



**TGH**

Making Integrations Simpler



# How to expose GraphQL API in MuleSoft

Author  
Vishnu R



## Expose GraphQL API in MuleSoft

In this blog, let us discuss how to expose GraphQL API in MuleSoft.

- We would be exposing a GraphQL API in MuleSoft which would allow us to get 2 resources that are exposed as REST APIs.
- Here, we can consume a single GraphQL API Endpoint to consume both resources rather than sending requests to the individual Exposed APIs.
- We can also limit the number of fields that would be returned using the request that would be sent to the GraphQL API itself and we don't need to make any changes in the individual REST APIs that are exposed.

Pre-Requisites:

- We need to have 2 REST APIs exposed which would be utilized by the GraphQL API.
- In my scenario, I am utilizing 2 REST APIs that are exposed. One of the endpoint is getProducts which is used to fetch the details regarding all the products present and the second endpoint is getProductsById which would return a particular product detail according to the id that is passed.

Now, let us see the steps to achieve this.

**Step 1:** First, we need to create a Schema. Schema will give the framework for the GraphQL API that we are going to expose.

- We need to create a schema file in our local directory which will be imported. For creating the same you can use a text file that needs to be saved as a .graphql file.

```
1  type Product{
2      id: ID!
3      title: String
4      description: String
5      price: String
6      discountPercentage: String
7      rating: String
8      stock: String
9      brand: String
10     category: String
11     thumbnail: String
12     images: [String]
13 }
14 type Query{
15     productById (id:ID): Product
16     products: [Product]
17 }
18 schema{
19     query: Query
20 }
```

**Step 2:** Now, we need to publish the schema to the AnyPoint Exchange. For that, we should navigate to the exchange and we need to select publish a new asset. There we need to fill in some details. Select the asset type as GraphQL API, select the schema file, and click on publish.



**Publish a new asset**

Name: product-graphql-schema-new

Asset types: GraphQL API

Method:  Upload a GraphQL API schema  
 Upload an Apollo GraphQL API schema

File upload: product.graphql

Main file: product.graphql

Exchange generates the `groupId`, `AssetId` and `Version` for you. If you need, you can change the `AssetId` and `Version`.

