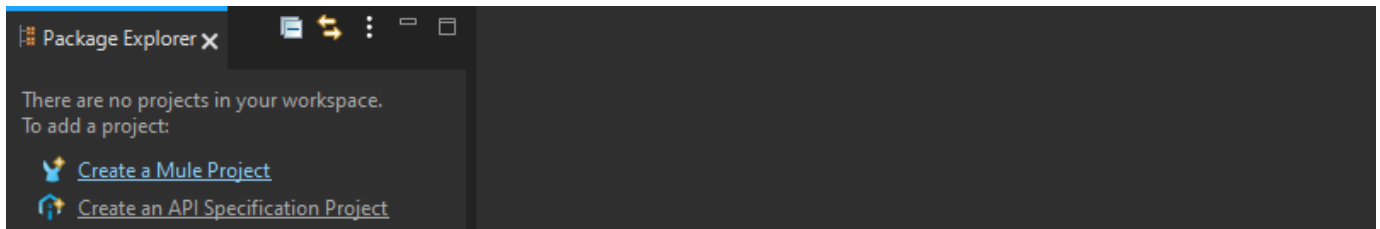


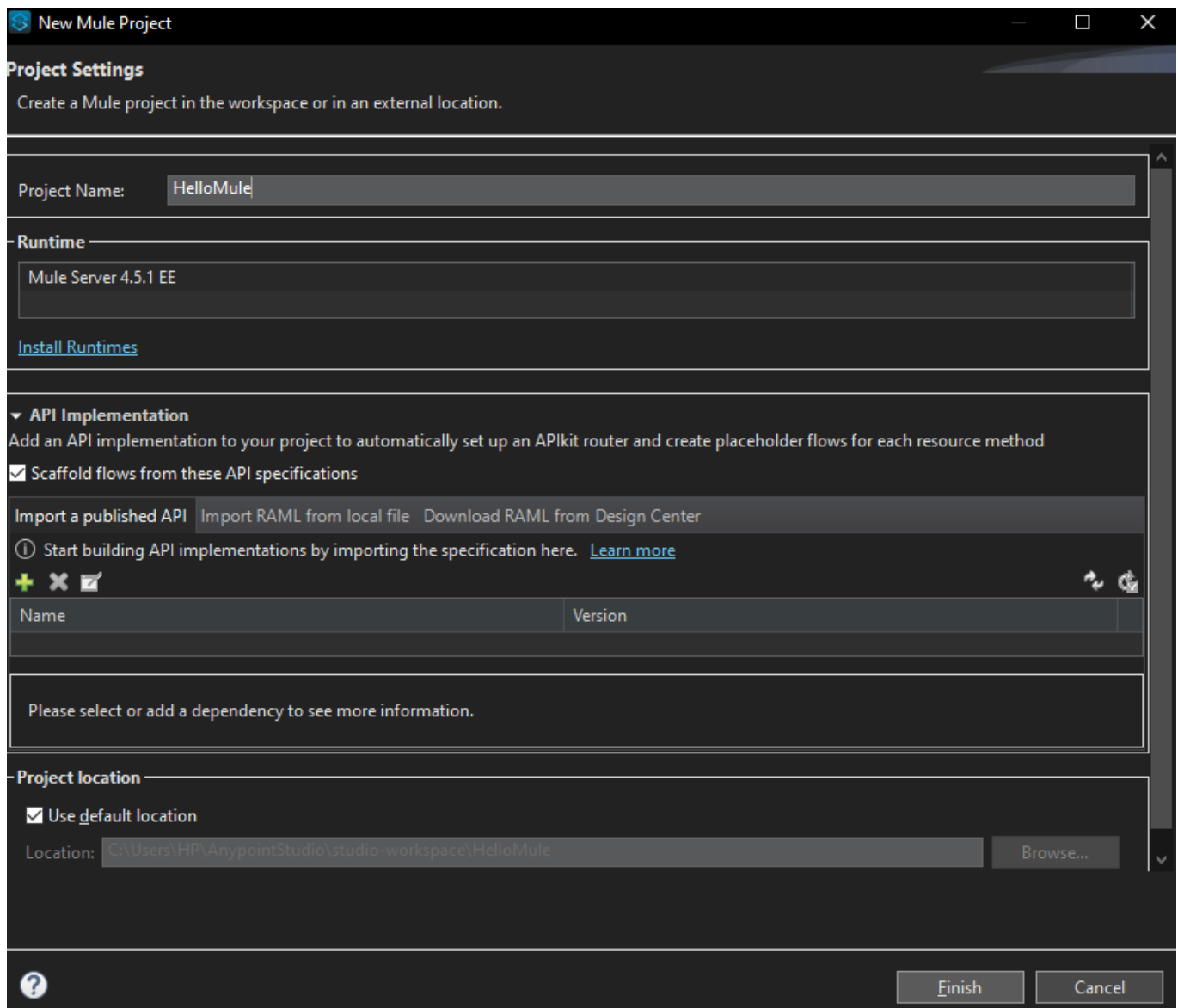
Ejercicio 1: Cree su primera aplicación Hello Mule.

Crea, prueba y despliega tu primer aplicación mule.

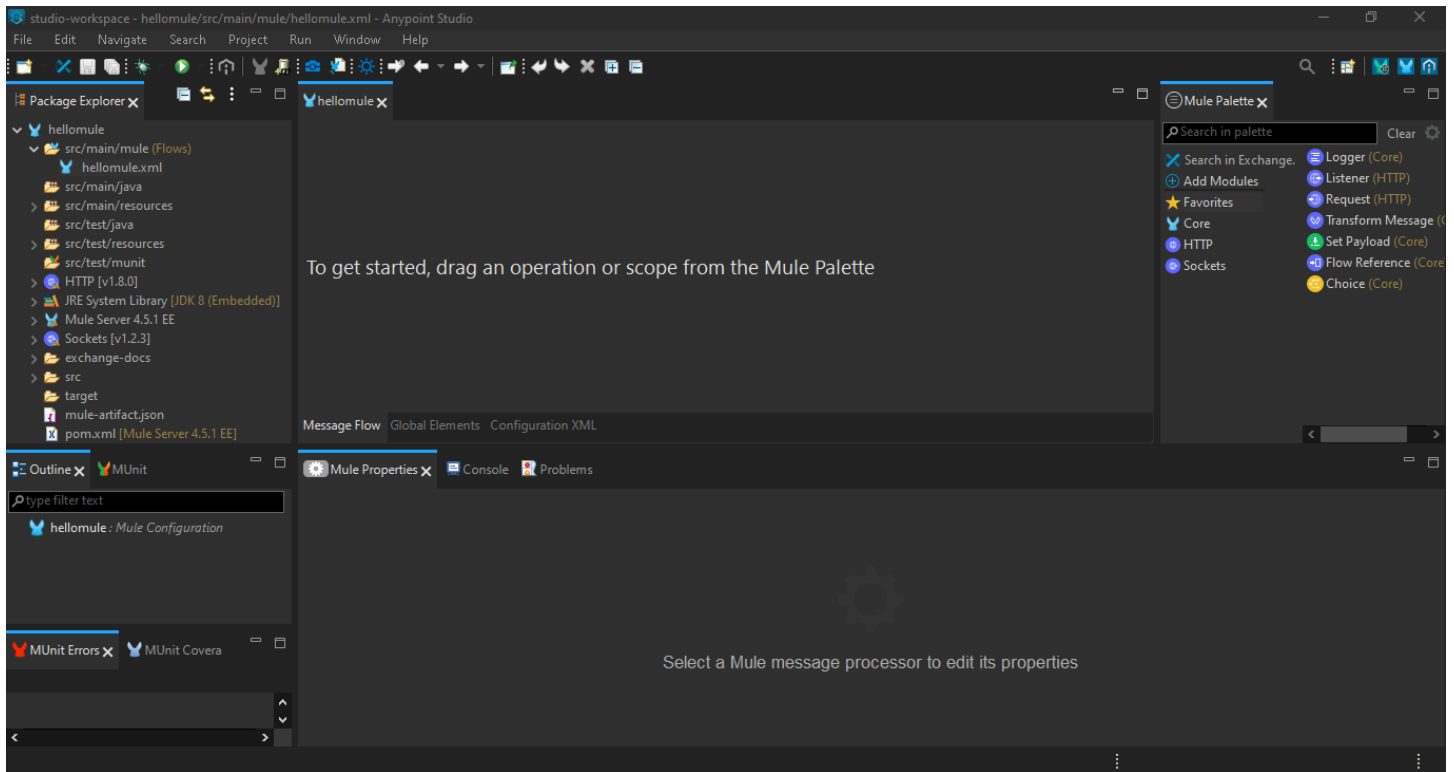
1. Create a new project in Anypoint Studio



Anypoint Studio will open the **New Mule Project** wizard. In the **Project Name** field, enter the value `HelloMule`, then click **Finish**.

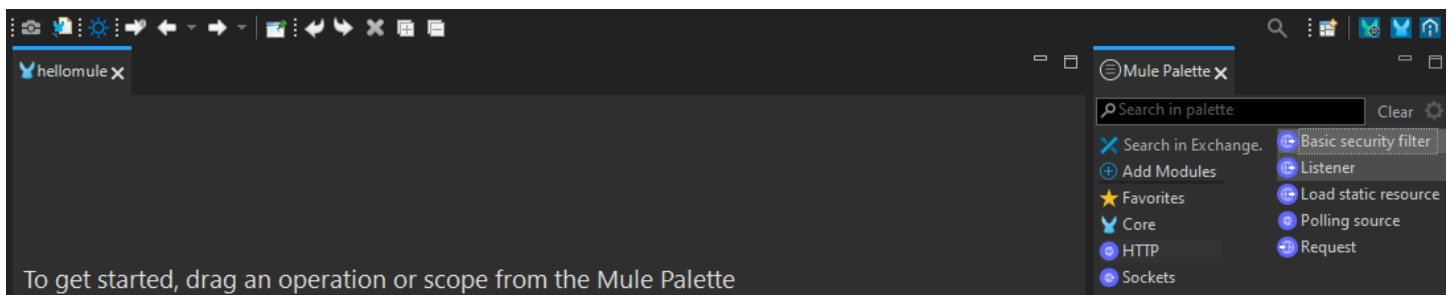


Anypoint Studio will open a new blank Project, for example.

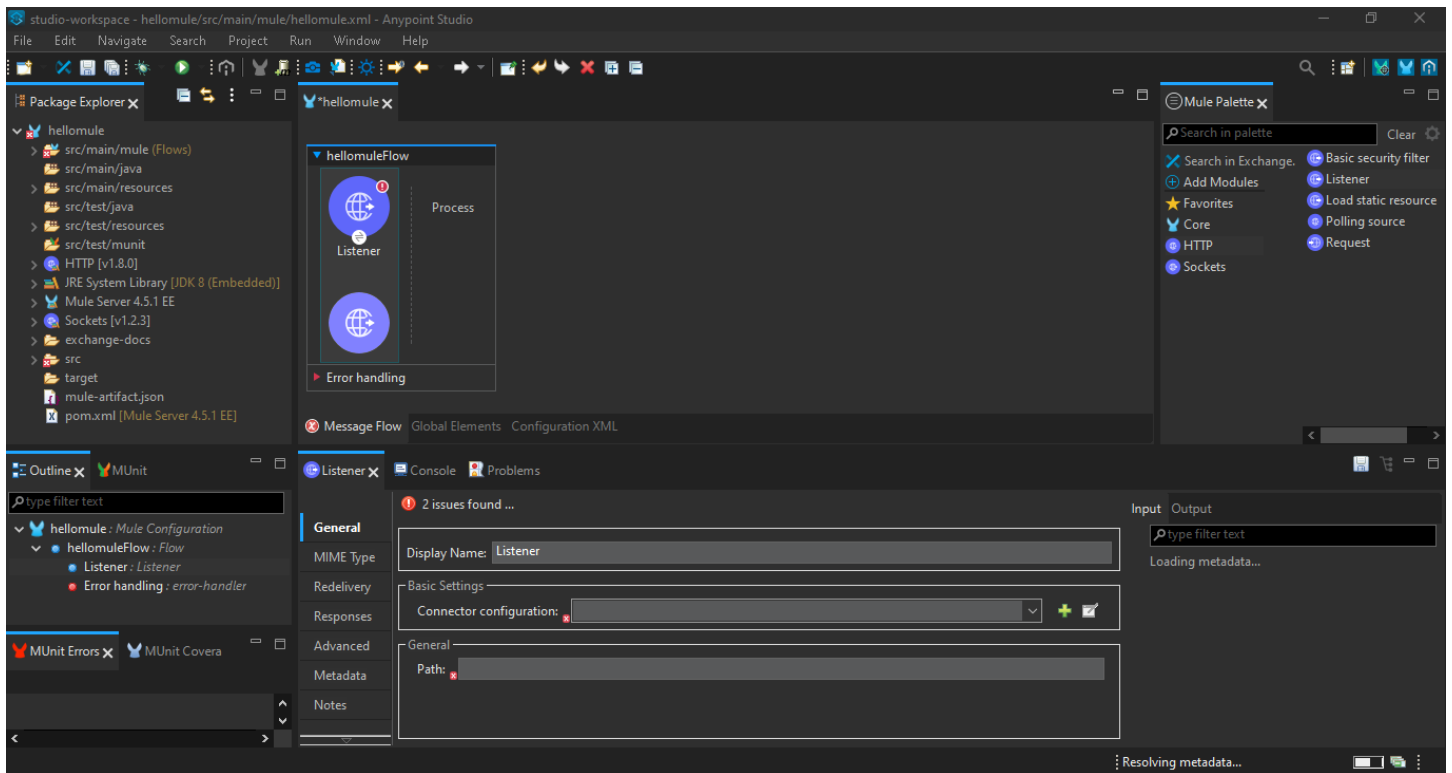


2. Add and configure the HTTP Listener connector

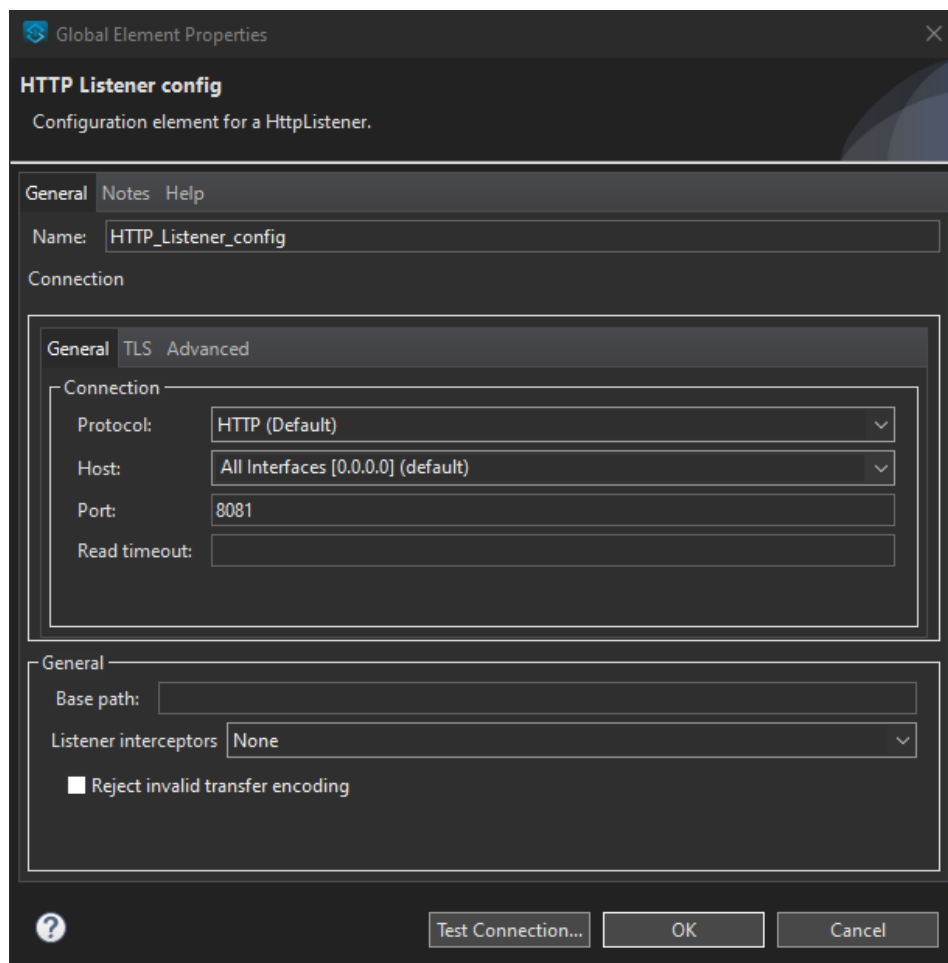
Drag and drop an **HTTP Listener** from the **Mule Palette** to the **canvas**. Anypoint Studio will automatically wrap the connector with a flow once you drag and drop it on the canvas.



An HTTP Listener is an HTTP endpoint that listens for an HTTP request to come to the URL you define. When the Listener receives an HTTP request, the contents inside of the flow will execute in the order that you define.



Next, click on the HTTP Listener connector, and in the Properties Editor below, click on the Green Plus (Add) button. The Green Plus button will create a configuration file under your Global Elements Configuration. Confirm the default values below and click the OK button.



Global Element Properties

HTTP Listener config

Configuration element for a HttpListener.

General Notes Help

Name:

Connection

General TLS Advanced

Connection

Protocol:

Host:

Port:

Read timeout:

General

Base path:

Listener interceptors:

☐ Reject invalid transfer encoding

? Test Connection... OK Cancel

Then under **General > Path** type in: `/hellomule`

Message Flow Global Elements Configuration XML

Listener x Console Problems

General

MIME Type

Redelivery

Responses

Advanced

Metadata

Notes

There are no errors.

Display Name:

Basic Settings

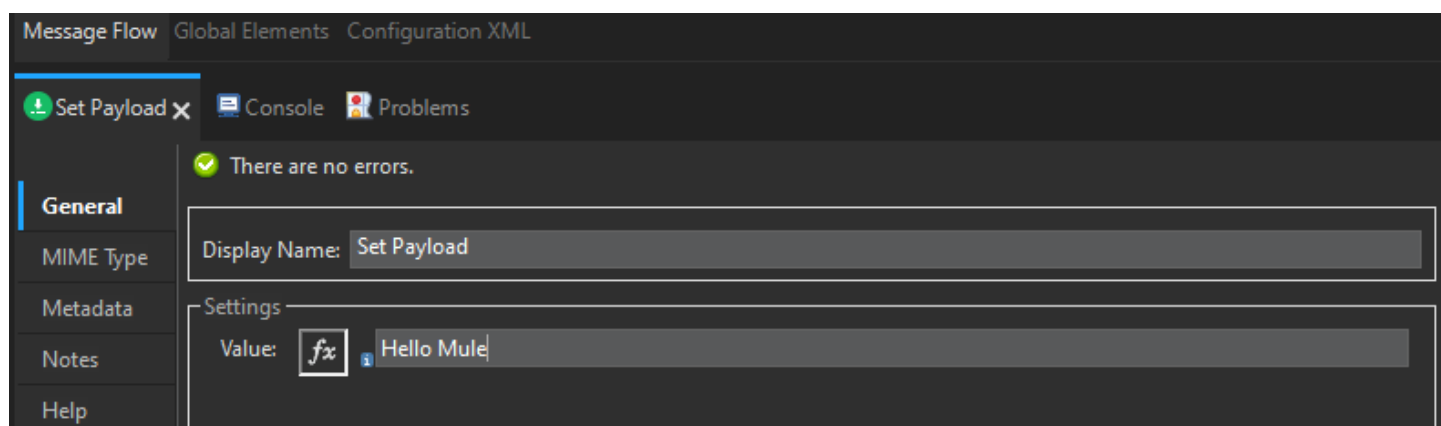
Connector configuration:

General

Path:

The Path you define is going to represent the endpoint that will execute your flow when an HTTP request is made to your HTTP Listener.

