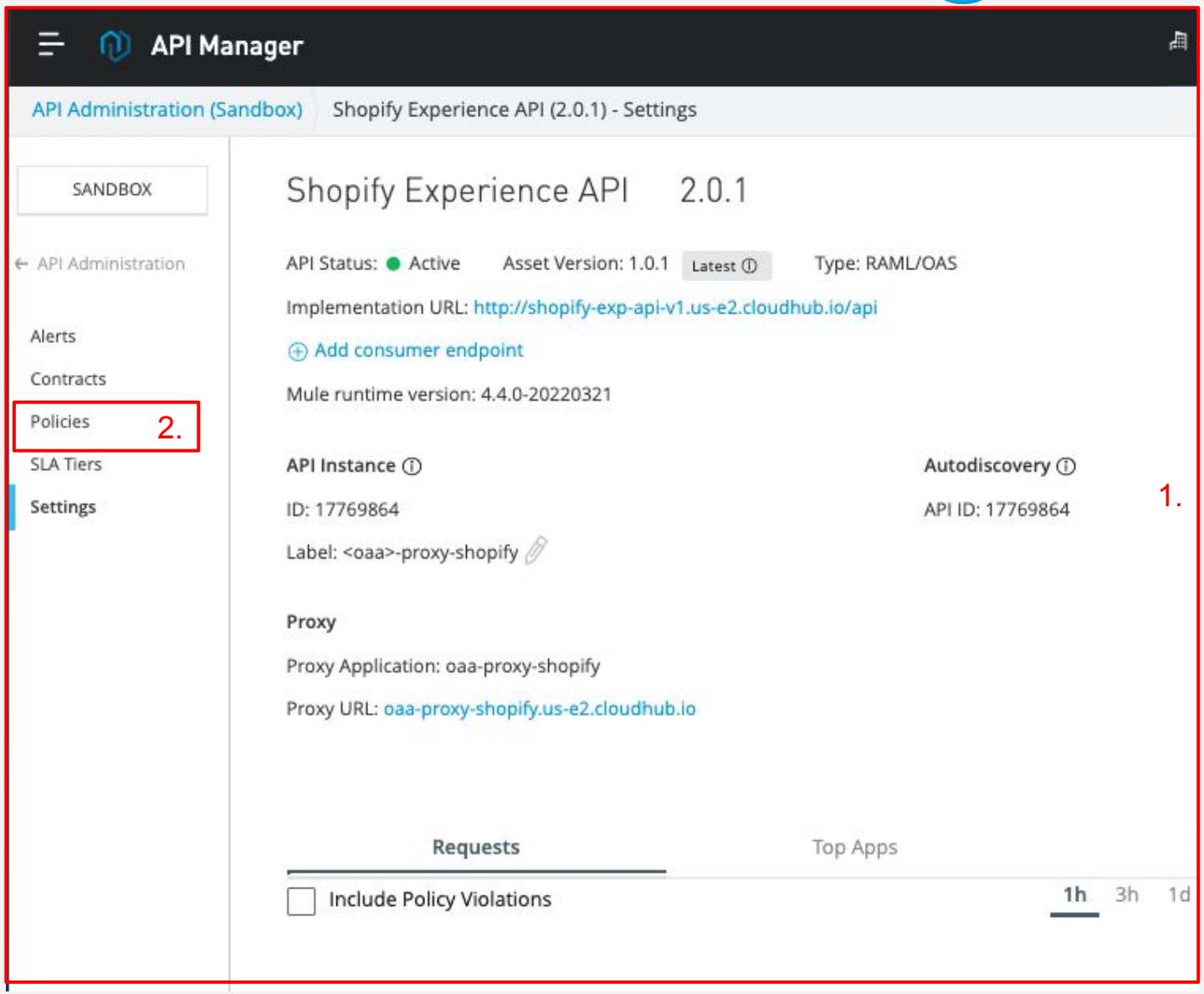
Module 3: Apply Policy to Shopify API: Lab 1 (16/23)

Step 4: Apply Policy To API

Your proxy API is now accessible via CloudHub.

