Serverless Registration Portal

03.19.2024

—

Rtt-41-2023

AWS re/start Cloud Practitioner Course

CLF-C02 Exam

Per Scholas

November 2023 - March 2024

Instructors: Sajini George & Jose Santos

Overview

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper.

Goals

1. Lorem ipsum dolor sit amet, consectetuer adipiscing elit
2. Sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Specifications

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.

Lorem Ipsum

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.

Steps

Console

Part 1: S3 Static Hosting

1. Go to your console
2. Go to S3
3. Create s3 bucket
4. Choose default settings

5. For ‘bucket name’ choose anything. E.g. bucket name: “my s3 bucket”
6. unblock public access, to make it public
7. click ‘i acknowledge’
8. create bucket
9. click on the newly created bucket
10. for objects tab, click ‘upload’ file “home.html” (this is home.html script) (Visual Code Studio to view this code easier)

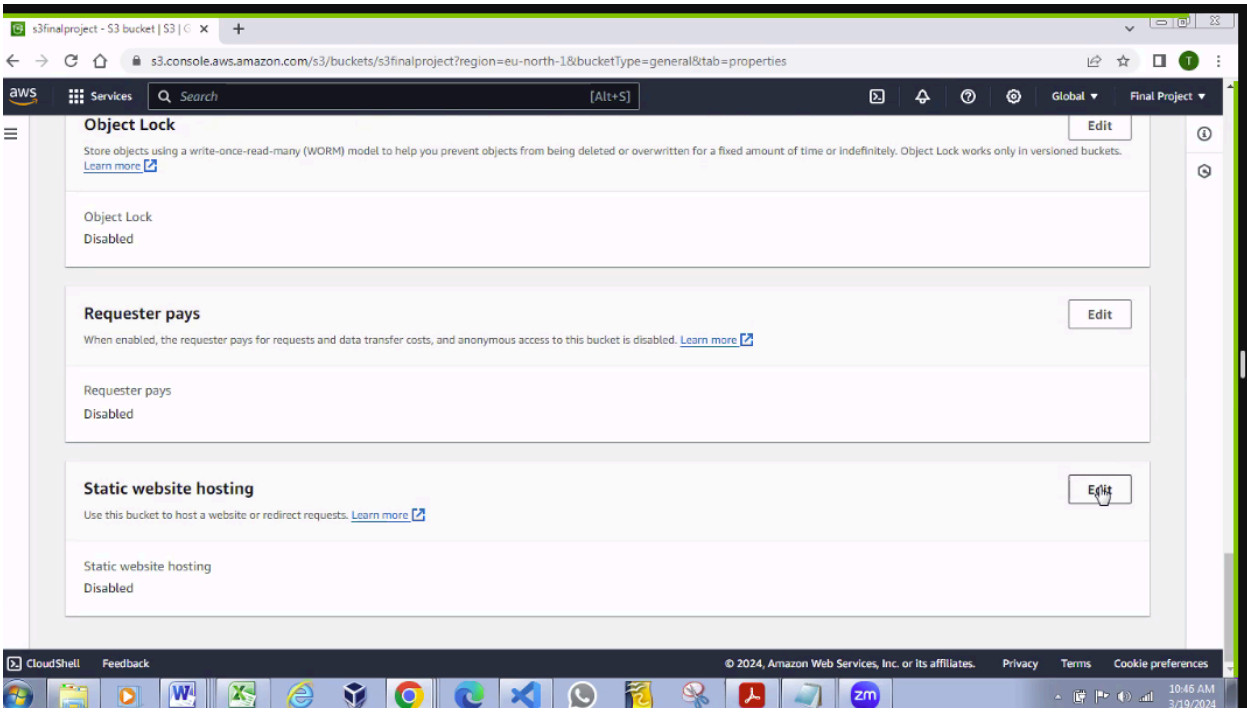
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>A Helping Hand: Community Volunteers for Senior Support</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }
        header {
            background-color: #002657;
            color: #fff;
            padding: 20px;
            text-align: center;
        }
        h1 {
            margin-top: 0;
        }
        .container {
            max-width: 800px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
```

```
        p {
            line-height: 1.6;
        }

        #welcomeHeading {
            color: #01377D; /* Change 'blue' to any color you desire */
        }

        #welcomeHeading1 {
            color:white; /* Change 'blue' to any color you desire */
        }
        nav ul {
            list-style-type: none;
            margin: 0;
            padding: 0;
        }
        nav ul li {
            display: inline;
            margin-right: 20px;
        }
        nav ul li a {
            color: #fff;
            text-decoration: none;
        }
    </style>
</head>
<body>
    <header>
        <h1 id="welcomeHeading1">A Helping Hand: Community Volunteers for Senior Support</h1>

    </header>
    <div class="container">
        <h2 id="welcomeHeading">Welcome to our Community!</h2>
        <p>We provide assistance and support to seniors in our community through dedicated volunteer efforts. Our volunteers offer various
```

```
    services such as companionship, assistance with daily tasks,
    transportation, and more.</p>
65.
66.         <img src="senior_support_image1.jpg" alt="Senior Support Image 1"
    style="width: 100%; max-width: 600px; display: block; margin: 0 auto
    20px;">
67.
68.         <h3 id="welcomeHeading">How to Get Involved</h3>
69.
70.         <p>If you're interested in becoming a volunteer or if you're a
    senior in need of assistance, please contact us using the information
    below:</p>
71.         <p><a href="Profile.html">Register by Submitting the Form</a></p>
72.
73.         <p>OR</p>
74.
75.         <p>Contact Email: volunteer@helpinghand.org</p>
76.         <p>Phone: (555) 123-4567</p>
77.         <p>Address: 123 Main Street, City, State, ZIP</p>
78.     </div>
79. </body>
80. </html>
```

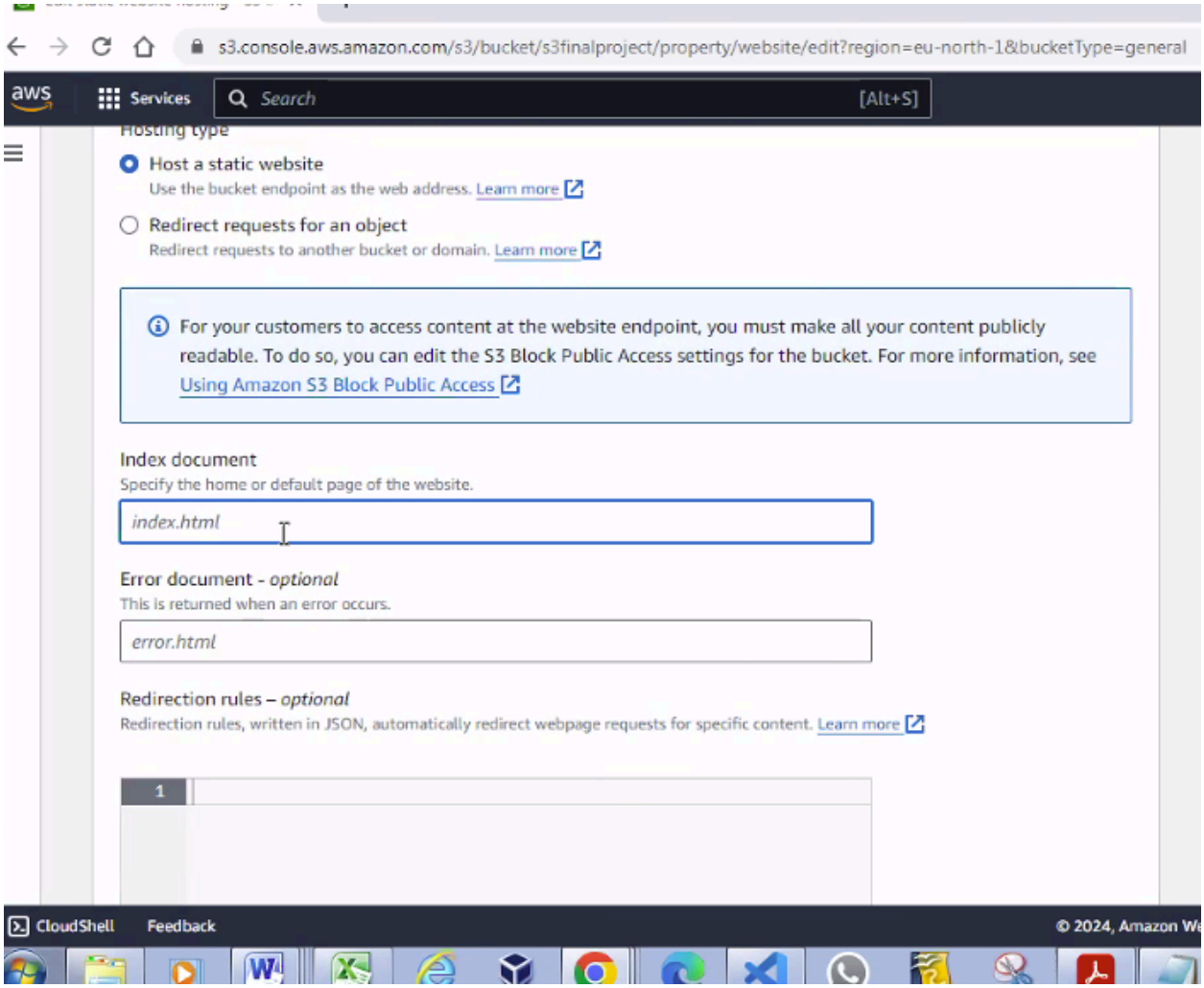
11. Add this image:

12. click on add files.
13. add those files
14. go down and the bottom corner will show ‘upload’ button
15. Click it
16. after uploading, click ‘close’
17. on the bucket ‘properties’ tab, click on that tab
18. scroll down

1. 

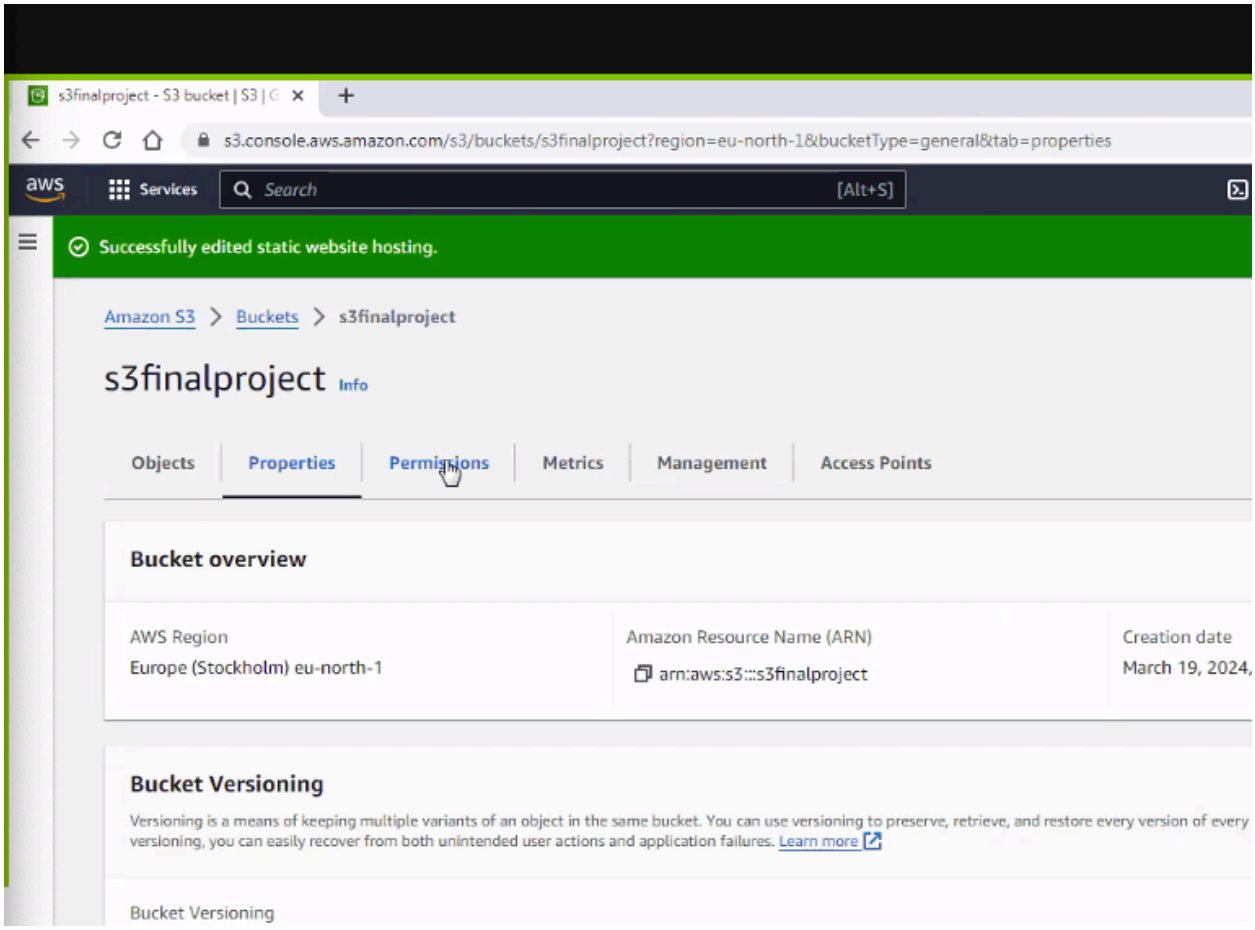
for ‘static website hosting’, click ‘edit’ and enable it
2.

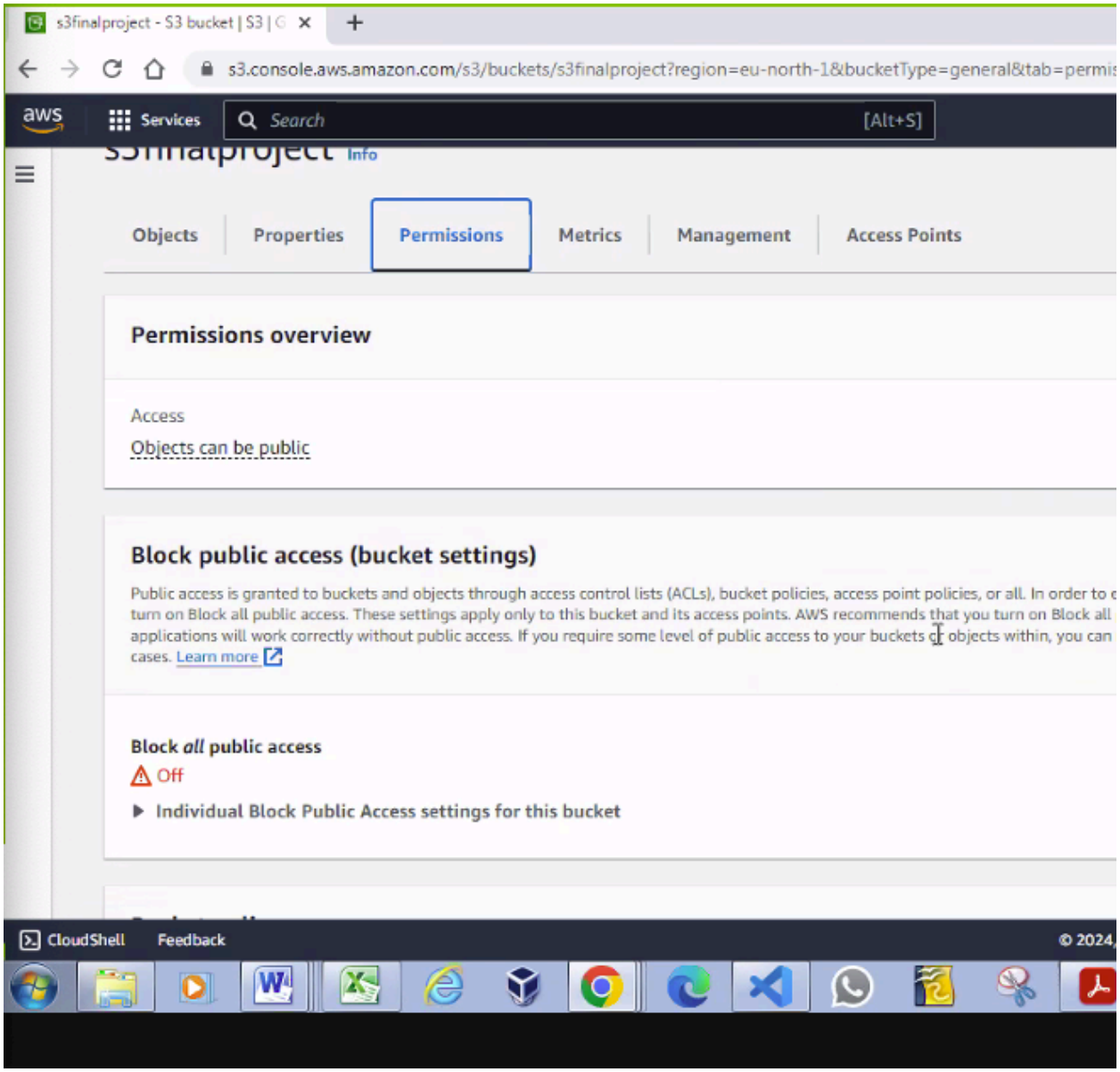
19. For ‘Index Document’ write “Home.html” [MAKE SURE IT IS WRITTEN WITH CAPITAL “H” !!!]



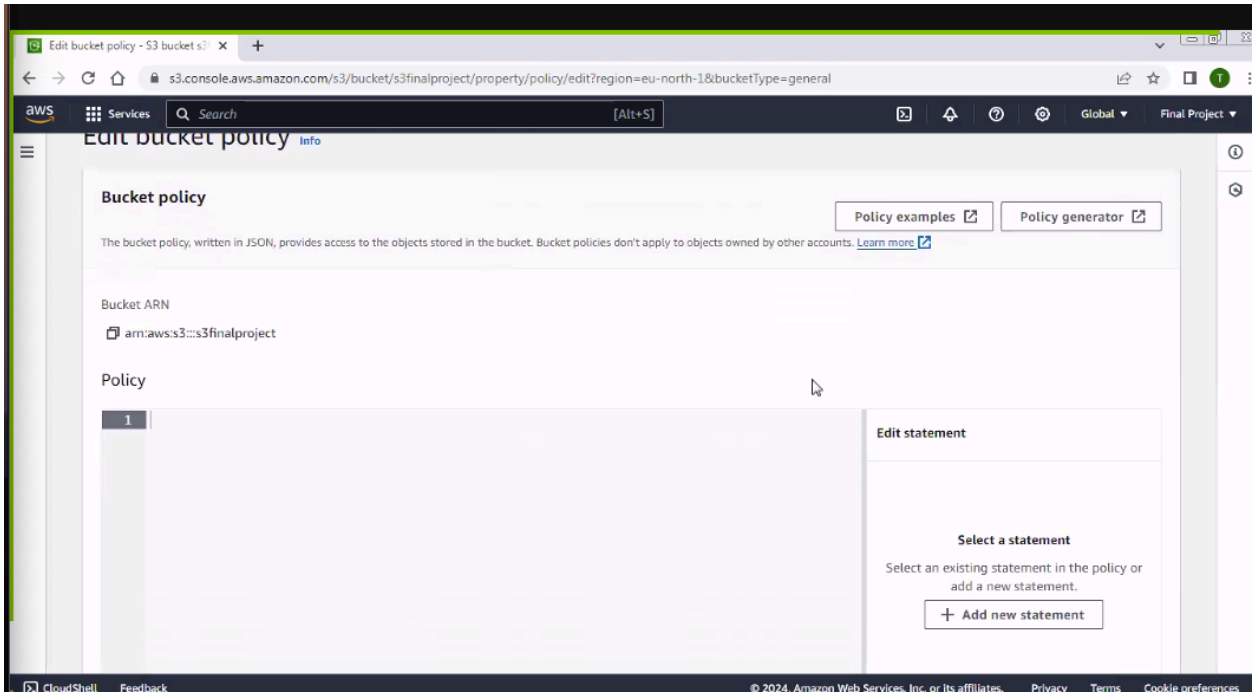
20. scroll down ‘save changes’

21. go ‘permissions’ tab




s3finalproject - S3 bucket | S3 |
s3.console.aws.amazon.com/s3/buckets/s3finalproject?region=eu-north-1&bucketType=general&tab=permis
aws
Services
Search
[Alt+S]
Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Permissions overview
Access
Objects can be public
Block public access (bucket settings)
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to
turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all
applications will work correctly without public access. If you require some level of public access to your buckets
objects within, you can
cases. Learn more
Block all public access
Off
Individual Block Public Access settings for this bucket
CloudShell
Feedback
© 2024,

22. go down click ‘edit’ on bucket policy (Take note of ‘Bucket ARN’)

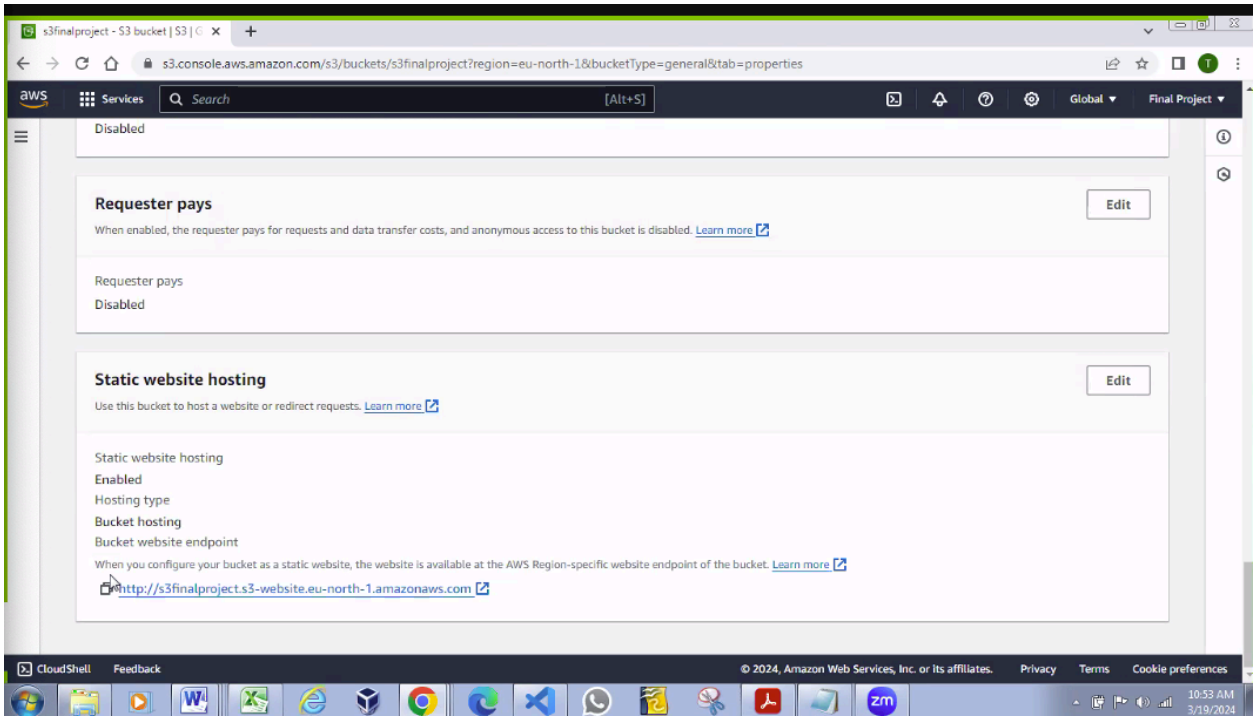


23. paste this script on the policy (NAMEOFYOURBUCKET should be your bucket name) Note: above ‘policy’ you will see “bucket ARN” copy and paste that into the ‘resource’ section below

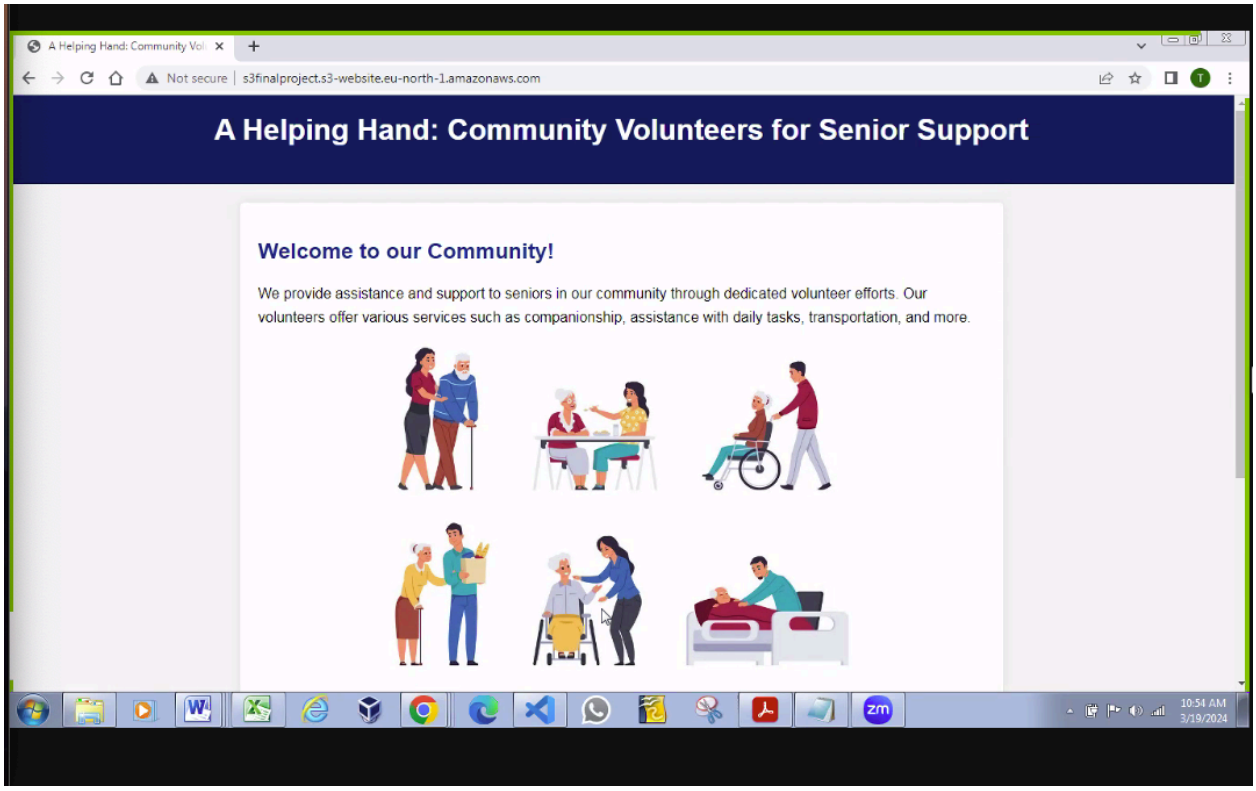
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::NAMEOFYOURBUCKET/*”
        }
    ]
}
```

24. then save changes
25. go to propterties tab

26. Go down



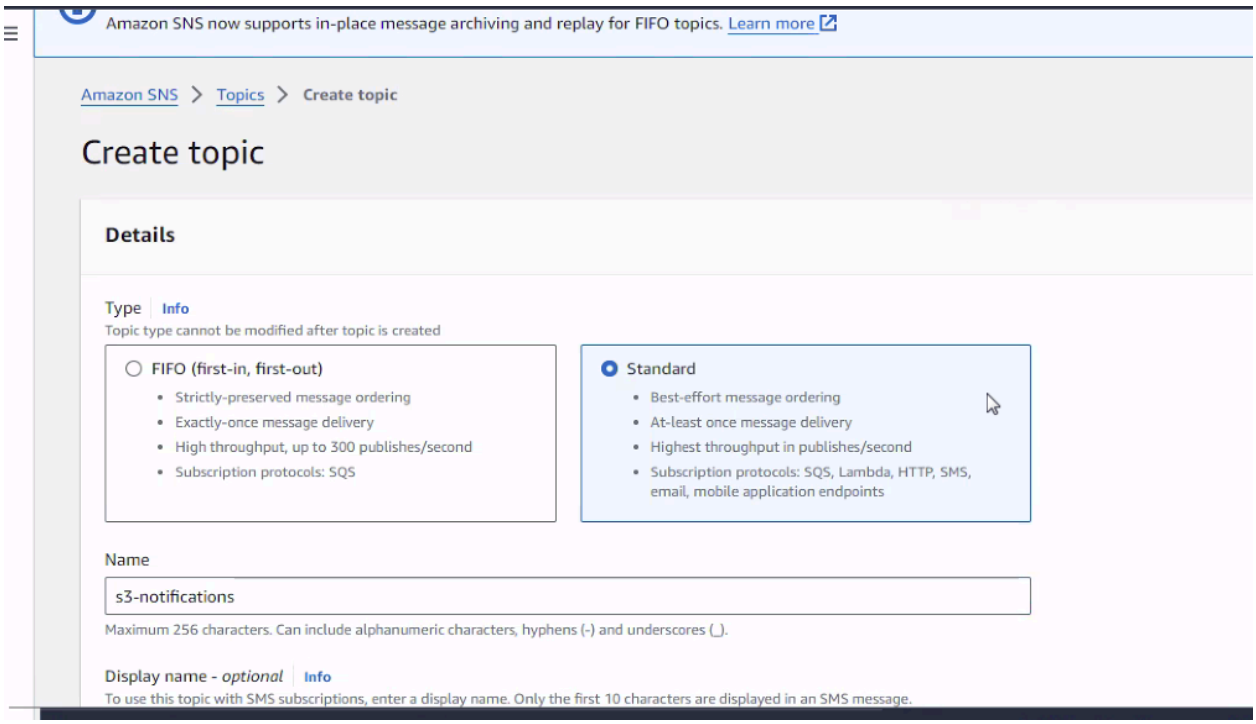
27. Copy that URL below 'bucket website endpoint'. Note: the bucket name will vary, it depends on what you named your bucket.
28. The link should take you to this:



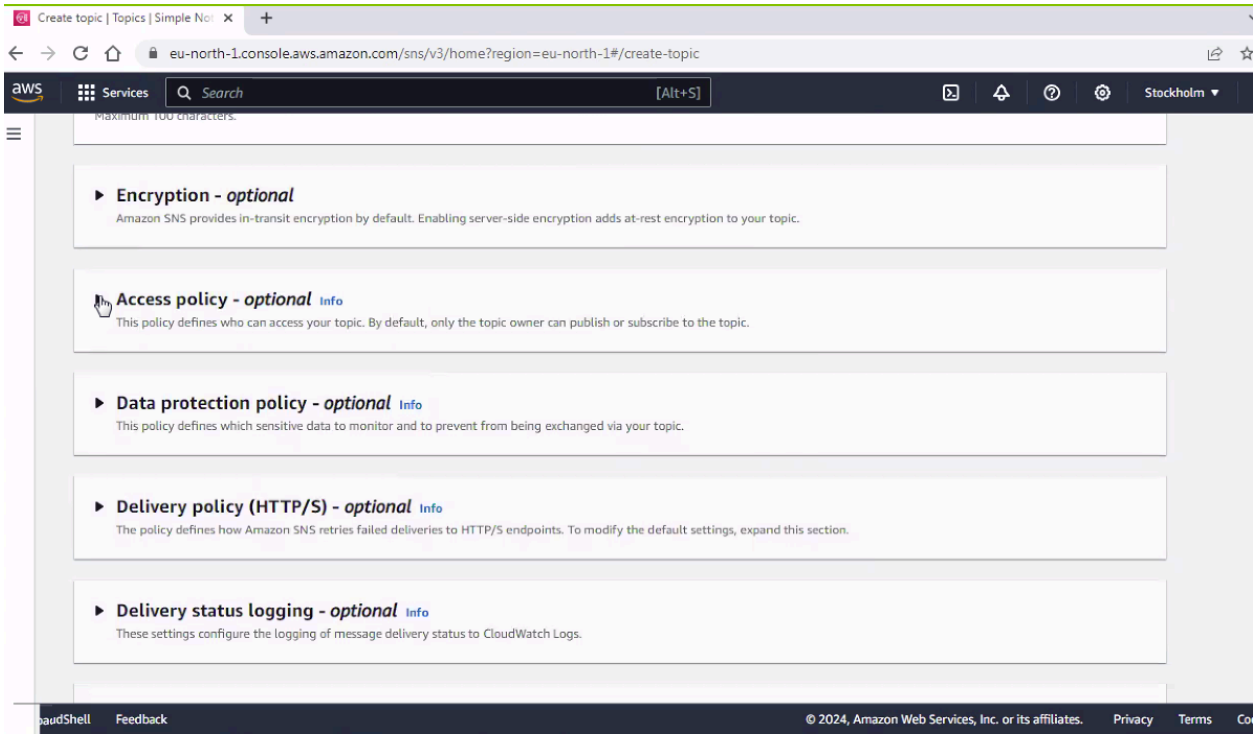
29. Finished Static Hosting part!

Part 2: SNS notifications

1. in search bar type “SNS” in the top of console
2. type in ‘create topic’ : “s3-notifications” and create it
3. The Name in our example is “ s3-notifications”, but you can name it whatever you want. Just make sure you remember



