AWS Cloud Resume Challenge:

Note: The HTML, CSS, and JavaScript code can be found in the GitHub repository linked below. Please replace the placeholder images with your own images.

For instructions on hosting your portfolio website using S3 and CloudFront, and configuring it with your custom domain, check out my video linked below.

Github Repository: https://github.com/suzal777/PortfolioWebsite.git

Video Link: https://youtu.be/ddrTULjH_bg

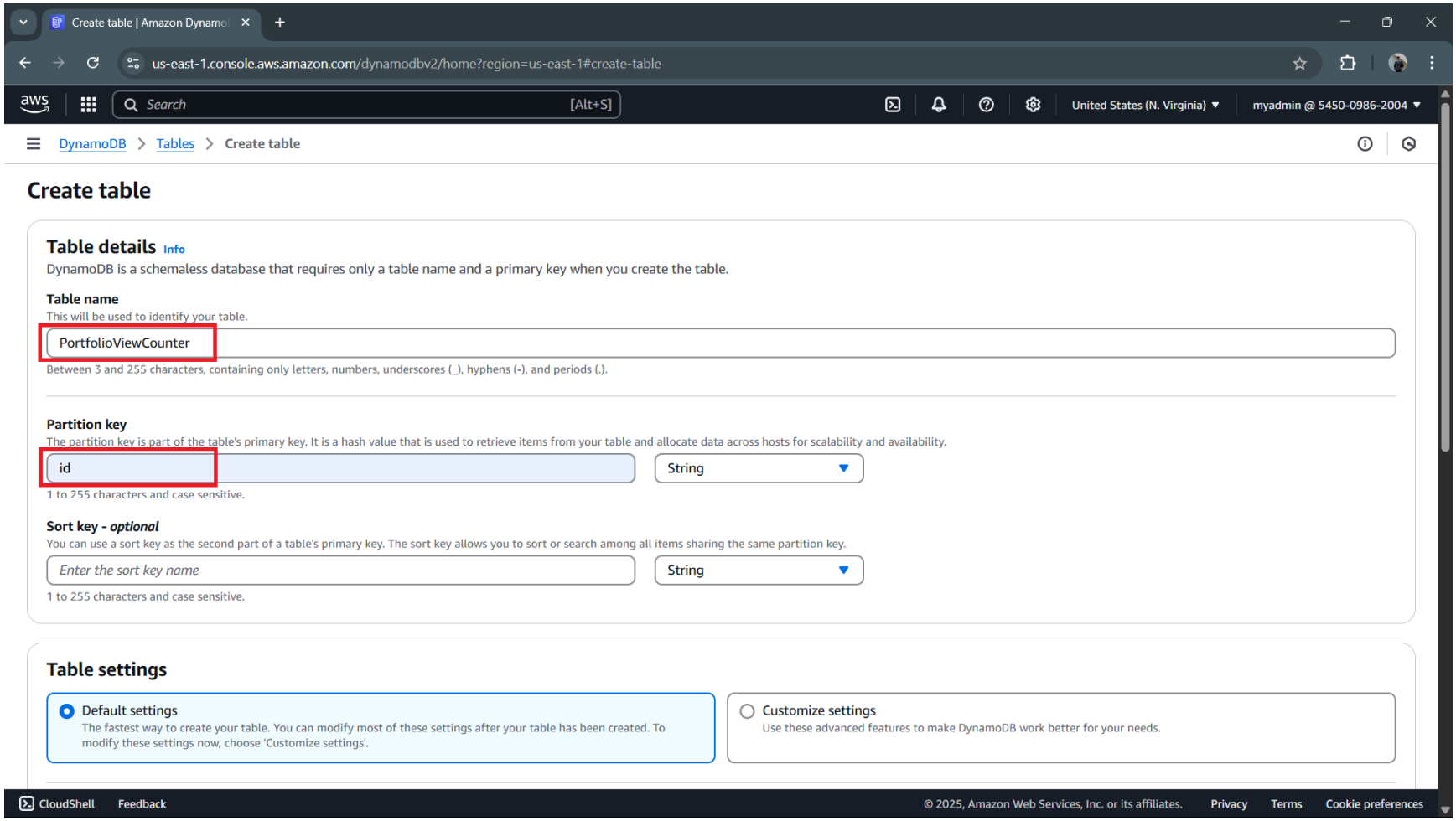
The below tutorials are for adding Views Count feature and Contact Me feature in the Portfolio Website.

All the best!

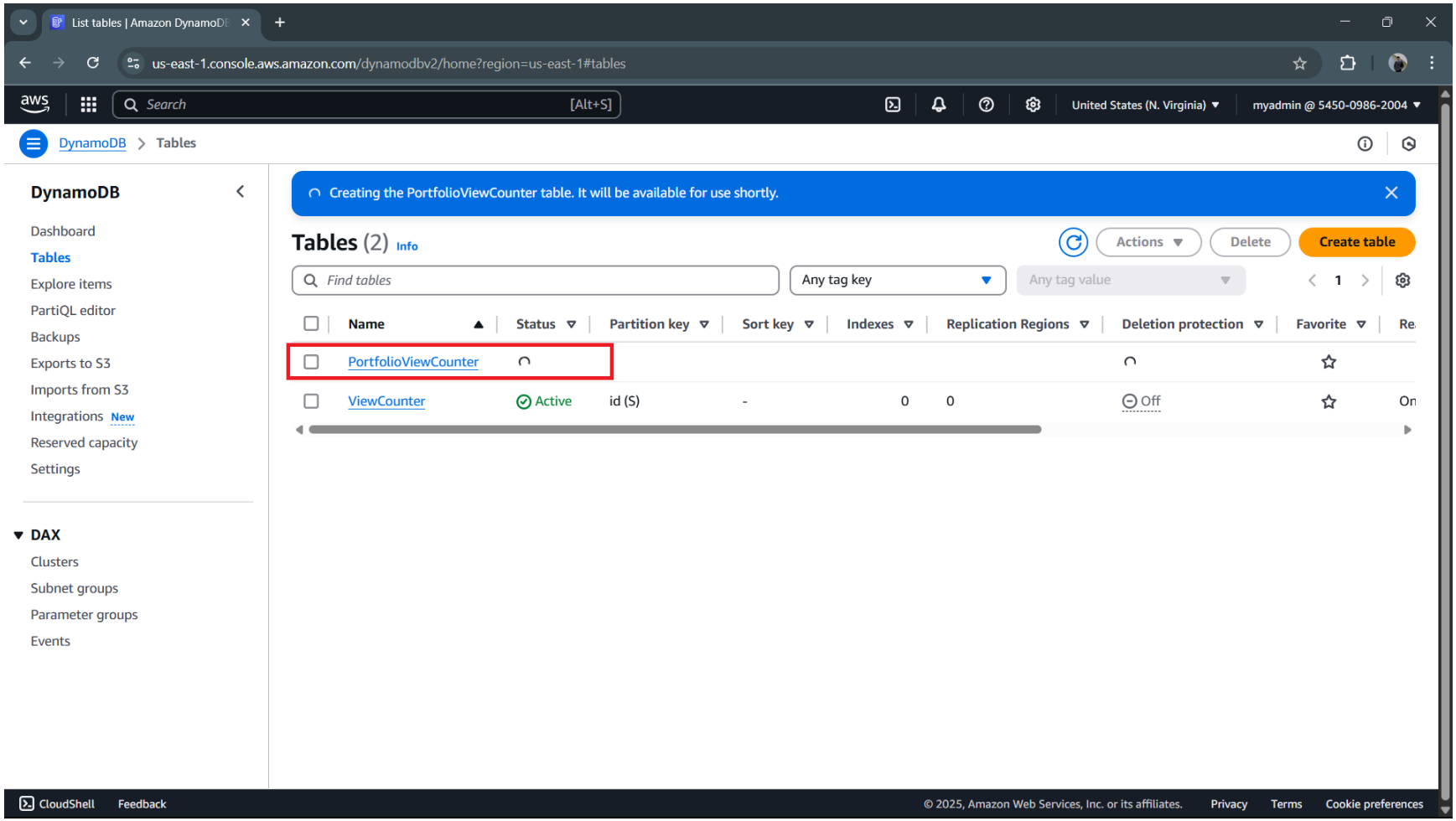
Views Count feature

Creating DynamoDB Table:

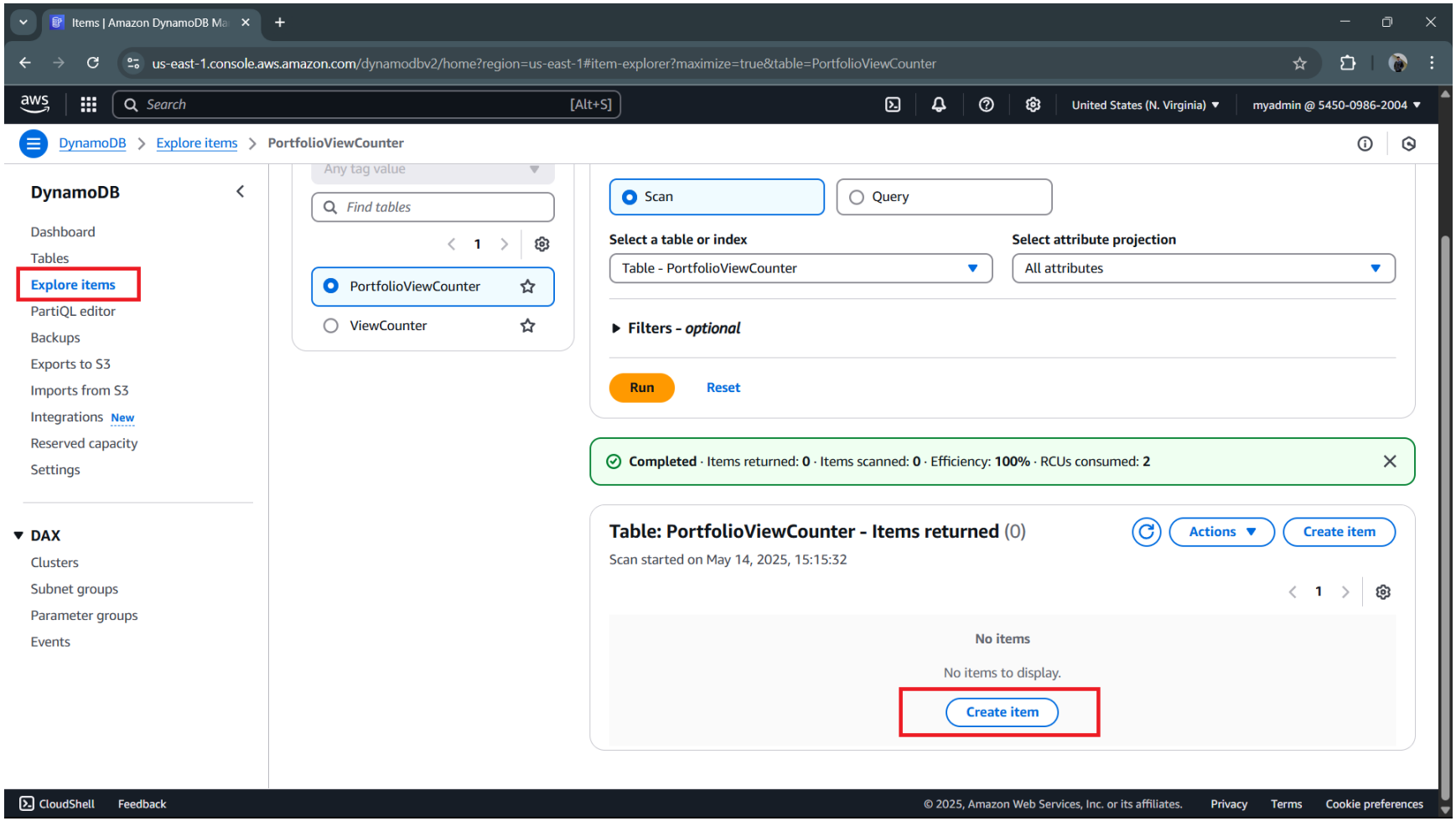
Create a DynamoDB table named "PortfolioViewCounter", with "id" defined as the partition key.



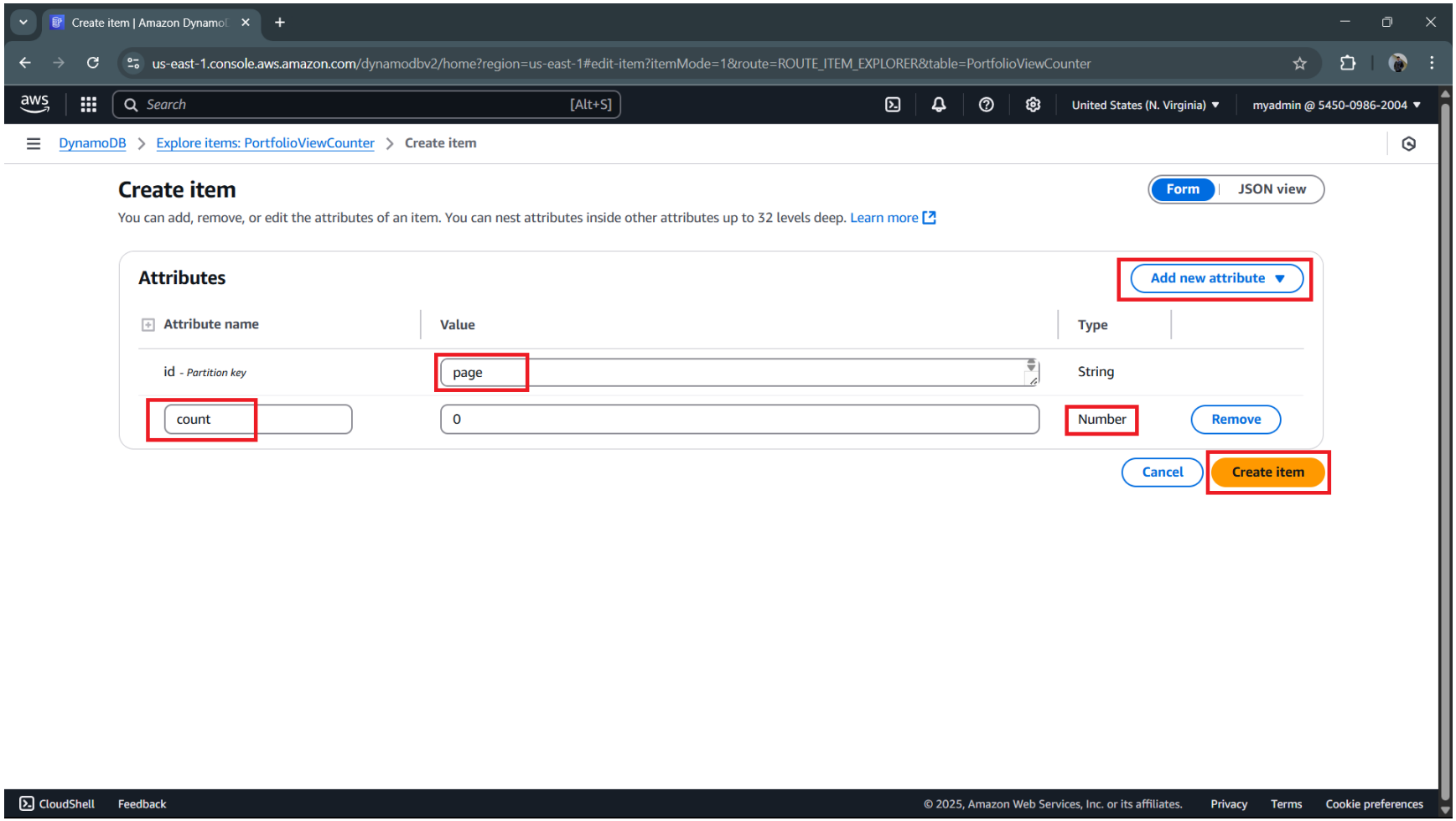
Click on the "PortfolioViewCounter" table.



Navigate to Explore Items and click on Create Item to add a new entry.

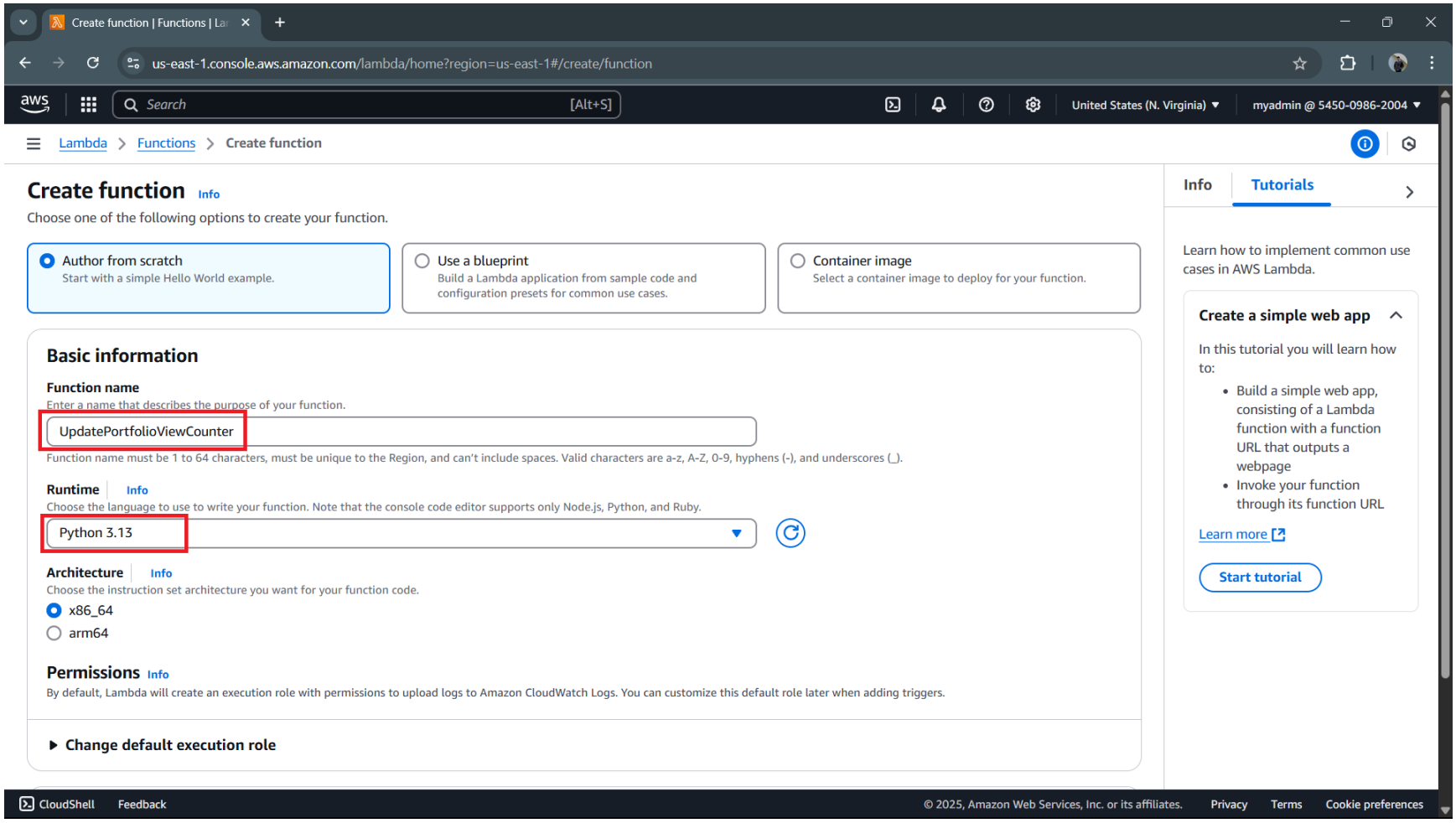


Enter "page" as the value for the partition key "id", then click on Add New Attribute. Set the attribute type to Number, name it "count", and leave the value as 0. Finally, click on Create Item to save the changes.