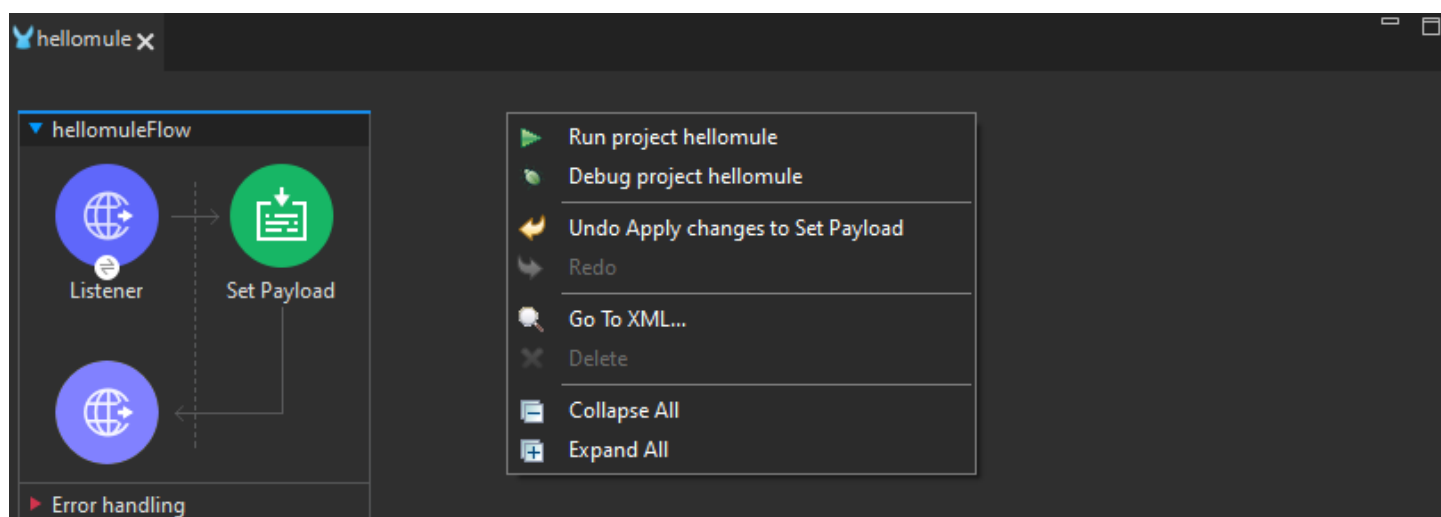
Now, drag the **Set Payload** connector into your flow, inside the **Process** section. You can find the Set Payload connector under the **Core** module in your Mule Palette. Uncheck the fx button and add the string: `Hello Mule`



Now go to **File > Save** to save your project.

3. Run the Mule Application locally

After saving, right-click on your canvas and click **Run project hellomule** to test it live from your local computer!



If you go into your Console view, you can check to see whether your application has been successfully deployed. If it says **DEPLOYED** at the bottom right, then you are ready to use a REST Client to test your first Mule Application.

```
Mule Properties Console Problems
hellomule [Mule Applications] Mule Server 4.5.1 EE (Terminated 16 nov 2023 2:46:23) [pid: 11344]
*****
* - - + DOMAIN + - - * - - + STATUS + - - *
*****
* default * DEPLOYED *
*****

*****
* - - + APPLICATION + - - * - - + DOMAIN + - - * - - + STATUS + - - *
*****
* hellomule * default * DEPLOYED *
*****

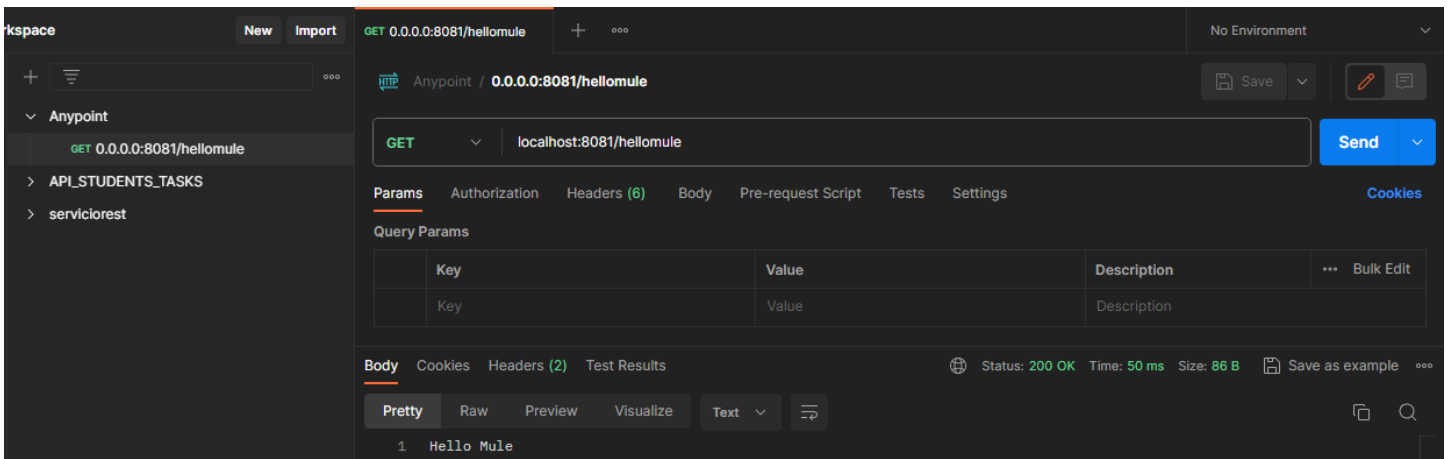
INFO 2023-11-16 02:49:34,501 [[MuleRuntime].uber.02: [hellomule].uber@org.mule.runtime.module.extension.internal.runtime.source.ExtensionMessageSource.1
```

4. Testing tool with POSTMAN – ID Client

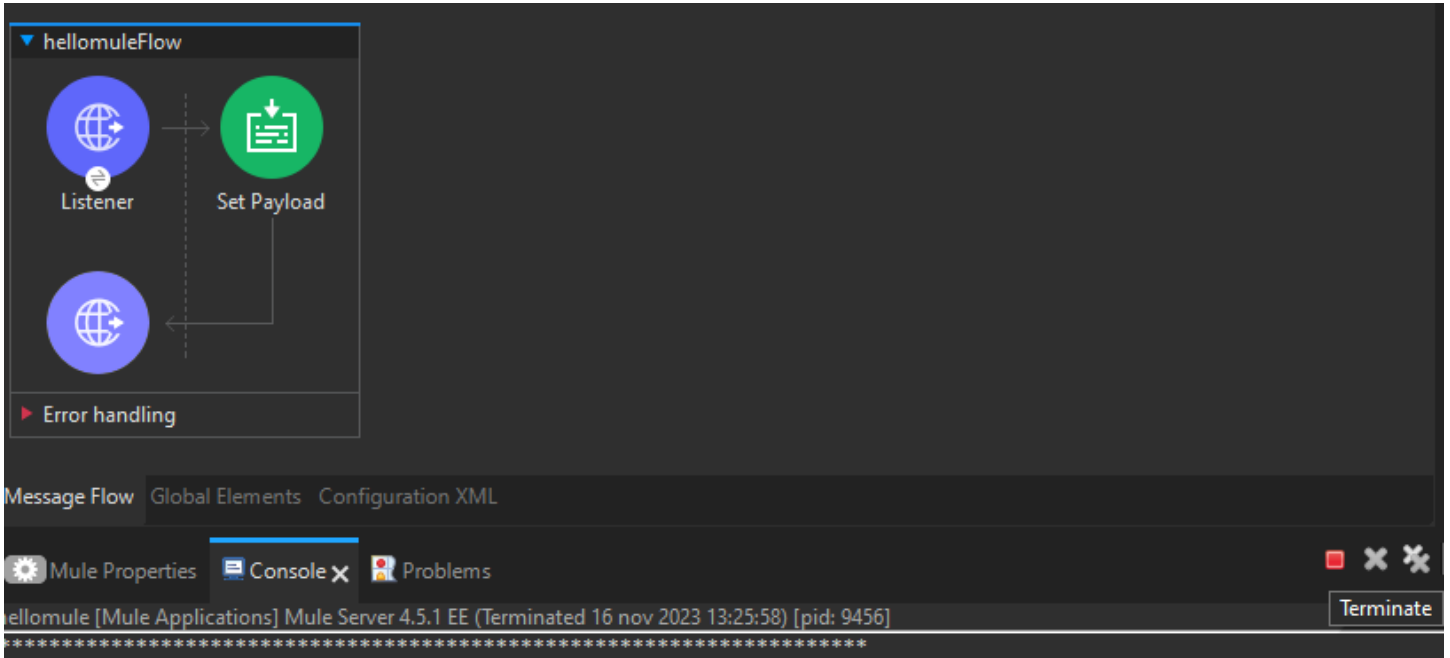
Open up your REST Client (POSTMAN) of choice. Type in the URL of your HTTP endpoint and path added after the URL. For this demo, you will use the address: `http://0.0.0.0:8081/hellomule`

You can replace `0.0.0.0` with `localhost`

Click the **Send** button and you should get `Hello Mule` as a **200 OK Response**

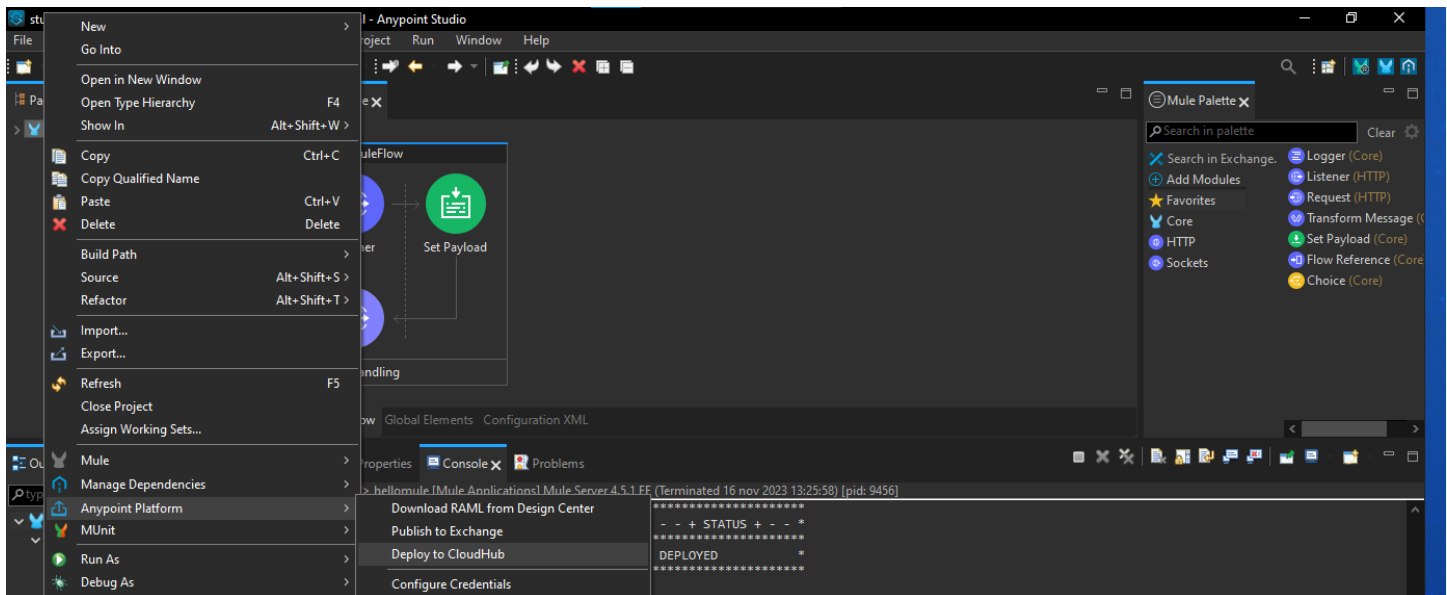


To stop the project from running locally, you can right-click again on the canvas and click **Stop project hellomule**.



5. Deploy the Mule Application to CloudHub

Now that you have learned how to deploy and test locally, let's deploy our application to CloudHub. CloudHub will issue your application with a publicly accessible endpoint URL after you complete the deployment process. Right-click on your project in your Package Explorer, and click on **Anypoint Platform > Deploy to CloudHub**.



First, you will need to log in with your Anypoint Platform credentials. After that, choose the Sandbox environment then choose a unique name for your application. If your application has a red x next to it, it means someone else has deployed with that application name. You just have to give it a new unique name.

A screenshot of the 'User login' dialog box in Anypoint Studio. The dialog has a title bar 'User login' and a 'Configure' button. The main content area has a blue header with the MuleSoft logo and 'from Salesforce'. Below the header, it says 'Sign in'. There are two input fields: 'Username' and 'Password'. The 'Username' field has a red border and a red 'Required' message below it. Below the 'Password' field is a blue 'Sign in' button. At the bottom, there are links for 'Forgot your credentials?' and 'Use custom domain'. At the very bottom, it says 'Login expires after: 7 days' and a 'Cancel' button.

Choose Environment

Design

Sandbox

When your application name has a green check next to it, click the Deploy Application button. It may take a few minutes to fully deploy to CloudHub. After that, click on the following button to log in to Anypoint Platform.

SANDBOX

Deploying Application

hellomule-ex1 ✓

Deployment Target

Shared Space US East (Ohio)

CloudHub 2.0

Application File

Runtime Manager

Deploying hellomule-ex1 to Cloudhub-US-East-2

You may close this window at any time.

Open in Browser

Close Window

Runtime Manager

Application hellomule-ex1 successfully deployed to Cloudhub-US-East-2

You may close this window at any time.

Open in Browser

Close Window

Navigate to Runtime Manager and find your application. Click on your application's name to open the Dashboard.

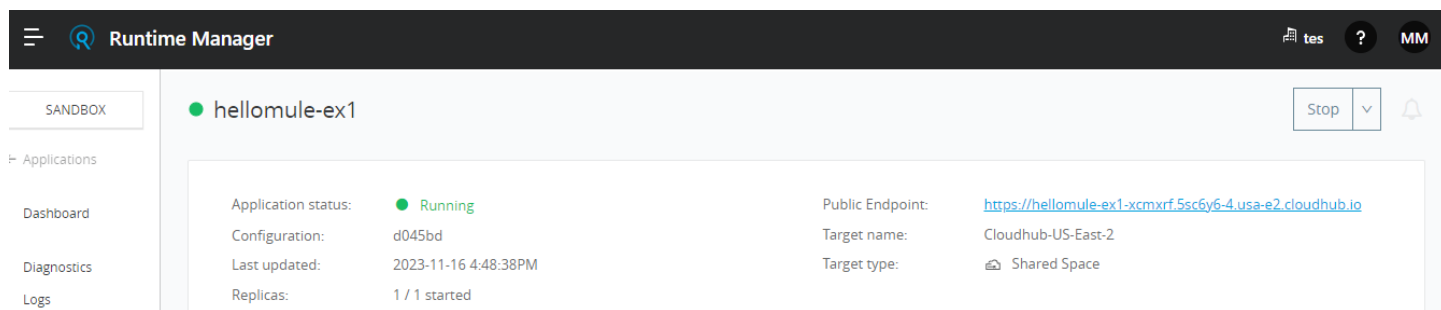
All Applications (1)

Update Available (0)

Name ^	Target Name	Target Type	Status	Runtime Version	Date Modified
hellomule-ex1	Shared Space	CloudHub 2.0 NEW	Running	4.5.2.1e	2023-11-16 16:48:38

Once your application is fully deployed, copy and paste your application's URL highlighted in blue into your REST Client and add your Endpoint Path after the URL.

In this case, it would be `hellomule-ex1-xcmxrf.5sc6y6-4.usa-e2.cloudhub.io/hellomule`



Runtime Manager

hellomule-ex1

Stop

Application status: Running

Configuration: d045bd

Last updated: 2023-11-16 4:48:38PM

Replicas: 1 / 1 started

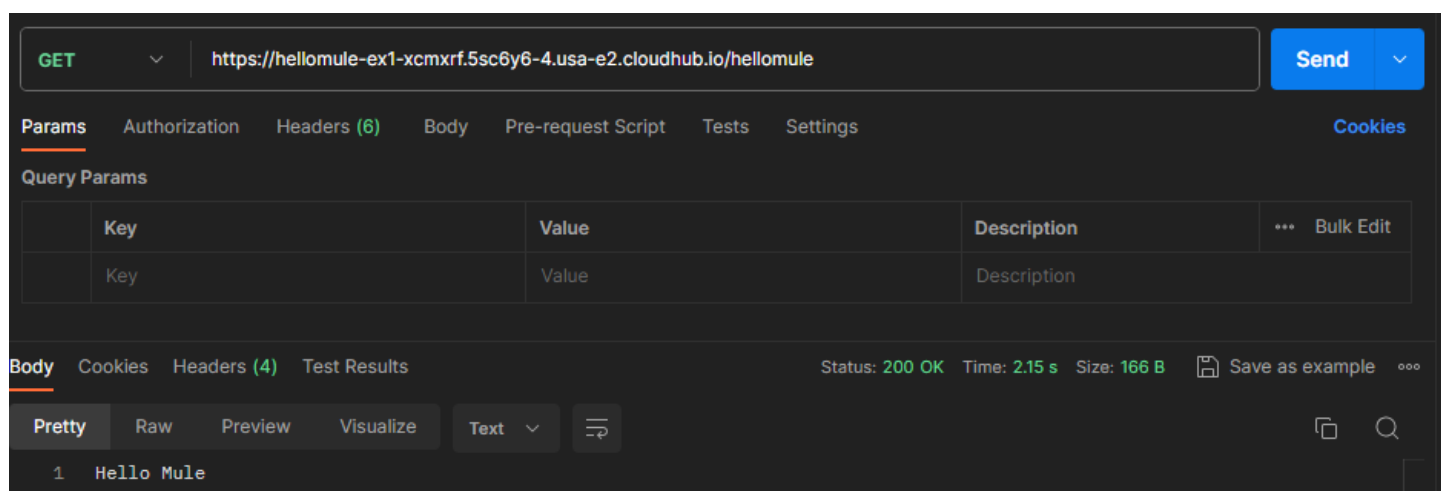
Public Endpoint: <https://hellomule-ex1-xcmxrf.5sc6y6-4.usa-e2.cloudhub.io>

Target name: Cloudhub-US-East-2

Target type: Shared Space

6. Testing with POSTMAN in the deployed API

When you send a request to your CloudHub endpoint, you should get *Hello Mule* as a **200 OK Response**.