GroupId: 2204692c-50f9-4927-b68d-c560b7539be3

AssetId: product-graphql-schema-new

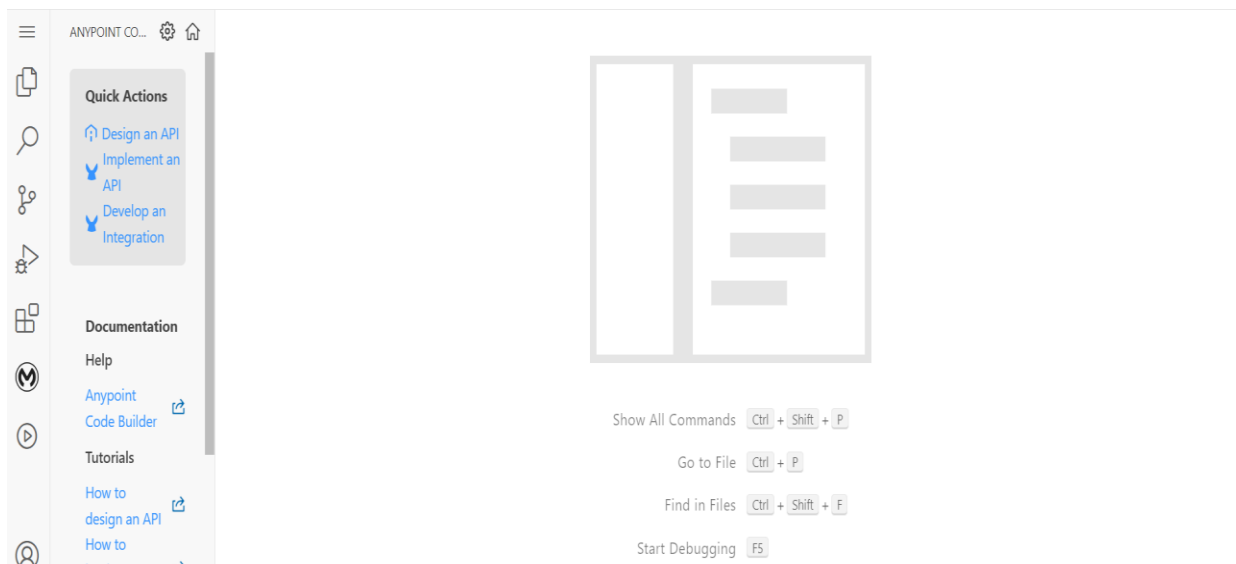
Version: 1.0.0

Version should use the `SemVer` format. Examples: 1.0.0 or 2.1.3

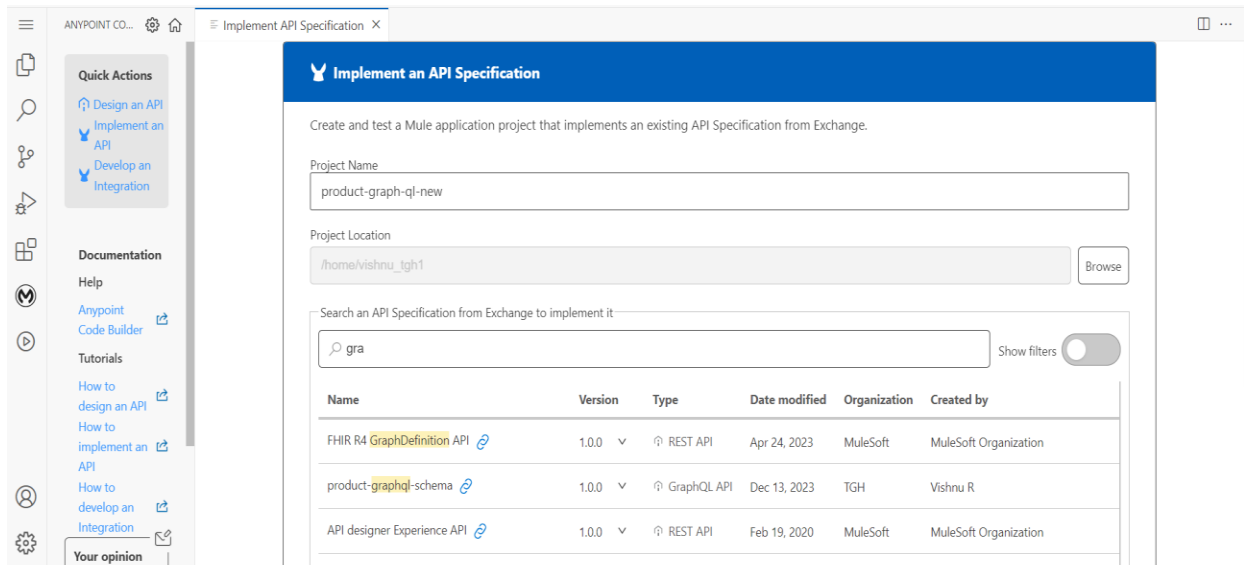
Lifecycle state:  Stable (Released and ready to consume)  
 Development (In process of design and development)

➤ Once we publish it, it will be visible in the exchange.

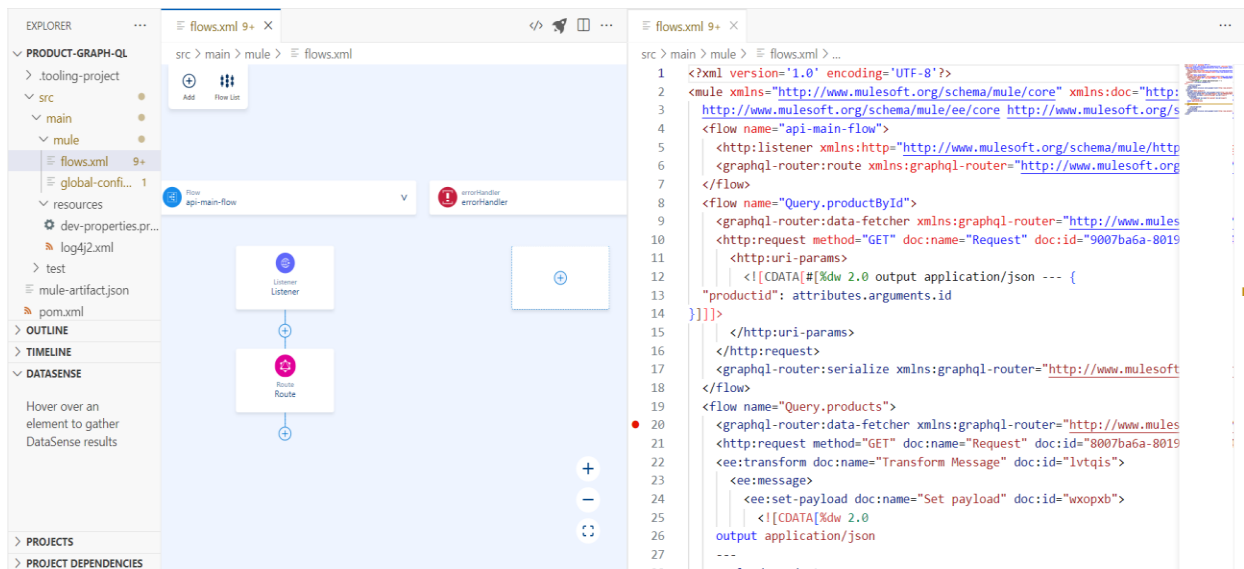
**Step 3:** Now we need to create an Implementation for the schema published in the exchange. I will be using AnyPoint Code Builder, a cloud-based IDE for building the implementation. In code builder, we need to select Implement an API from the quick actions.



**Step 4:** Once we click the implement an API option, we need to specify the name of the project, select the home folder, and select the schema that we have published in the exchange. Once the options are selected, we can click Create Project.



**Step 5:** Once you click create the project, the entire flow will get generated just like in AnyPoint Studio which will contain an explorer on the left-hand side, a graphical representation of the flow in the middle, and the XML related to the flow in the right-hand side.