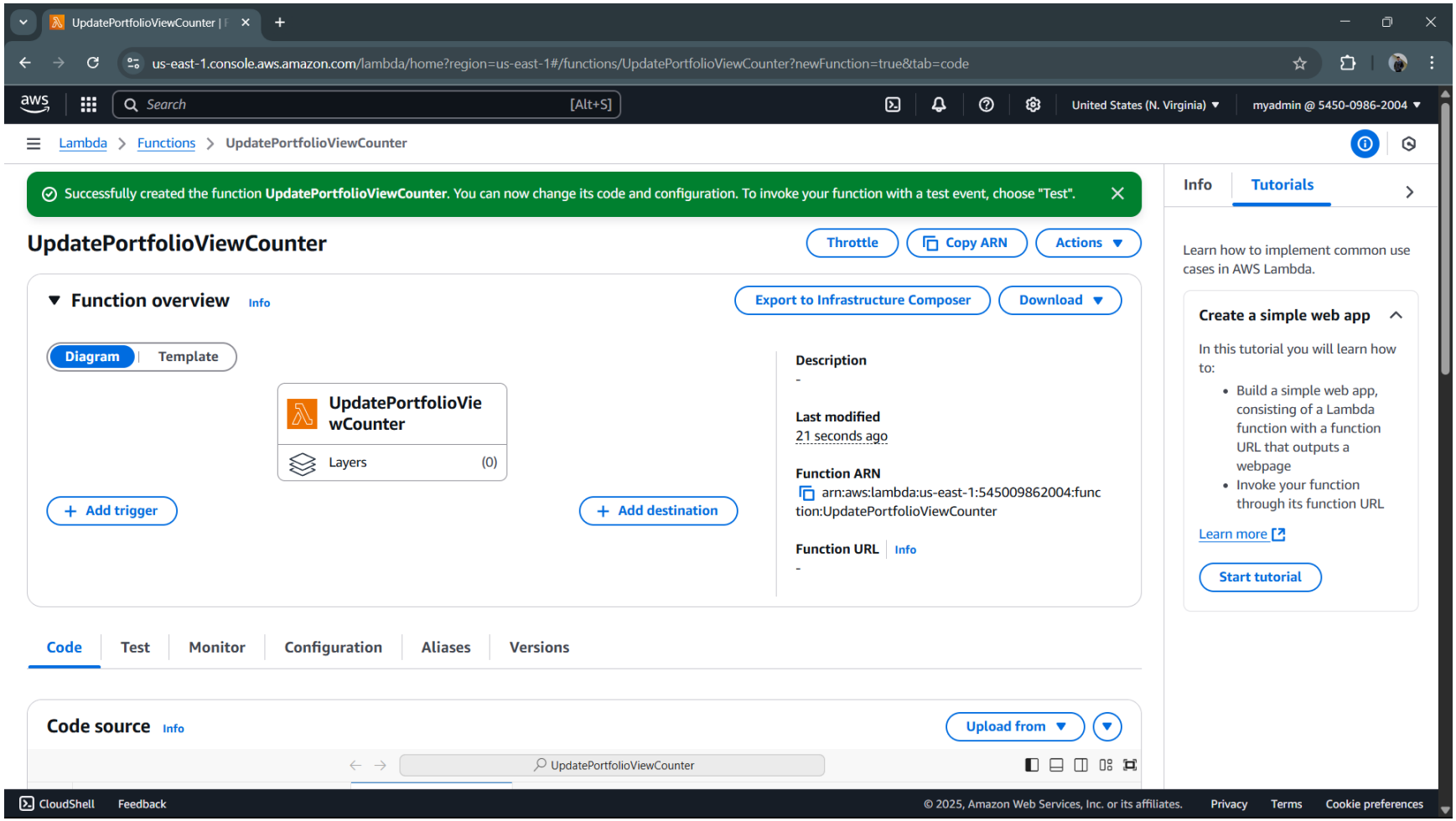


Creating Lambda Function:

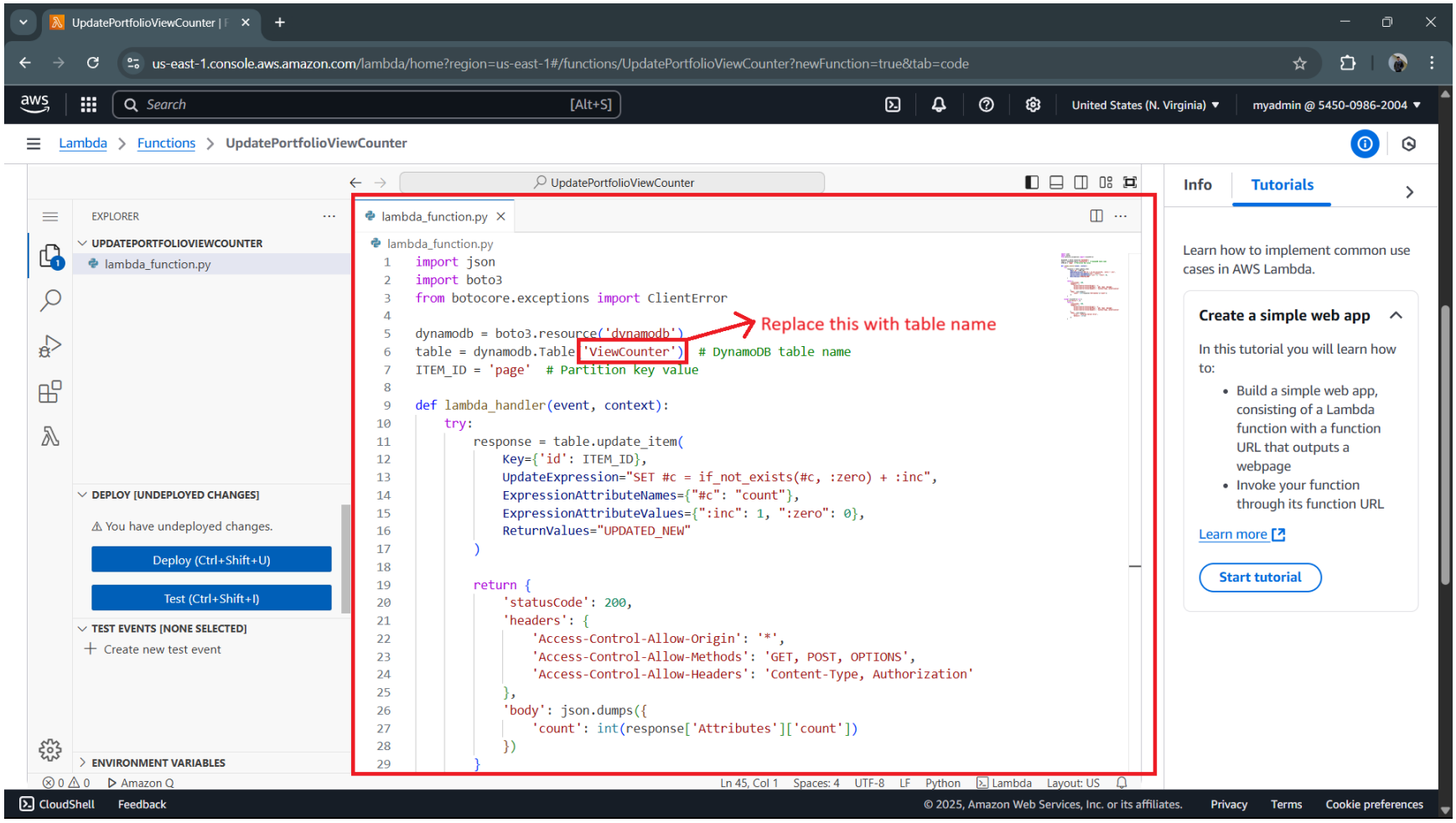
Create a Lambda function named "UpdatePortfolioViewCounter" with the runtime set to Python 3.13, then click on Create Function to finalize the creation.



The Lambda function is created.



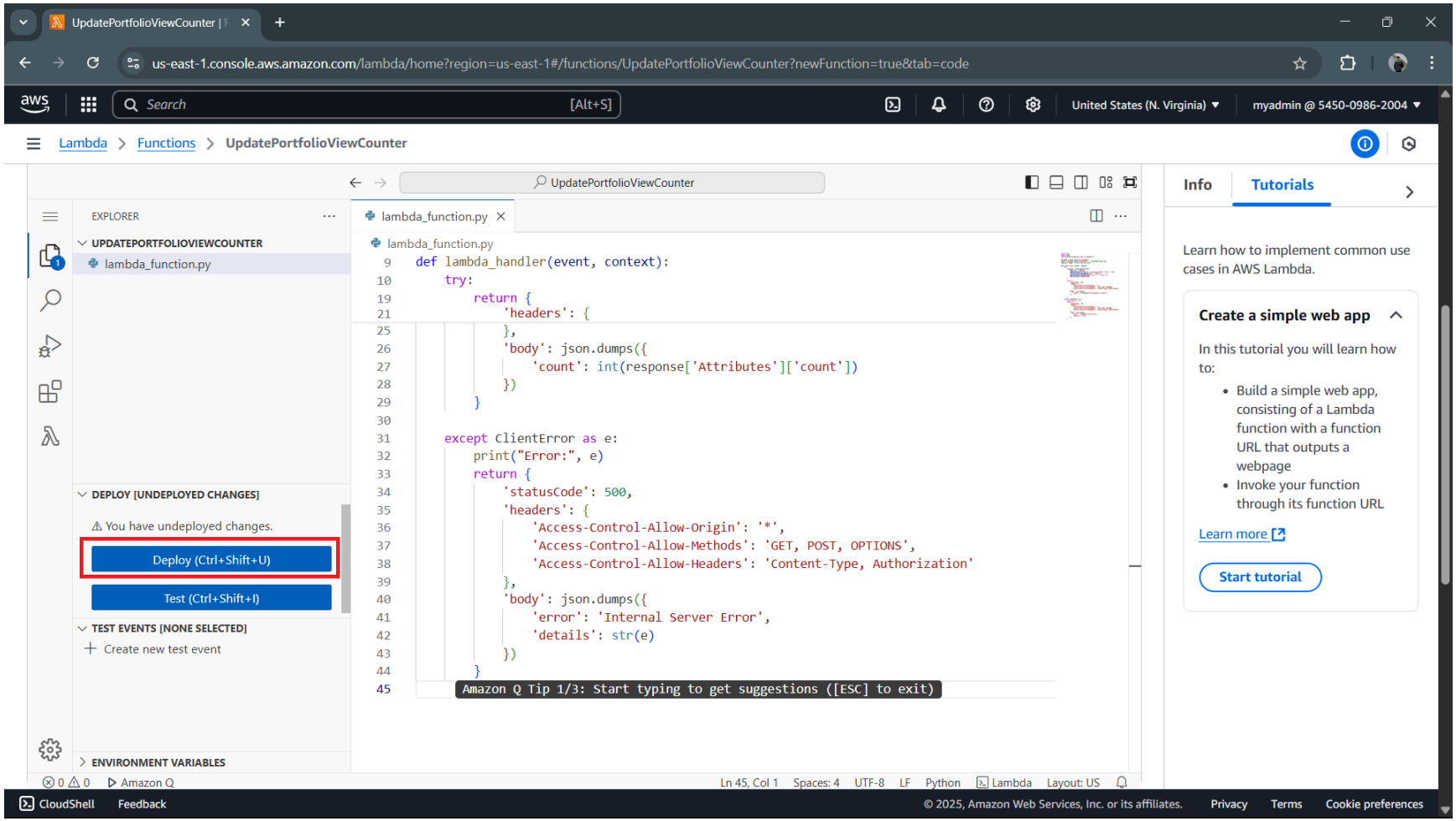
Next, navigate to the Code section and replace the existing code with the code provided below. Note: The code below has already replaced the table name as "PortfolioViewCounter".



```
import json
import boto3
from botocore.exceptions import ClientError

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('PortfolioViewCounter')  # DynamoDB table name
ITEM_ID = 'page'  # Partition key value

def lambda_handler(event, context):
    try:
        response = table.update_item(
            Key={'id': ITEM_ID},
            UpdateExpression="SET #c = if_not_exists(#c, :zero) + :inc",
            ExpressionAttributeNames={"#c": "count"},
            ExpressionAttributeValues={":inc": 1, ":zero": 0},
            ReturnValues="UPDATED_NEW"
        )
        return {
            'statusCode': 200,
            'headers': {
                'Access-Control-Allow-Origin': '*',
```

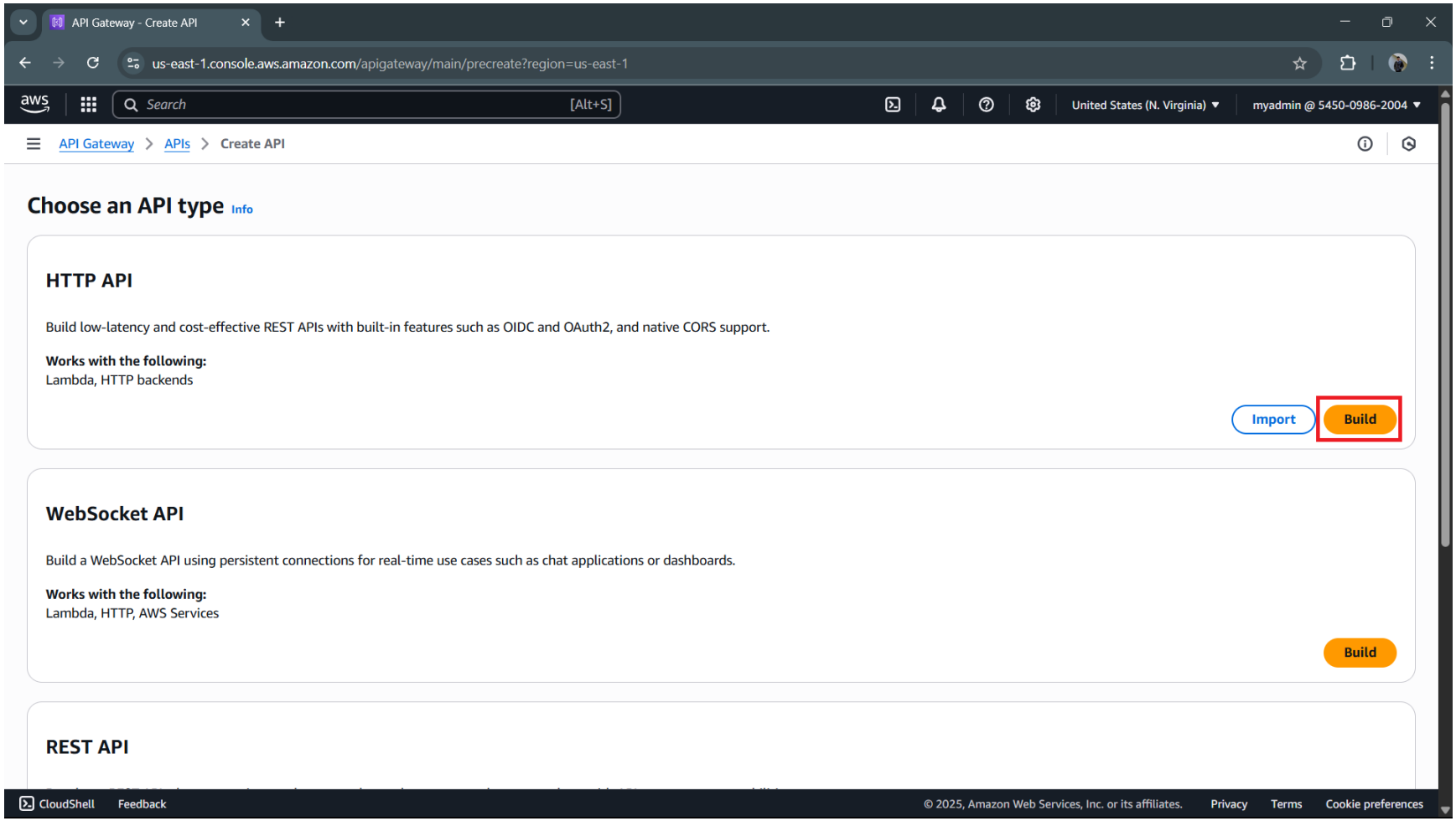
```
            'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
            'Access-Control-Allow-Headers': 'Content-Type, Authorization'
        },
        'body': json.dumps({
            'count': int(response['Attributes']['count'])
        })
    }
except ClientError as e:
    print("Error:", e)
    return {
        'statusCode': 500,
        'headers': {
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
            'Access-Control-Allow-Headers': 'Content-Type, Authorization'
        },
        'body': json.dumps({
            'error': 'Internal Server Error',
            'details': str(e)
        })
    }
```

After pasting the code, click on Deploy on the left to save and apply the changes.