GET <https://hellomule-ex1-xcmxrf.5sc6y6-4.usa-e2.cloudhub.io/hellomule> Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (4) Test Results Status: 200 OK Time: 2.15 s Size: 166 B Save as example

Pretty Raw Preview Visualize Text

1 Hello Mule

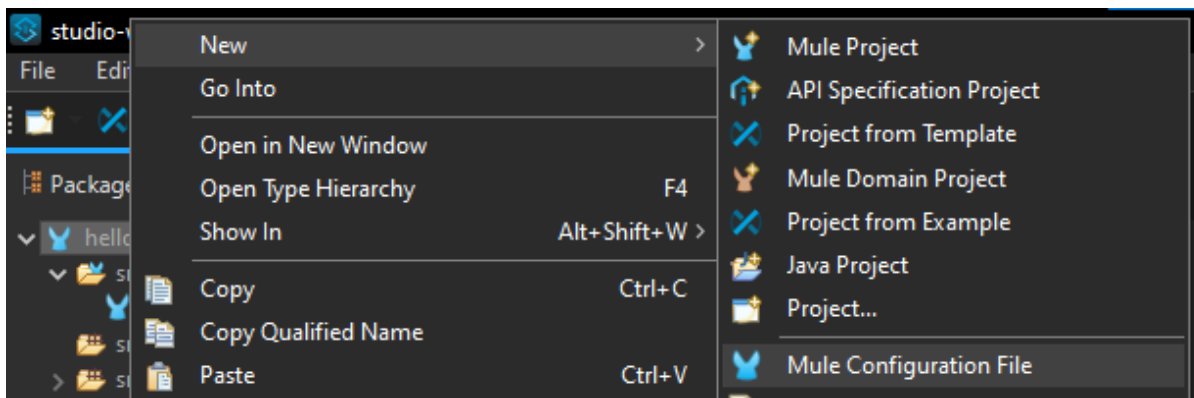
Ejercicio 2: Cómo configurar tus archivos globales de elementos y propiedades en Anypoint Studio.

Utiliza el archivo de propiedades para mantener y referenciar datos sensibles por separado al código generado.

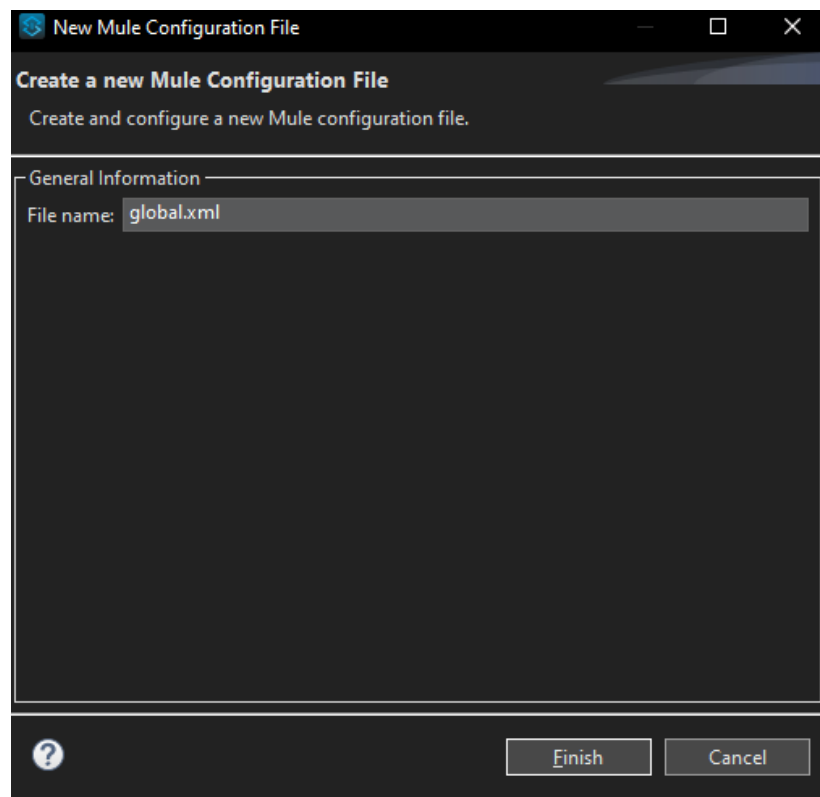
1. Create a global.xml file to keep the global elements

It is a best practice to create a new **global.xml** file to keep all our global elements stored there. This is useful when we have more XML files and we don't want to be looking for our global elements in each file.

Right-click on the **src/main/mule** folder. Select **New > Mule Configuration File**.

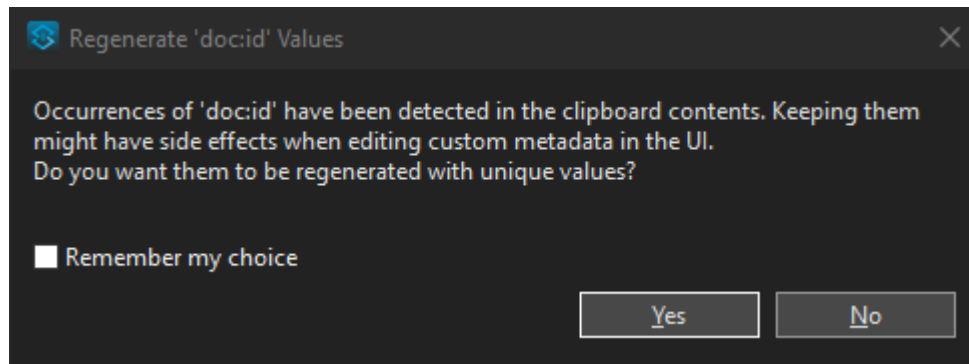


Add the name `global.xml` and click Finish.



Go back to the **hellomule.xml** file and switch to the Configuration XML view. Locate the code and cut everything between the tags. `http:listener-config`

Go to the **global.xml** file and switch to the Configuration XML view. Paste this code **inside the mule tags**. If you get an alert regarding the 'doc:id' values, you can select **Yes** to regenerate the IDs of each component. Unless you purposely want to paste the exact same components, it is recommended to regenerate the IDs when copy/pasting code. Don't select the checkbox in case you need to copy and paste the exact same IDs in the future.

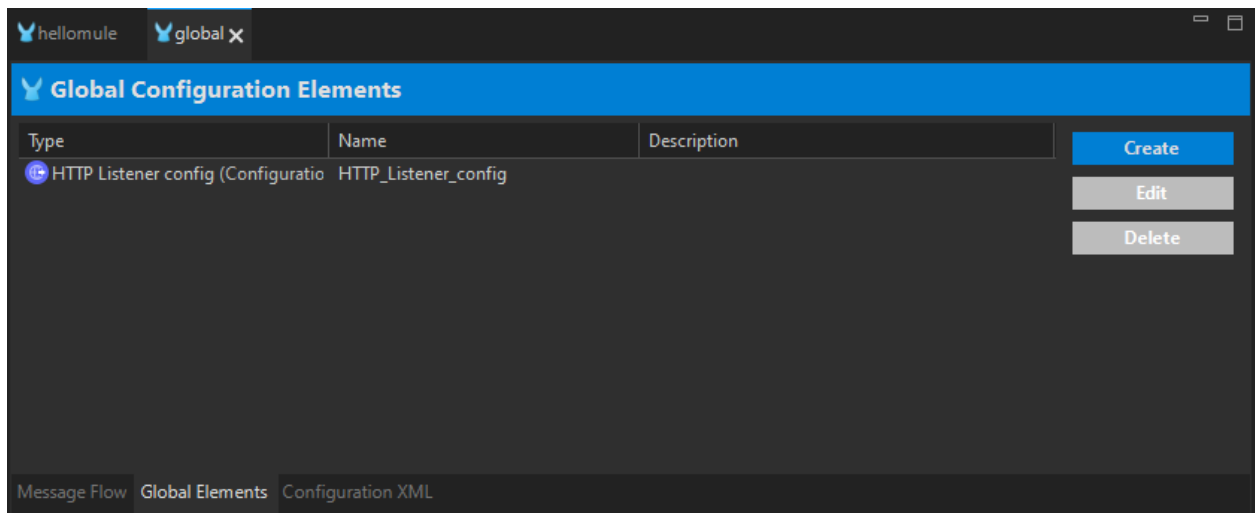
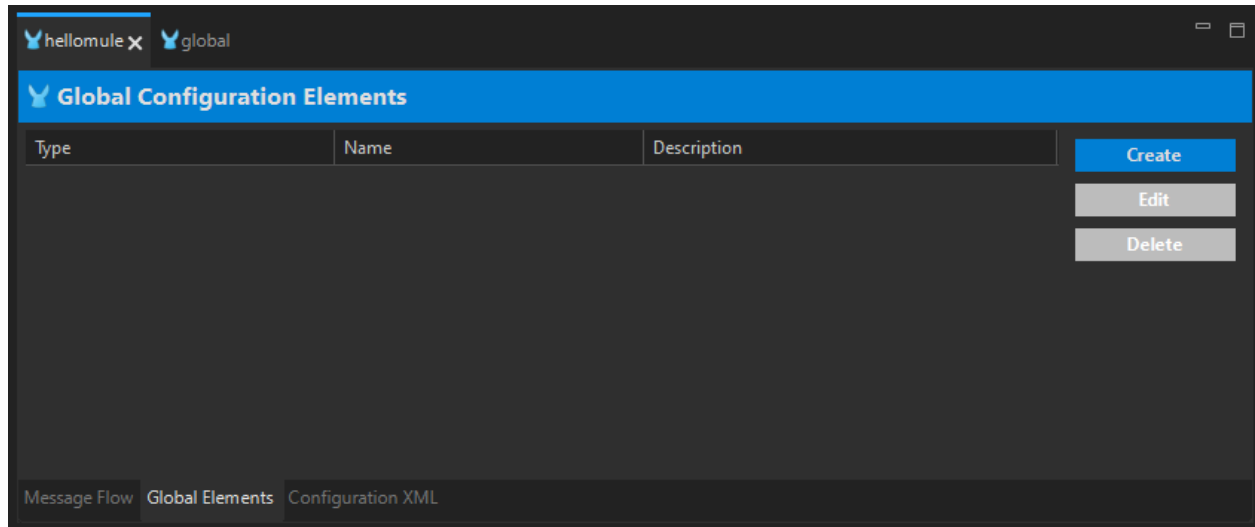


Notice how you may be getting an error at this point. This is because the **hellomule.xml** file has not been saved and Studio thinks you now have two global elements with the same name.

You can save each individual file separately, or you can click on the **Save All** button at the top left of the screen.

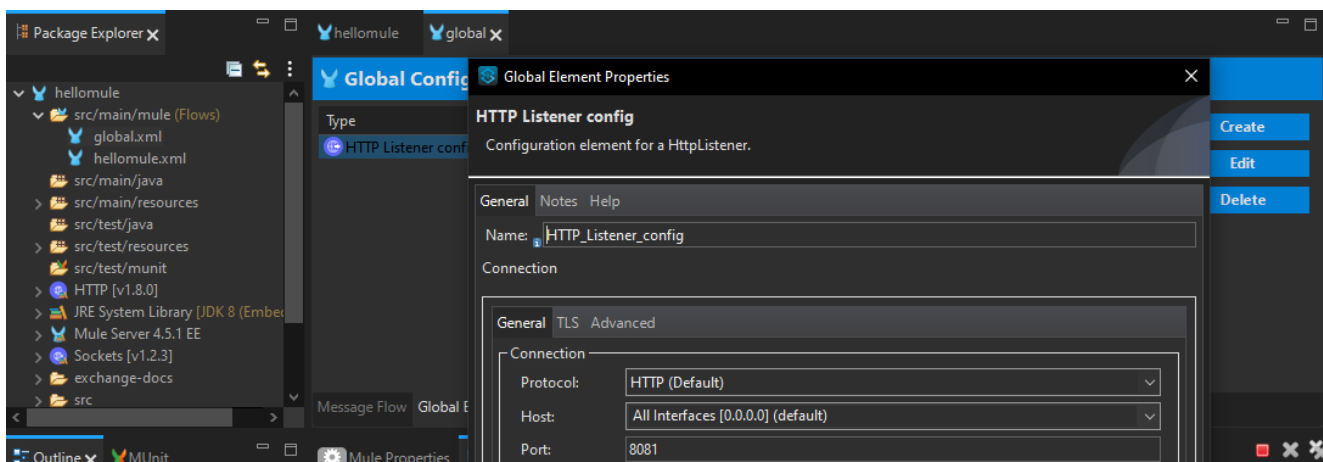
The error disappears after saving both files.

If you switch to the Global Elements tab for both files, you will now notice that the **HTTP_Listener_config** is no longer present in the **hellomule.xml** file and it was moved to the **global.xml** file.



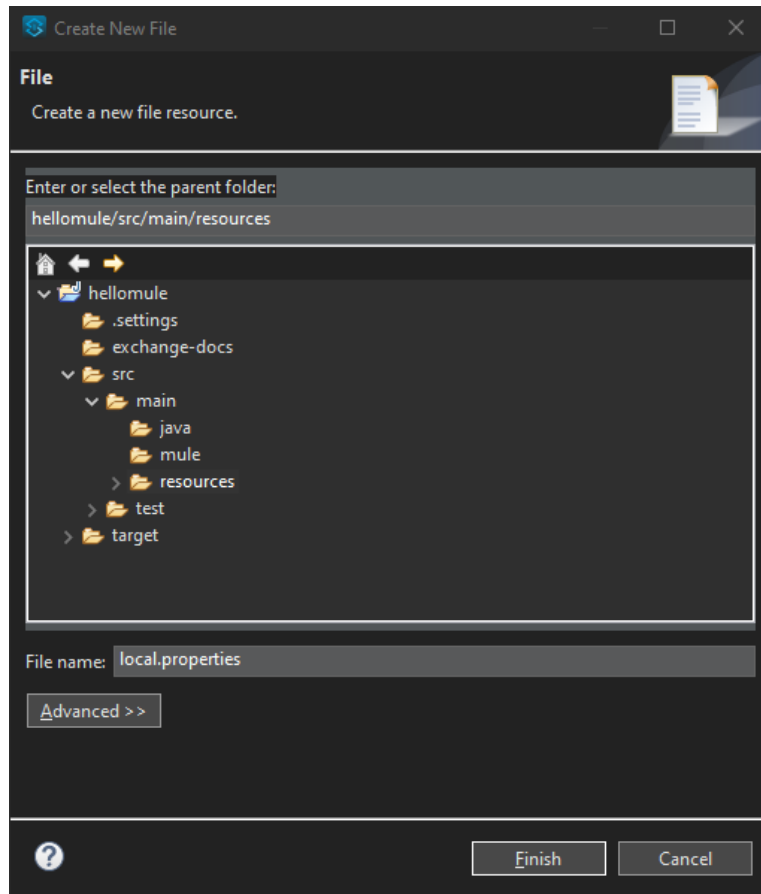
2. Externalize the hardcoded values into properties

In the **global.xml** file, switch to the Global Elements view. Select the **HTTP_Listener_config** element and click on the **Edit** button to the right or double-click it. Remember the default values for the Host and the Port? These are hardcoded values. This means that whenever we want to change them, we have to find the correct element and edit it. Maybe it's not hard to do right now because we only have one global element, but this becomes an issue in bigger projects.

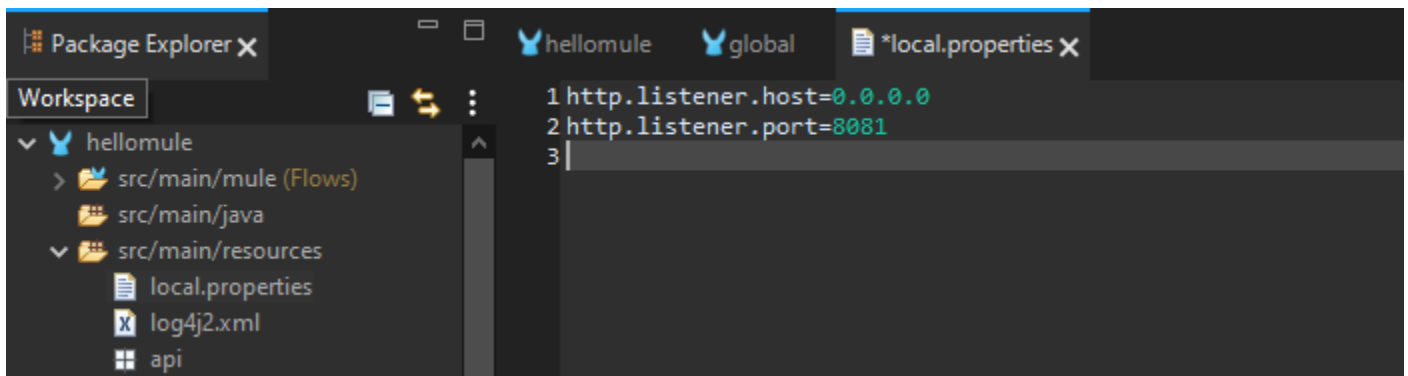


To avoid having hardcoded values in our code, it is a best practice to externalize them into properties files. Right-click on the **src/main/resources** folder and select **New > File**.

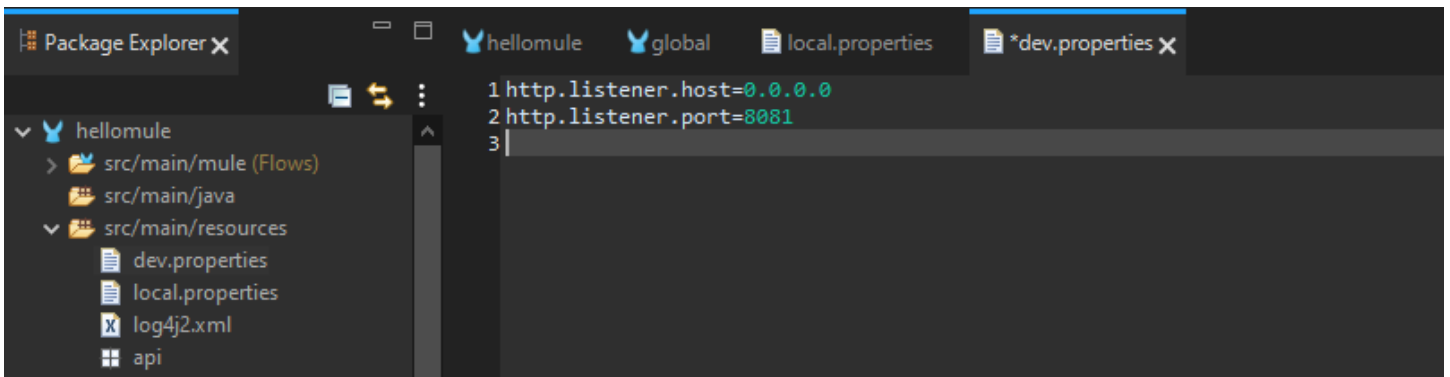
Create a **local.properties** file under src/main/resources.