**Step 6:** Here, in the graphical interface you can add new components and the details related to the component should be added in the XML. There would be 3 flows that would be created in my case the main flow, the get products flow, and the get products by id flow. You can switch between the flows by using the flow list option at the top. If you notice, the XML that is created will contain all the details related to all the 3 flows. We need to make some changes to the XML. Where we need to add the following details:

```
<?xml version='1.0' encoding='UTF-8'?>

<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
xmlns:http="http://www.mulesoft.org/schema/mule/http" xmlns:graphql-
router="http://www.mulesoft.org/schema/mule/graphql-router"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
xsi:schemaLocation="http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/graphql-router
http://www.mulesoft.org/schema/mule/graphql-router/current/mule-graphql-
router.xsd
http://www.mulesoft.org/schema/mule/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd">
  <flow name="api-main-flow">
    <http:listener xmlns:http="http://www.mulesoft.org/schema/mule/http"
config-ref="http-listener-config" path="{http.listener.path}" />
    <graphql-router:route xmlns:graphql-
router="http://www.mulesoft.org/schema/mule/graphql-router" config-
ref="GraphQL_Router_Config" />
  </flow>
  <flow name="Query.productById">
    <graphql-router:data-fetcher xmlns:graphql-
router="http://www.mulesoft.org/schema/mule/graphql-router" config-
ref="GraphQL_Router_Config" objectType="Query" fieldName="productById" />
    <http:request method="GET" doc:name="Request" doc:id="9007ba6a-8019-438a-
9415-892c8294a14d" config-ref="Product API Config"
path="/products/{productid}">
      <http:uri-params>
        <![CDATA[#[%dw 2.0 output application/json --- {
"productid": attributes.arguments.id
}]]]>
      </http:uri-params>
    </http:request>
    <graphql-router:serialize xmlns:graphql-
router="http://www.mulesoft.org/schema/mule/graphql-router" config-
ref="GraphQL_Router_Config" objectType="Query" fieldName="productById" />
  </flow>
```

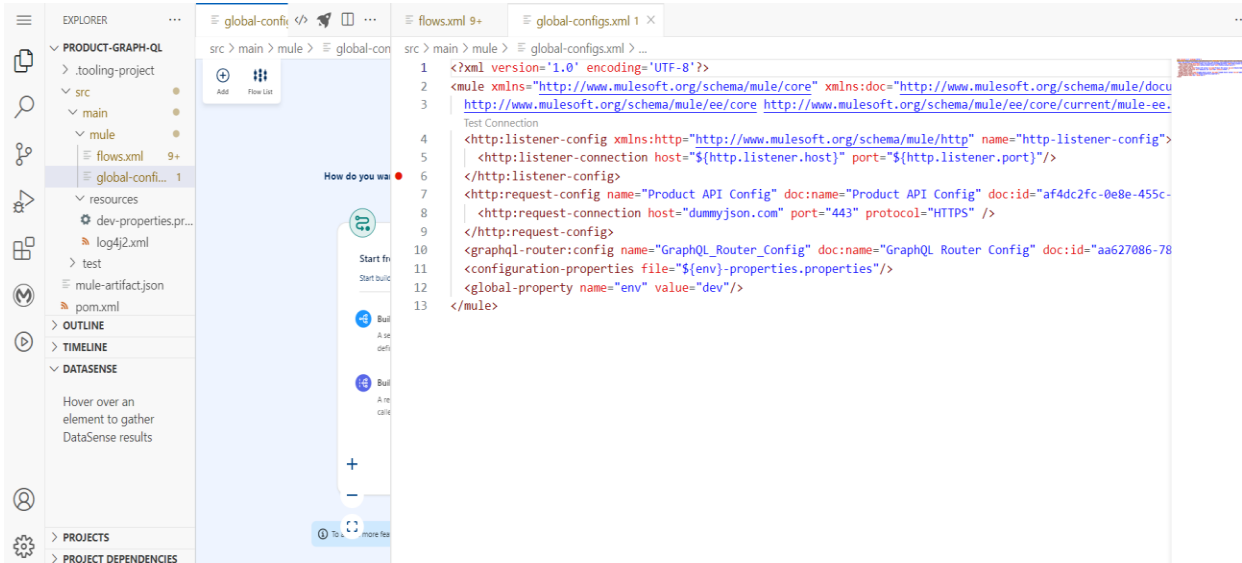
```

<flow name="Query.products">
  <graphql-router:data-fetcher xmlns:graphql-
router="http://www.mulesoft.org/schema/mule/graphql-router" config-
ref="GraphQL_Router_Config" objectType="Query" fieldName="products" />
  <http:request method="GET" doc:name="Request" doc:id="8007ba6a-8019-438a-
9415-892c8294a14d" config-ref="Product API Config"
path="/products"></http:request>
  <ee:transform doc:name="Transform Message" doc:id="lvtqis">
    <ee:message>
      <ee:set-payload doc:name="Set payload" doc:id="wxopxb">
        <![CDATA[%dw 2.0
output application/json
---
payload.products
]]>
      </ee:set-payload>
    </ee:message>
  </ee:transform>
  <graphql-router:serialize xmlns:graphql-
router="http://www.mulesoft.org/schema/mule/graphql-router" config-
ref="GraphQL_Router_Config" objectType="Query" fieldName="products" />
</flow>
</mule>

```

- Here, I have added the details for each flow that is needed where, In the main flow I would have a Listener and a GraphQL Router. In the get products flow I will have a Data fetcher at the start, a request component, and a transform message which will set the response payload as JSON and in the end a Serialize component. In the get products by ID flow, I will have a Data Fetcher at the start, we will have a request component, in which I will be setting a URI parameter and a Serialize component at the end.