1. Toggle to your previous Browser tab that has details on your Shopify API as seen:

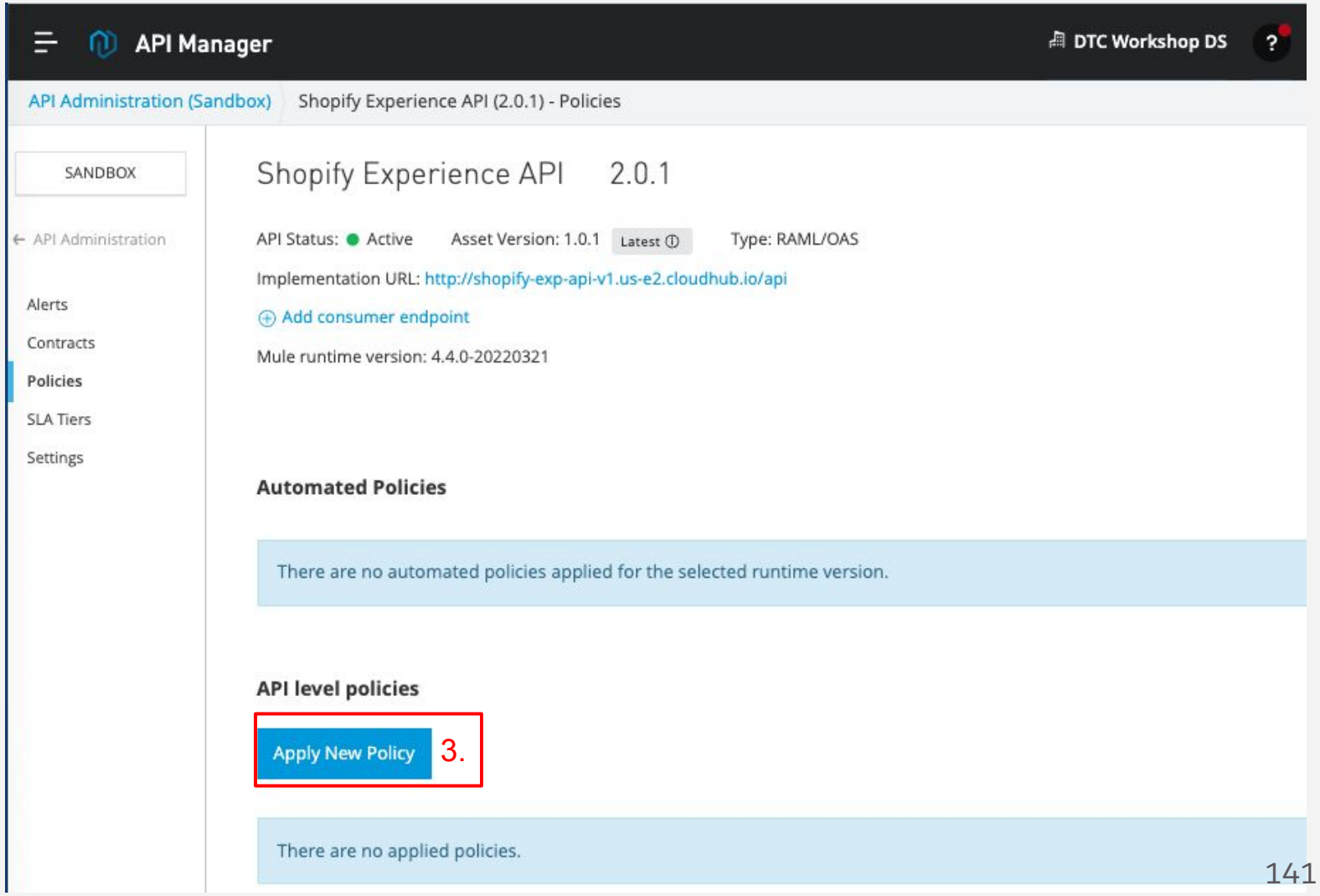
2. Click **Policies** on the left menu pane.