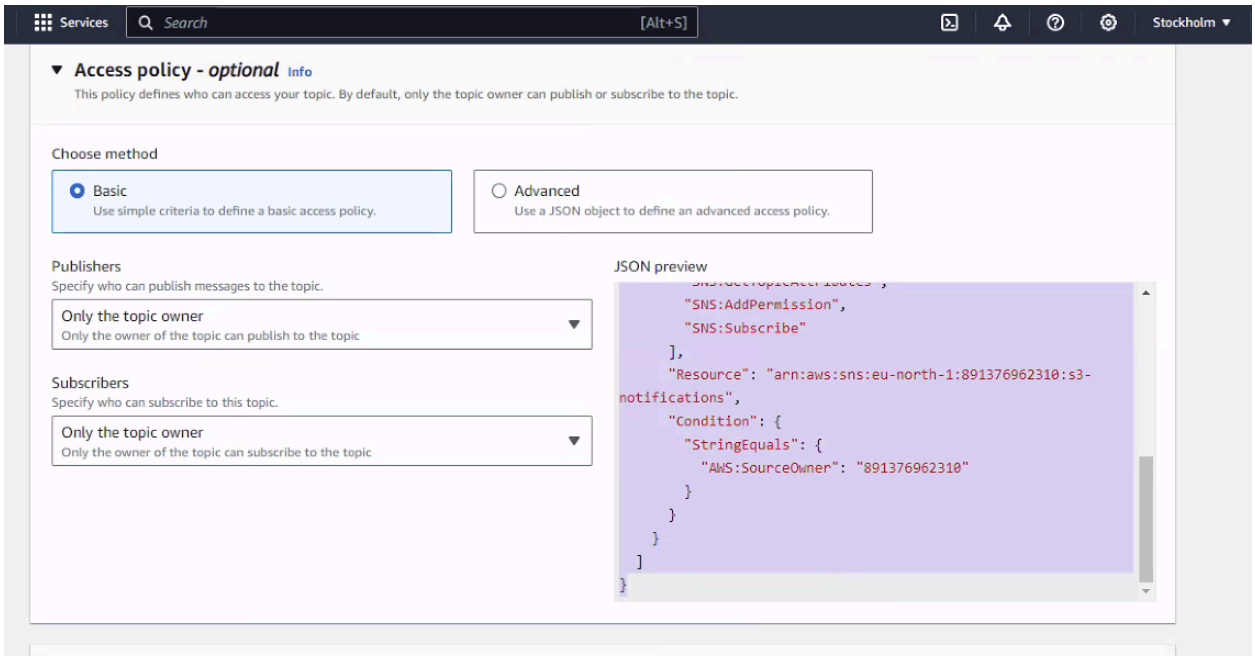
4. Click on Access Policy



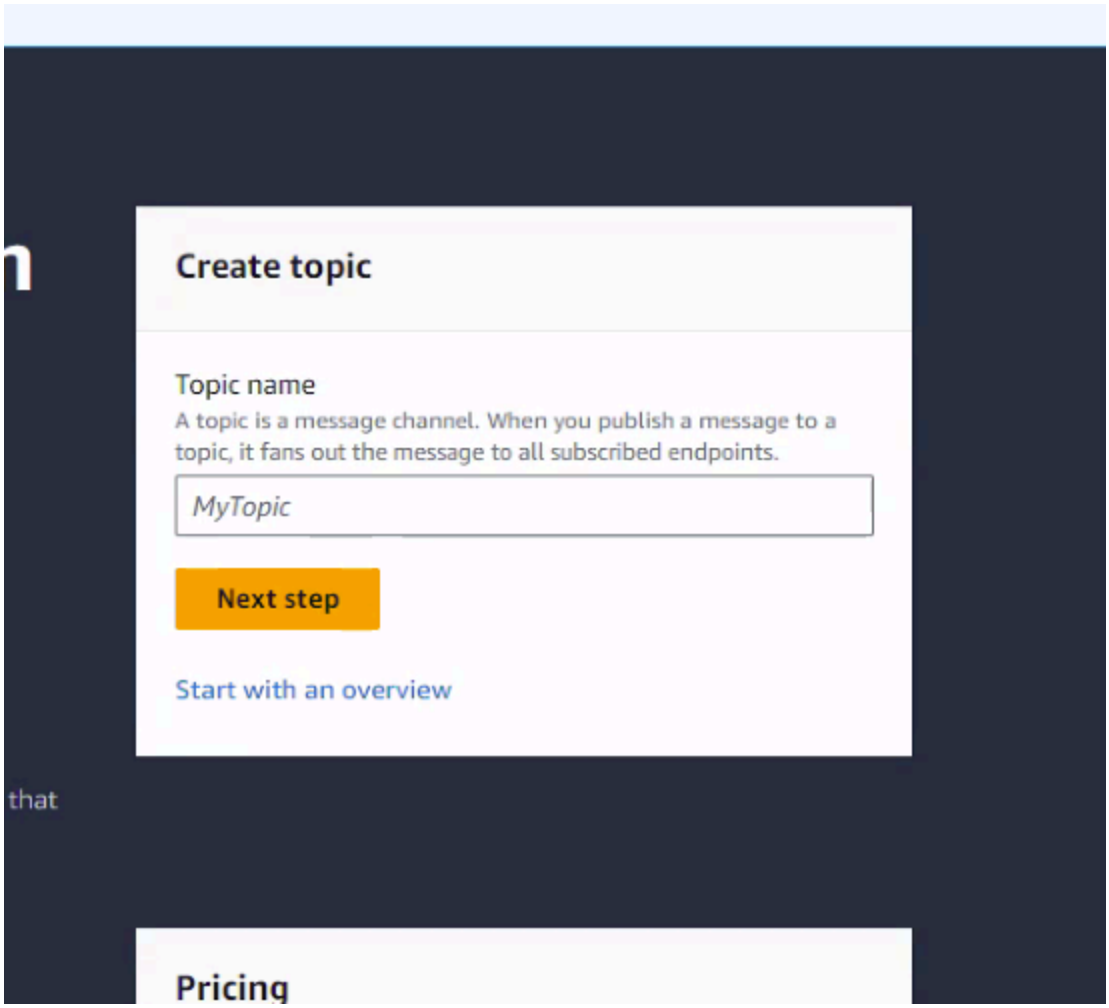
5. put in this code in JSON preview

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "S3_notification",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-1:339712730992:S3_notification",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:::registration-portal-bucket"
        }
      }
    }
  ]
}
```

6. It should look like this:

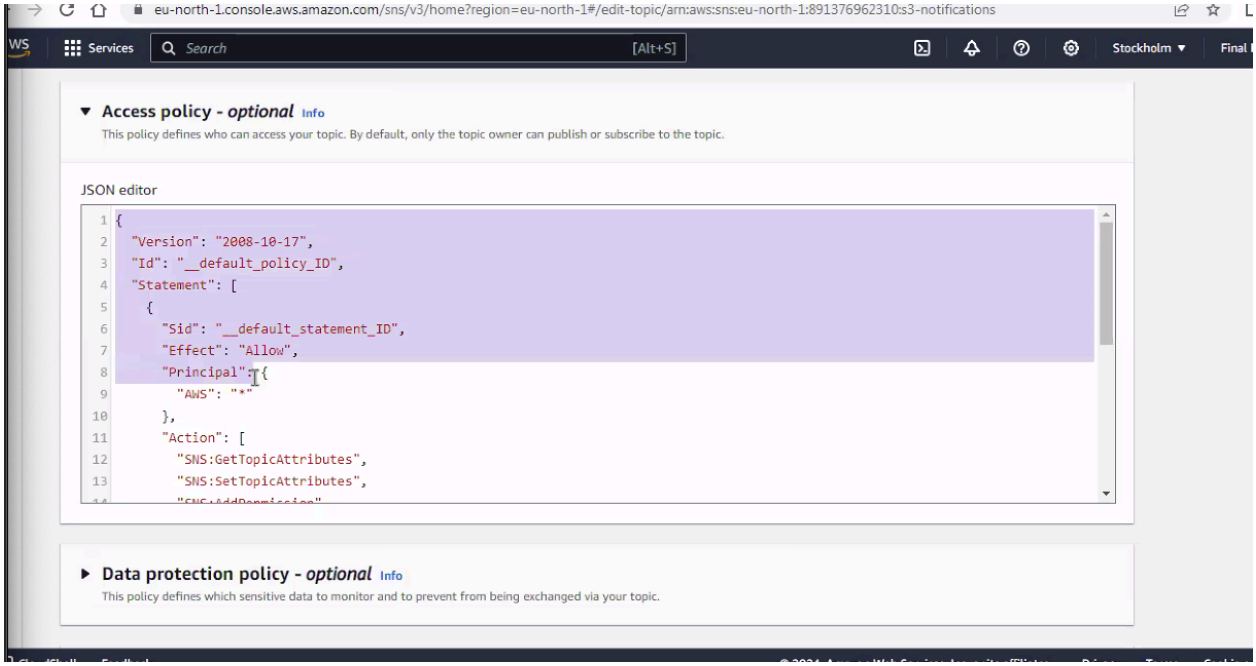


7. click ‘create topic’:



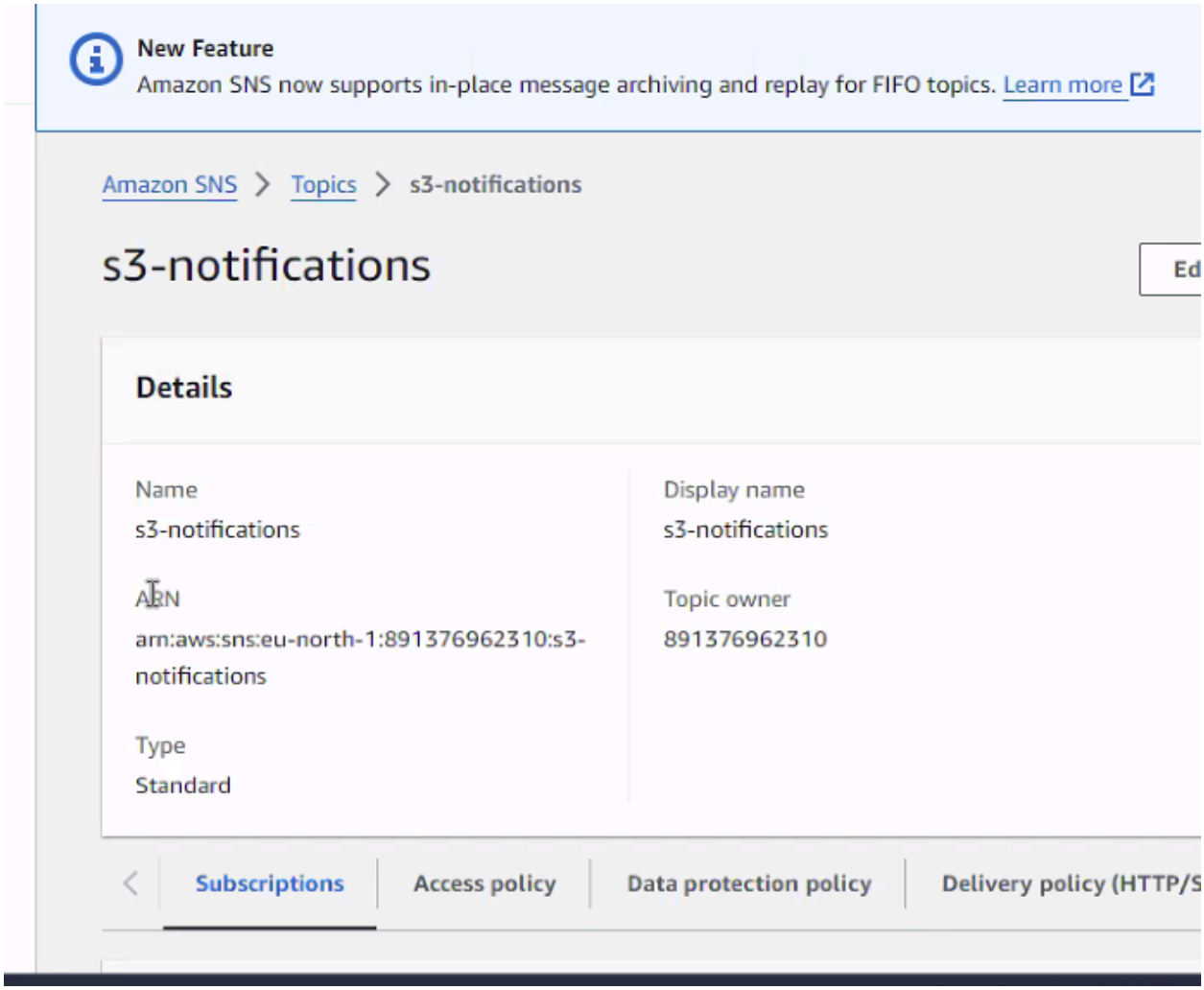
8. then click on ‘edit’ on that topic

9. insert that into the JSON editor in access policy:

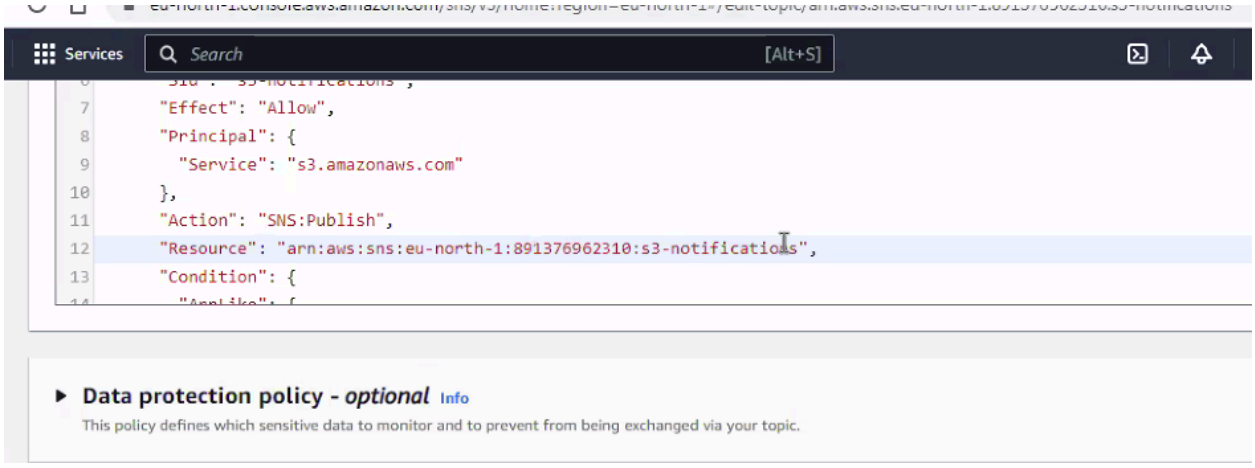


10. scroll up
11. make that change to the code. it needs to match exactly!
12. make sure Sid matches with the name of the Topic you created earlier:

13. go to SNS service.
14. click on Topic tab on the left hand screen
15. click on your topic (in our example 's3-notifications'_
16. copy that ARN:

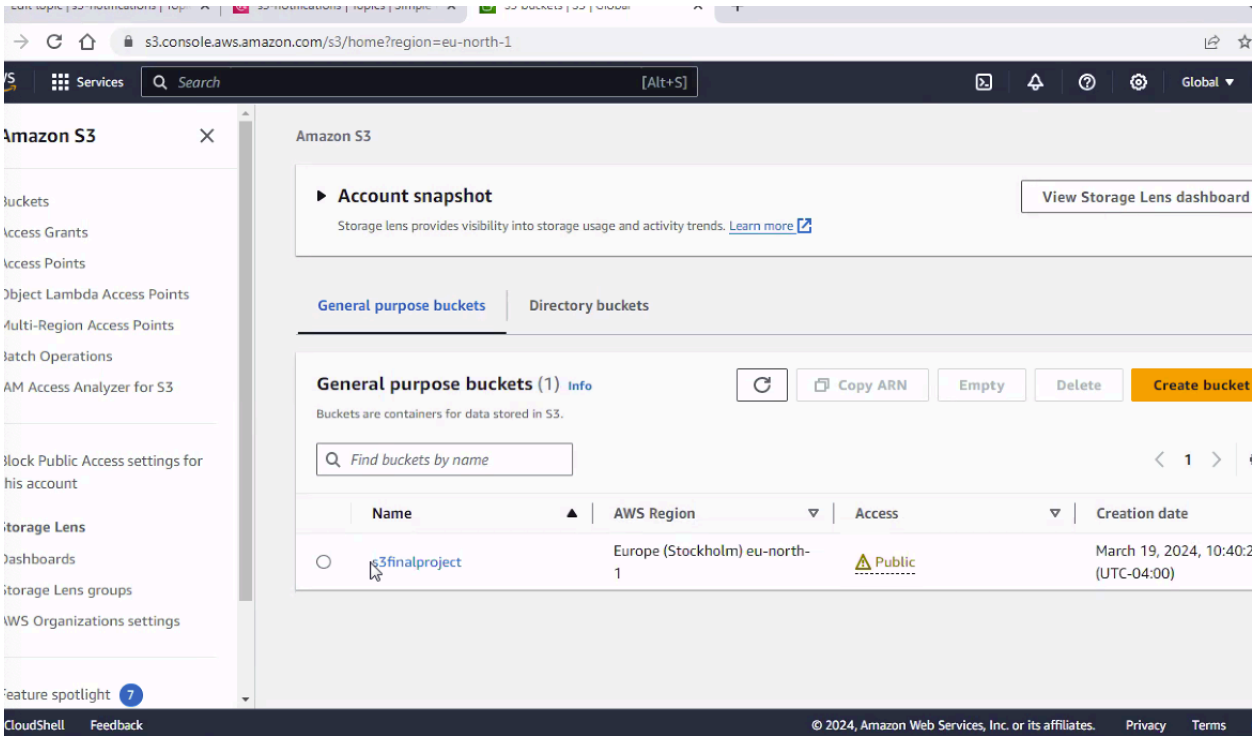
17. 

18. then go back to that JSON editor and add that ARN to the ‘resource’ section of code:

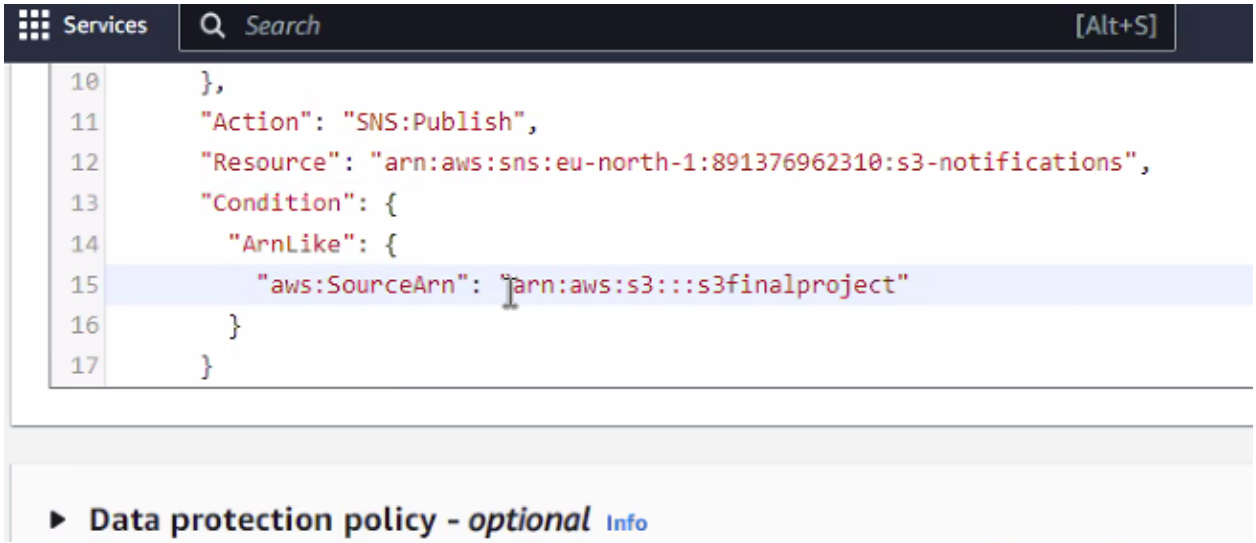


19. go to s3

20. Click on your bucket:

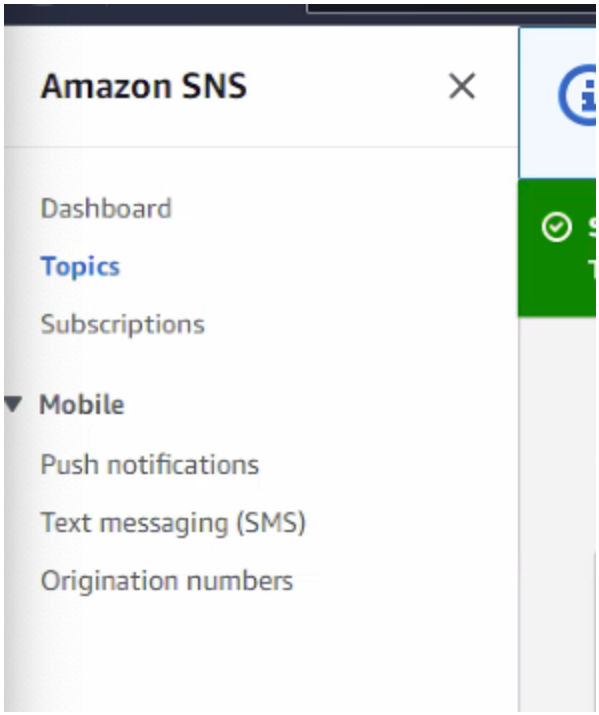


21. copy the name of that bucket exactly
22. then go back to SNS service
23. then in JSON editor, paste that bucket name in SourceARN:

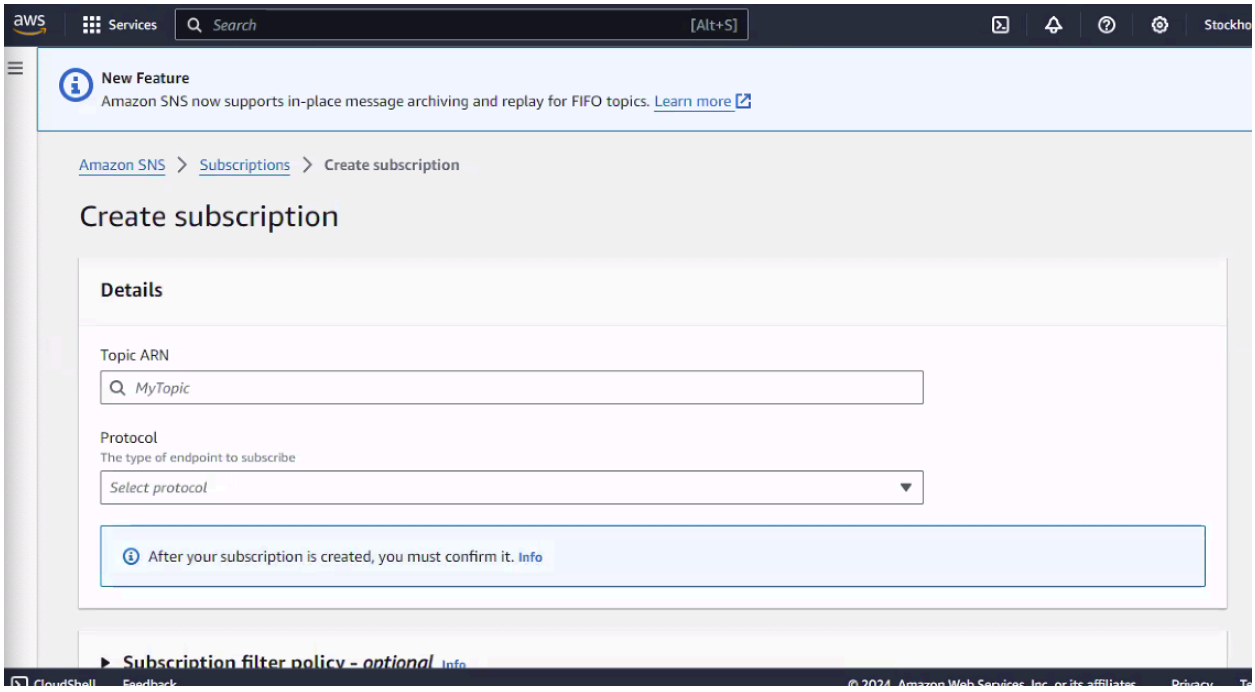


24. then go down to the page and save changes

25. then click on subscriptions on the left hand side (below Topics):

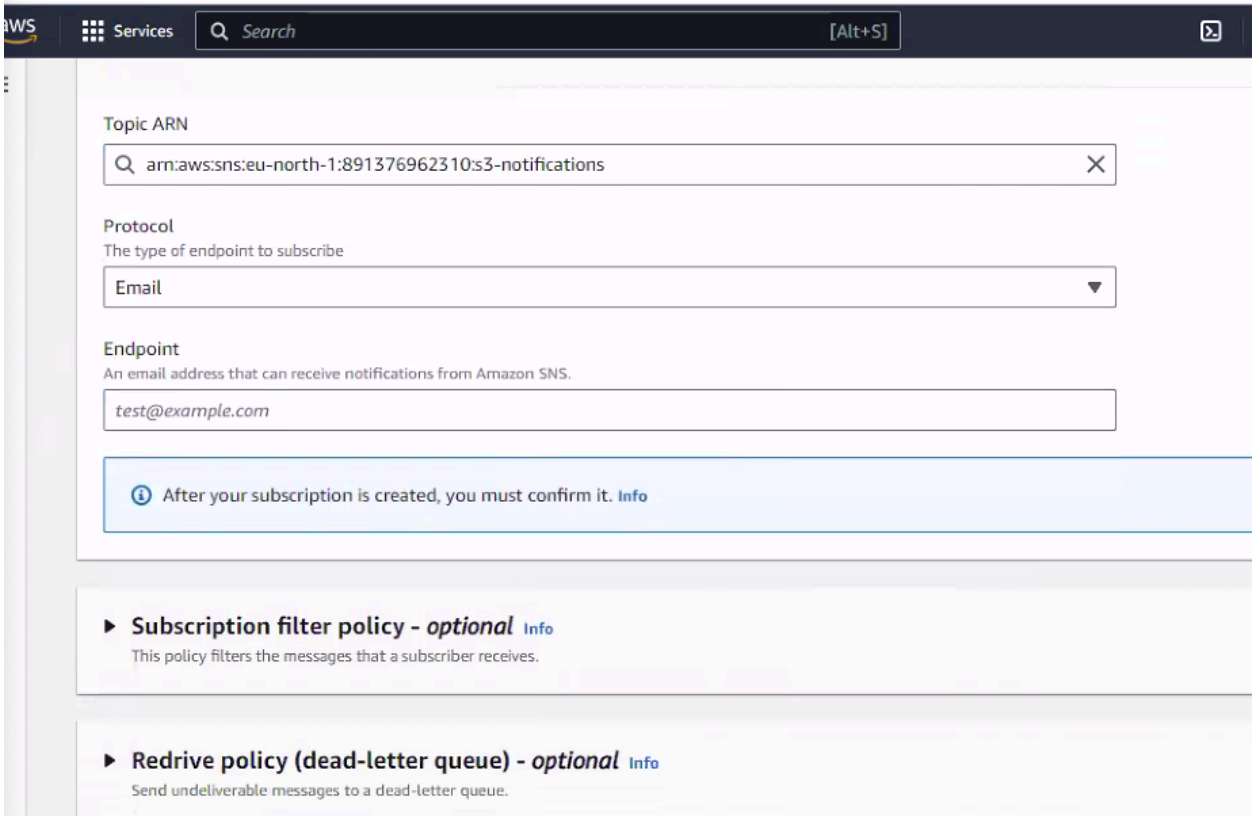


26. Create one:

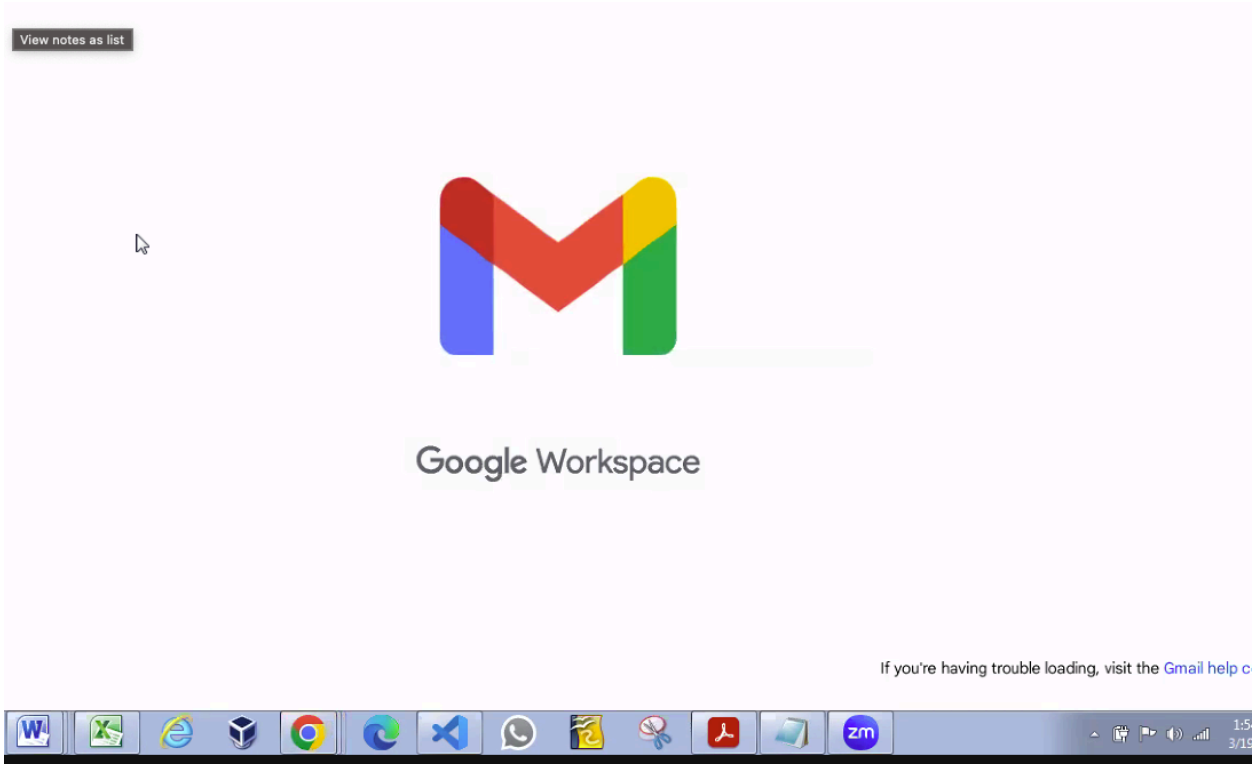


27. 'Topic ARN' should be there in the drop-down menu
28. 'Protocol' should be 'email' in the selection drop-down

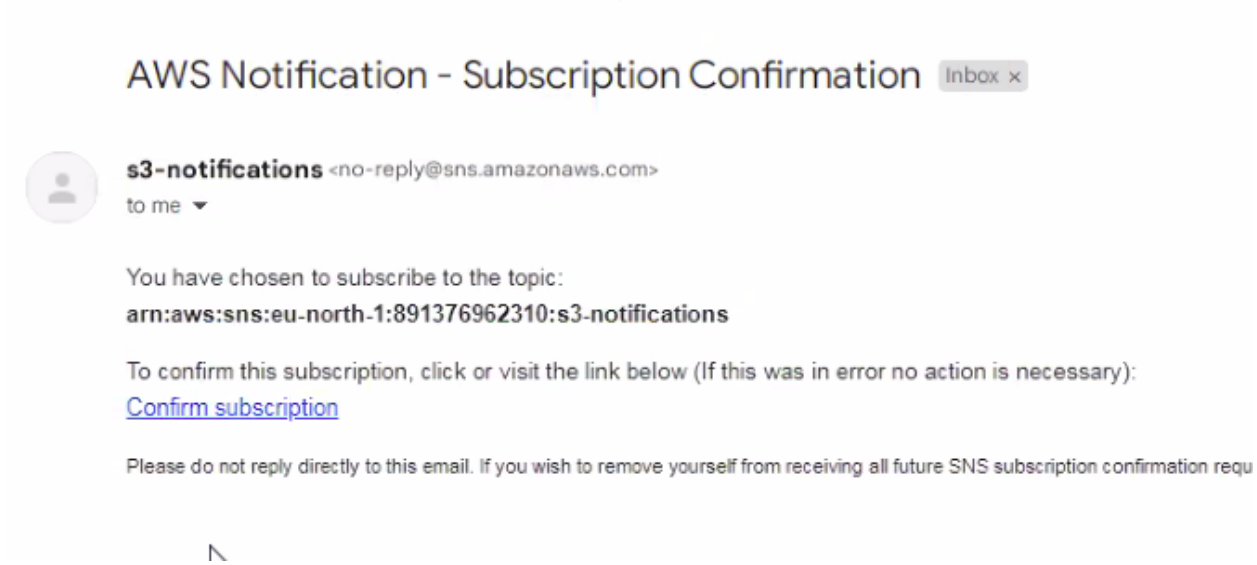
29. For ‘Endpoint’, let it be your own personal email to receive notification:



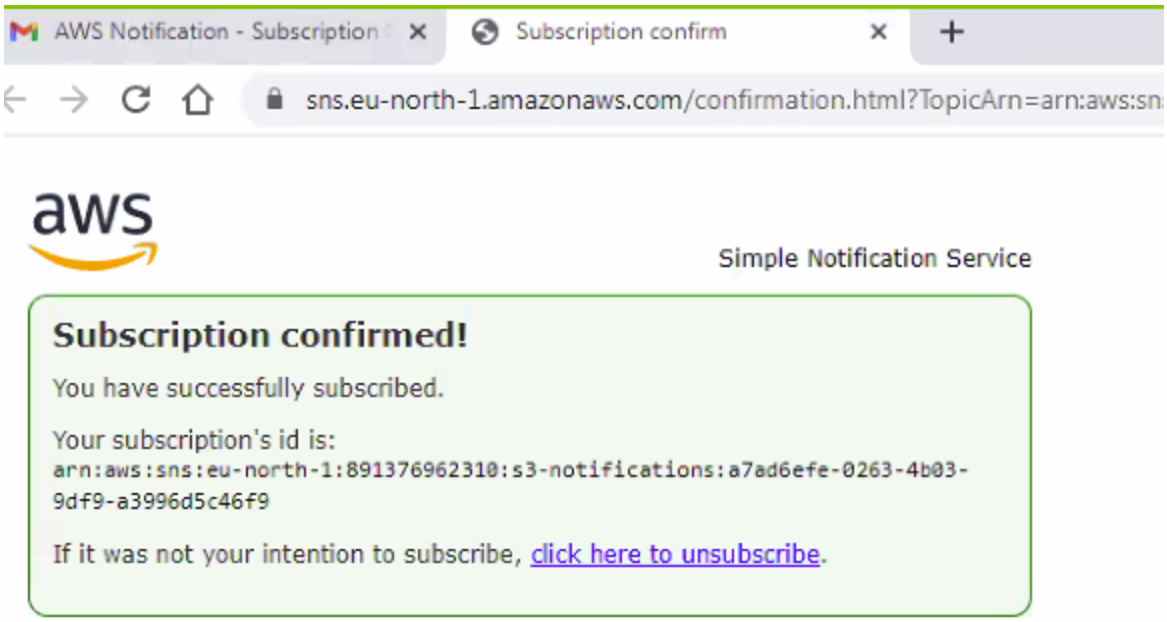
30. then create the subscription
31. Go to your email e.g. :



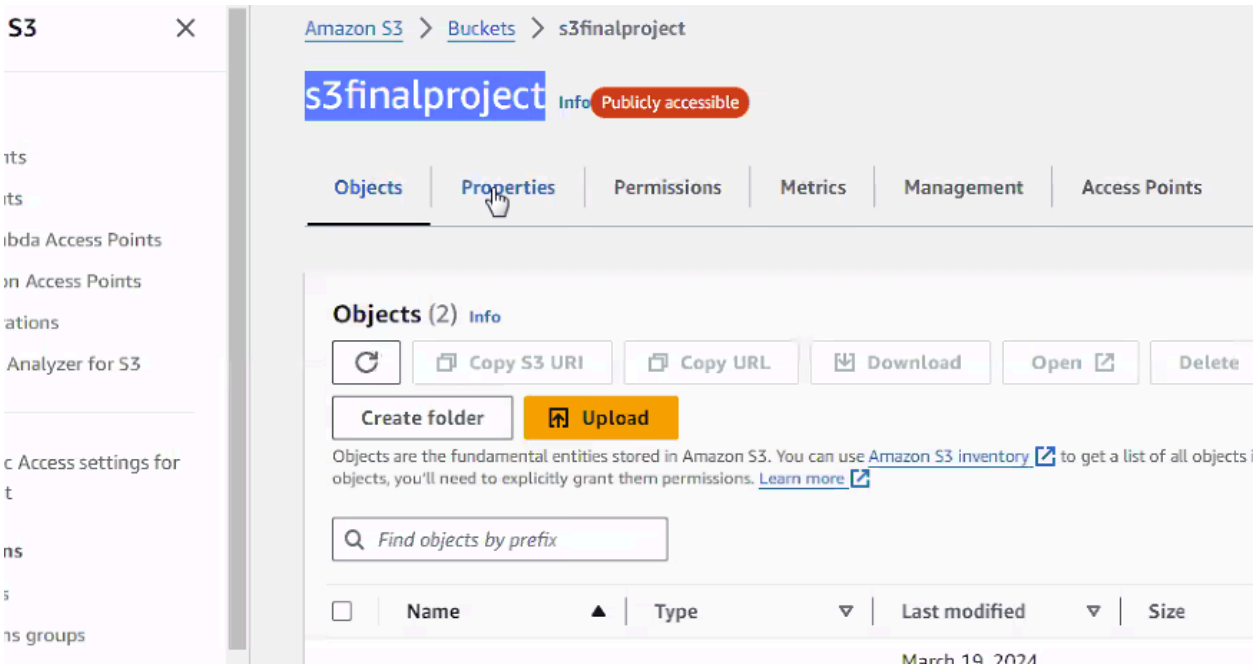
32. Go to your your personal email and click on the email from s3 notifications:



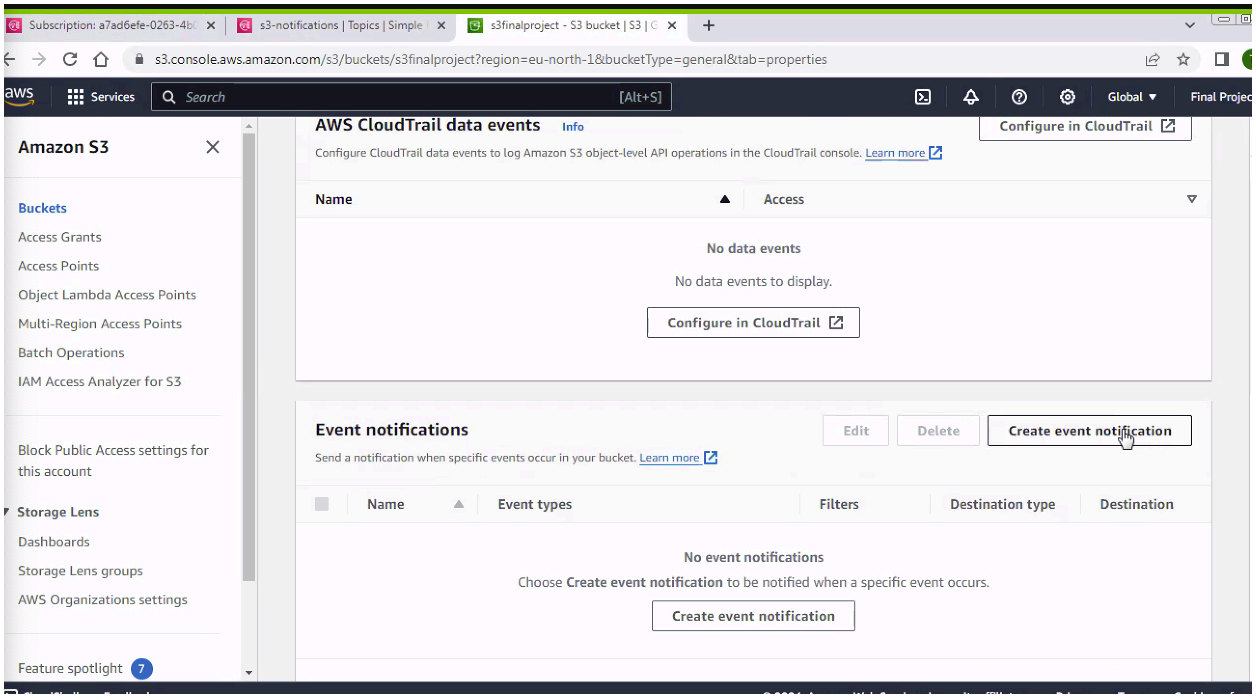
33. Click on the link to 'confirm subscription'



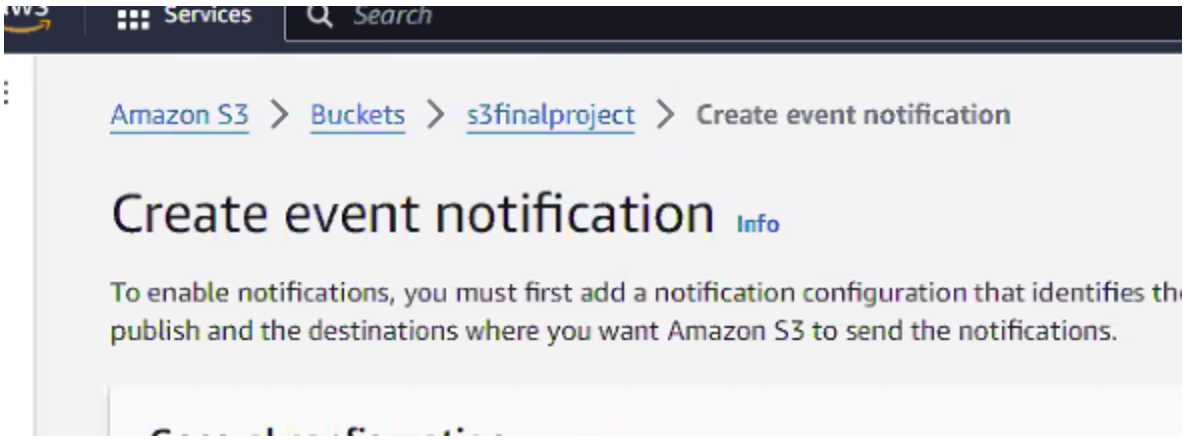
34. Go back to your console:



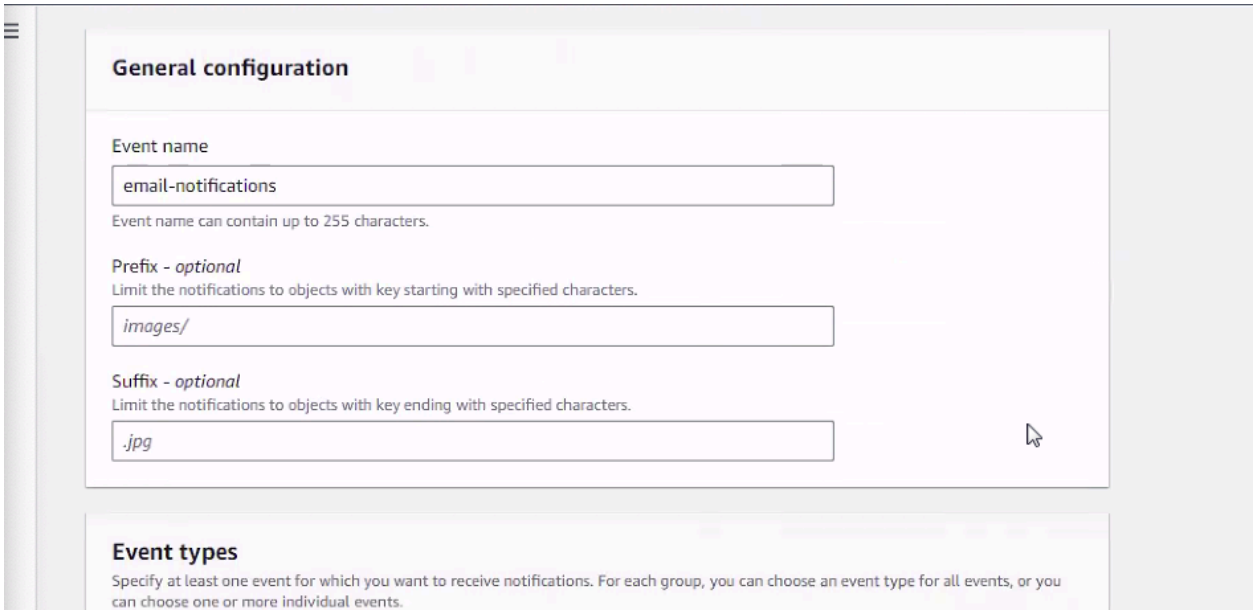
35. click on properties tab
36. Create event notifications button:



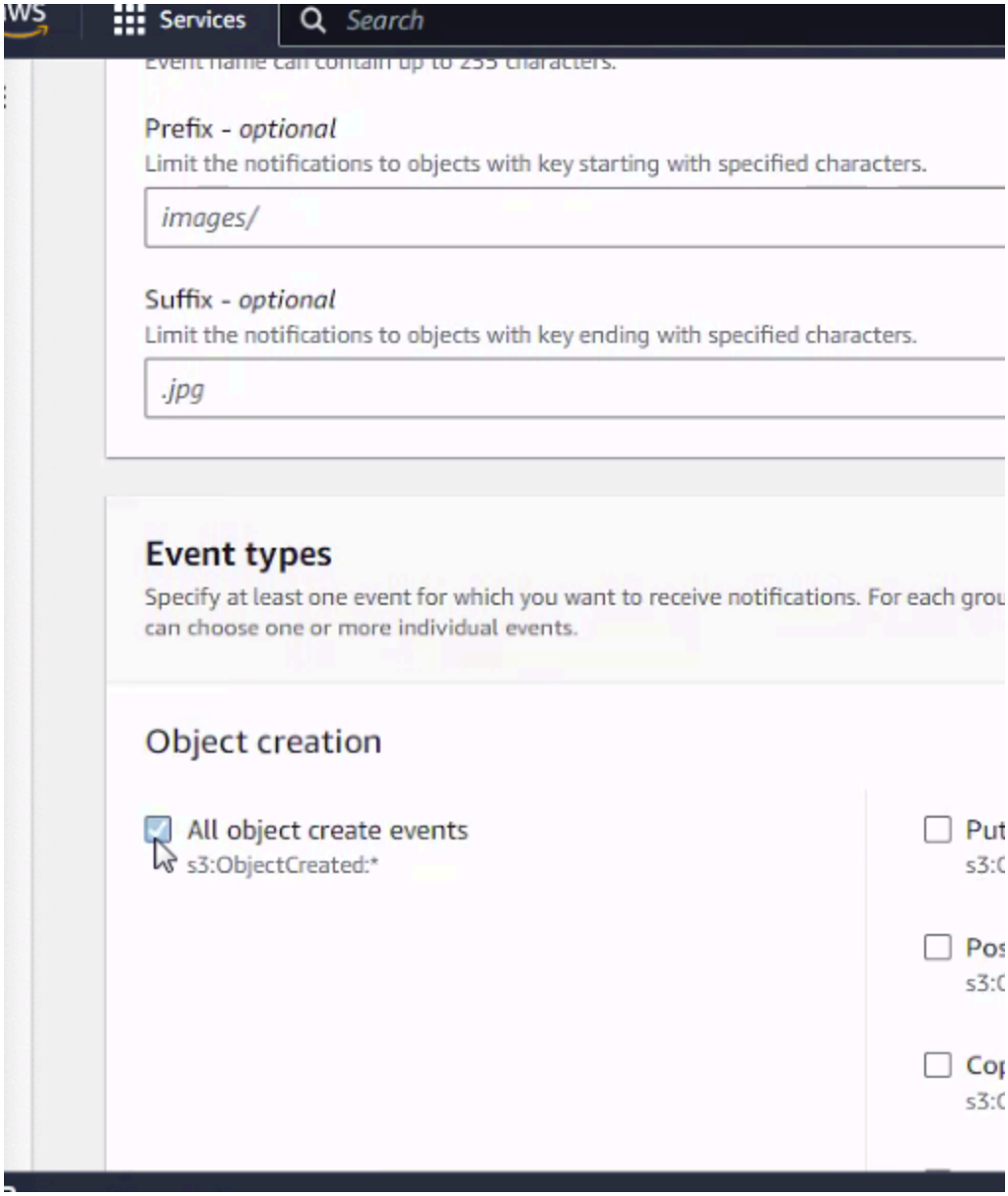
37. It should look like this:



38. and name it something, in this case its “email notifications”:

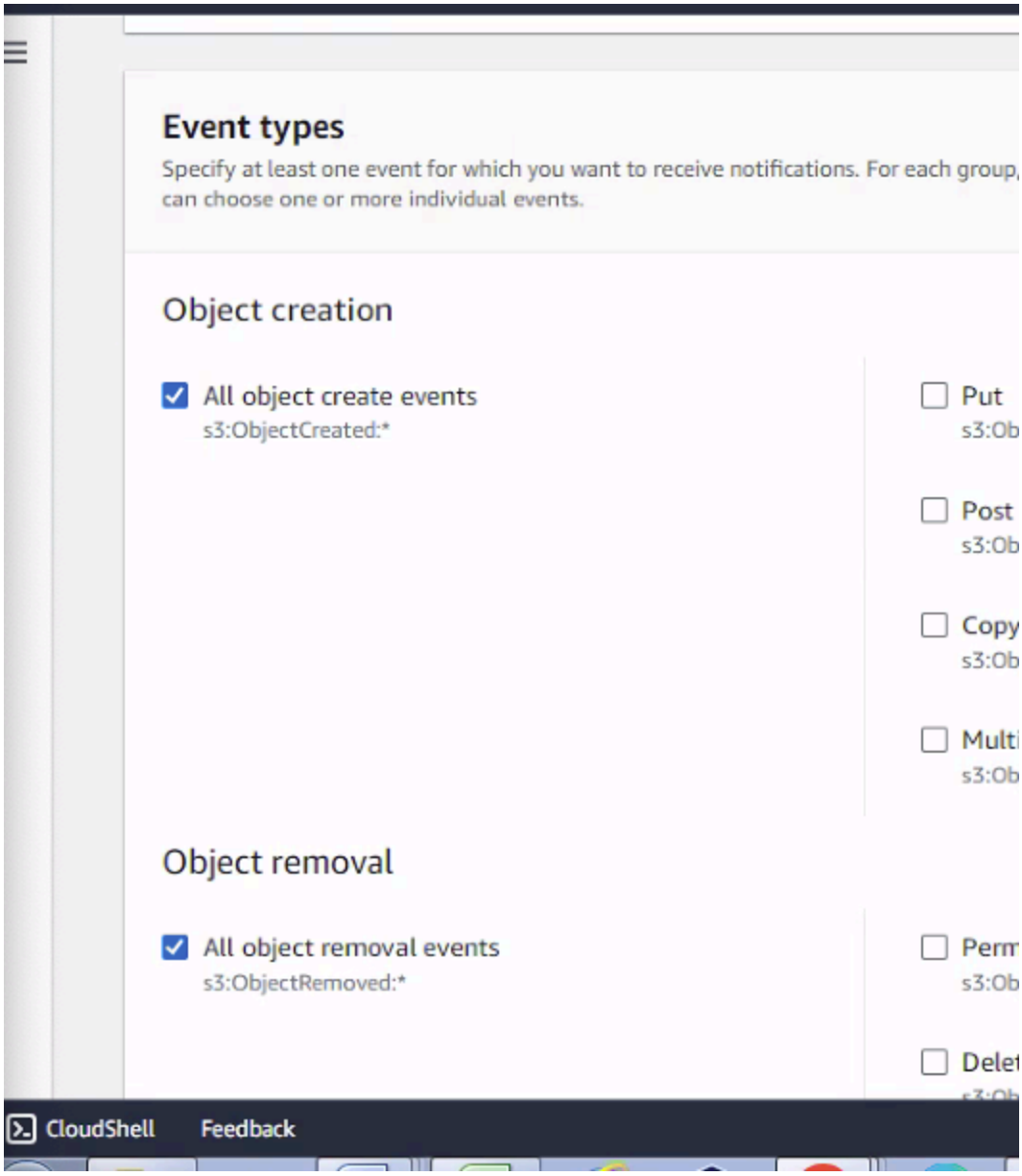


39. scroll down to “event types”:

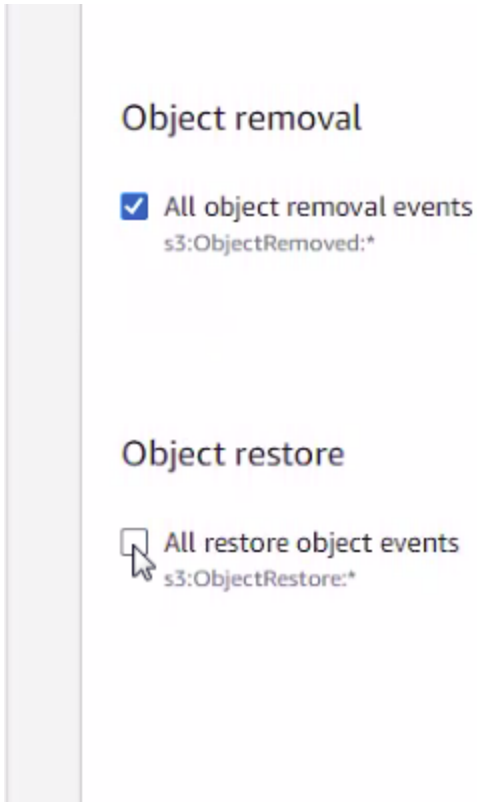


40. on object creation part
41. click on ‘all object create events’

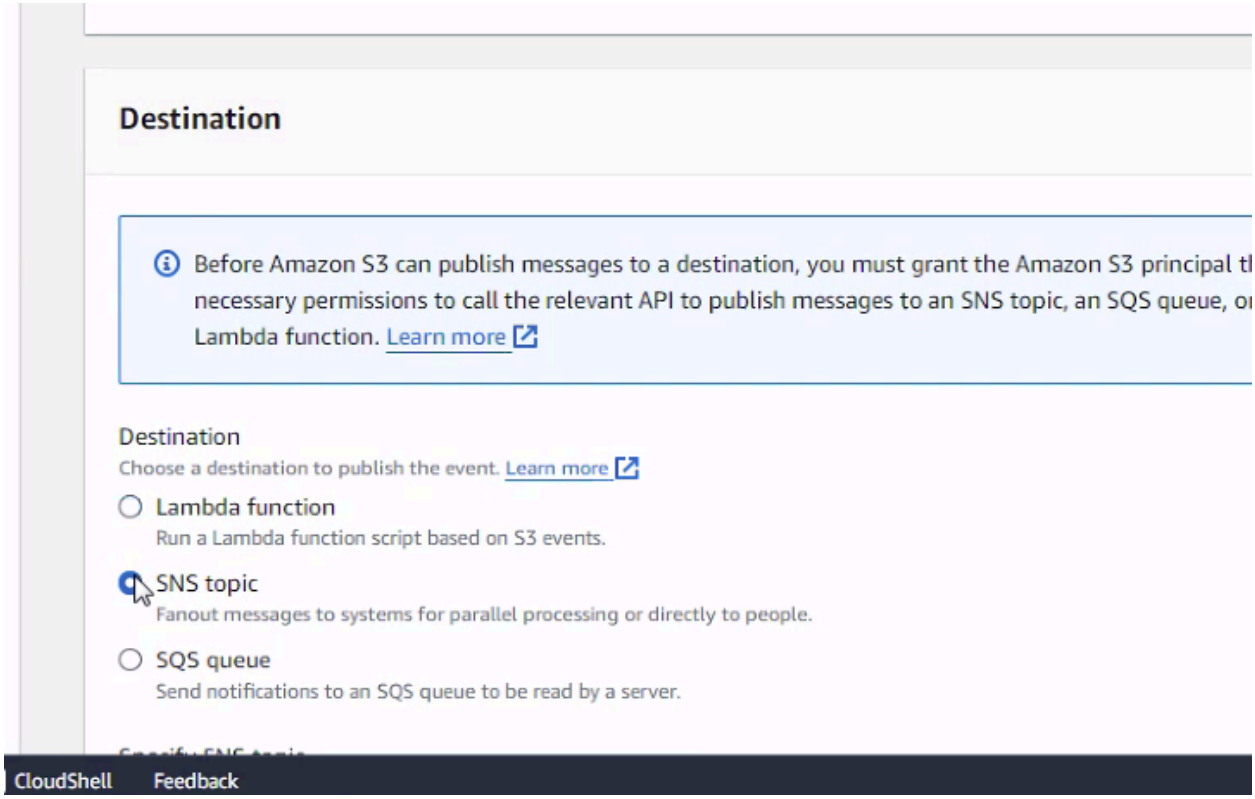
42. click on that object removal:



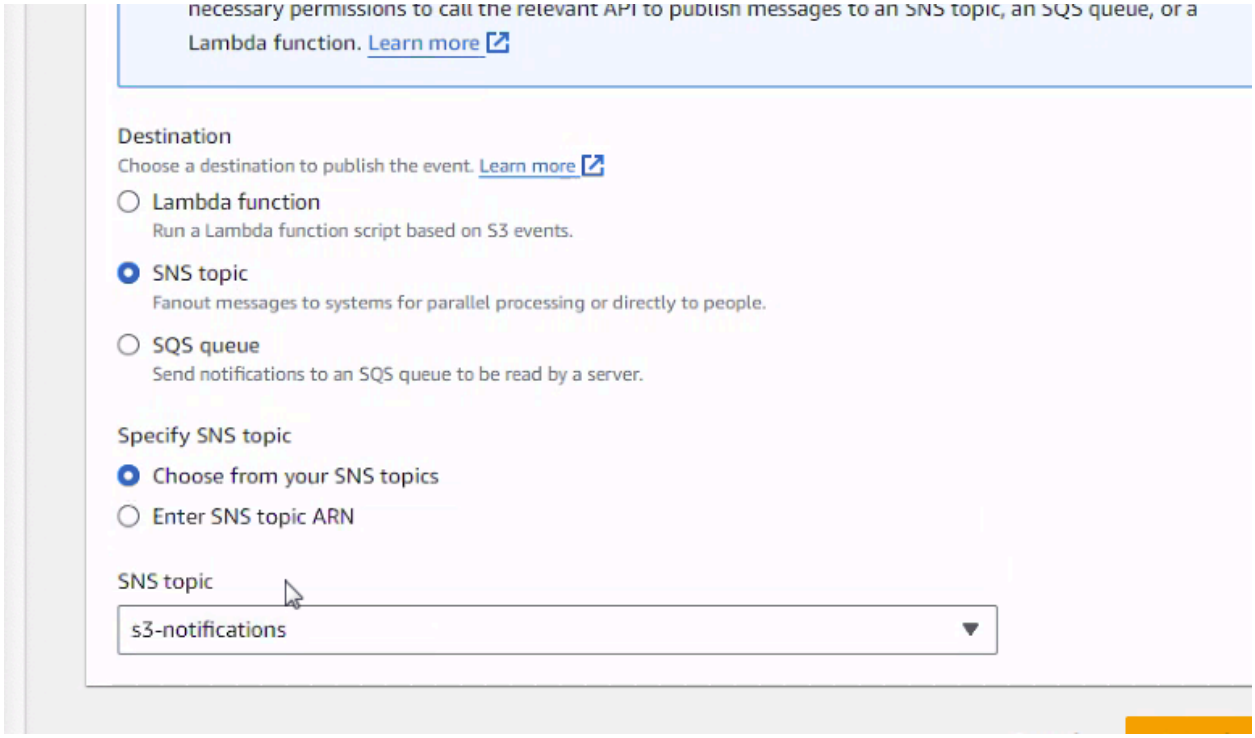
43. Click on ‘object restore’ checkbox too :



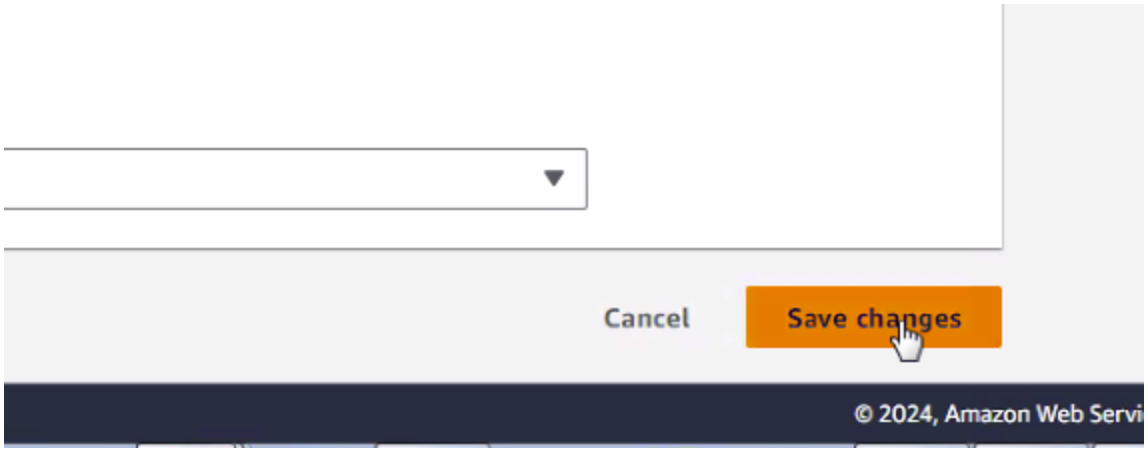
44. Click on ‘SNS Topic’



45. In the ‘SNS Topic’ dropdown menu, choose your saved topic e.g. “s3-notifications”:

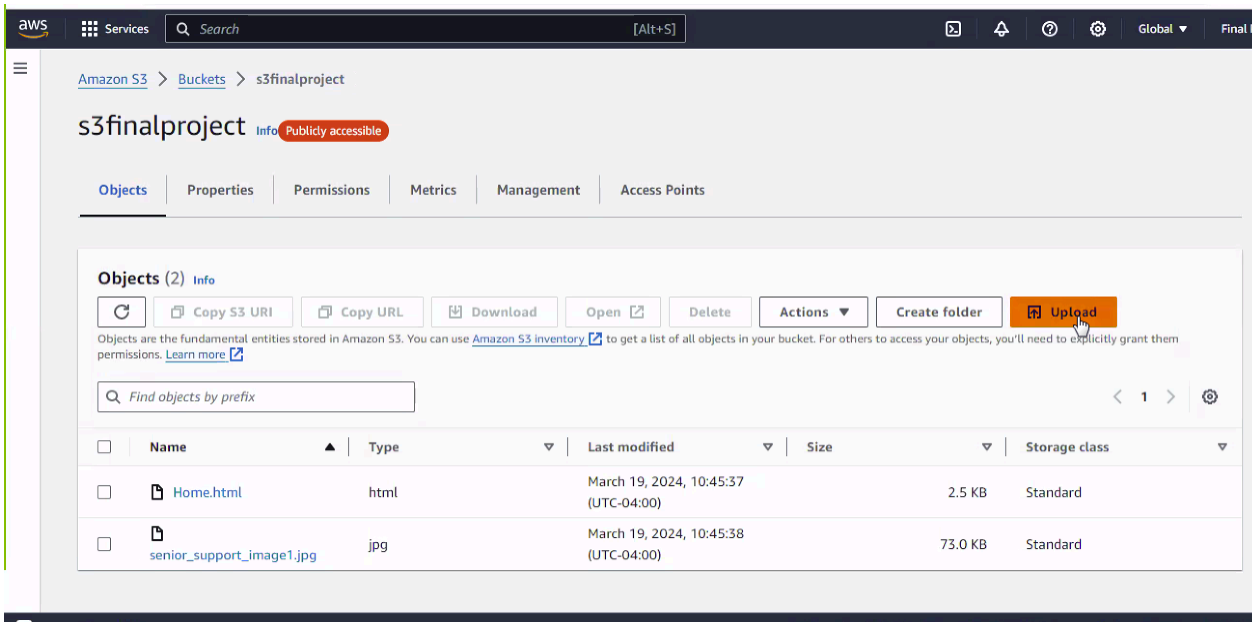


46. Save changes

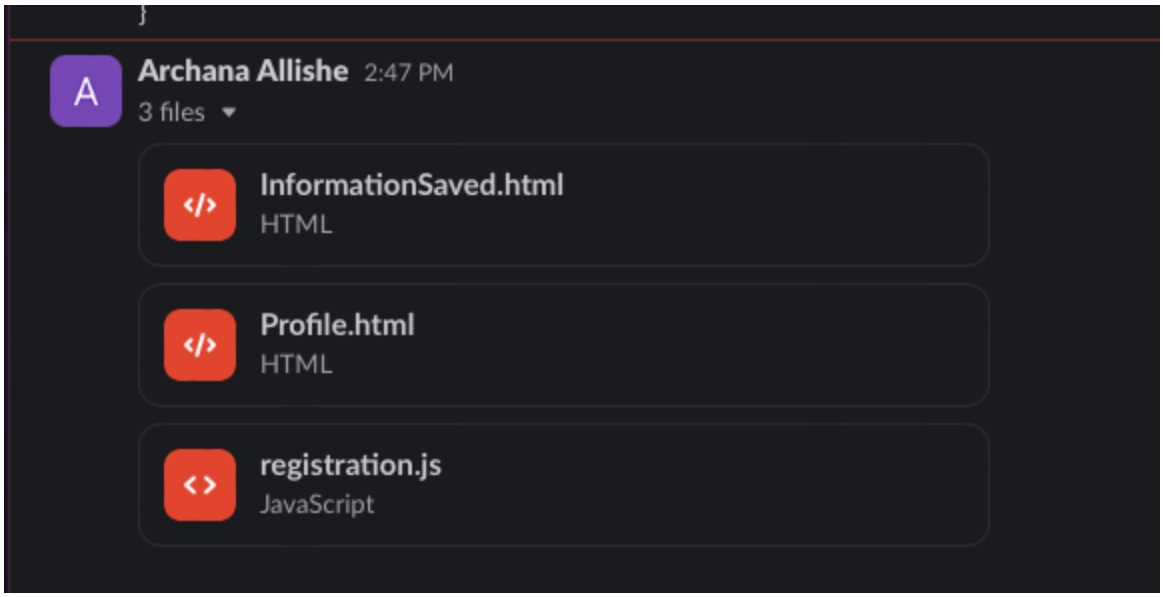


47. Go to your S3 page
48. Go to your S3 bucket (e.g. s3finalproject)
49. Click on ‘Objects’ tab

50. Click on ‘Upload’ button on the far right:



51. put these 3 files in the s3 bucket (use Visual Code Studio to view their code easier)



InformationSaved.html: (be sure you copy the whole code)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Registration Submitted</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
        }
        .message-container {
            border: 1px solid #ccc;
            padding: 20px;
            max-width: 600px;
            margin: 0 auto;
            text-align: center;
        }
        h1 {
            margin-bottom: 20px;
        }
    </style>
</head>
<body>

<div class="message-container">
    <h1>Your Registration Form is Submitted</h1>
    <p>We will contact you soon!</p>
    <a href="Home.html">Go back to Home</a>
</div>

</body>
</html>
```

