Hands-on Lab - Querying with GraphQL using Python

Skills Network

Estimated Time: 60 minutes

In this lab, you will become familiar with using GraphQL to query the APIs served through a GraphQL service running on a server using Postman.

Learning Objectives:

After completing this exercise, you should be able to perform the following tasks:

- Use GraphQL with Postman to access GraphQL Service
- Structure queries to access APIs to get selective information that you need

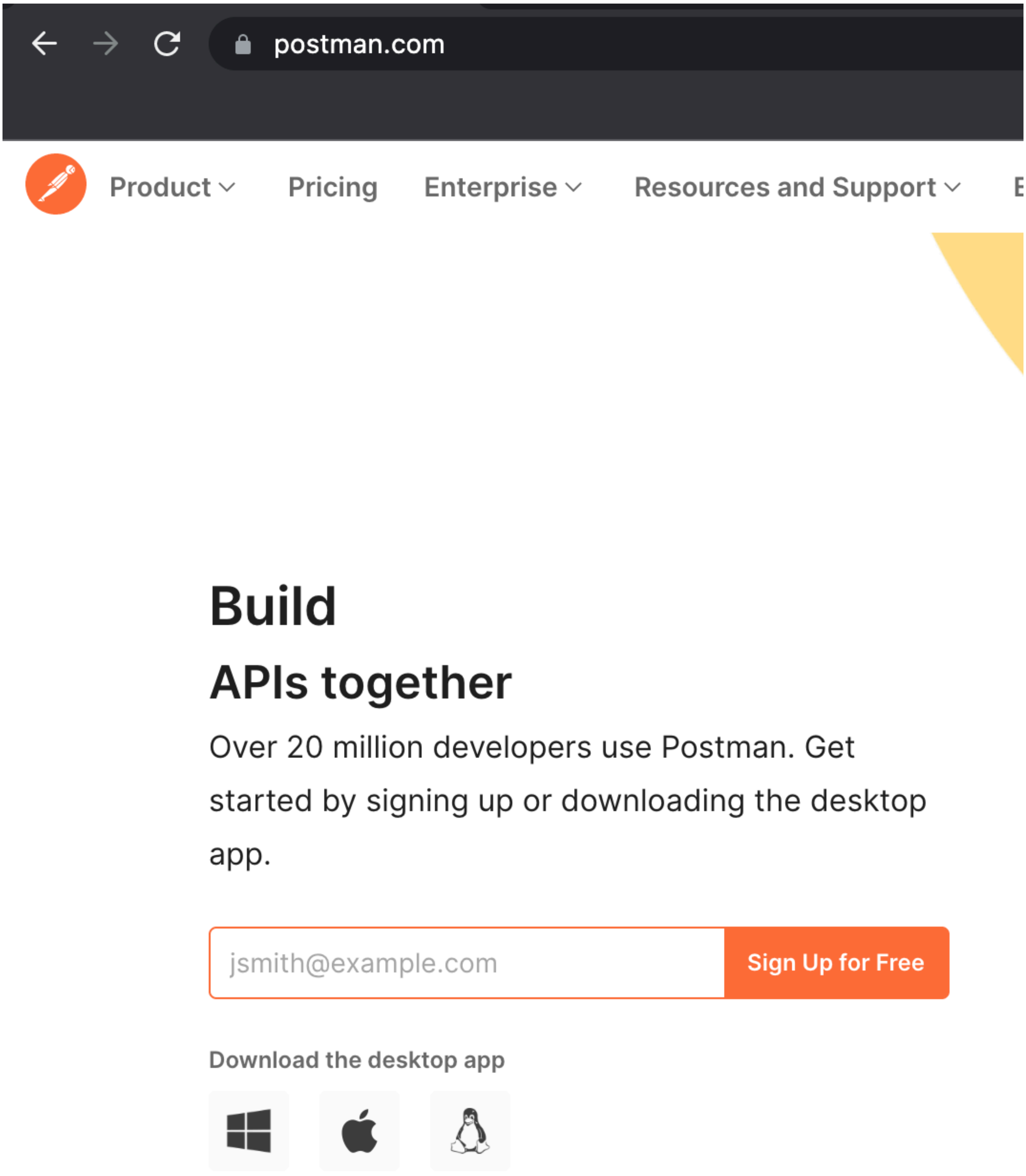
Pre-requisites

- You must be familiar with Docker applications and commands.
- You must have a good understanding of API.
- You must be familiar with using Postman.

