PRACTICAL: 4

AIM:

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 Food Products, where you stored information about café menu items. You then loaded data that was formatted in JavaScript Object Notation (JSON) into the database table. 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.

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 do the following:

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).

THEORY:

In this lab, you will create a mock REST API using **Amazon API Gateway** to simulate backend functionality for a café web application. These mock endpoints will act as placeholders for real backend operations, allowing you to test your application without direct integration with a database or other backend services. You will also enable **Cross-Origin Resource Sharing (CORS)** to allow your frontend application to communicate with the API Gateway.

The mock endpoints you will create include:

1. **[GET] /products**: Simulates retrieving all menu items, representing a DynamoDB table scan.

2. **[GET] /products/on_offer**: Simulates filtering menu items on offer, using a Global Secondary Index (GSI).
3. **[POST] /create_report**: Simulates triggering a batch process for generating reports.

By creating these endpoints, you will:

- Test API functionality without actual database integration.
- Enable frontend and backend teams to work independently.
- Simulate edge cases and handle error responses during development.

You will also configure **CORS** to ensure your API accepts requests from other domains securely, allowing seamless communication between your web application and the API.

At the end of the lab, you will have a fully functional mock API that lays the groundwork for replacing mock endpoints with real database connections in the next phase of development. This approach enables rapid iteration, thorough testing, and flexibility during the development process.

CODE:

create_products_api.py

```
import boto3, json

client = boto3.client('apigateway', region_name='us-east-1')

response = client.create_rest_api(
    name='ProductsApi',
    description='API to get all the food products.',
    minimumCompressionSize=123,
    endpointConfiguration={
        'types': [
            'REGIONAL',
        ]
    }
)
api_id = response["id"]

resources = client.get_resources(restApiId=api_id)
root_id = [resource for resource in resources["items"] if resource["path"] ==
"/"][0]["id"]

products = client.create_resource(
    restApiId=api_id,
    parentId=root_id,
    pathPart='products'
)
products_resource_id = products["id"]


product_method = client.put_method(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    authorizationType='NONE'
)

product_response = client.put_method_response(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    statusCode='200',
    responseParameters={
        'method.response.header.Access-Control-Allow-Headers': True,
        'method.response.header.Access-Control-Allow-Origin': True,
        'method.response.header.Access-Control-Allow-Methods': True
    },
    responseModels={
        'application/json': 'Empty'
    }
)

product_integration = client.put_integration(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    type='MOCK',
    requestTemplates={
        'application/json': '{"statusCode": 200}'
    }
)


product_integration_response = client.put_integration_response(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    statusCode='200',
    responseTemplates={
        "application/json": json.dumps({
            "product_item_arr": [
                {
                    "product_name_str": "apple pie slice",
                    "product_id_str": "a444",
                    "price_in_cents_int": 595,
                    "description_str":"amazing taste",
                    "tag_str_arr": ["pie slice","on offer"],
                    "special_int": 1
                },{
                    "product_name_str": "chocolate cake slice",
                    "product_id_str": "a445",
                    "price_in_cents_int": 595,
                    "description_str":"chocolate heaven",
                    "tag_str_arr": ["cake slice","on offer"]
                },{
                    "product_name_str": "chocolate cake",
                    "product_id_str": "a446",
                    "price_in_cents_int": 4095,
                    "description_str": "chocolate heaven",
                    "tag_str_arr": ["whole cake", "on offer"]
                }
            ]
        })
    },
    responseParameters={
        'method.response.header.Access-Control-Allow-Headers': '"Content-Type,X-Amz
Date,Authorization,X-Api-Key,X-Amz-Security-Token"',
        'method.response.header.Access-Control-Allow-Methods': '"GET"',
        'method.response.header.Access-Control-Allow-Origin': '"*"'
    }
)
```

create_on_offer_api.py

```
import boto3, json

client = boto3.client('apigateway', region_name='us-east-1')



api_id = 'mmx908m544'
parent_id = 'p454lc'
```

```
products = client.create_resource(
    restApiId=api_id,
    parentId=parent_id,
    pathPart='on_offer'
)
products_resource_id = products["id"]


product_method = client.put_method(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    authorizationType='NONE'
)

product_response = client.put_method_response(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    statusCode='200',
    responseParameters={
        'method.response.header.Access-Control-Allow-Headers': True,
        'method.response.header.Access-Control-Allow-Origin': True,
        'method.response.header.Access-Control-Allow-Methods': True
    },
    responseModels={
        'application/json': 'Empty'
    }
)

product_integration = client.put_integration(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    type='MOCK',
    requestTemplates={
        'application/json': '{"statusCode": 200}'
    }
)


product_integration_response = client.put_integration_response(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    statusCode='200',
    responseTemplates={
        "application/json": json.dumps({
            "product_item_arr": [{
                "product_name_str": "apple pie slice",
                "product_id_str": "a444",
                "price_in_cents_int": 595,
                "description_str": "amazing taste",
                "tag_str_arr": [
                  "pie slice",
                  "on offer"
                ],
                "special_int": 1
              }]
        })
    },
    responseParameters={
        'method.response.header.Access-Control-Allow-Headers': "'Content-Type,X-Amz
Date,Authorization,X-Api-Key,X-Amz-Security-Token'",
        'method.response.header.Access-Control-Allow-Methods': "'GET'",
        'method.response.header.Access-Control-Allow-Origin': "'*'"
    }
)
```

create_report_api.py

```
import boto3, json

client = boto3.client('apigateway', region_name='us-east-1')

api_id = 'mmx908m544'

resources = client.get_resources(restApiId=api_id)
root_id = [resource for resource in resources["items"] if resource["path"] ==
"/"][0]["id"]




report_resource = client.create_resource(
    restApiId=api_id,
    parentId=root_id,
    pathPart='create_report'
```

```
)
report_resource_id = report_resource["id"]



report_method = client.put_method(
    restApiId=api_id,
    resourceId=report_resource_id,
    httpMethod='POST',
    authorizationType='NONE'
)

report_response = client.put_method_response(
    restApiId=api_id,
    resourceId=report_resource_id,
    httpMethod='POST',
    statusCode='200',
    responseParameters={
        'method.response.header.Access-Control-Allow-Headers': False,
        'method.response.header.Access-Control-Allow-Origin': False,
        'method.response.header.Access-Control-Allow-Methods': False
    },
    responseModels={
        'application/json': 'Empty'
    }
)


report_integration = client.put_integration(
    restApiId=api_id,
    resourceId=report_resource_id,
    httpMethod='POST',
    type='MOCK',
    requestTemplates={
        'application/json': '{"statusCode": 200}'
    }
)


report_integration_response = client.put_integration_response(
    restApiId=api_id,
    resourceId=report_resource_id,
    httpMethod='POST',
    statusCode='200',
    responseParameters={
        'method.response.header.Access-Control-Allow-Headers': '\'Content-Type,X-
Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token\'',
        'method.response.header.Access-Control-Allow-Methods': '\'POST\'',
        'method.response.header.Access-Control-Allow-Origin': '\'*\''
    },
    # ithink they have an issue with real JSON here
    responseTemplates={
        "application/json": json.dumps({
            "msg_str": "report requested, check your phone shortly"
        })
    }
)
```