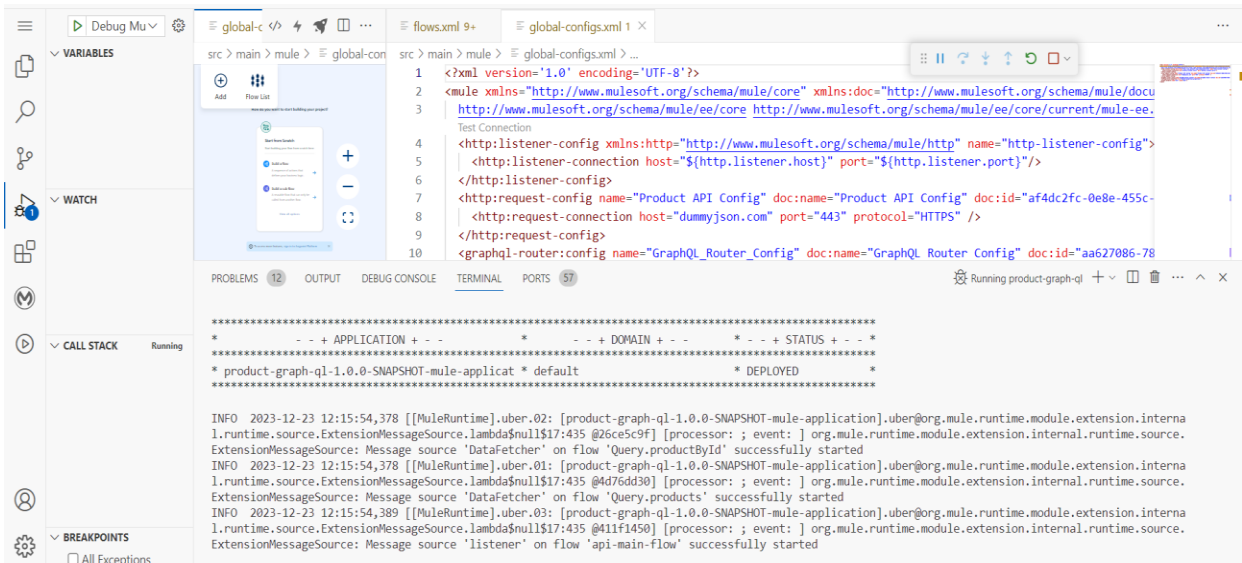
**Step 7:** Now, there is a global-config file that is used to define the configurations for the components as well as to define configuration property and global property values. We need to configure the Listener config, Request config as well as GraphQL Router config.



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <mule xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:doc="http://www.mulesoft.org/schema/mule/doc"
3   http://www.mulesoft.org/schema/mule/ee/core http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee-
4   Test Connection
5   <http:listener-config xmlns:http="http://www.mulesoft.org/schema/mule/http" name="http-listener-config">
6     <http:listener-connection host="${http.listener.host}" port="${http.listener.port}"/>
7   </http:listener-config>
8   <http:request-config name="Product API Config" doc:name="Product API Config" doc:id="af4dc2fc-0e8e-455c-
9     <http:request-connection host="dummyjson.com" port="443" protocol="HTTPS" />
10  </http:request-config>
11  <graphql-router:config name="GraphQL_Router_Config" doc:name="GraphQL Router Config" doc:id="aa627086-78
12  <configuration-properties file="${env}-properties.properties"/>
13  <global-property name="env" value="dev"/>
</mule>
  
```

**Step 8:** Once the configurations are done, we can deploy and test the API. To deploy it we can use the Run and Debug option and click the play icon and you would be able to see that the API is getting deployed in the console.

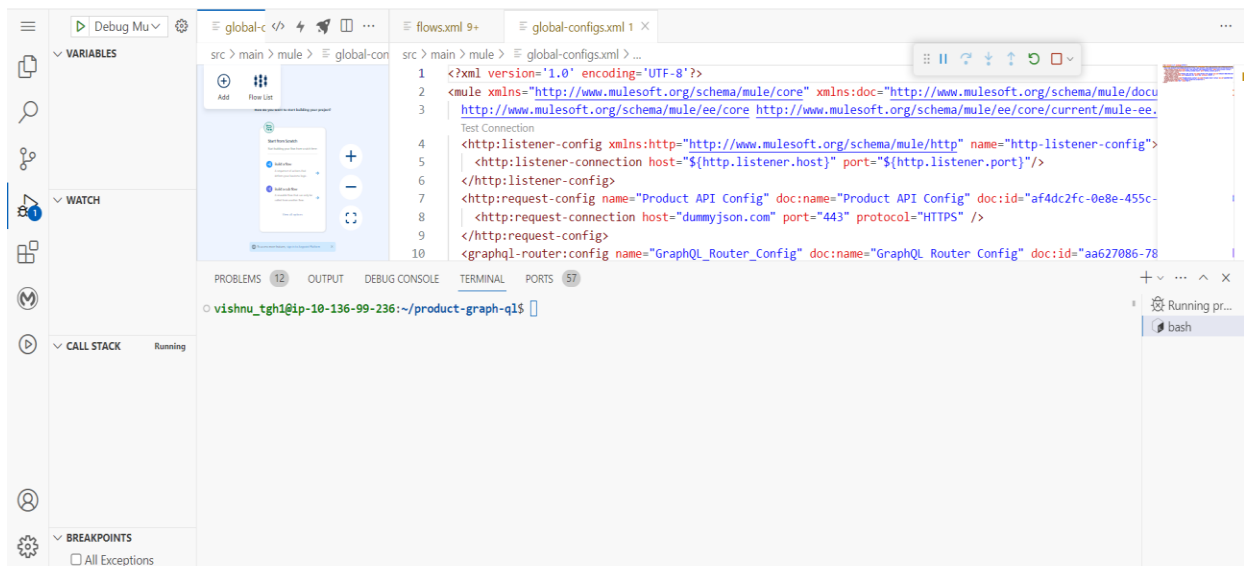