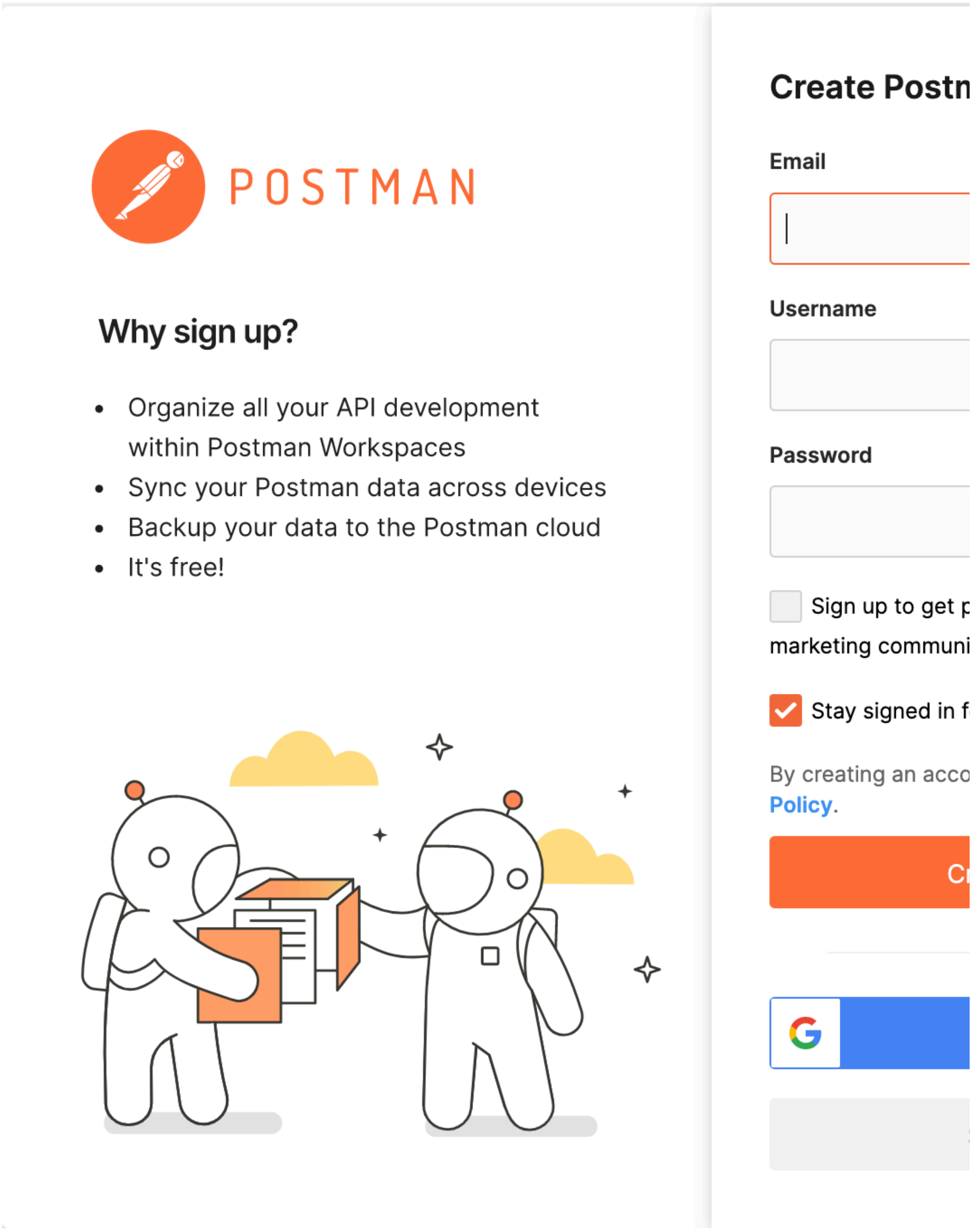
Task 1 - Sign-up for Postman

> If you already have a Postman account, you can skip this task.

1. On your browser, go to www.postman.com



2. Click **Sign Up for Free**.
3. Enter your email and choose a password, confirm it, and create a free account. You can alternatively use your Google account to sign in too. You may notice that **Stay signed in for 30 days** is selected by default. It is highly recommended that you uncheck that option if you are not using a personal computer for this lab.



You have successfully created a Postman account.

Task 2 - Run a GraphQL service

1. Open a terminal window by using the top menu in the IDE: **Terminal > New Terminal.**
2. In the terminal, clone the repository which has the GraphQL service code ready by pasting the following command. The repository that you clone has code that will run a GraphQL service where we can query details on US cities and states.

```
git clone https://github.com/ibm-developer-skills-network/jmgdo-microservices.git
```

3. Change the working directory to **jmgdo-microservices/graphql_example** by running the following command.

   ```
   cd jmgdo-microservices/graphql_example
   ```

4. Build the application by running the following command. This will build the docker application using the docker file in the current directory and tag it **graphqlservice**.