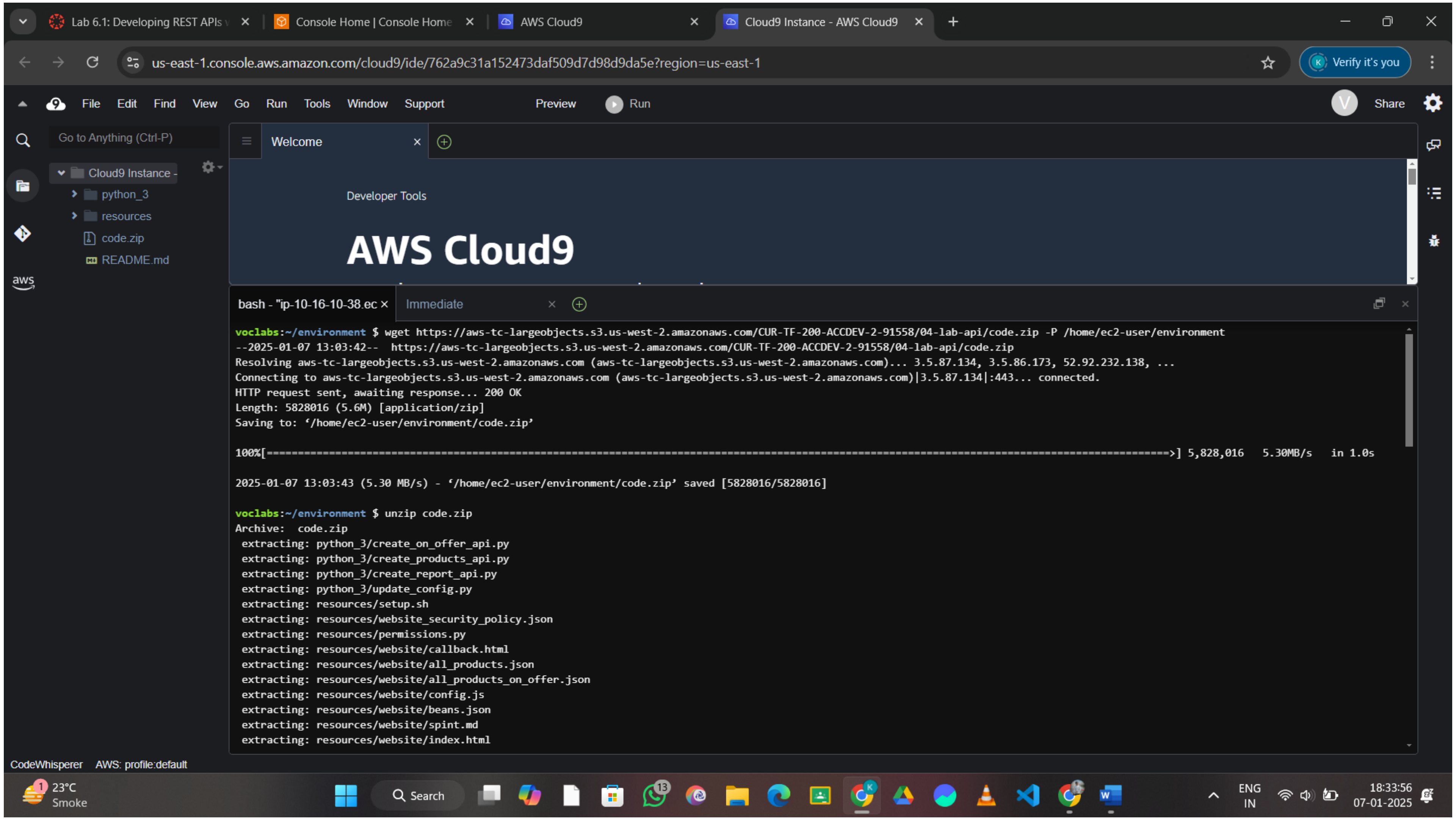
Config.js

```
window.COFFEE_CONFIG = {
          API_GW_BASE_URL_STR: "https://mmx908m544.execute-api.us-east-
1.amazonaws.com/prod",
          COGNITO_LOGIN_BASE_URL_STR: null
};
```

update_config.py

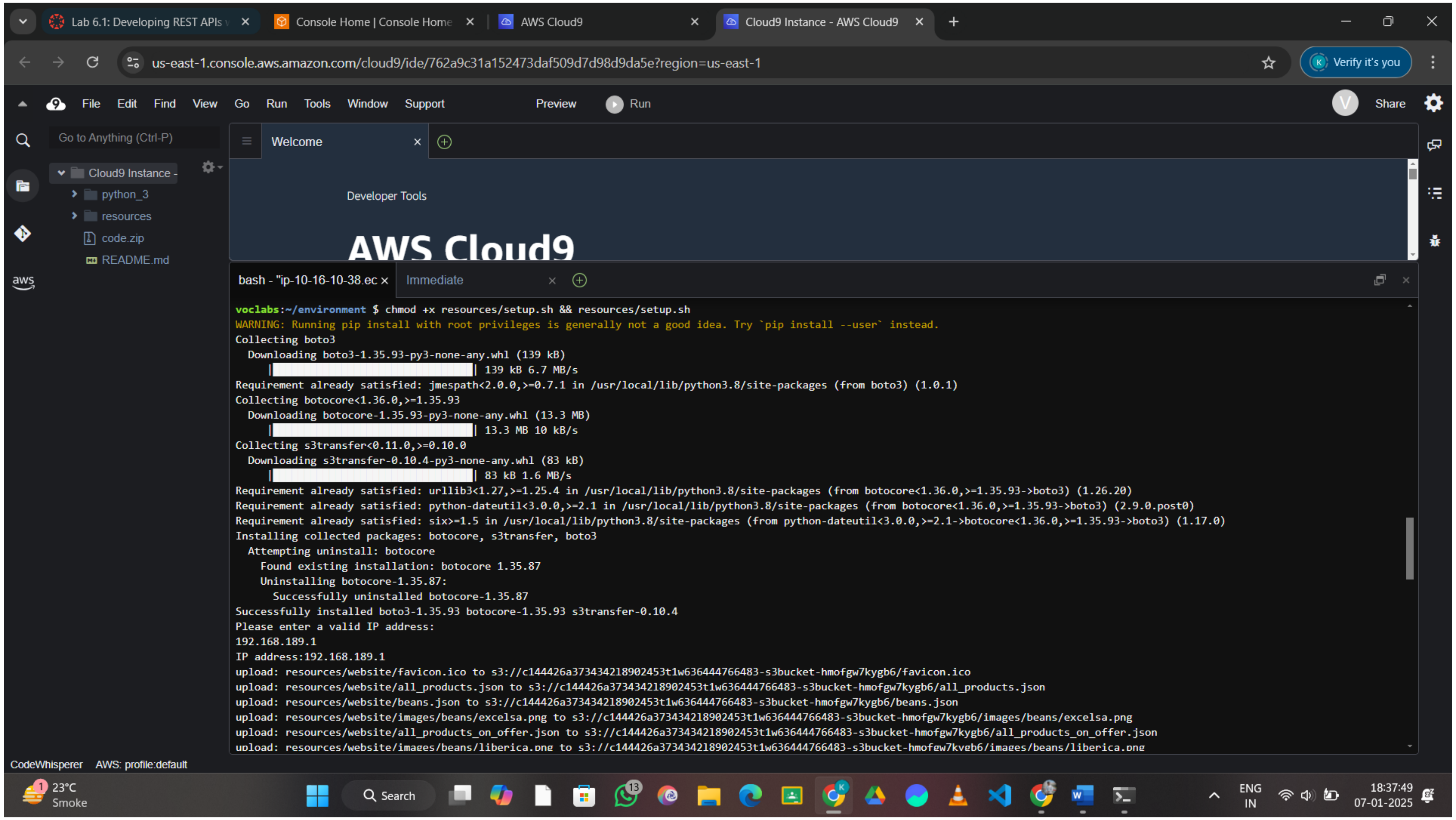
```
import boto3
S3API = boto3.client("s3", region_name="us-east-1")
bucket_name = "c144426a3734342l8902453t1w636444766483-s3bucket-
hmofgw7kygb6"

filename = "/home/ec2-user/environment/resources/website/config.js"
S3API.upload_file(filename, bucket_name, "config.js", ExtraArgs={'ContentType':
"application/js", "CacheControl": "max-age=0"})
```