Inside this new file, copy and paste the following properties and save the file:

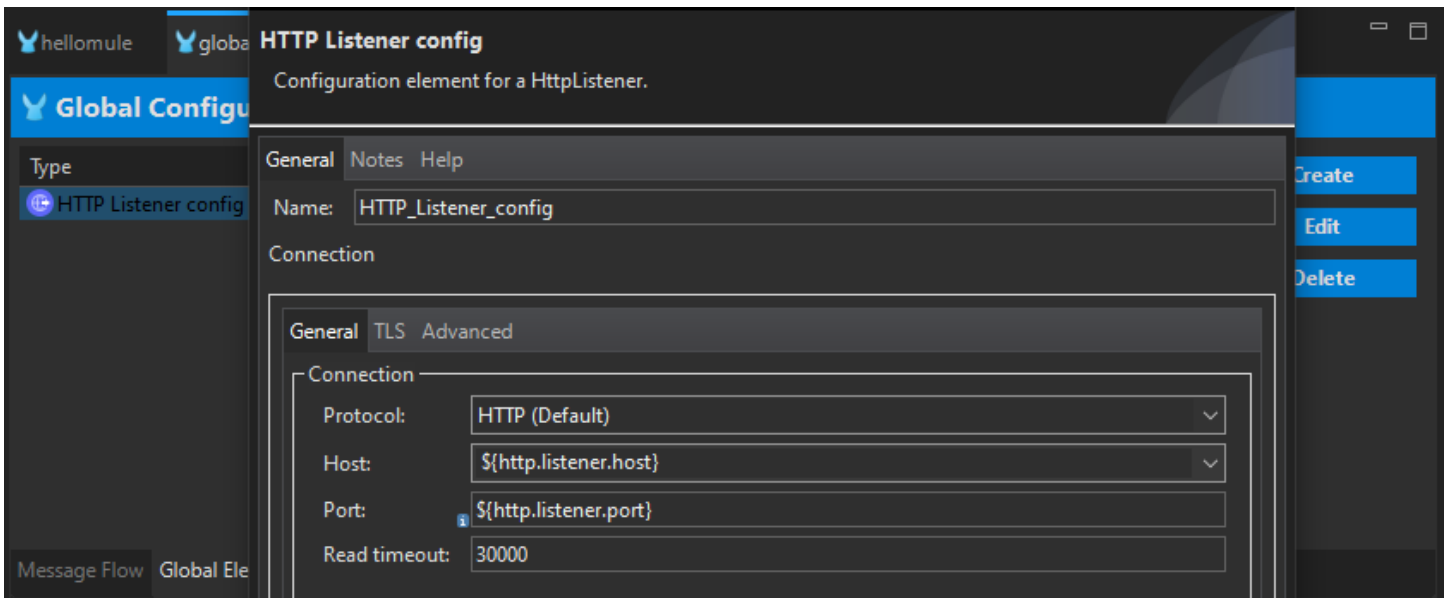


Repeat the same steps for a **dev.properties** file. This helps us separate the properties per environment. When we run our Mule app on our local computer, we'll be using the local.properties file. When we deploy our application to CloudHub, we'll be using the dev.properties file. In this case, both properties have the same values. But it's a good practice to separate your properties files by environment.



Go back to the **global.xml** file and switch to the **Global Elements** view. Select the **HTTP_Listener_config** element and click on **Edit**. In order to refer to a property from any connector configuration, you need to use the following syntax: `${your.property.name}`

Replace each hardcoded value with the corresponding property, using the previous syntax. You should end up with something like this:

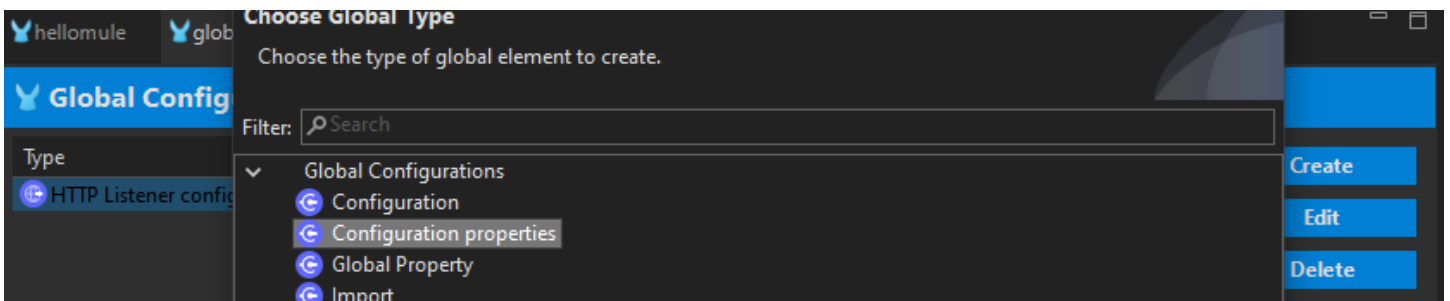


Click **OK** and save all files.

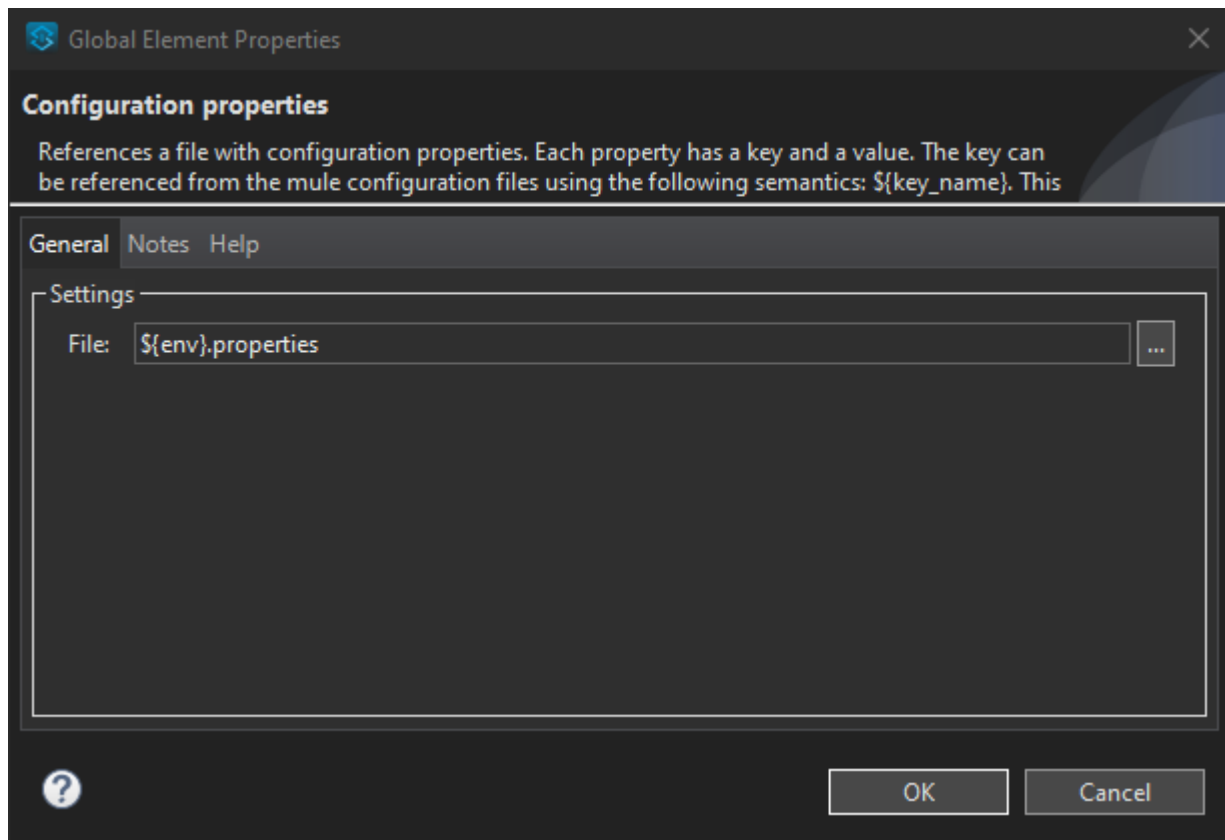
3. Set up the Configuration Properties global element

Adding the properties file is not enough to let the Mule Application know where to look for them. If you try to run the app without this configuration, it will throw an error stating that it wasn't able to find your properties.

In your **global.xml** file, switch to the **Global Elements** view. Click on **Create** to add a new global element. Search for **Configuration properties**. Select the element and click **OK**.

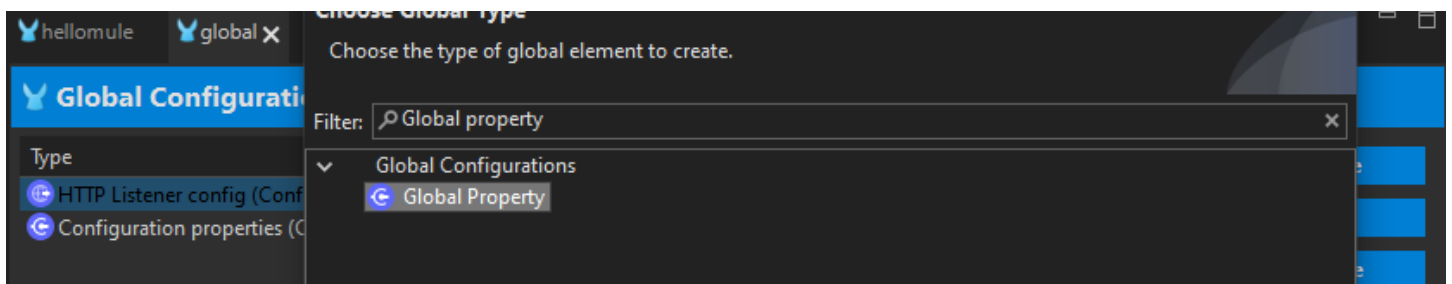


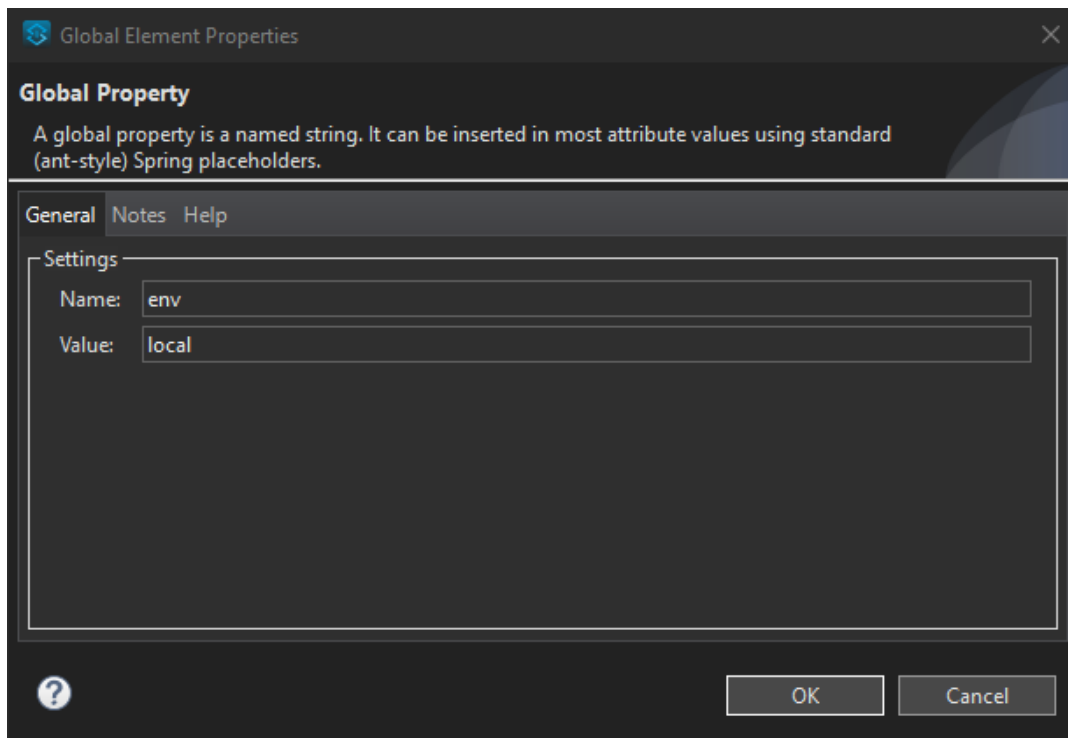
Because we're not using static properties files, we need to be able to switch between files depending on the environment. Remember that the `${}` syntax is used to reference properties. Let's add the following properties file: `${env}.properties`



You should now have two global elements in the **global.xml** file. We still need to add the **env** property that we just referenced in the previous step. There are two ways to do that: 1) adding it to the environment variables at runtime, or 2) creating a global property in the global configuration elements. The first way is preferred when you don't want to save the property in the source code. We'll see how to do this in the next tutorial for secure properties. Since this is not a secure property, we'll add it to the global configuration elements.

Click **Create** and select **Global Property**. The name should be **env** and the value **local**.

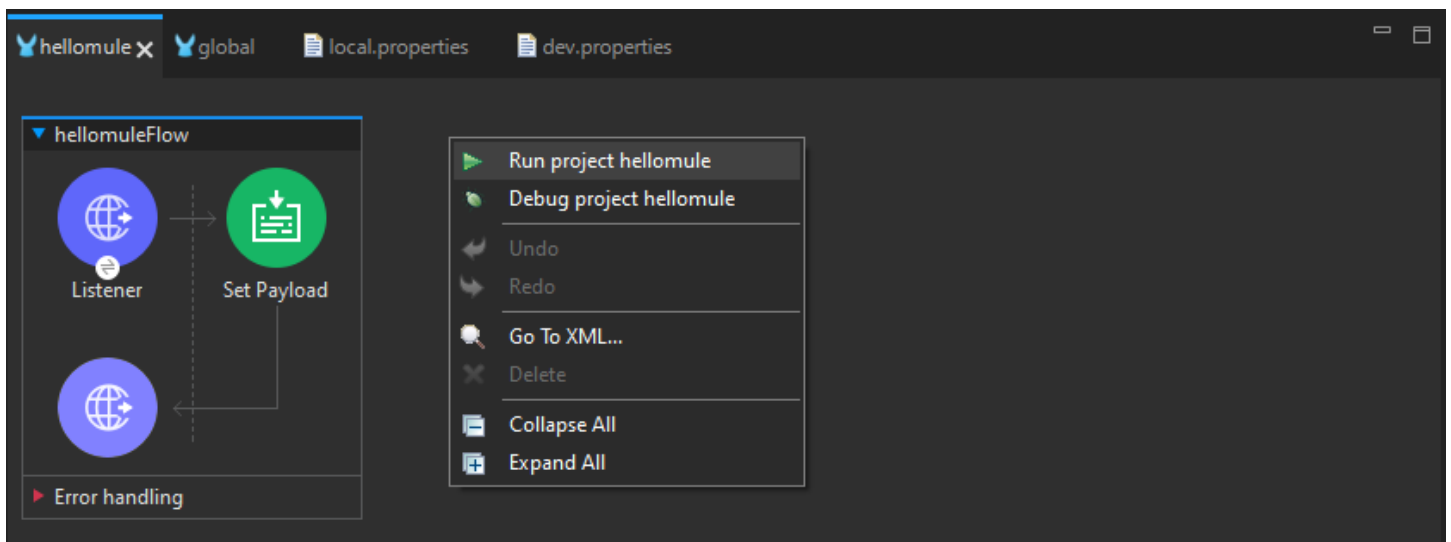




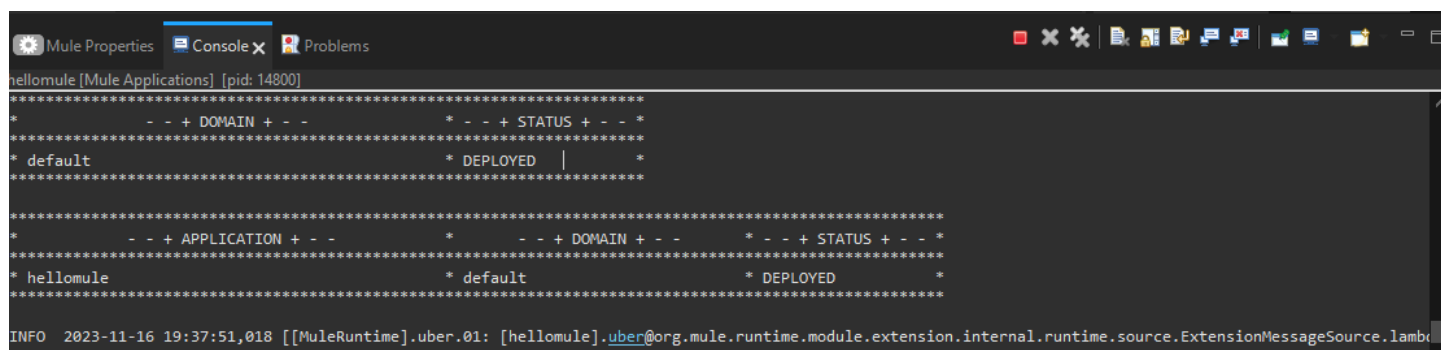
Click **OK** and save all files.

4. Verify the Mule project is still working as expected

Make sure you saved all the files. Go to any of the XML files and switch to the **Message Flow** view. Right-click inside the canvas and select **Run project hellomule**.



Verify that there are no errors in the Console and the Status of the application is **DEPLOYED**.