   ```
   docker build . -t graphqlservice
   ```

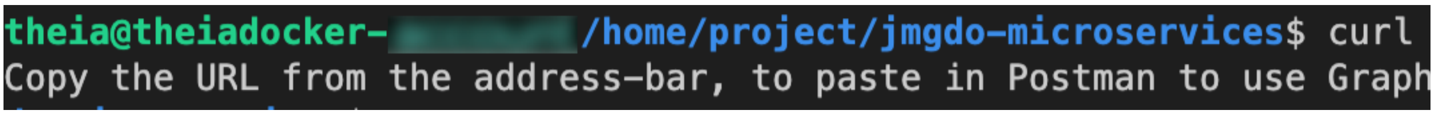
5. Run the application that you just built on port 8080, by running the following command on the terminal.

   ```
   docker run -dp 8080:4000 graphqlservice
   ```

   You will get a hex code that indicates the application has started.

6. To confirm that the service is running, execute the following command.

   ```
   curl localhost:8080
   ```

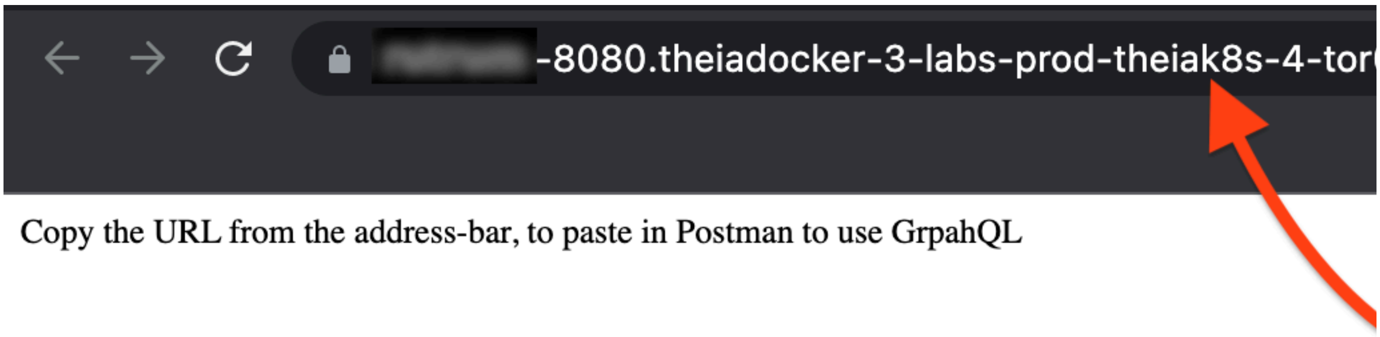


```
theia@theiadocker-[redacted]/home/project/jmgdo-microservices$ curl
Copy the URL from the address-bar, to paste in Postman to use Graph
```

If the output is as seen in the image above, your service is up and running.

7. Click on the Skills Network button on the left, it will open the "Skills Network Toolbox". Then click Launch Application, from there you enter the port no. as 8080 and click Your Application button. This will open a browser window.