OUTPUT:



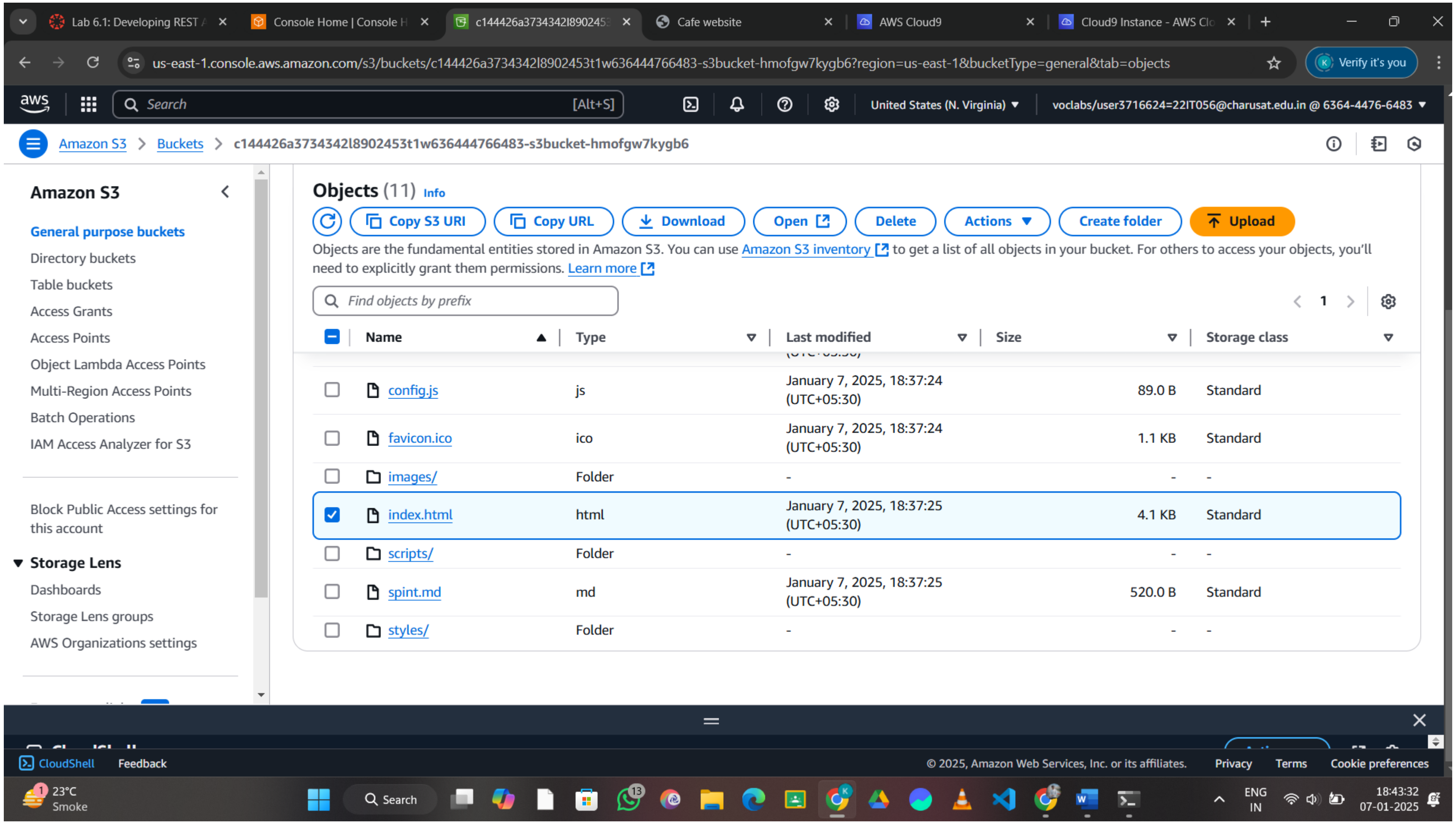
```
voclabs:~/environment $ wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/04-lab-api/code.zip -P /home/ec2-user/environment
--2025-01-07 13:03:42--  https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/04-lab-api/code.zip
Resolving aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)... 3.5.87.134, 3.5.86.173, 52.92.232.138, ...
Connecting to aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)|3.5.87.134|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5828016 (5.6M) [application/zip]
Saving to: ‘/home/ec2-user/environment/code.zip’

100%[==============================================================================================================================>] 5,828,016   5.30MB/s   in 1.0s

2025-01-07 13:03:43 (5.30 MB/s) - ‘/home/ec2-user/environment/code.zip’ saved [5828016/5828016]

voclabs:~/environment $ unzip code.zip
Archive:  code.zip
 extracting: python_3/create_on_offer_api.py
 extracting: python_3/create_products_api.py
 extracting: python_3/create_report_api.py
 extracting: python_3/update_config.py
 extracting: resources/setup.sh
 extracting: resources/website_security_policy.json
 extracting: resources/permissions.py
 extracting: resources/website/callback.html
 extracting: resources/website/all_products.json
 extracting: resources/website/all_products_on_offer.json
 extracting: resources/website/config.js
 extracting: resources/website/beans.json
 extracting: resources/website/spint.md
 extracting: resources/website/index.html
```

Figure 1: Download a zip file and extract it



```
voclabs:~/environment $ chmod +x resources/setup.sh && resources/setup.sh
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip install --user` instead.
Collecting boto3
  Downloading boto3-1.35.93-py3-none-any.whl (139 kB)
     |████████████████████████████████| 139 kB 6.7 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.8/site-packages (from boto3) (1.0.1)
Collecting botocore<1.36.0,>=1.35.93
  Downloading botocore-1.35.93-py3-none-any.whl (13.3 MB)
     |████████████████████████████████| 13.3 MB 10 kB/s
Collecting s3transfer<0.11.0,>=0.10.0
  Downloading s3transfer-0.10.4-py3-none-any.whl (83 kB)
     |████████████████████████████████| 83 kB 1.6 MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/local/lib/python3.8/site-packages (from botocore<1.36.0,>=1.35.93->boto3) (1.26.20)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.8/site-packages (from botocore<1.36.0,>=1.35.93->boto3) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.36.0,>=1.35.93->boto3) (1.17.0)
Installing collected packages: botocore, s3transfer, boto3
  Attempting uninstall: botocore
    Found existing installation: botocore 1.35.87
    Uninstalling botocore-1.35.87:
      Successfully uninstalled botocore-1.35.87
Successfully installed boto3-1.35.93 botocore-1.35.93 s3transfer-0.10.4
Please enter a valid IP address:
192.168.189.1
IP address:192.168.189.1
upload: resources/website/favicon.ico to s3://c144426a373434218902453t1w636444766483-s3bucket-hmofgw7kygb6/favicon.ico
upload: resources/website/all_products.json to s3://c144426a373434218902453t1w636444766483-s3bucket-hmofgw7kygb6/all_products.json
upload: resources/website/beans.json to s3://c144426a373434218902453t1w636444766483-s3bucket-hmofgw7kygb6/beans.json
upload: resources/website/images/beans/excelsa.png to s3://c144426a373434218902453t1w636444766483-s3bucket-hmofgw7kygb6/images/beans/excelsa.png
upload: resources/website/all_products_on_offer.json to s3://c144426a373434218902453t1w636444766483-s3bucket-hmofgw7kygb6/all_products_on_offer.json
upload: resources/website/images/beans/liberica.png to s3://c144426a373434218902453t1w636444766483-s3bucket-hmofgw7kygb6/images/beans/liberica.png
```

