**Aim** :- Developing REST APIs with API Gateway.

**Lab overview and objectives**

In this lab, you will create a REST application programming interface (API) by using Amazon API Gateway.

After completing this lab, you should be able to:

* Create simple mock endpoints for REST APIs and use them in your website.
* Enable Cross-Origin Resource Sharing (CORS)

**AWS service restrictions**

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

**Scenario**

In the *previous lab*, you took on the role of Sofía to build a web application for the café. As part of this process, you created an *Amazon DynamoDB table* that was named *FoodProducts*, where you stored information about café menu items.

You then loaded data that was formatted in JavaScript Object Notation (JSON) into the database table. The table structure looked similar to the following table (one line item of table data is shown as an example):

| **product\_name** | **description** | **price\_in\_cents** | **product\_id** | **tags** | **special** |
| --- | --- | --- | --- | --- | --- |
| apple pie slice | A delicious slice of Frank's homemade pie. | 595 | a444 | [ { "S" : "pie slice" }, { "S" : "on offer" } ] | 1 |

In the *previous* lab you also configured code that used the AWS SDK for Python (Boto3) to:

* Scan a DynamoDB table to retrieve product details.
* Return a single item by product name using get-item as a proof of concept
* Create a Global Secondary Index (GSI) called **special\_GSI** that you could use to filter out menu items that are on offer and not out of stock.

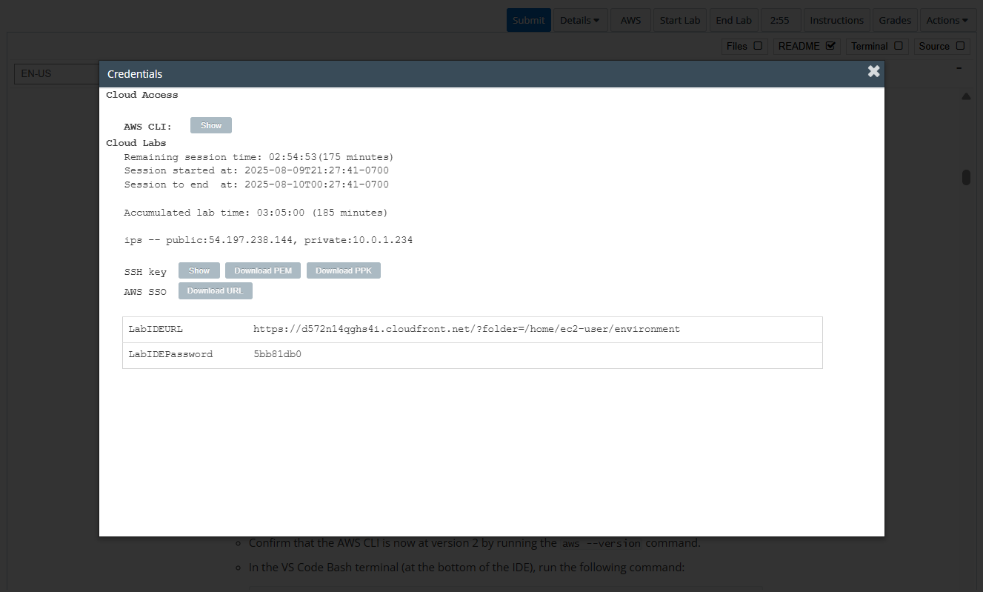
In *this* lab, you will continue to play the role of Sofía. You will use Amazon API Gateway to configure **mock data endpoints**. There are three that you will create:

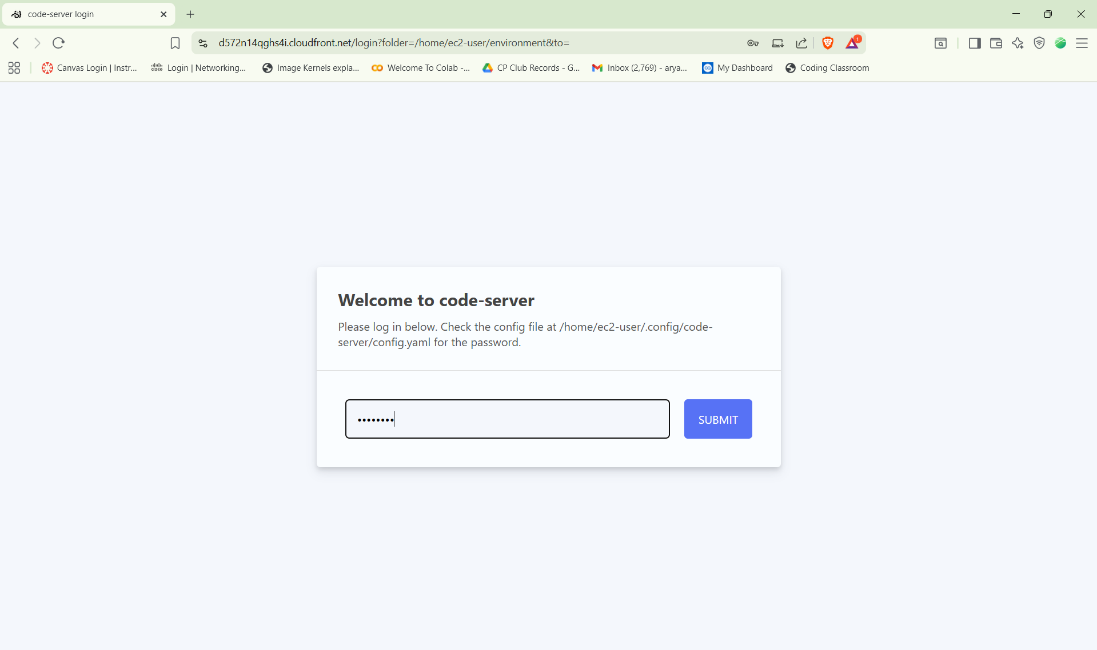
* [**GET**] /products (which will *eventually* invoke a DynamoDB table scan)
* [**GET**] /products/on\_offer (which will *eventually* invoke a DynamoDB index scan and filter)
* [**POST**] /create\_report (which will *eventually* trigger a batch process that will send out a report)

Then in the lab that follows this one, you will replace the mock endpoints with real endpoints, so that the web application can connect to the DynamoDB backend.