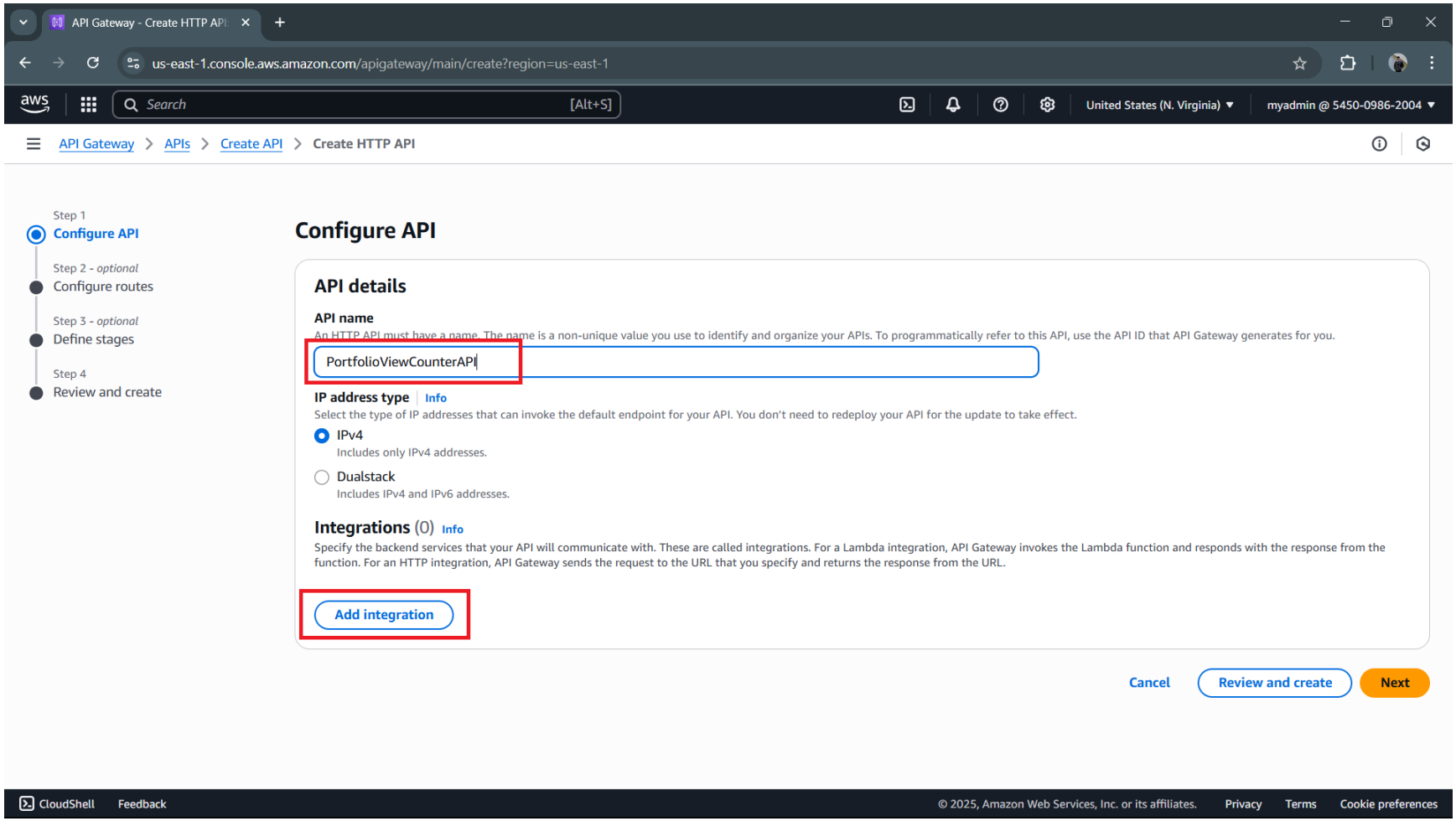


Creating API Gateway:

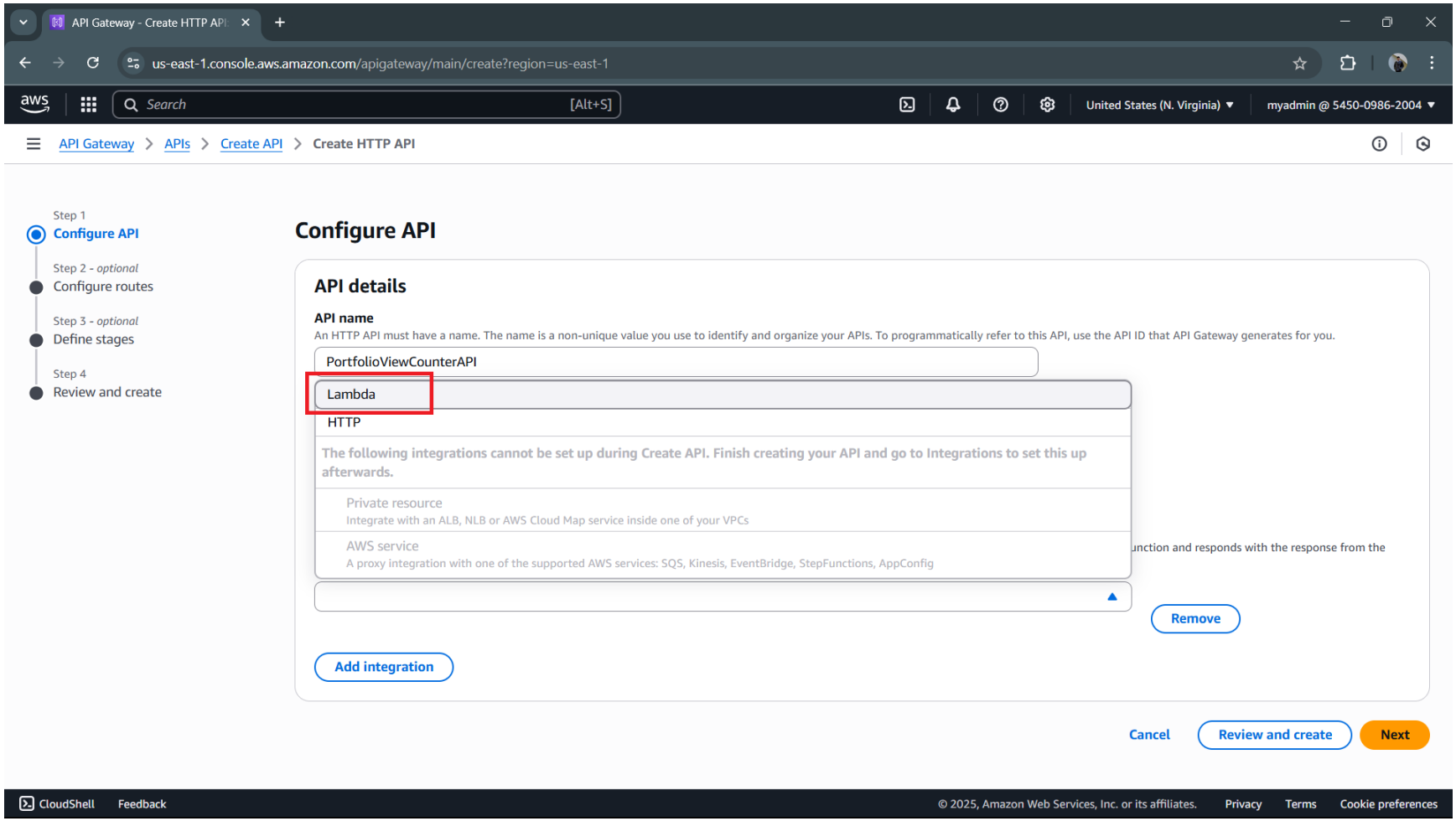
Now, create an API Gateway by selecting HTTP API, then click on Build to start the configuration.



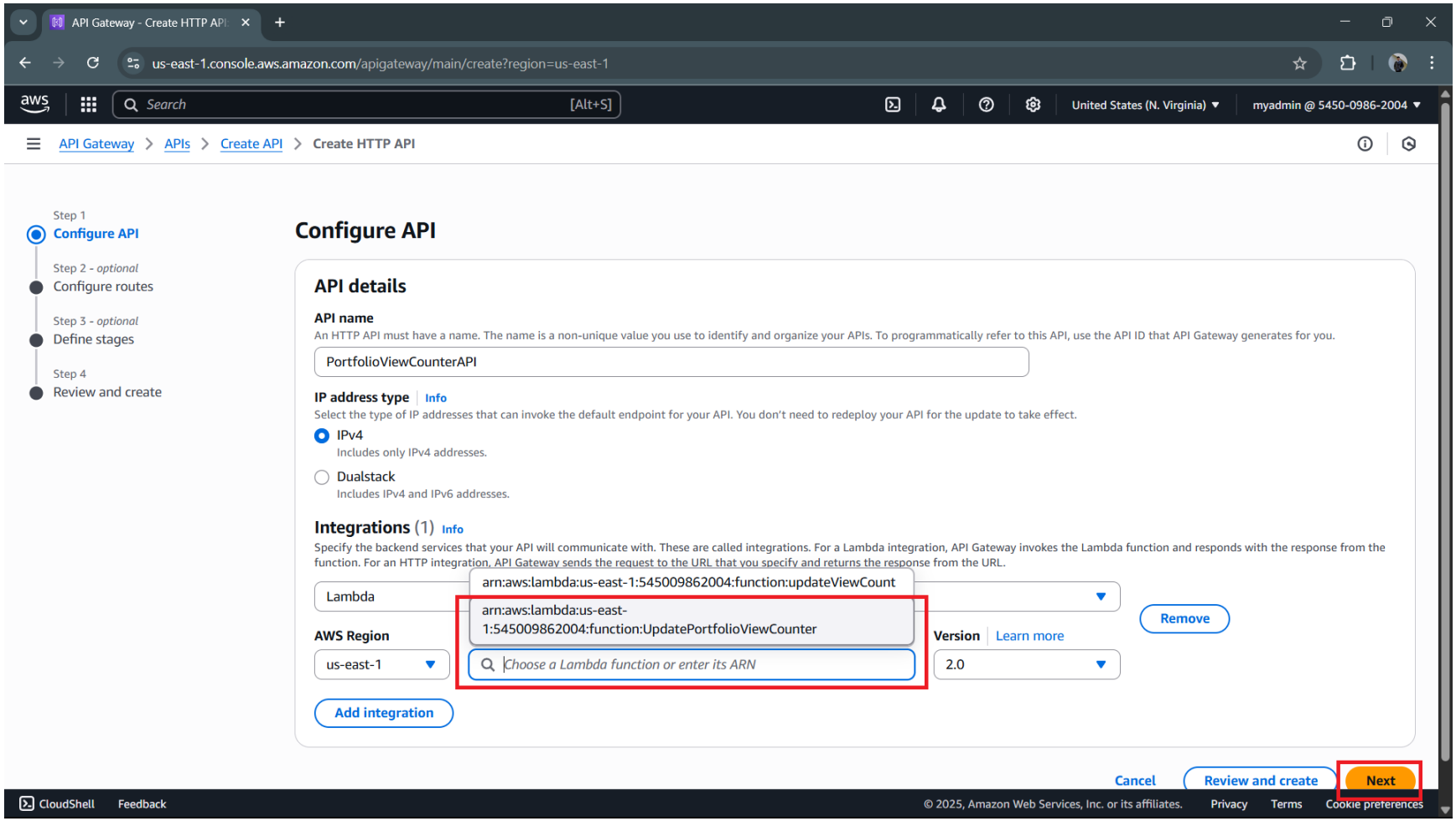
Enter "PortfolioViewCounterAPI" as the API name, then click on Add Integration to proceed.



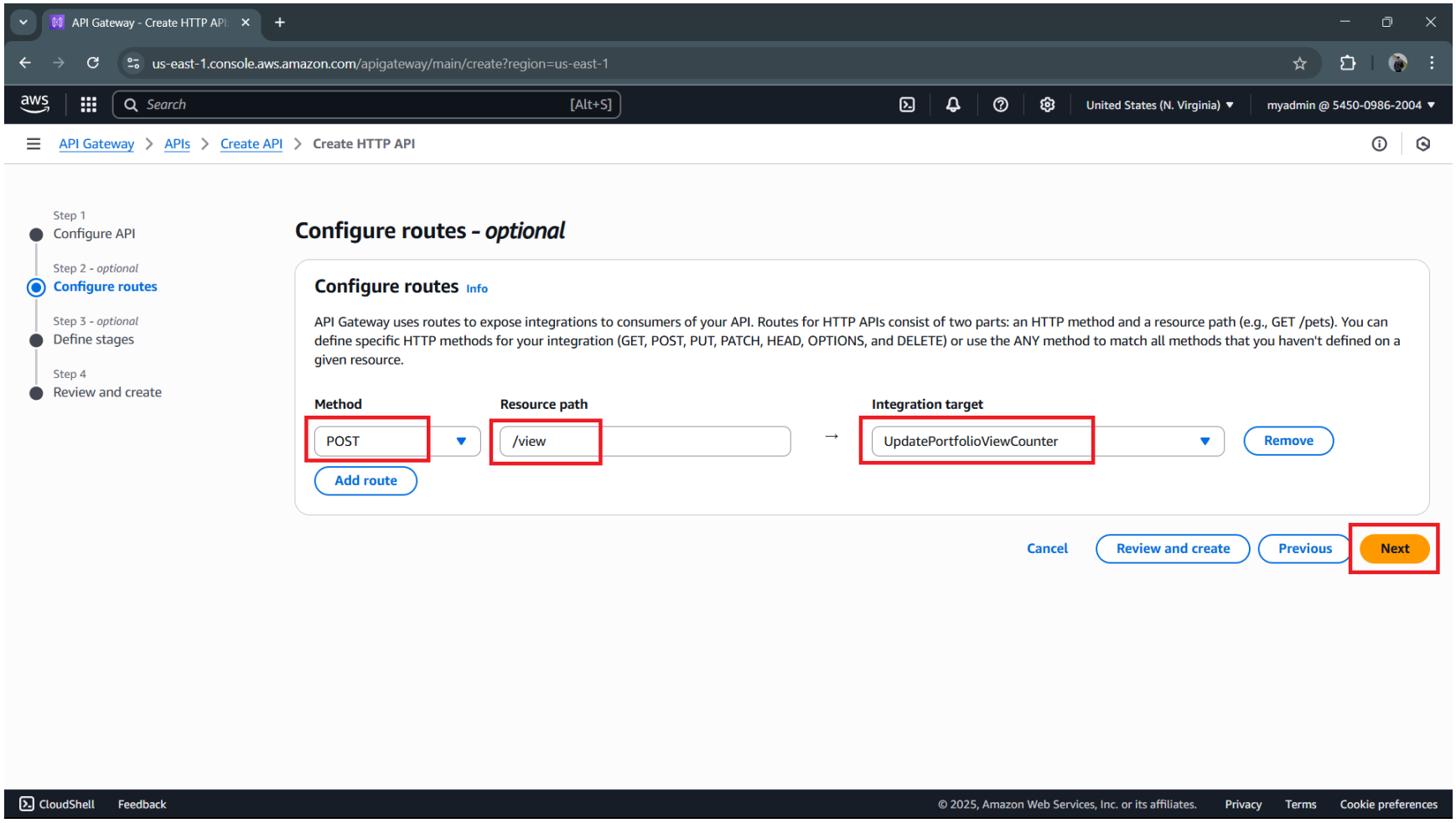
Then, select Lambda as the integration type.



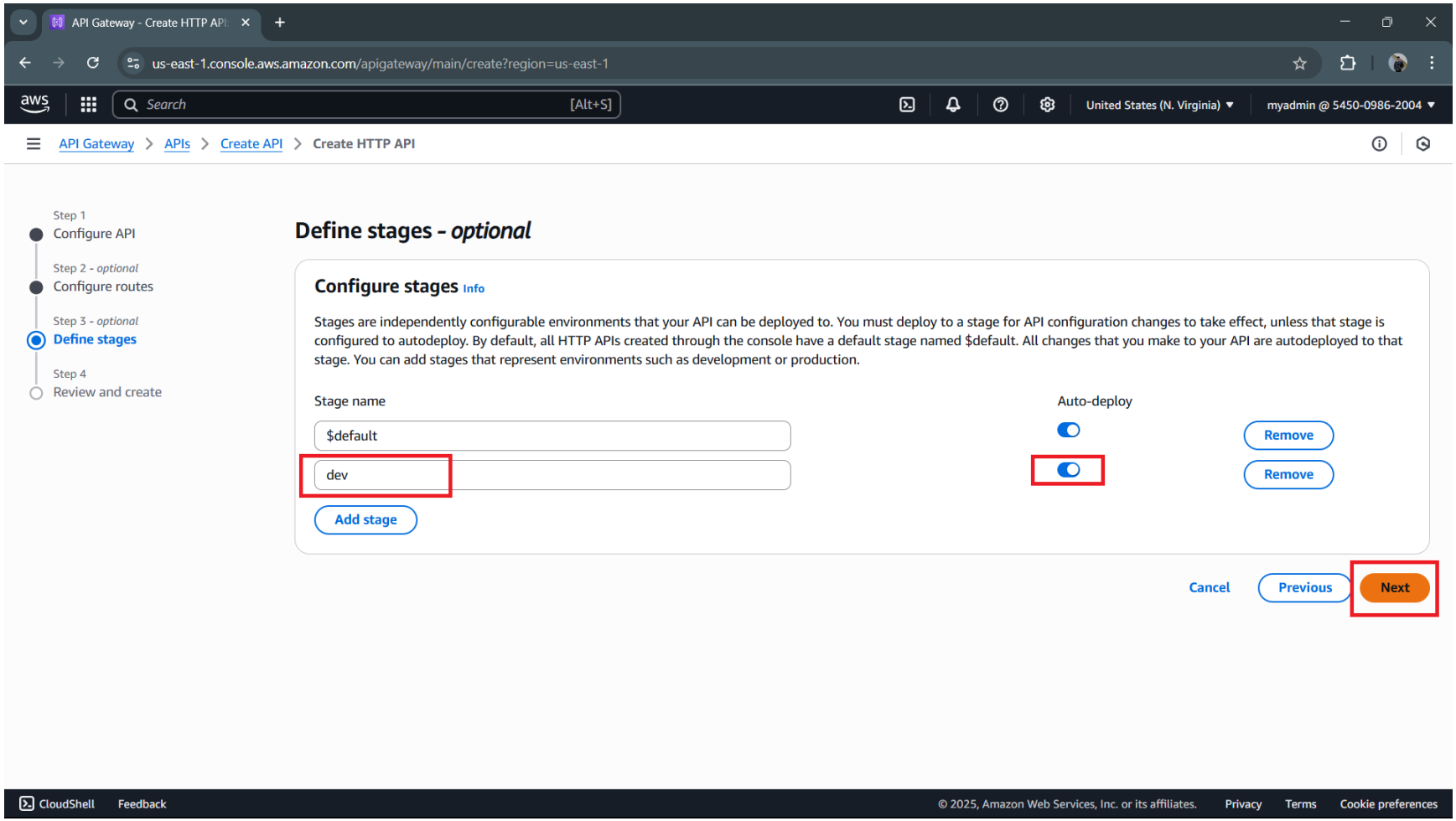
Select the "UpdatePortfolioViewCounter" Lambda function from the list and Click on Next.