Figure 2: Set up resources with our ip address



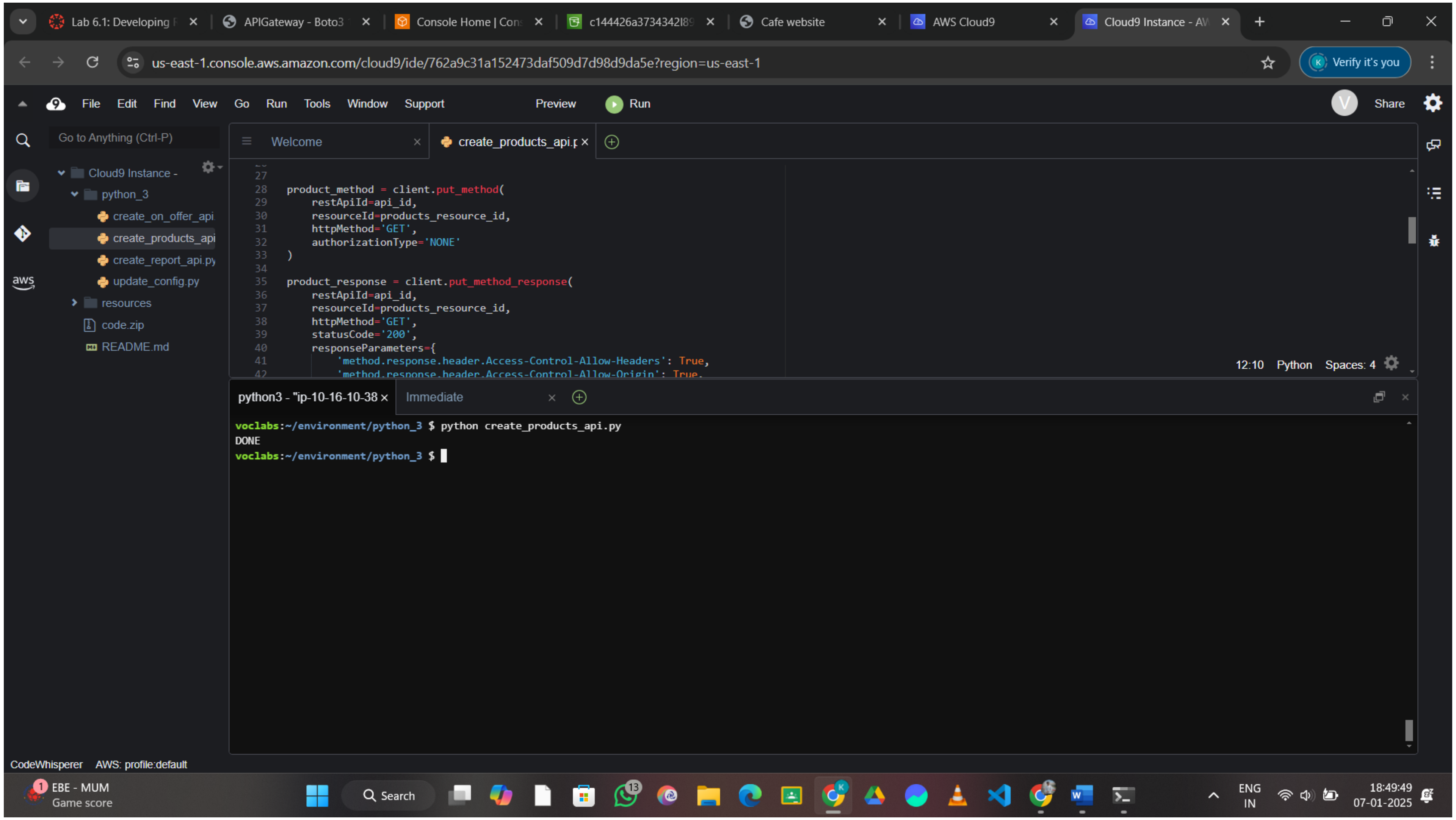
Amazon S3 > Buckets > c144426a3734342l8902453t1w636444766483-s3bucket-hmofgw7kygb6

Objects (11)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

	Name	Type	Last modified	Size	Storage class
☐	config.js	js	January 7, 2025, 18:37:24 (UTC+05:30)	89.0 B	Standard
☐	favicon.ico	ico	January 7, 2025, 18:37:24 (UTC+05:30)	1.1 KB	Standard
☐	images/	Folder	-	-	-
✓	index.html	html	January 7, 2025, 18:37:25 (UTC+05:30)	4.1 KB	Standard
☐	scripts/	Folder	-	-	-
☐	spint.md	md	January 7, 2025, 18:37:25 (UTC+05:30)	520.0 B	Standard
☐	styles/	Folder	-	-	-

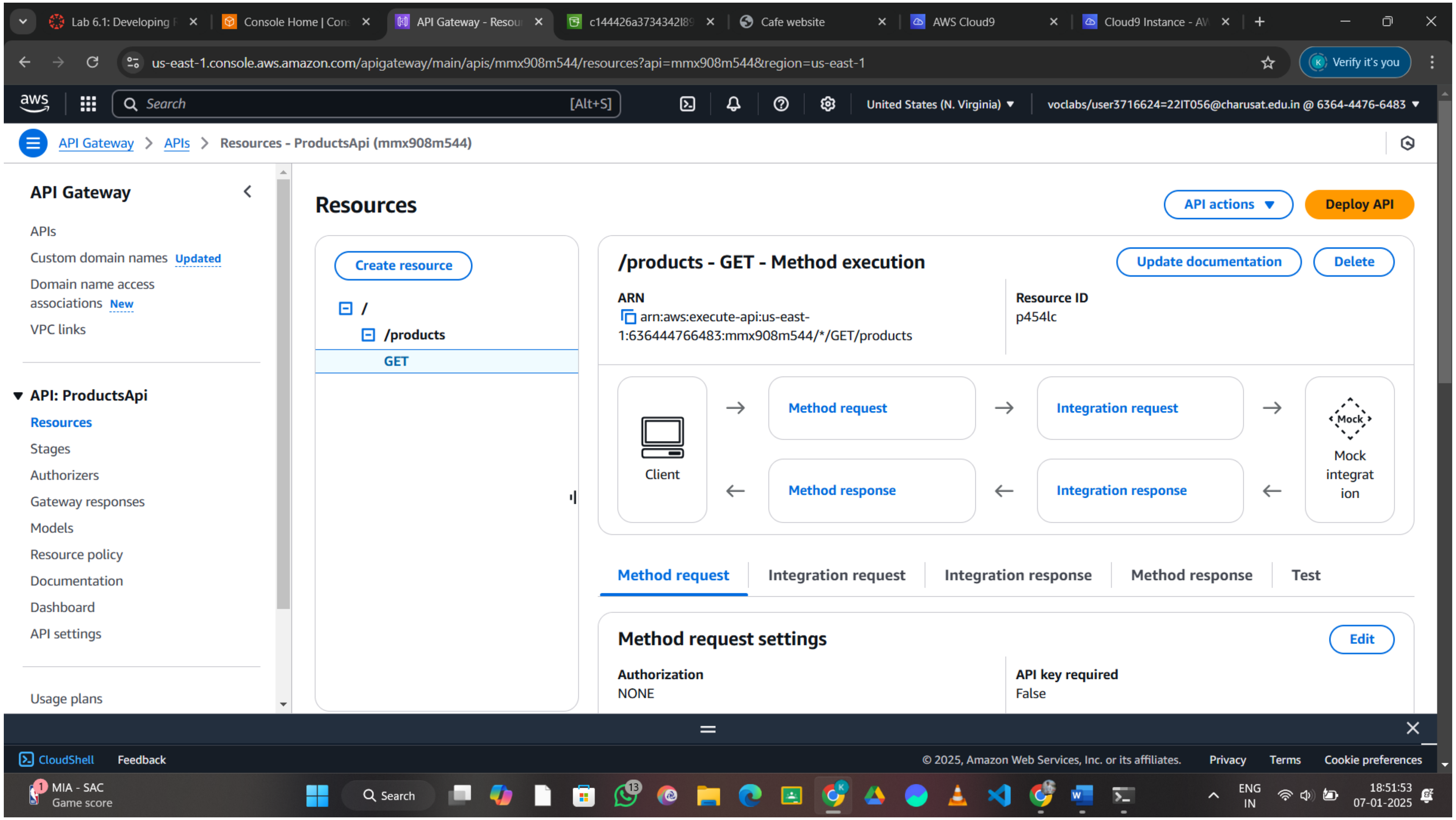
Figure 3: open s3 bucket and run a downloaded project



```python
product_method = client.put_method(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    authorizationType='NONE'
)

product_response = client.put_method_response(
    restApiId=api_id,
    resourceId=products_resource_id,
    httpMethod='GET',
    statusCode='200',
    responseParameters={
        'method.response.header.Access-Control-Allow-Headers': True,
        'method.response.header.Access-Control-Allow-Origin': True,
```

```
voclabs:~/environment/python_3 $ python create_products_api.py
DONE
voclabs:~/environment/python_3 $
```