**Task 1: Preparing the lab**

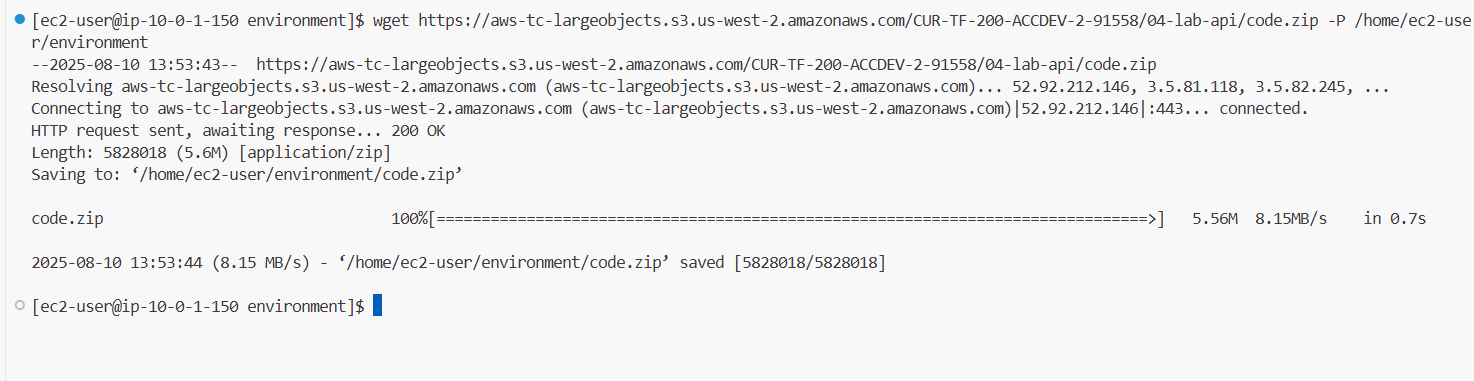
Connect to the VS Code IDE.

1. At the top of these instructions, choose Details followed by **AWS: Show**
2. Copy values from the table **similar** to the following and paste it into an editor of your choice for use later.
   1. **LabIDEURL**
   2. **LabIDEPassword**
3. In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
4. On the prompt window **Welcome to code-server**, enter the value for **LabIDEPassword** you copied to the editor earlier, choose **Submit** to open the VS Code IDE.

****

1. Download and extract the files that you need for this lab.

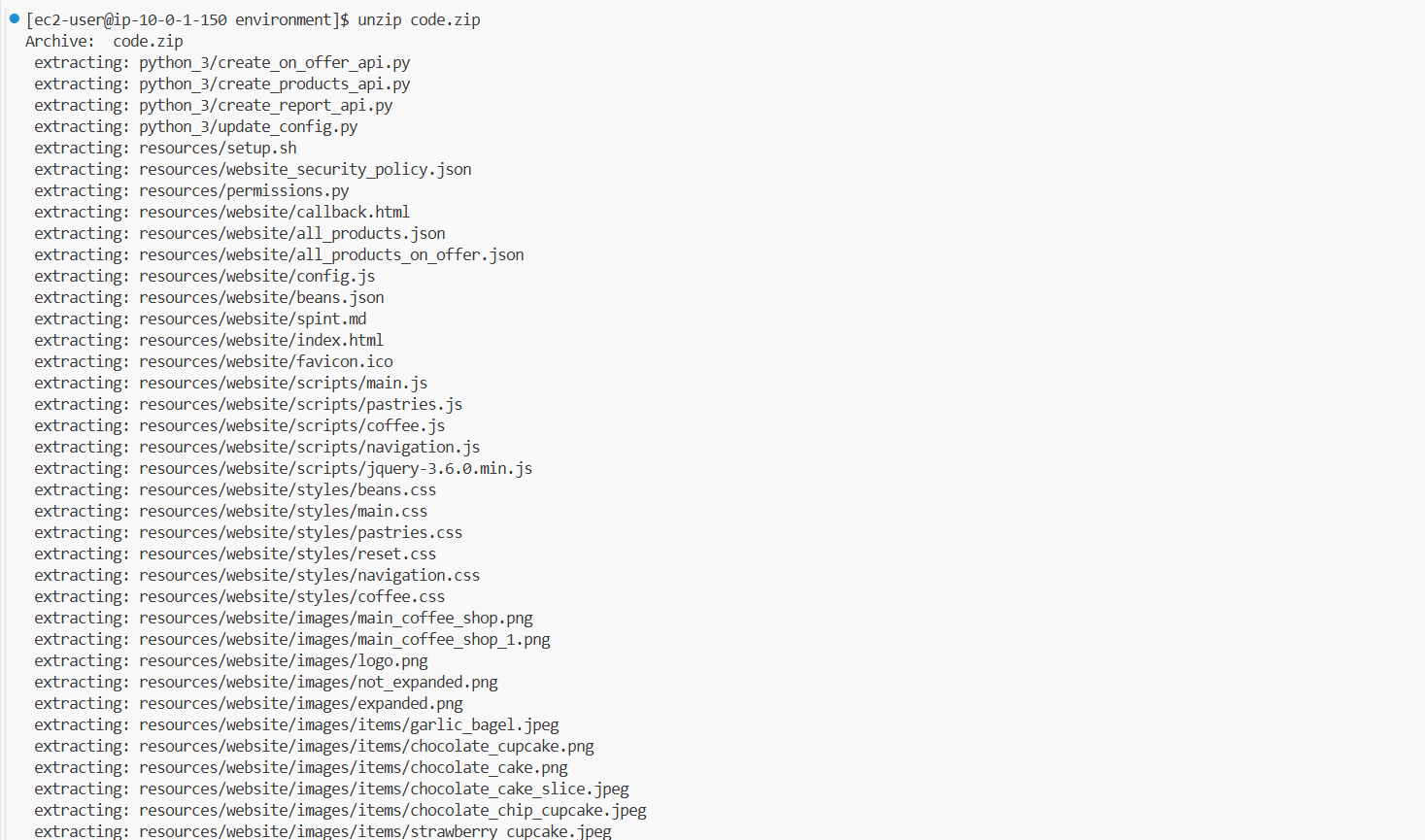
* In the VS Code bash terminal (located at the bottom of the IDE), run the following commands:

 wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/04-lab-api/code.zip -P /home/ec2-user/environment

1. You should see that the **code.zip** file was downloaded to the VS Code IDE and is now in the left navigation pane.

* Extract the file by running the following command:

unzip code.zip



1. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.