```

*****
* - - + APPLICATION + - - * - - + DOMAIN + - - * - - + STATUS + - - *
*****
* product-graph-ql-1.0.0-SNAPSHOT-mule-applicat * default * DEPLOYED *
*****

INFO 2023-12-23 12:15:54,378 [[MuleRuntime].uber.02: [product-graph-ql-1.0.0-SNAPSHOT-mule-application].uber@org.mule.runtime.module.extension.interna
1.runtime.source.ExtensionMessageSource.lambda$null$17:435 @26ce5c9f] [processor: ; event: ] org.mule.runtime.module.extension.internal.runtime.source.
ExtensionMessageSource: Message source 'DataFetcher' on flow 'Query.productsById' successfully started
INFO 2023-12-23 12:15:54,378 [[MuleRuntime].uber.01: [product-graph-ql-1.0.0-SNAPSHOT-mule-application].uber@org.mule.runtime.module.extension.interna
1.runtime.source.ExtensionMessageSource.lambda$null$17:435 @4d76dd30] [processor: ; event: ] org.mule.runtime.module.extension.internal.runtime.source.
ExtensionMessageSource: Message source 'DataFetcher' on flow 'Query.products' successfully started
INFO 2023-12-23 12:15:54,389 [[MuleRuntime].uber.03: [product-graph-ql-1.0.0-SNAPSHOT-mule-application].uber@org.mule.runtime.module.extension.interna
1.runtime.source.ExtensionMessageSource.lambda$null$17:435 @411f1450] [processor: ; event: ] org.mule.runtime.module.extension.internal.runtime.source.
ExtensionMessageSource: Message source 'listener' on flow 'api-main-flow' successfully started
  
```

**Step 9:** To test this API, we can make use of the terminal which is available in the code builder itself, where we need to pass the request in the format of curl. We can use the + symbol in the right-hand corner for creating a new terminal for testing.



**Step 10:** First let's start with testing get products,

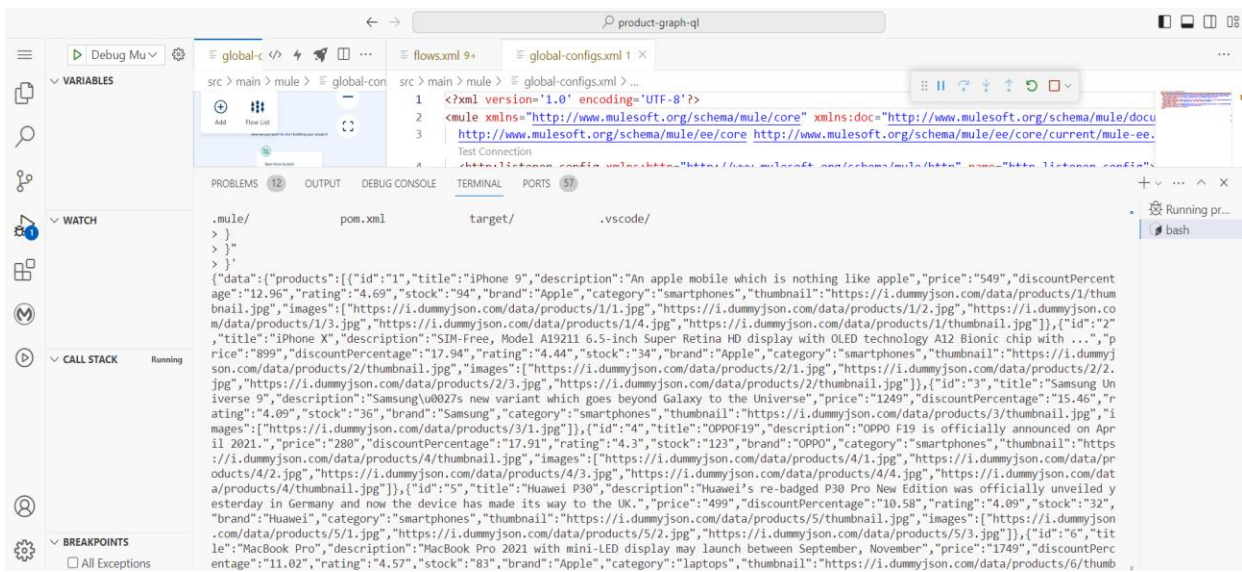
- Scenario-1: We need to get all the fields related to the products. The curl that should be sent would look like this:

Request:

```
curl --request POST \
--location 'http://localhost:8081/graphql' \
--header 'Content-Type: application/json' \
--data '{
  "query": "query products {
    products{
      id
      title
      description
      price
      discountPercentage
      rating
      stock
      brand
      category
      thumbnail
      images
    }
  }"
}'
```



## Response:



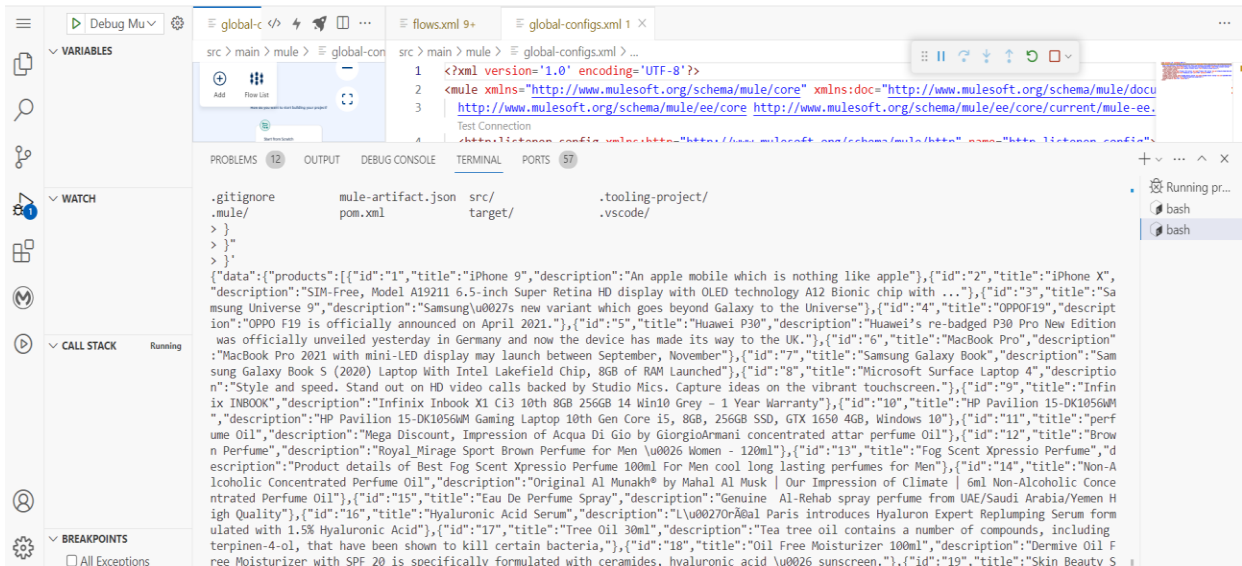
- Scenario-2: We need to get only the fields id, title, and description related to the products. The curl that should be sent would look like this:

## Request:

```

curl --request POST \
--location 'http://localhost:8081/graphql' \
--header 'Content-Type: application/json' \
--data '{
  "query": "query products {
    products {
      id
      title
      description
    }
  }"
}'
  
```

Response:



```

1 <?xml version='1.0' encoding='UTF-8'?>
2 <mule xmlns='http://www.mulesoft.org/schema/mule/core' xmlns:doc='http://www.mulesoft.org/schema/mule/docu
3 http://www.mulesoft.org/schema/mule/ee/core http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee-
4 Test Connection
  
```