Figure 4: Create a product api using python in the create_product_api file



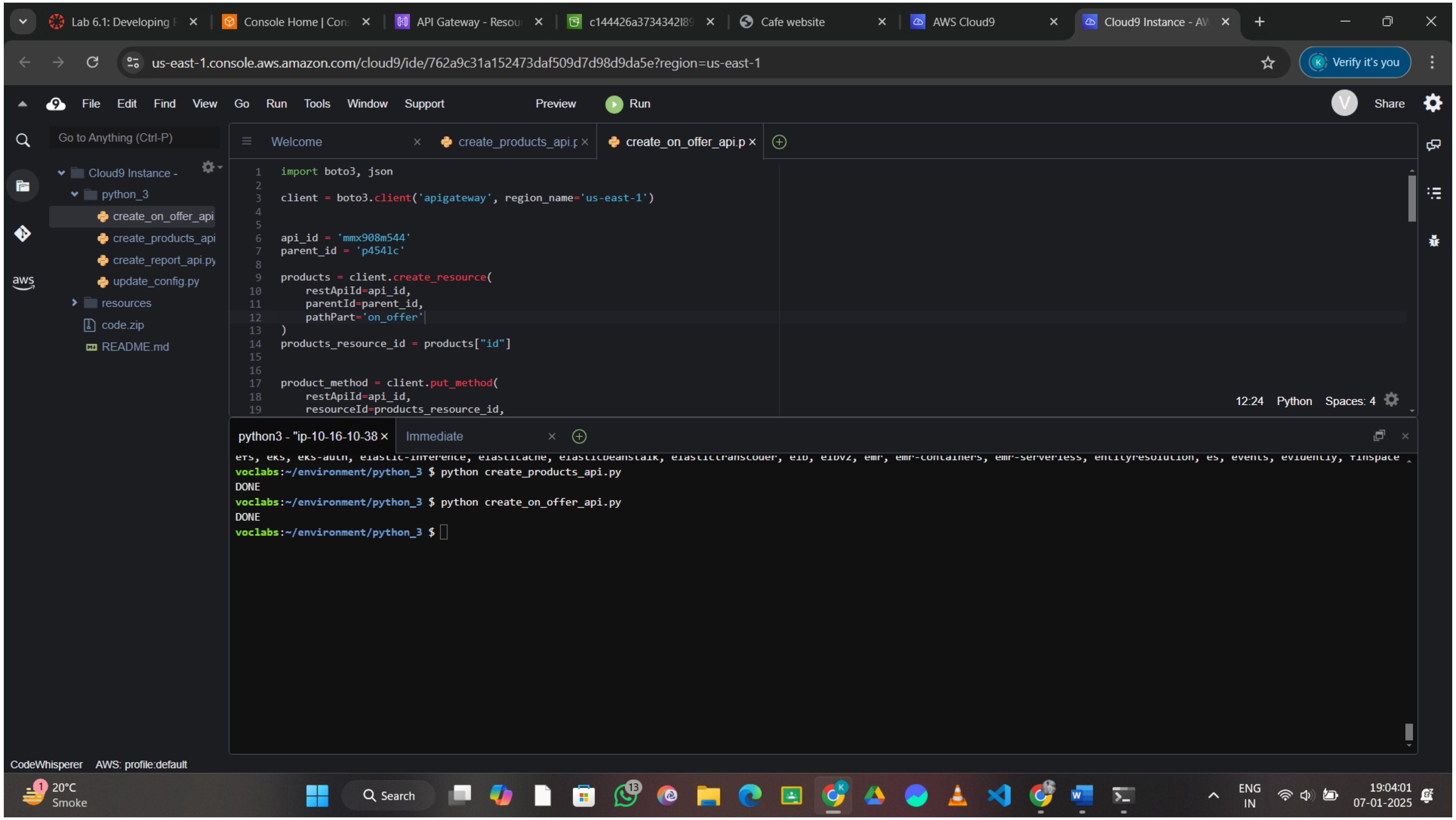
ARN
arn:aws:execute-api:us-east-1:636444766483:mmx908m544/*/GET/products