52. Profile.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <style>
        body {
```

```
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            text-align: center;
            color:#01377D;
        }
        label {
            display: block;
            margin-bottom: 5px;
            color:#01377D;
        }
        input[type="text"] {
            width: 100%;
            padding: 8px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
            box-sizing: border-box;
        }
        input[type="submit"] {
            width: 100%;
            padding: 10px;
```

```
            background-color: #01377D;
            color: #fff;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
        }
        input[type="submit"]:hover {
            background-color:#01377D;
        }
        #DataSaved {
            text-align: center;
            margin-top: 20px;
            color: green;
            font-weight: bold;
        }

    </style>
</head>
<body>
    <div class="container">
        <h1>Volunteer Registration Form</h1>
        <form id="registrationForm">
            <label for="useremail">User email:</label>
            <input type="text" name="useremail" id="useremail" required>

            <label for="firstname">First name:</label>
            <input type="text" name="firstname" id="firstname" required>

            <label for="lastname">Last name:</label>
            <input type="text" name="lastname" id="lastname" required>

            <label for="usergender">Gender:</label>
```

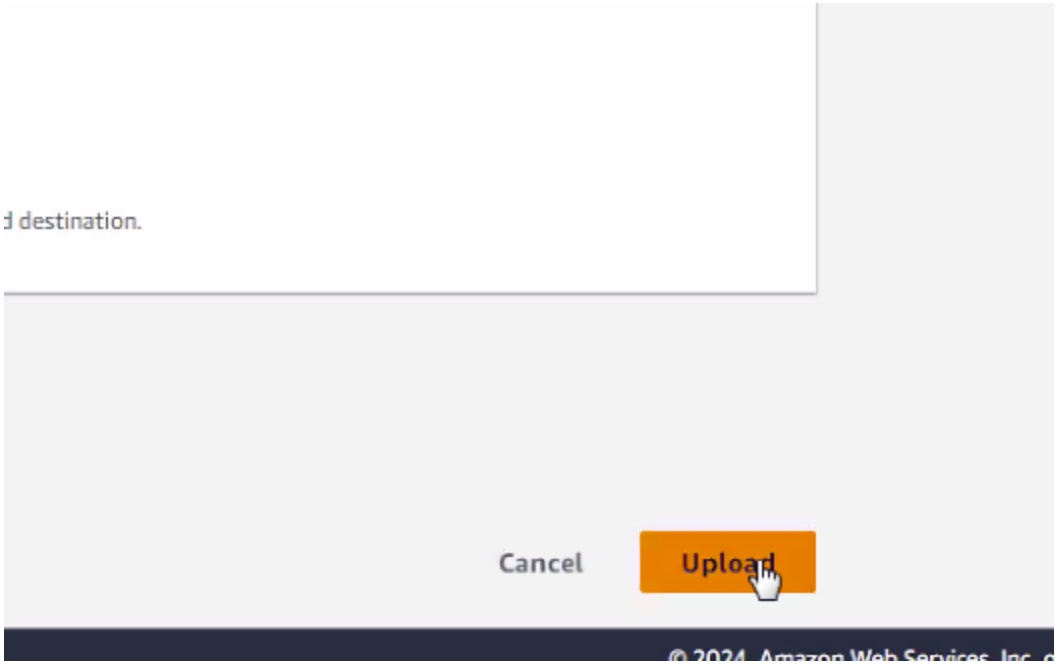
```
            <input type="text" name="usergender" id="usergender" required>

            <label for="userage">User Age:</label>
            <input type="text" name="userage" id="userage" required>

            <input type="submit" id="Submit" value="Save Profile">
        </form>
        <p id="DataSaved"></p>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="registration.js"></script>
</body>
</html>
```

53. Registration.js:

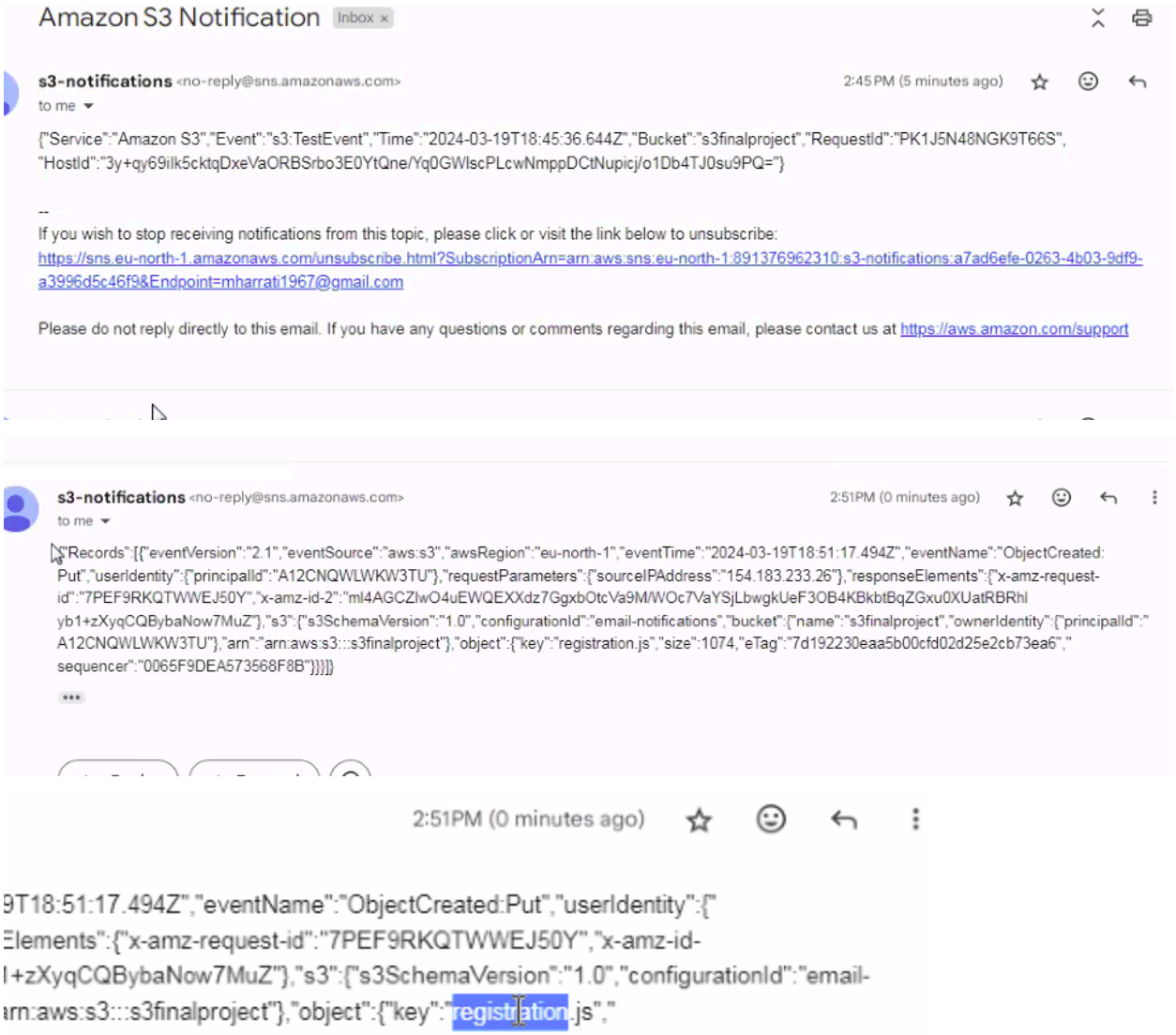
```
var API_ENDPOINT = "https://7qmc7yvqc1.execute-api.us-east-1.amazonaws.com/prod";

$(document).ready(function() {
    $('#Submit').click(function(event) {
        event.preventDefault(); // Prevent default form submission behavior

        var inputData = {
            "userEmail": $('#useremail').val(),
            "userFirstName": $('#firstname').val(),
            "userLastName": $('#lastname').val(),
            "userGender": $('#usergender').val(),
            "userAge": $('#userage').val()
        };

        $.ajax({
            url: API_ENDPOINT,
            type: 'POST',
            data: JSON.stringify(inputData),
            contentType: 'application/json; charset=utf-8',
```

```
            success: function(response) {
                $('#DataSaved').text('Data Submitted');
                window.location.href = "InformationSaved.html"; // Redirect to the next page
            },
            error: function() {
                alert("Error occurred while submitting data.");
            }
        });
    });
});
```

54. Upload it :



55. Go check your email for the notification:

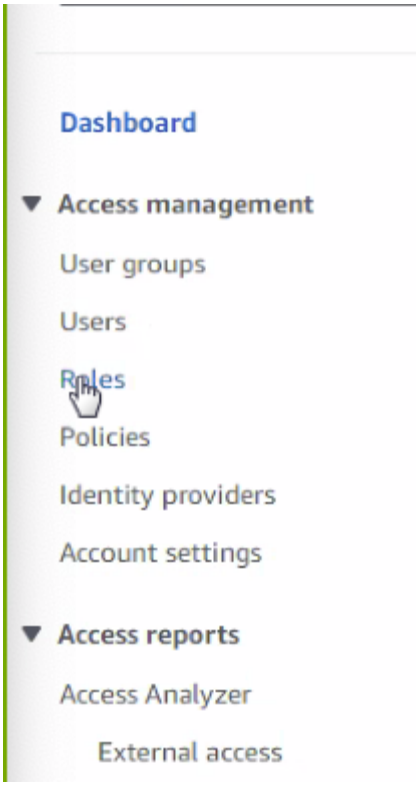


56. this shows that the files have been uploaded and you received an email notification “registration.js” is the name of the file that it uploaded to s3. and SNS notified us. it means it has the data and it is taking the front end info from the text boxes and connect it to the API

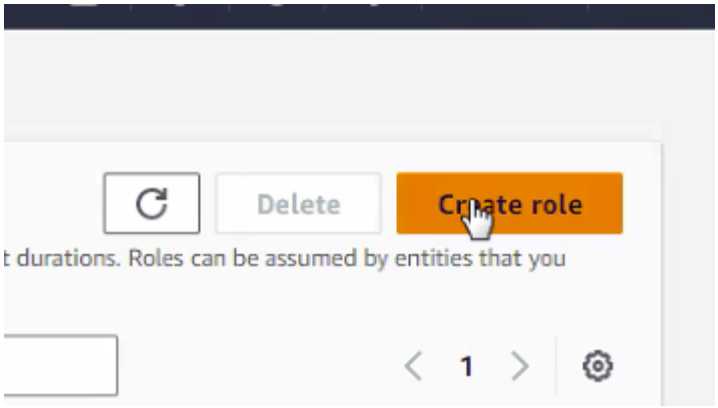
Part 3: IAM setup

1. Go to IAM on your console:

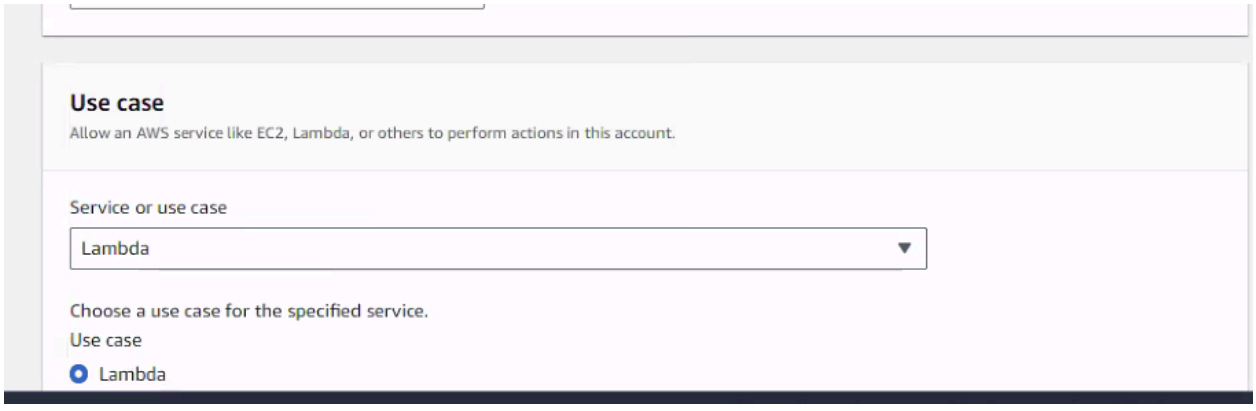
2. Go to ‘Roles’ on the left hand side of the console:

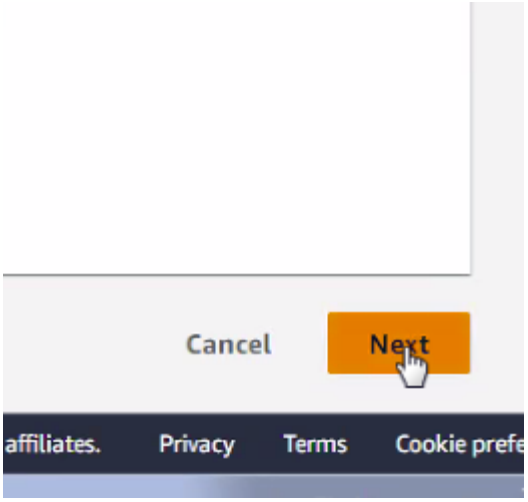


3. Create Role:

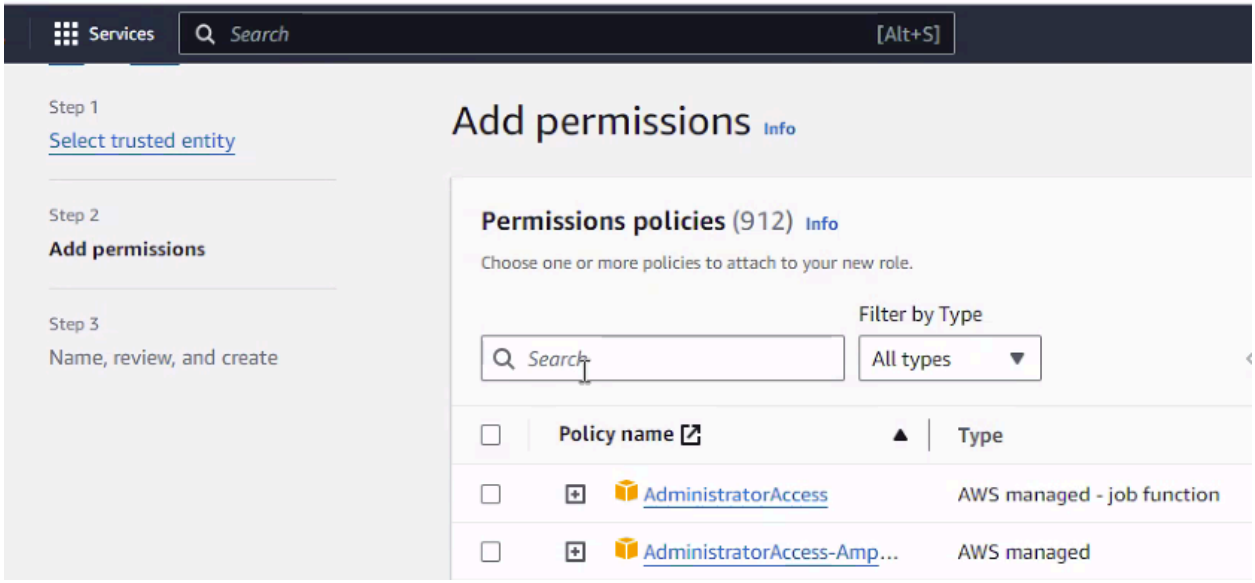


4. For ‘Use Case’ select Lambda:

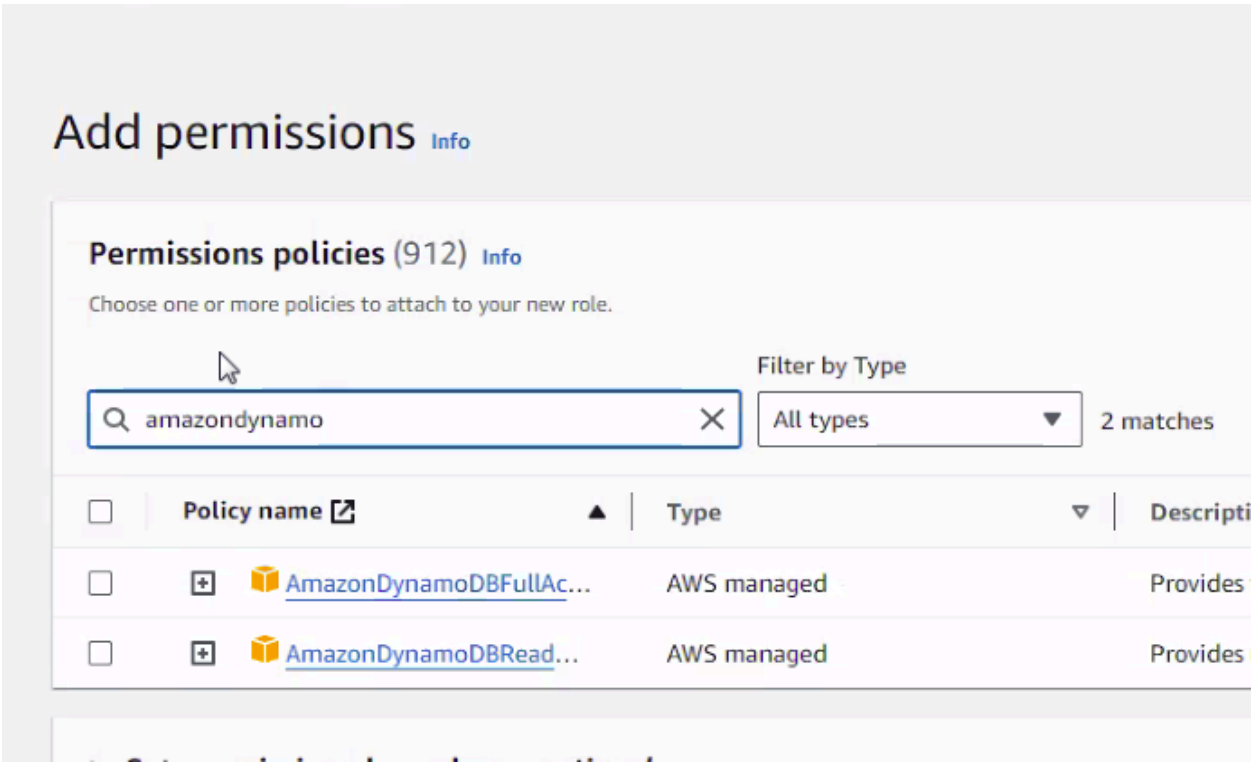




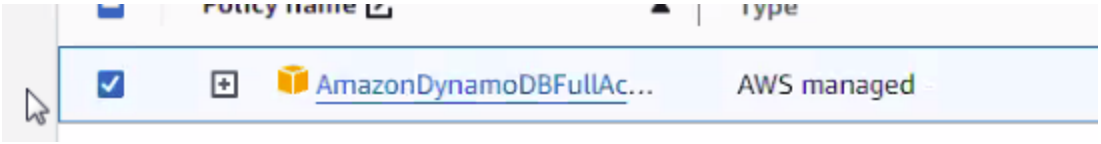
5. Go to Next:
6. Add permissions
7. In the Search box click on it:



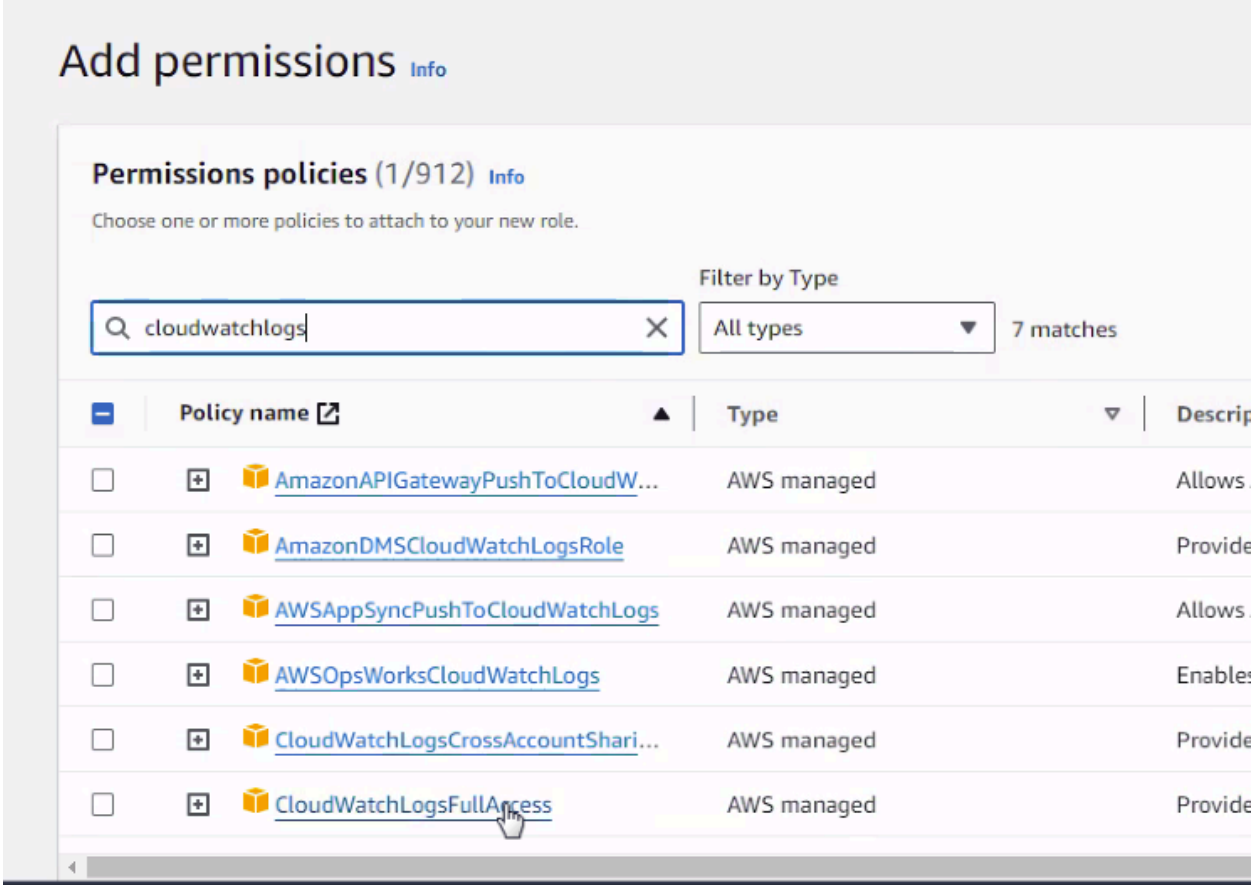
8. Type in “AmazonDynamoDB”:



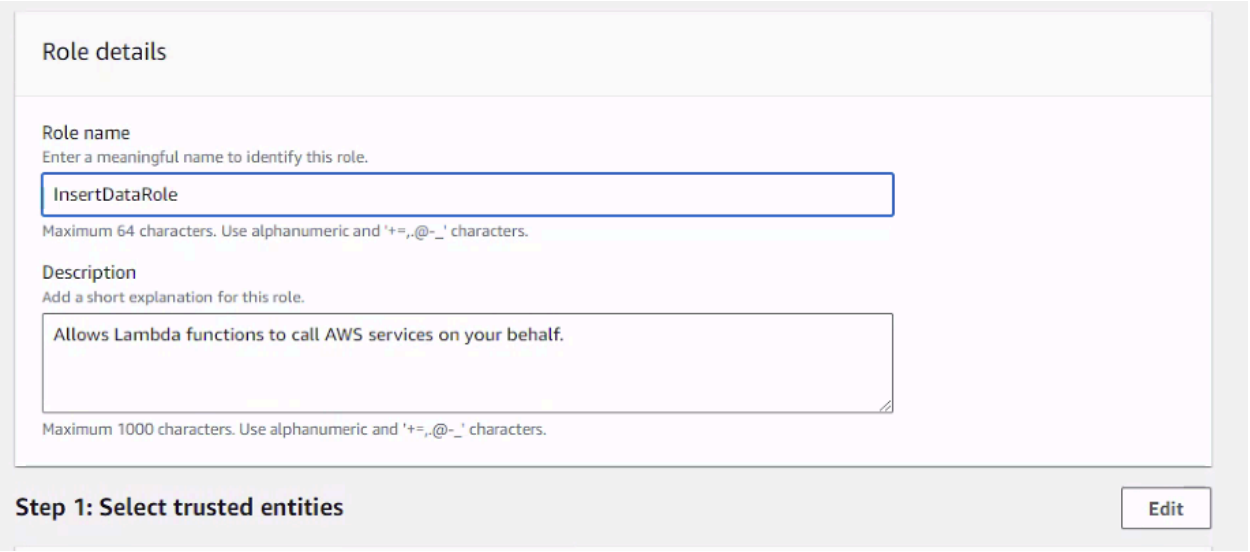
9. Then select the one that has “AmazonDynamoDBFullAccess:



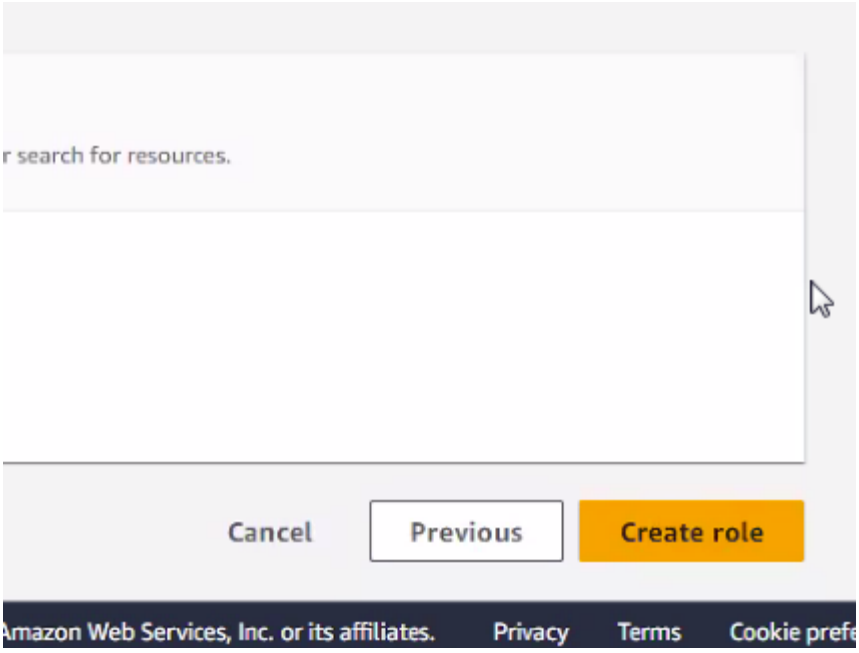
10. Search for “CloudWatchLogsReadOnlyAccess”:



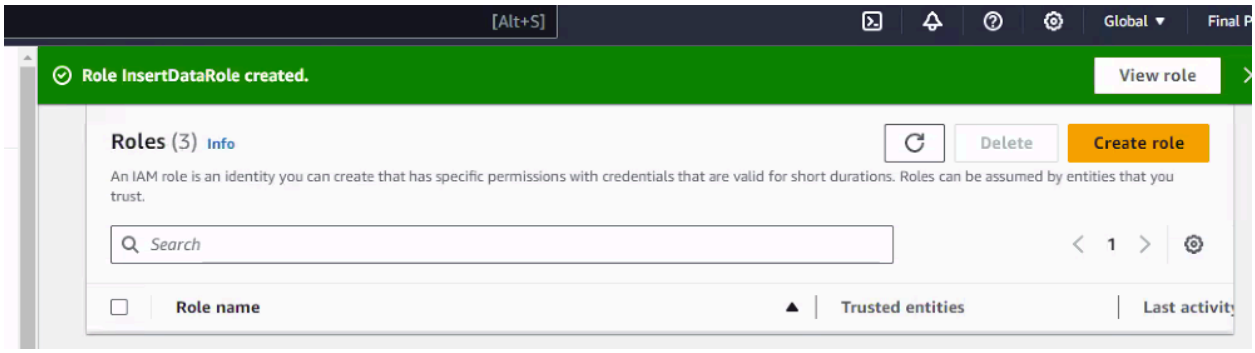
11. Go next
12. Then in Role Name: “InsertDataRole” (you can use whatever name you desire, just keep that in mind when using this lab’s code):



13. Scroll down

14. Create Role: 

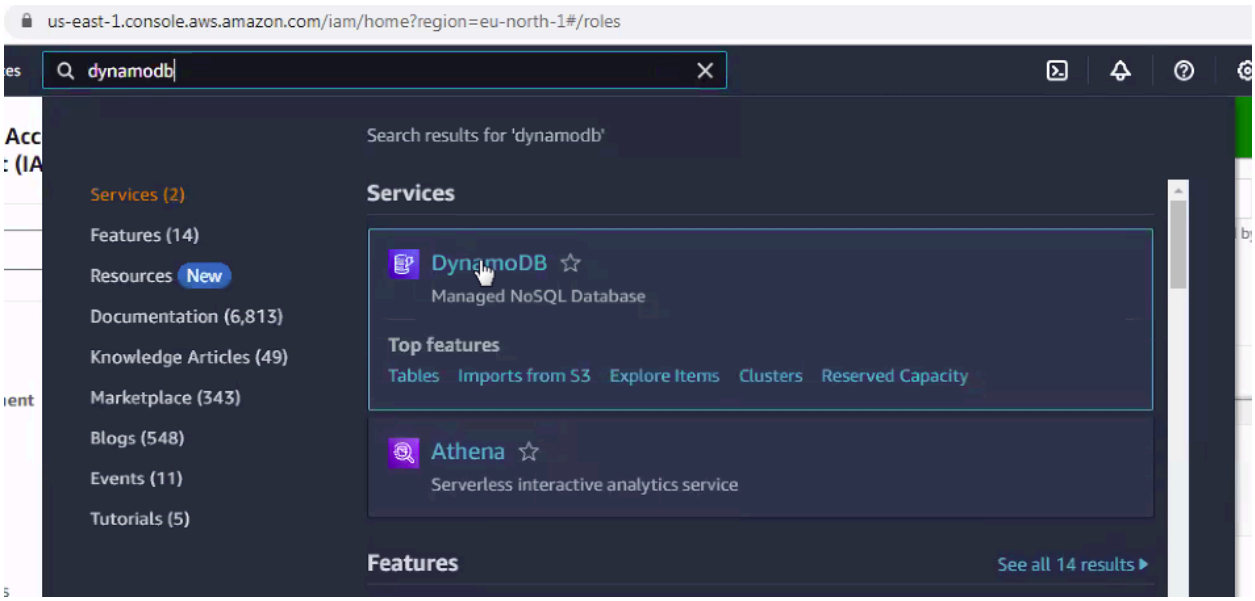
15. Success:



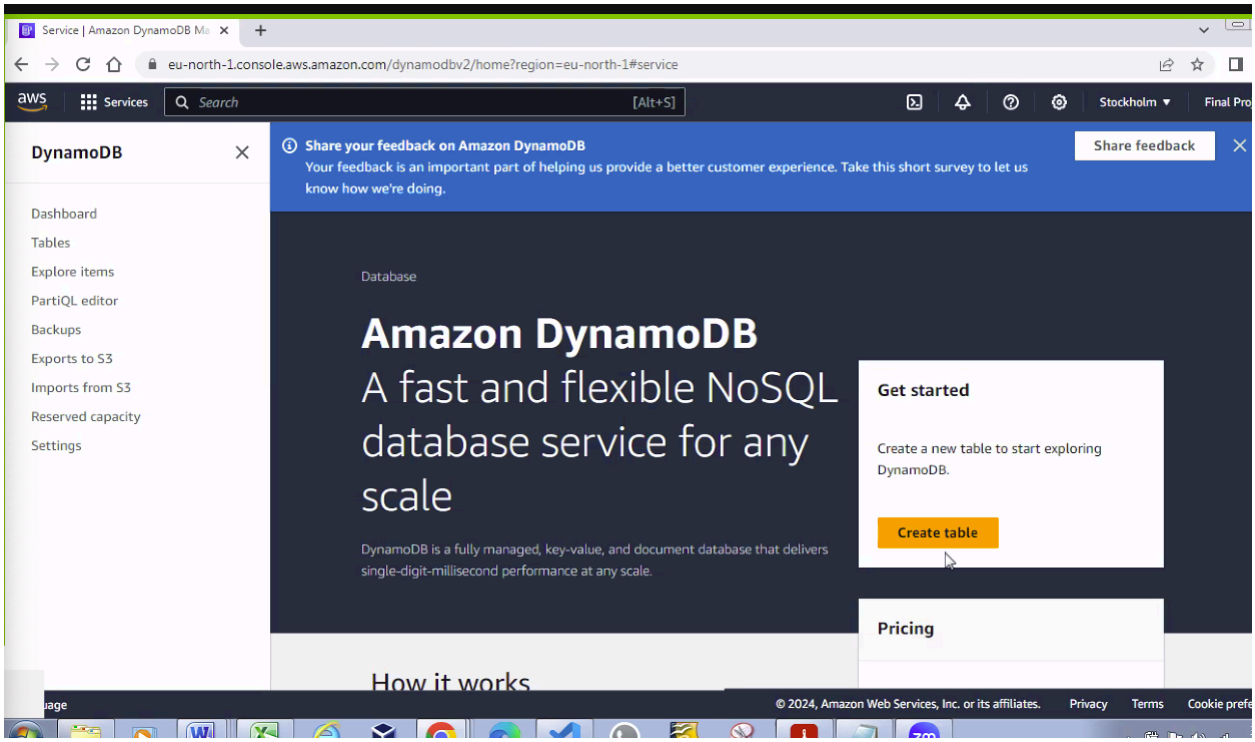
16. Section complete!

Part 4: DynamoDB

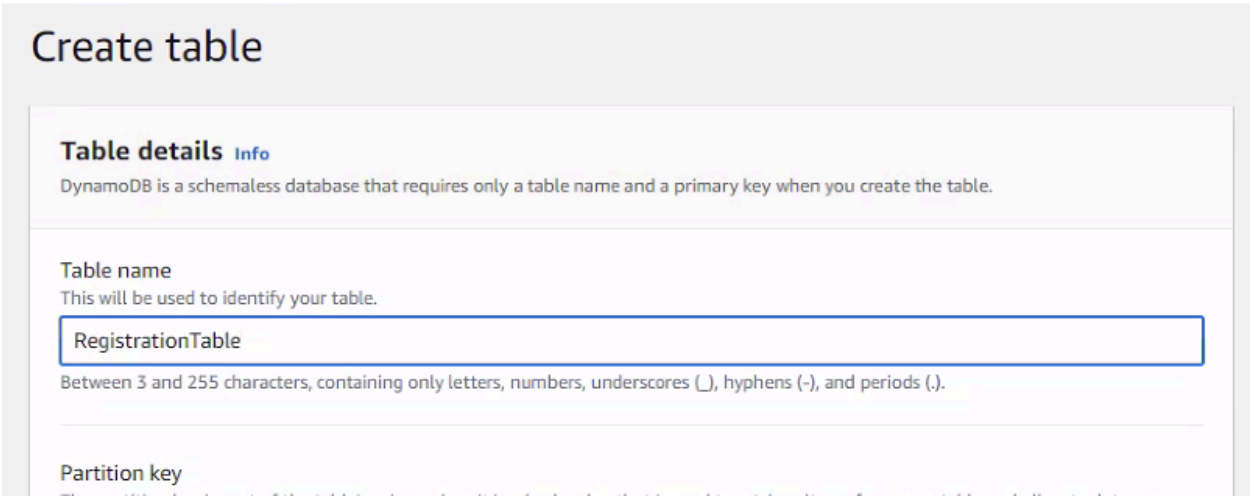
1. Go to DynamoDB on your console:



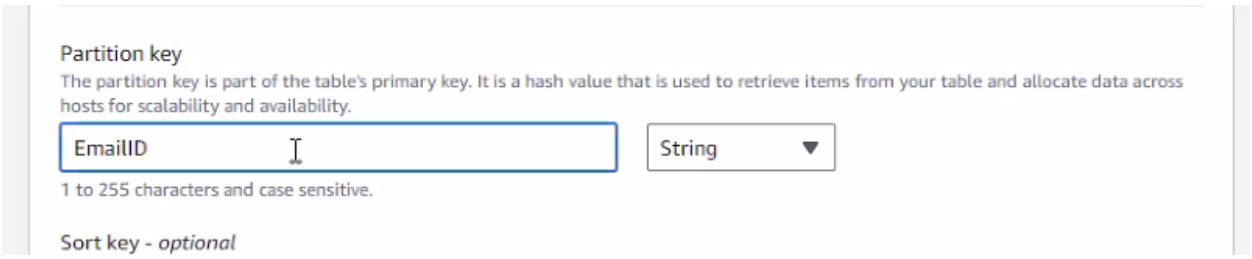
2. Create a Table:



3. Name Table name: ‘RegistrationTable’

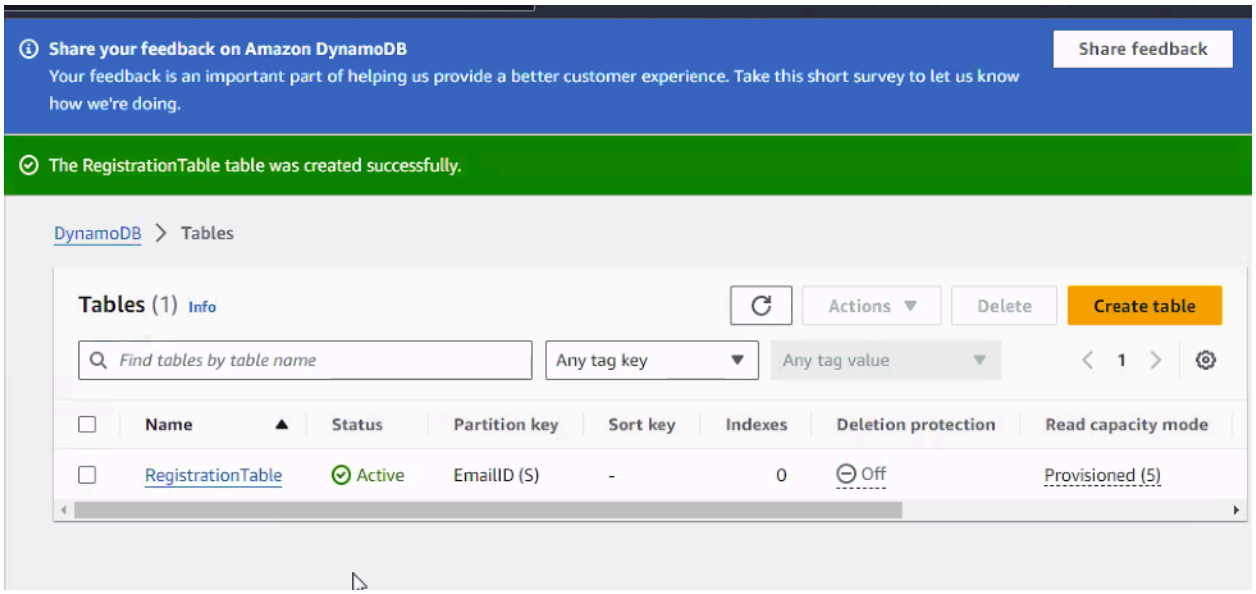


4. Partition Key: ‘EmailID’ :



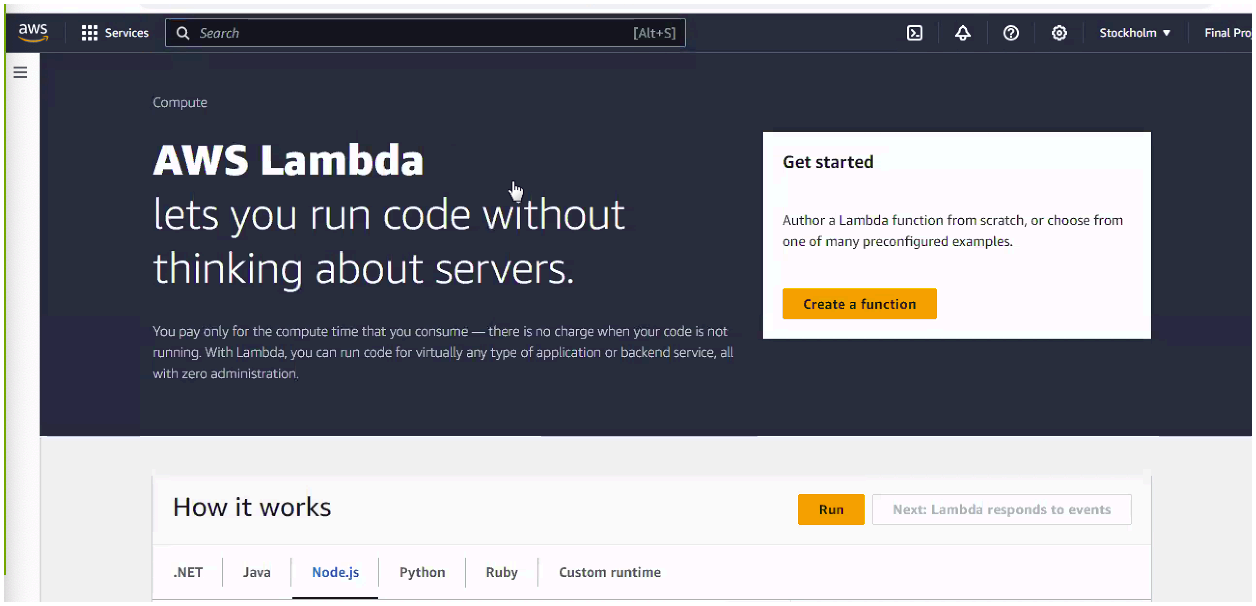
5. Scroll down and create table:

6. Success:

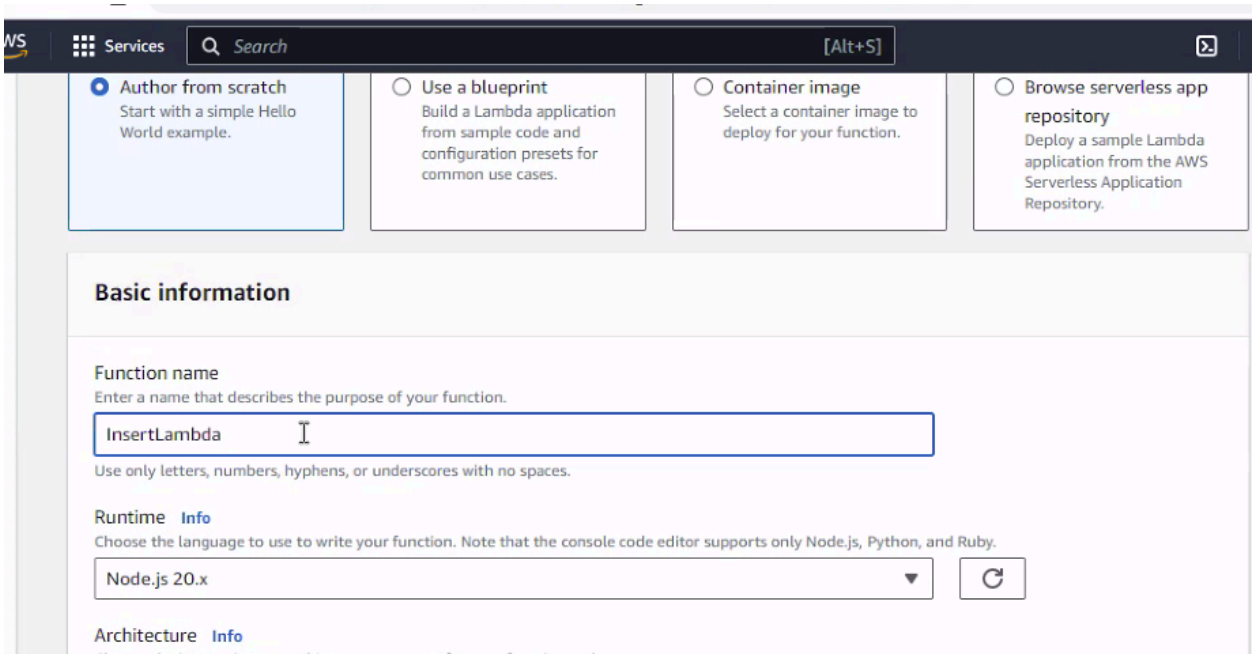


Part 5: Creating a Lambda

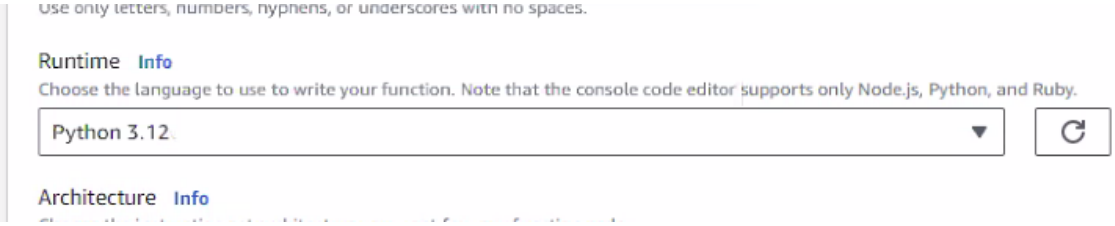
1. Go to Lambda on your console
2. Click on 'Create a function' on the right side of the screen (yellow-orange button):



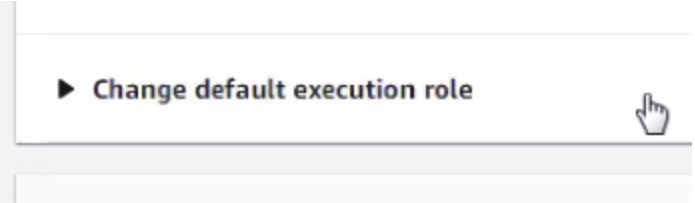
3. For ‘Function name’ : “InsertLambda”



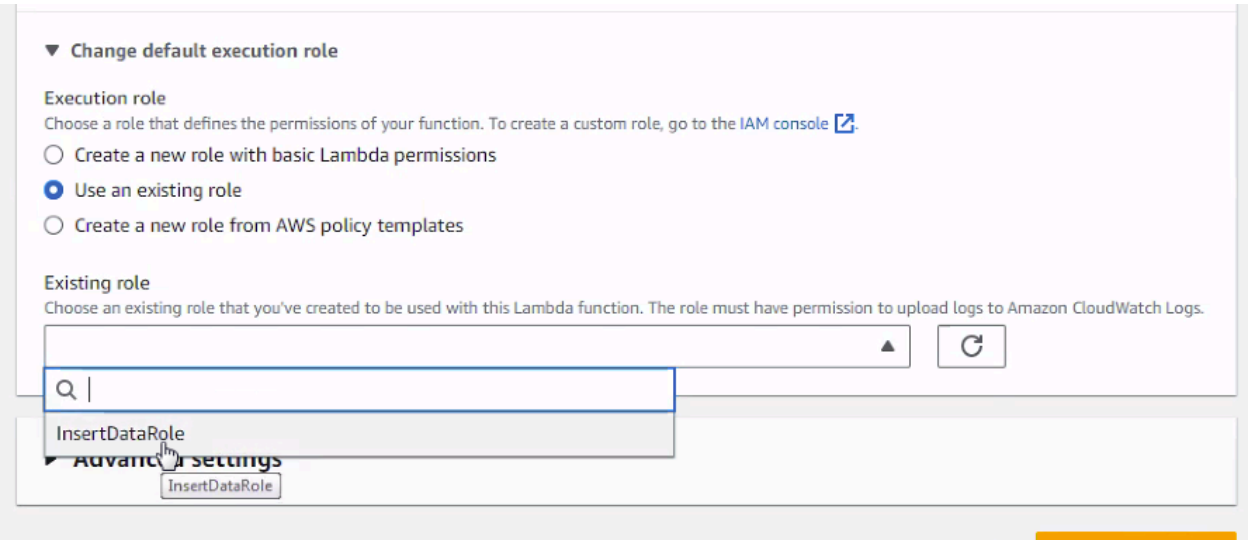
4. Runtime: ‘Python 3.12’



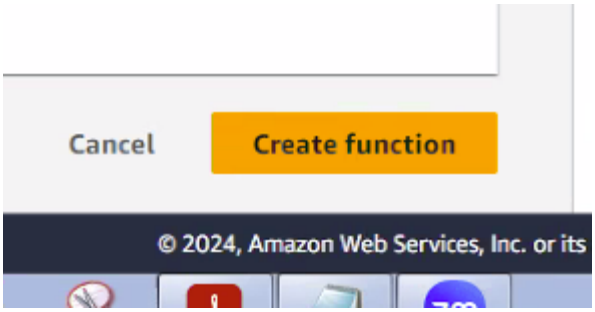
5. Click on ‘Change default execution role’ to expand it:



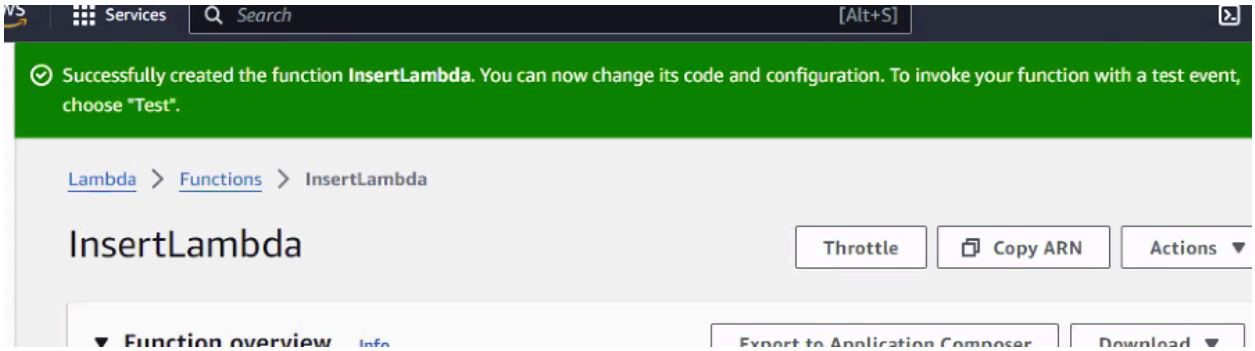
6. For ‘Existing role’ select the Role that you created previously (e.g. InsertDataRole)):



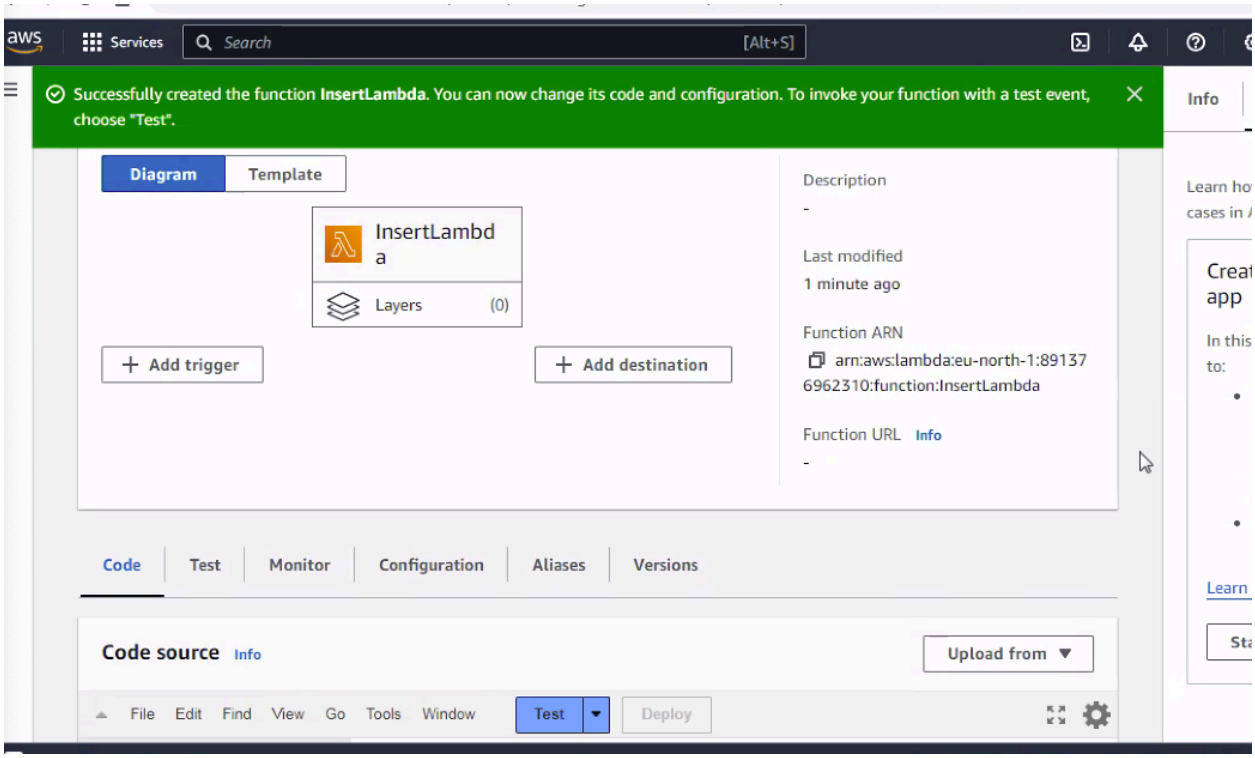
7. Click ‘Create function’:



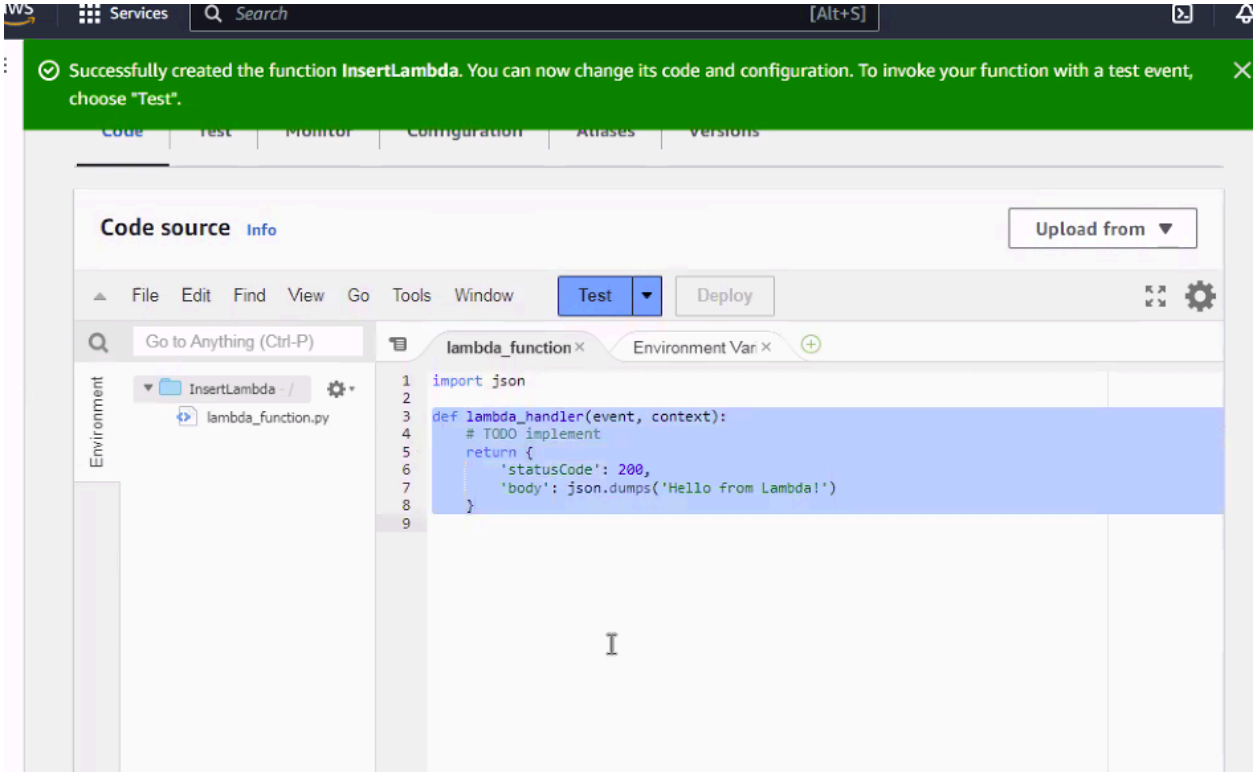
8. Success



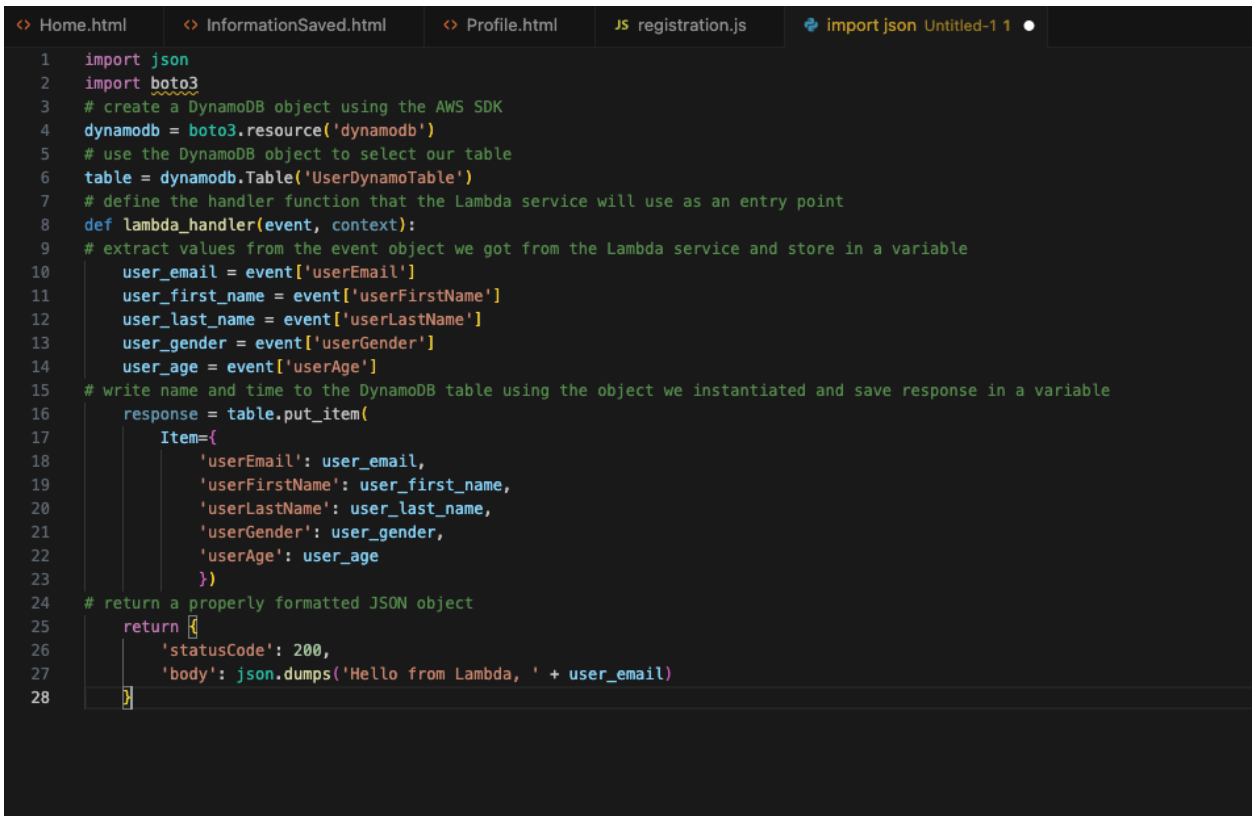
9. Scroll down to ‘Source Code’



10. Then delete the default code in the box:



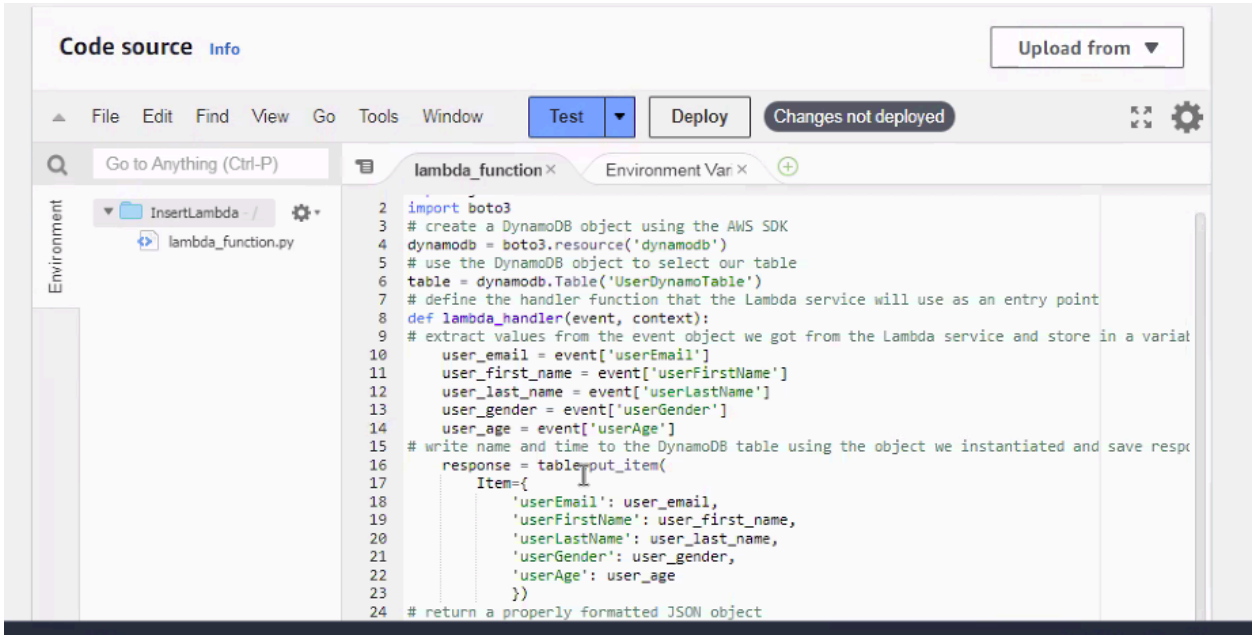
11. Insert the following code into that code editor box (the “lambda_function” tab)Insert this