* To set permissions on the script and then run it, run the following commands in the Bash terminal:

chmod +x ./resources/setup.sh && ./resources/setup.sh

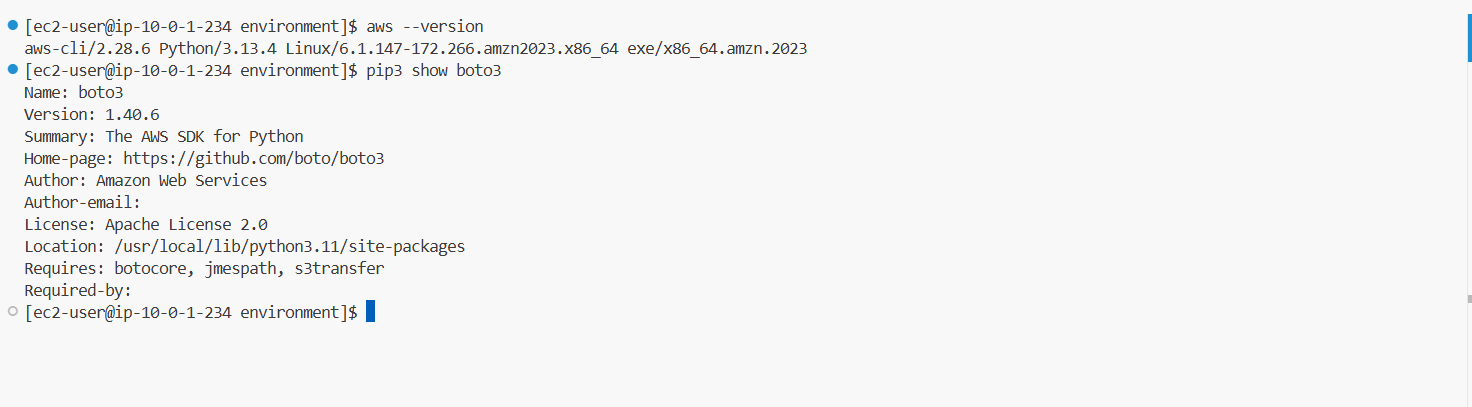
The script will prompt you for the **IP address** by which your computer is known to the internet.

Use [www.whatismyip.com](http://www.whatismyip.com/) to discover this address and then paste the IPv4 address into the command prompt and finish running the script.



1. Verify the AWS CLI version and also verify that the SDK for Python is installed.

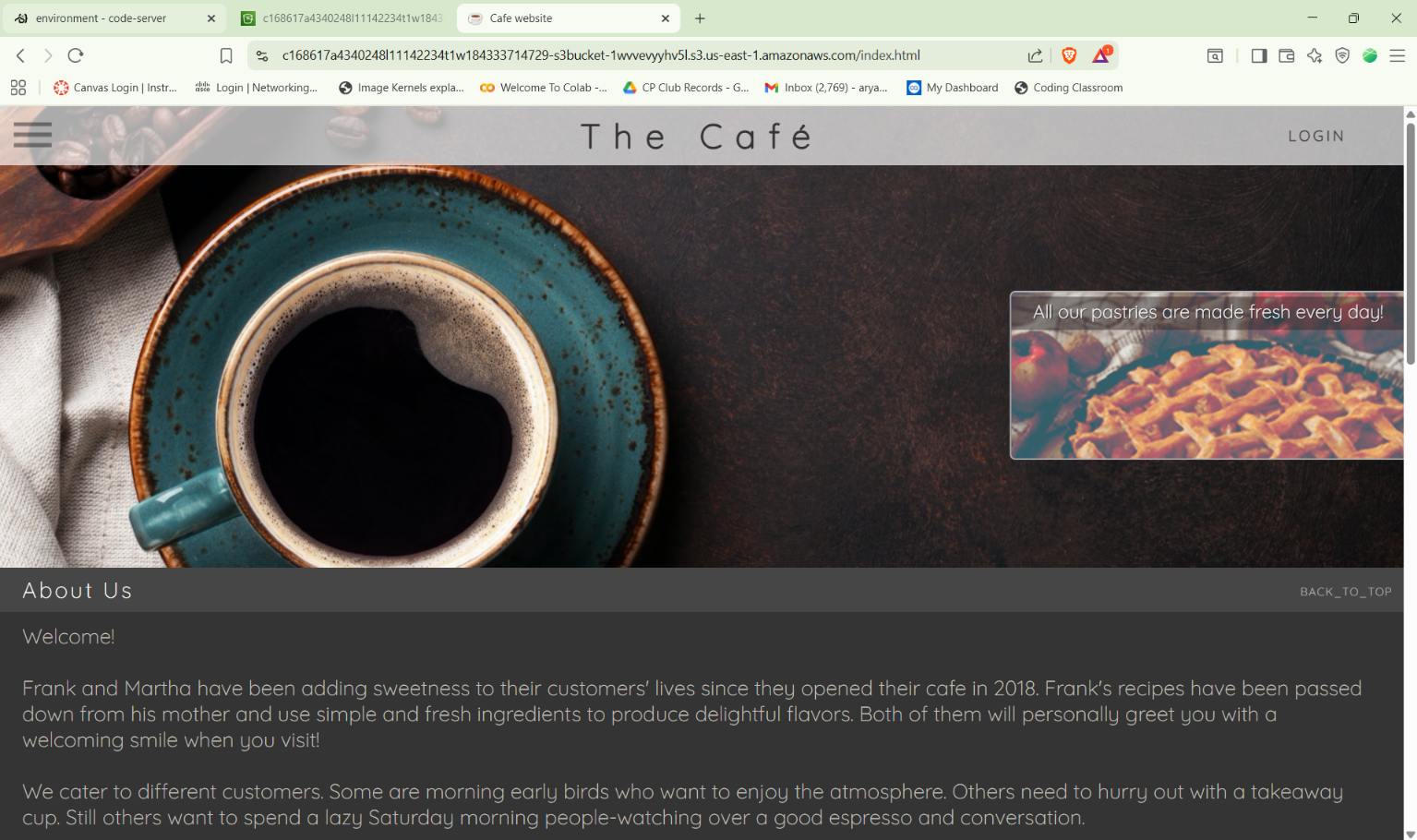
* Confirm that the AWS CLI is now at version 2 by running the **aws --version** command.
* In the VS Code Bash terminal (at the bottom of the IDE), run the following command:

**pip3 show boto3**

1. Verify that the cafe website can be loaded in a browser tab.
   * Load the website in a browser tab.
     + In a browser tab, open the Amazon S3 console.
     + Choose your bucket name, and then choose **Objects**.