8. This will launch a browser page. Copy the URL that you see in the address bar.



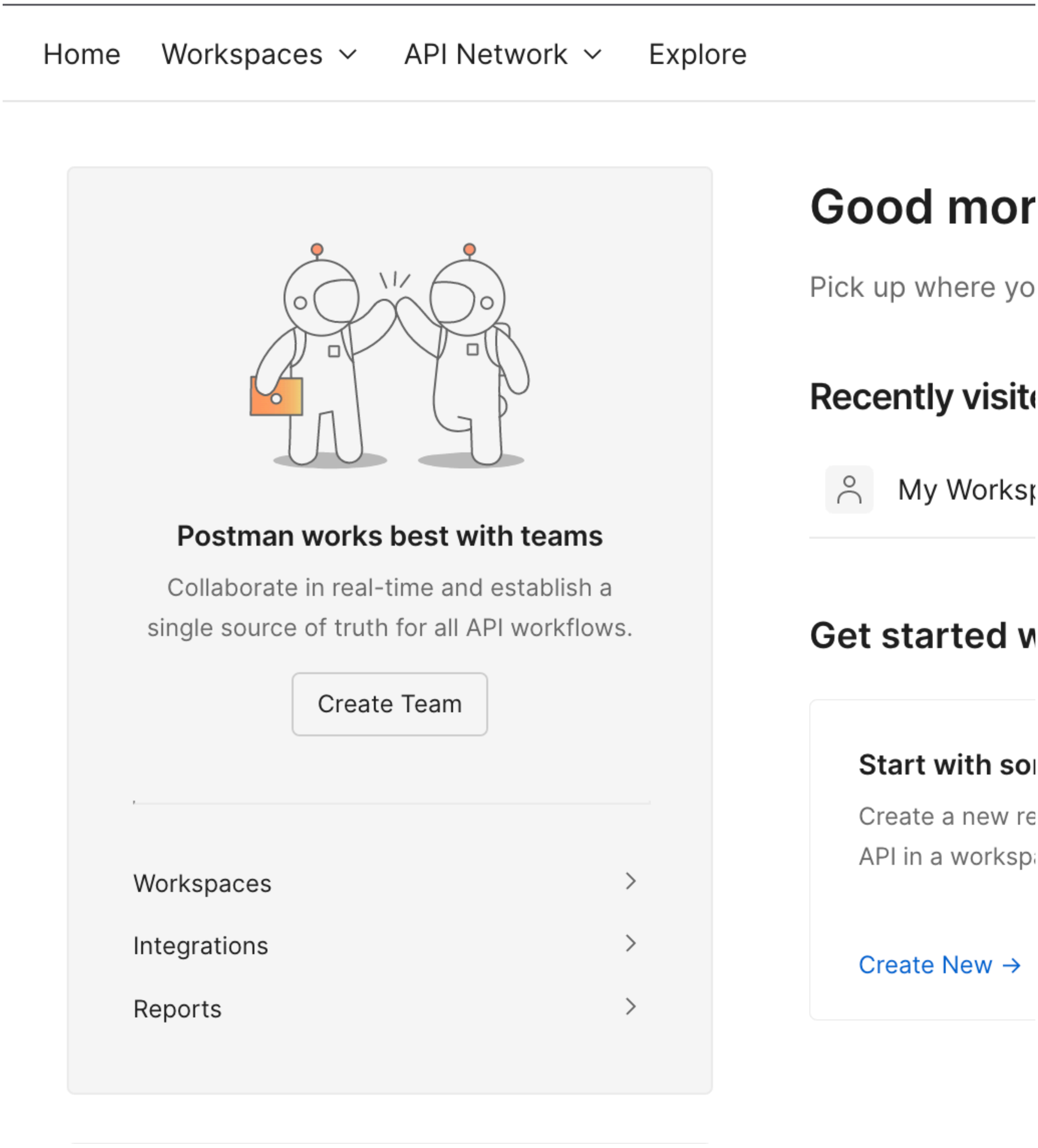
[redacted]-8080.theiadocker-3-labs-prod-theiak8s-4-tor

Copy the URL from the address-bar, to paste in Postman to use GrpahQL

Task 3 - Run GraphQL queries on Postman

1. Go to www.postman.com and sign in to Postman. Please note that the home page may look different for every person depending on the day of the year and the region you are signing in from.

2. On the homepage, click **Create new**.


Home
Workspaces
API Network
Explore

Postman works best with teams

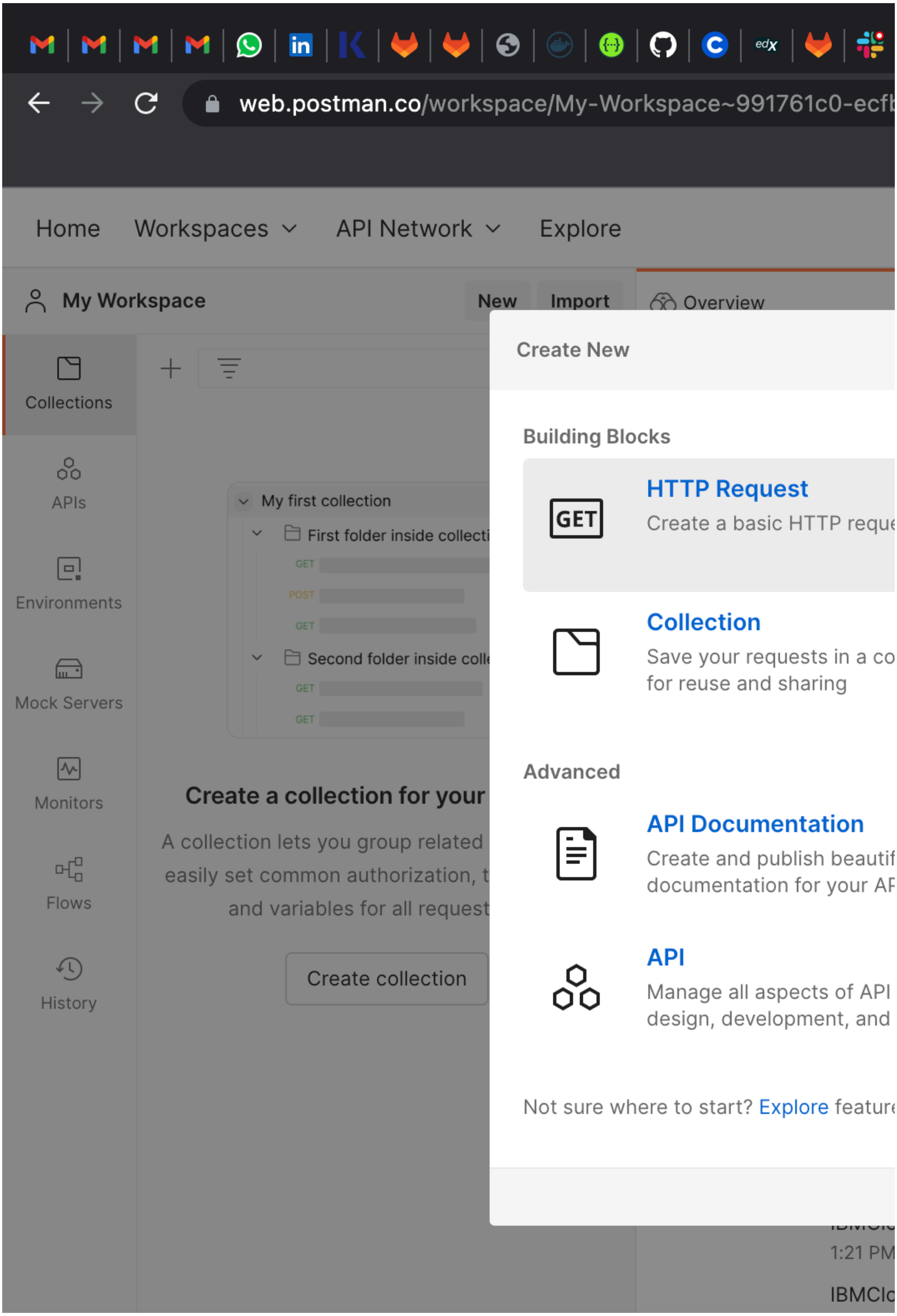
Collaborate in real-time and establish a single source of truth for all API workflows.