```
1. import json
2. import boto3
3. # create a DynamoDB object using the AWS SDK
4. dynamodb = boto3.resource('dynamodb')
5. # use the DynamoDB object to select our table
6. table = dynamodb.Table('UserDynamoTable')
7. # define the handler function that the Lambda service will use as an
   entry point
8. def lambda_handler(event, context):
9. # extract values from the event object we got from the Lambda service
   and store in a variable
10.    user_email = event['userEmail']
11.    user_first_name = event['userFirstName']
12.    user_last_name = event['userLastName']
13.    user_gender = event['userGender']
14.    user_age = event['userAge']
```

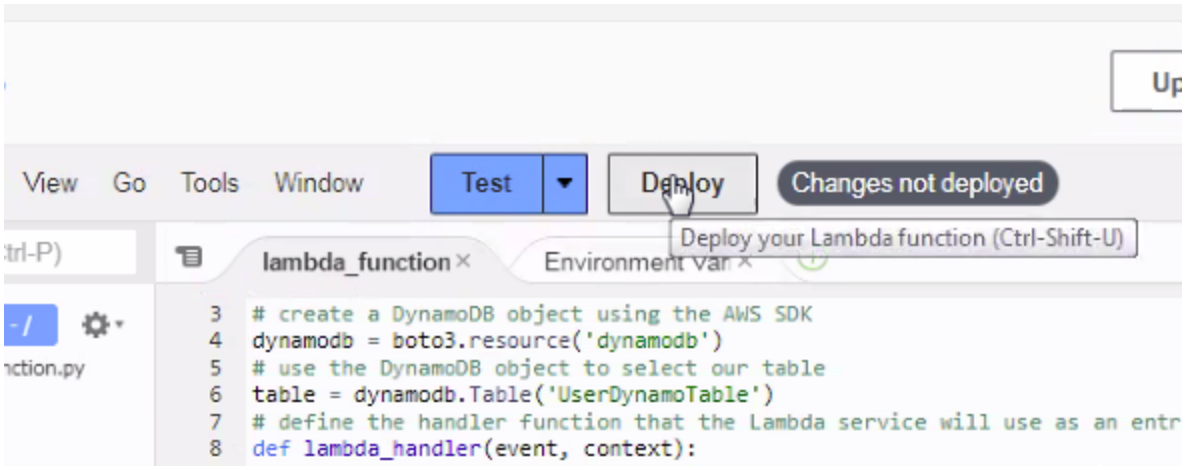
```
# write name and time to the DynamoDB table using the object we instantiated and save response in a variable
    response = table.put_item(
        Item={
            'userEmail': user_email,
            'userFirstName': user_first_name,
            'userLastName': user_last_name,
            'userGender': user_gender,
            'userAge': user_age
            })
# return a properly formatted JSON object
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda, ' + user_email)
    }
```

12. Paste, and it should look like this:

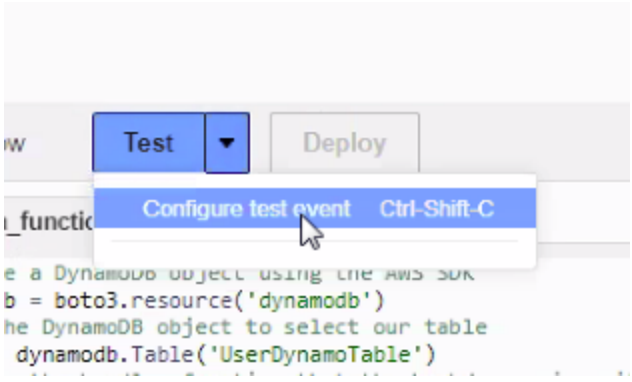


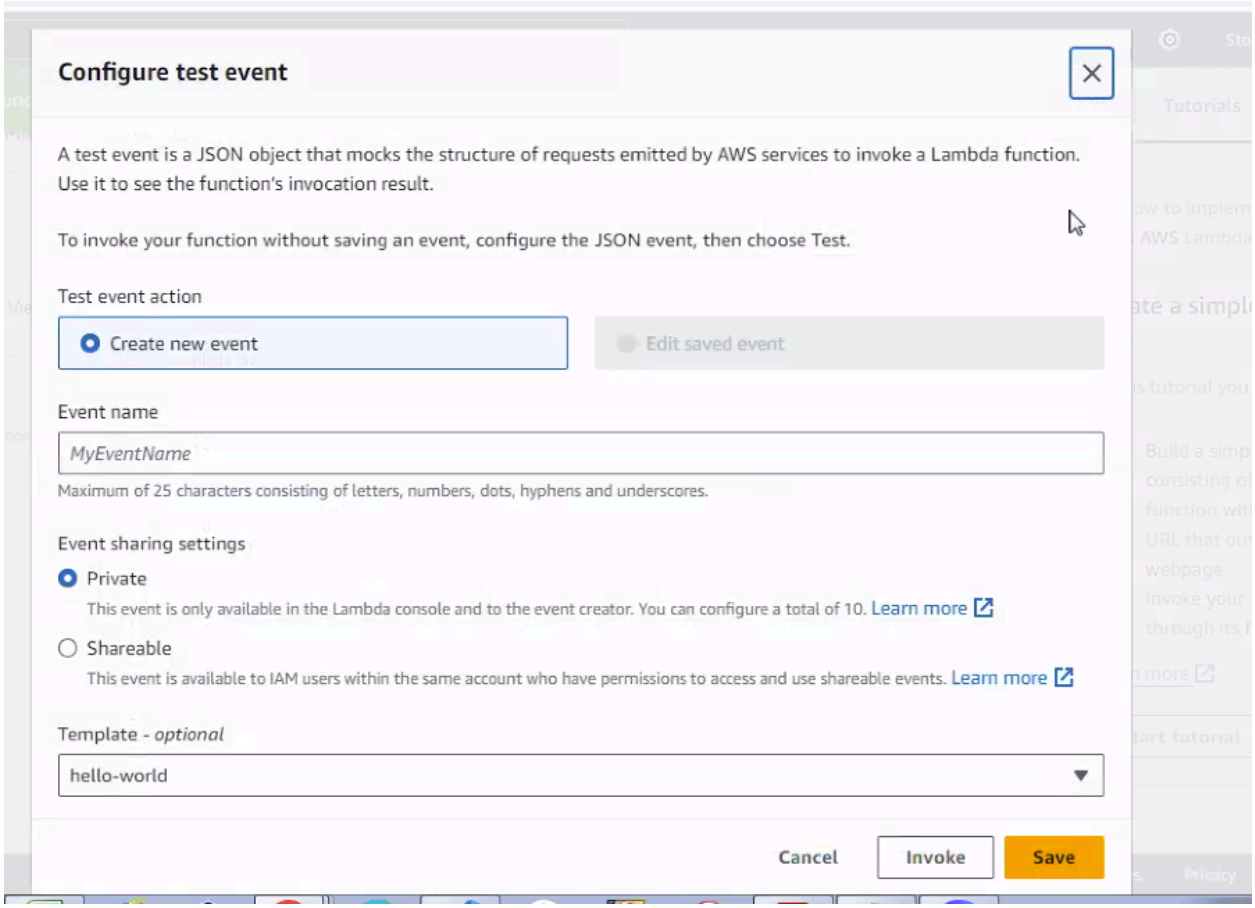
13. crtl+s to save

14. then deploy:

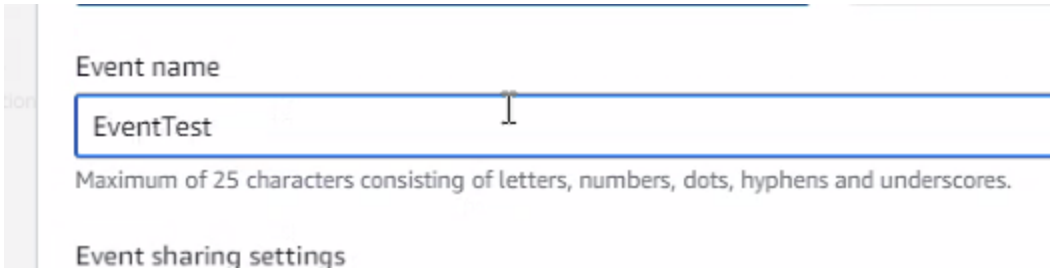


15.



16. 

17. Event name: 'EventTest'



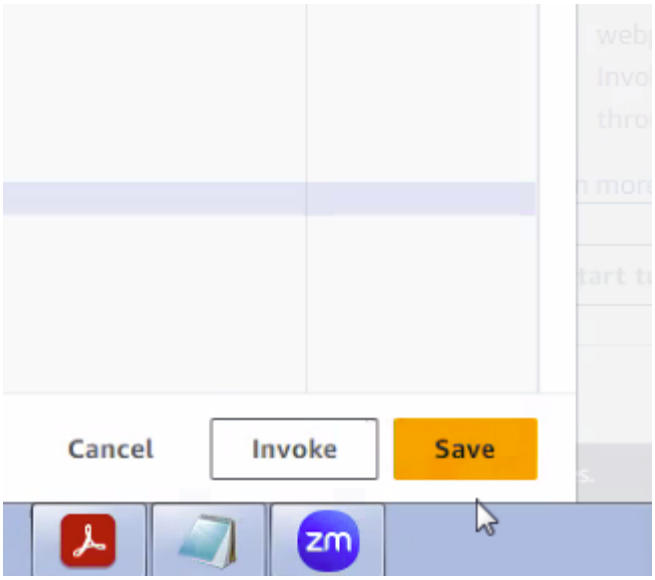
18. Proceed

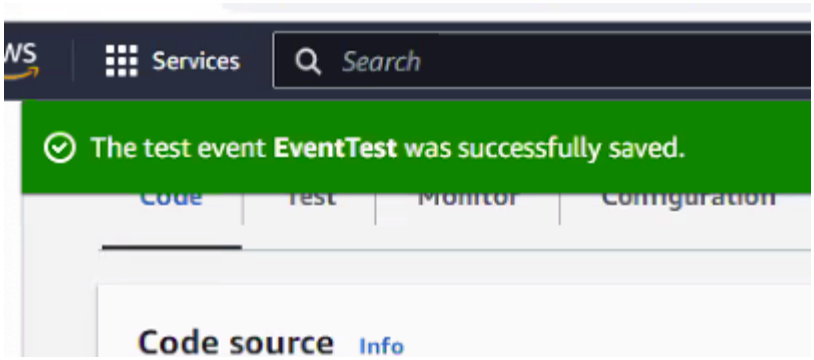
19. For Event JSON box, delete the default code already there:

20. Delete that code
21. Then insert this code in its place:

```
{
  "userEmail": "Eva@email.com",
  "userFirstName": "Eva",
  "userLastName": "Kore",
  "userGender": "Female",
  "userAge": "23"
}
```

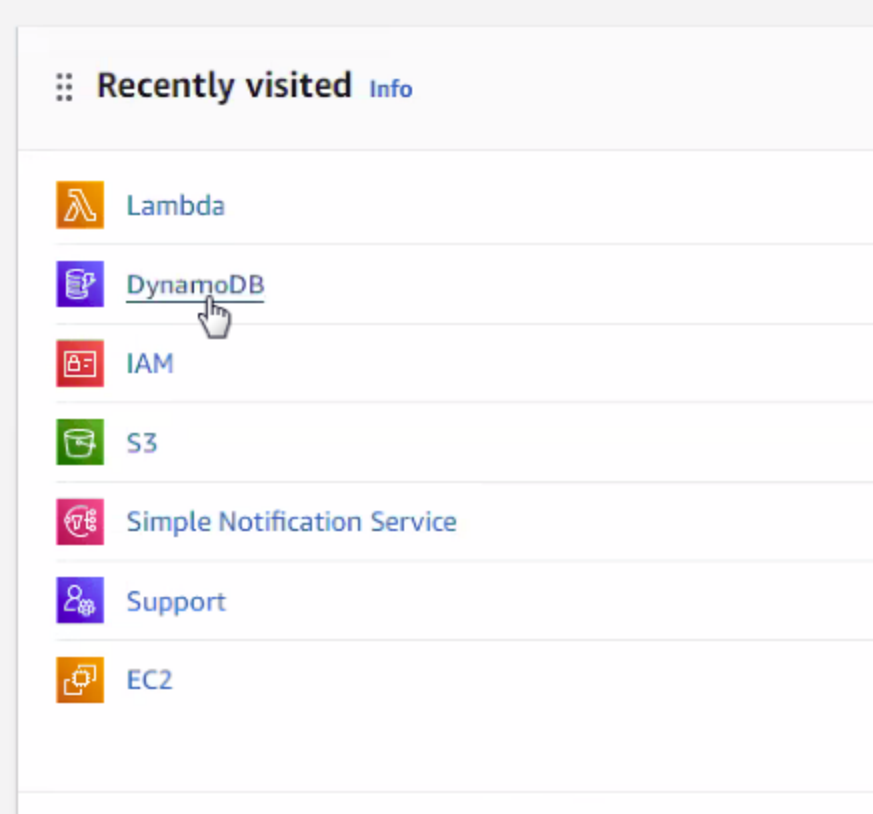
22.

23. Save it 

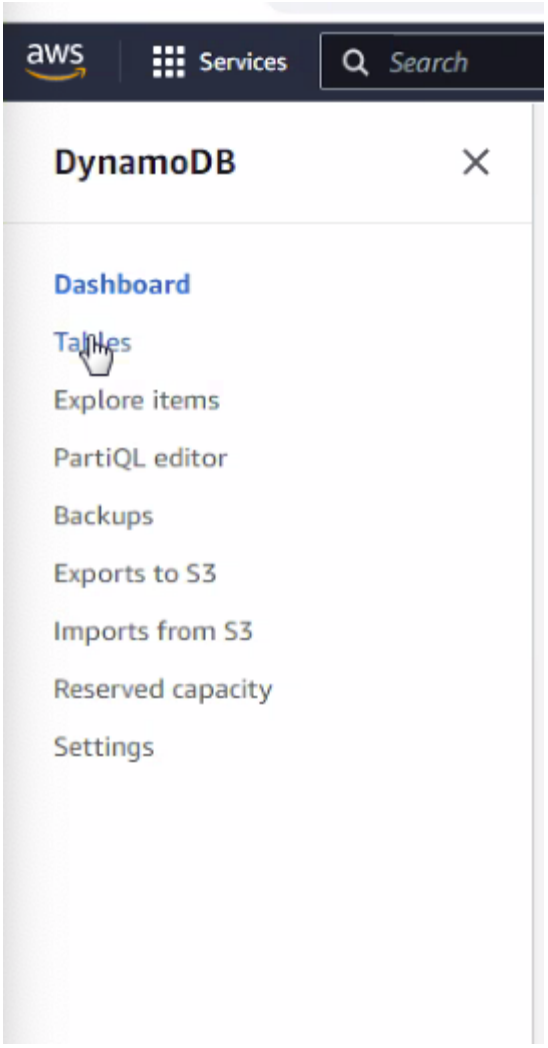
24. Success ! 

Part 6: Returning back to DynamoDB

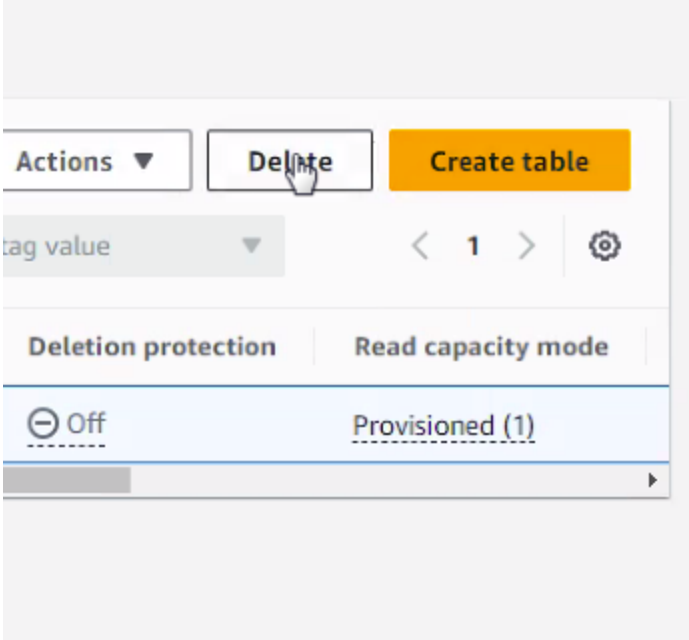
1. Go back to dynamoDB

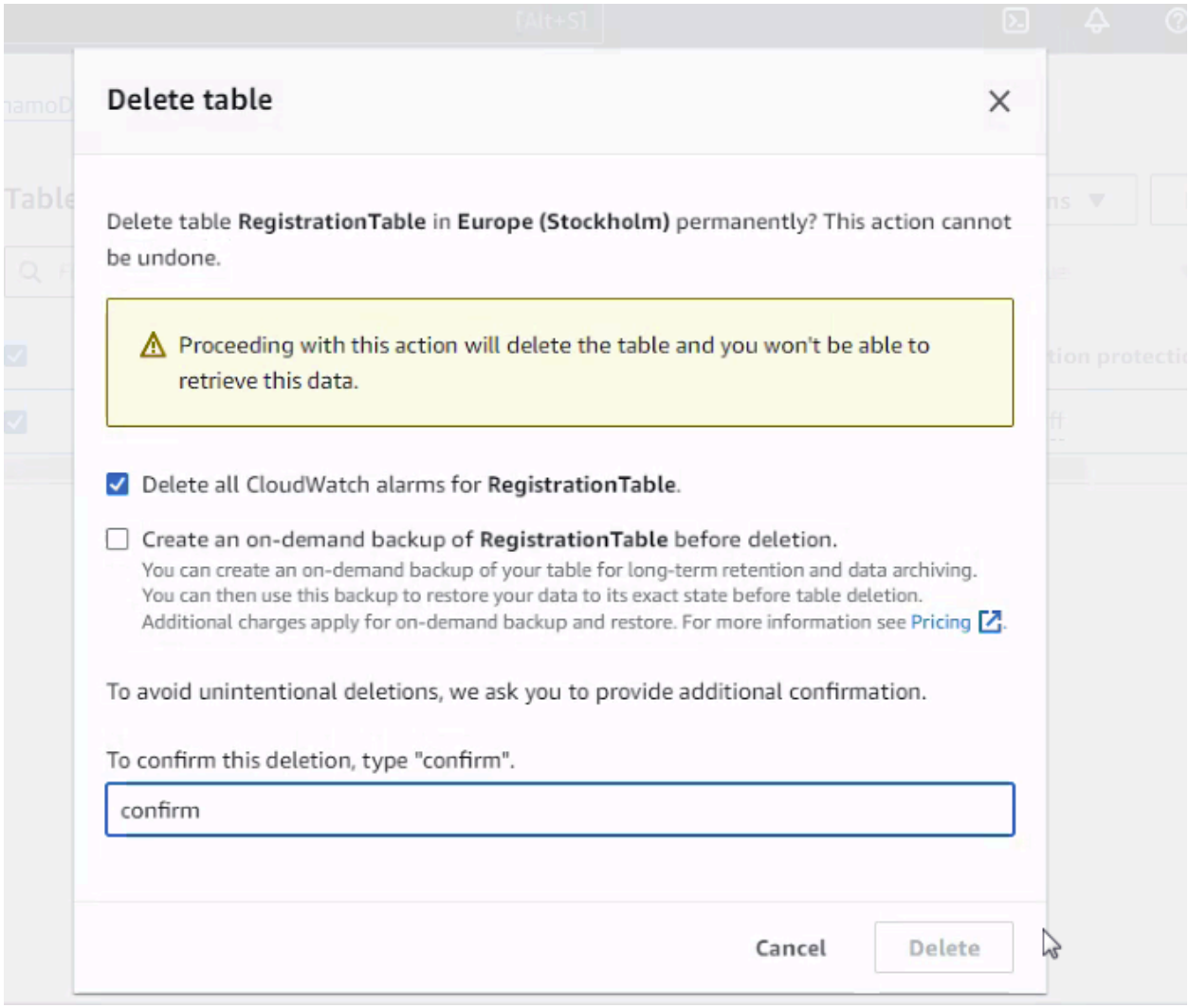


2. Go to tables

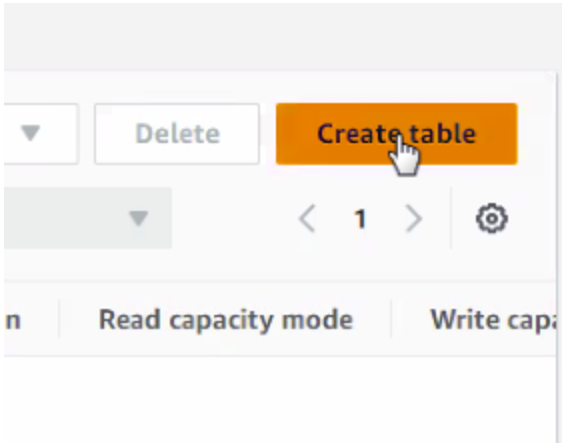


3. [are these steps necessary to keep? Or delete?] Click on your table that you created, click its check box and delete the table

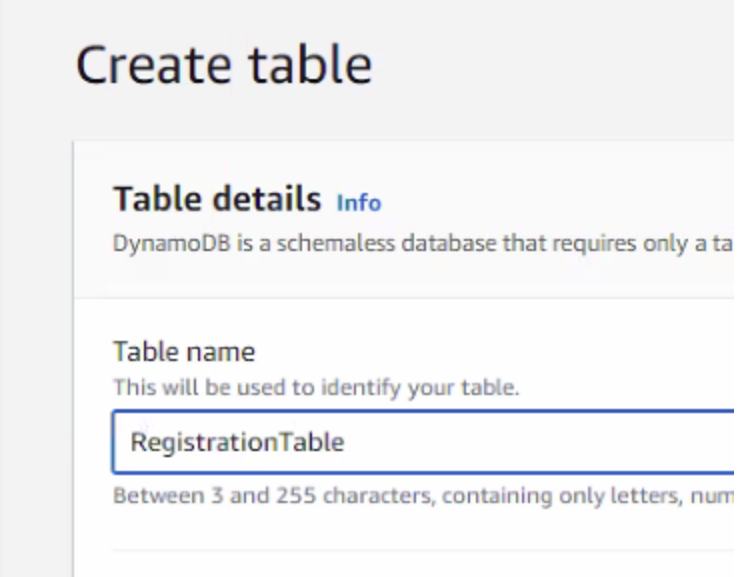


4. 

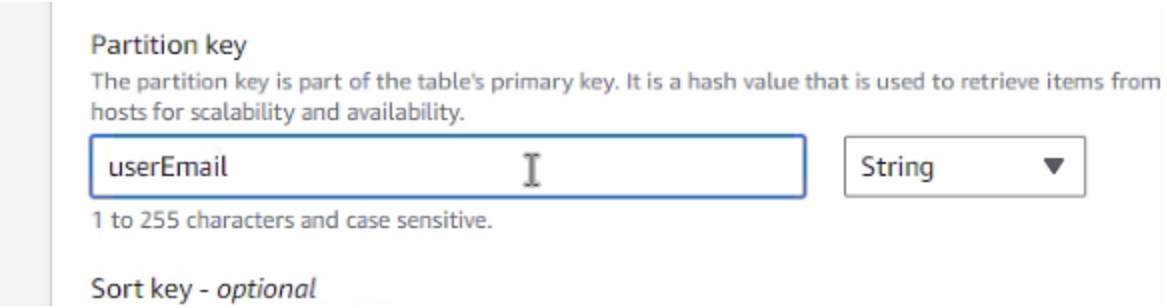
5. [due to project not working as initially planned] Create the table again



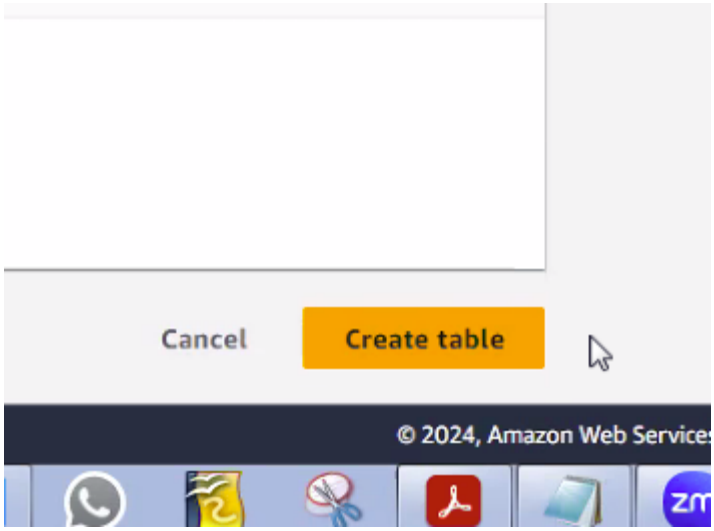
6. Name the Registration table “RegistrationTable



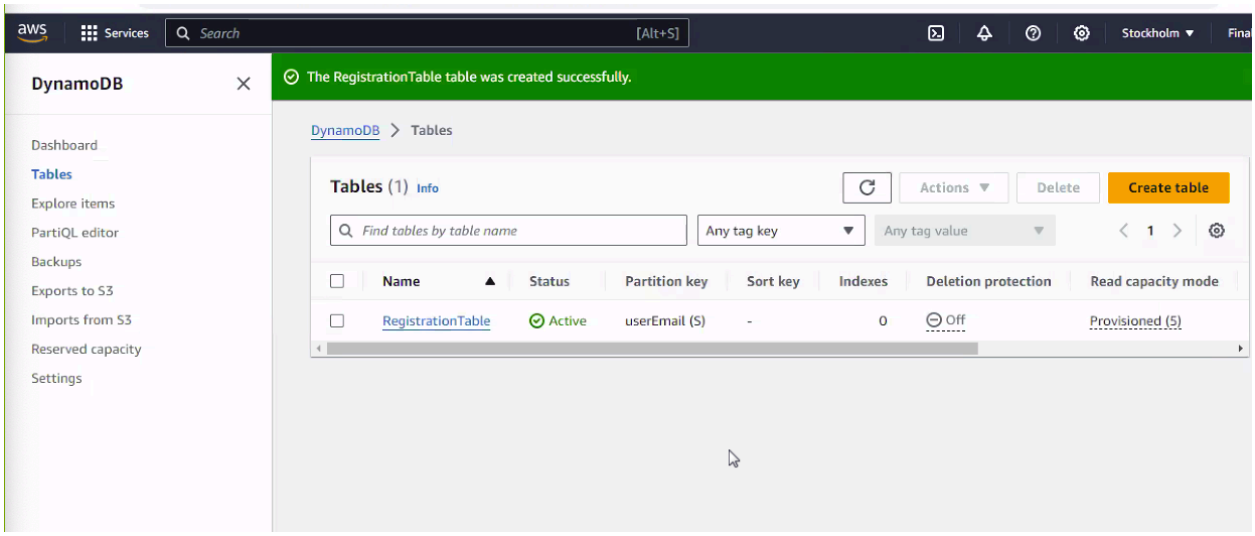
7. Name ‘Partition Key’ as “userEmail”



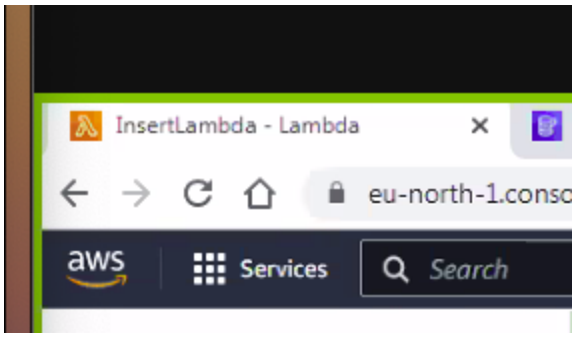
8. Scroll down and create table:



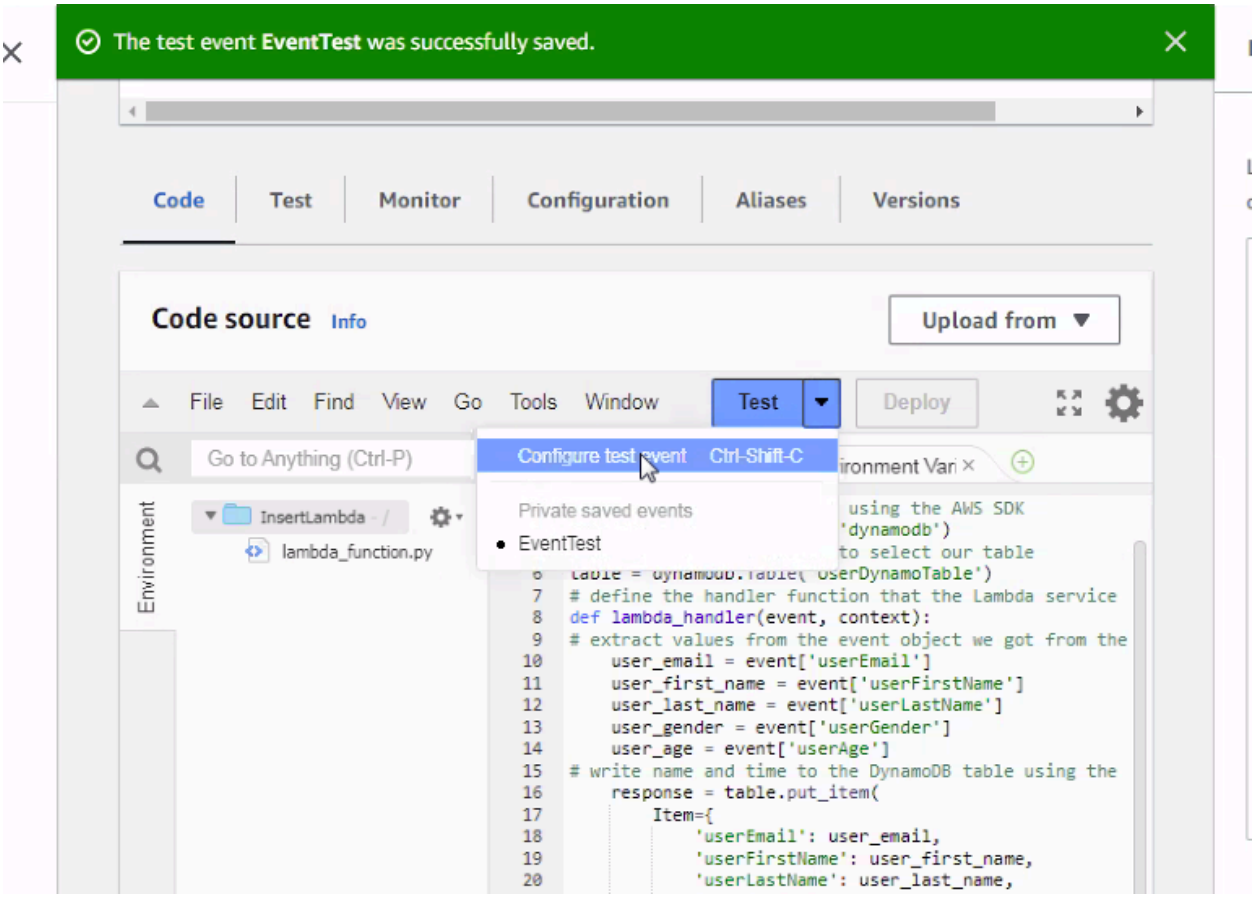
9. Success :



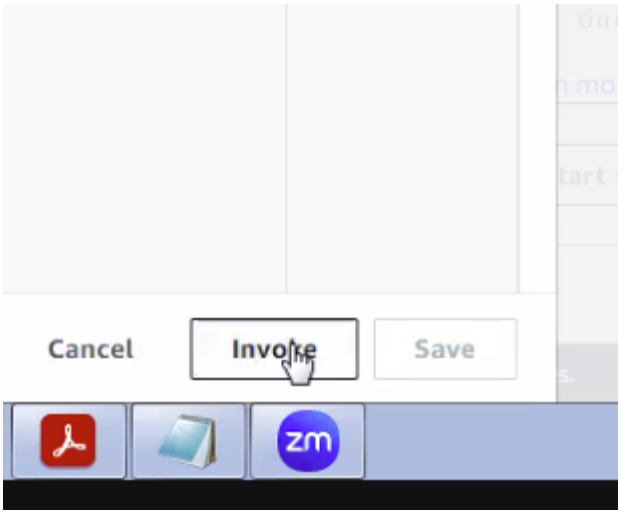
10. Go to your Lambda service tab InsertLambda:



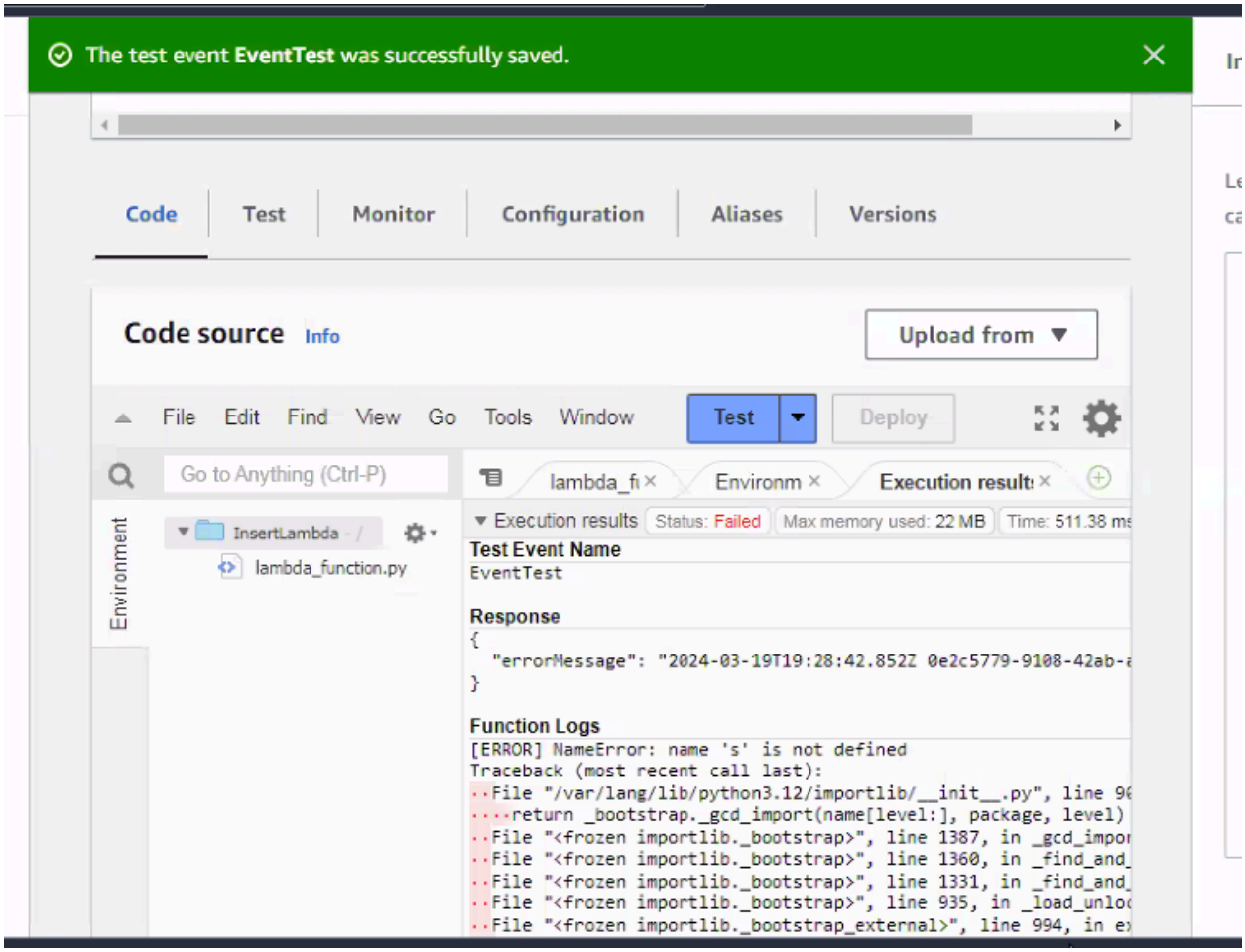
11. Configure test event



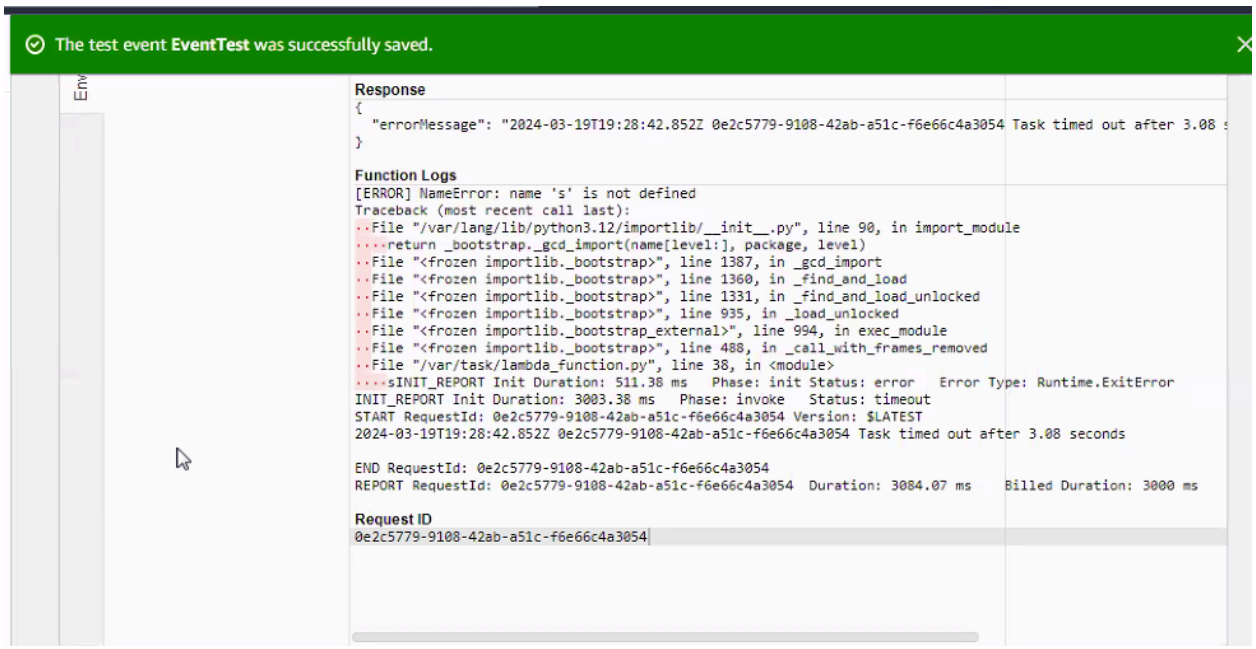
12. Press 'Invoke' down on the bottom:



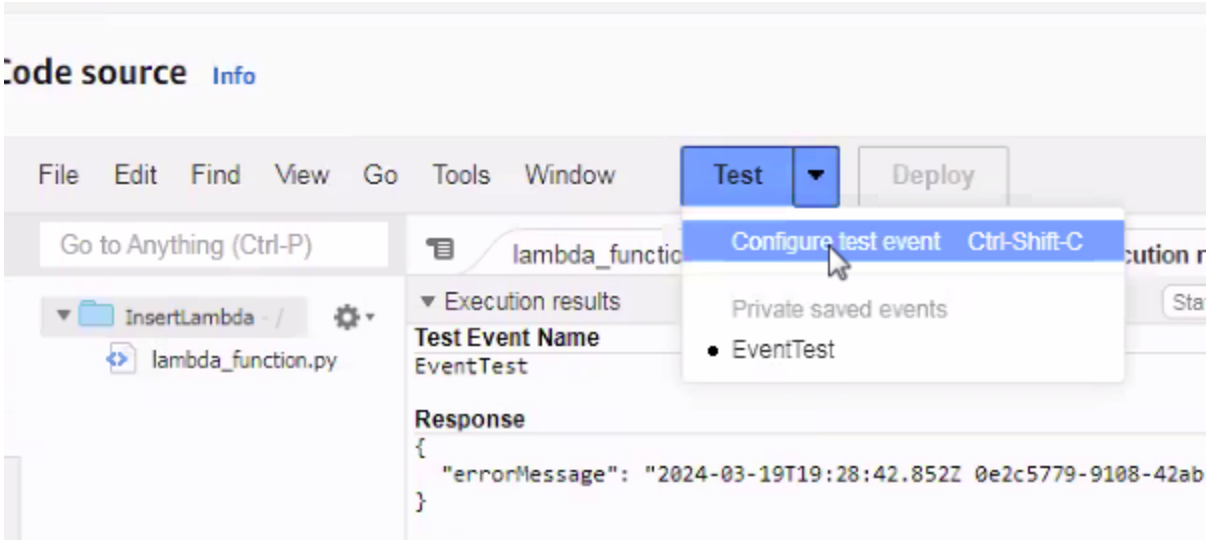
13. [error! You need to investigate what went wrong!]



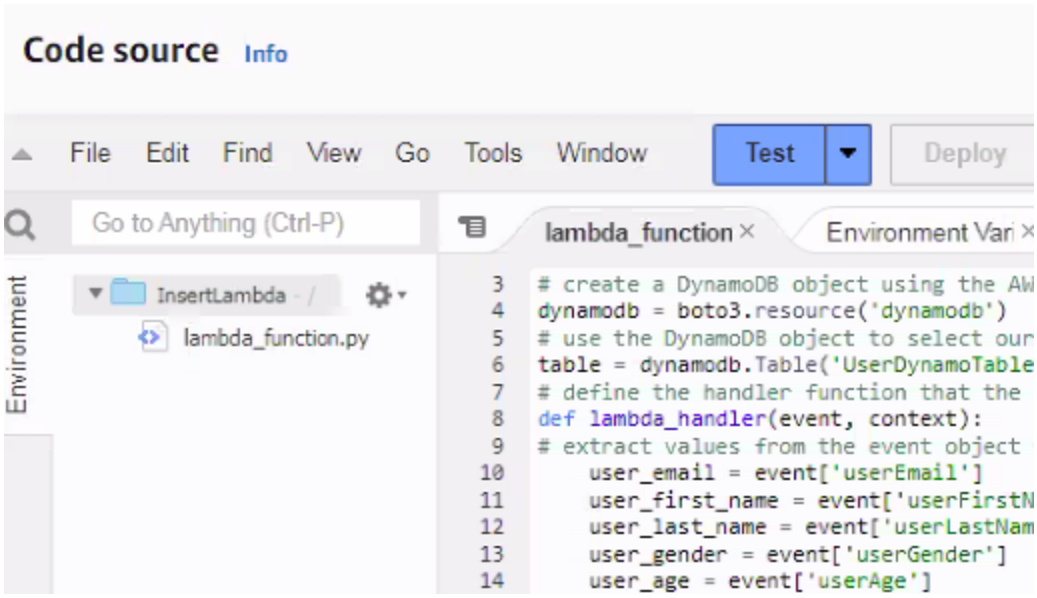
14. Status: FAILED! Read the Function Logs to see the problem. What do you think went wrong?



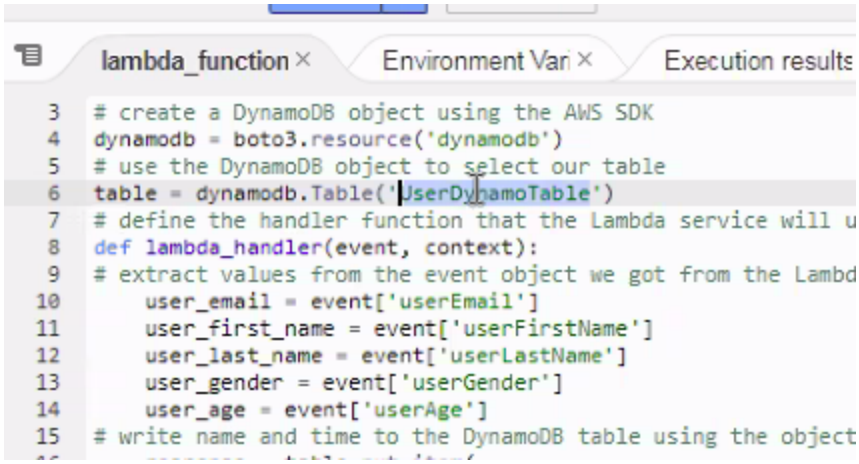
15. Check the log, it mentions “line 38” what is on line 38? (we will find out in the next few steps)
16. Go back here to investigate the error:



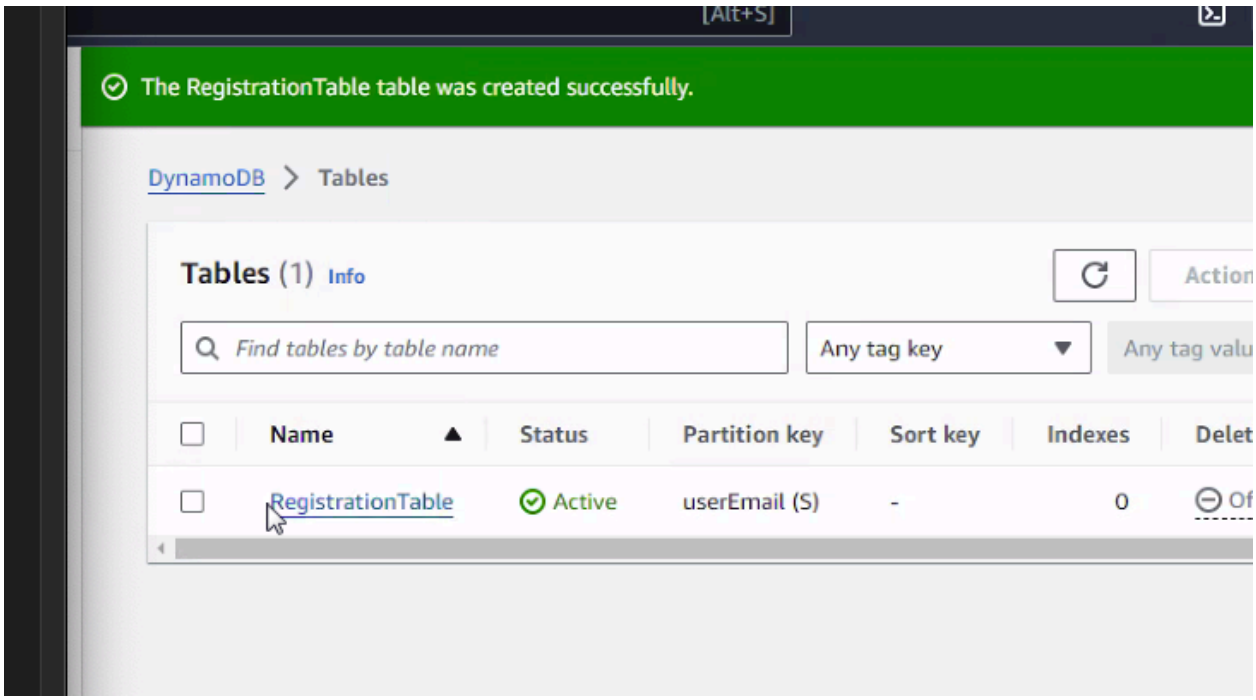
17. Click on 'lambda_function' tab in the Code Source editor



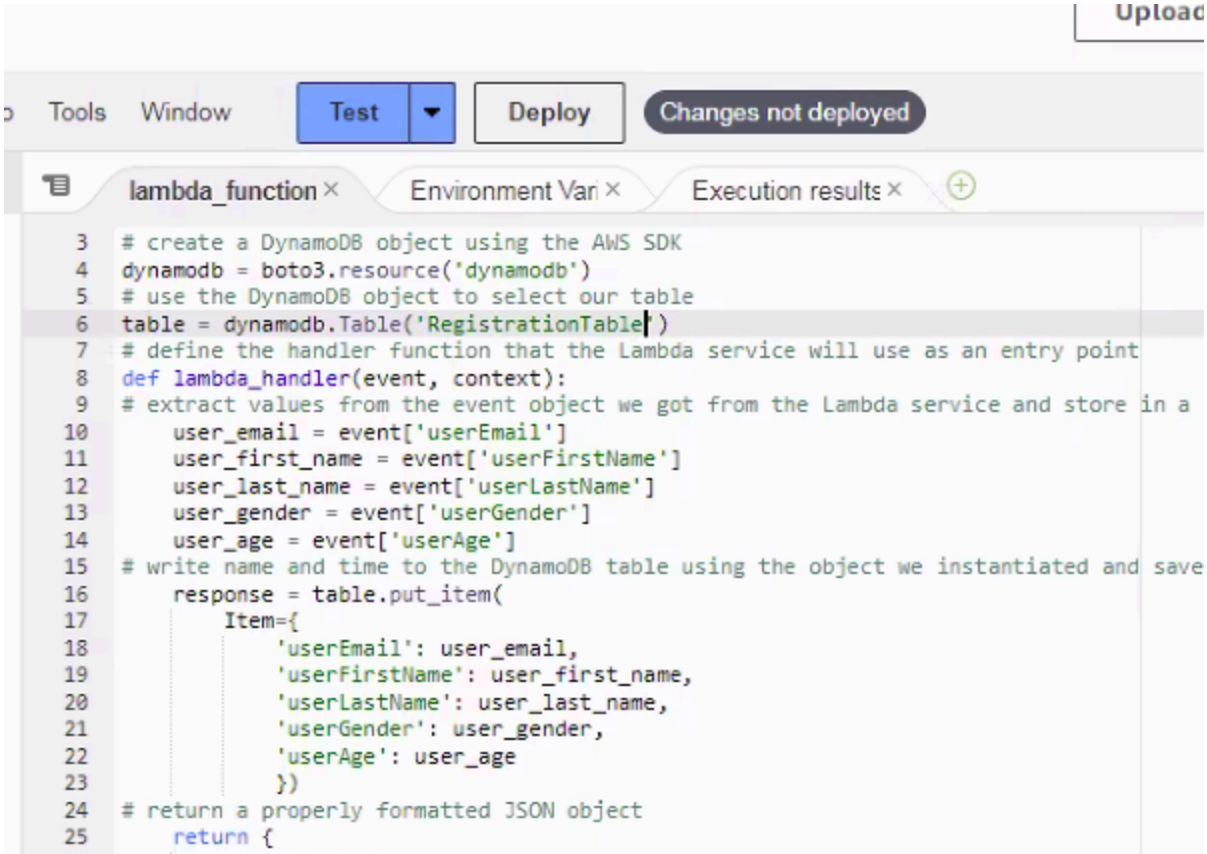
18. Line 6! fix that. The code had a table name from another specific user's account. Change it to yours. Name the table what you named the table in your system, the script has named it "UserDynamoTable" by default



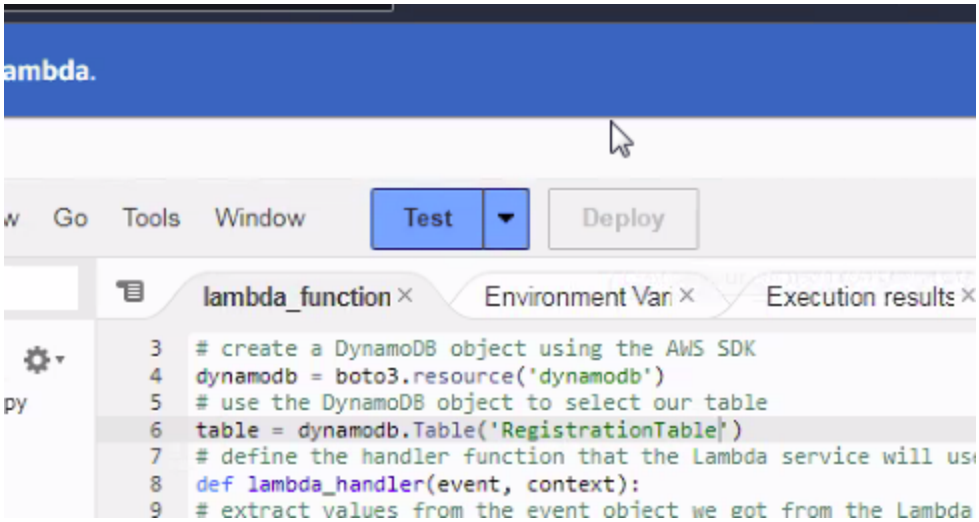