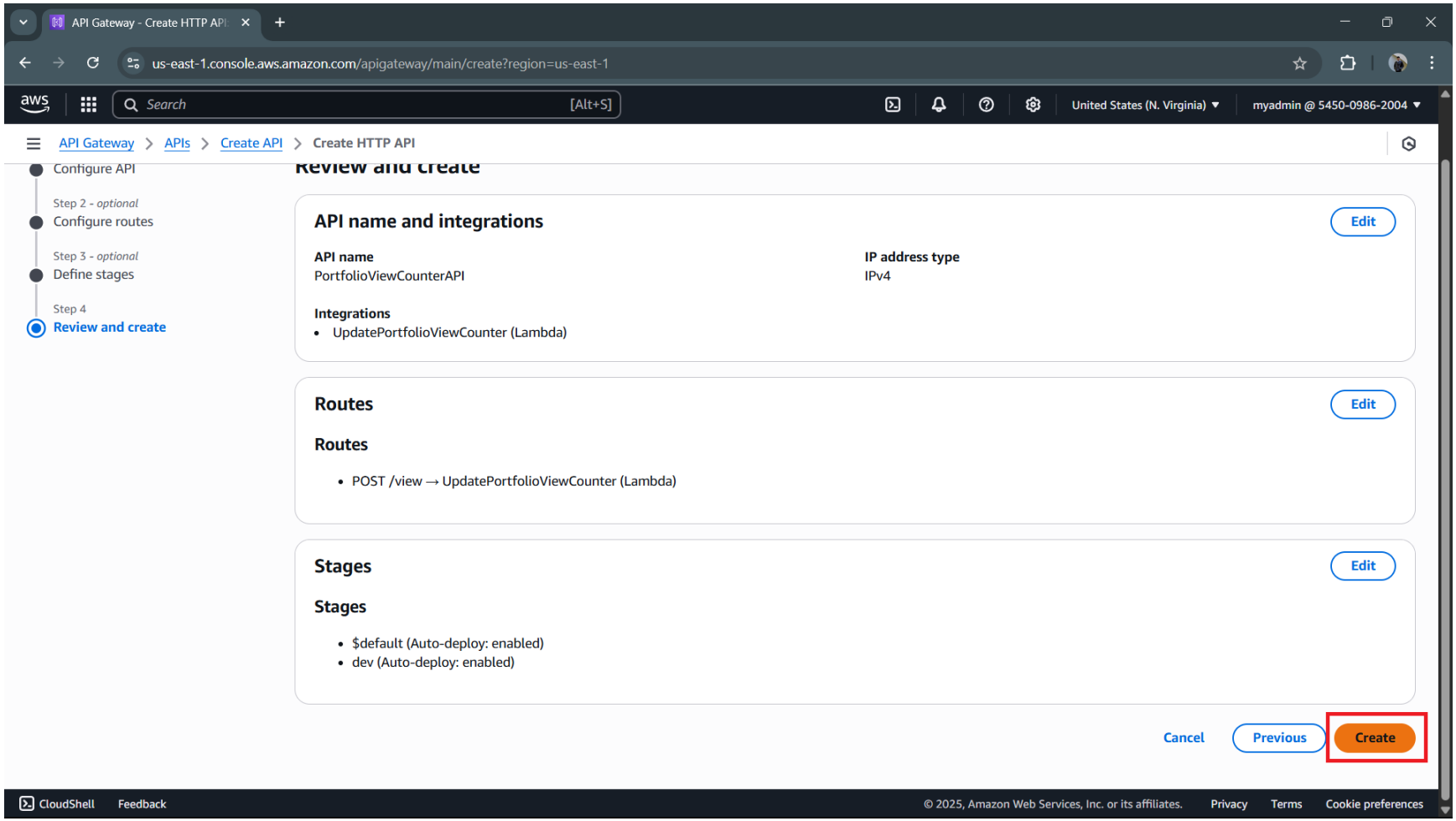
Set the method to POST, enter /view as the resource path, choose "UpdatePortfolioViewCounter" as the integration target, and then click on Next to continue.



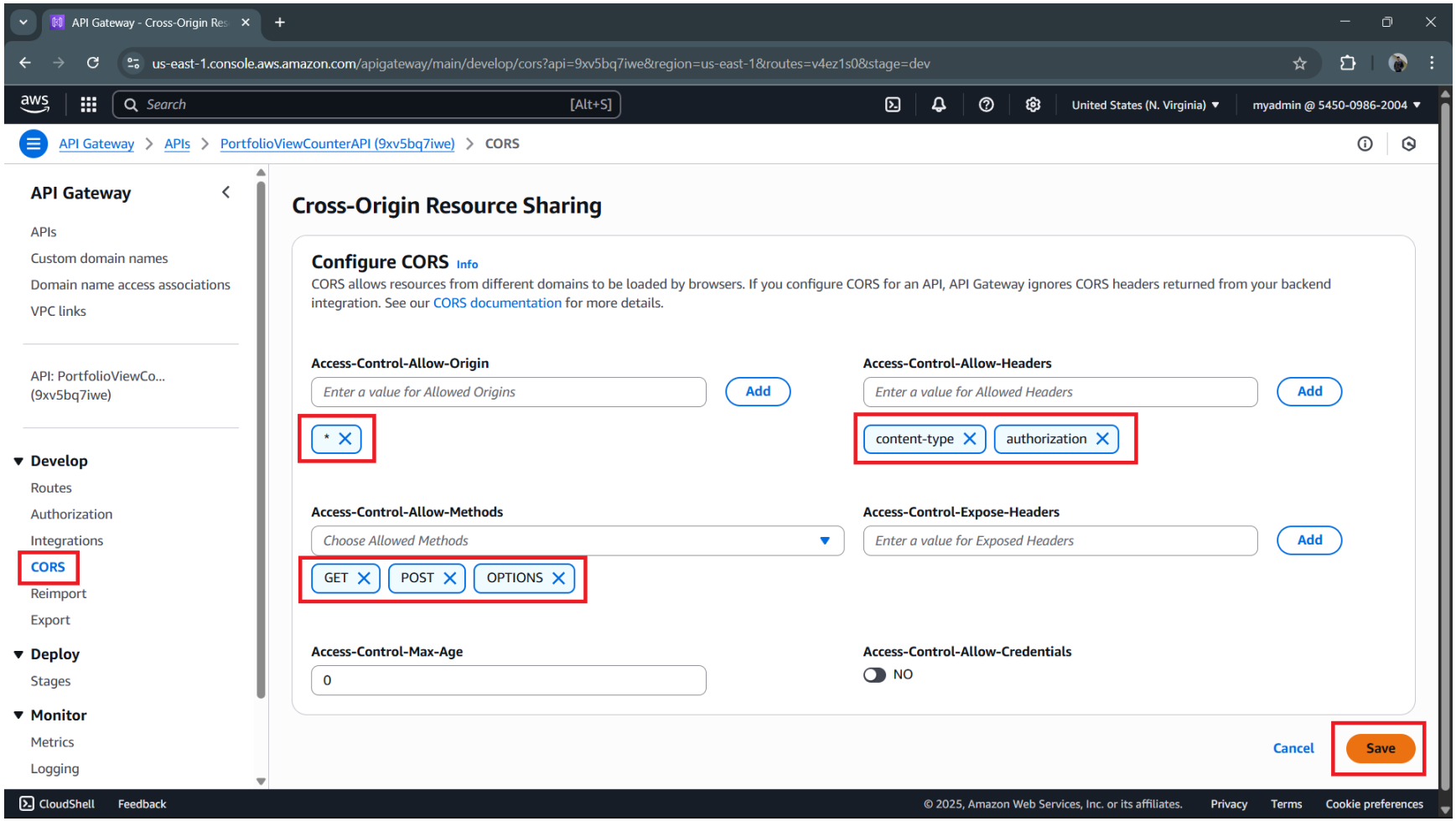
Click on Add Stage, enter "dev" as the stage name, enable the Auto deploy toggle, and then click on Next to proceed.



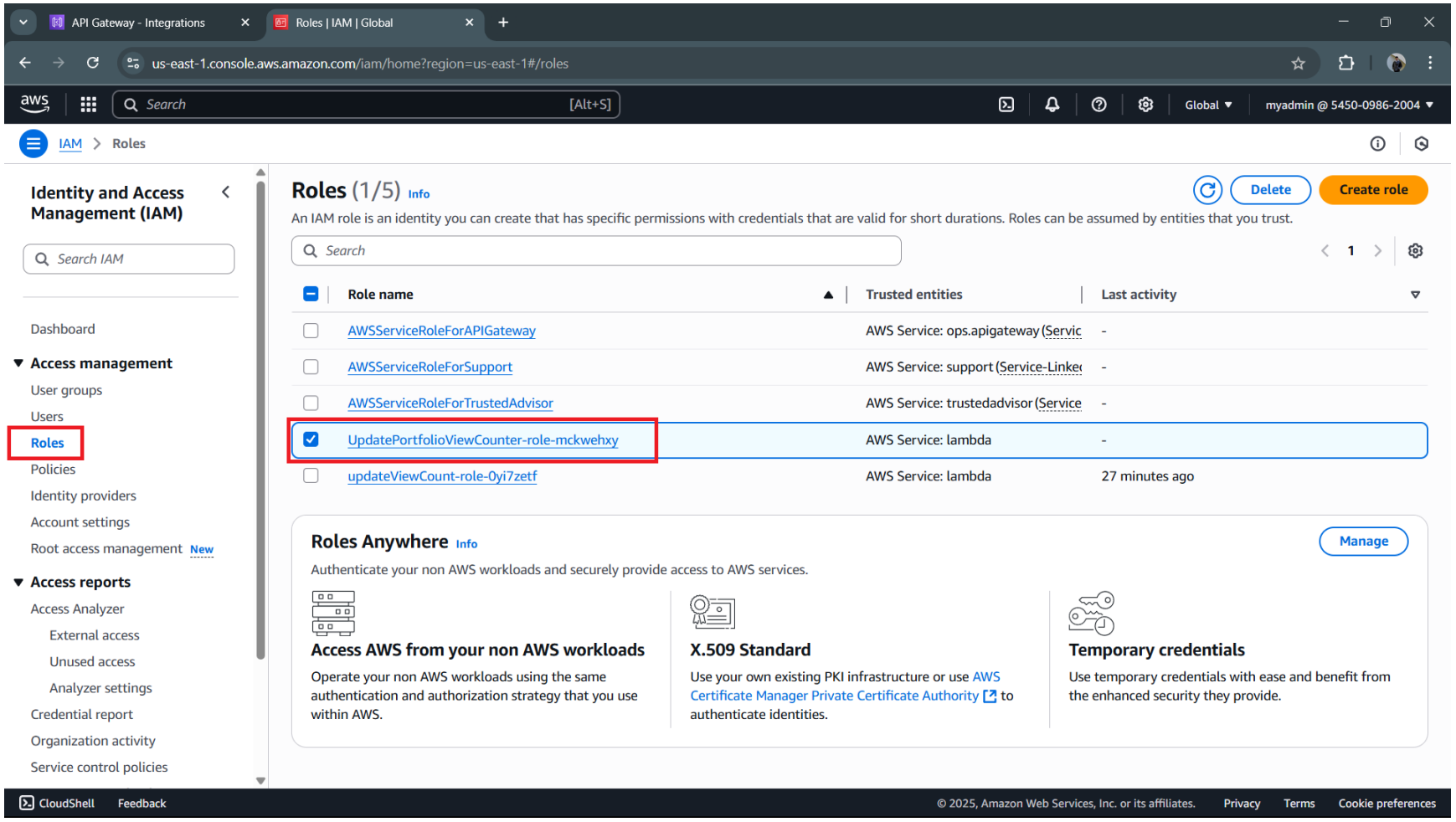
Finally, click on Create to deploy and create the API.



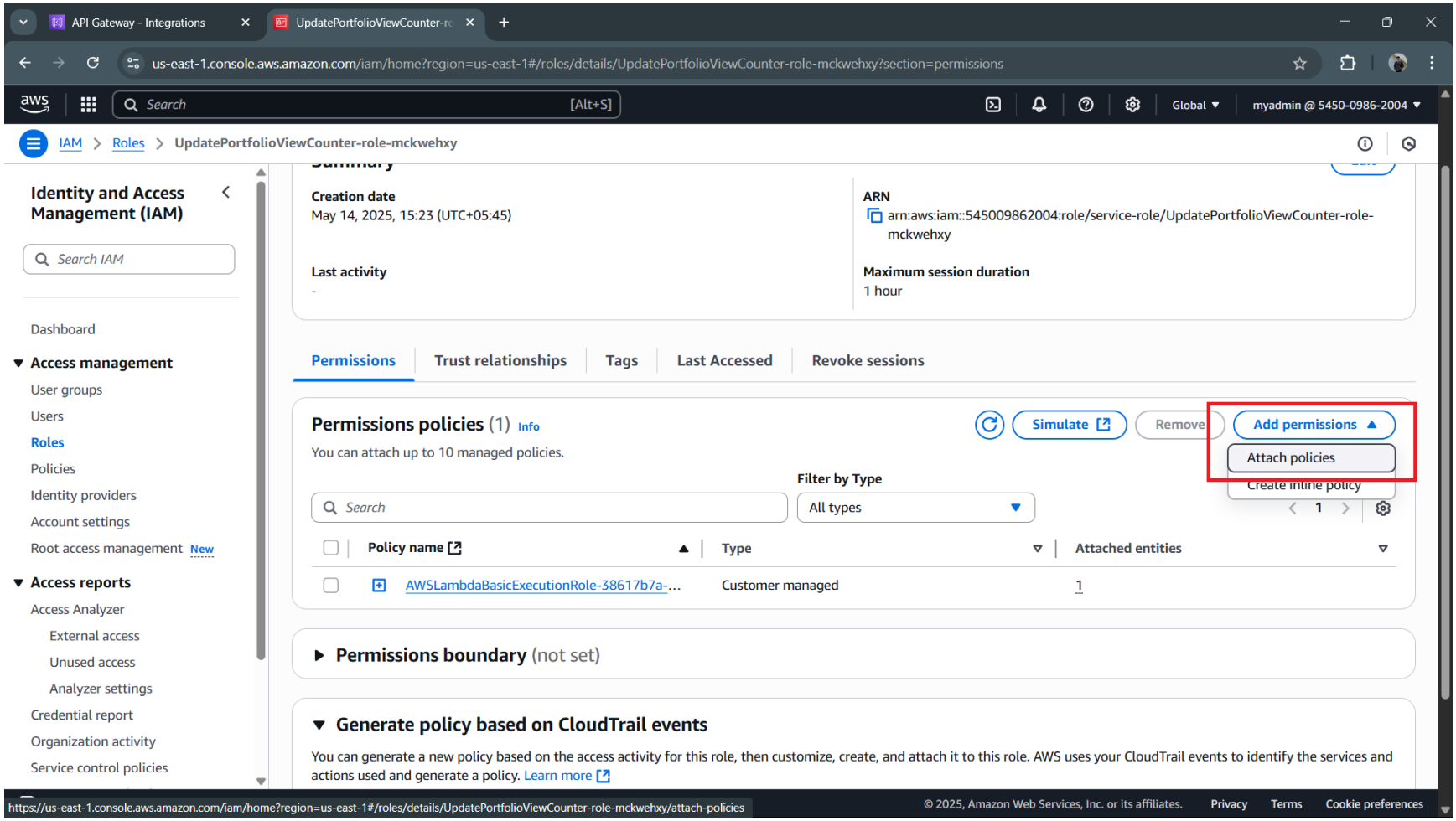
Now, navigate to the CORS settings and configure it as follows: a) Set Access-Control-Allow-Origin to * b) Select GET, POST, and OPTIONS under Access-Control-Allow-Methods c) Add Content-Type and Authorization under Access-Control-Allow-Headers. Then, click on Save to apply the changes.



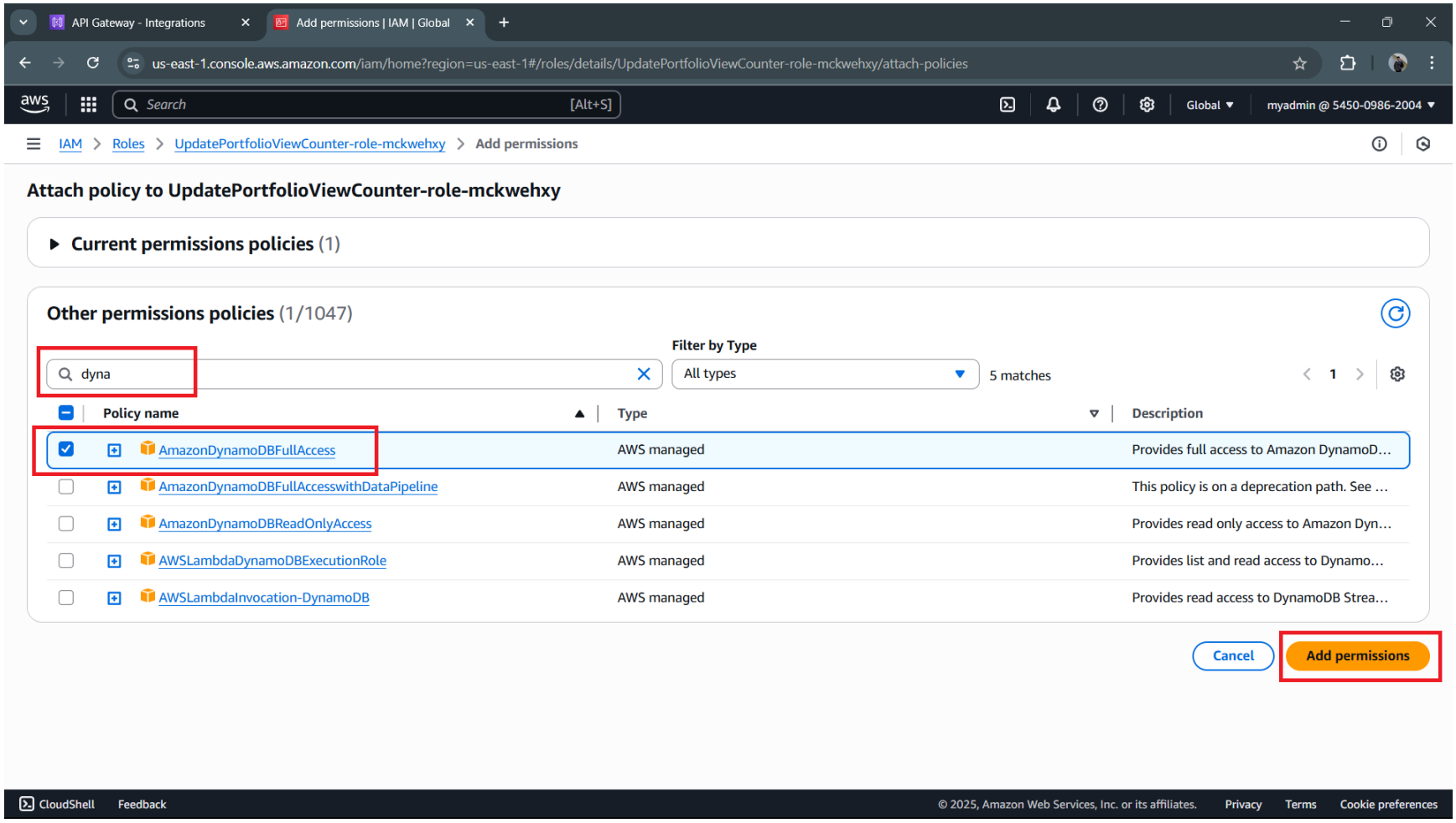
Now, navigate to IAM Roles, and click on the role named "UpdatePortfolioViewCounter-role" associated with the Lambda function. This role will be updated to grant the necessary permissions to access DynamoDB.