Module 3: Apply Policy to Shopify API: Lab 1 (17/23)

Step 4: Apply Policy To API

3. In the next Screen Click the **Apply New Policy** Button.



Module 3: Apply Policy to Shopify API: Lab 1 (18/23)

Step 4: Apply Policy To API

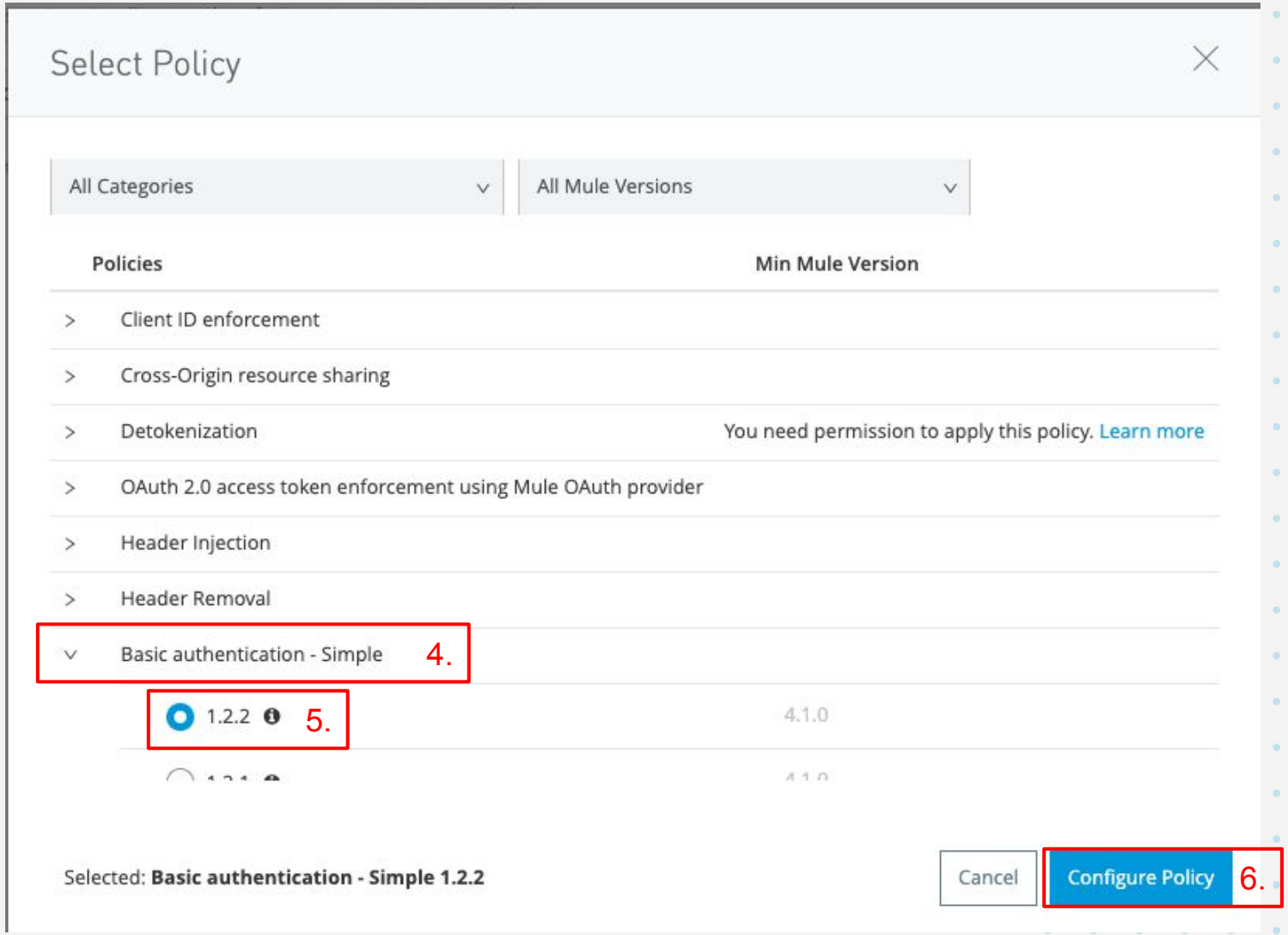
You will see a screen showing a list of different Policies. At the top **All Categories** you can filter the Policy listing by categories (e.g. Security, Quality of Service and so on).