Create Team

Workspaces
Integrations
Reports

Good mor
Pick up where yo
Recently visit
My Worksp
Get started w
Start with so
Create a new re
API in a worksp
Create New →

3. This brings up the set of options you can use with Postman. Choose **HttpRequest**.


web.postman.co/workspace/My-Workspace~991761c0-ecf

Create a collection for your

A collection lets you group related
easily set common authorization, t
and variables for all request

Create New

Building Blocks

HTTP Request
Create a basic HTTP reque

Collection
Save your requests in a co
for reuse and sharing

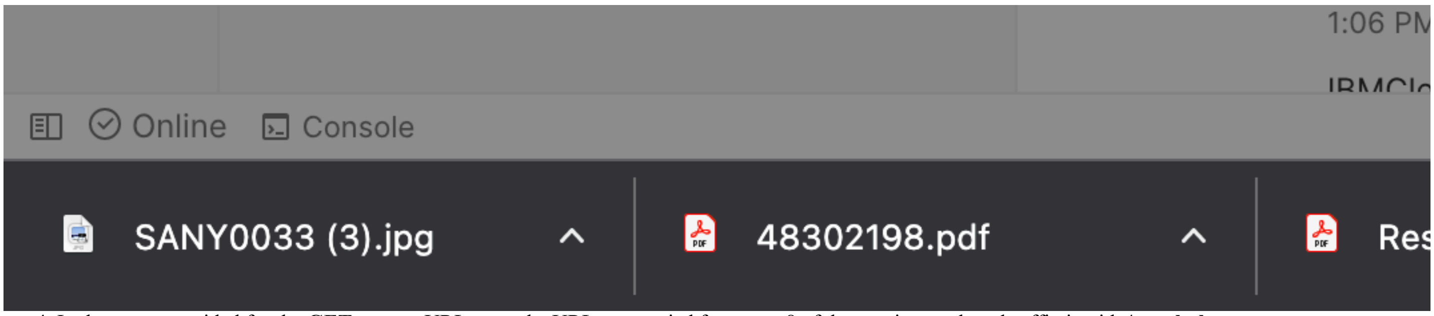
Advanced

API Documentation
Create and publish beautif
documentation for your AF

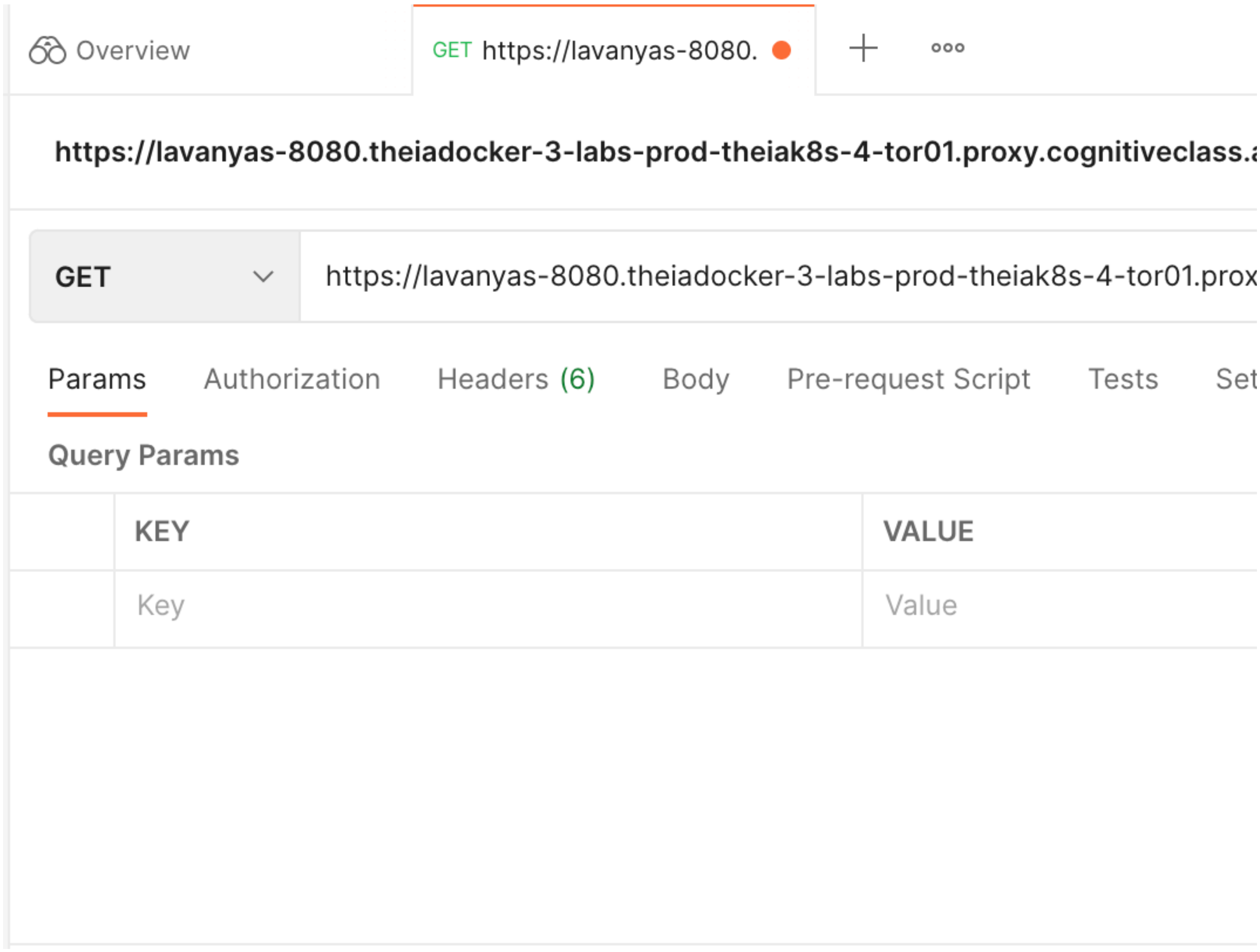
API
Manage all aspects of API
design, development, and