Resource ID
p454lc

Authorization
NONE

API key required
False

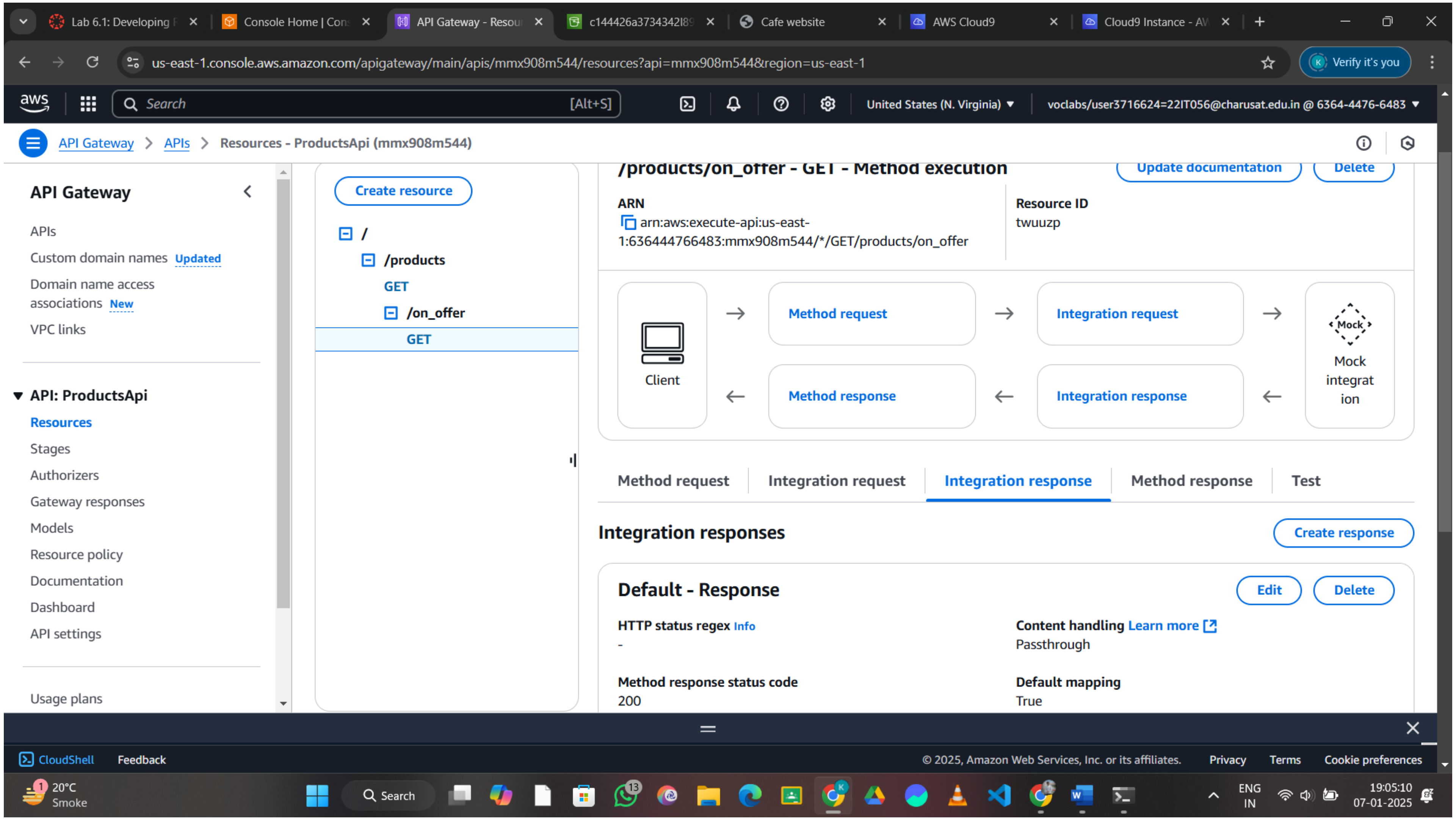
Figure 5: Check a api in api gateway and test it



```
import boto3, json

client = boto3.client('apigateway', region_name='us-east-1')


api_id = 'mmx908m544'
parent_id = 'p4541c'

products = client.create_resource(
    restApiId=api_id,
    parentId=parent_id,
    pathPart='on_offer'
)
products_resource_id = products["id"]


product_method = client.put_method(
    restApiId=api_id,
    resourceId=products_resource_id,
```

```
ers, eks, eks-auth, elastic-inference, elasticache, elasticbeanstalk, elastictranscoder, elb, elbv2, emr, emr-containers, emr-serverless, entityresolution, es, events, evidently, finspace
voclabs:~/environment/python_3 $ python create_products_api.py
DONE
voclabs:~/environment/python_3 $ python create_on_offer_api.py
DONE
voclabs:~/environment/python_3 $ 
```

Figure 6: Create offer api using python in the create_on_offer_api file



ARN
arn:aws:execute-api:us-east-1:636444766483:mmx908m544/*/GET/products/on_offer