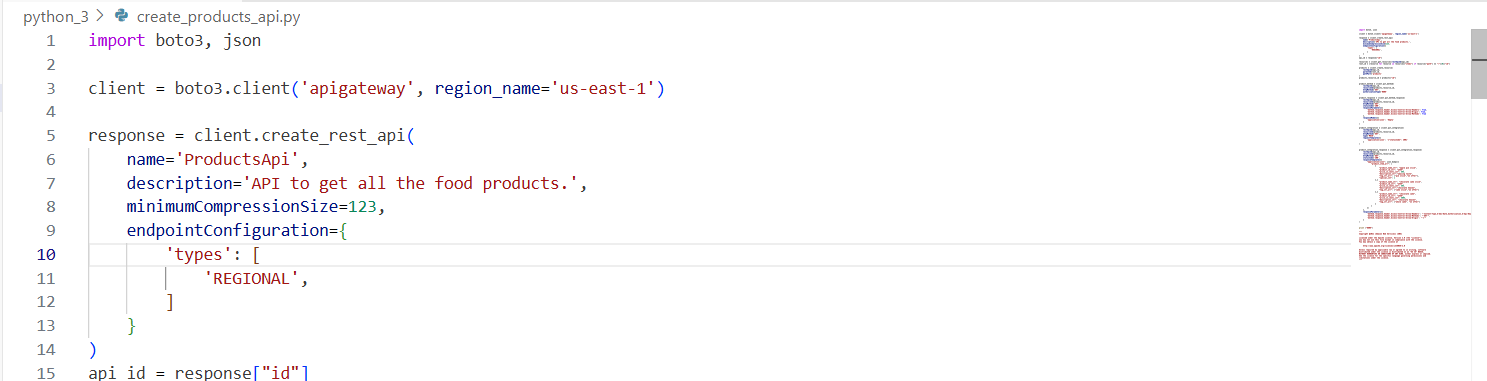
If the files that the script just uploaded do not display, choose the refresh icon to view them.

* + - Choose the **index.html** file.
    - Copy the **Object URL**. It will be in the following format. https://<bucket-name>.s3.amazon.com/index.html
    - Verify that the website displays by pasting the full URL into your browser.



**Task 2: Creating the first API endpoint (GET**

1. In the VS Code IDE navigation pane, expand the **python\_3** directory and open the file named **create\_products\_api.py**.

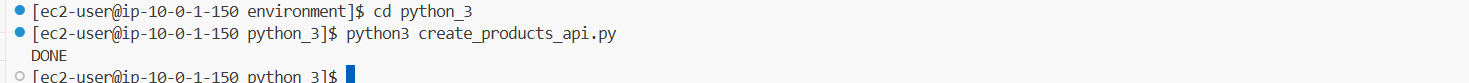


1. Run the code.

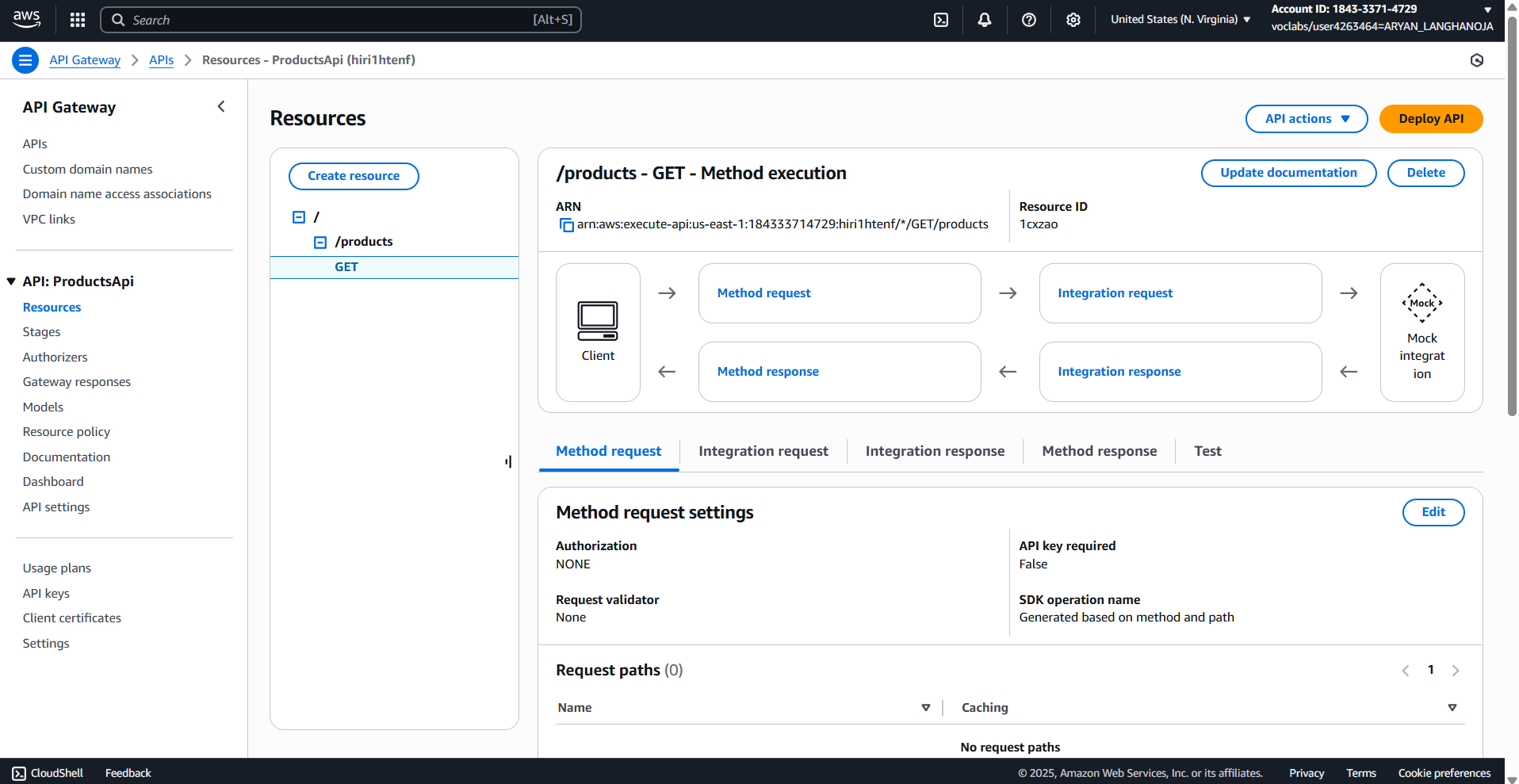
* In the VS Code Bash terminal, go to the directory that contains the Python code file, and run the code.