Not sure where to start? Explore featur

1:21 PM
IBMClo



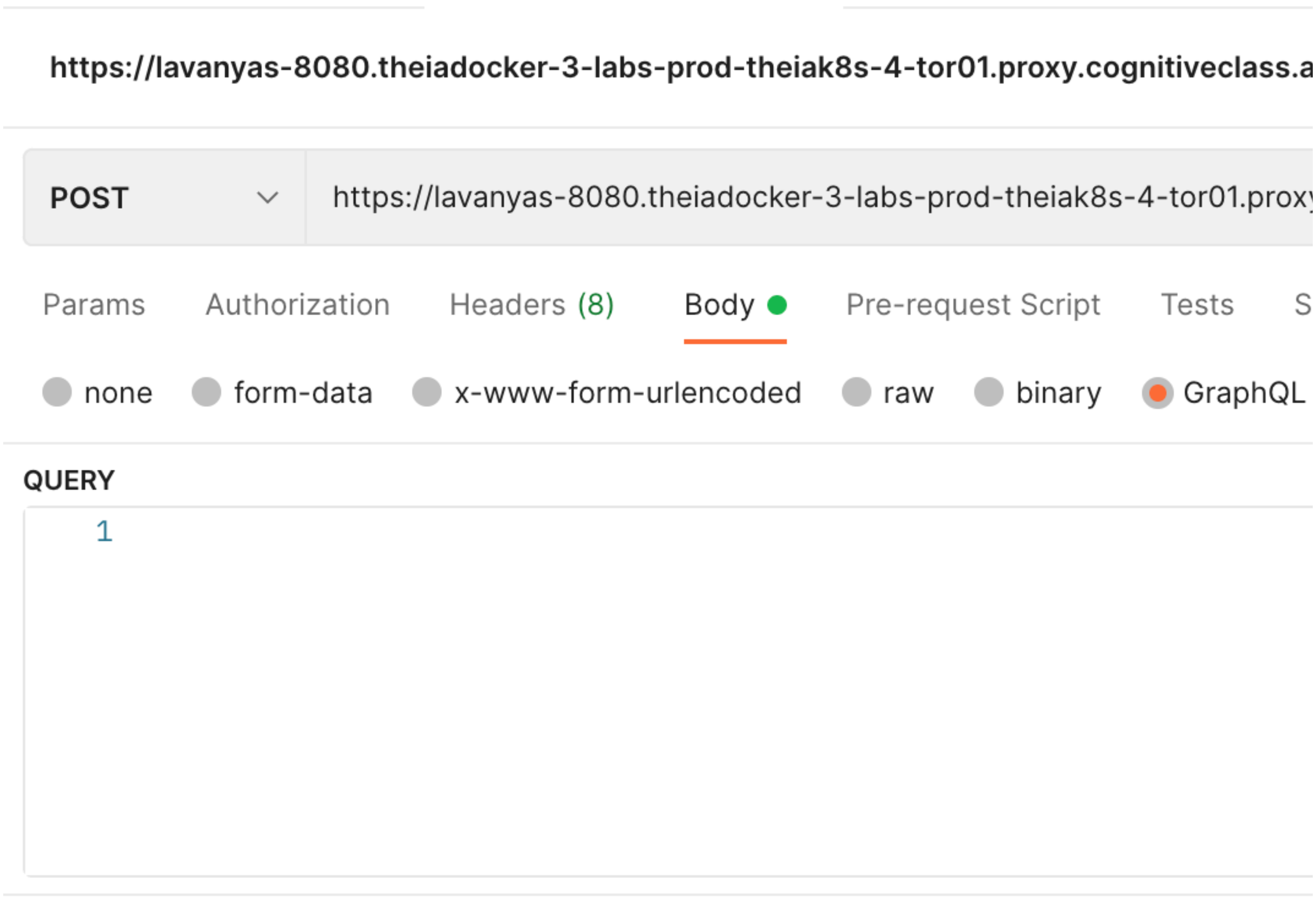
4. In the space provided for the **GET** request URL paste the URL you copied from step 8 of the previous task and suffix it with **/graphql**.



https://lavanyas-8080.theiadocker-3-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.a