```

.gitignore      mule-artifact.json  src/             .tooling-project/
.mule/          pom.xml           target/         .vscode/
>
>
>
>
{"data":{"products":[{"id":"1","title":"iPhone 9","description":"An apple mobile which is nothing like apple"},{"id":"2","title":"iPhone X",
"description":"SIM-Free, Model A19211 6.5-inch Super Retina HD display with OLED technology A12 Bionic chip with ..."}, {"id":"3","title":"Sa
msung Universe 9","description":"Samsung\u0027s new variant which goes beyond Galaxy to the Universe"}, {"id":"4","title":"OPPOF19","descript
ion":"OPPO F19 is officially announced on April 2021."}, {"id":"5","title":"Huawei P30","description":"Huawei's re-badged P30 Pro New Edition
was officially unveiled yesterday in Germany and now the device has made its way to the UK."}, {"id":"6","title":"MacBook Pro","description"
:"MacBook Pro 2021 with mini-LED display may launch between September, November"}, {"id":"7","title":"Samsung Galaxy Book","description":"Sam
sung Galaxy Book S (2020) Laptop With Intel Lakefield Chip, 8GB of RAM Launched"}, {"id":"8","title":"Microsoft Surface Laptop 4","descriptio
n":"Style and speed. Stand out on HD video calls backed by Studio Mics. Capture ideas on the vibrant touchscreen."}, {"id":"9","title":"Infin
ix INBOOK","description":"Infinix Inbook X1 Ci3 10th 8GB 256GB 14 Win10 Grey - 1 Year Warranty"}, {"id":"10","title":"HP Pavilion 15-DK1056WM
","description":"HP Pavilion 15-DK1056WM Gaming Laptop 10th Gen Core i5, 8GB, 256GB SSD, GTX 1650 4GB, Windows 10"}, {"id":"11","title":"perf
ume Oil","description":"Mega Discount, Impression of Acqua Di Gi\u00f2 by GiorgioArmani concentrated attar perfume Oil"}, {"id":"12","title":"Brow
n Perfume","description":"Royal Mirage Sport Brown Perfume for Men \u0026 Women - 120ml"}, {"id":"13","title":"Fog Scent Xpressio Perfume","d
escription":"Product details of Best Fog Scent Xpressio Perfume 100ml For Men cool long lasting perfumes for Men"}, {"id":"14","title":"Non-A
lcoholic Concentrated Perfume Oil","description":"Original Al Munakh\u2122 by Mahal Al Musk | Our Impression of Climate | 6ml Non-Alcoholic Conce
ntrated Perfume Oil"}, {"id":"15","title":"Eau De Perfume Spray","description":"Genuine Al-Rehab spray perfume from UAE/Saudi Arabia/Yemen H
igh Quality"}, {"id":"16","title":"Hyaluronic Acid Serum","description":"\u0027Or\u00c3al Paris introduces Hyaluron Expert Replumping Serum form
ulated with 1.5% Hyaluronic Acid"}, {"id":"17","title":"Free Oil 30ml","description":"Tea tree oil contains a number of compounds, including
terpinen-4-ol, that have been shown to kill certain bacteria,"}, {"id":"18","title":"Oil Free Moisturizer 100ml","description":"Dermive Oil F
ree Moisturizer with SPF 20 is specifically formulated with ceramides, hyaluronic acid \u0026 sunscreen."}, {"id":"19","title":"Skin Beauty S
  
```

**Step 11:** Now, let's test get products by id:

- Scenario-1: We need to get all the fields related to a particular id. The curl would look like:

Request:

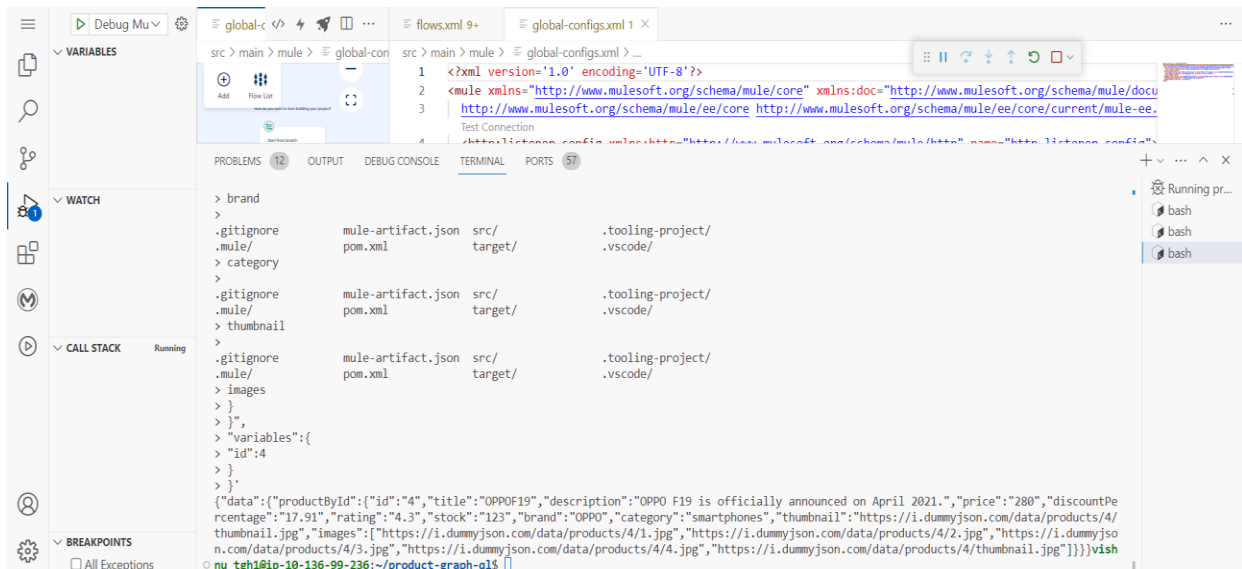
```

curl --request POST \
--location 'http://localhost:8081/graphql' \
--header 'Content-Type: application/json' \
--data '{
  "query": "query productById($id: ID){
    productById(id: $id) {
      id
      title
      description
      price
      discountPercentage
      rating
      stock
      brand
      category
      thumbnail
      images
    }
  }",

```