**cd python\_3**

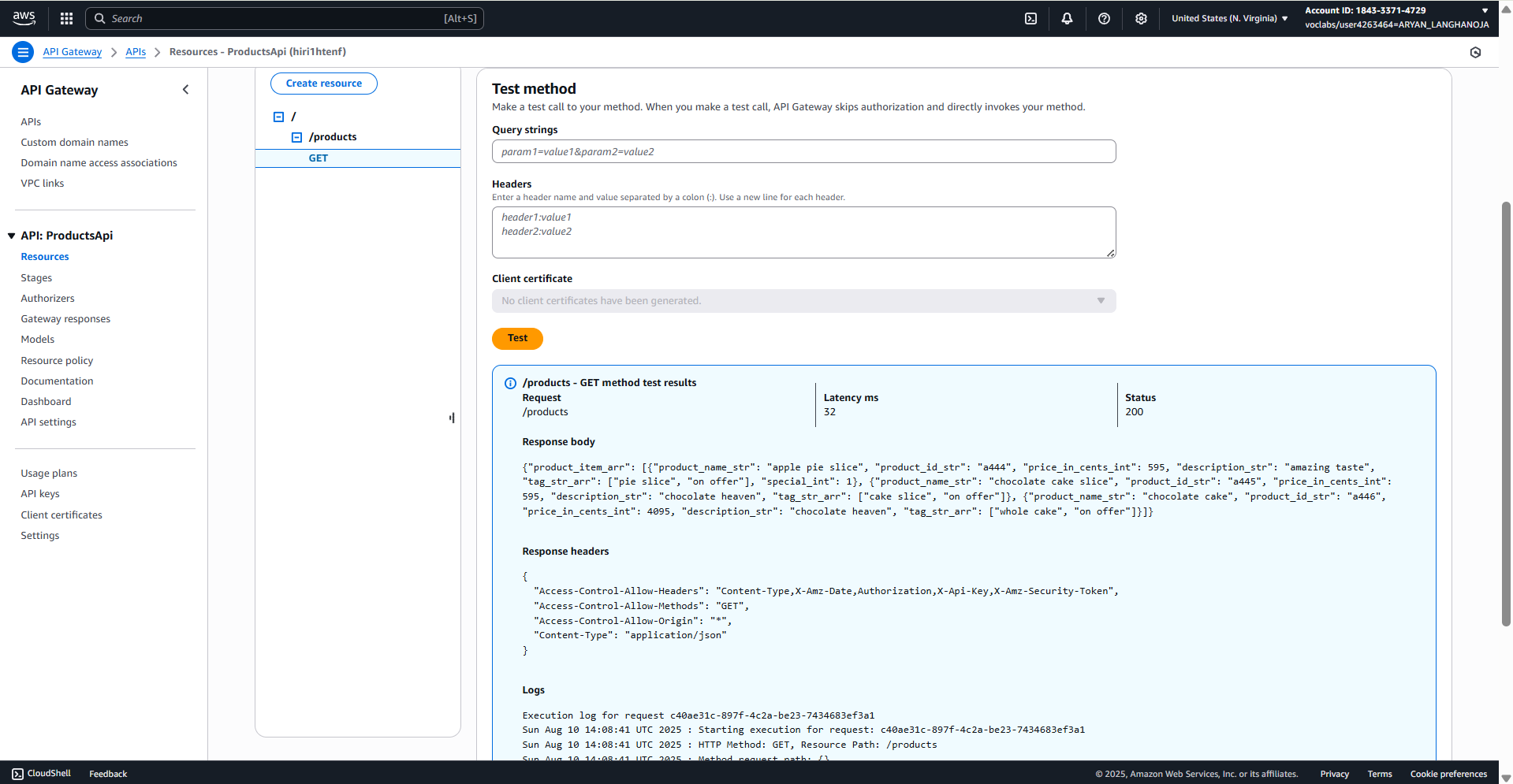
**python3 create\_products\_api.py**

****

1. Return to the **AWS Management Console** browser tab, and open the API Gateway console.
2. Open the **ProductsApi** that you just created by choosing the link.
3. Choose the **GET** method that you defined.

****

1. From the API Gateway console **lower panel**, choose the **TEST** link, then scroll to the bottom and choose the **Test** button.

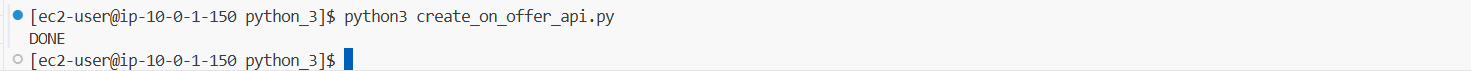


**Task 3: Creating the second API endpoint (GET)**

1. In the VS Code IDE navigation pane, expand the **python\_3** directory and open the file named **create\_on\_offer\_api.py**.
2. Replace <FMI\_1> and <FMI\_2> with the correct values so that this code file will add another resource to the API that you defined in the previous task.
   * In a browser tab, go to the **API Gateway** console and choose the **ProductsApi** API that you created a moment ago.
   * In the panel on the left, choose **Resources**.
   * Choose **GET** under products
   * In the breadcrumb navigation at the top of the screen (above the Actions menu), you can see APIs > **ProductsAPI** followed by an id in parenthesis.
     + This is the **api\_id**.
   * On the same line, you will see **/** at the top, you will noticed **Resource ID**.
     + This is the resource **parent\_id**
3. Create the API resource.