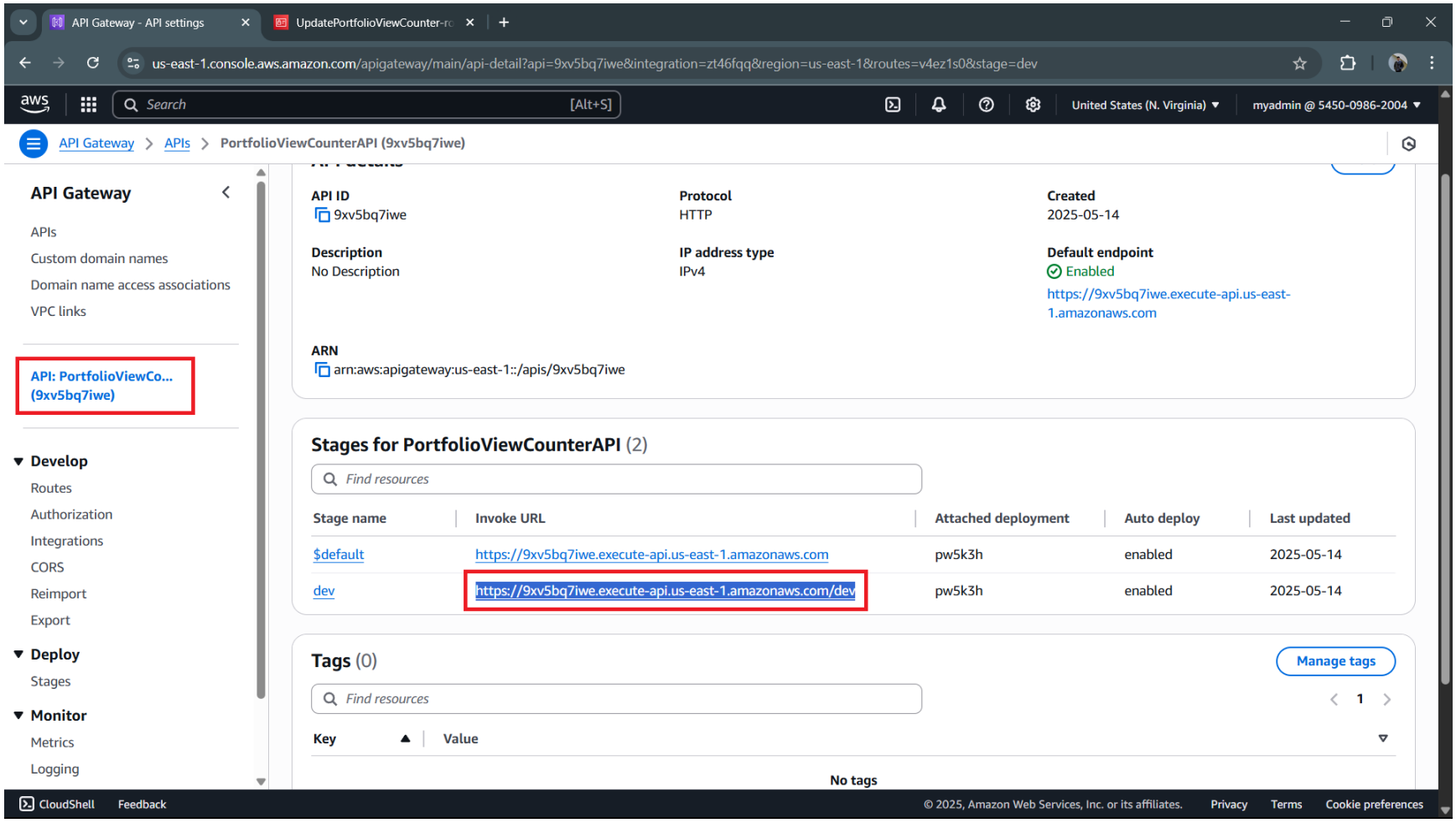
Click on Add Permissions, then select Attach Policy to proceed with granting the required permissions.



Search for DynamoDB, select the AmazonDynamoDBFullAccess policy, and then click on Add Permissions to grant the necessary access.

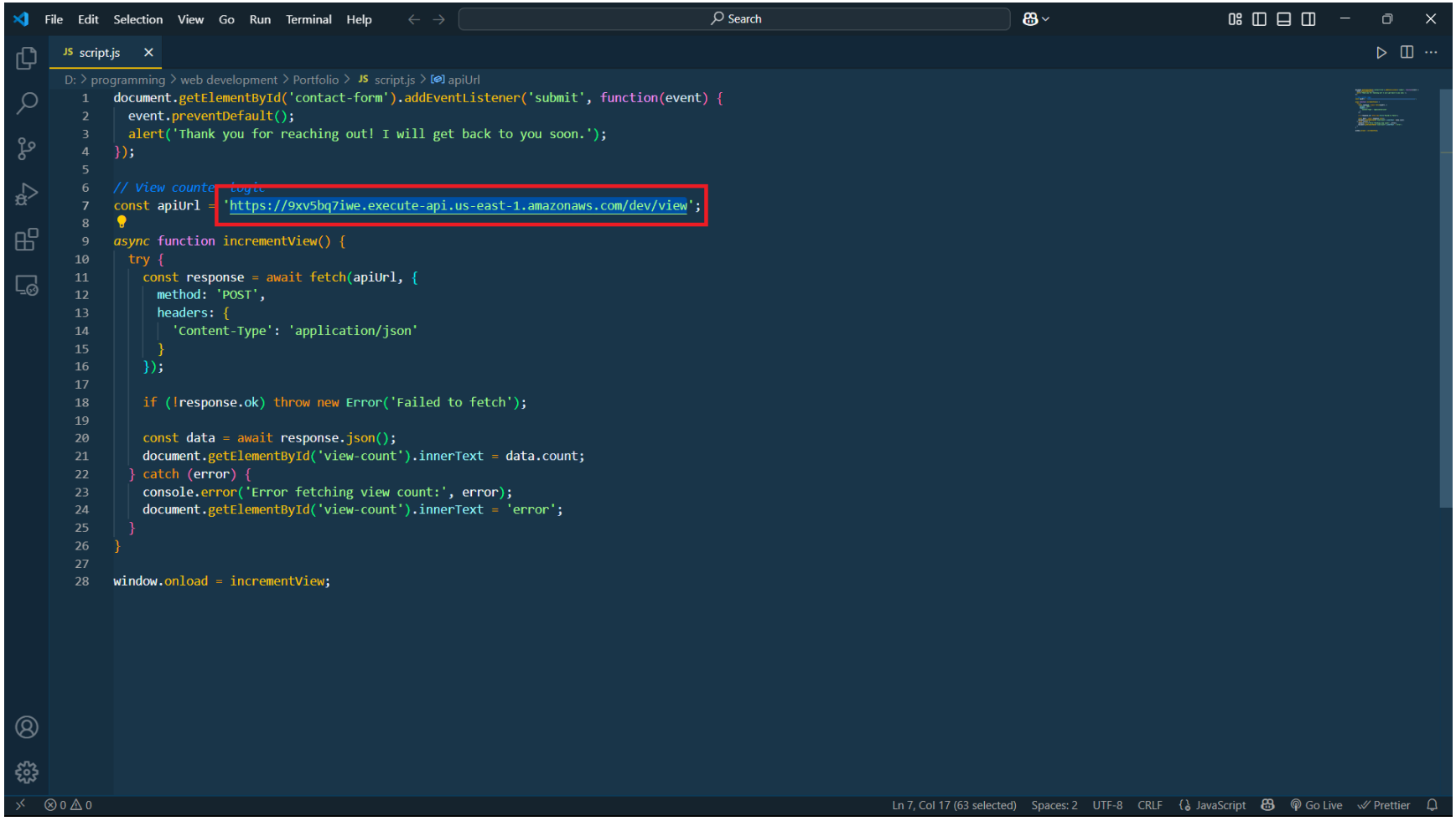


Navigate to API Gateway, click on PortfolioViewCounterAPI on the left side, then copy the Invoke URL for the dev stage.

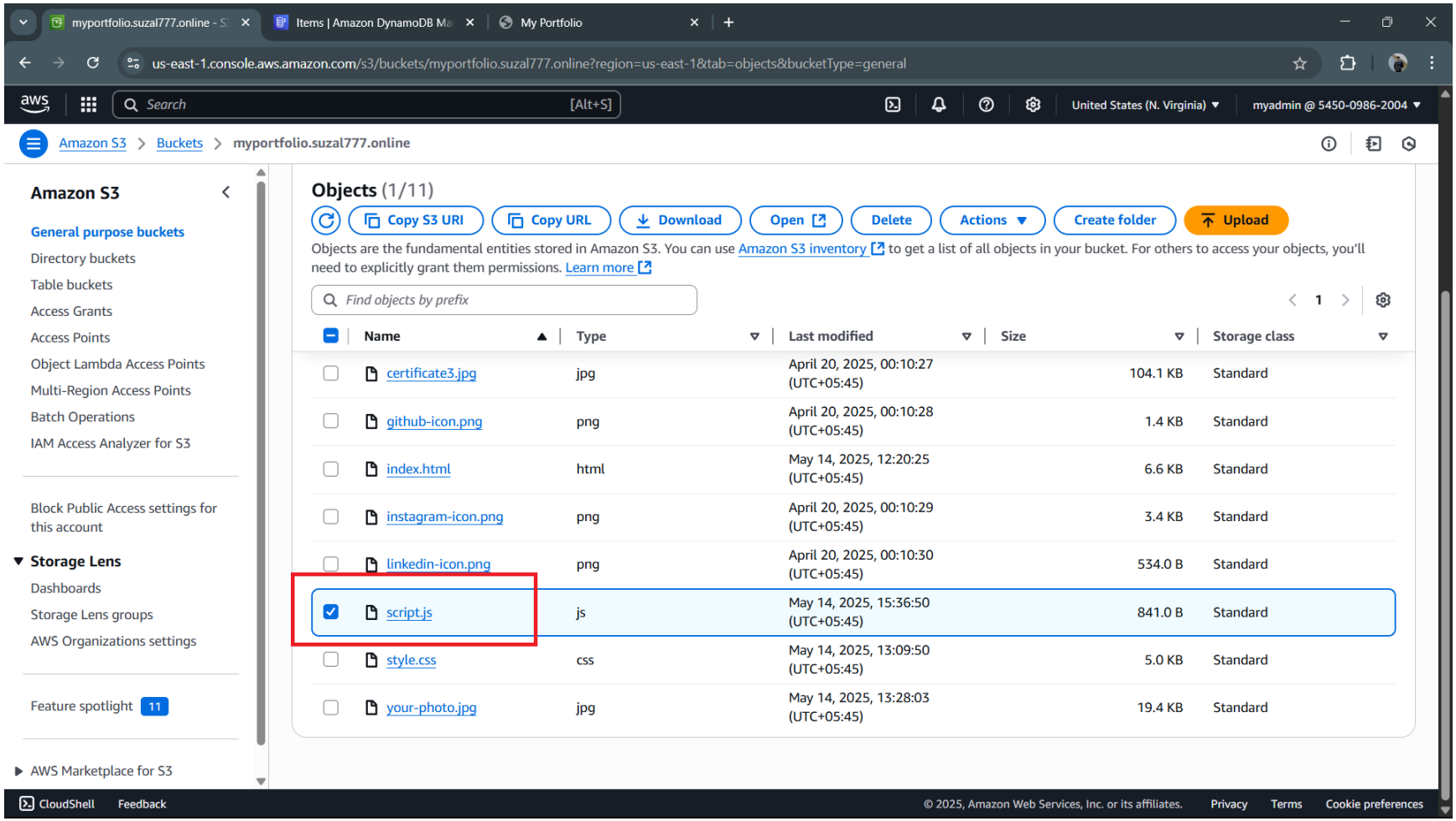


Open the script.js file and replace the apiUrl with the copied URL, appending /view at the end.

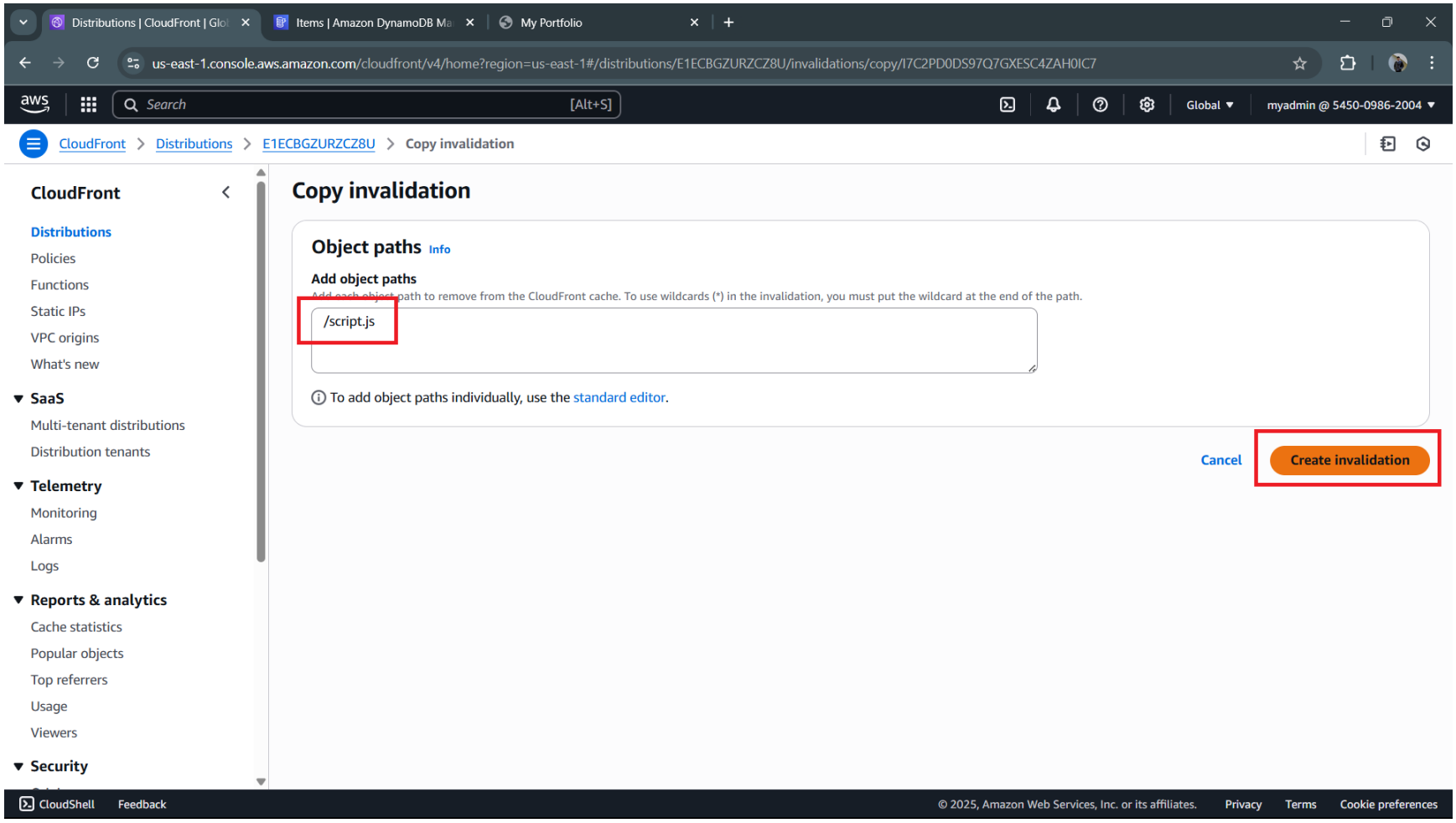
Note: In the updated javascript code, you can directly replace apiUrl in fetch section.



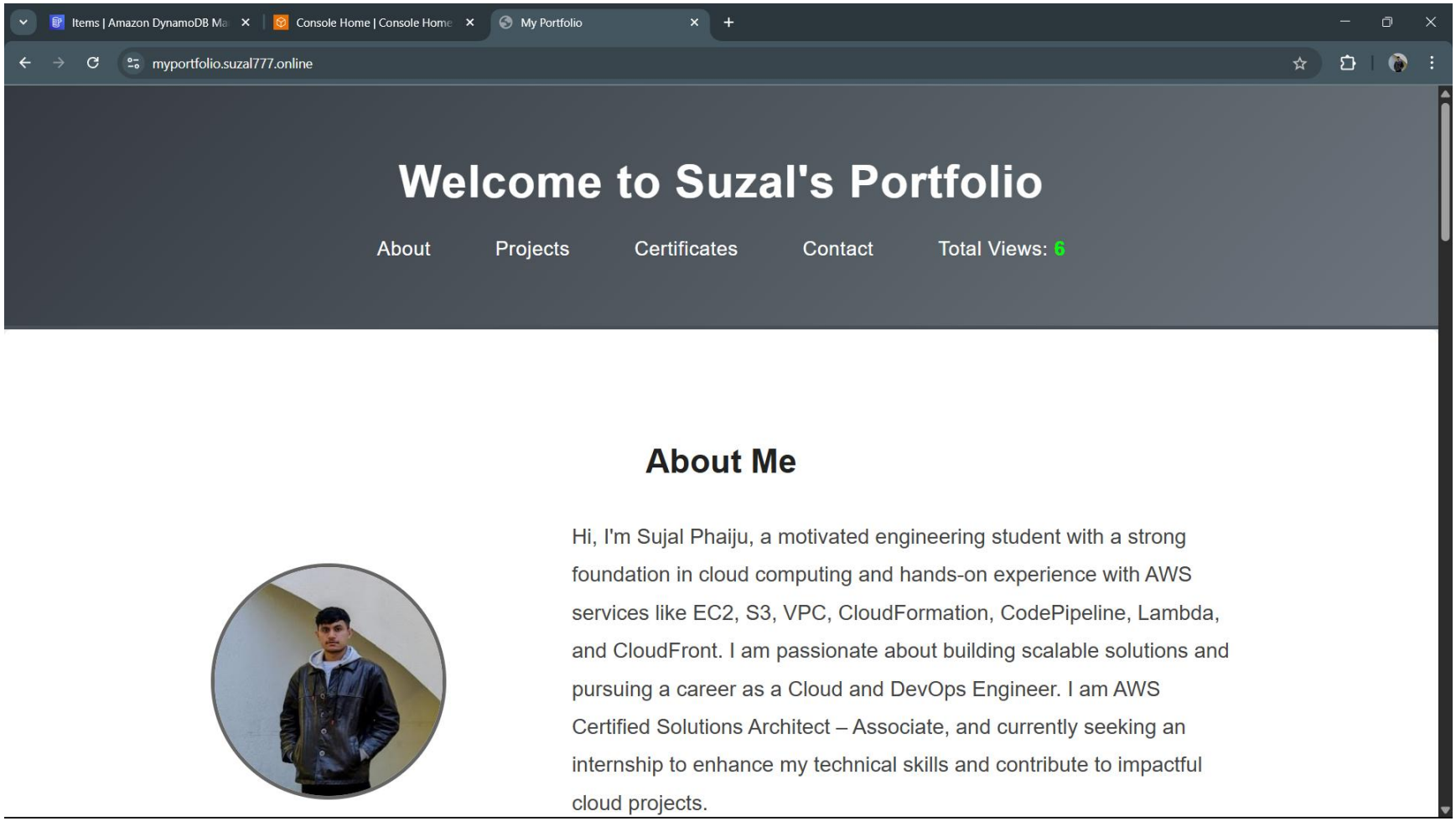
Upload the updated script.js file to your S3 bucket, replacing the previous version.