GET https://lavanyas-8080.theiadocker-3-labs-prod-theiak8s-4-tor01.prox

5. Choose **Body** and **GraphQL** as shown in the image below to start entering your query. Method will change to **POST**.


https://lavanyas-8080.theiadocker-3-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.a

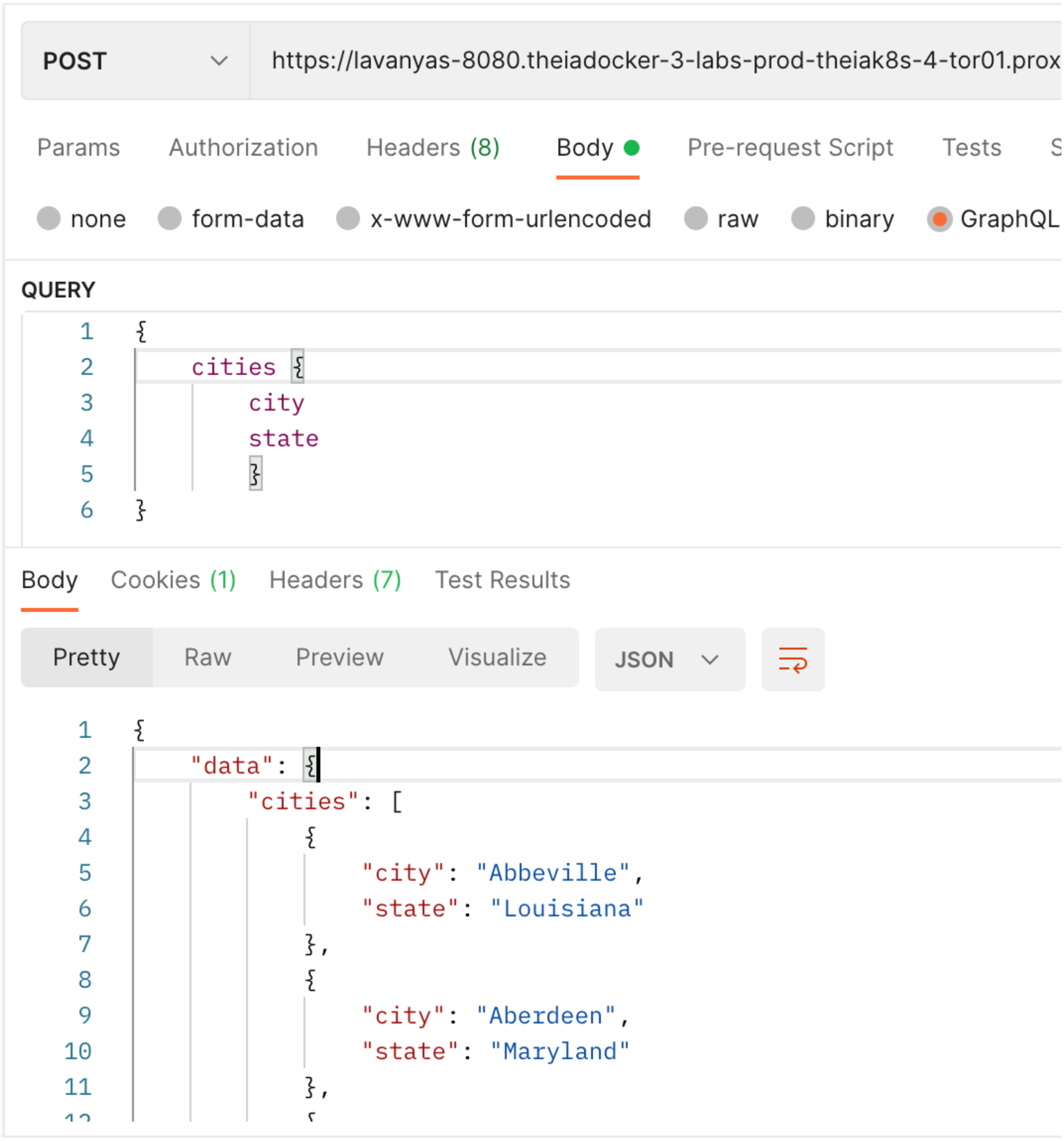
POST https://lavanyas-8080.theiadocker-3-labs-prod-theiak8s-4-tor01.proxy