19. Check what you named your DynamoDB table? E.g. this lab has name its table ‘RegistrationTable’



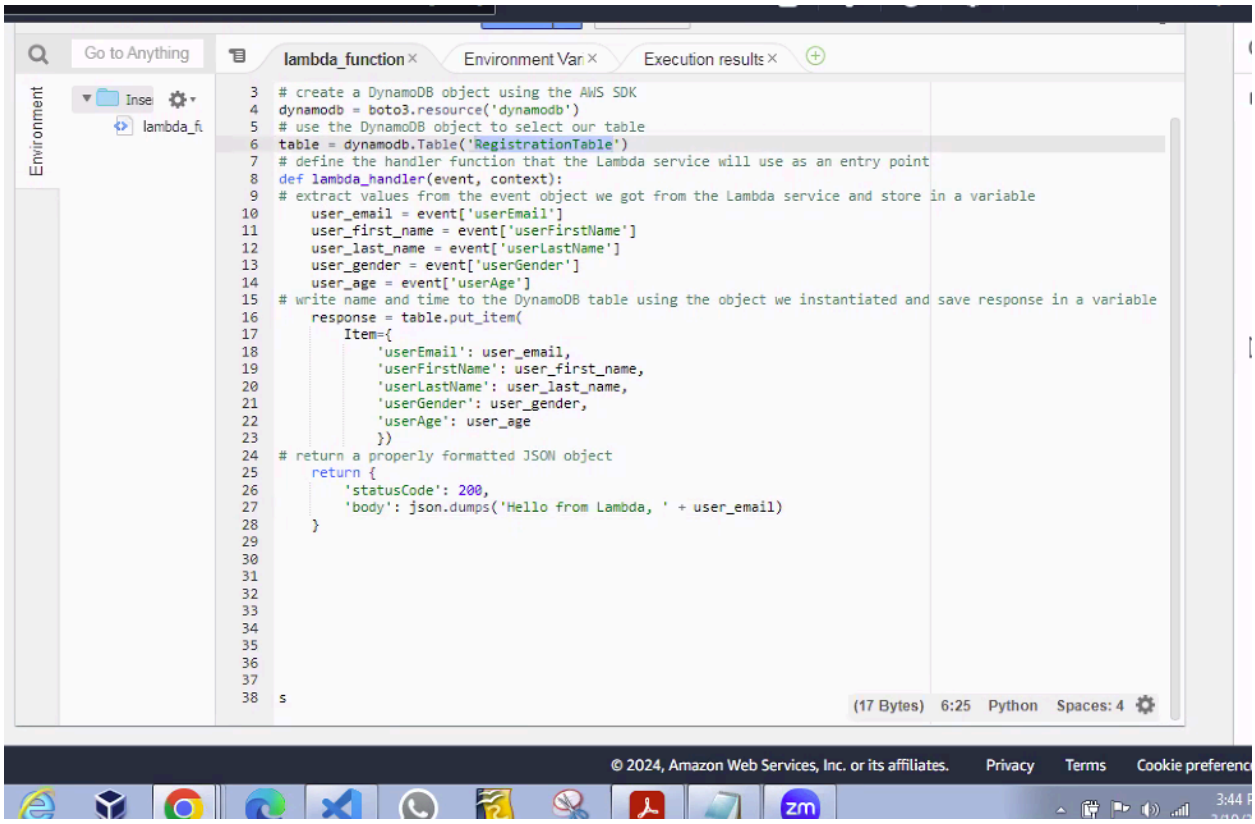
20. Click ‘deploy’ button



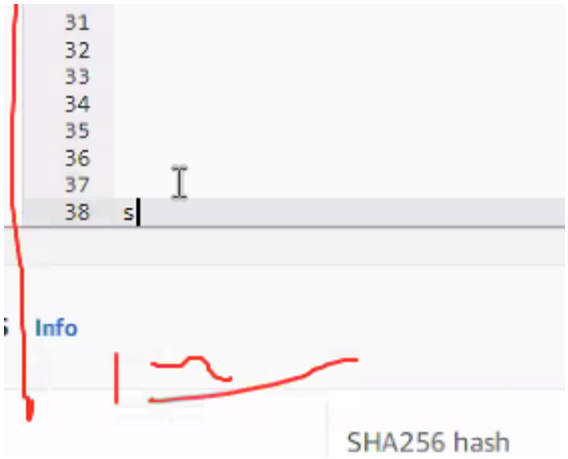
21. Now click ‘Test’



22. See anything odd?

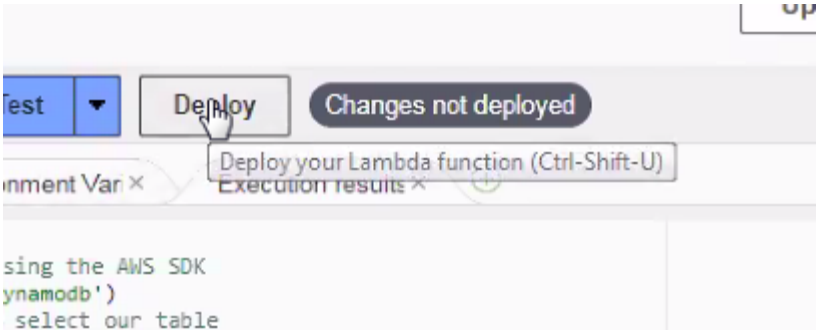


23. Look at line 38:

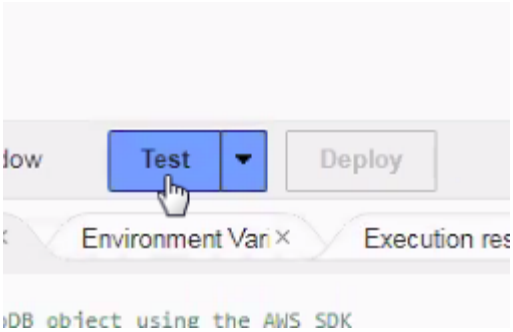


24. Delete that letter ‘s’ on line 38

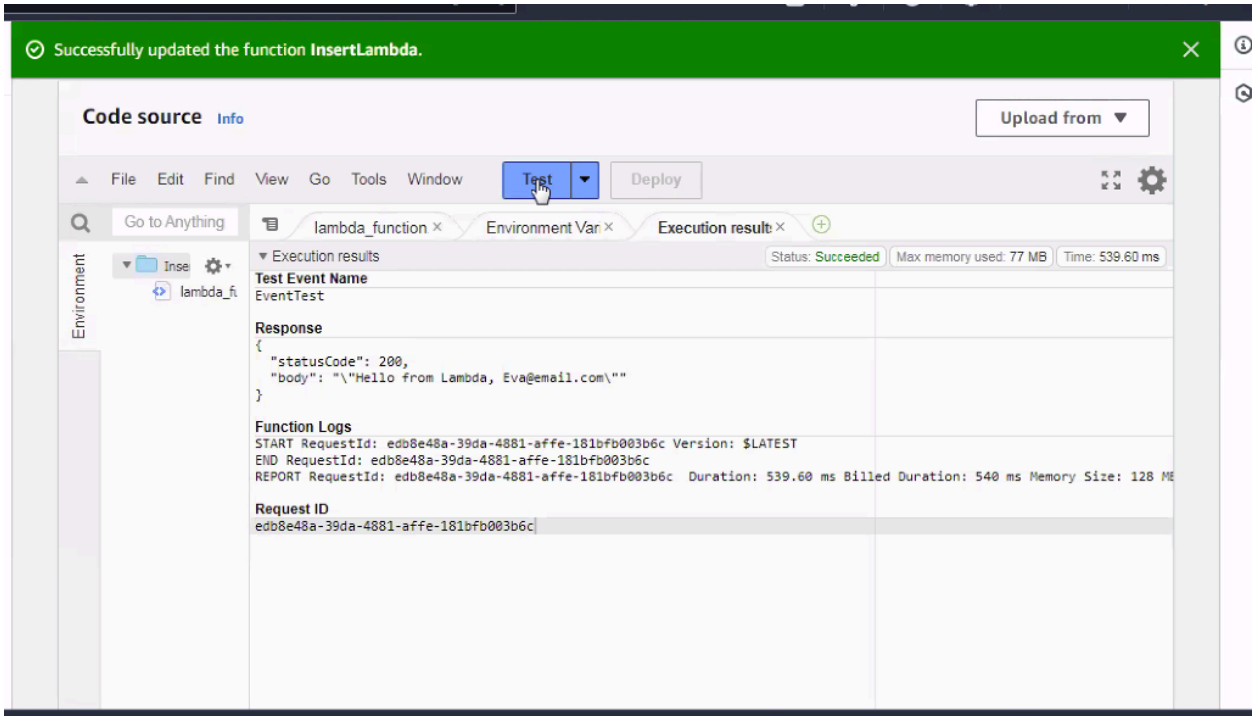
25. Click ‘deploy’



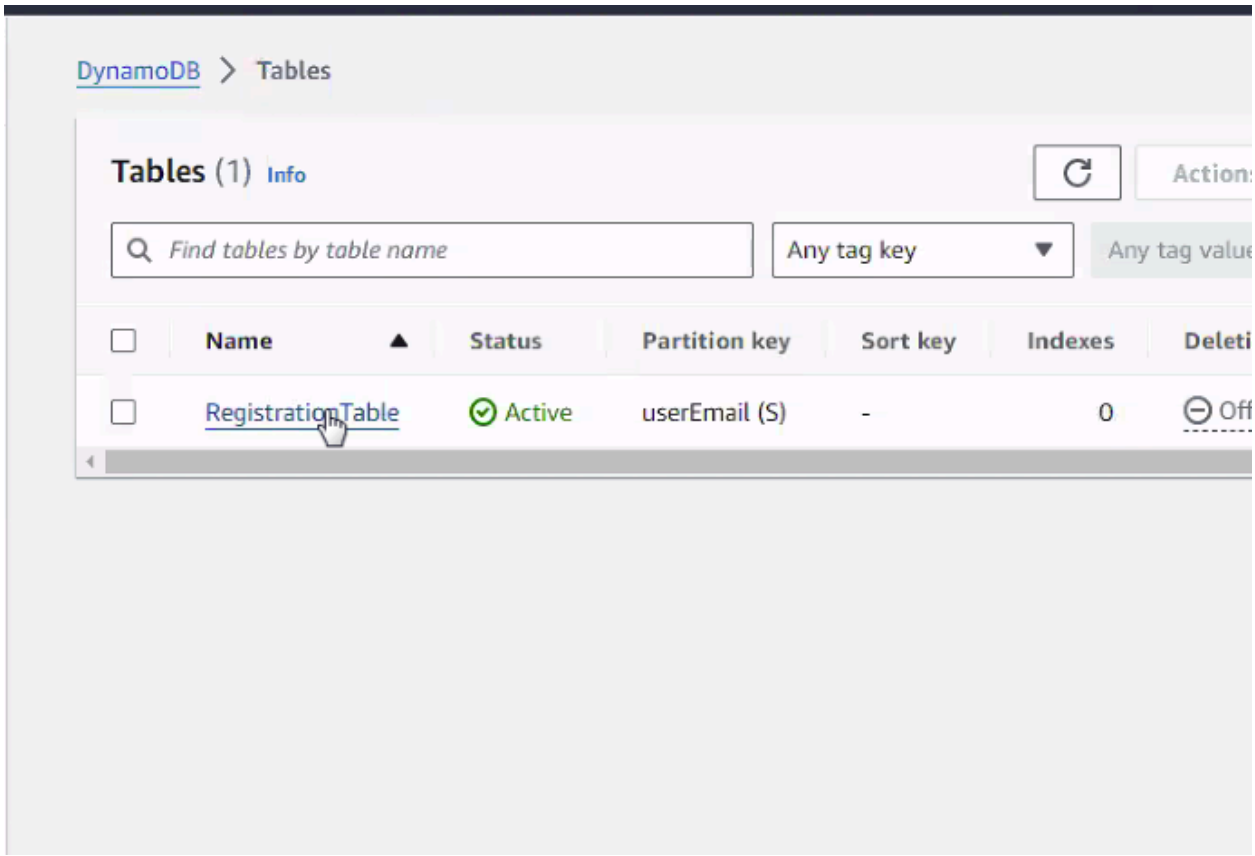
26. Click ‘test’



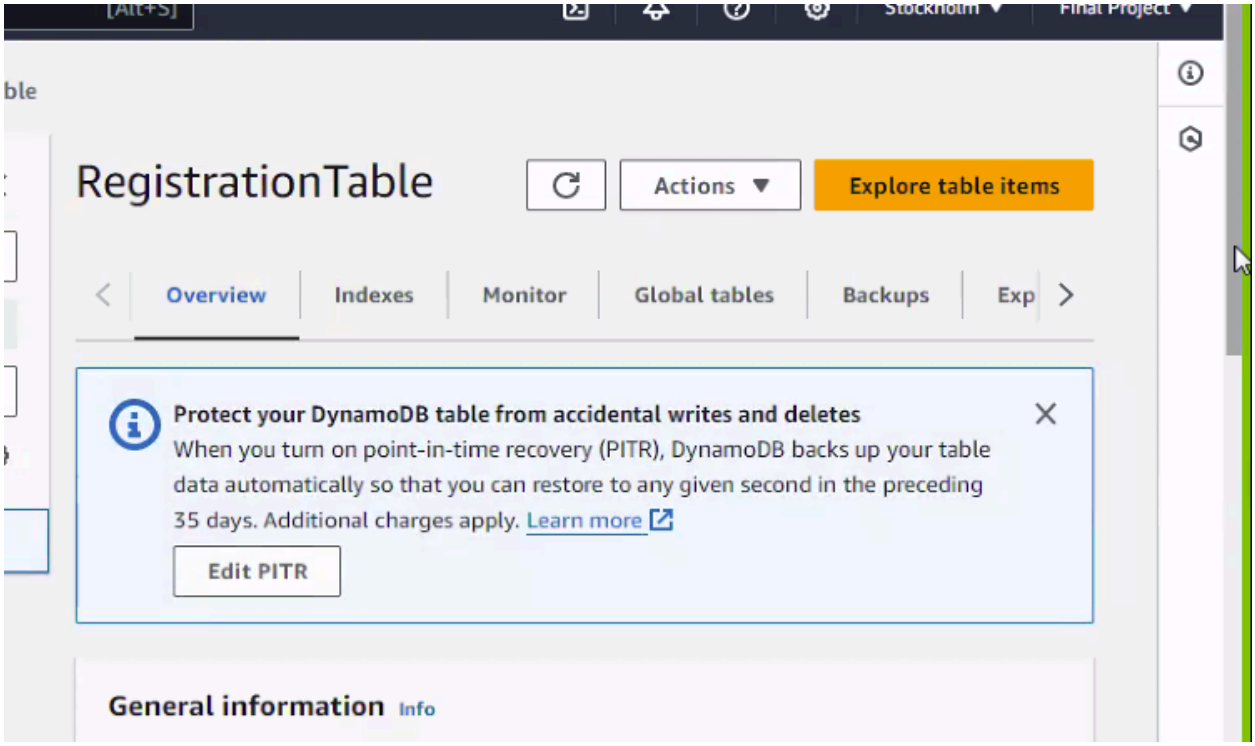
27. Success!

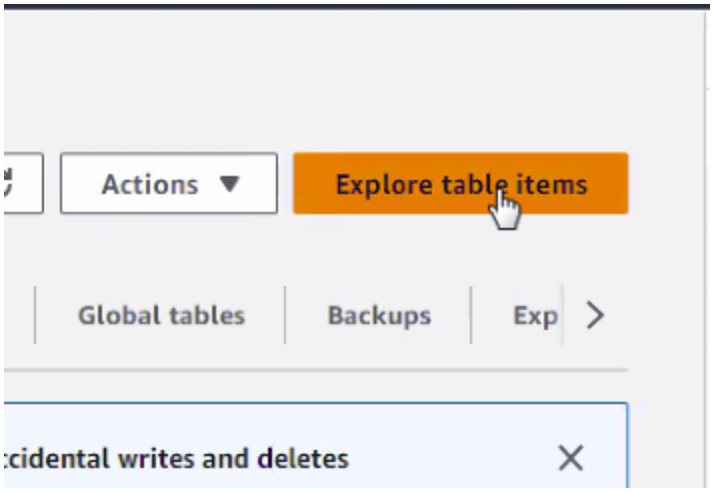


28. Go to DynamoDB and click on your table:

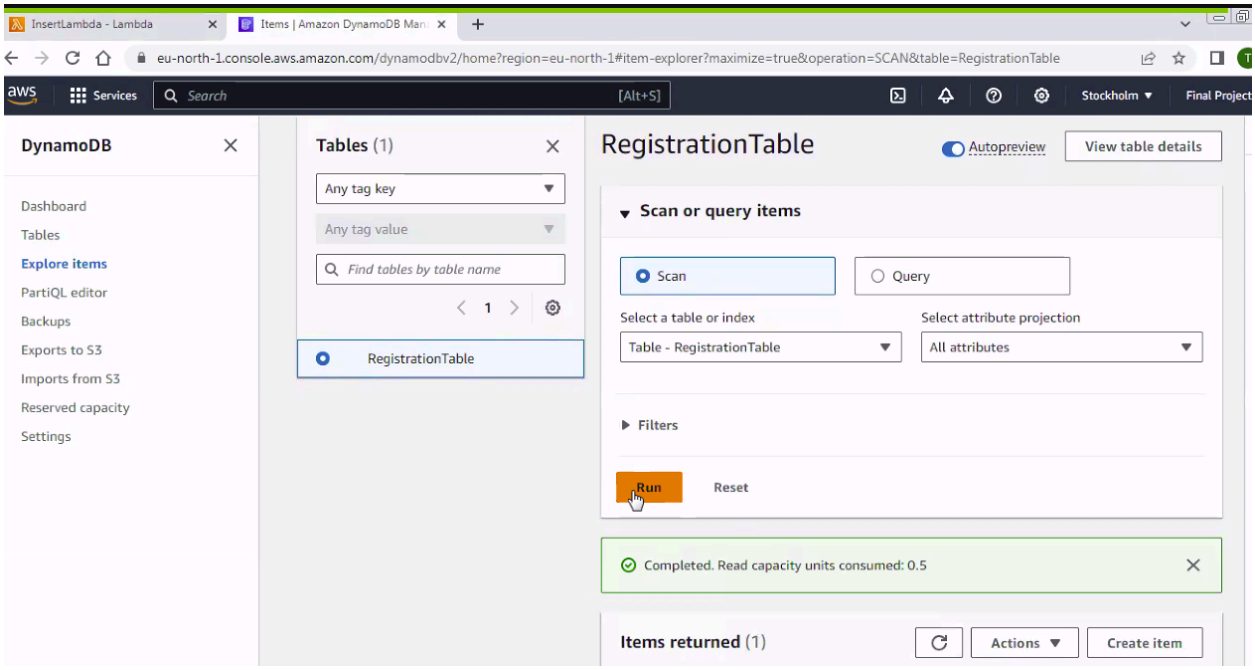


29. On DynamoDB click on ‘explore table items’ button on the right:

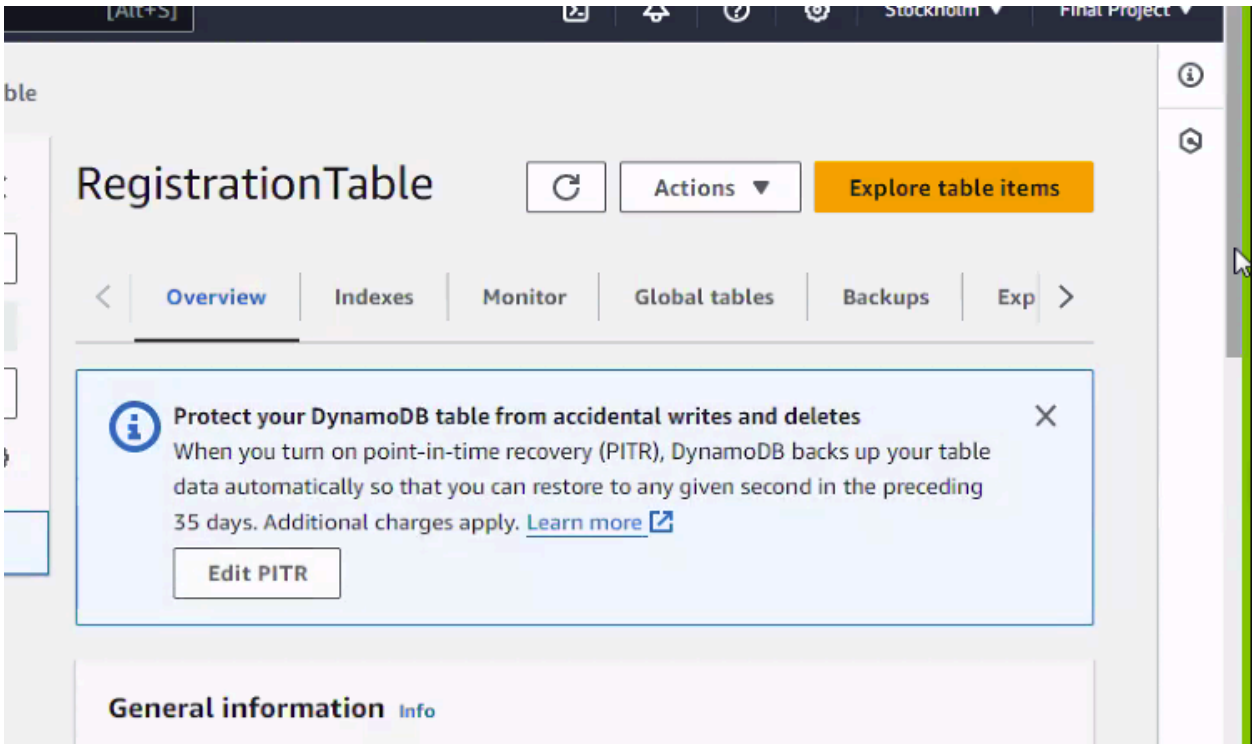




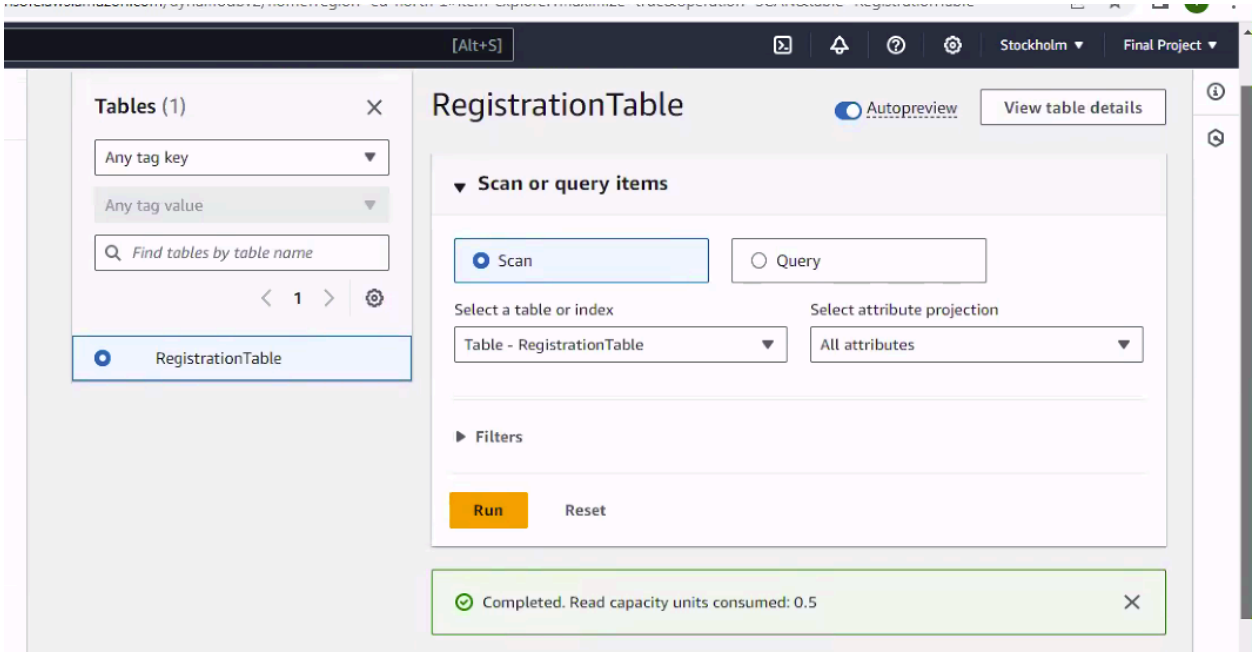
30. Press ‘Run’



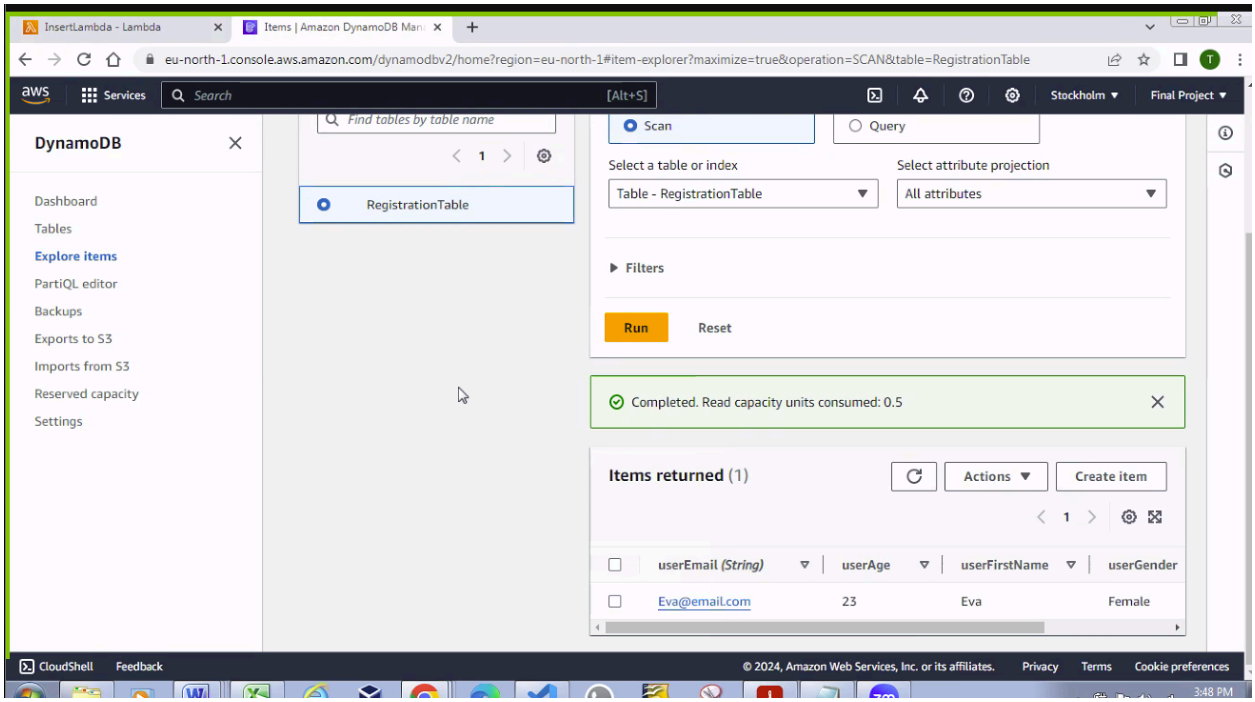
31. [this may be repeat from step 29] Click ‘explore table items’



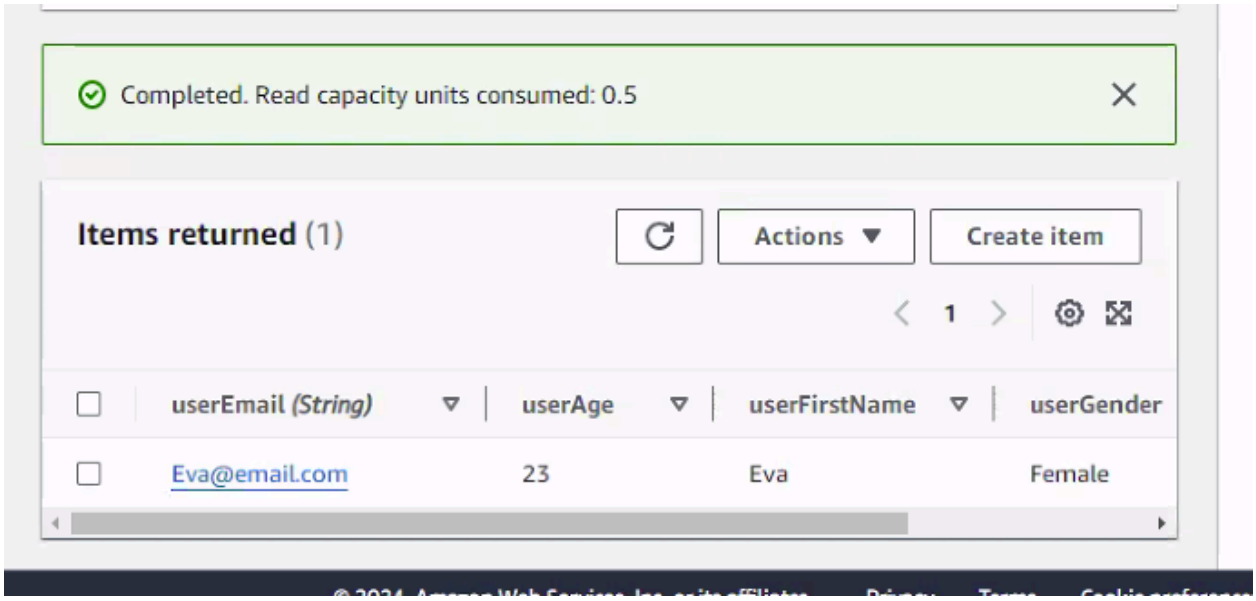
32. Then press ‘Run’. It should say ‘completed’,



33. Scroll down



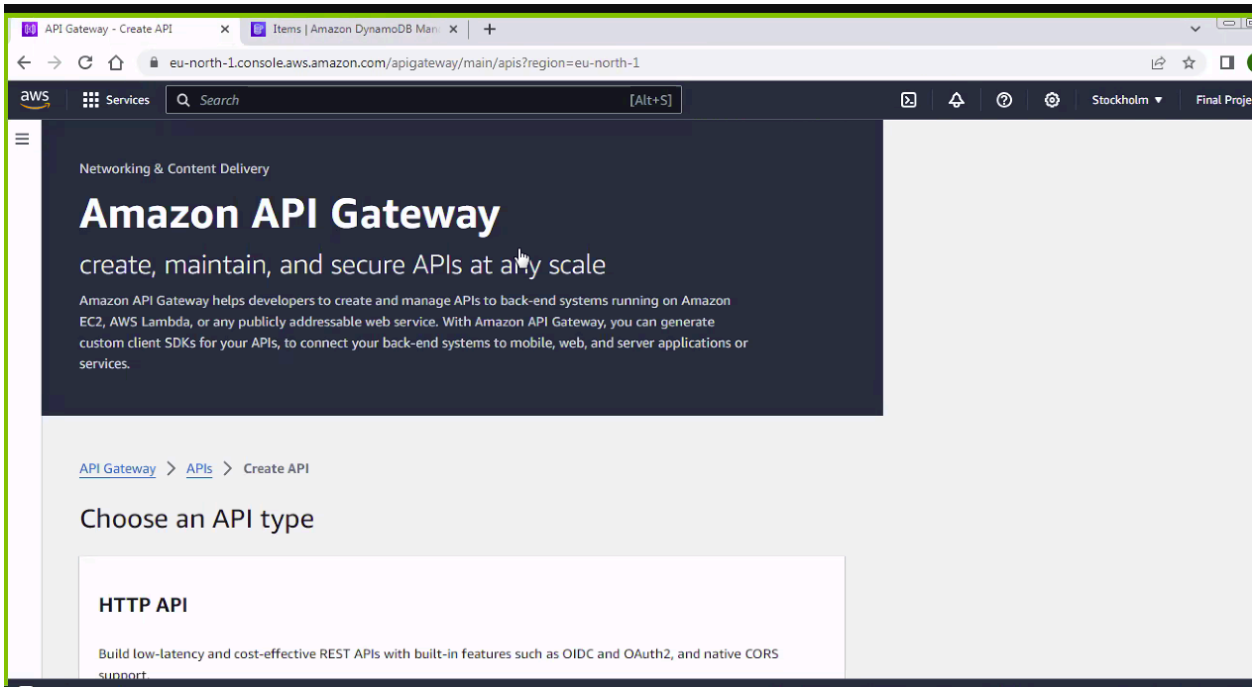
34. Check the 'Items returned' section. There should be one entry there, the email you previously input:



35. Success. This shows that it is being stored in this table.

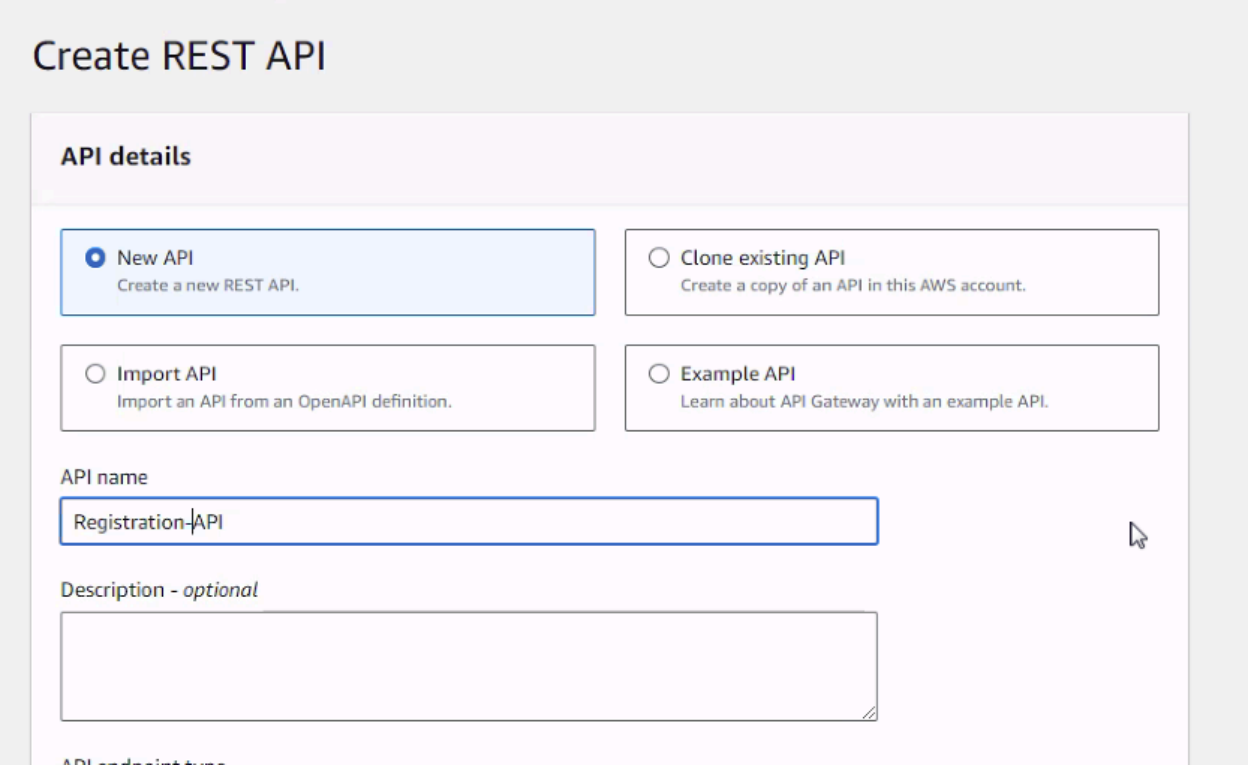
Part 7: API Gateway

1. Search in your console for ‘API gateway’



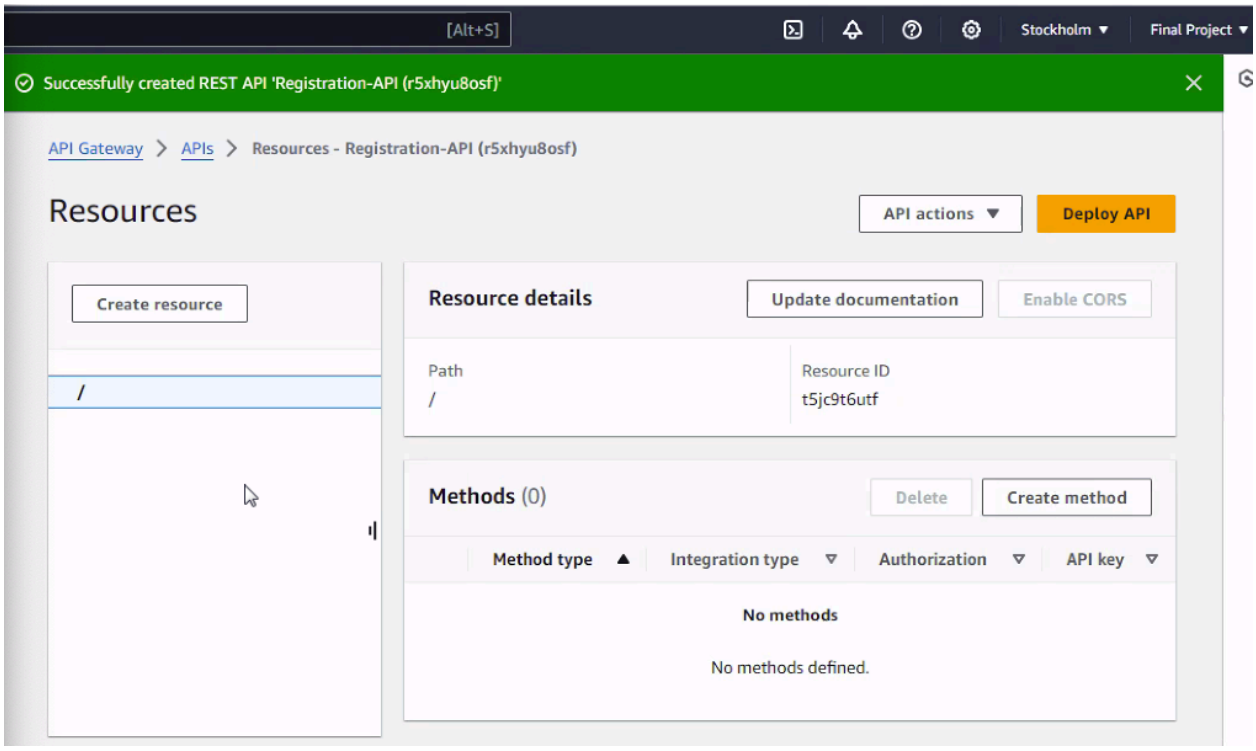
2. Scroll down to REST API
3. Click ‘Build’

4. In the ‘API name’ box, name it ‘Registration-API’

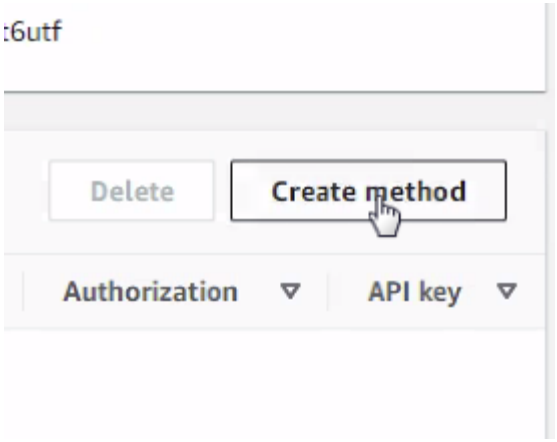


5. Go down and click ‘Create API’

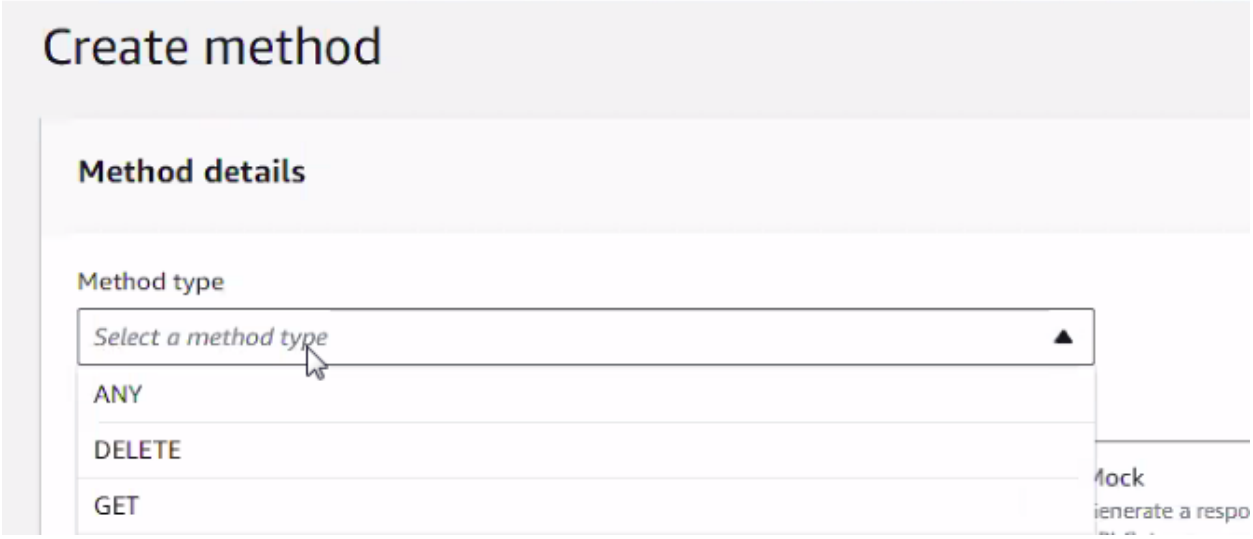
6. Success



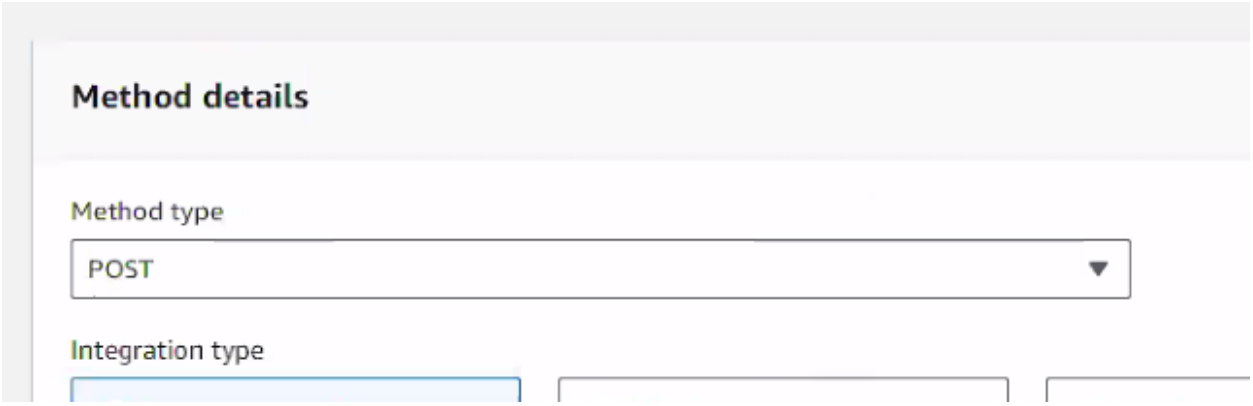
7. Below 'Resource details' in the 'Methods' section, click 'Create method' button on the right side



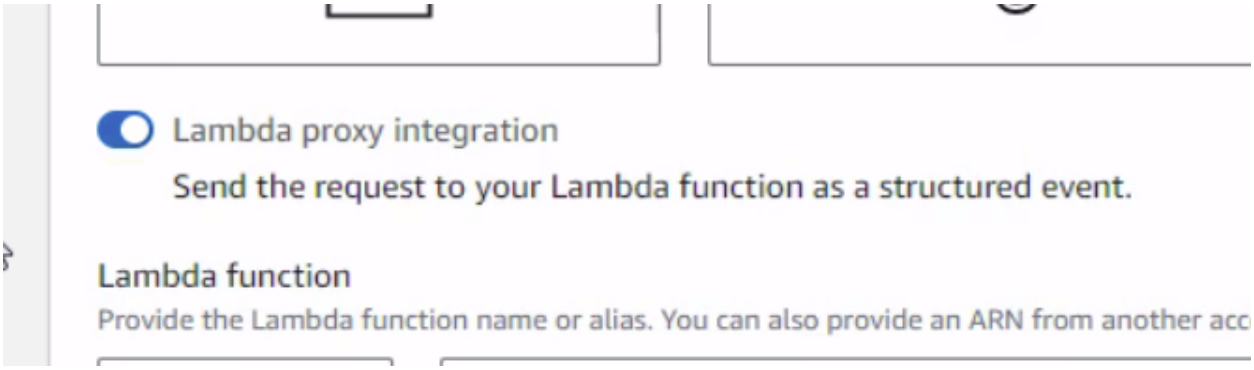
8. In the ‘Create method’ section, click ‘Method type’ dropdown menu



9. Select ‘POST’

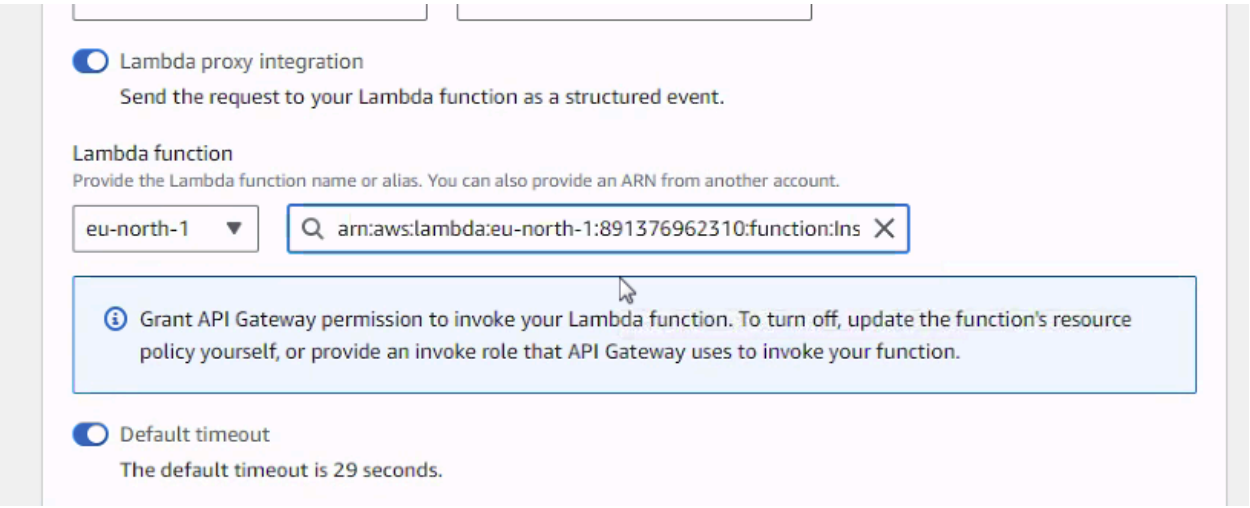


10. Scroll down to ‘Lambda proxy integration’ and click on it to activate it

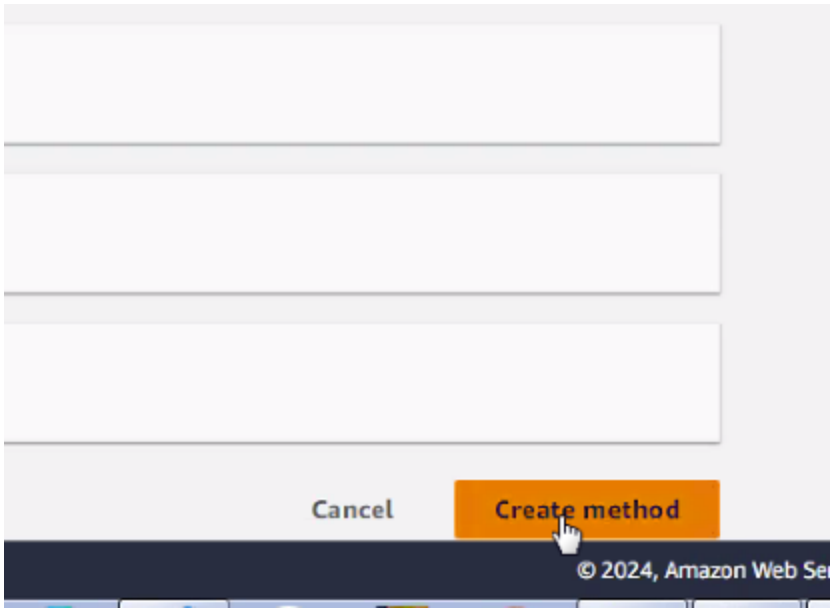