Resource ID
twuuzp

Integration responses

Default - Response

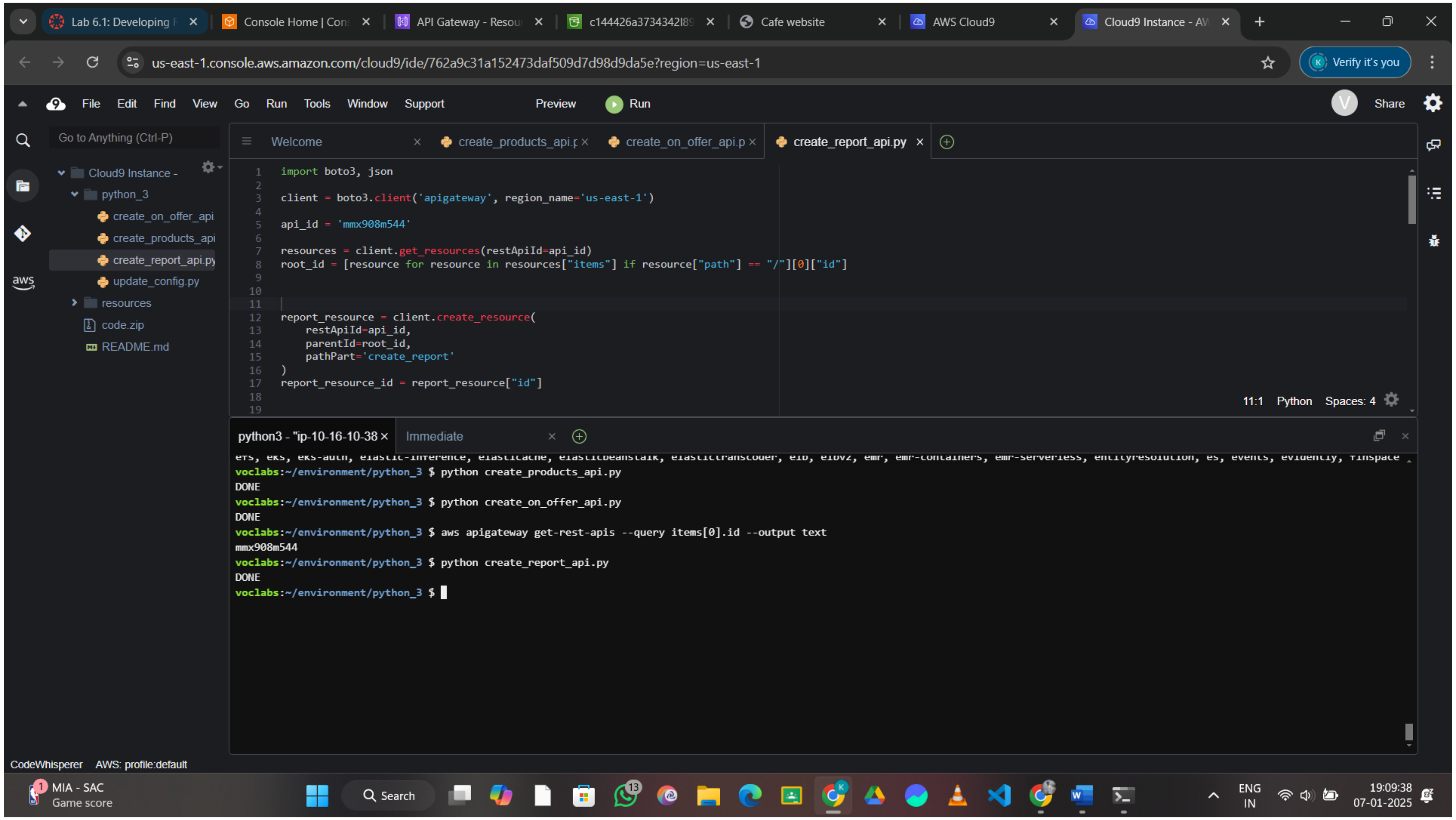
HTTP status regex
-

Content handling
Passthrough

Method response status code
200

Default mapping
True

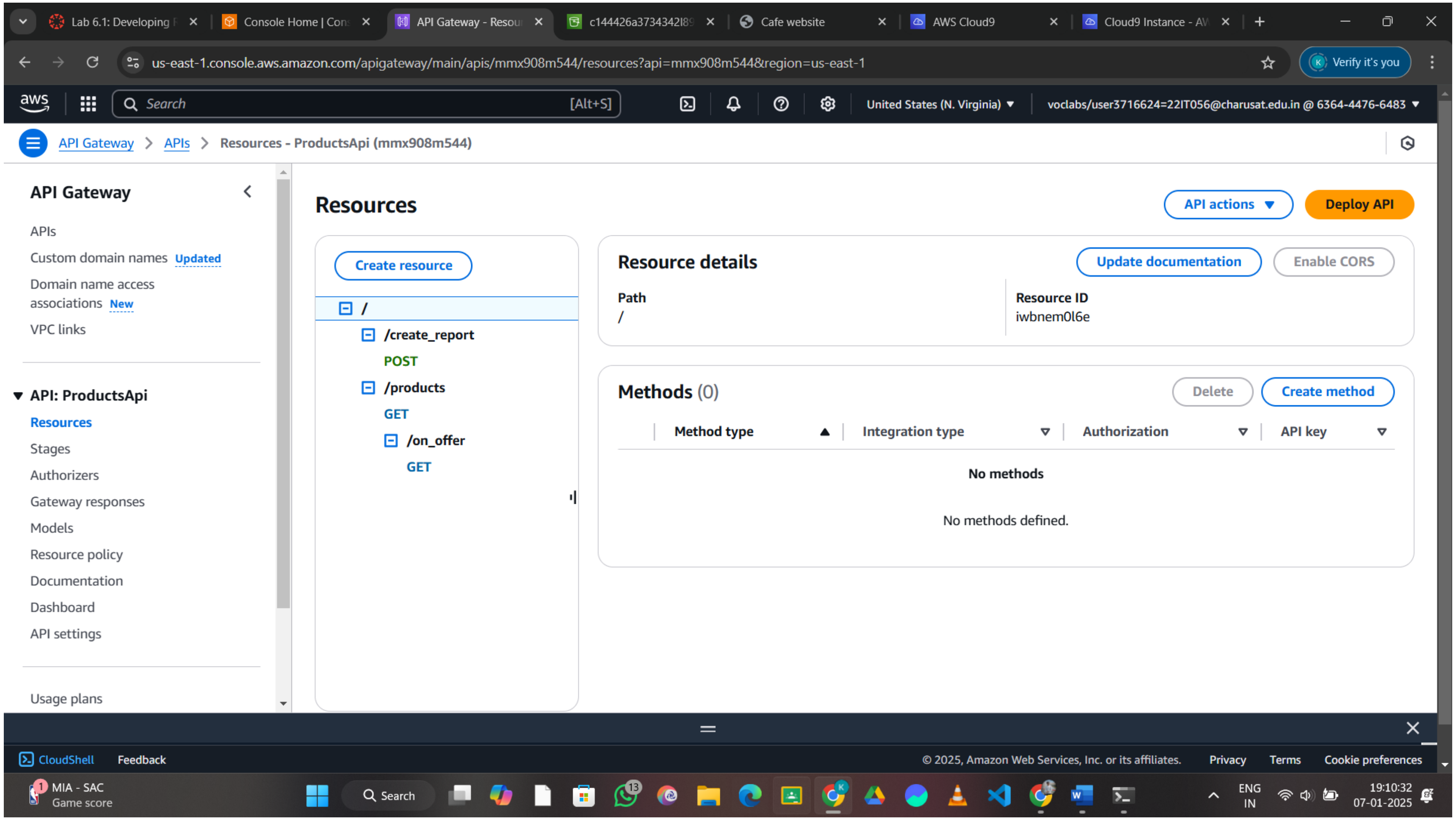
Figure 7: Check a api in api gateway and test it



```python
import boto3, json

client = boto3.client('apigateway', region_name='us-east-1')

api_id = 'mmx908m544'

resources = client.get_resources(restApiId=api_id)
root_id = [resource for resource in resources["items"] if resource["path"] == "/"][0]["id"]



report_resource = client.create_resource(
    restApiId=api_id,
    parentId=root_id,
    pathPart='create_report'
)
report_resource_id = report_resource["id"]
```

```
voclabs:~/environment/python_3 $ python create_products_api.py
DONE
voclabs:~/environment/python_3 $ python create_on_offer_api.py
DONE
voclabs:~/environment/python_3 $ aws apigateway get-rest-apis --query items[0].id --output text
mmx908m544
voclabs:~/environment/python_3 $ python create_report_api.py
DONE
voclabs:~/environment/python_3 $
```

Figure 8: Create report api using python in create_report_api file



us-east-1.console.aws.amazon.com/apigateway/main/apis/mmx908m544/resources?api=mmx908m544&region=us-east-1

API Gateway > APIs > Resources - ProductsApi (mmx908m544)

Resources

Resource details

Path
/

Resource ID
iwbnem0l6e

Resources tree:

```
/
  /create_report
    POST
  /products
    GET
    /on_offer
      GET
```

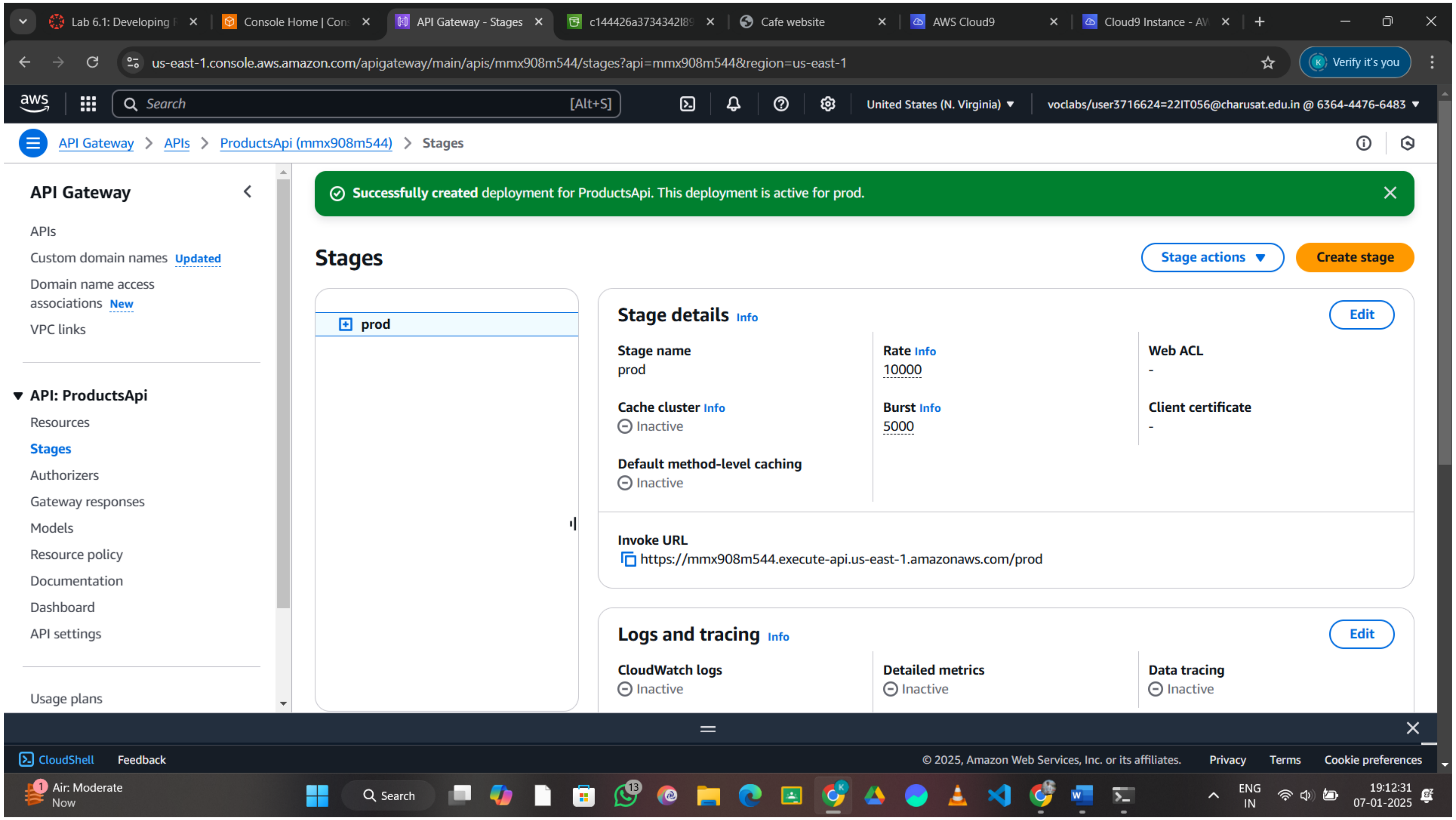
Methods (0)