none form-data x-www-form-urlencoded raw binary [✓ GraphQL]

QUERY

6. You will first query for all the cities in the US. What you can get is the city name and the state they are in. Paste the following in the **Query** tab and click **Send**.

```
{
    cities {
        city
        state
        }
}
```


POST https://lavanyas-8080.theiadocker-3-labs-prod-theiak8s-4-tor01.prox

Params Authorization Headers (8) Body ● Pre-request Script Tests S

none form-data x-www-form-urlencoded raw binary [✓ GraphQL]

QUERY

```
{
    cities {
        city
        state
        }
}
```

Body Cookies (1) Headers (7) Test Results

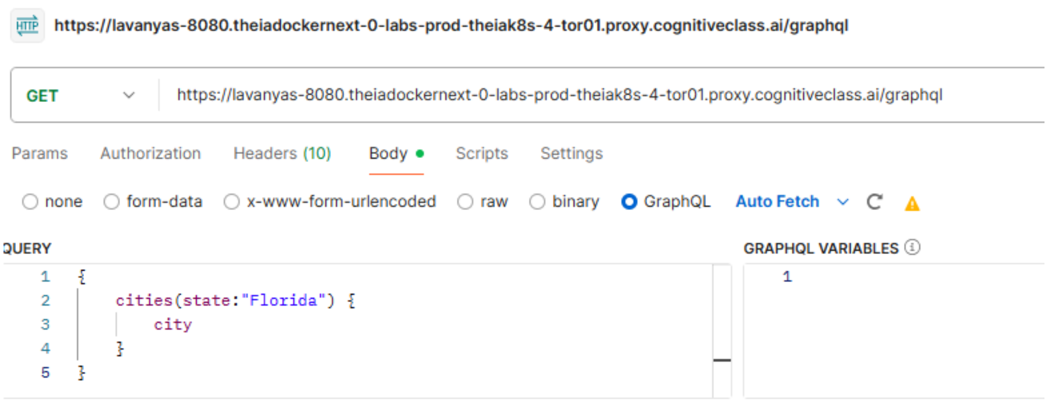
Pretty Raw Preview Visualize JSON

```
{
    "data": {
        "cities": [
            {
                "city": "Abbeville",
                "state": "Louisiana"
            },
            {
                "city": "Aberdeen",
                "state": "Maryland"
            },
            {
```

As you can see, it lists all the 5980 cities in the US in the output tab.

7. You can now try to retrieve just the names of the cities in a particular state. To do this, you need to pass the state as a parameter. You will now request the state name in the return value as you are querying only for one state.

```
{
    cities(state:"Florida") {
        city
    }
}
```


https://lavanyas-8080.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/graphql

GET https://lavanyas-8080.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/graphql

Params · Authorization · Headers (10) · Body · Scripts · Settings

○ none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ● GraphQL Auto Fetch

QUERY

```
{
    cities(state:"Florida") {
        city
    }
}
```

GRAPHQL VARIABLES

```
1
```

As you can see, only the names of the cities in Florida are returned.

Additional Task

1. Construct a query to return all the cities in the state of “Ohio”, along with the state name.

▶ Click here for the solution

Congratulations! You have learned to construct queries using GraphQL.

Tutorial details

Author: Lavanaya T S

Contributors: Pallavi Rai