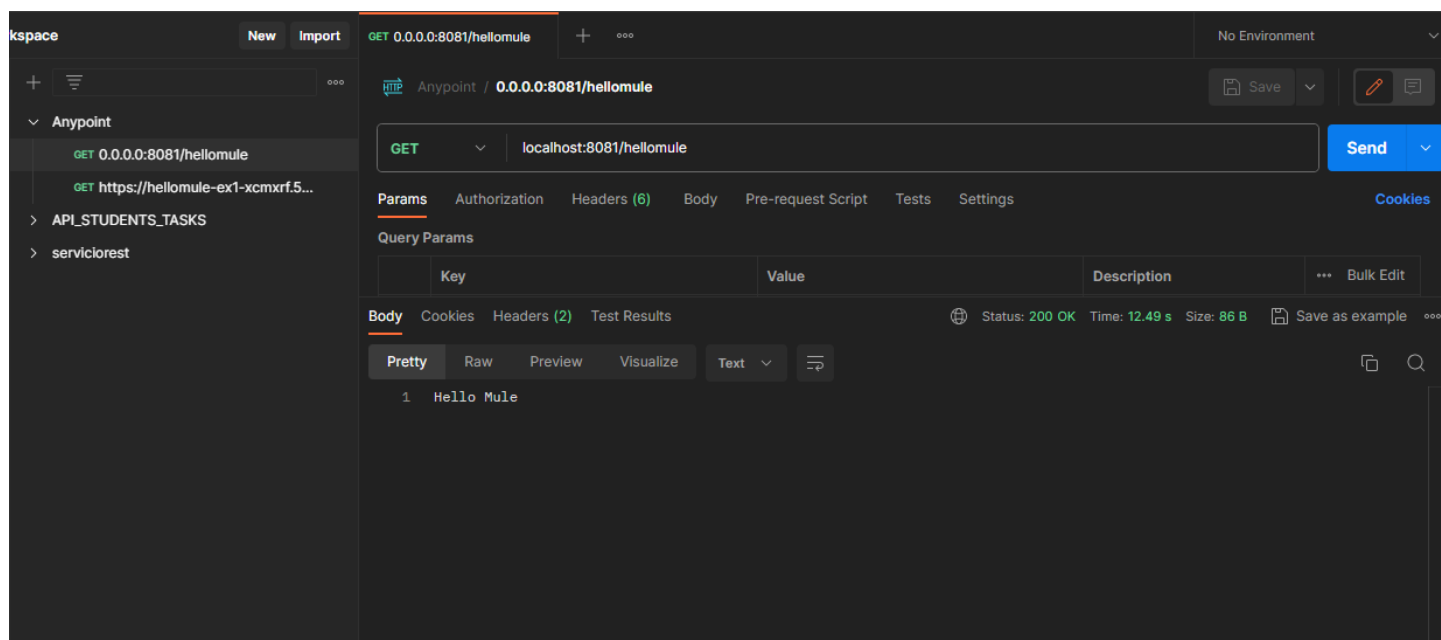
The screenshot shows the Mule IDE Console with the following output:

```
*****  
* - - + DOMAIN + - - * - - + STATUS + - - *  
*****  
* default * DEPLOYED | *  
*****  
  
*****  
* - - + APPLICATION + - - * - - + DOMAIN + - - * - - + STATUS + - - *  
*****  
* hellomule * default * DEPLOYED *  
*****  
  
INFO 2023-11-16 19:37:51,018 [[MuleRuntime].uber.01: [hellomule].uber@org.mule.runtime.module.extension.internal.runtime.source.ExtensionMessageSource.lambda$
```

5. Testing with POSTMAN

In the REST Client and send a request to the local application: `localhost:8081/hellomule`

You should get back the `Hello Mule` response and a **200 OK** status.



To stop the project from running locally, you can right-click again on the canvas and click **Stop project hellomule**.

Ejercicio 3: Cómo proteger las propiedades antes de la implementación en Anypoint Studio.

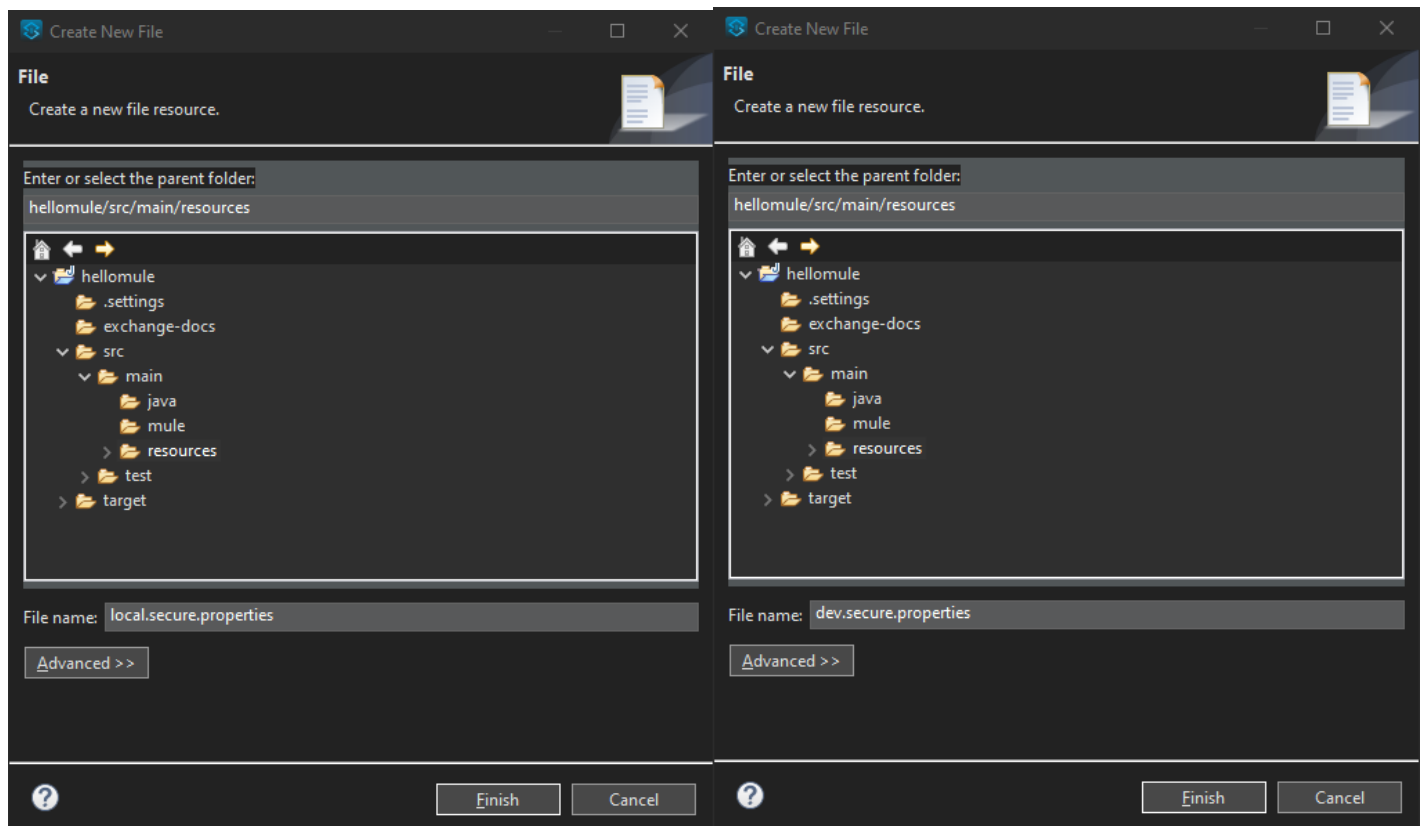
Crea un archivo de propiedades seguras para proteger datos sensibles que representan un riesgo mantenerlos en claro.

1. Create your environments' secure properties files

Whenever you are storing global variables, unique tokens or keys, or login credentials, you should always store them in a properties file. Storing all of your variables in a single file will help keep your project organized, and if you ever need to make a change to an existing system or migrate the integration, you won't have to make any manual modifications to your existing code.

In order to have a better organization of your properties, it is a best practice to separate your secure properties in files per environment. You can create a **local.secure.properties** file for your local settings, a **dev.secure.properties** for your dev environment, a **qa.secure.properties** for your testing environment, etc.

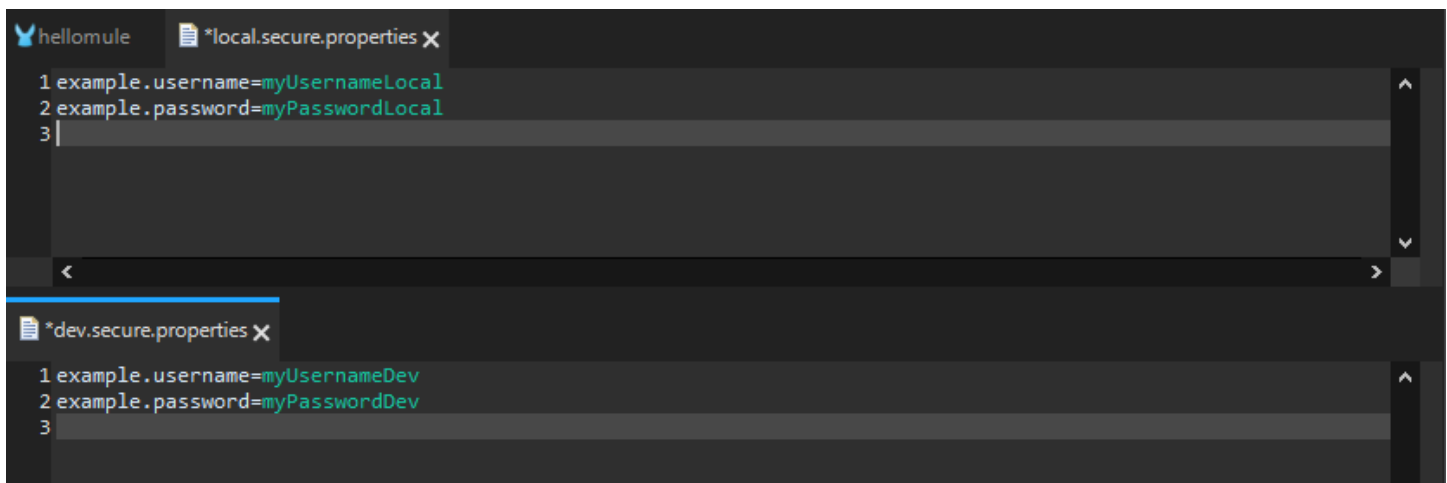
To create a local.secure.properties file, right-click on **src/main/resources** and click on **New > File** and name the file `local.secure.properties`



Now that you have successfully created the file, double click on your **local.secure.properties** file to add a new entry.

After you have entered your private credentials into the **local.secure.properties** file, repeat the same steps for a **dev.secure.properties** file.

Add the properties:

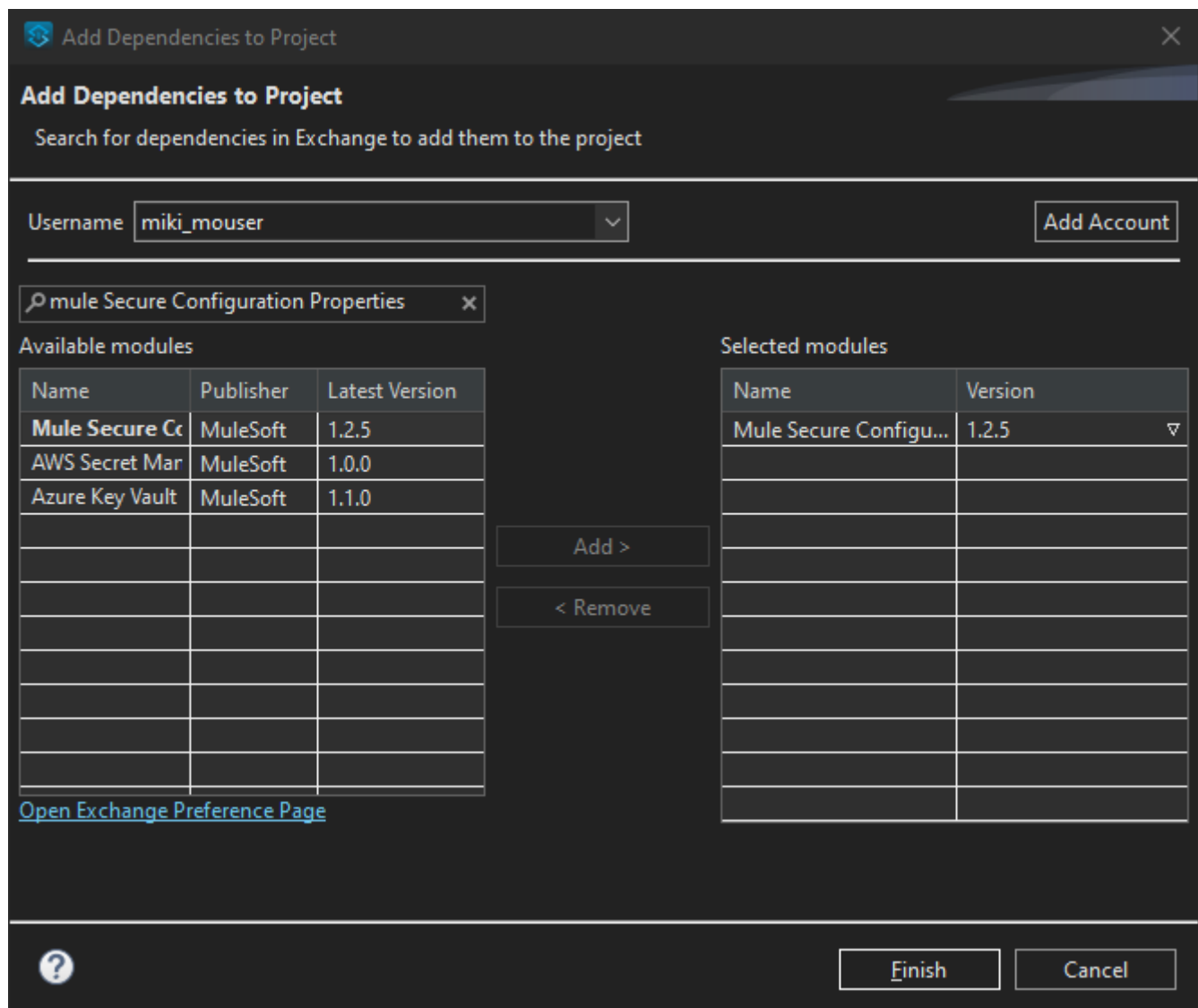


The screenshot shows two IDE windows. The top window is titled 'hellomule' and contains a file named '*local.secure.properties'. It has three lines of text: '1 example.username=myUsernameLocal', '2 example.password=myPasswordLocal', and '3'. The bottom window is titled '*dev.secure.properties' and also has three lines of text: '1 example.username=myUsernameDev', '2 example.password=myPasswordDev', and '3'.

```
1 example.username=myUsernameLocal
2 example.password=myPasswordLocal
3

1 example.username=myUsernameDev
2 example.password=myPasswordDev
3
```

The next step is to search Exchange for the **Mule Secure Configuration Properties** module. In your **global.xml** file, select the **Search in Exchange** option from the Mule Palette. Search for the module and click on **Add >**.



The screenshot shows the 'Add Dependencies to Project' dialog box. At the top, it says 'Add Dependencies to Project' and 'Search for dependencies in Exchange to add them to the project'. Below this, there is a 'Username' field with the value 'miki_mouser' and an 'Add Account' button. A search bar contains the text 'mule Secure Configuration Properties'. Below the search bar, there are two tables: 'Available modules' and 'Selected modules'. The 'Available modules' table has columns 'Name', 'Publisher', and 'Latest Version'. It lists 'Mule Secure Configuration Properties' (MuleSoft, 1.2.5), 'AWS Secret Manager' (MuleSoft, 1.0.0), and 'Azure Key Vault' (MuleSoft, 1.1.0). The 'Selected modules' table has columns 'Name' and 'Version'. It lists 'Mule Secure Configuration Properties' (1.2.5). Between the tables are 'Add >' and '< Remove' buttons. At the bottom left, there is a link 'Open Exchange Preference Page'. At the bottom right, there are 'Finish' and 'Cancel' buttons.

Add Dependencies to Project

Search for dependencies in Exchange to add them to the project

Username: miki_mouser Add Account

Search: mule Secure Configuration Properties

Name	Publisher	Latest Version
Mule Secure Configuration Properties	MuleSoft	1.2.5
AWS Secret Manager	MuleSoft	1.0.0
Azure Key Vault	MuleSoft	1.1.0

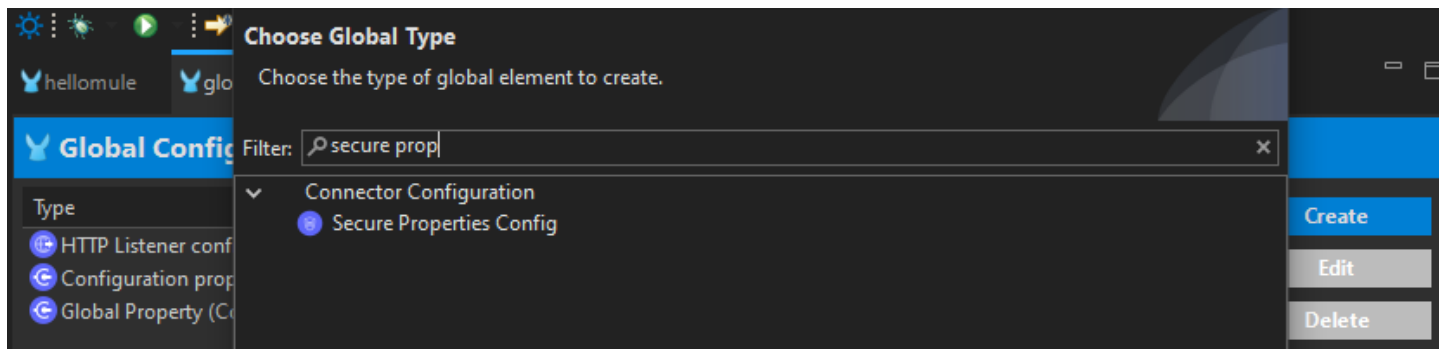
[Open Exchange Preference Page](#)

Name	Version
Mule Secure Configuration Properties	1.2.5

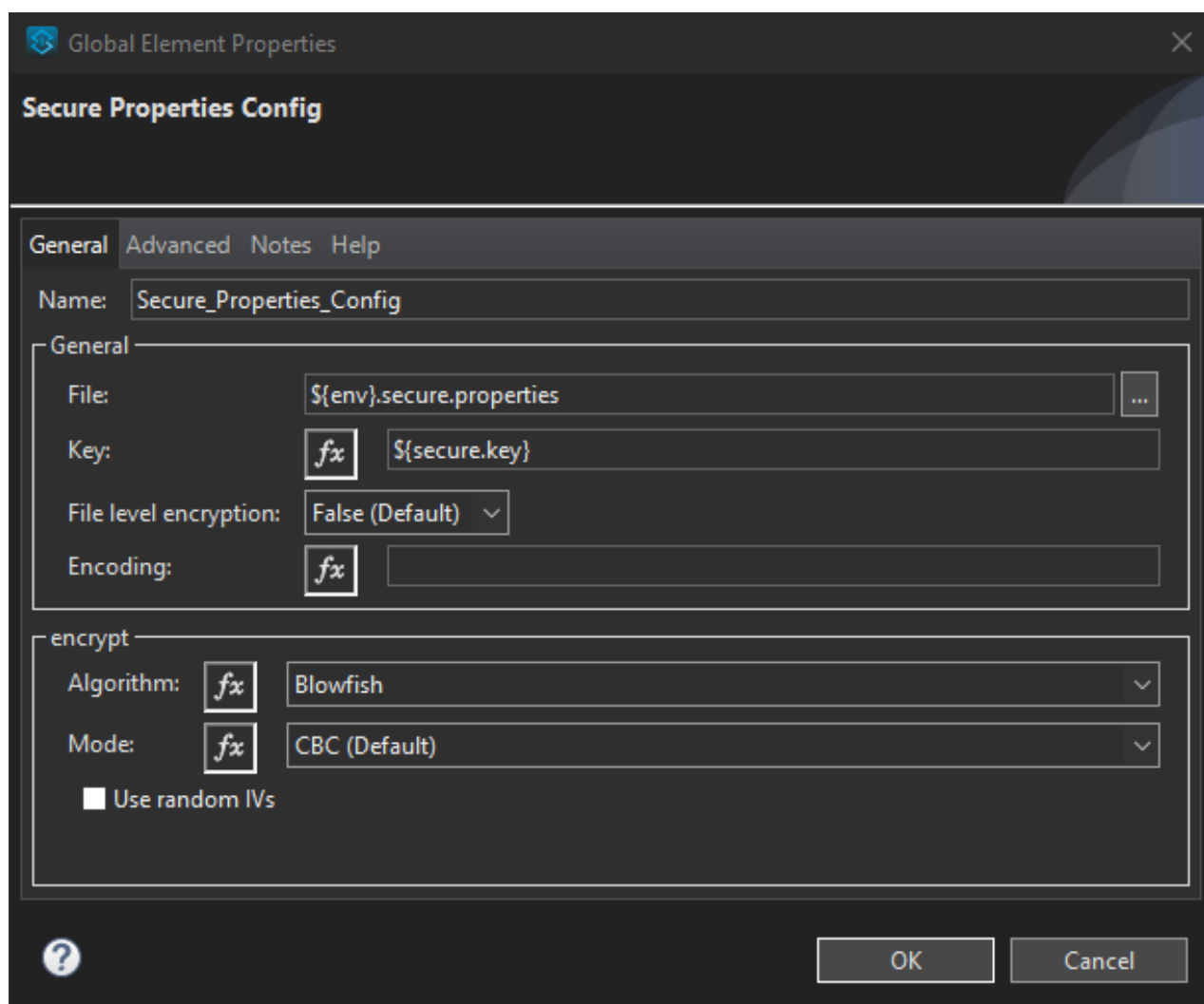
Add > < Remove

? Finish Cancel

Once you import your Secure Properties module, go to the **Global Elements** view. Click the **Create** button and create a **Secure Properties Config**. Under File, type `${env}.secure.properties`. This is a dynamic file. The Mule App will read the **env** property first and then the appropriate file based on it. In this case, **local.secure.properties**.