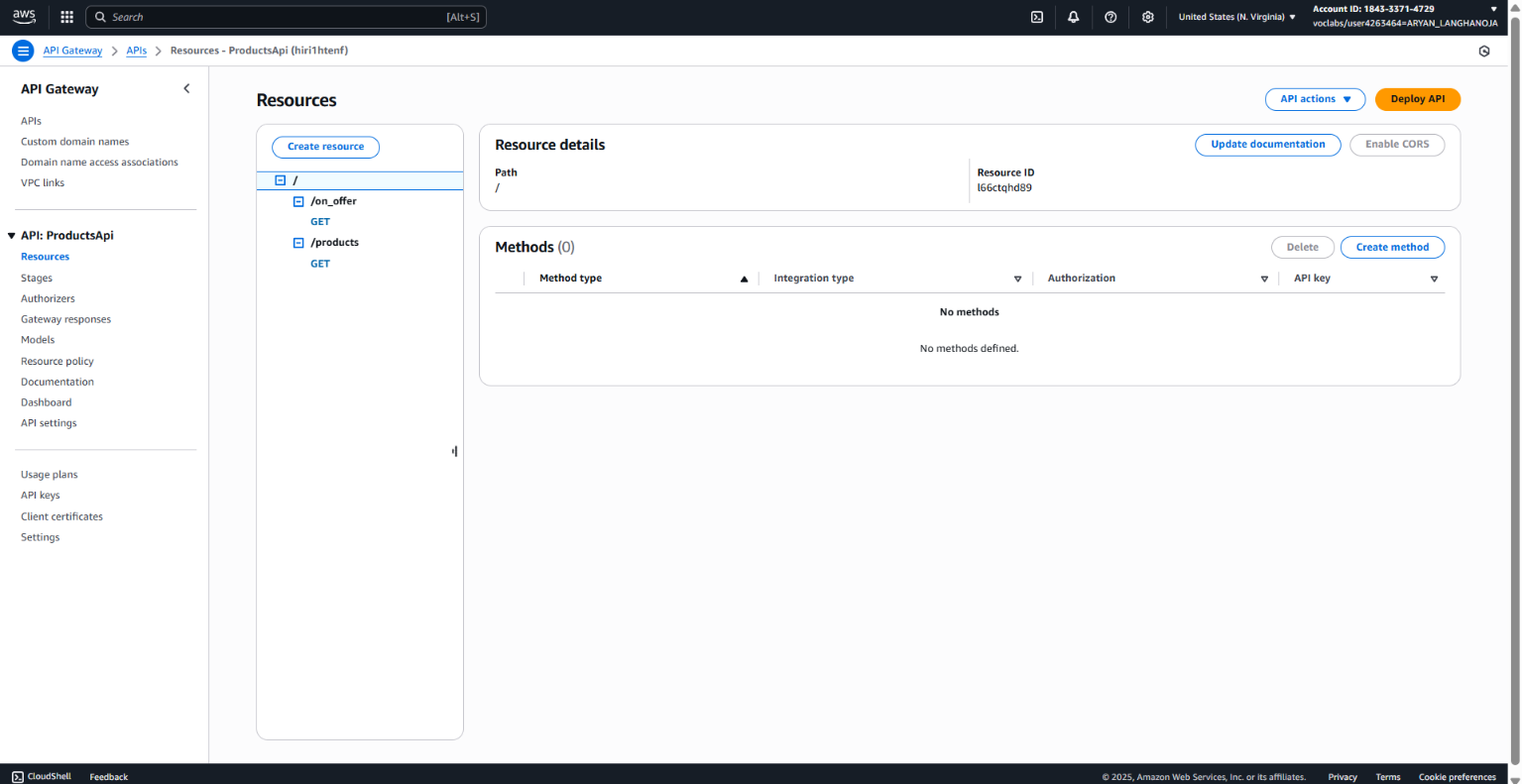
* Close the file by choosing **X** from the top. (Your changes are saved automatically).
* Then in the Bash terminal, verify that the current directory is python\_3 and run the code.

**python3 create\_on\_offer\_api.py**



1. Observe the results.

* Return to the **AWS Management Console** browser tab, and open the **API Gateway** console.
* Choose the **APIs** link in the breadcrumb navigation above, then on the left, open the **ProductsApi** by choosing the link.
* Notice that there is now a nested resource called /on\_offer under the /products resource.



1. Test the /on\_offer resource.

* From the lower pane **Test**, choose the **Test** button to test the same way you tested the first resource in the previous task.
* You should receive a 200 HTML status code response.



**Task 4: Creating the third API endpoint (POST)**

21. In the VS Code IDE, if the **create\_products\_api.py** file is not already open, open it (you ran this file in Task 2).

22. Next, in the **python\_3** directory, also open the **create\_report\_api.py** file.

23. In the main code editor window, *right-click* the **create\_report\_api.py** file tab and choose **Split Pane in Two Columns**.

24. Analyze and update the **create\_report\_api.py** code. Be sure to compare the code in this file to the **create\_products\_api.py** code while you do the analysis and updates.

* Replace the <FMI\_1> that appears on line 5 with the correct value.

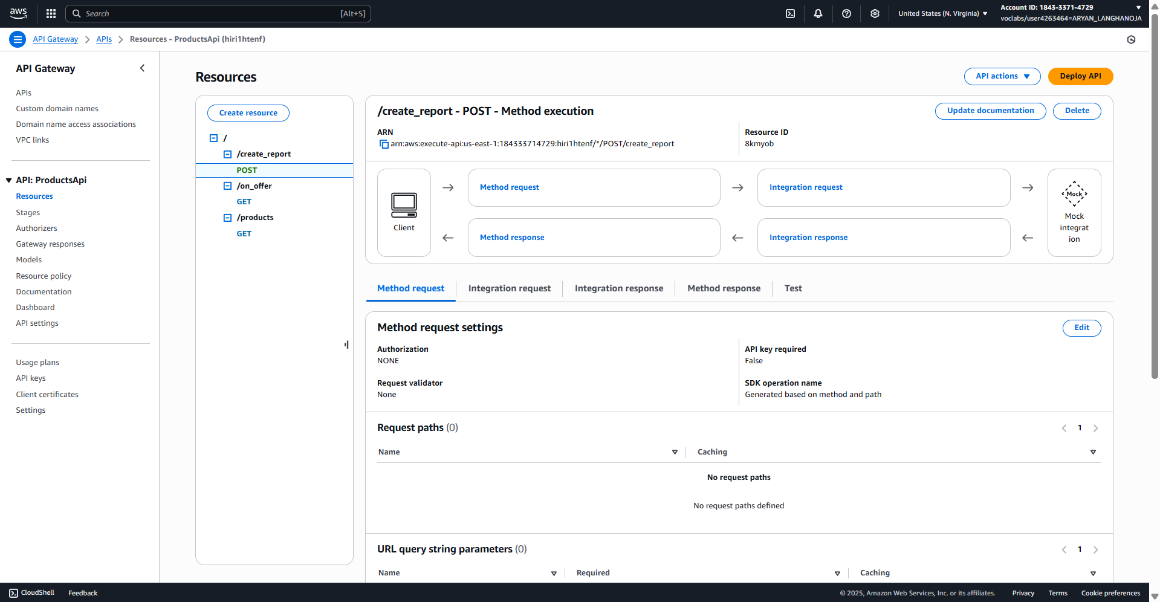
25. In the terminal, confirm that you are in the **python\_3** directory, and then run the code to create the third endpoint.

**python3 create\_report\_api.py**

****

26. In the API Gateway console, view the details of the *report* API that you configured.

* Return to the API Gateway console tab and refresh the page.
* Confirm that you are in *ProductsApi*.
* In the navigation pane, confirm that **Resources** is selected, and choose **/create\_report** > **POST**.

You should see the details of the POST method execution.

27. From the lower pane, choose **TEST** , then choose the **Test** button at the bottom of the screen.

In the panel, you should see the following response body, response headers, and log information.