Method type	Integration type	Authorization	API key

No methods

No methods defined.

Figure 9: Check a api in api gateway and test it


Successfully created deployment for ProductsApi. This deployment is active for prod.

Stages

Stage details

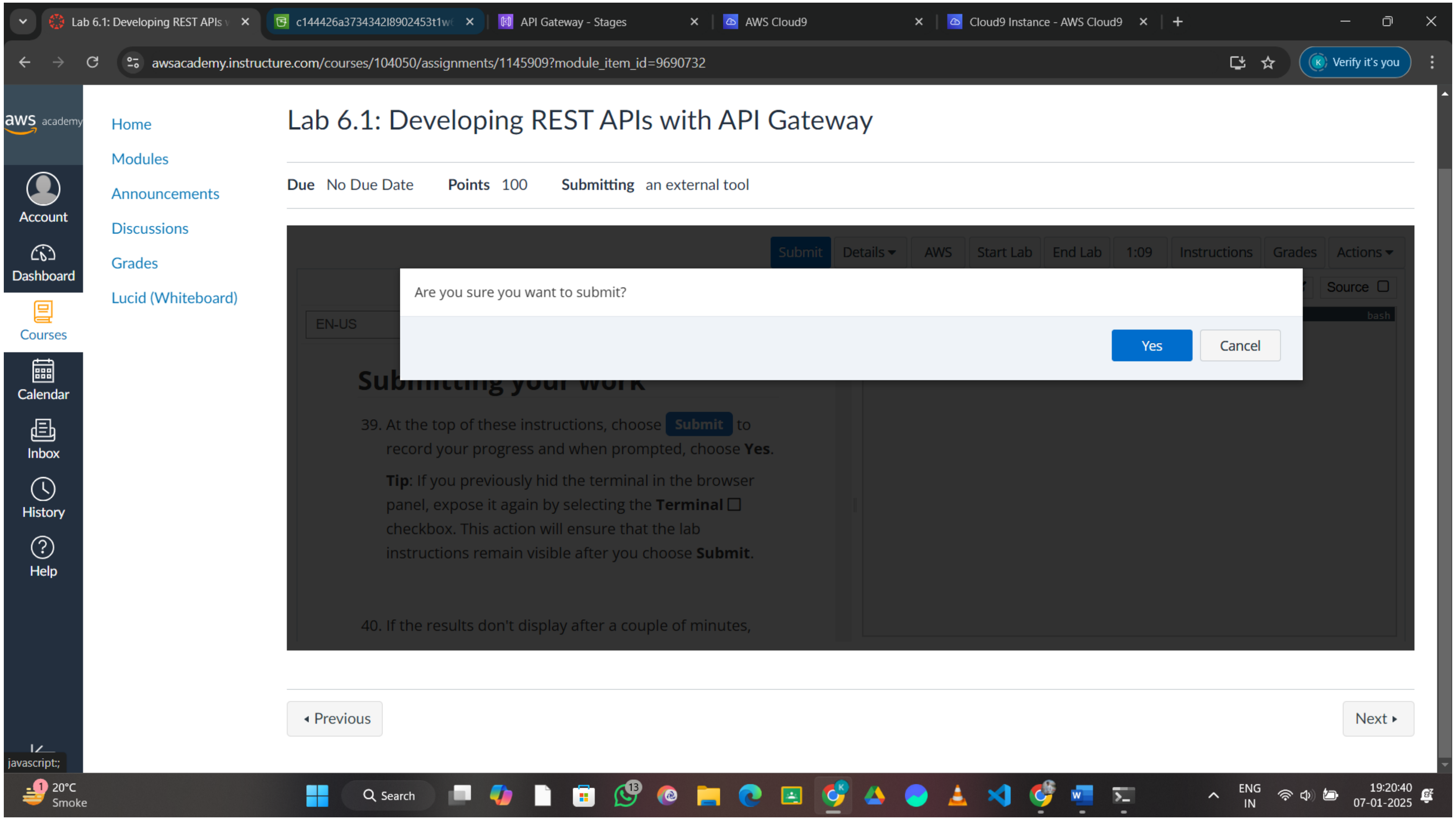
Stage name: prod
Rate: 10000
Web ACL: -
Cache cluster: Inactive
Burst: 5000
Client certificate: -
Default method-level caching: Inactive

Invoke URL: https://mmx908m544.execute-api.us-east-1.amazonaws.com/prod

Logs and tracing

CloudWatch logs: Inactive
Detailed metrics: Inactive
Data tracing: Inactive

Figure 10: Deployed our created api



Lab 6.1: Developing REST APIs with API Gateway

Due No Due Date Points 100 Submitting an external tool

Are you sure you want to submit?

39. At the top of these instructions, choose Submit to record your progress and when prompted, choose Yes.

Tip: If you previously hid the terminal in the browser panel, expose it again by selecting the Terminal checkbox. This action will ensure that the lab instructions remain visible after you choose Submit.

40. If the results don't display after a couple of minutes,

Figure 11: Submit lab

LATEST APPLICATIONS:

1. Microservices Architecture
2. Serverless Applications
3. IoT Backend Integration
4. Mobile and Web Application Backends
5. Data and Analytics Pipelines
6. Chat Applications
7. E-Commerce Platforms
8. AI and ML Model Deployment
9. Content Management and Delivery
10. Gaming Applications

LEARNING OUTCOME:

By completing this lab, you will gain a solid understanding of **Amazon API Gateway** and its role in creating RESTful APIs, including mock endpoints for simulating backend functionality. You will learn how to enable **CORS** for secure cross-origin communication, integrate mock APIs with web applications, and prepare for transitioning to real database integrations such as DynamoDB. This experience will enhance your skills in designing scalable, secure, and efficient API architectures, while also exposing you to modern use cases like serverless applications, IoT backends, and real-time communication. Ultimately, you will develop a foundational understanding of API development workflows, preparing you for building robust, production-ready APIs.

REFERENCE:

1. https://awsacademy.instructure.com/courses/104050