Under Key, type and select **Blowfish** as the Algorithm. The secure.key value will also be a property, but we don't want to add this property in our code directly because that is a possible security breach. We will be passing the key every time we start the application, either from our local computer or from CloudHub.



Save changes in all your files.

2. Set up a logger to read your secure credentials

We'll set up a logger component to output the credentials into the console. Go to the **hellomule.xml** file and add a **Logger** component from the Mule Palette - you can find it under the **Core** module.



In the configuration, click on the **fx** button and then on the Show Graphical View button to the right. This will open a bigger view to add your DataWeave code. You can use the buttons next to the blue **Done** button to switch between the different views available.

The screenshot shows the Logger configuration dialog. The 'Display Name' is 'Logger'. The 'Message' field is set to 'fx' and the 'Level' is set to 'INFO (Default)'. The 'Show Graphical View' button is visible.

The screenshot shows the Logger configuration dialog with the 'Message' field set to 'fx'. A syntax error is displayed: 'Sorry, syntax errors prevented the mappings display'. The 'Done' button is visible.

Add this code to output a String with the two secured properties:

The screenshot shows the Logger configuration dialog with the 'Message' field set to the following DataWeave code:

```
1 output application/java
2 ---
3 'Username: ' ++ Mule::p('secure:example.username')
4 ++ ' - ' ++
5 'Password: ' ++ Mule::p('secure:example.password')
6
```

Note the `secure::` that we added before the property name. This tells the application it needs to decrypt these values before using them. Save all files.

3. Encrypt your properties files' values

Download the Secure Properties Tool Jar file from the official [MuleSoft documentation](#). Open a terminal or command line prompt in the directory where this Jar file is located and run the following:

In Linux:

```
java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool \  
string \  
encrypt \  
Blowfish \  
CBC \  
MyMuleSoftKey \  
"myUsernameLocal"
```

In Windows:

```
java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool ^  
string ^  
encrypt ^  
Blowfish ^  
CBC ^  
MyMuleSoftKey ^  
"myUsernameLocal"
```

Or also in Windows you can only in one line:

```
java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfish CBC  
MyMuleSoftKey "myUsernameLocal"
```

Command explanation:

- **string:** Indicates that the type of property to be encrypted is a string.
- **encrypt:** Specifies that an encryption operation is to be performed.
- **Blowfish:** Indicates the encryption algorithm to be used (in this case, Blowfish).
- **CBC:** Cipher Block Chaining mode of block cipher operation.
- **"MyMuleSoftKey":** The value to be encrypted. In your case, this would be the value you want to protect, such as a password.
- **mulesoft:** The key used for encryption and decryption. Be sure to change this to a secure key and keep it secure.

This will return the encrypted value of **myUsernameLocal**, which is the property we are using as an example. Note that the key we're using to encrypt the property is **MyMuleSoftKey**. We will be using this value as the **secure.key** property to decrypt the values at runtime.

```
C:\Windows\System32\cmd.exe
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\HP\Downloads>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool
string encrypt Blowfish CBC MyMuleSoftKey "myUsernameLocal"
HbsuWJRjiubchmzQREGdsA==

C:\Users\HP\Downloads>
```

All:

```
C:\Windows\System32\cmd.exe
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\HP\Downloads>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool
string encrypt Blowfish CBC MyMuleSoftKey "myUsernameLocal"
HbsuWJRjiubchmzQREGdsA==

C:\Users\HP\Downloads>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool
string encrypt Blowfish CBC MyMuleSoftKey "myPasswordLocal"
zhoritn2db6Ud+9QuhyViQ==

C:\Users\HP\Downloads>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool
string encrypt Blowfish CBC MyMuleSoftKey "myUsernameDev"
HbsuWJRjiuZZis+2oogkdQ==

C:\Users\HP\Downloads>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool
string encrypt Blowfish CBC MyMuleSoftKey "myPasswordDev"
zhoritn2db6oII22158LPQ==

C:\Users\HP\Downloads>
```

After you get the encrypted value, you need to add the following syntax in the properties files:

`![encryptedValue]`

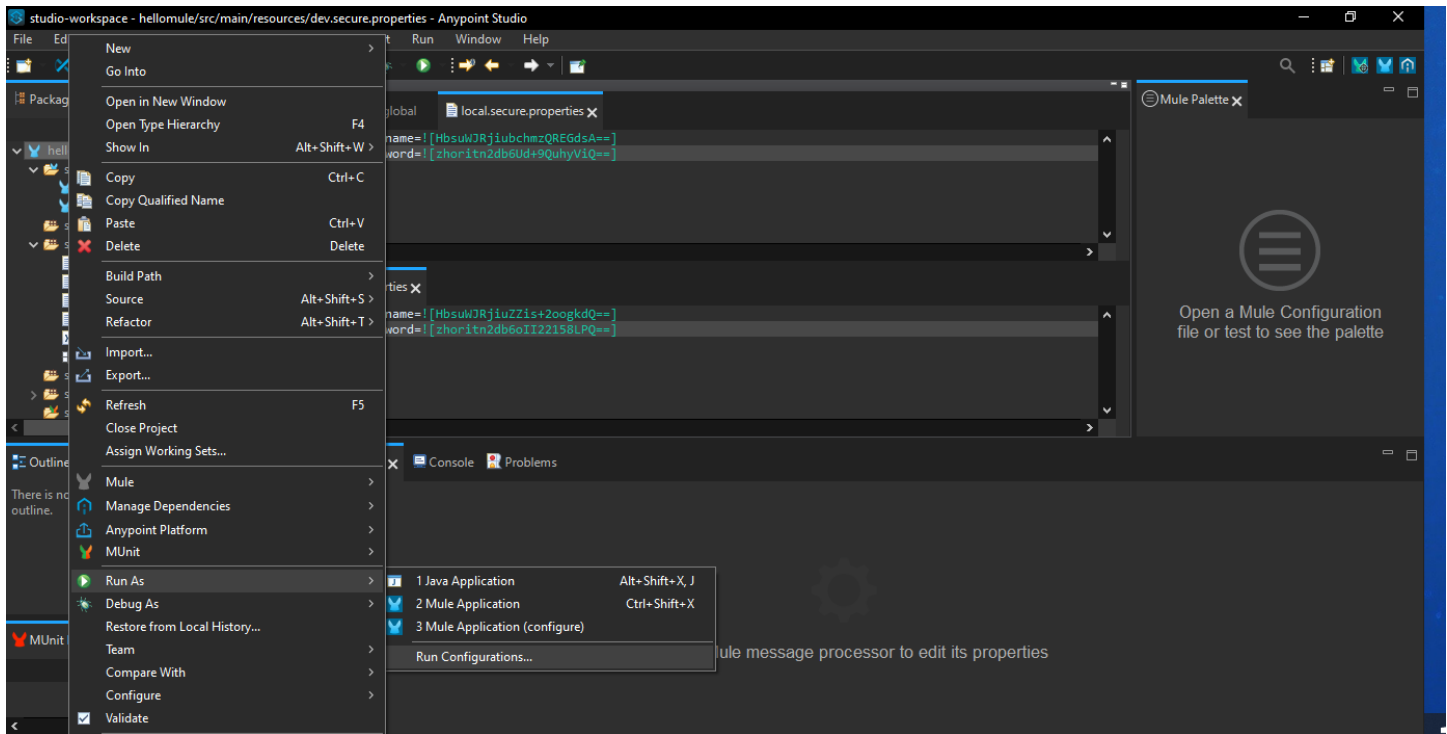
If your encrypted value was `HbsuWJRjiubchmzQREGdsA==`, then your property should be `![HbsuWJRjiubchmzQREGdsA==]`. Repeat for each property in `local.secure.properties` and `dev.secure.properties`. You should end up with something like this:

```
hellomule  global  *local.secure.properties X
1 example.username=![HbsuWJRjiubchmzQREGdsA==]
2 example.password=![zhoritn2db6Ud+9QuhyViQ==]
3

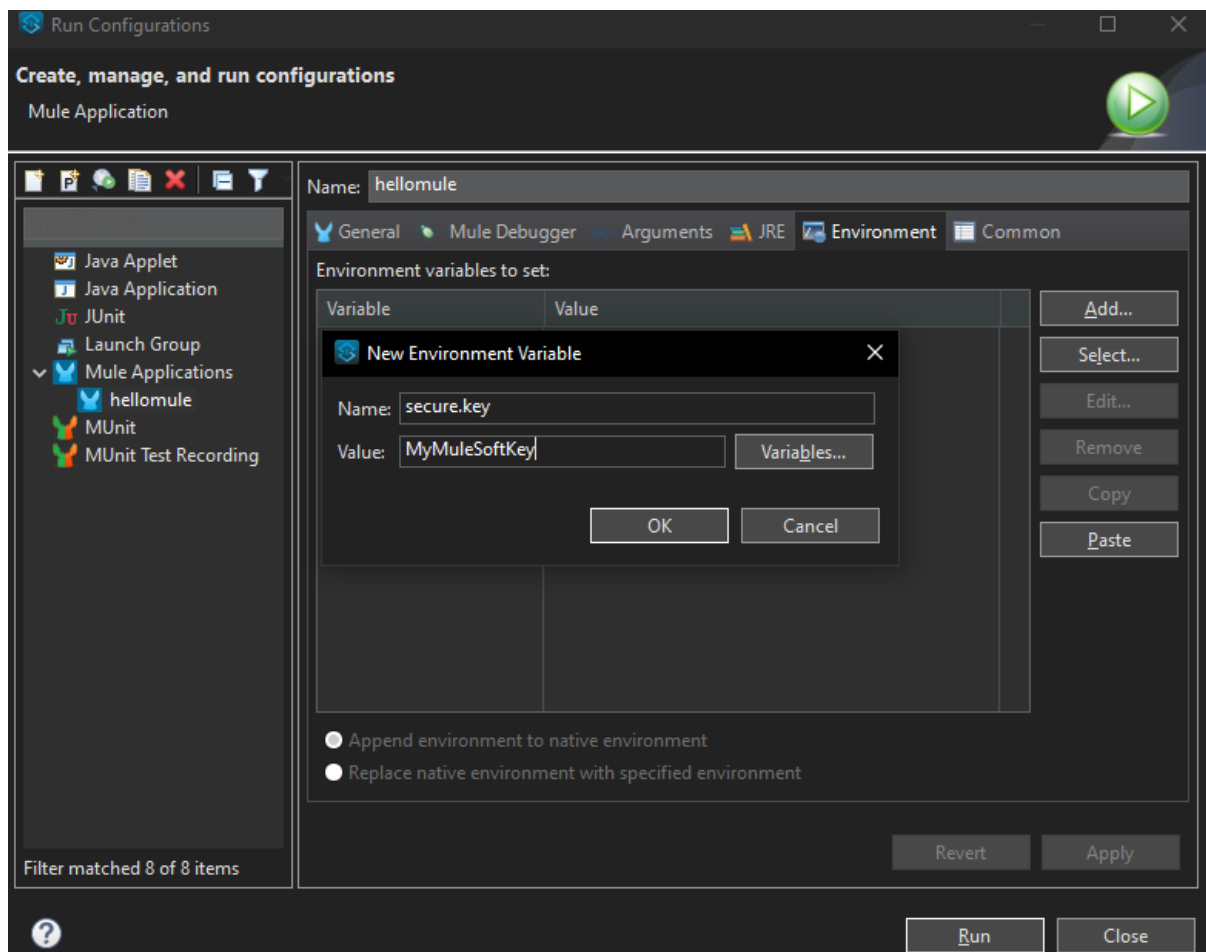
*dev.secure.properties X
1 example.username=![HbsuWJRjiuZZis+2oogkdQ==]
2 example.password=![zhoritn2db6oII22158LPQ==]
3
```

4. Test your application on your local computer

Right-click on your Mule project and select **Run As > Run Configurations**.



Go to the **Environment** tab and click on **Add**. Input the following and click **OK**:

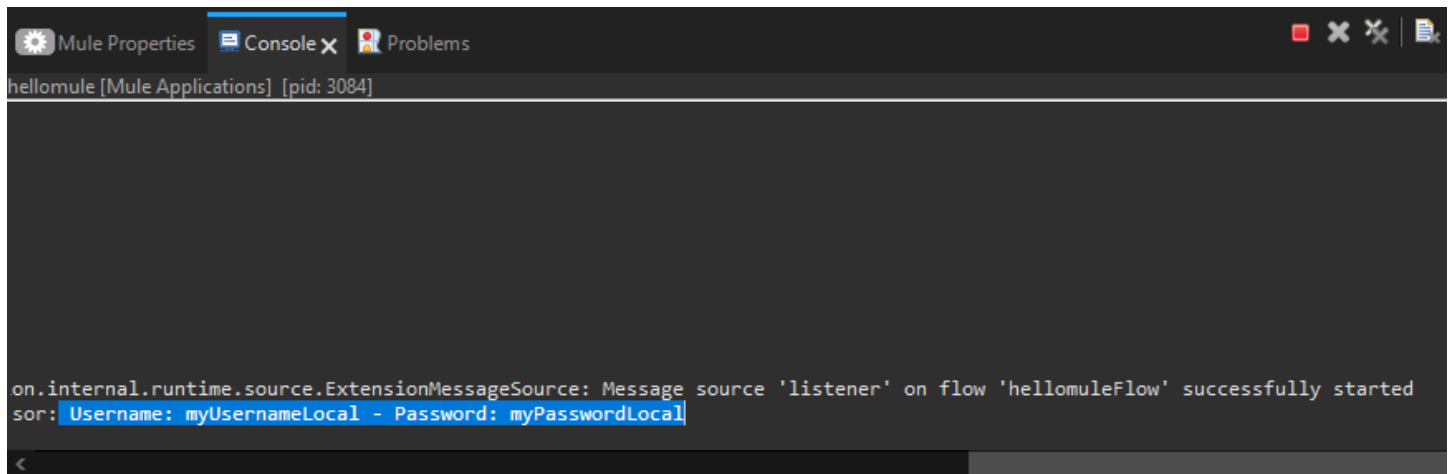


Once that's set, click on **Apply** and **Run**. This will send the **secure.key** property at runtime and it will not be saved in your code. We do this to prevent security breaches if someone has access to the source code of our application.

After you get the DEPLOYED status, go back to your REST client (in this case we'll be using Postman).

Send a request to **localhost:8081/hellomule**.

You should see the decrypted values in your Console.



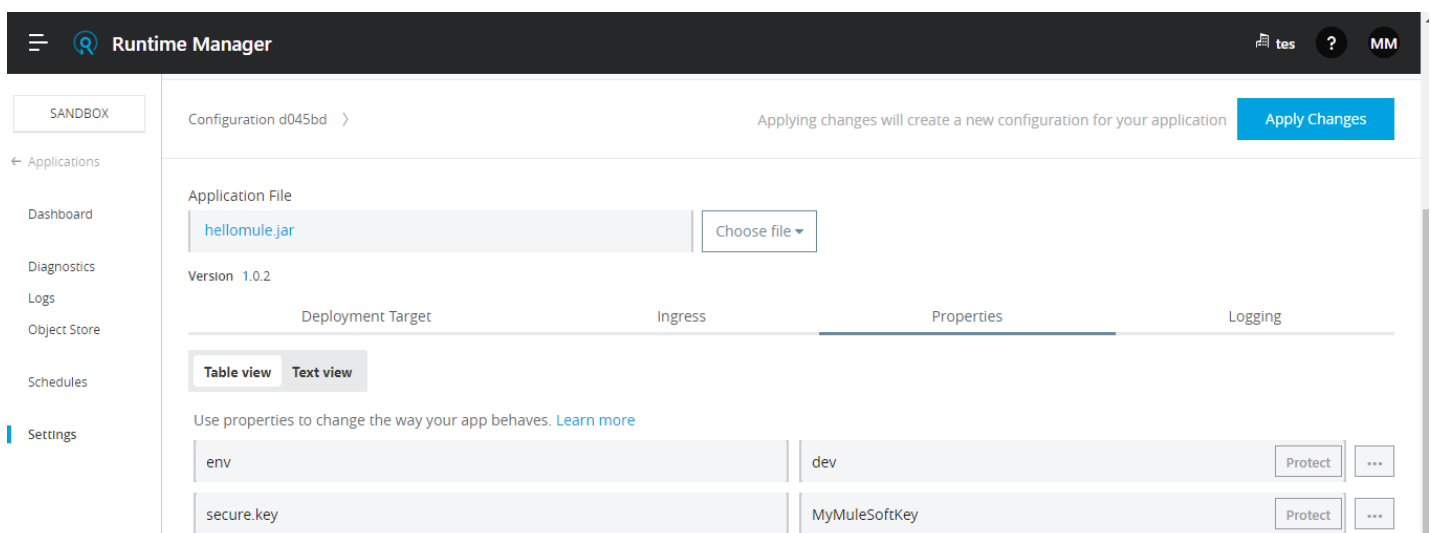
The screenshot shows the Mule IDE interface with the Console tab selected. The console output for the application 'hellomule' (pid: 3084) displays a log message: 'on.internal.runtime.source.ExtensionMessageSource: Message source 'listener' on flow 'hellomuleFlow' successfully started'. Below this message, the decrypted values are shown: 'Username: myUsernameLocal - Password: myPasswordLocal'.

Stop your application.