Since we are trying to get NTO's Shopify Order data our requirement is to have some sort of system authentication before giving retrieving that data from Shopify.

4.Select **Basic Authentication - Simple**

5. Choose radio button **1.2.2**

6. Click **Configure Policy** Button Below.

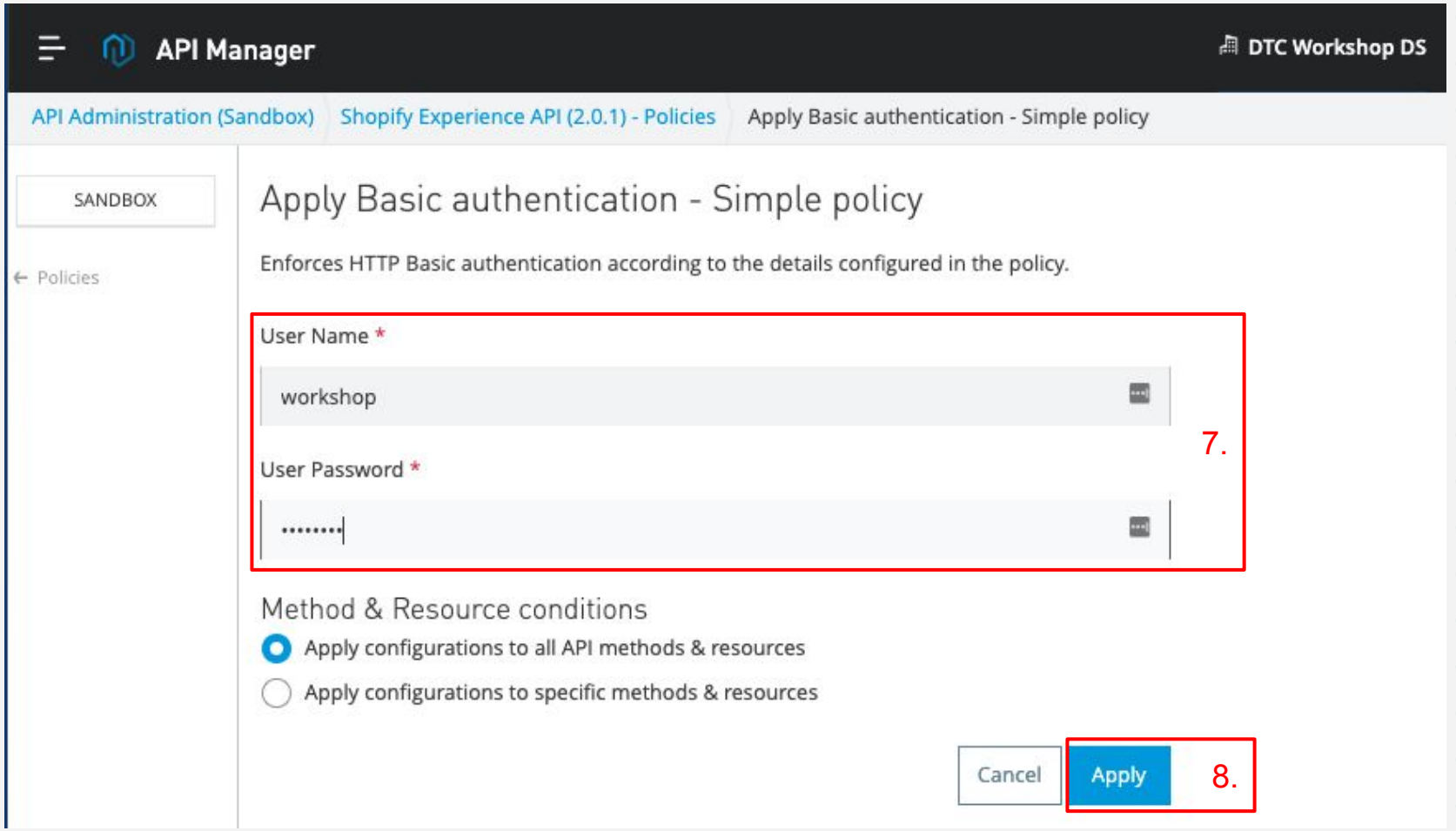


Module 3: Apply Policy to Shopify API: Lab 1 (19/23)