**Task 5: Deploying the API**

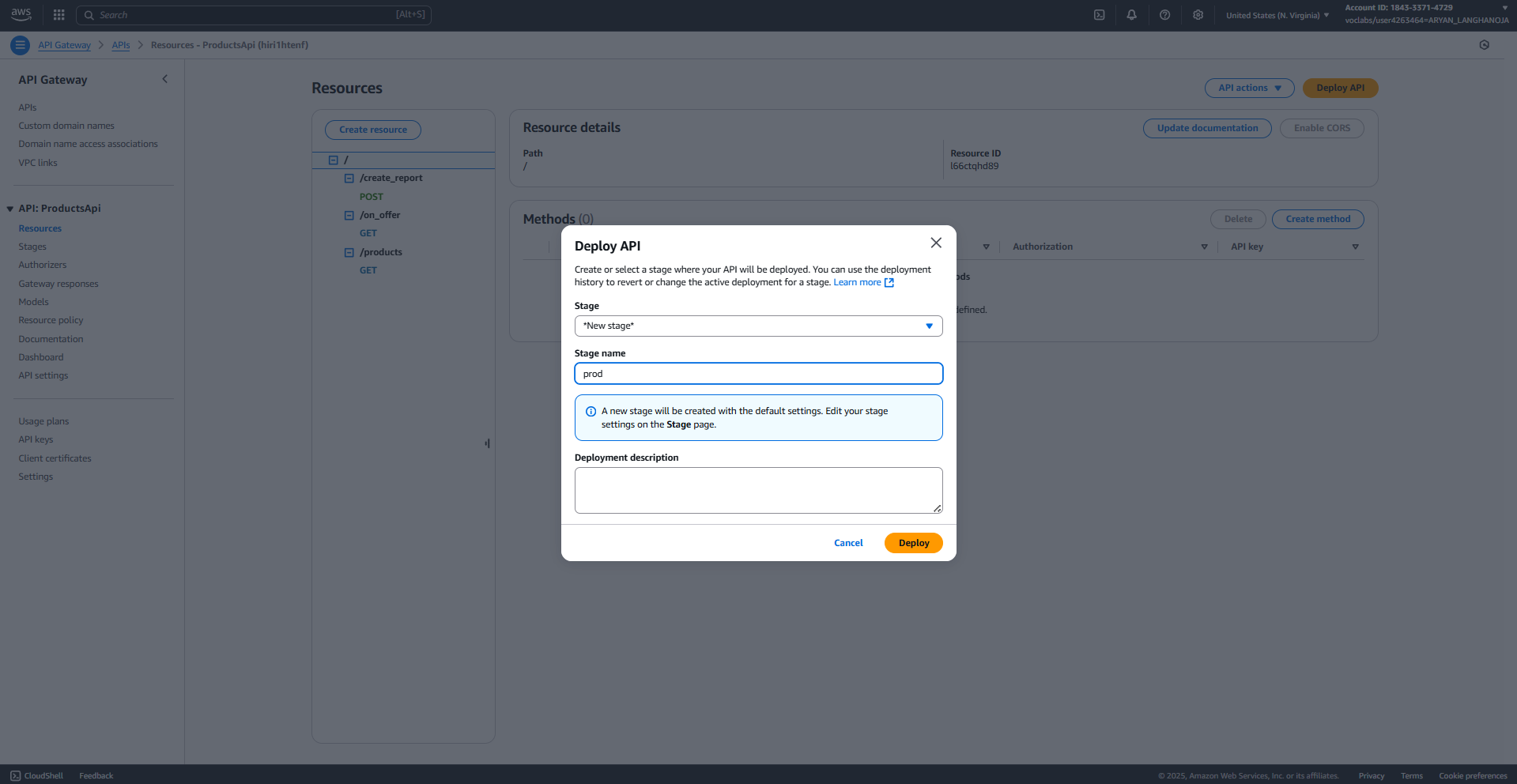
Now that you have defined all three resources in the API, the next step is to deploy the API.

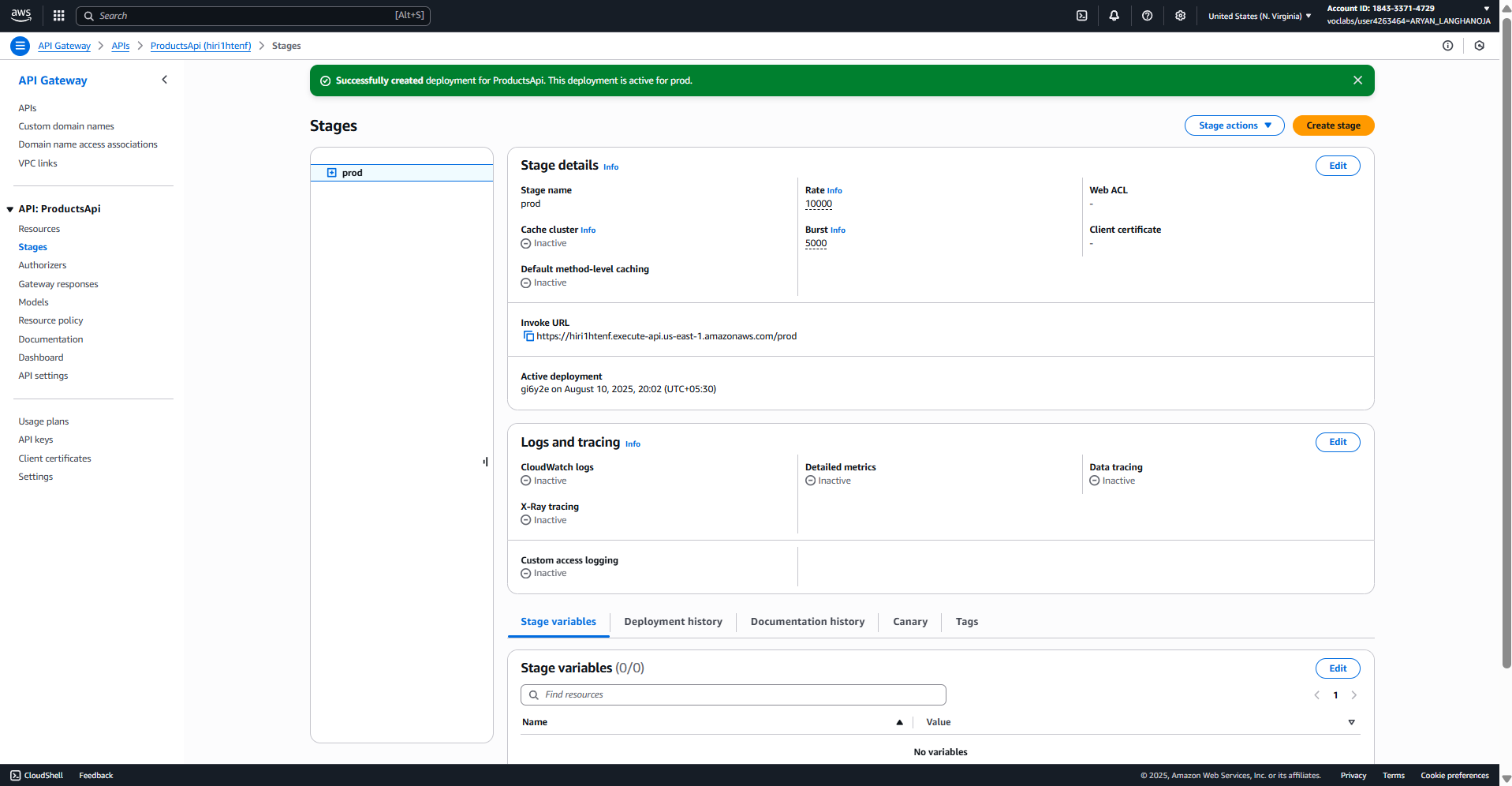
28. Deploy the API.