Navigate to your CloudFront distribution, go to the Invalidation section, and create a new invalidation. Add /script.js as the object path to clear the cached file.



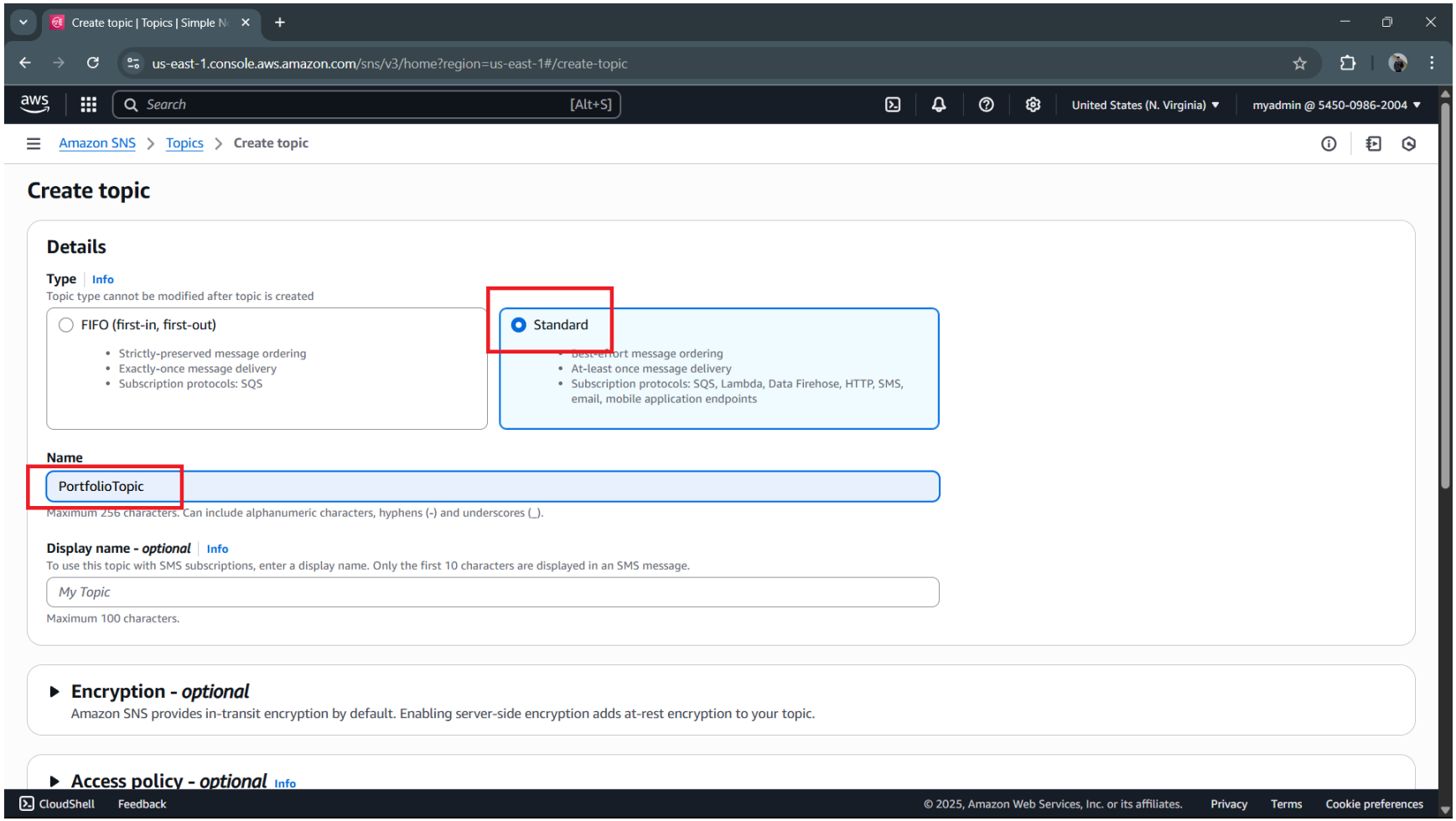
Your view counter is now operational on your portfolio website.



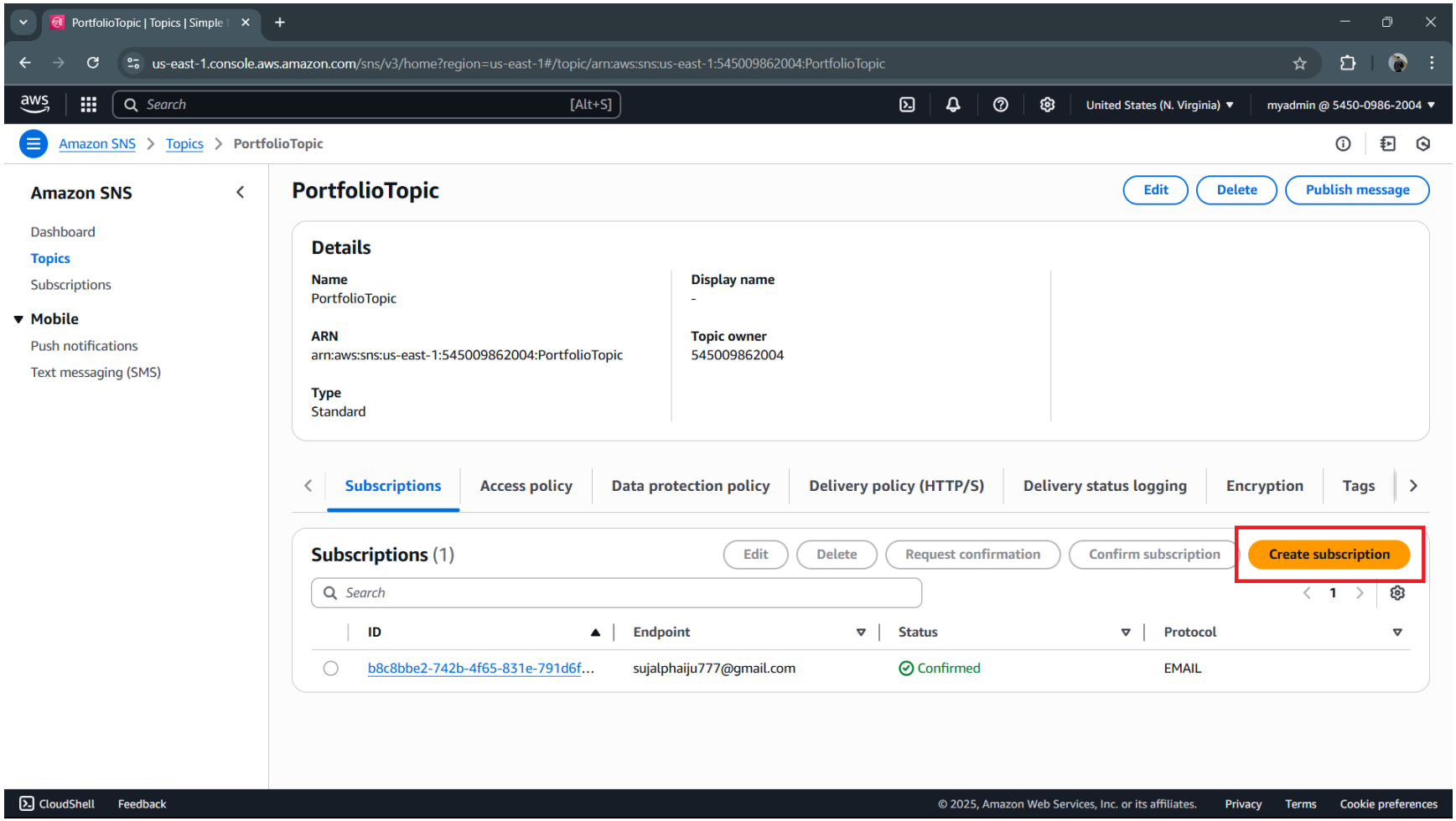
Contact Me feature

Creating SNS Topic :

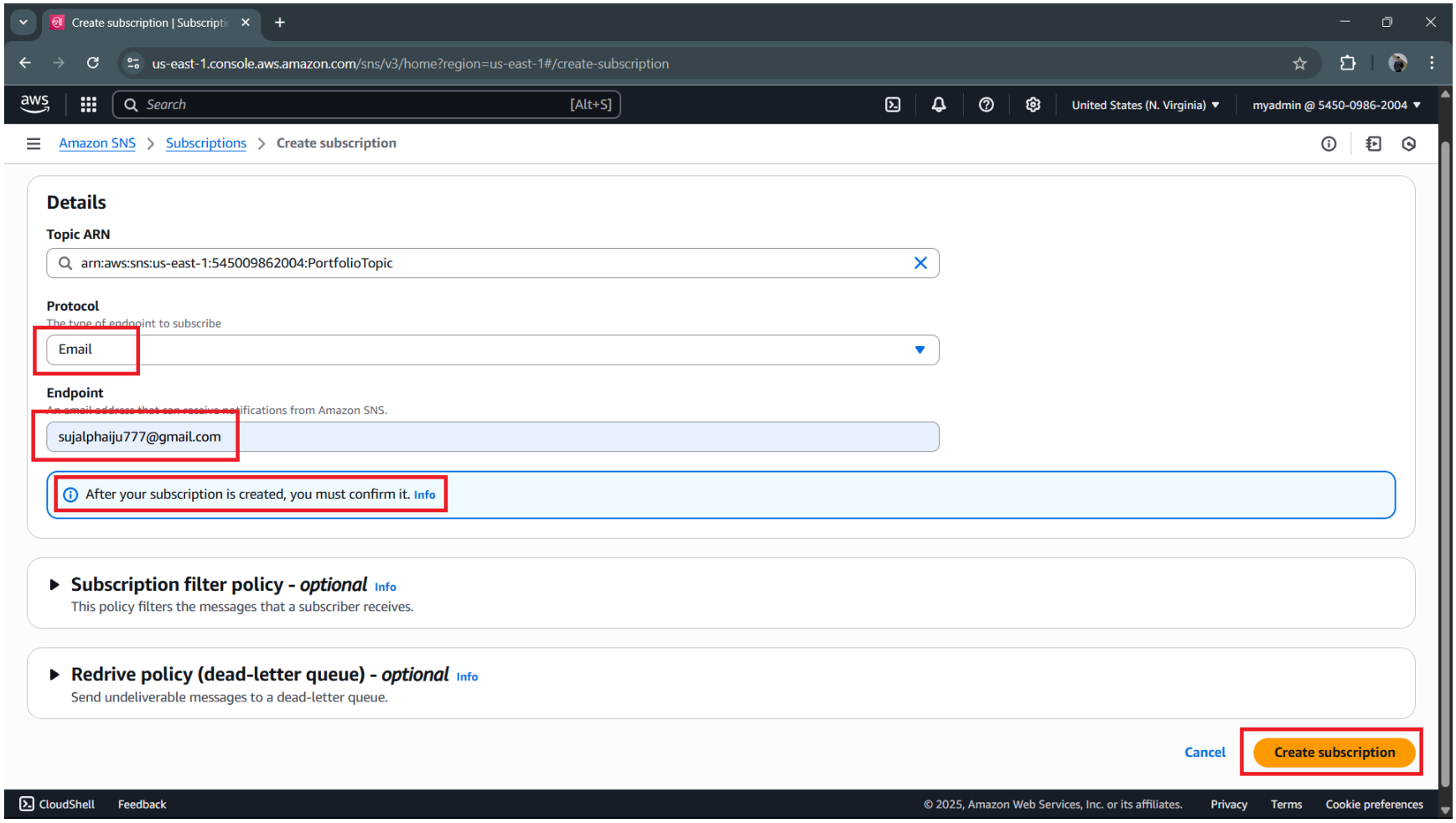
Go to SNS, navigate to the Topics section, and click on Create Topic. Select Standard as the topic type, enter "PortfolioTopic" as the name, and then click on Create Topic.



Click on the "PortfolioTopic" you just created, then click on Create Subscription to add a subscriber.

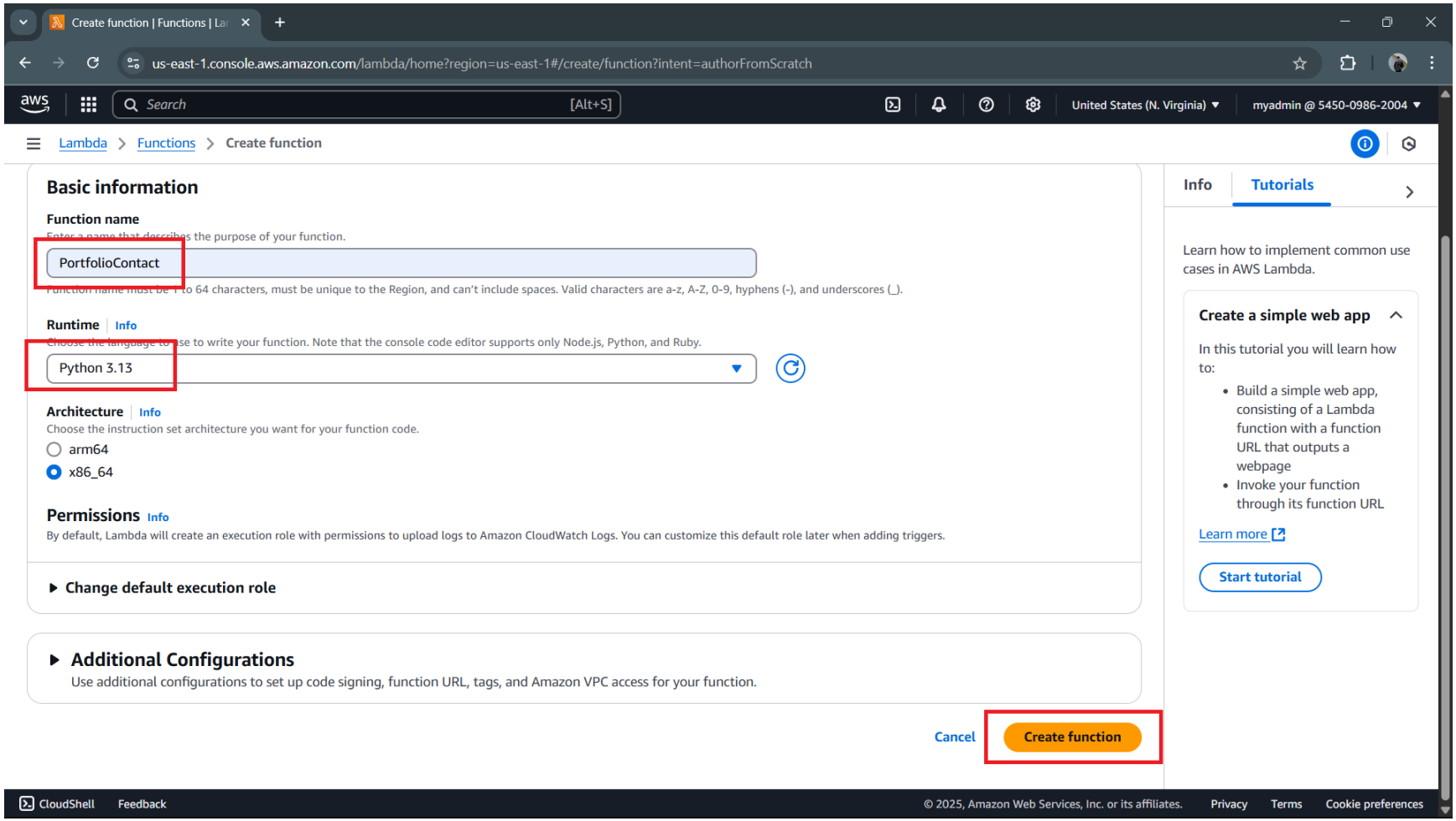


Set the Protocol to Email and enter your Gmail address as the Endpoint. Click on Create Subscription, then go to your inbox and confirm the subscription by clicking the link in the confirmation email.

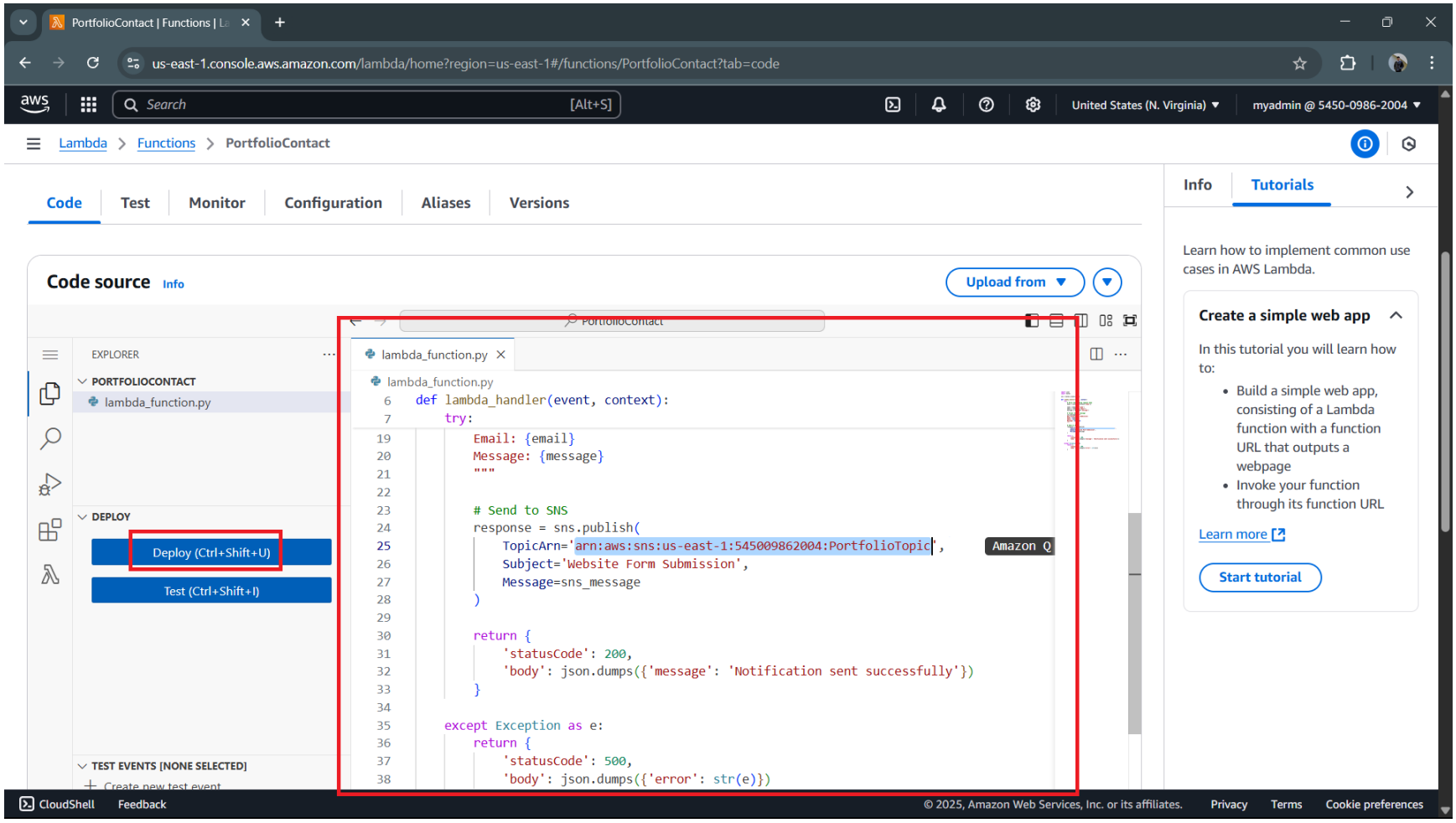


Creating Lambda Function:

Go to Lambda, create a new function named "PortfolioContact", select Python 3.13 as the runtime, and then click on Create Function to proceed.