5. Deploy to CloudHub with your secure credentials

It's important that before you deploy your application to CloudHub, that you add your **secure.key** and your **env** values to your deployment properties. Sign in to [Anypoint Platform](#) and go to **Runtime Manager**. Select your **Sandbox** environment and click on your previously deployed application.

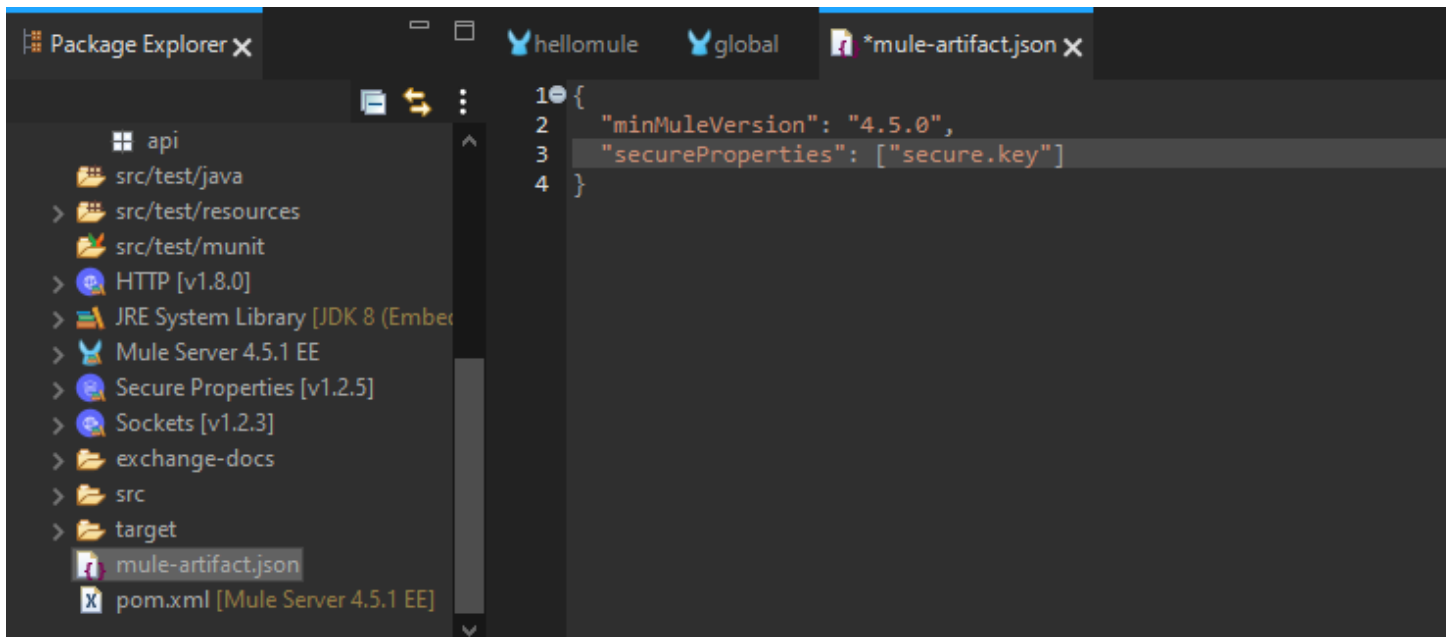
Now, go to the **Properties** tab and add the two properties. This time the **env** property should be set to **dev**. Click on Apply Changes.



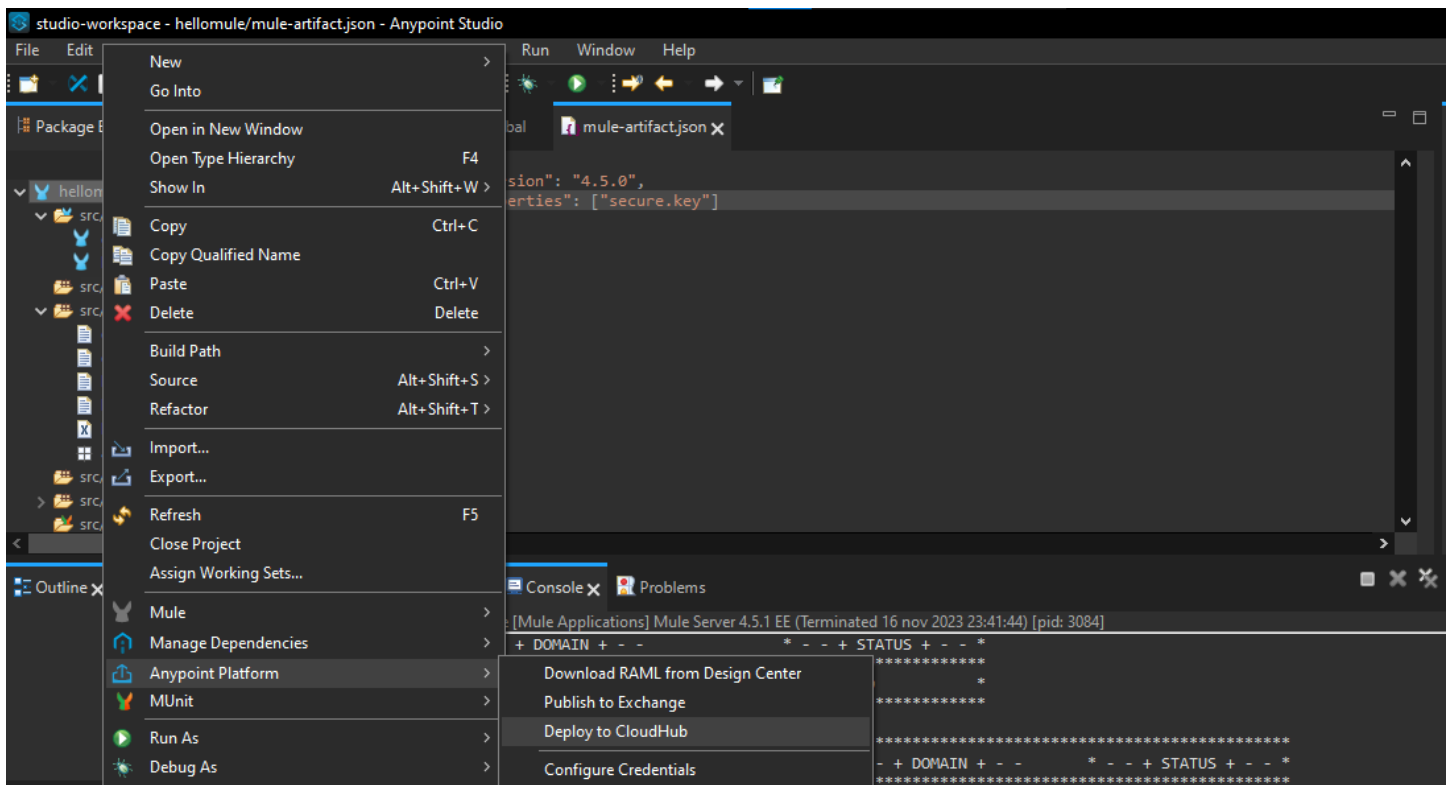
The screenshot shows the Anypoint Platform Runtime Manager interface. The left sidebar contains navigation links: Applications, Dashboard, Diagnostics, Logs, Object Store, Schedules, and Settings. The main area displays the configuration for 'Configuration d045bd'. The 'Application File' is 'hellomule.jar' and the 'Version' is '1.0.2'. The 'Properties' tab is selected, showing a table of properties. The table has columns for the property name, value, and actions (Protect and a menu icon).

Property Name	Value	Protect	Menu
env	dev	Protect	...
secure.key	MyMuleSoftKey	Protect	...

Notice how the `secure.key` value is not currently hidden. To hide it, go to your **mule-artifact.json** file in Anypoint Studio and add this line before the closing brackets (don't forget to add a comma at the end of the `minMuleVersion` line): `"secureProperties": ["secure.key"]`



Save all your files and deploy your Mule App to CloudHub by selecting **Anypoint Platform > Deploy to CloudHub**.



Make sure you select the previously deployed application as the **Deploying Application**. After it finishes deploying, refresh the page, and your `secure.key` property should now be hidden in Runtime Manager.

Apply Changes

The screenshot shows the Anypoint Platform interface. On the left is a 'SANDBOX' tab. The main area has a top bar with 'tes' and 'MM' icons. Below this is a blue 'Apply Changes' button. Underneath is an 'Application File' input field. A tabbed interface shows 'Deployment Target', 'Ingress', 'Properties' (selected), and 'Logging'. The 'Properties' tab has 'Table view' and 'Text view' buttons. Below these is a link: 'Use properties to change the way your app behaves. [Learn more](#)'. A table lists properties:

Property	Value	Action
env	dev	Protect ...
secure.key	MyMuleSoftKey	Protect ...

Verify:

The screenshot shows the Runtime Manager interface. A green circle icon is next to the text: 'Application hellomule-ex1 successfully deployed to Cloudhub-US-East-2'. Below this is the text: 'You may close this window at any time.' At the bottom right are two buttons: 'Open in Browser' and 'Close Window'.

Testing in POSTMAN:

The screenshot shows the Postman interface. The top bar has a 'GET' method and the URL 'https://hellomule-ex1-xcmxrf.5sc6y6-4.usa-e2.cloudhub.io/hellomule'. A blue 'Send' button is on the right. Below the URL bar are tabs: 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', 'Settings', and 'Cookies'. The 'Params' tab is selected. Below it is a 'Query Params' table:

Key	Value	Description

Below the table are tabs: 'Body', 'Cookies', 'Headers (4)', and 'Test Results'. The 'Body' tab is selected. At the bottom, the response is shown in 'Pretty' format: '1 Hello Mule'. The status bar at the bottom right shows: 'Status: 200 OK Time: 4.88 s Size: 166 B Save as example'.

You should see the decrypted values in the **Logs** tab after calling your CloudHub application. Remember you can call your CloudHub app by using the **App url** ending in **.cloudhub.io**.

Runtime Manager

tes ? MM

SANDBOX

← Applications

Dashboard

Diagnostics

Logs

Object Store

Schedules

Settings

hellomule-ex1

Config e7bb5a (Last Successful)

Search

Time range

Log Levels (5/5)

Last deployed 12 minutes ago - 2023-11-17 00:22 CST

Info

11 minutes ago - 2023-11-17 00:24:21.552 CST - CollectorAgentBase
\$wrapperListener_start_runner - Fast header injection enabled for app: hellomule-ex1

Replica 86kqt

Info

11 minutes ago - 2023-11-17 00:24:21.741 CST - ApplicationDeploymentListener
\$wrapperListener_start_runner - Anypoint monitoring custom file appender disabled.

Replica 86kqt

Warn

11 minutes ago - 2023-11-17 00:24:21.743 CST - ApplicationDeploymentListener
\$wrapperListener_start_runner - creating custom anypoint file appender

Replica 86kqt

Info

11 minutes ago - 2023-11-17 00:24:21.743 CST - ApplicationDeploymentListener
\$wrapperListener_start_runner - Disable console logging

Replica 86kqt

Info

11 minutes ago - 2023-11-17 00:24:21.851 CST - AbstractConnector
\$wrapperListener_start_runner - Started ServerConnector@1dc01d9f{HTTP/1.1, (http/1.1)}{0.0.0.0:7777}

Replica 86kqt

Info

11 minutes ago - 2023-11-17 00:24:22.148 CST - ExtensionMessageSource
\$[MuleRuntime].uber.02: [hellomule-ex1].uber@org.mule.runtime.module.extension.internal.runtime.source.ExtensionMessageSource.lambda\$reallyDoStart\$17:462 @41023bbe - Message source 'listener' on flow 'hellomuleFlow' successfully started

Replica 86kqt

Info

5 minutes ago - 2023-11-17 00:29:42.364 CST - LoggerMessageProcessor - 8efb3606-4431-8d43-c9b576f3324e
\$[MuleRuntime].uber.03: [hellomule-ex1].hellomuleFlow.CPU_LITE @2ef9bc8 - Username: myUsernameDev - Password: myPasswordDev

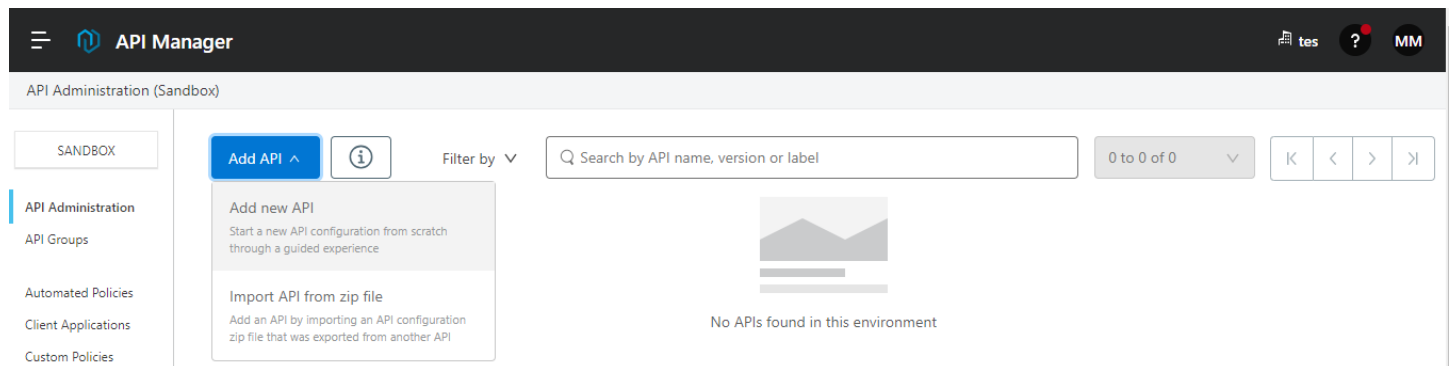
Replica 86kqt

Ejercicio 4: Cómo configurar la detección automática de API en Anypoint Studio

Crear una API en API Manager para permitir aplicar políticas.

1. Create a new API in API Manager

The first step to managing your API and enabling security policies is to create a new API in API Manager. To do this, [log in](#) or [sign up](#) for a free Anypoint Platform account and go to **API Manager**. Once you're there, select **Manage API > Create new API**.



Select Option Mule Gateway

[APIs](#) / Add API

Runtime API Downstream Upstream Review

Runtime

Choose the runtime where your API instance will run on.

* Required field

Select runtime

☐

Flex Gateway

NEW

Ultrafast API gateway designed to manage and secure APIs running anywhere.

☒

Mule Gateway

API gateway embedded in Mule runtime. Connect directly to an existing Mule app or deploy a new proxy app.

☐

Service Mesh

Manage Kubernetes-based non-Mule microservices with Anypoint Service Mesh.

Click on **Next Button**.

Add a name for your API and select **HTTP API**. Note that this name doesn't need to be unique like the Runtime Manager one. Click on **Continue**.

APIs / Add API

Runtime

API

Downstream

Upstream

Review

API

Select the API you want to manage.

Select API from Exchange

Create new API

Once the API is created it will be published in Exchange in stable state.

Name

hellomuleapi

Asset types ⓘ

HTTP API

Click on **Next Button**.

When you create your new API, you will be granted an **API Autodiscory API ID**. You will need to input your Autodiscovery API ID in Anypoint Studio to link your mule application with your API in API Manager. For now, just make a note of your API ID.

APIs / hellomuleapi / API Summary

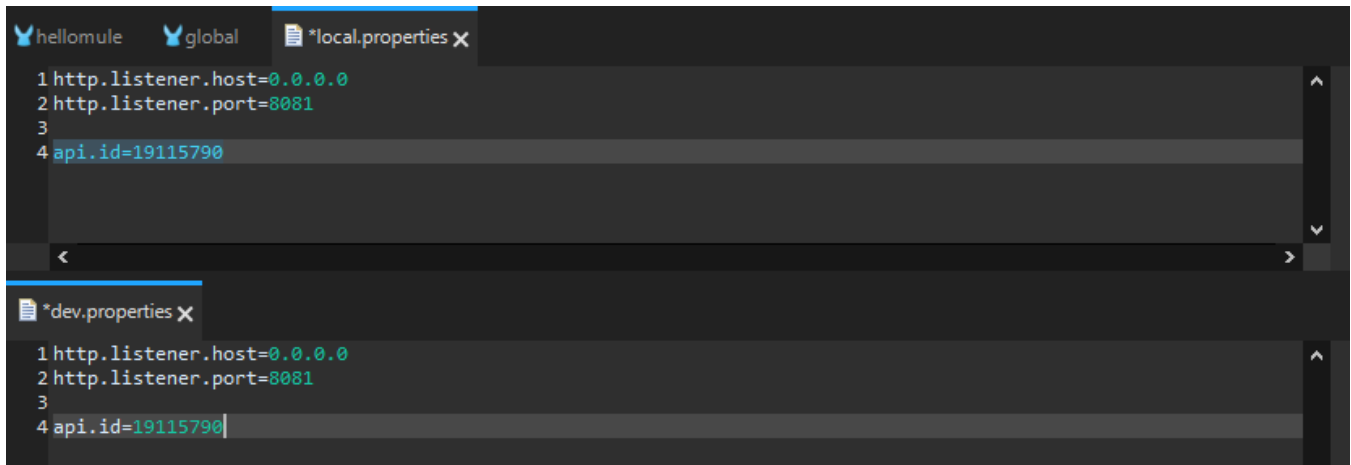
Actions ▾

To complete the registration process, you need to connect this API to your Mule application using Autodiscovery. [Learn more](#)

Type	Asset Version	Implementation URI ⓘ	
HTTP	1.0.0 (Latest)	N/A	
API Label ⓘ	API Version	API Status	
-	v1	Unregistered	
Consumer Endpoint	API Instance ID ⓘ		
N/A	19115790		
Tags			
ADD A TAG			

2. Create a new api.id property per environment

It's always a best practice to keep our properties in external files instead of hardcoding the values in our components. In your Anypoint Studio project, open the **local.properties** and **dev.properties** files under **src/main/resources**. Add a new **api.id** property in both files with the API ID value you got from API Manager.