```
"variables": {
  "id":4
}
```

Response:

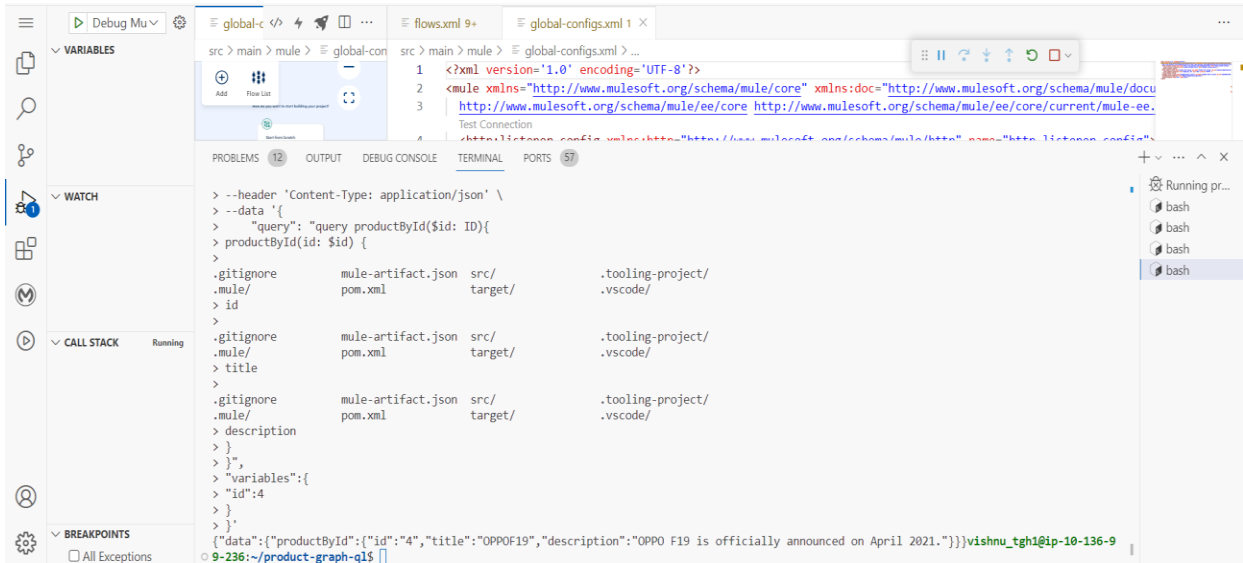


- Scenario-2: We need to get only the id, title, and description of the product based on the id. The curl that should be sent would look like this:

Request:

```
curl --request POST \
--location 'http://localhost:8081/graphql' \
--header 'Content-Type: application/json' \
--data '{
  "query": "query productById($id: ID){
    productById(id: $id) {
      id
      title
      description
    }
  }",
  "variables":{
    "id":4
  }
}'
```

## Response:



```

1 <?xml version='1.0' encoding='UTF-8'?>
2 <mule xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:doc="http://www.mulesoft.org/schema/mule/doc"
3 http://www.mulesoft.org/schema/mule/ee/core http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.
Test Connection
PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS 57
> --header 'Content-Type: application/json' \
> --data '{
>   "query": "query productById($id: ID){
>     productById(id: $id) {
>     }
>   }'
.gitignore      mule-artifact.json  src/      .tooling-project/
.mule/          pom.xml             target/   .vscode/
> id
>
.gitignore      mule-artifact.json  src/      .tooling-project/
.mule/          pom.xml             target/   .vscode/
> title
>
.gitignore      mule-artifact.json  src/      .tooling-project/
.mule/          pom.xml             target/   .vscode/
> description
> }
> },
> "variables":{
>   "id":4
> }
> }'
{"data":{"productById":{"id":4,"title":"OPPO F19","description":"OPPO F19 is officially announced on April 2021."}}vishnu_tgh1@ip-10-136-9
9-236:~/product-graph-ql$

```



# TGH

Making Integrations Simpler

## TGH Software Solutions Pvt. Ltd.

[www.techygeekhub.com](http://www.techygeekhub.com)

At TGH, we specialize in driving digital transformation through seamless Integration Technologies.

Operating as an INTEGRATION FACTORY, we serve as a one-stop shop for all your integration needs. Our expert team is well-versed in enterprise software and legacy system integration, along with leading iPaaS technologies like Boomi, MuleSoft, Workato, OIC, and more.

We're committed to enhancing business processes and solving problems through our integration expertise.



### Email address

[connect@techygeekhub.com](mailto:connect@techygeekhub.com)



### Phone number

+ 011-40071137  
+ 91-8810610395



### Our offices

#### Noida Office

iThum  
Plot No -40, Tower A,  
Office No: 712,  
Sector-62, Noida,  
Uttar Pradesh, 201301

#### Hyderabad Office

Plot no: 6/3, 5th Floor,  
Techno Pearl Building,  
HUDA Techno Enclave,  
HITEC City, Hyderabad,  
Telangana 500081