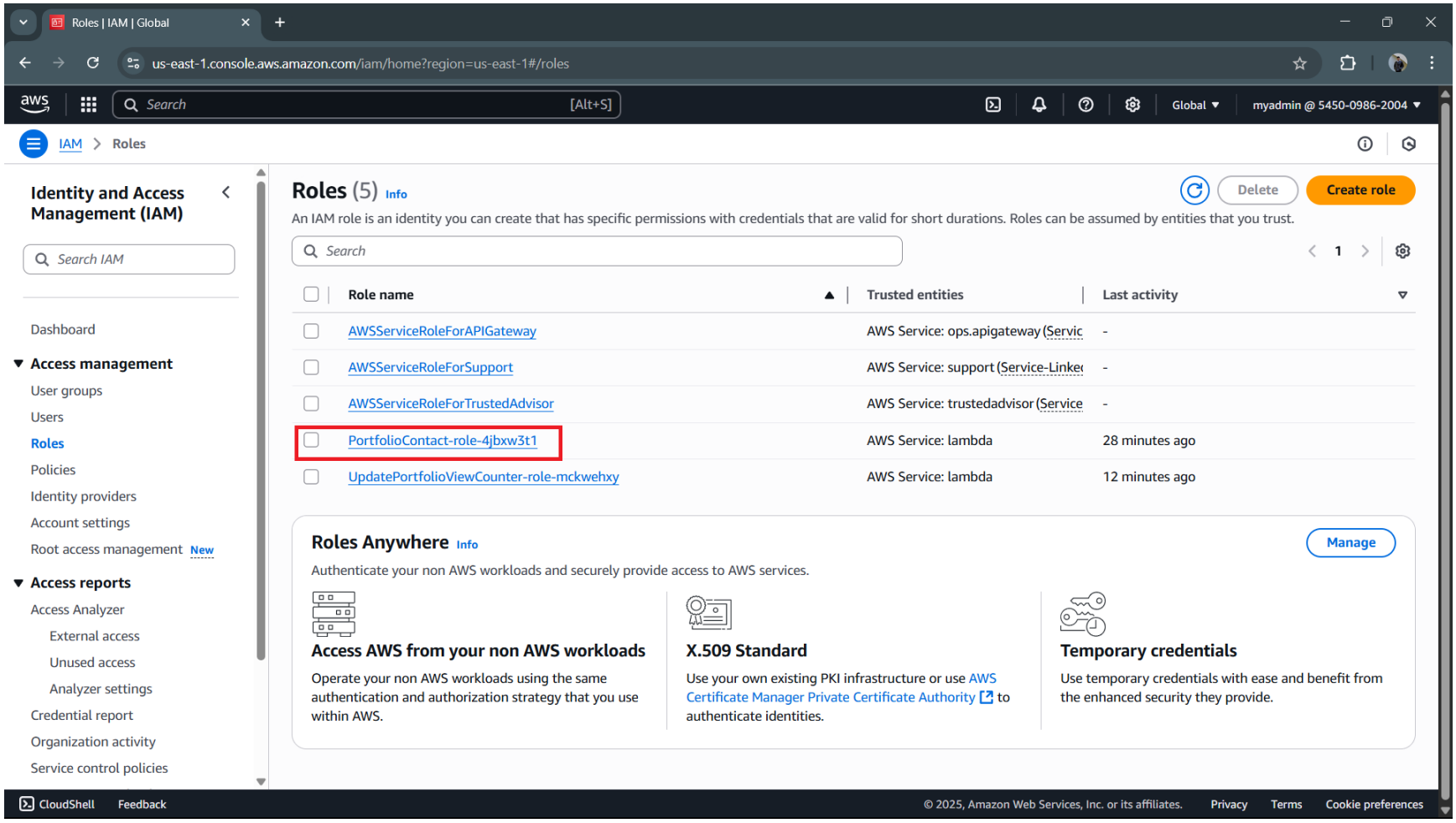
Copy the code provided below and paste it into the Code section of your PortfolioContact Lambda function. Make sure to update the TopicArn with the ARN of your SNS topic, then click on Deploy to apply the changes.



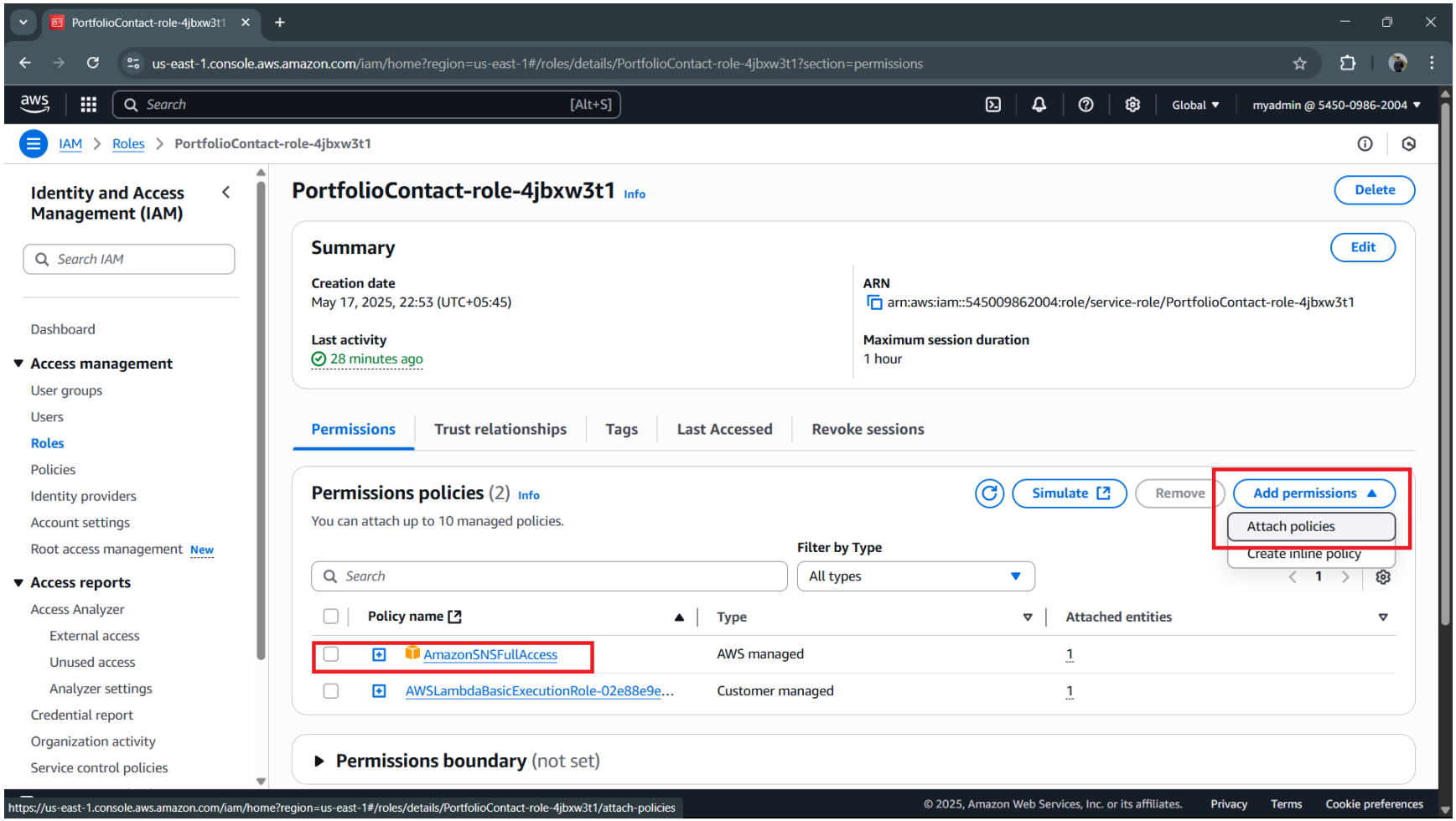
```python
import json
import boto3

sns = boto3.client('sns')

def lambda_handler(event, context):
    try:
        # Parse the incoming request body
        body = json.loads(event['body'])

        name = body.get('name')
        email = body.get('email')
        message = body.get('message')

        # Format the SNS message
        sns_message = f"""
        New Contact Form Submission:
        Name: {name}
        Email: {email}
        Message: {message}
        """

        # Send to SNS
        response = sns.publish(
            TopicArn='arn:aws:sns:us-east-1:545009862004:PortfolioTopic',
            Subject='Website Form Submission',
            Message=sns_message
        )

        return {
            'statusCode': 200,
            'body': json.dumps({'message': 'Notification sent successfully'})
```

```
    }

except Exception as e:
    return {
        'statusCode': 500,
        'body': json.dumps({'error': str(e)})
    }
```

Navigate to IAM Roles and select the role named "PortfolioContact-role" associated with your PortfolioContact Lambda function.

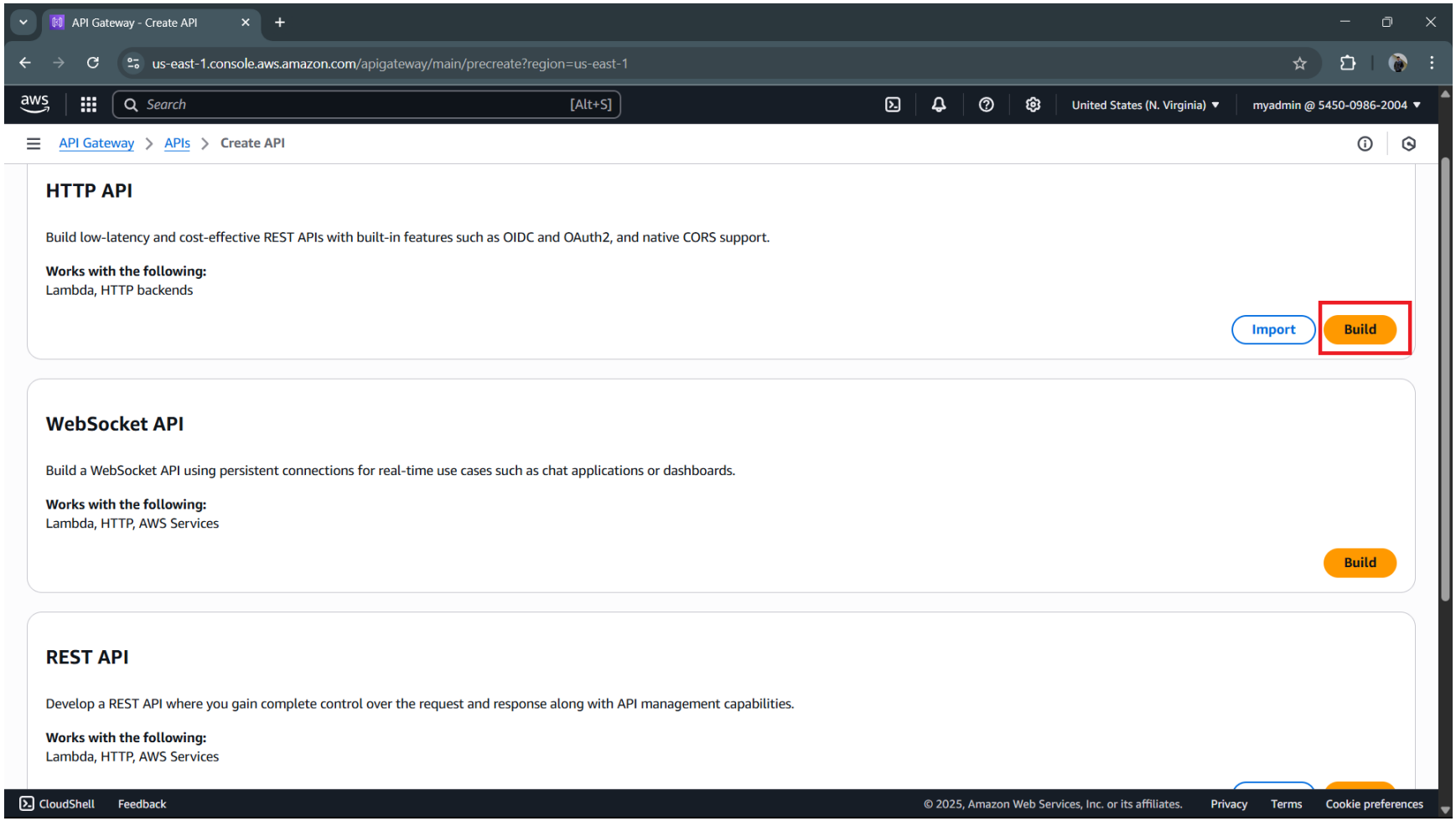


Attach the AmazonSNSFullAccess policy to the PortfolioContact-role to grant the necessary permissions for publishing messages to the SNS topic.

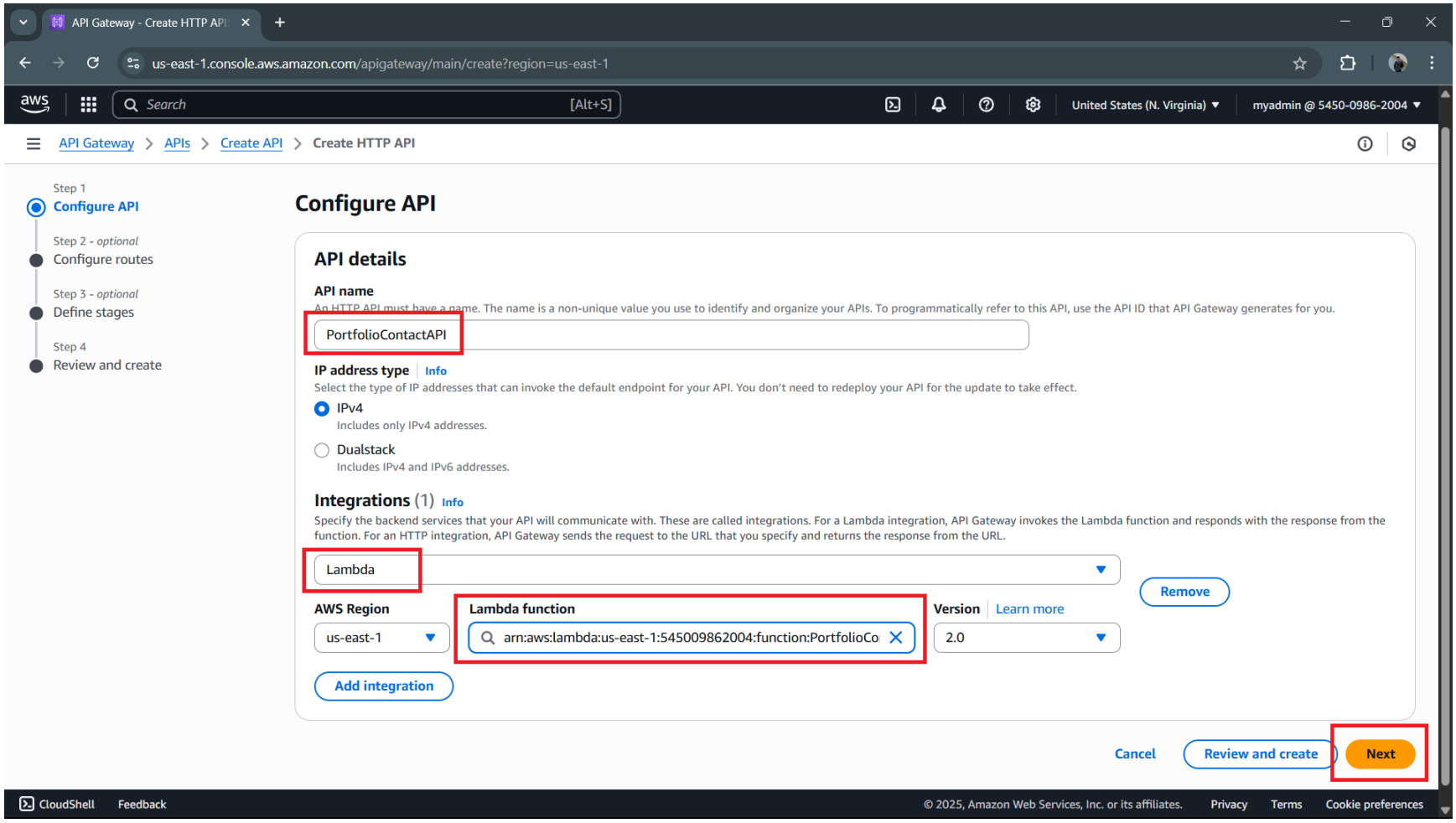


Creating API Gateway:

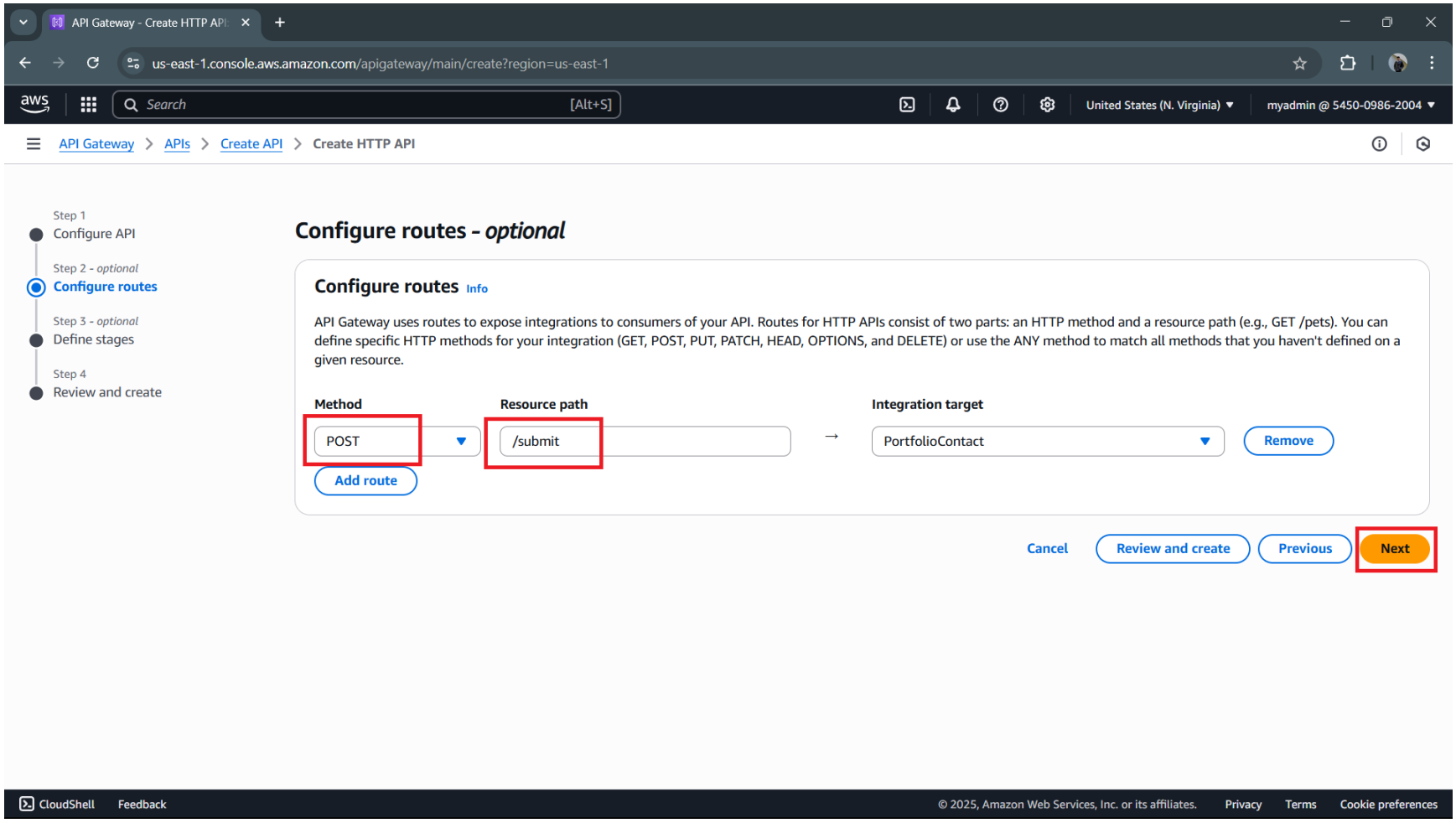
Go to API Gateway, create a new API, select HTTP API, and click on Build to begin the setup process.



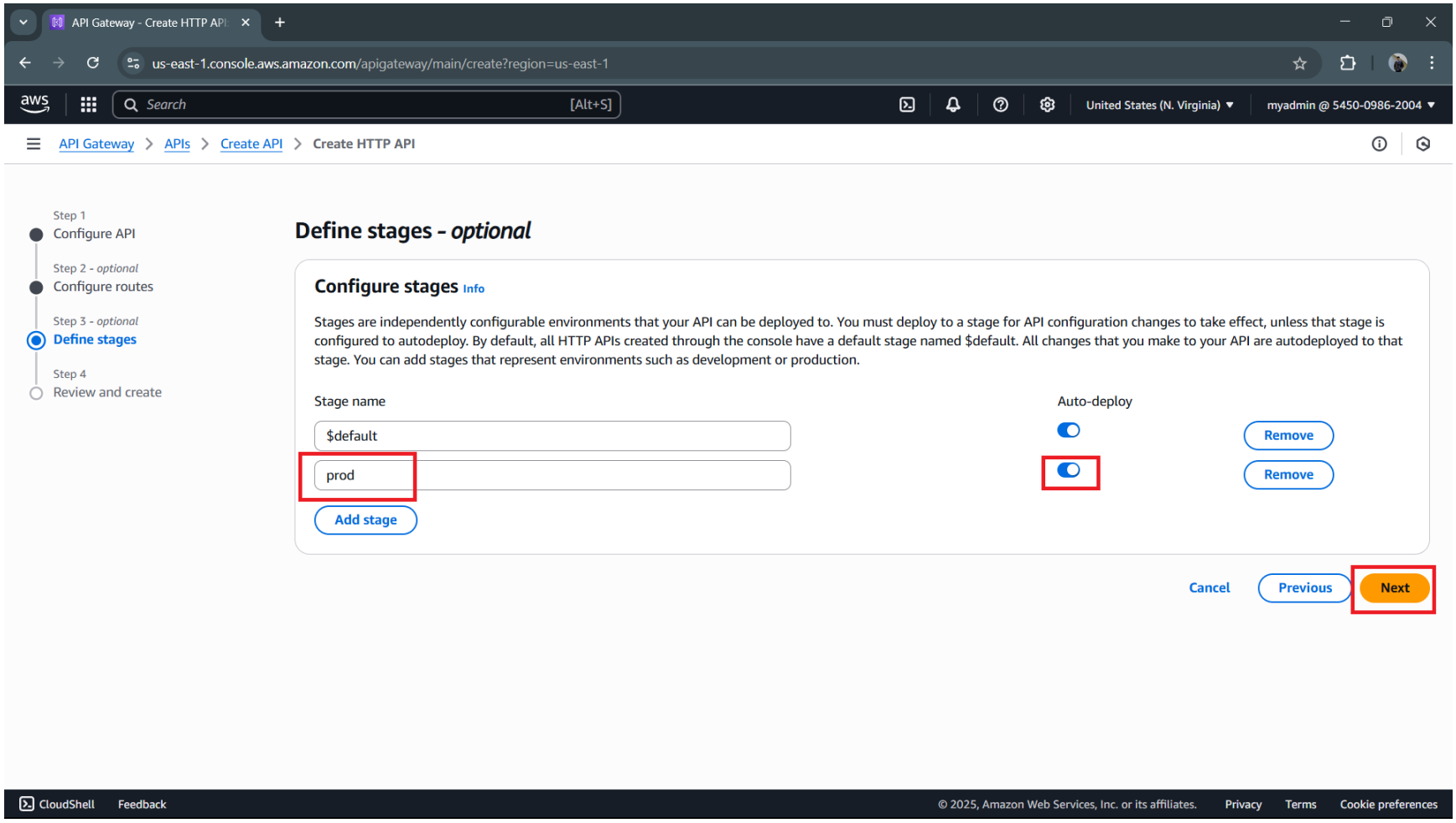
Enter "PortfolioContactAPI" as the API name. Choose Lambda as the integration type, and select the "PortfolioContact" Lambda function you just created.



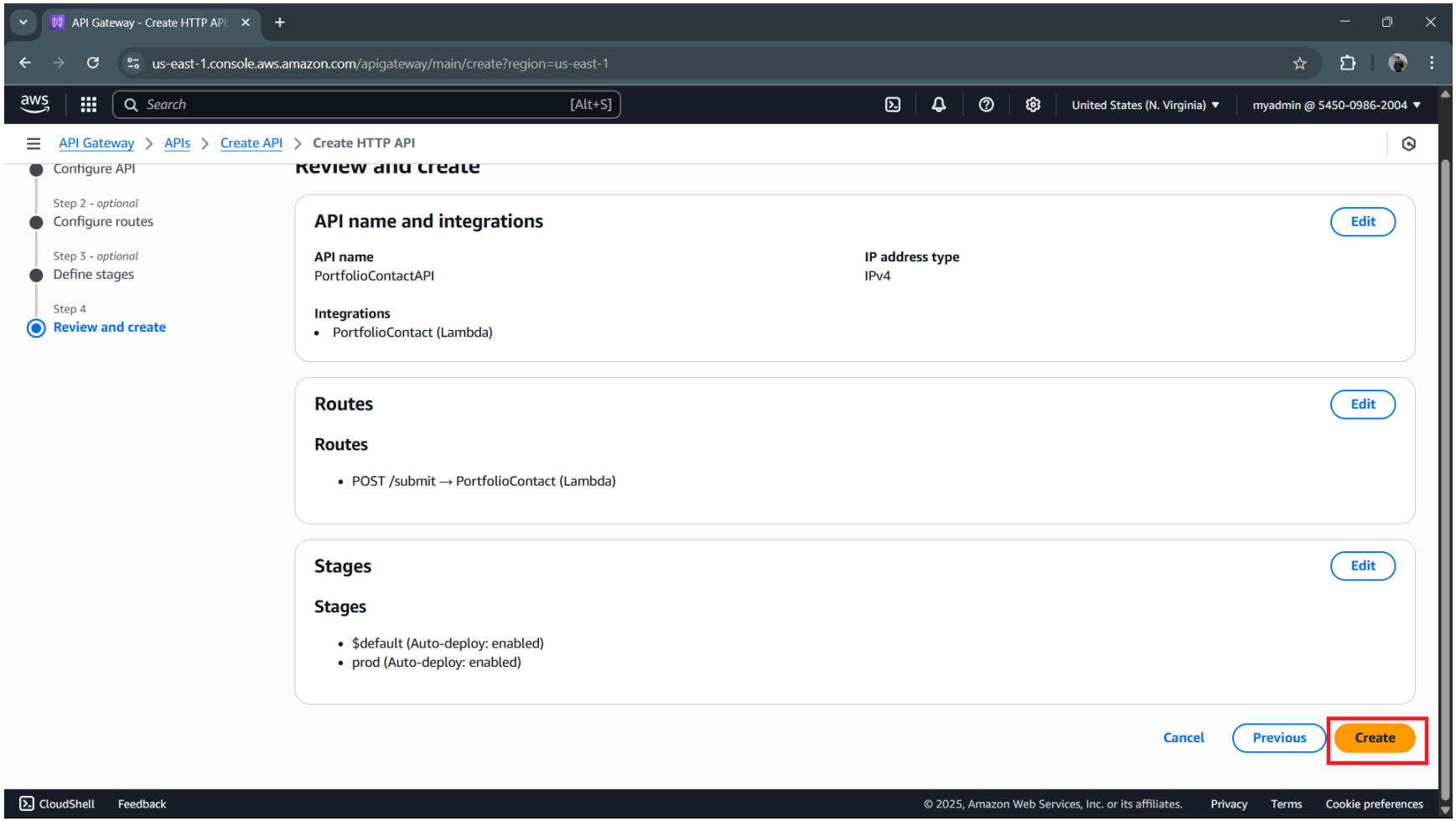
Set the method to POST and specify /submit as the resource path.



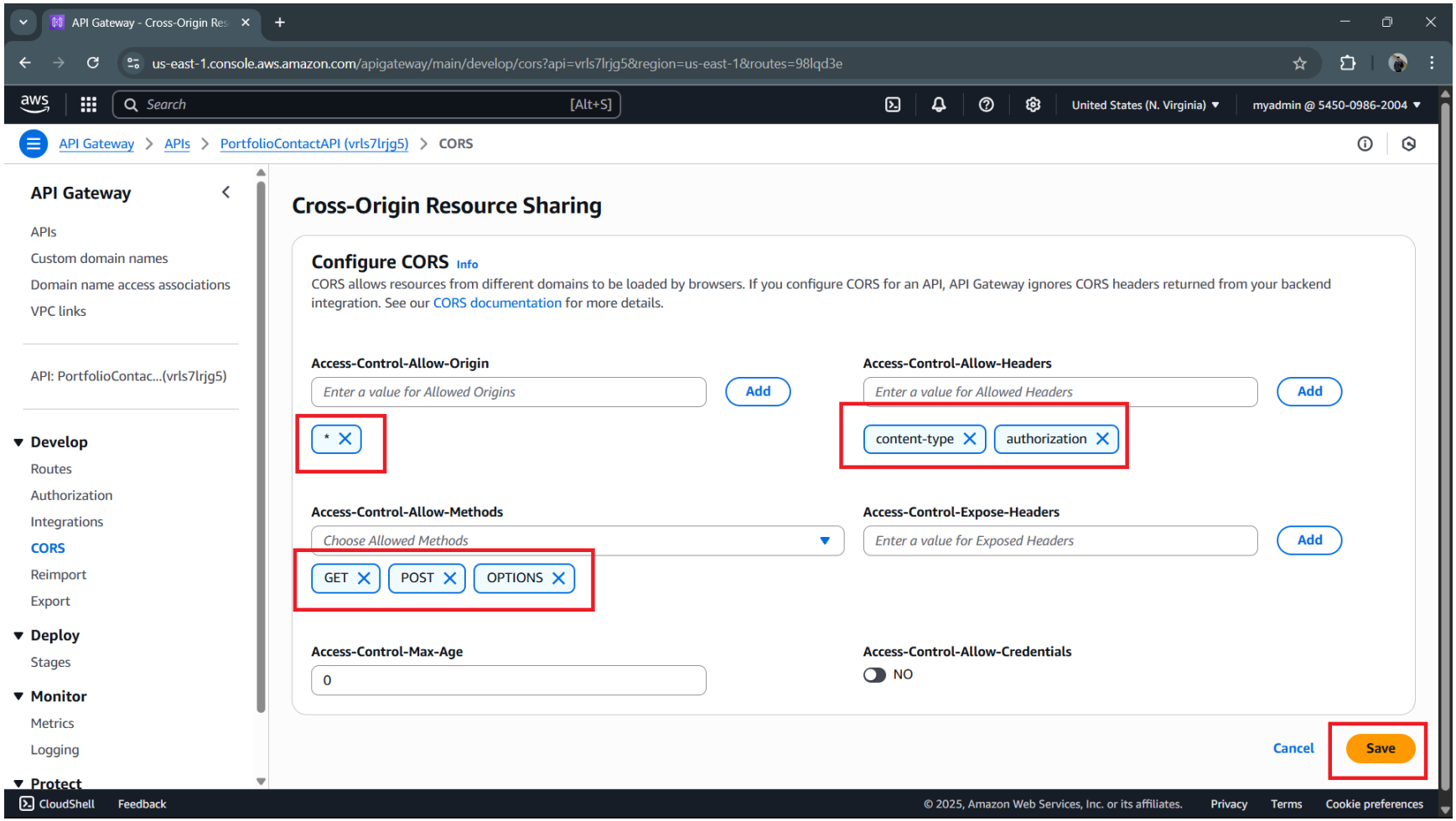
Add a stage named prod and enable Auto Deploy.



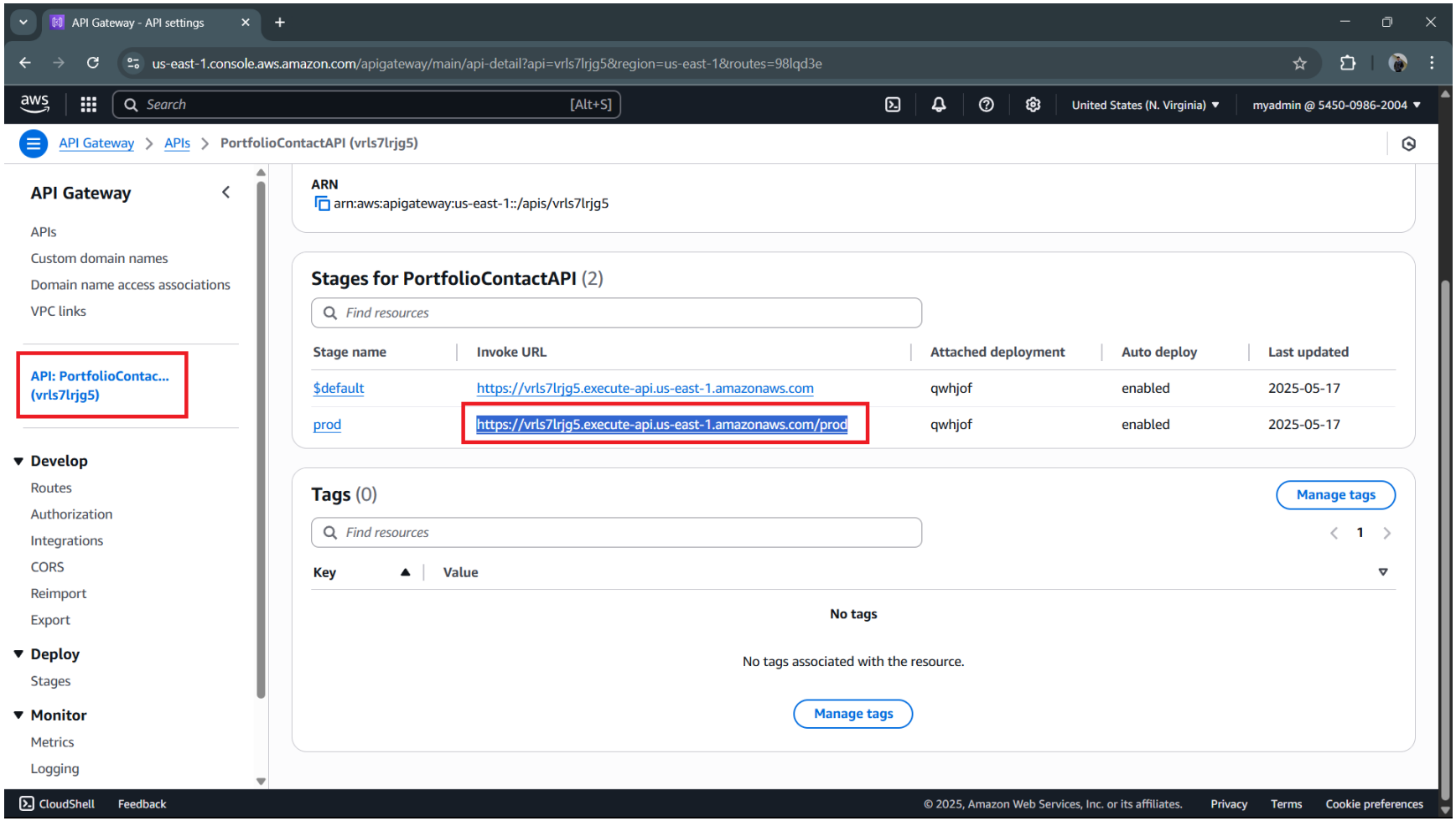
Review your settings and click on Create API to finalize the setup.



Navigate to the CORS settings for the API. Configure the settings exactly as you did for the previous API by setting Access-Control-Allow-Origin to *. Allow the methods GET, POST, and OPTIONS, and add Content-Type and Authorization to the allowed headers. Finally, save the changes.



Click on your API and copy the Invoke URL for the prod stage.



Paste the copied URL into your script.js file, appending /submit at the end. Update the script.js file in your S3 bucket with the new version. Then, create an invalidation in CloudFront to clear the cached file. If necessary, perform a hard refresh in your browser to see the changes.