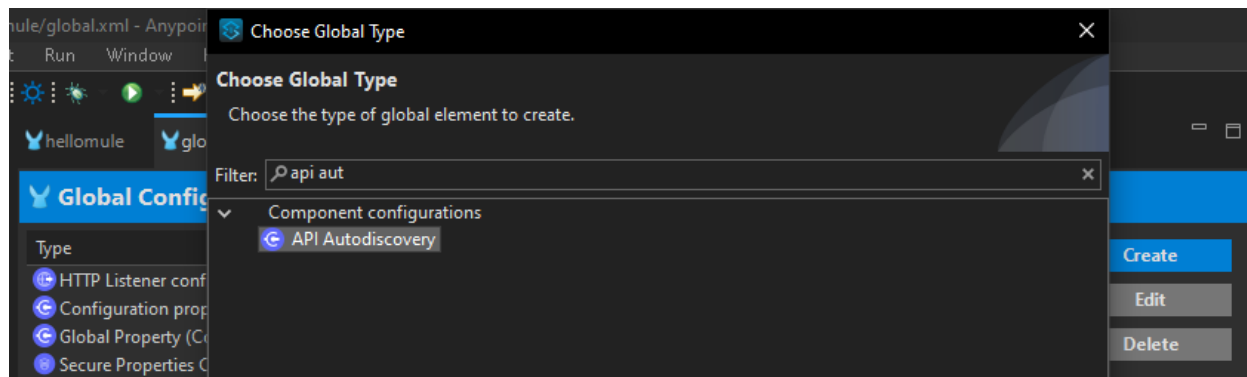
The screenshot shows two files open in Anypoint Studio: ***local.properties** and ***dev.properties**. Both files contain the following properties:

```
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
3
4 api.id=19115790
```

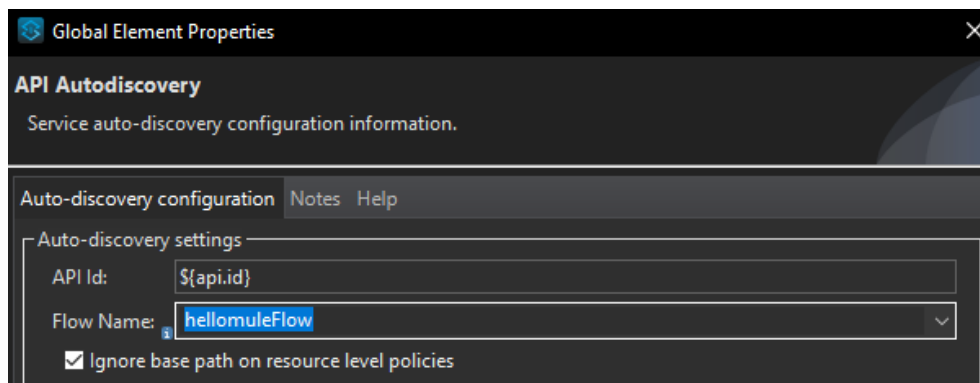
If you had more environments in API Manager like QA or PROD, your API ID would be different for each one and you would have to update your qa.properties or prod.properties files accordingly. In this case, we want to sync our API (from API Manager) to both our dev and local environments.

3. Configure Anypoint Studio with API Autodiscovery

Go to your **global.xml** file and switch to the **Global Elements** view. Click the **Create** button, then select **API Autodiscovery**.



To link the API Id field to your new property, just reference to it using the following syntax: `${api.id}`. For the flow name, you can just select the **hellomuleFlow**. Learn more by [visiting the documentation](#).



4. Test your application on your local computer

Solution to the fact that in the current version it is now shown directly to the environments: [To Create a New Environment](#)

Access Management

Users

Teams

Business Groups

Identity Providers

Client Providers

Audit Logs

Connected Apps

External Access

Composer Sync

Trusted Domains

Create business group

Business groups are isolated scopes for managing access. Users and teams may access resources in a business group through their permissions. [Learn more](#)

Filter business groups

Name	Environments	Total vCores	
tes	2	2	

Click the name of your root organization.

Access Management

Users

Teams

Business Groups

Identity Providers

Client Providers

Audit Logs

Connected Apps

External Access

Composer Sync

Trusted Domains

Business Groups / tes

Settings

Access Overview

Child Groups

Environments

Roles

Limits

Create environment

Environments are isolated scopes within a business group for deploying applications and APIs. Some permissions must be applied to a specific environment. [Learn more](#)

Filter environments

Name	Type	Default client provider	
Design	Design	Anypoint	
Sandbox	Sandbox	Anypoint	

Click the Environments tab.

Click Add Environment.

Configuration Environment

Add environment

Name

Sandbox

Type

Production

Sandbox

Design

Cancel

Create

Before we run the application, we need to add two more values to be able to connect to our API Manager. Navigate back to Anypoint Platform and go to **Access Management**. Go to your Environments, and select the Environment you are going to deploy to (in this case, Sandbox). You will need to copy and paste your **Client ID** and **Client Secret** and include those properties in your Deployment Properties and in Anypoint Studio.

Edit environment

Name

Sandbox-1

Client ID

b01e3d4a9cc146dea13413dee0cfb0e0

Client Secret

E56Da1301f8c45C3a01726B3158fA044

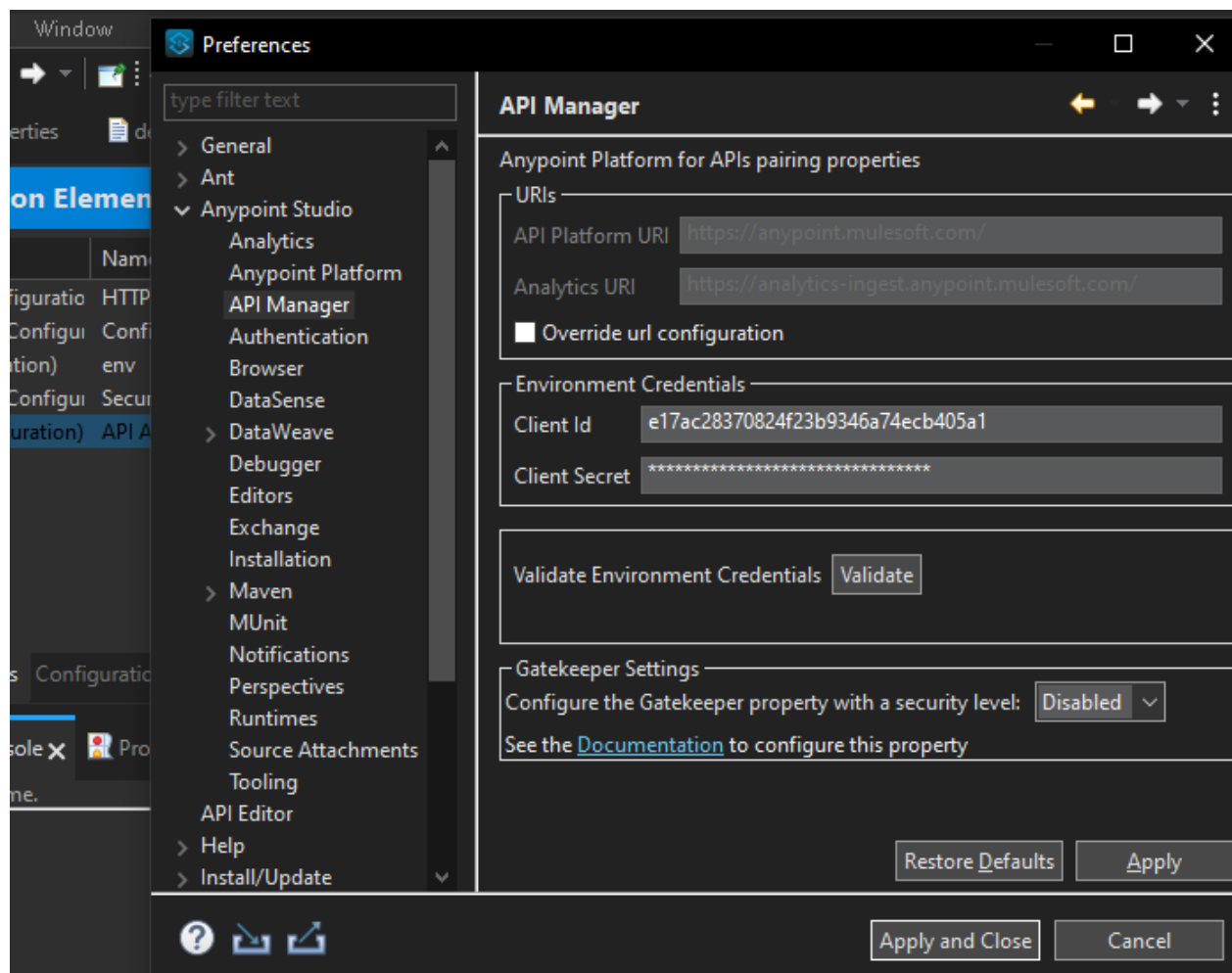
Hide

Delete Environment

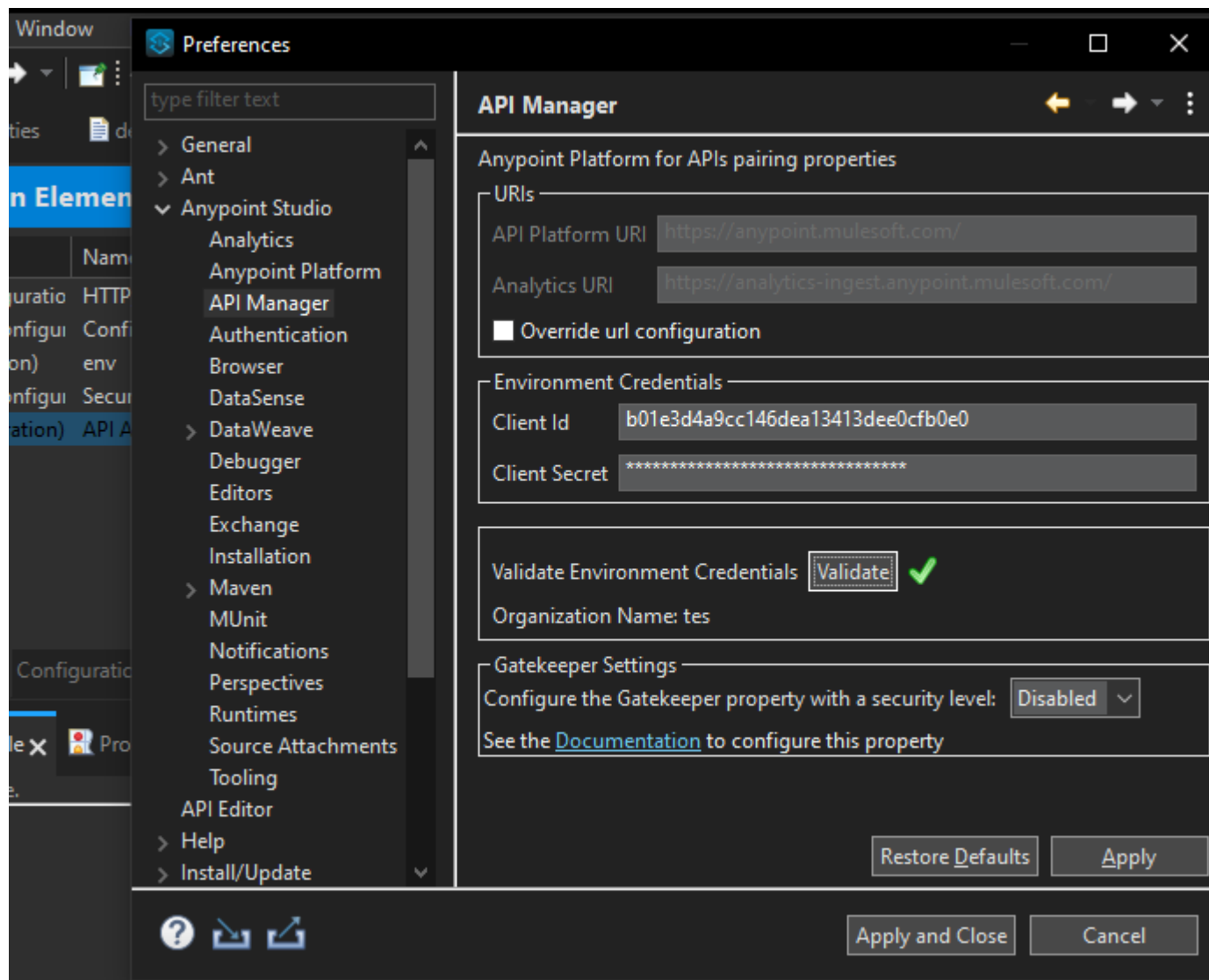
Cancel

Update

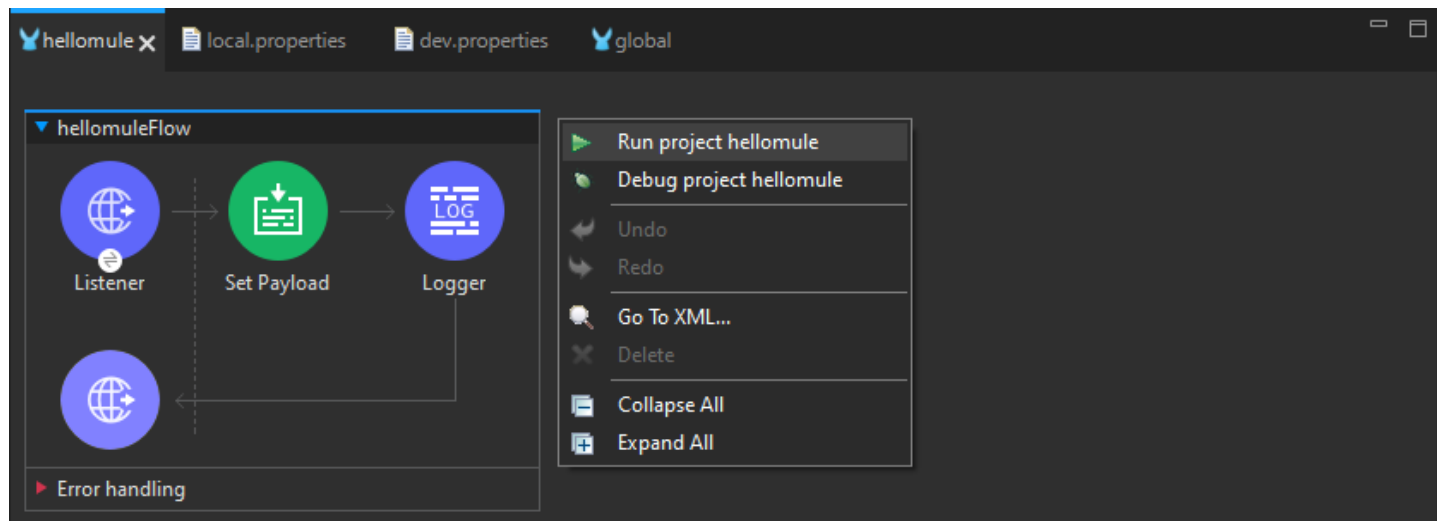
Go to your Anypoint Studio's (in this case **Window**) preferences or settings.



Navigate to **Anypoint Studio > API Manager**. Paste here the credentials you copied from Access Management (Client Id and Client Secret) and click **Validate**. The organization name should match what you have on the top-right corner of your Anypoint Platform account.



Once that's set, click on **Apply and Close** and run your Mule app. Verify that there are no errors in the Console and the Status of the application is **DEPLOYED**.



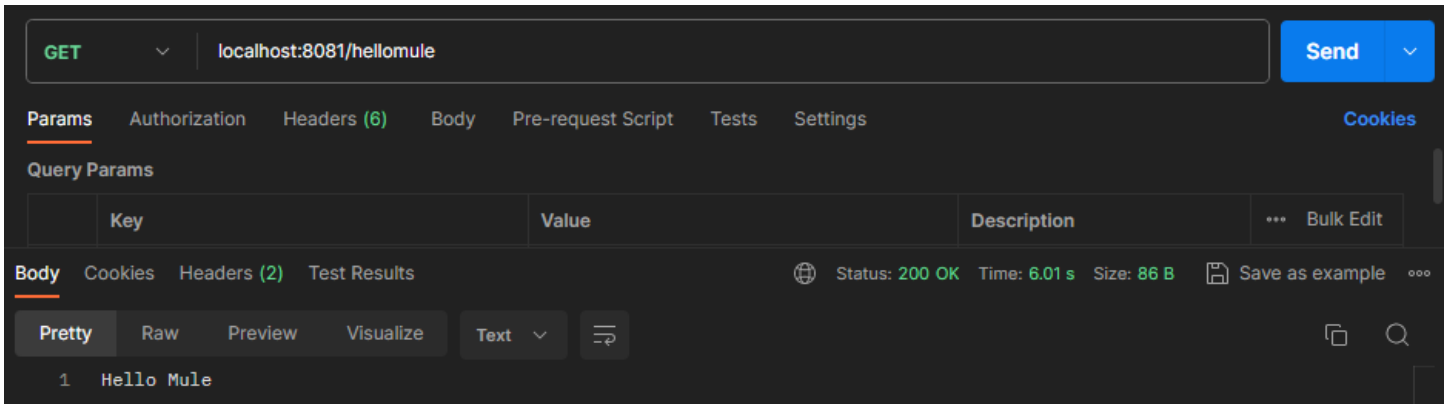
```

INFO 2023-11-17 11:19:37,453 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging.LogUtil
*****
*      - - + DOMAIN + - -      *      - - + STATUS + - - *
*****
* default      * DEPLOYED      *
*****

*****
*      - - + APPLICATION + - -      *      - - + DOMAIN + - -      *      - - + STATUS + - - *
*****
* hellomule      * default      * DEPLOYED      *
*****

```

Go to your REST Client and send a request to the local application. You should get back the **Hello Mule** response and a **200 OK** status.



Stop the application.

5. Deploy to CloudHub with your environment credentials

In your Anypoint Studio, open the **mule-artifact.json** file to add these two properties to the secure properties list. This will hide both properties in our Runtime Manager for security reasons.

```

hellomule local.properties dev.properties global mule-artifact.json x
1 {
2   "minMuleVersion": "4.5.0",
3   "secureProperties": [
4     "secure.key",
5     "anypoint.platform.client_id",
6     "anypoint.platform.client_secret"
7   ]
8 }

```

Save this file and right-click on your project. Select **Anypoint Platform > Deploy to CloudHub**.



Deploying Application

hellomule



Deployment Target



Shared Space

US East (Ohio)

CloudHub 2.0



Application File

Runtime

Ingress

Properties

Logging

Table view

Text view

Use properties to change the way your app behaves. [Learn more](#)

New Key

New Value

Deployments with less than 0.1 vCPU available may take up to 10 minutes to start

Cancel

Deploy Application

Make sure you select the previously deployed application and check the **Overwrite Existing Application** option. Go to the **Properties** tab. Enter your Client ID and Client Secret like this:



SANDBOX

hellomule



Deployment Target



Shared Space

US East (Ohio)

CloudHub 2.0



Application File

Runtime

Ingress

Properties

Logging

Table view

Text view

Use properties to change the way your app behaves. [Learn more](#)

anypoint.platform.client_id

b01e3d4a9cc146dea13413dee0cfb0e0

Protect



anypoint.platform.client_secret

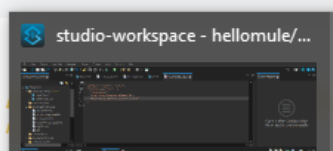
E56Da1301f8c45C3a01726B3158fA044

Protect



New Key

New Value



Cancel

Deploy Application

Click **Deploy Application**. After your application finishes deploying in Runtime Manager, you should see the status of your API set as Active in API Manager. Now you can enforce policies directly from API Manager without needing to redeploy your Mule application.

Ejercicio 5: Aplicar ID Client a una API en API Manager.

Ejercicio 6: Conoce sobre el enfoque de la primera API de Especificación.

Ejercicio 7: Construye una API de Especificación en API Designer.

Ejercicio 8: Habilitar el consumo de la aplicación Mule por medio de HTTPS.

Ejercicio 9: Desplegar la aplicación Mule en CloudHub utilizando Maven.