Step 4: Apply Policy To API

7. In User Name and Password Fields type: **"workshop"**

8. Click the **Apply** Button.

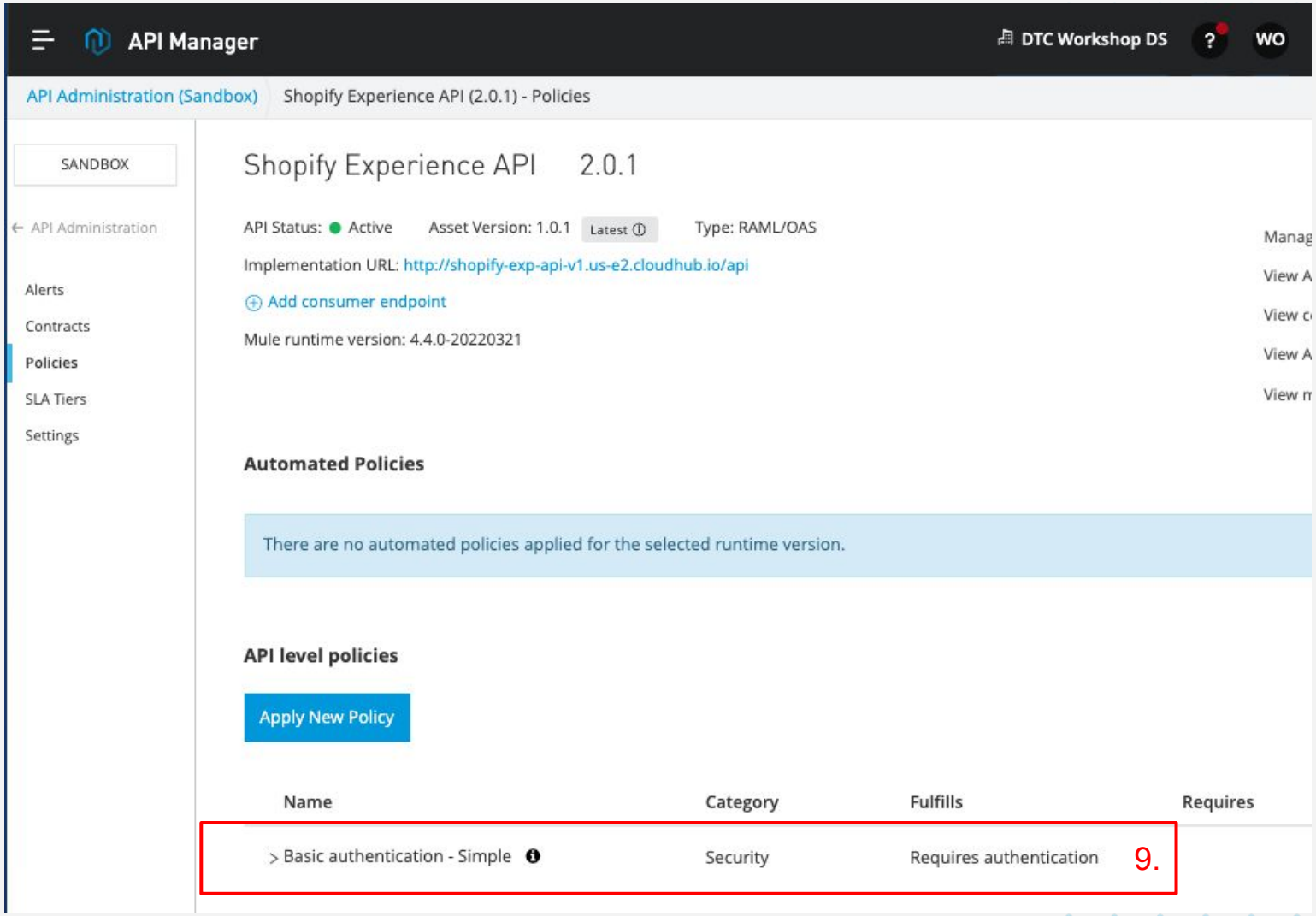


Module 3: Apply Policy to Shopify API: Lab 1 (20/23)

Step 4: Apply Policy To API

9. You will see that the new Policy, **Basic authentication - Simple** is being enforced for our Shopify API.

It may take a several minutes before the policy takes effect, Great time to take a small break and come back.