* + Still in the API Gateway console where you have the ProductsApi details open, under **Resources** select the root /
  + From the top, choose **Deploy API** button and then fill in the details:.
    - Deployment stage: **\*New Stage\***.
    - Stage name: **prod**
    - Stage description: (leave blank)
    - Deployment description: (leave blank)
  + Choose **Deploy**

**Tip**: If you see a warning that you do not have ListWebACLs and AssociateWebACL permissions for Web Application Firewall (WAF Regional), you can ignore the message and close it.

29. Copy the **Invoke URL** value to your clipboard. You will use it next.



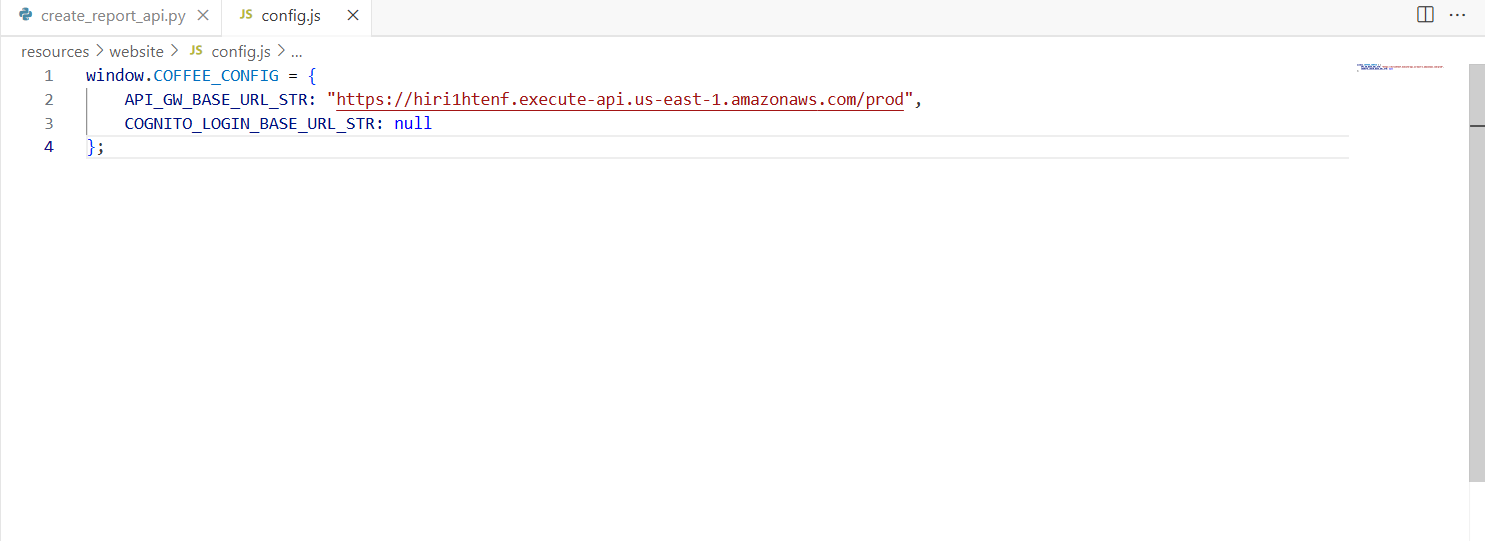


**Task 6: Updating the website to use the APIs**

In this final task in the lab, you will update and then test the website files that are hosted on Amazon S3. After you complete these updates, the website will invoke the REST API that you just created.

30. Update the website's **config.js** file.

* + In the VS Code IDE, open resources/website/**config.js**



31.Update and then run the **update\_config.py** script.

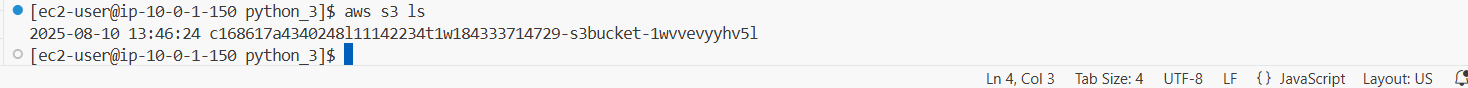
* + Open **python\_3/update\_config.py** in the text editor.
  + Replace the <FMI\_1> placeholder with the name of your bucket.

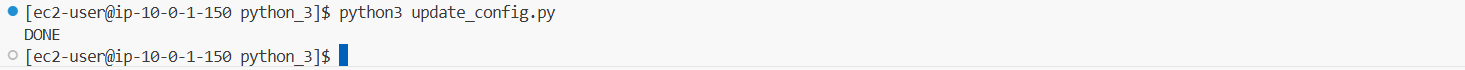
**Tip**: You can find the bucket name in the S3 console, or by running this command:

**aws s3 ls**

* + Notice that this script uploads the config.js file that you just editing the previous step, and uploads it to the S3 bucket.
  + Close the file by choosing **X** from the top. (Your changes are saved automatically).

**python3 update\_config.py**





**Conclusion:-**

* In this Lab I Learned the AWS API Gateway Service.
* I had created the REST API of GET Method /product to retrive all the product.
* I had created the REST API of GET Method /product/on\_offer to retrive all the product having offer.
* I had created the REST API of POST Method /create\_report to create the report.
* I had deployed that API