Module 3: Apply Policy to Shopify API: Lab 1 (21/23)

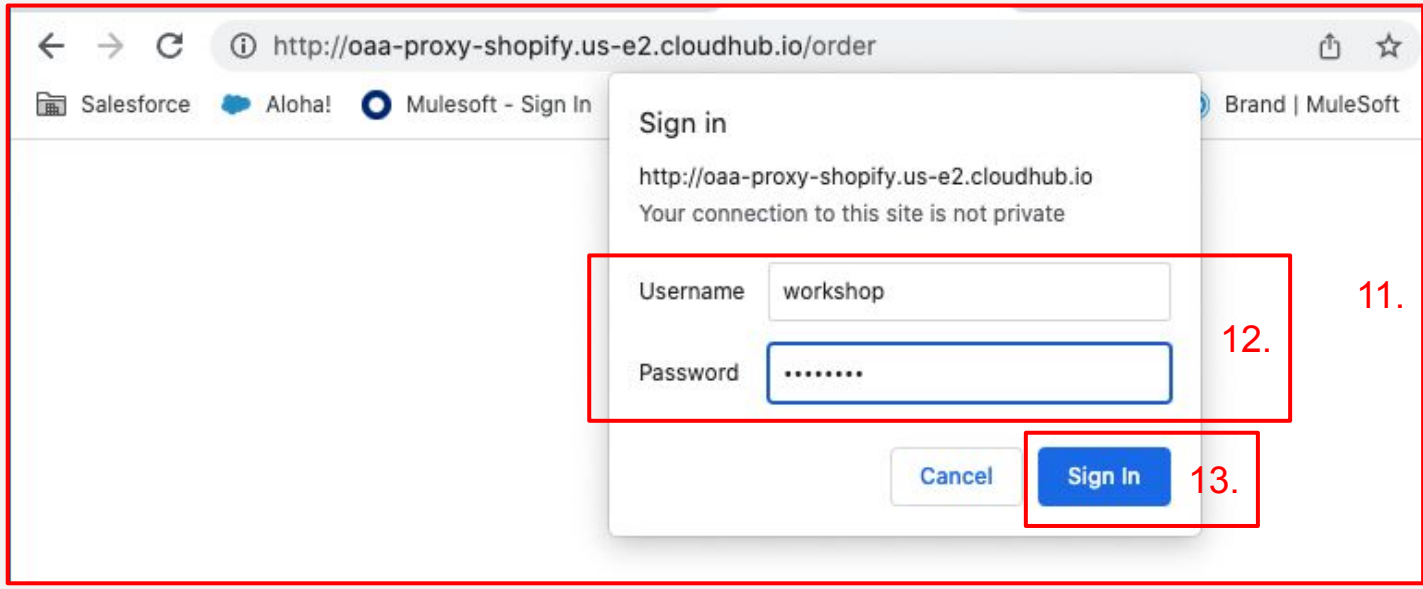
Step 4: Apply Policy To API

10. Toggle back to the Browser Tab you had for proxy URL: **<your first name or initials>-shopify.us-e2.cloudhub.io/order**

11. Continue to **Refresh** the Browser until you see the following:

12. Enter **“workshop”** In the Username and Password Fields.

13. Click **Sign In**



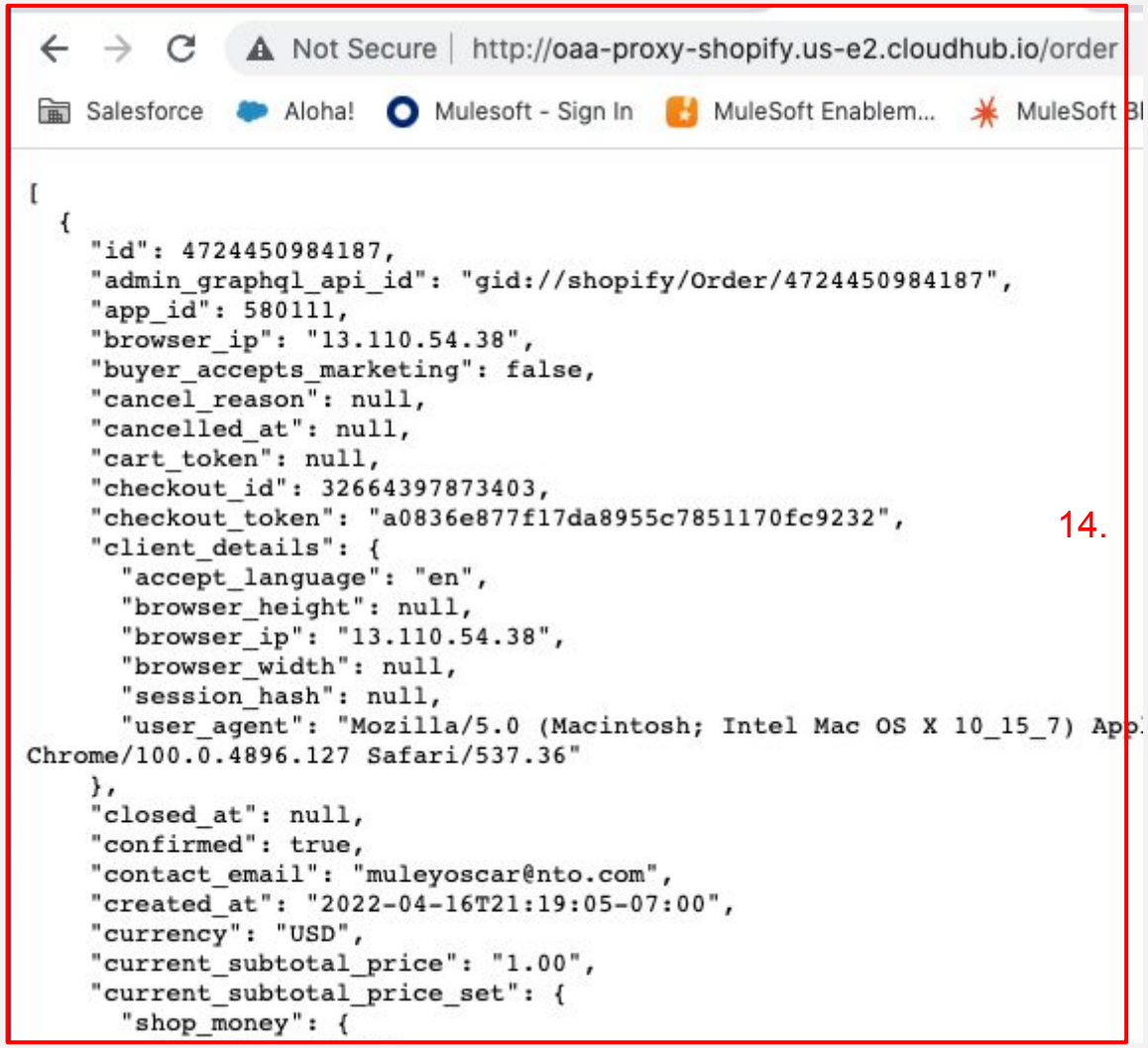
Module 3: Apply Policy to Shopify API: Lab 1 (22/23)

Step 4: Apply Policy To API

14. You should see the Shopify Order Payload Response.

We did it! We just created a security policy for our Shopify API!

Plus, we have completed all Modules and Lab for this DTC Workshop! CONGRATS!!!



Module 3: Apply Policy to Shopify API: Lab 1 (23/23)

Summary

In this lab, we completed the following steps:

- Overview of Anypoint API Manager
- Configure the API proxy
- Test the API proxy using Postman

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
- Deployment Options

Thank you!