11. In ‘Lambda function’ , click on the search bar and your ARN should show up automatically in the drop down menu

e.g.arn:aws:lambda:eu-north-1.891376962310:function.....

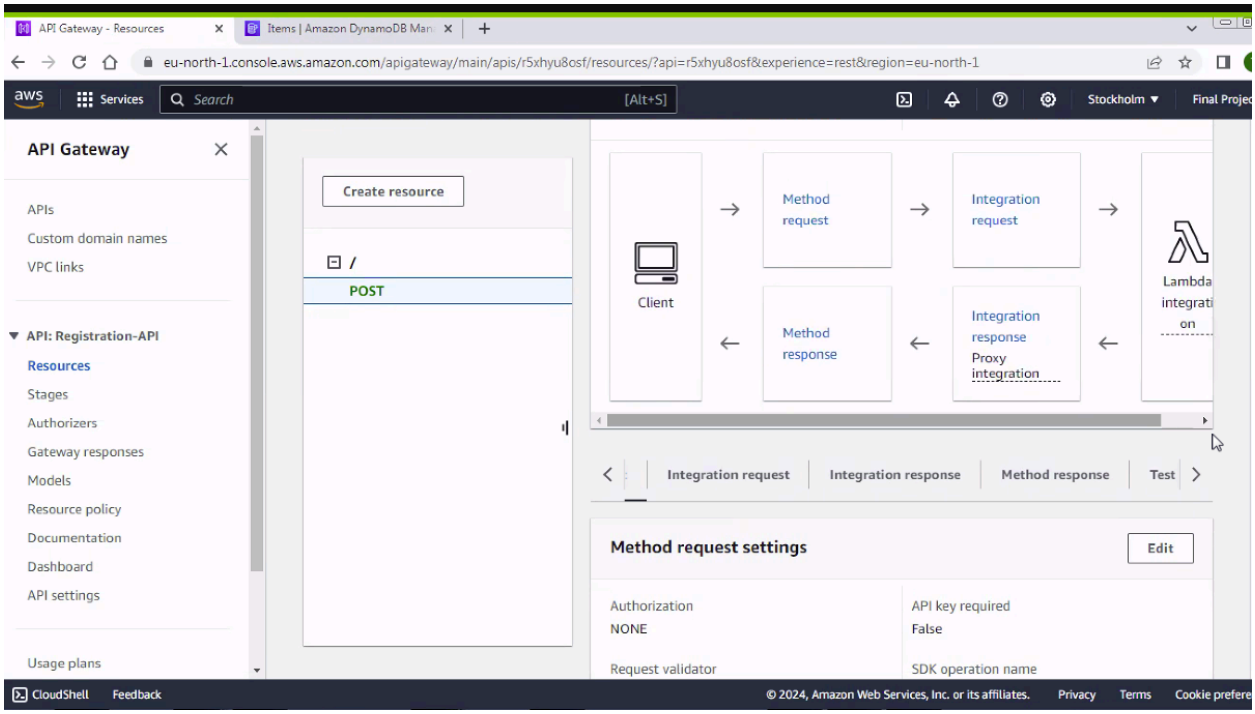


12. Go to the bottom and click 'create method'

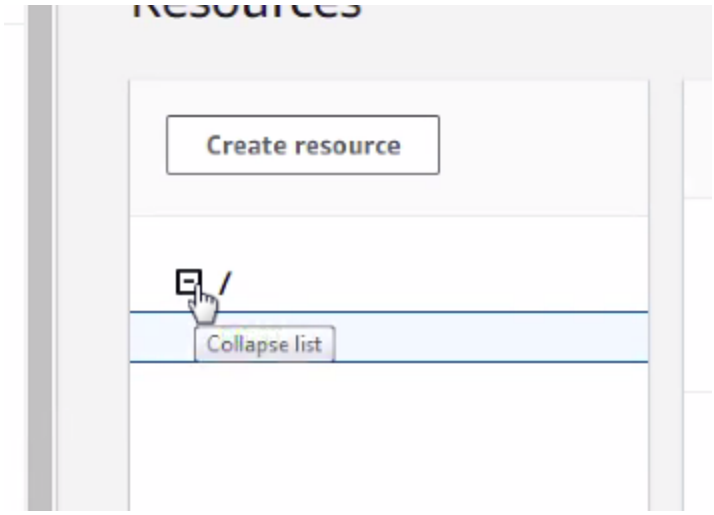


13. Success

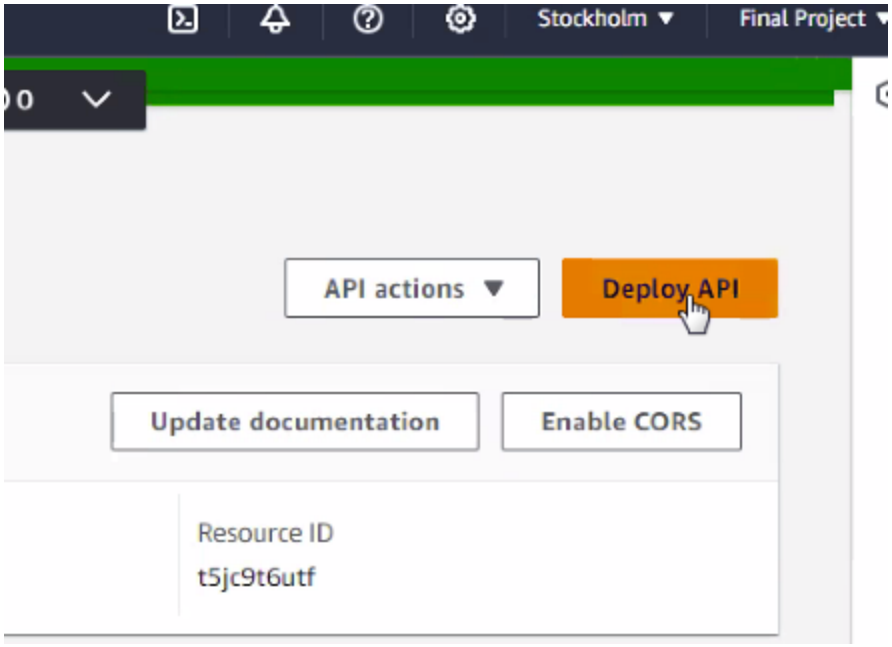
14. Go back to API gateway page on console



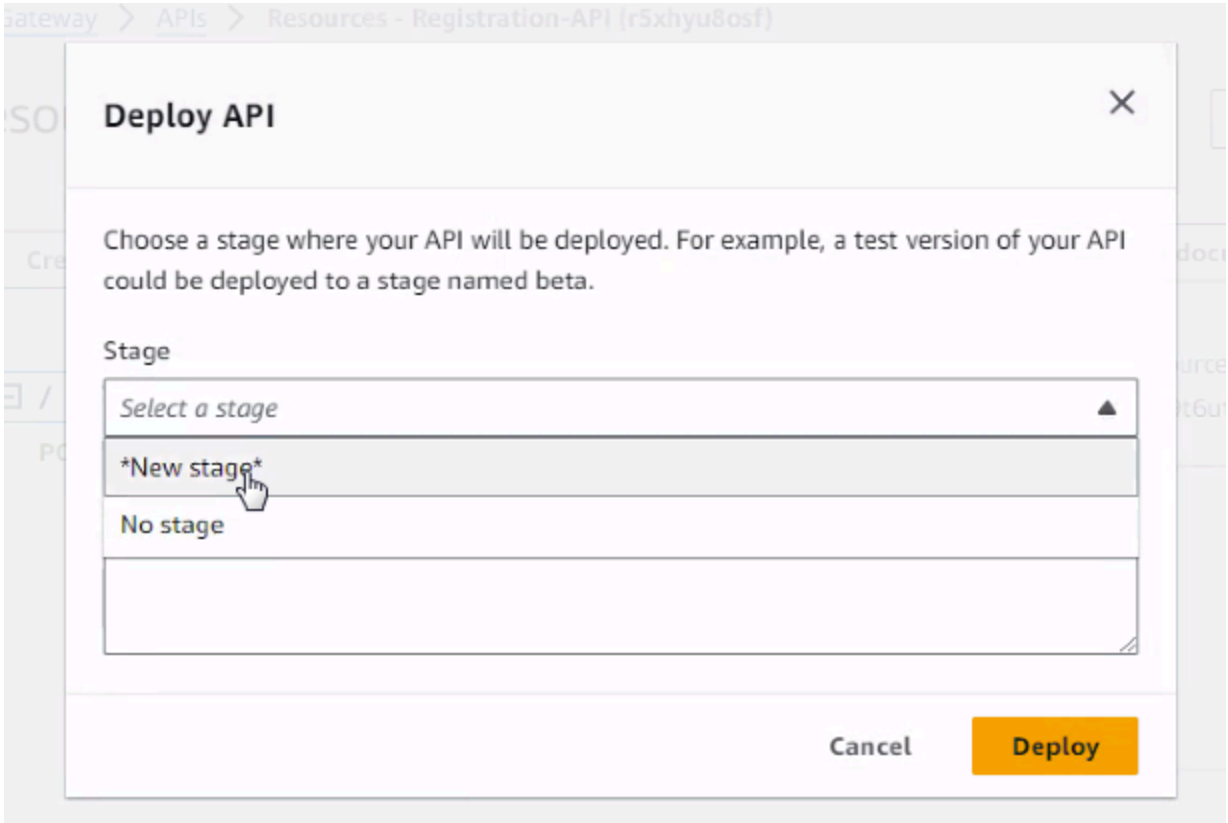
15. Below the 'create resource' button, click on the "-" sign to collapse the list



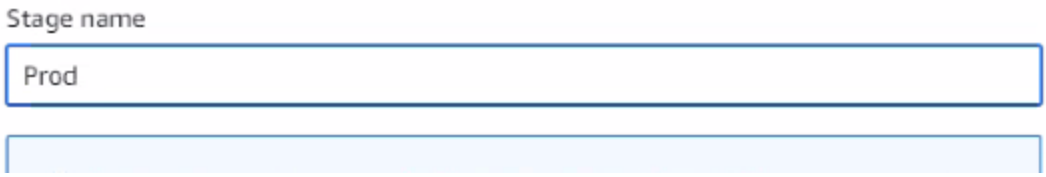
16. Click ‘Deploy API’ on the right side



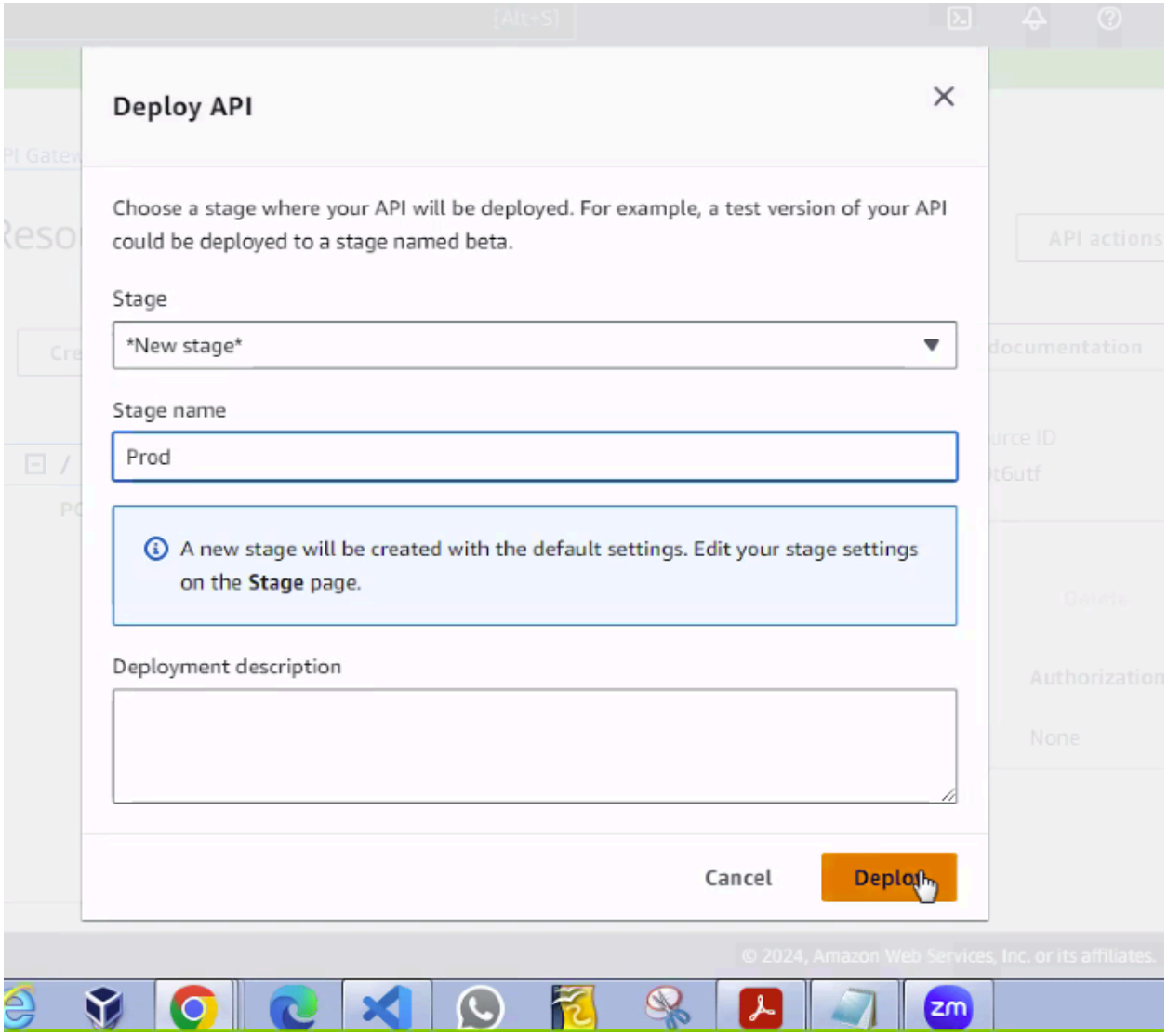
17. In the ‘Stage’ box, select ‘New stage’ in the drop down menu



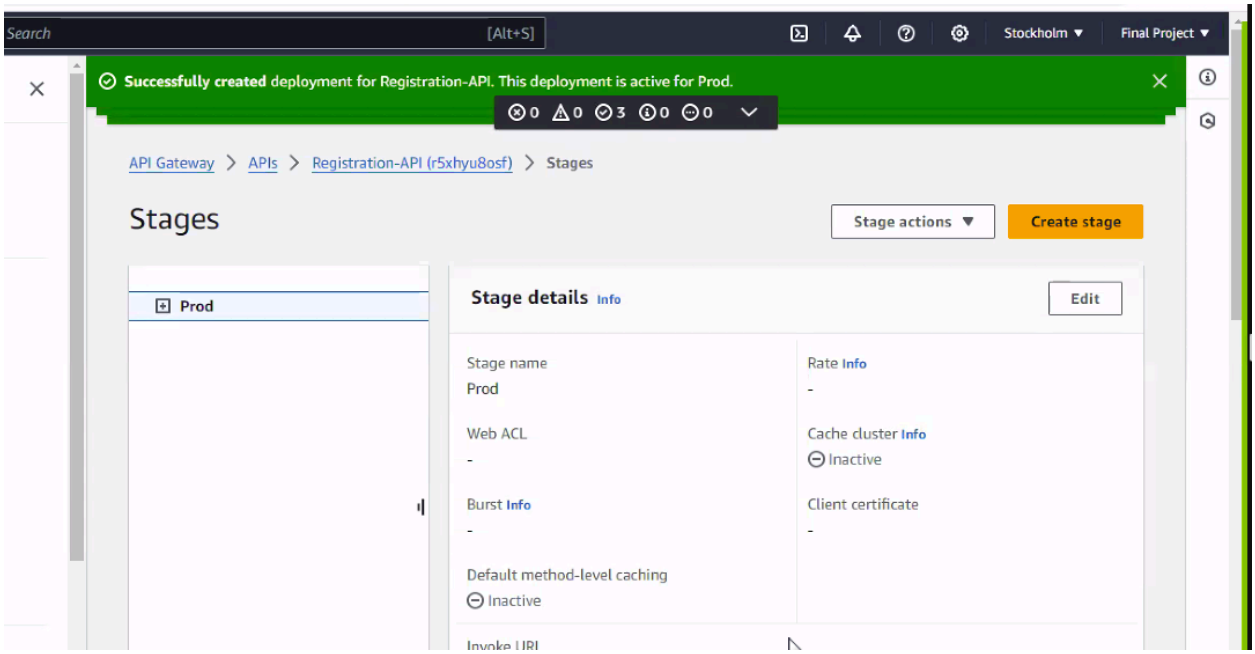
18. Under ‘Stage name’ type in “Prod”



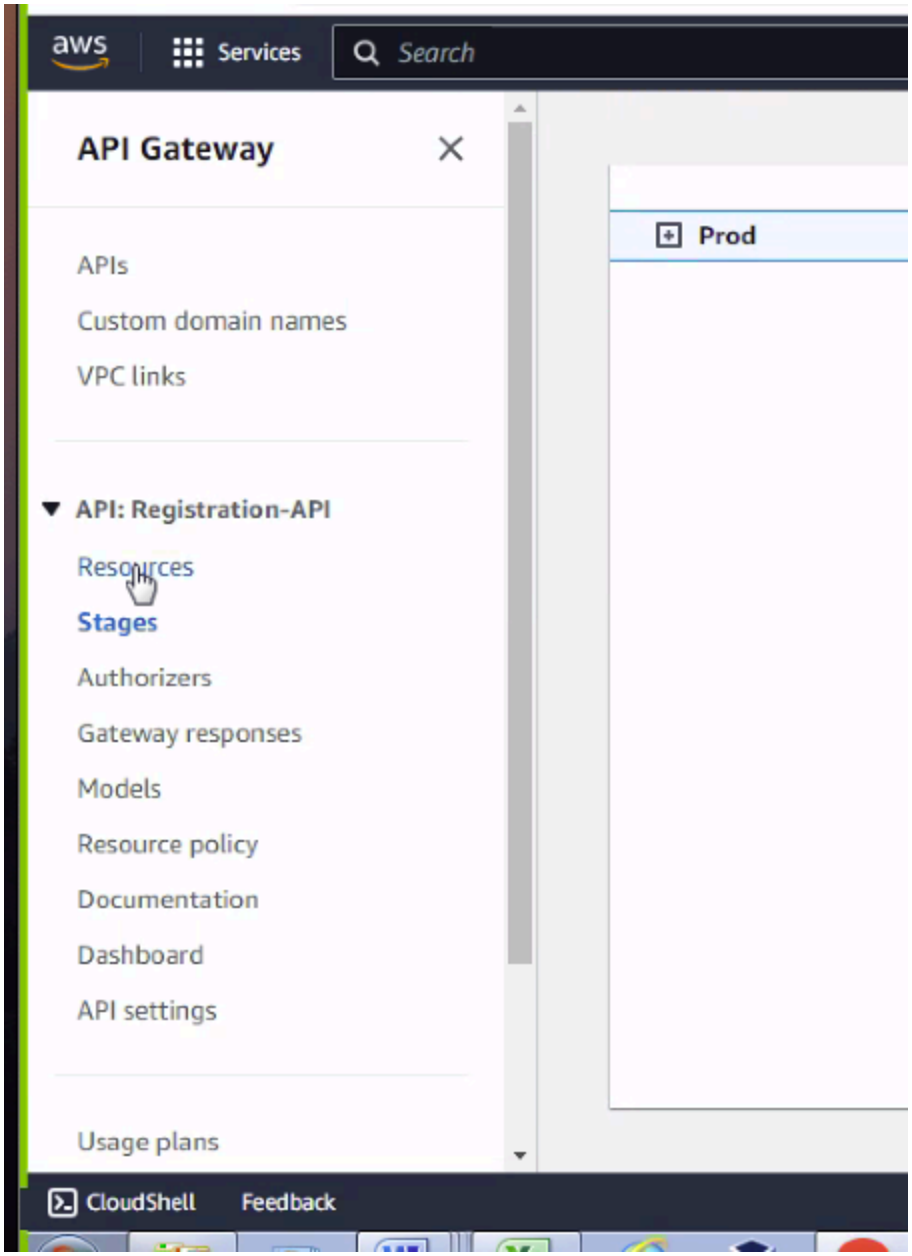
19. Then click on the ‘Deploy’ button



20. Success

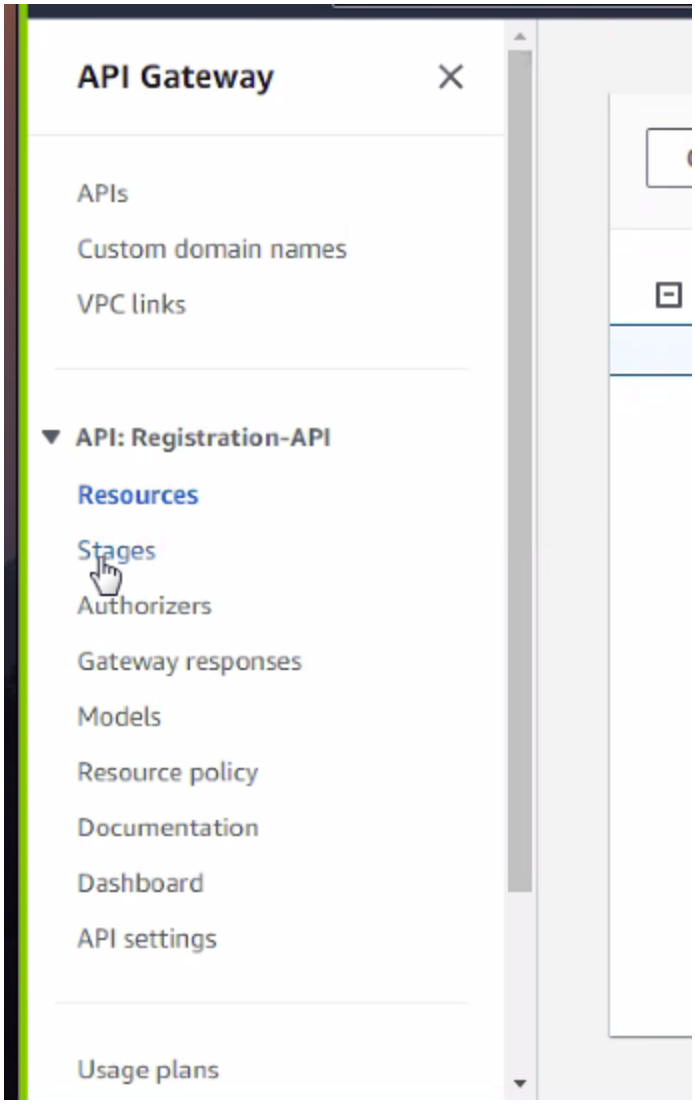

Search
[Alt+S]
Stockholm
Final Project
Successfully created deployment for Registration-API. This deployment is active for Prod.
API Gateway
APIs
Registration-API (r5xhyu8osf)
Stages
Stages
Stage actions
Create stage
Prod
Stage details Info
Edit
Stage name
Prod
Rate Info
-
Web ACL
-
Cache cluster Info
Inactive
Burst Info
-
Client certificate
-
Default method-level caching
Inactive
Invoke URL

21. Go to the Resources section of the left hand side

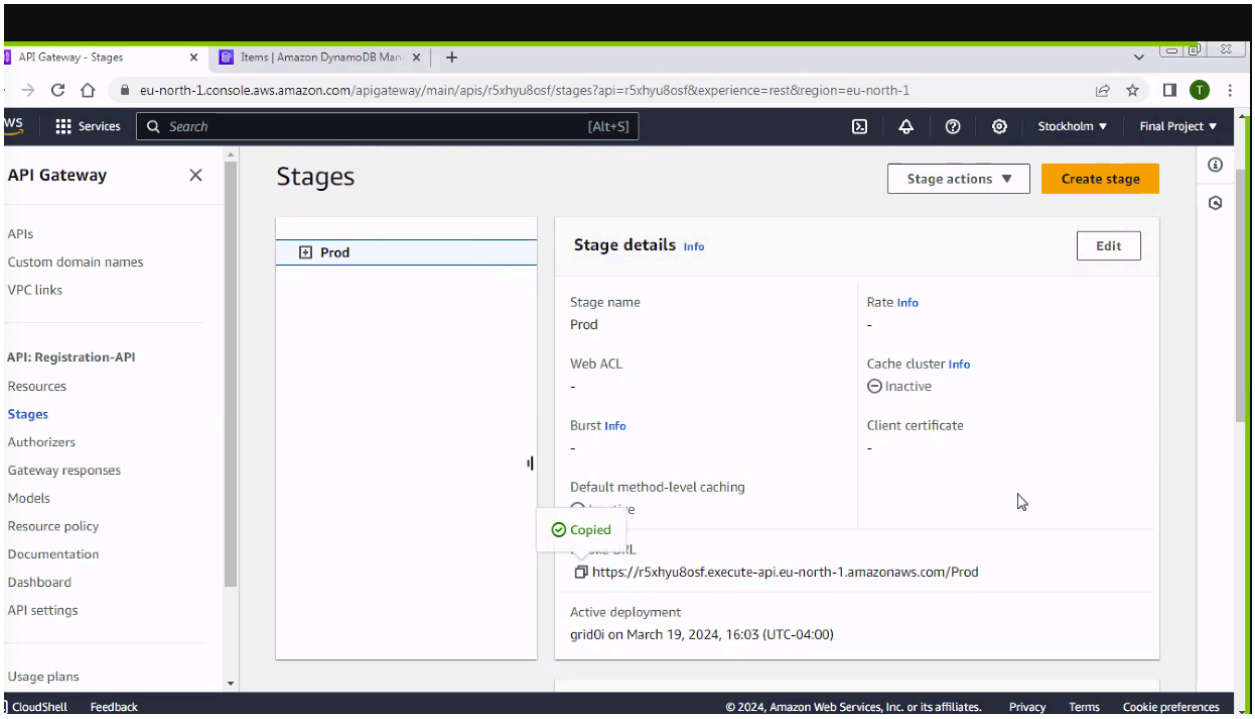


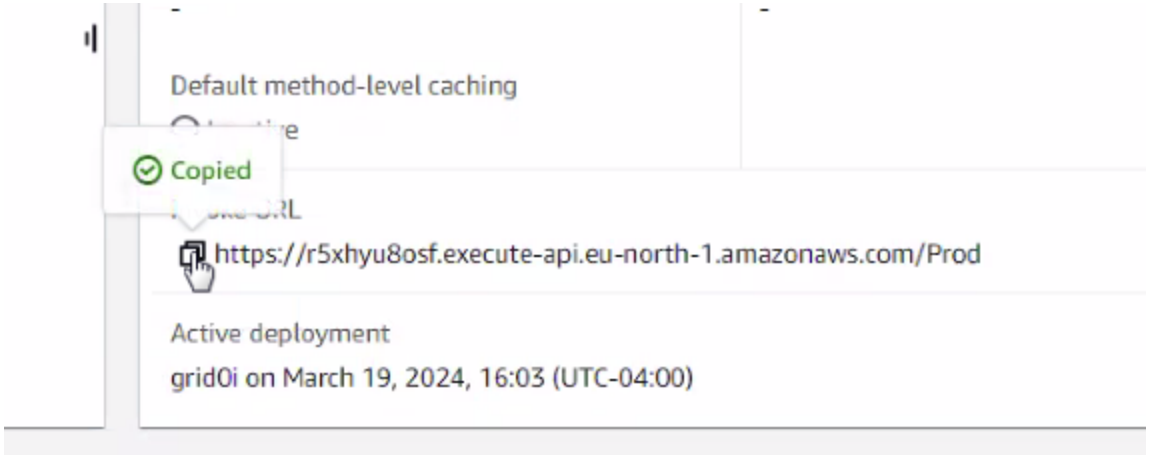
22. Nevermind

23. Click on ‘Stages’ link on the left hand side

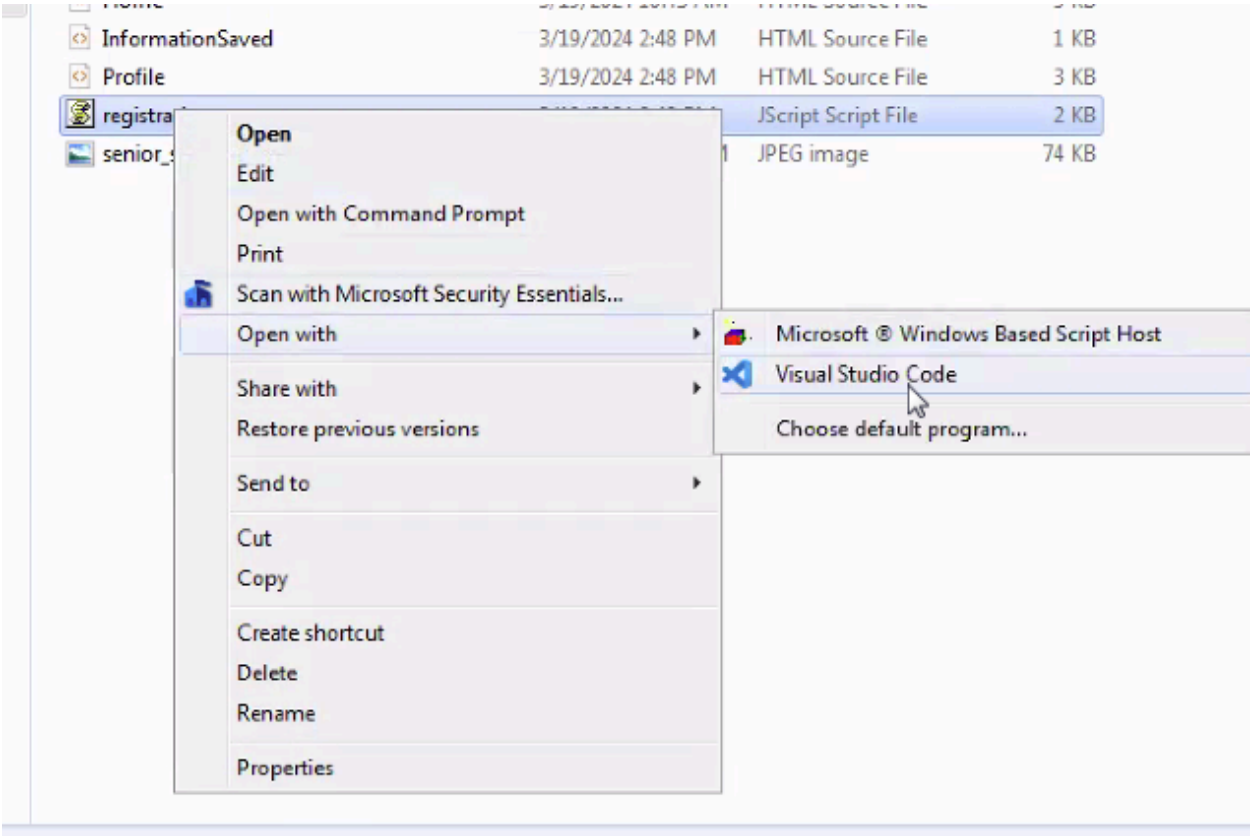


24. In the ‘Stage details’ click to copy the URL

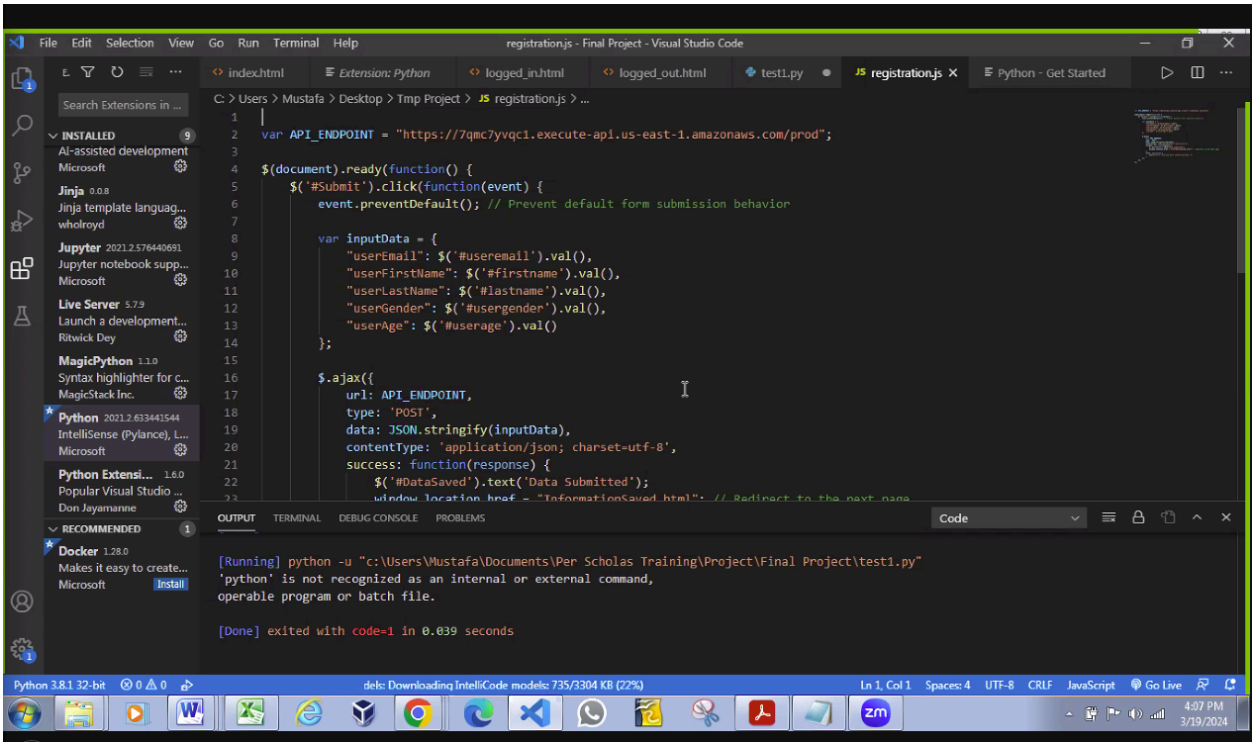




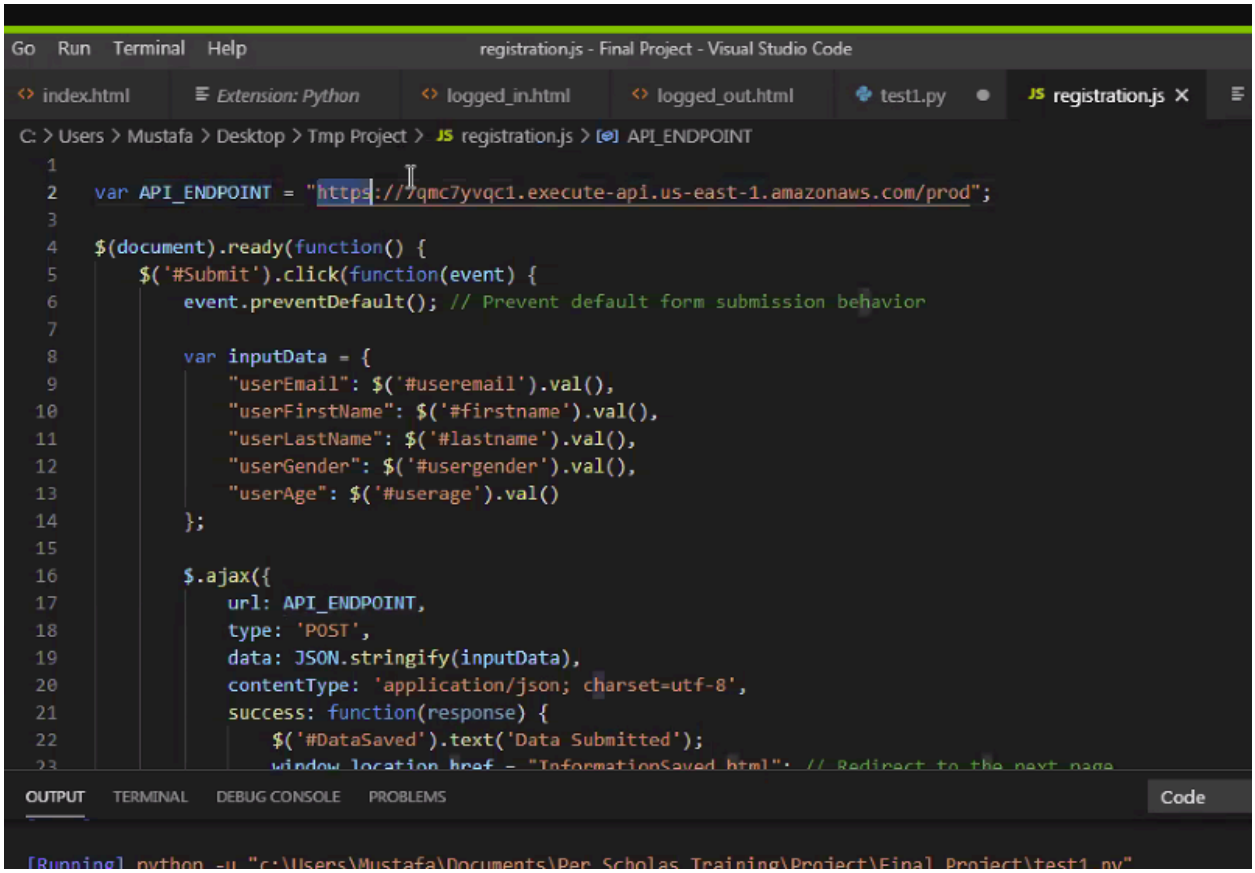
25. Then go to your desktop on your computer and open ‘registration.js’ document with Visual Studio Code



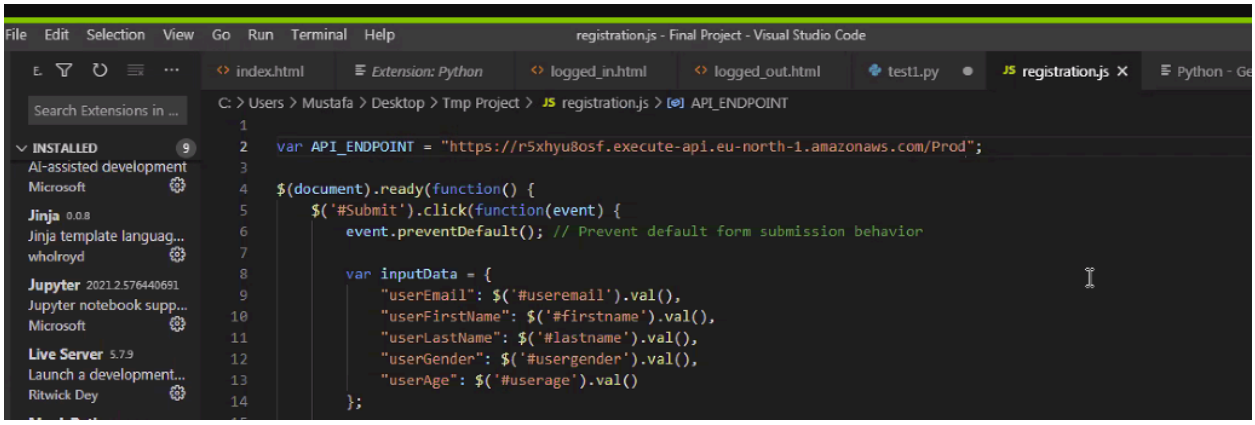
26. Visua Studio Code:



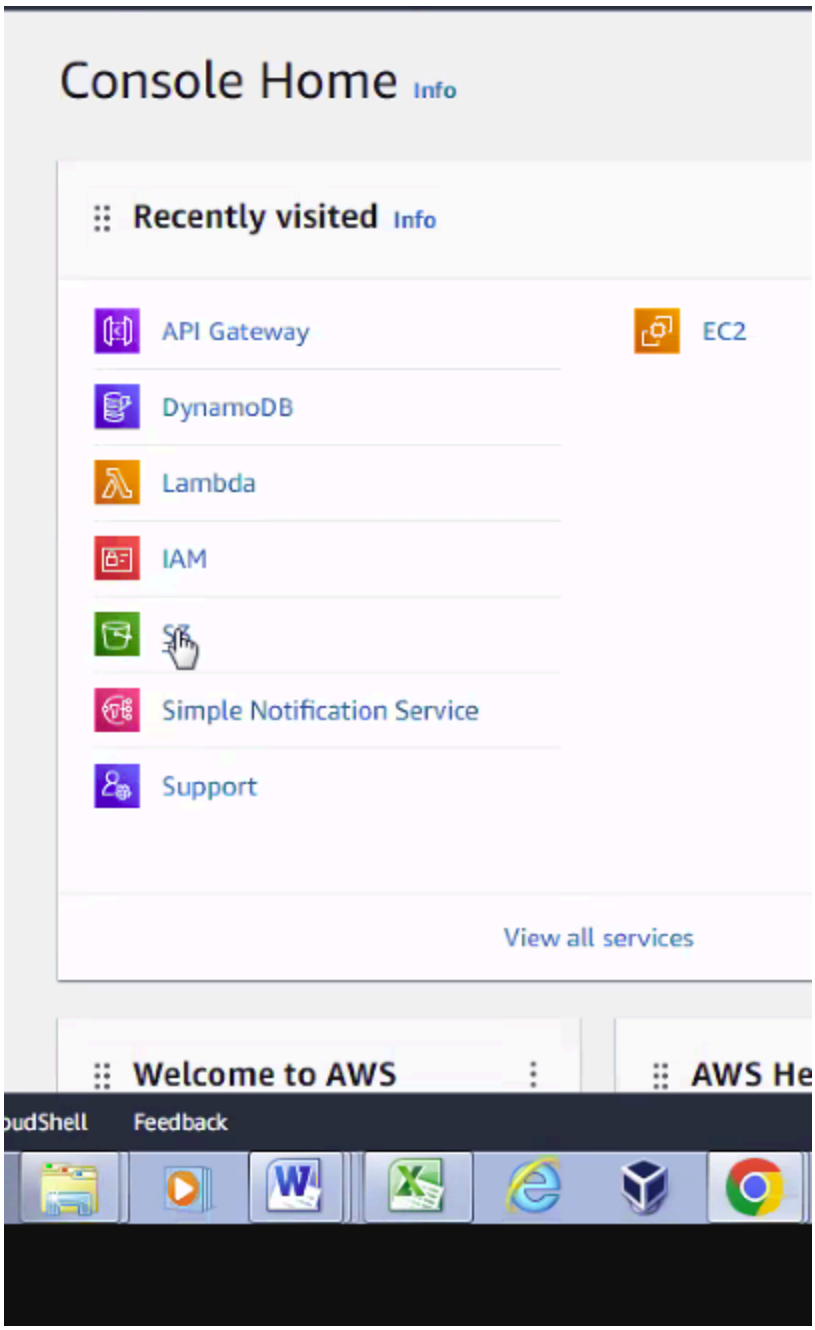
27. Do you see line 2? Replace that default address with your own URL link you copied earlier (e.g. https://r5xhyu8osf.....amazobnaws.com/Prod")
28. Replace the old default code in line 2 with your address:



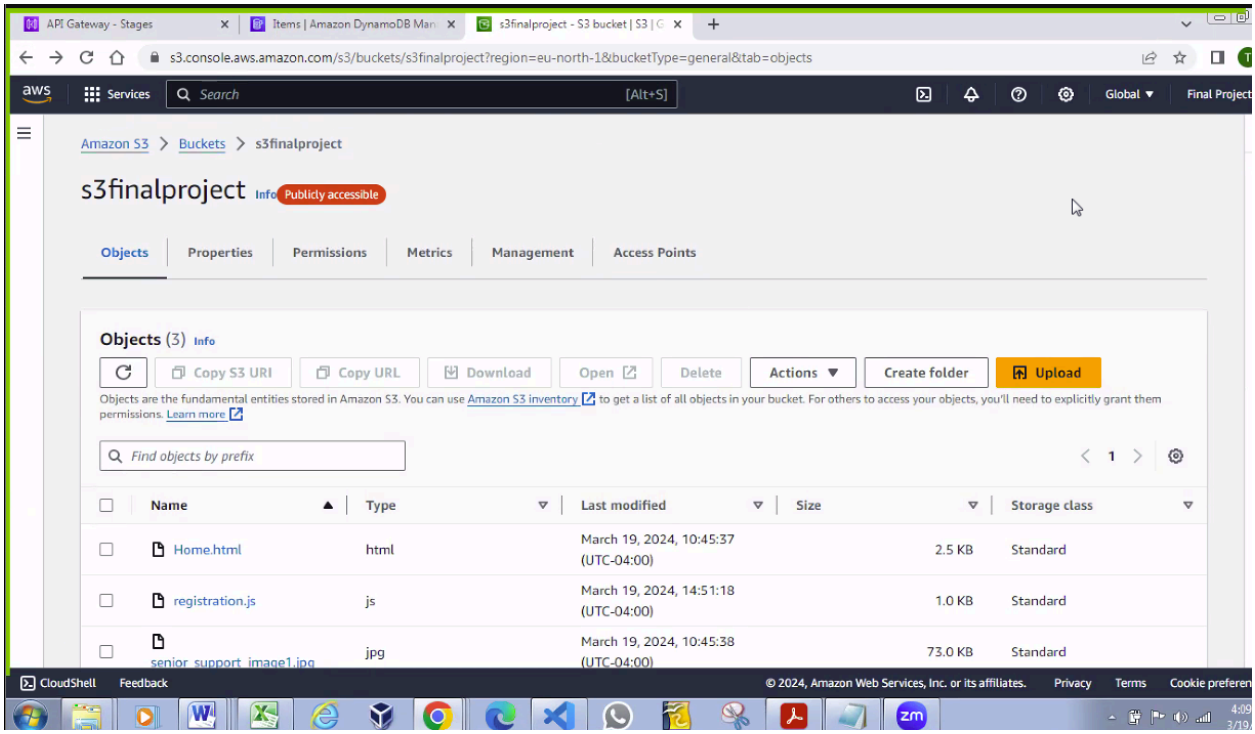
29. See the change?



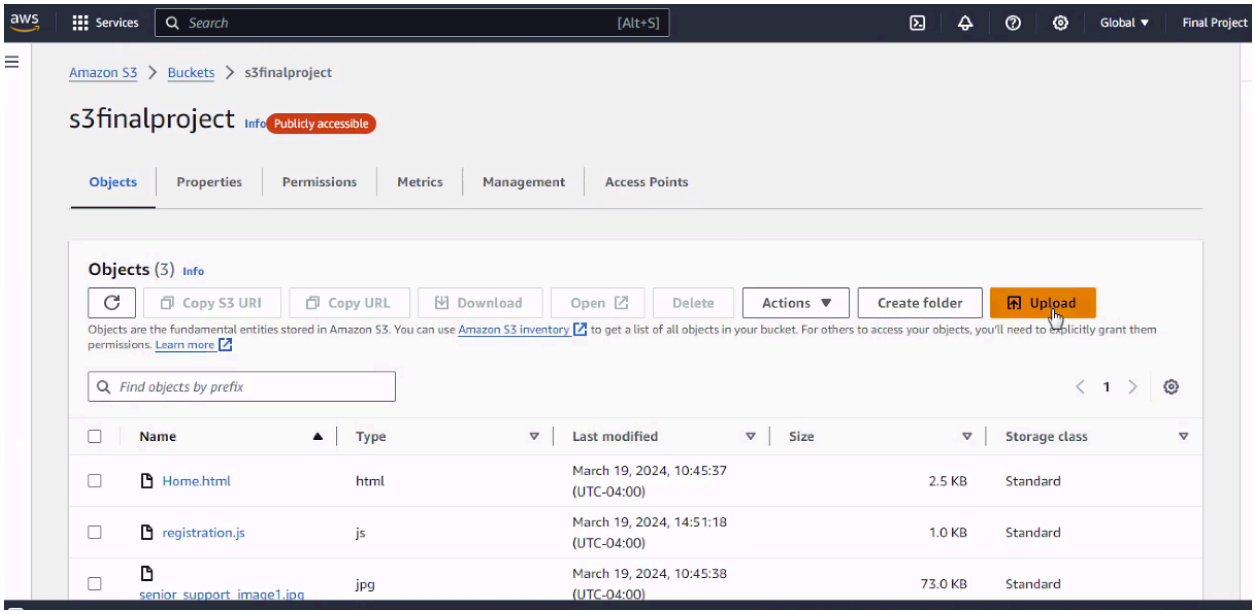
30. Now travel back to your S3 bucket.



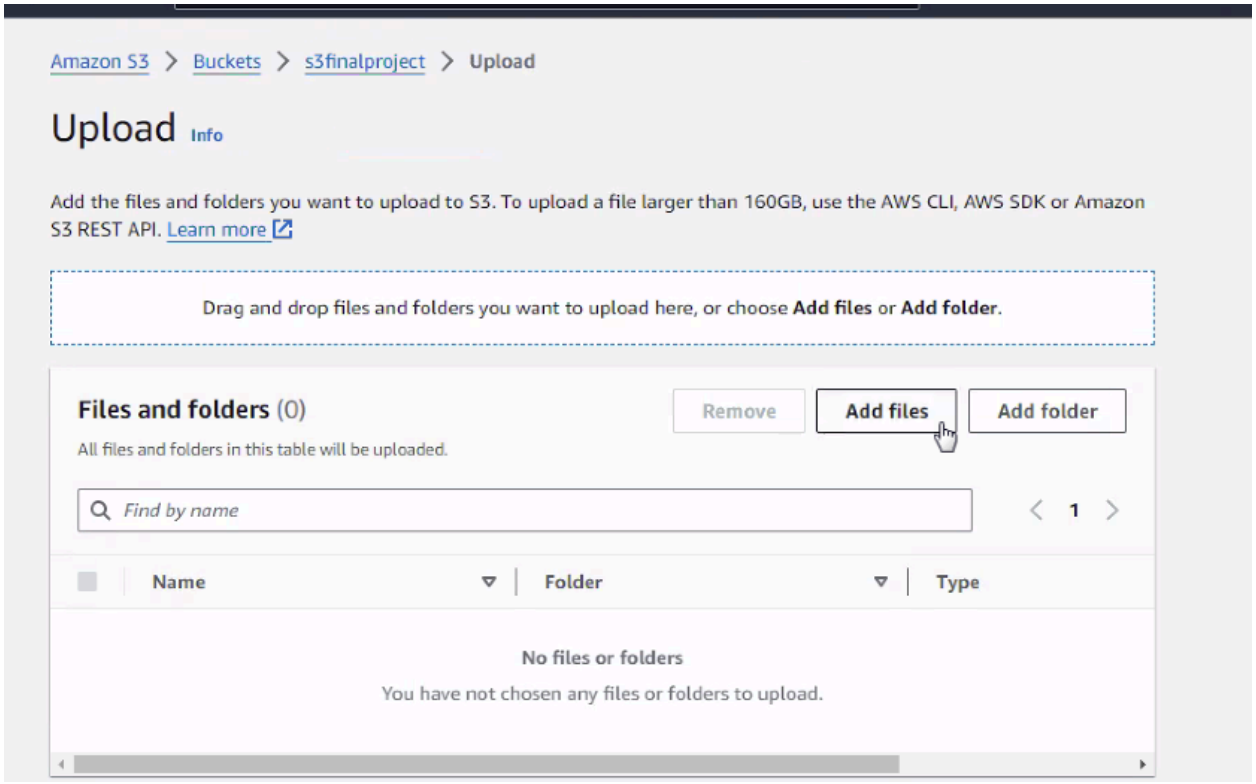
31. Go click on your bucket



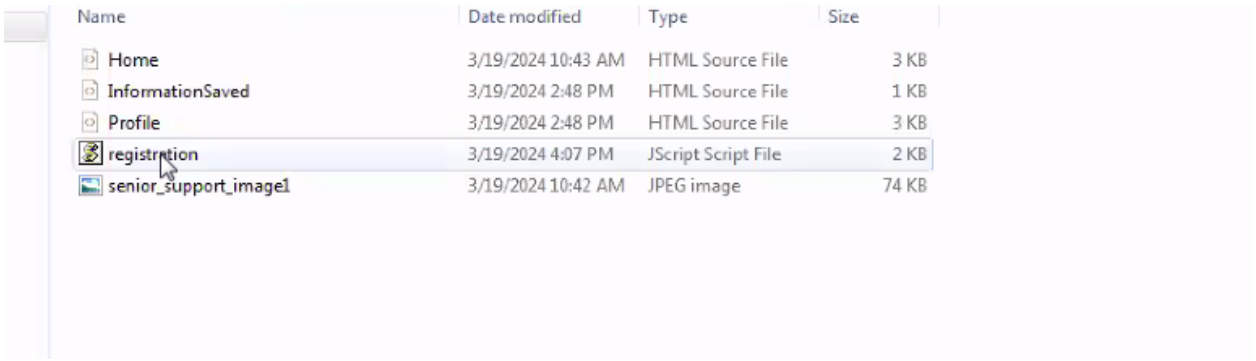
32. On the right side, click on that orange button that says 'Upload'



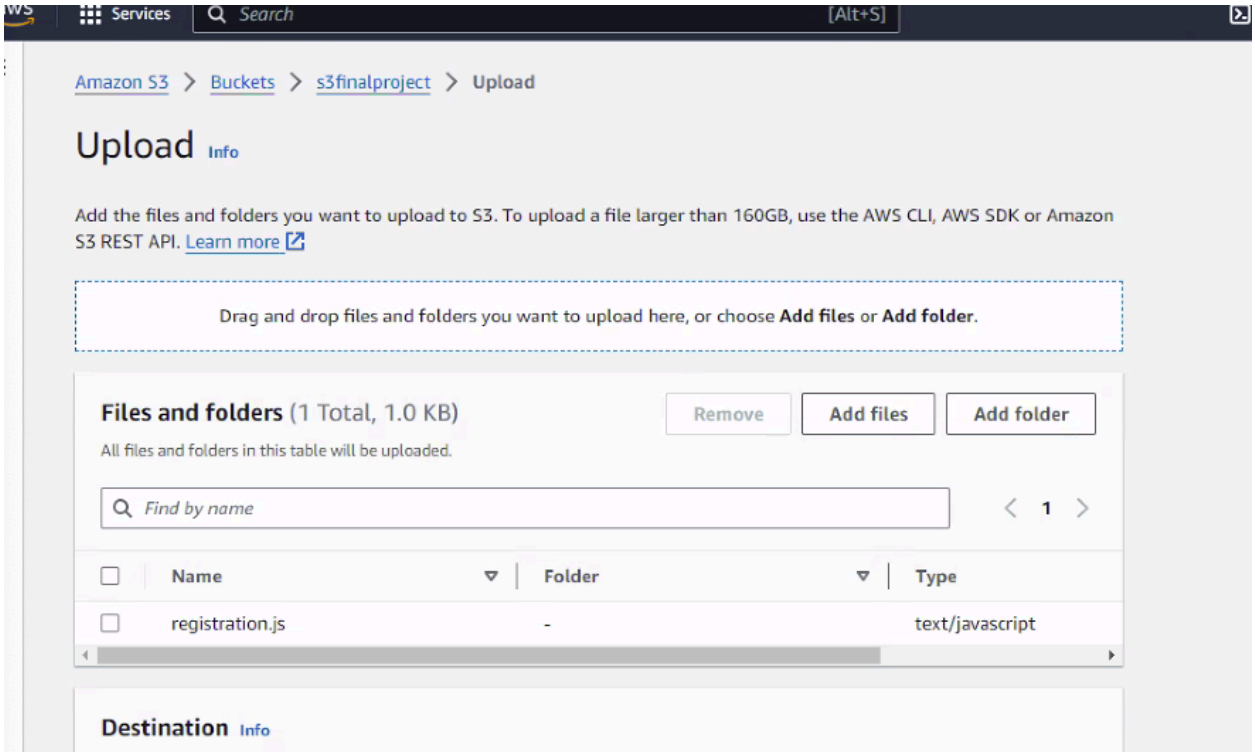
33. In ‘Upload’, click on the ‘add files’ button over to the right:



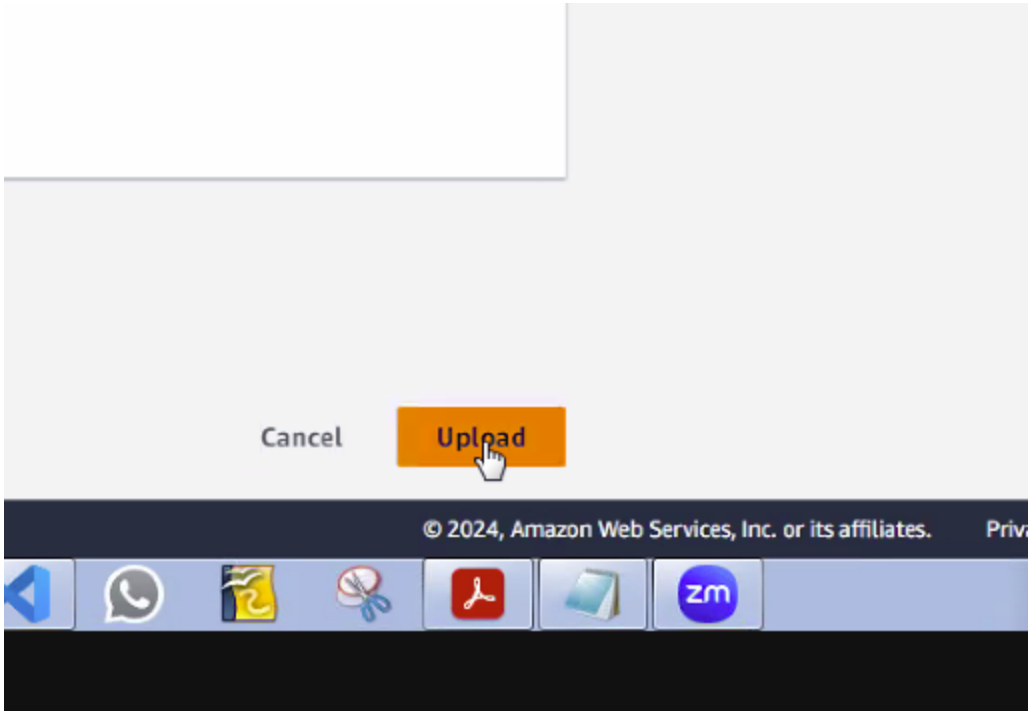
34. Select the ‘registration.js’ file from your computer files



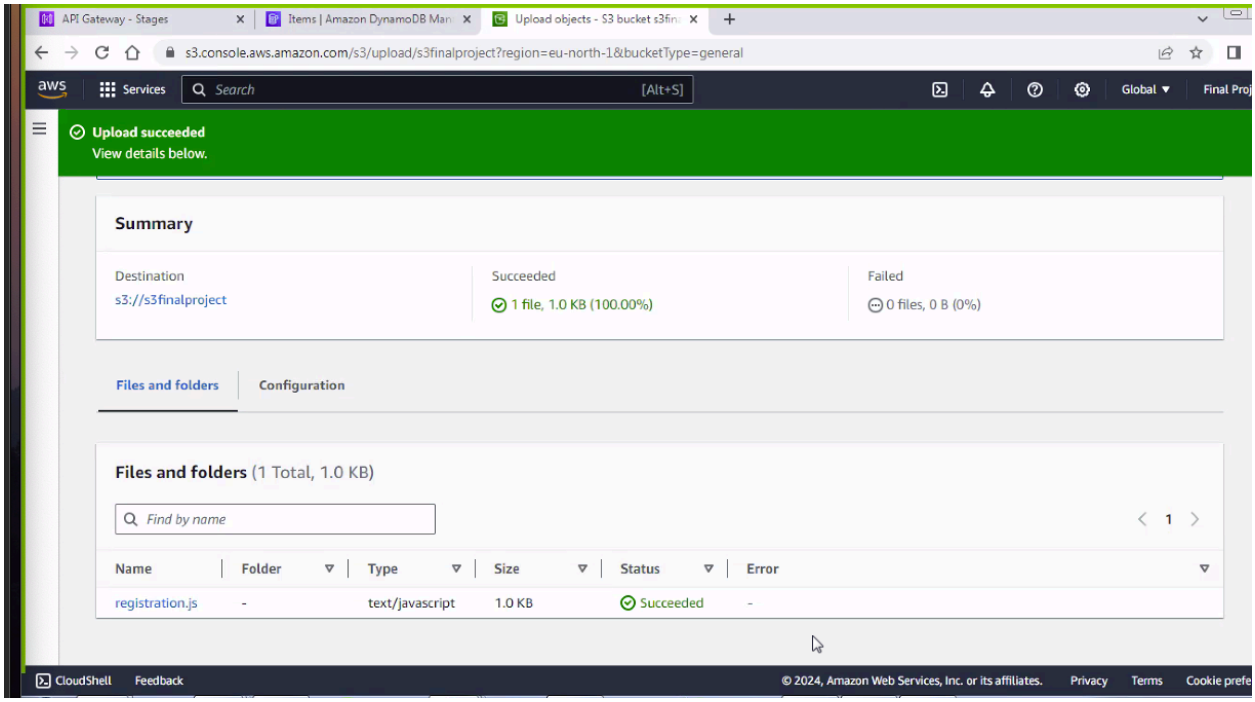
35. Add it



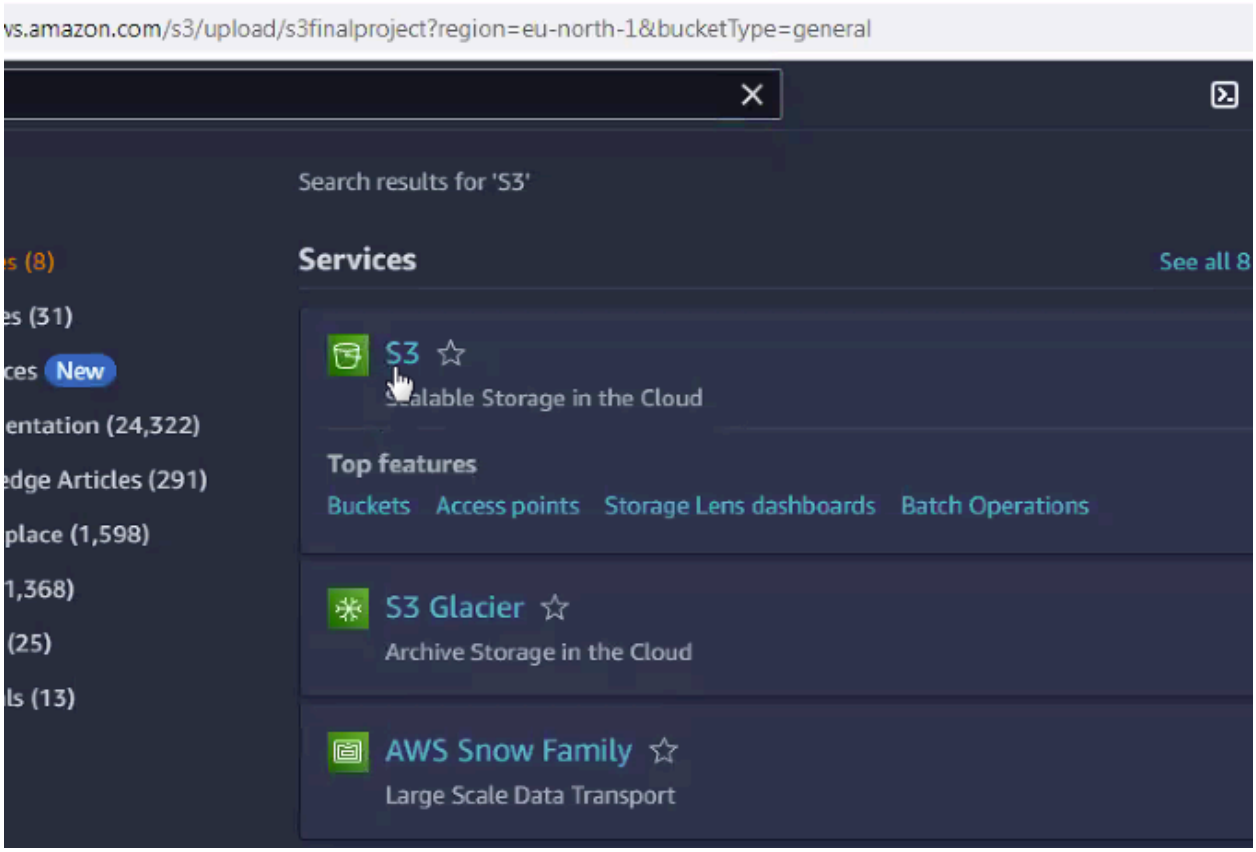
36. Scroll down and click 'Upload'



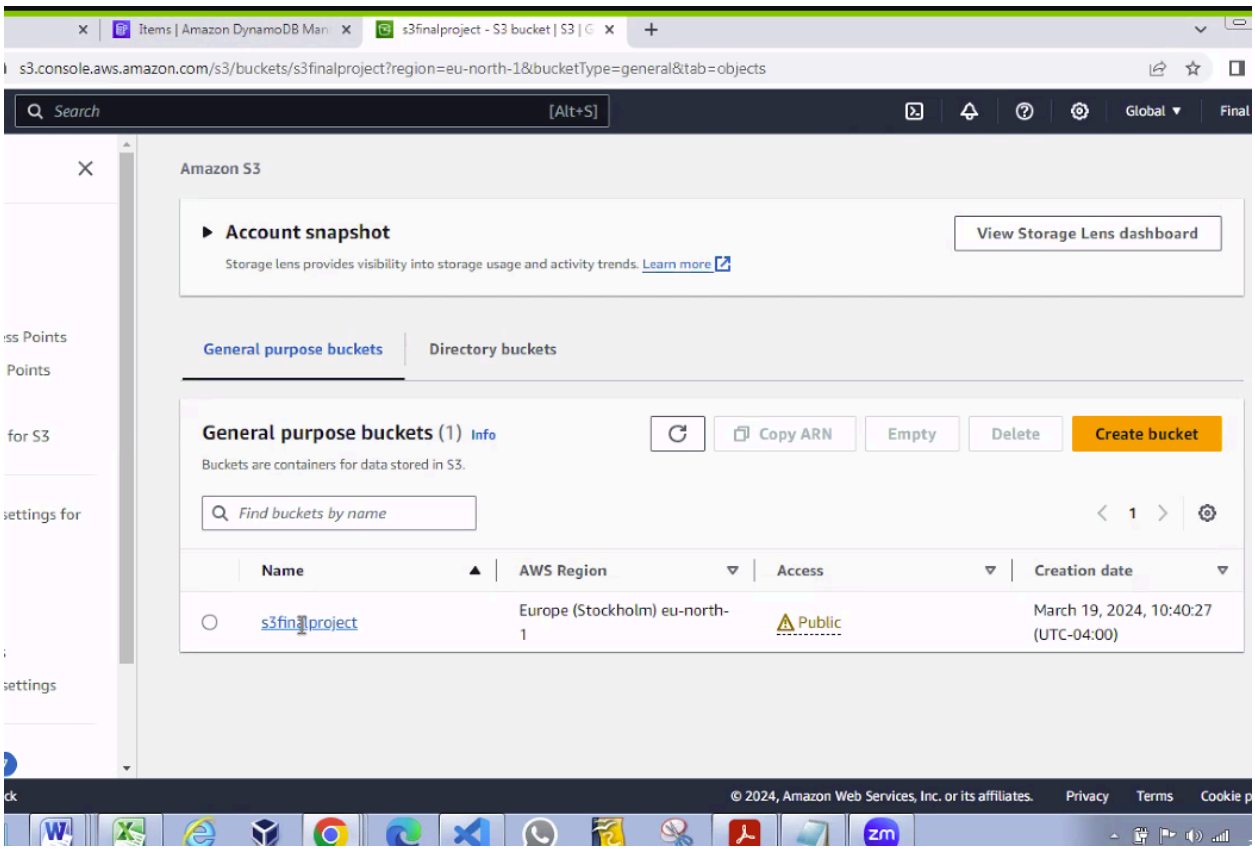
37. Success



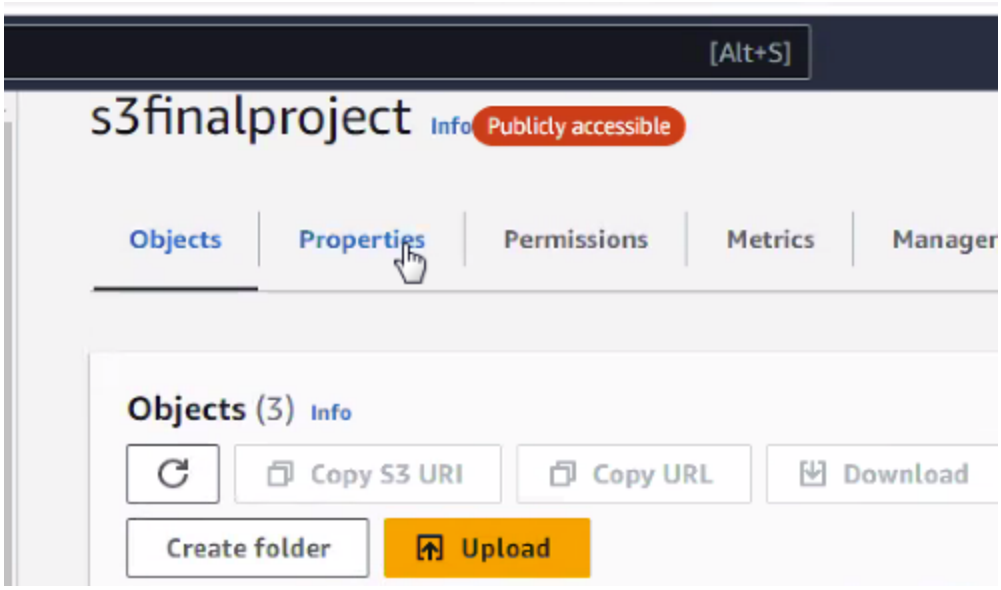
38. In your console search bar on the top, type in 'S3'



39. In S3, click on your bucket link (e.g. s3finalproject)

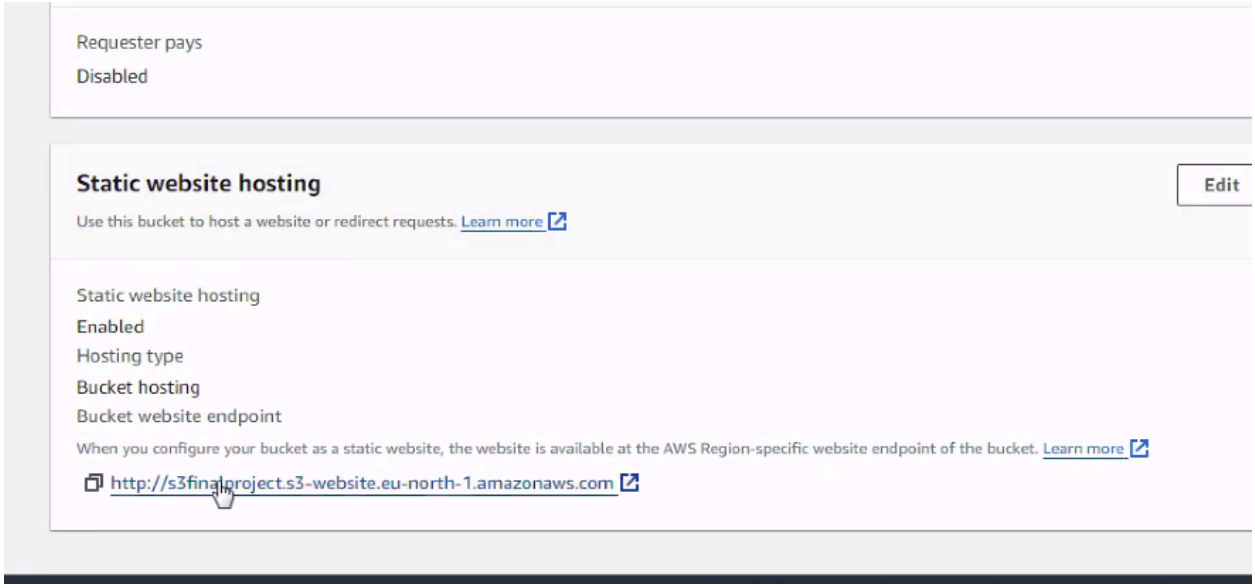


40. While in that bucket, click on the 'Properties' tab

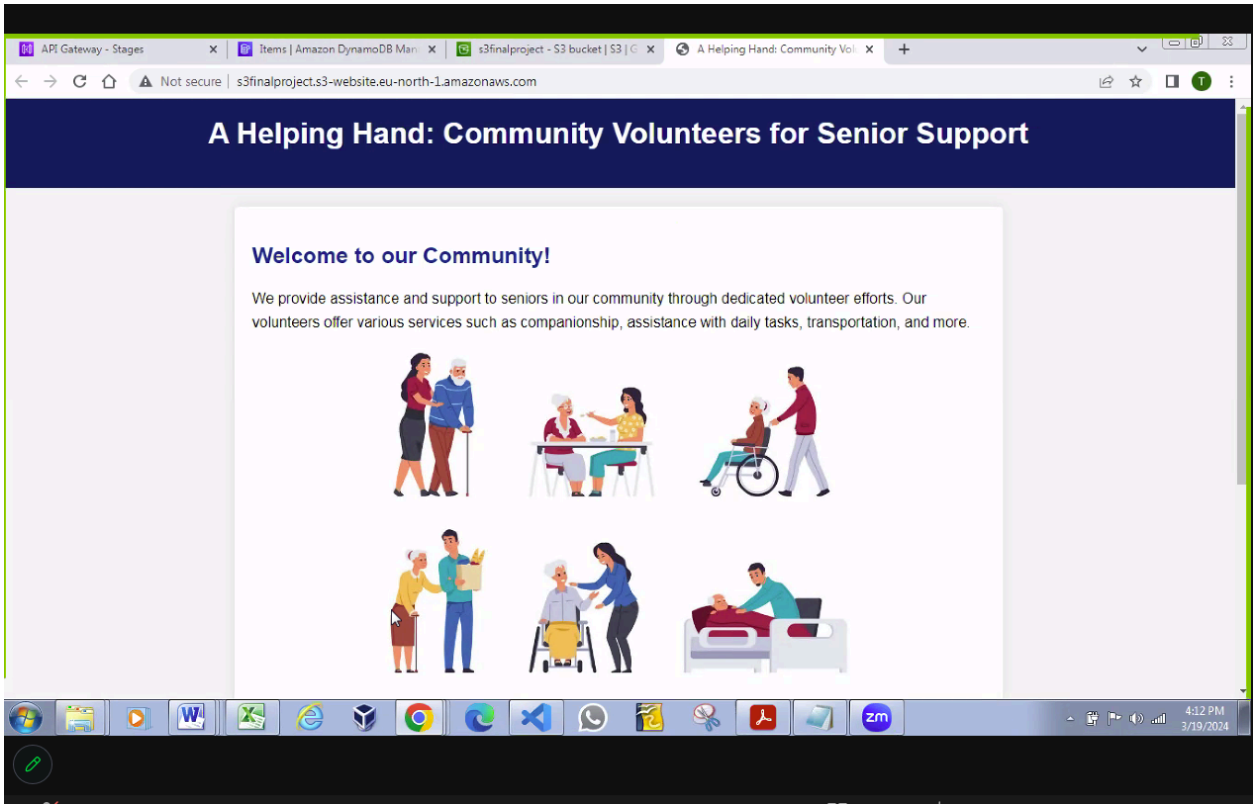


41. Scroll down

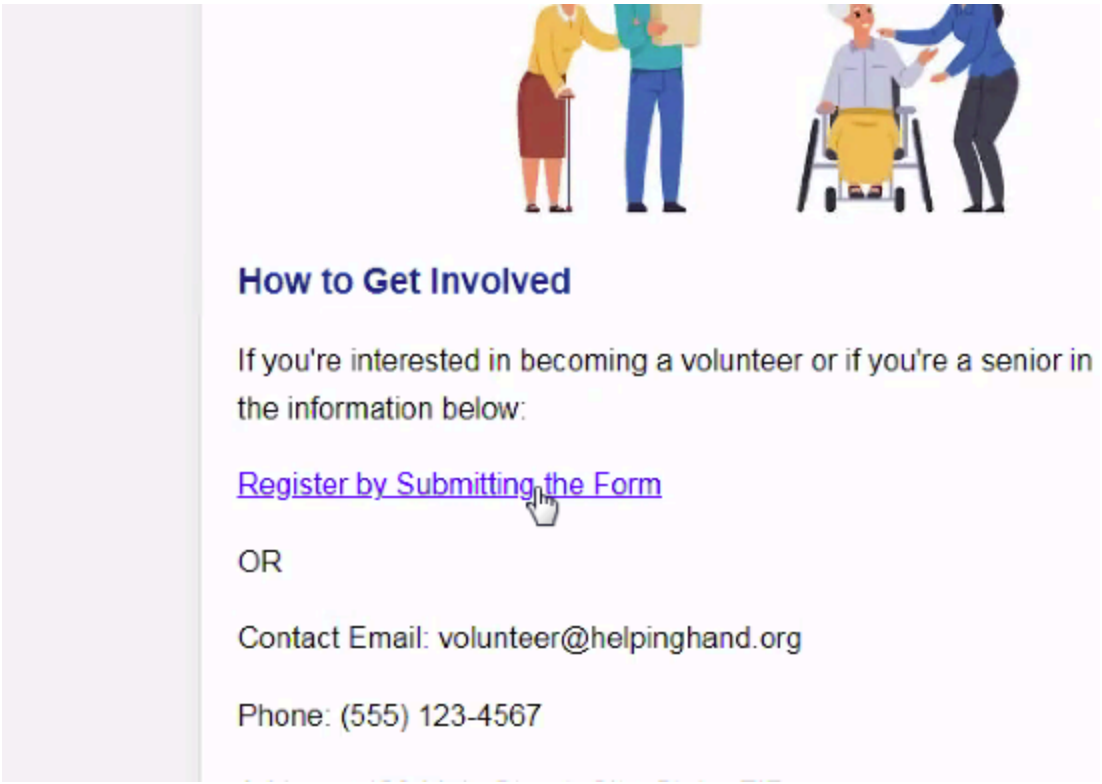
42. In the ‘Static website hosting’, click on the URL link to copy it



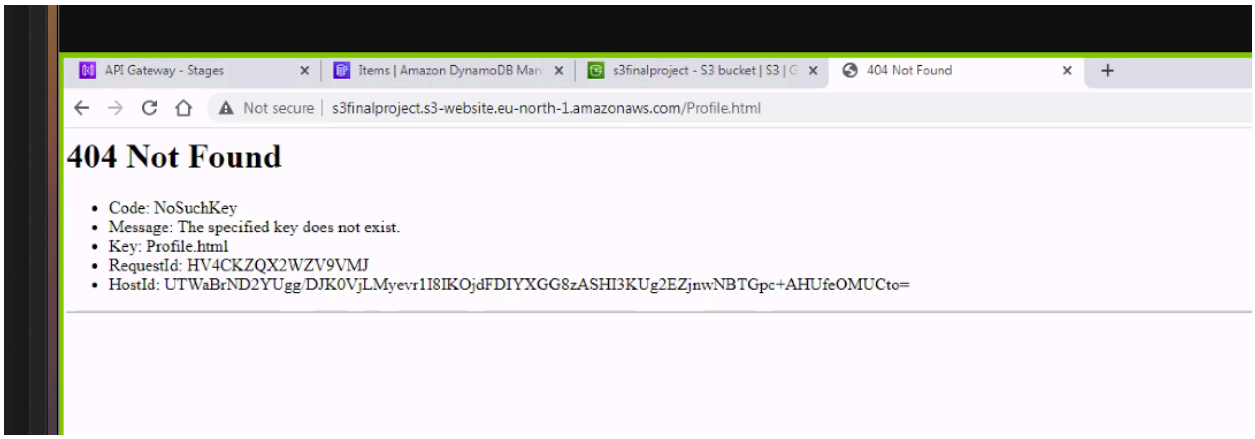
43. You should be taken to a webpage that looks like this:



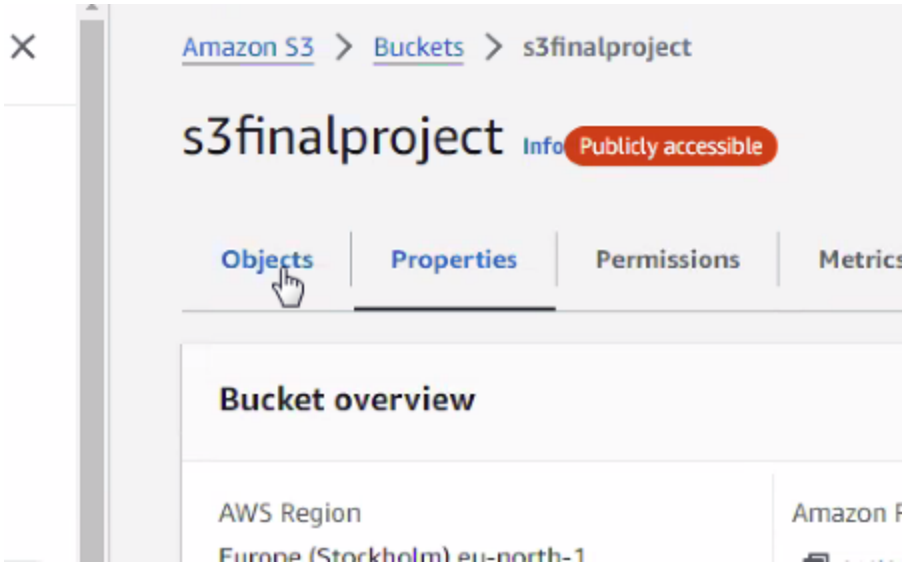
44. Scroll down and click on the link ‘Register by Submitting the Form’



45. [404 error! Mistake is: profile.html document was not added to the S3 bucket!]

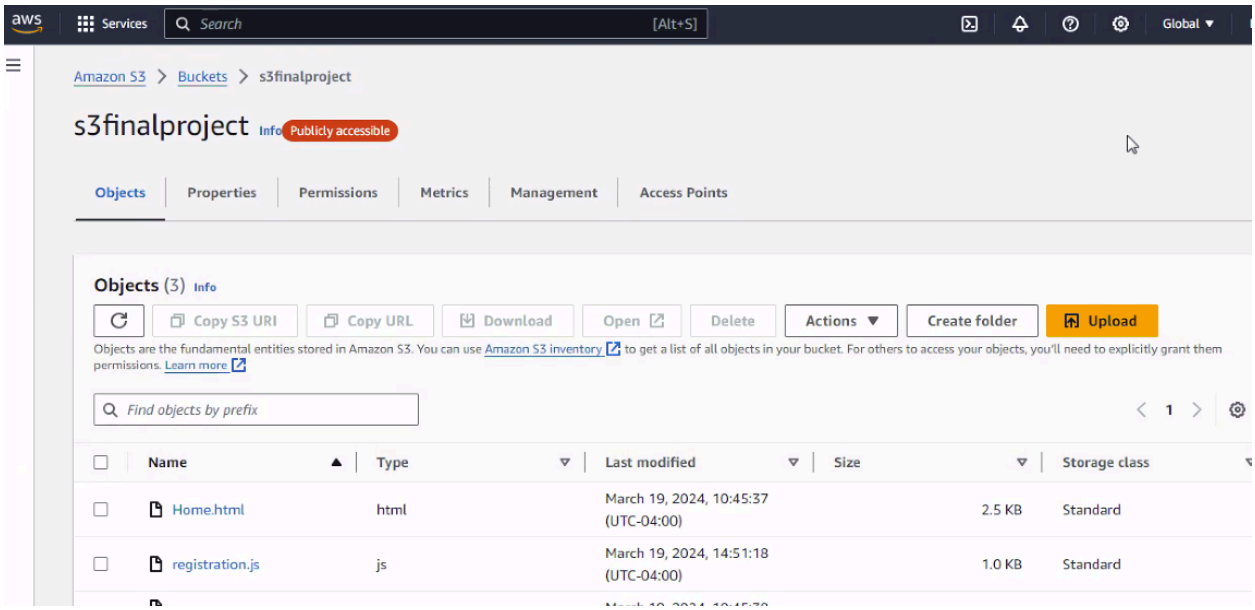


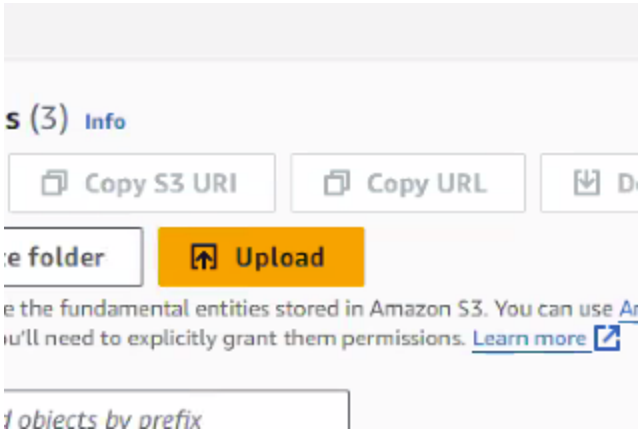
46. Go back to S3 bucket



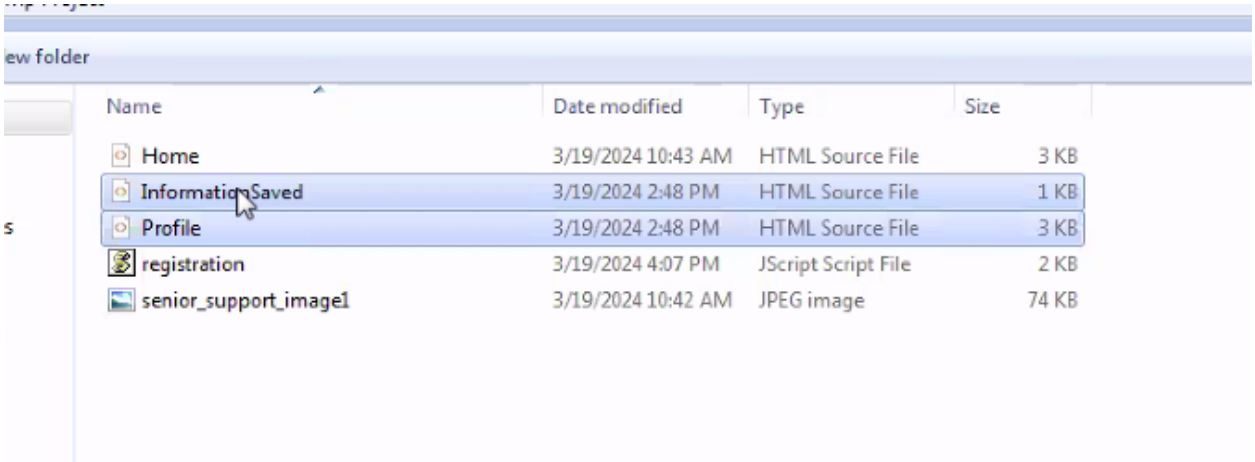
47. Click on 'Objects tab'

48. Click on ‘Upload’ orange button on the right side

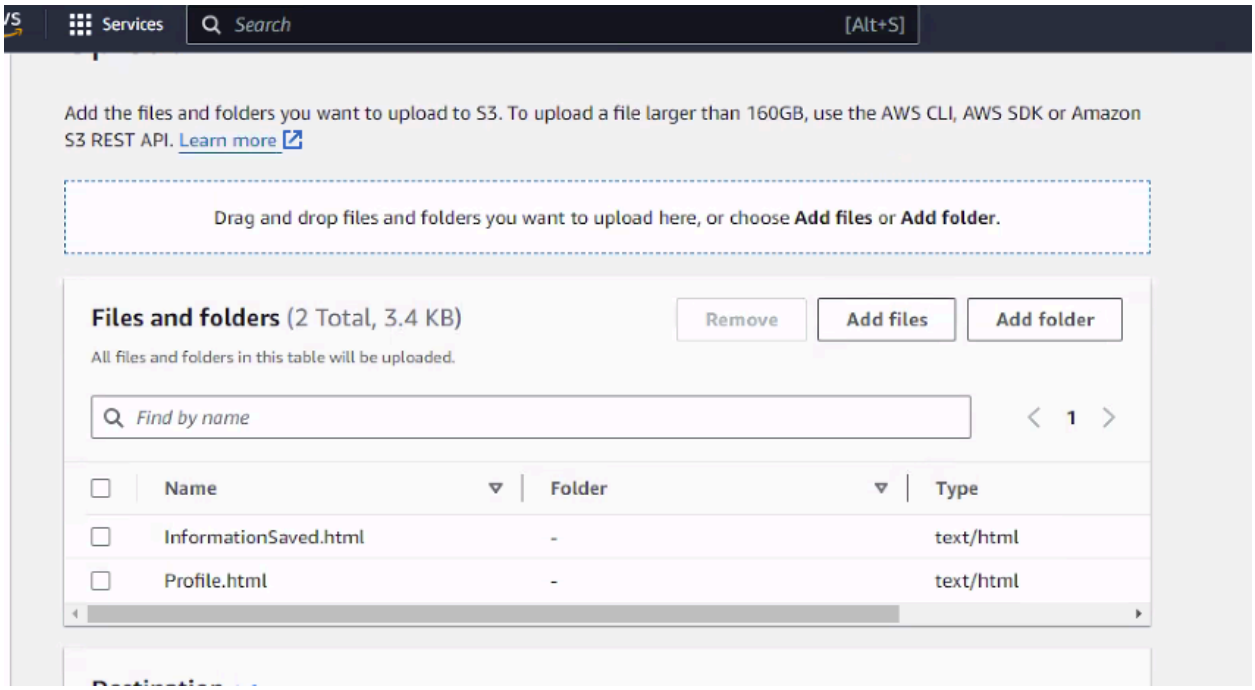




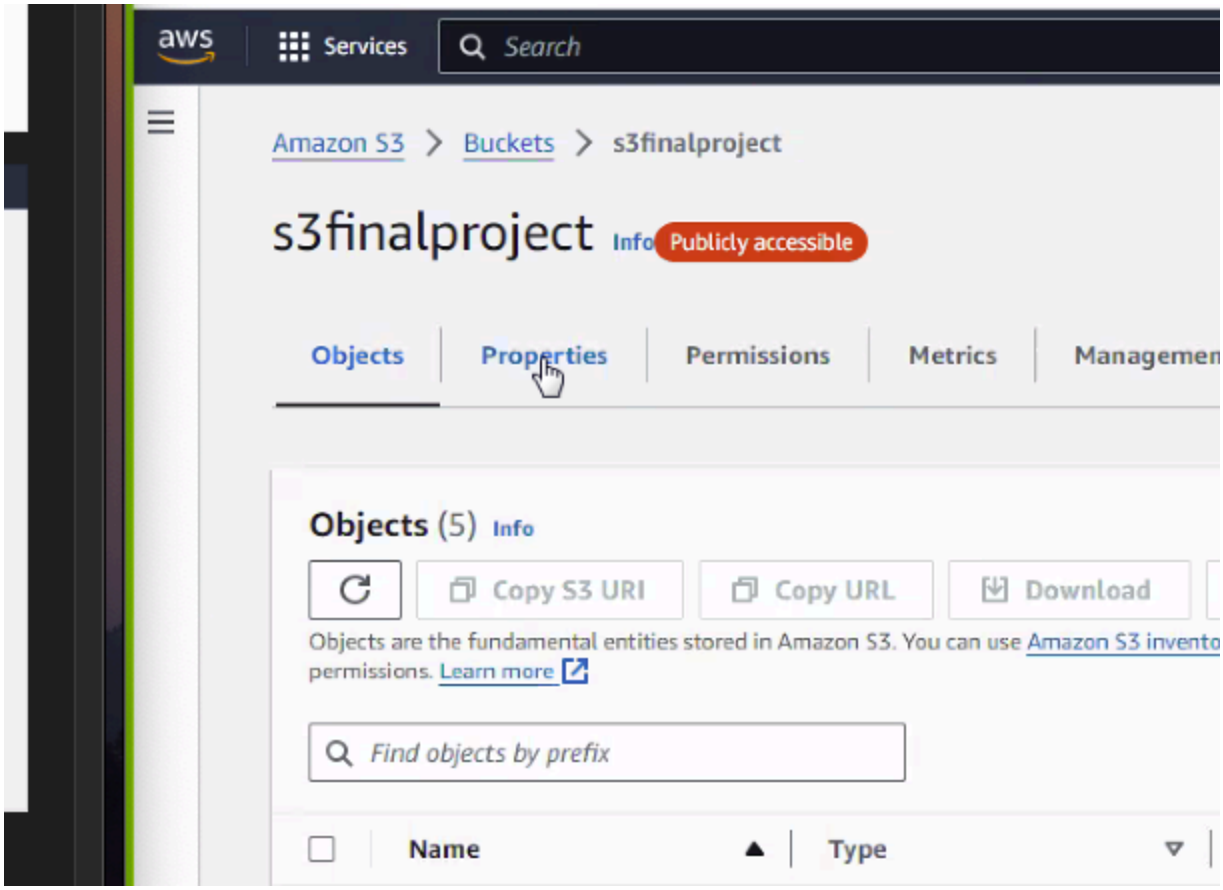
49. Add Profile.html and InformationSaved.html



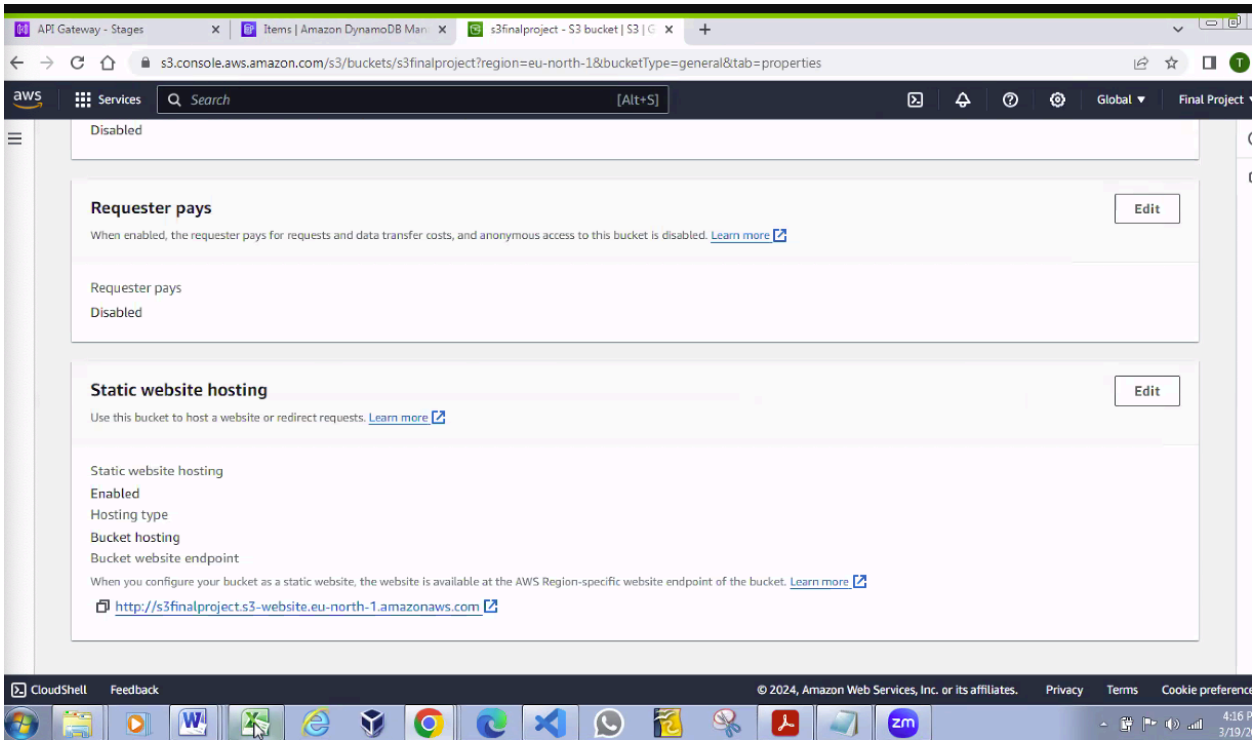
50. It should look like this:



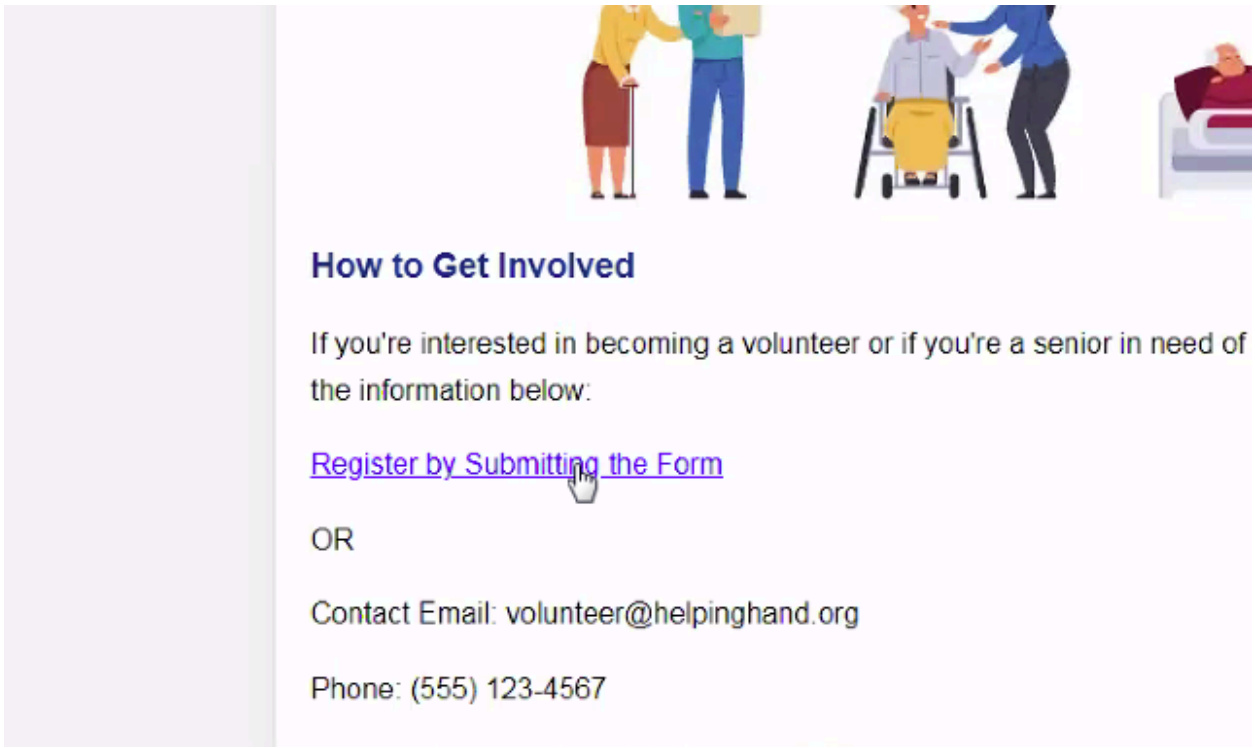
51. Scroll down to click 'Upload' button
52. In you S3 bucket, click on 'properties' tab



53. Scroll down and go click on that very same URL we visited earlier

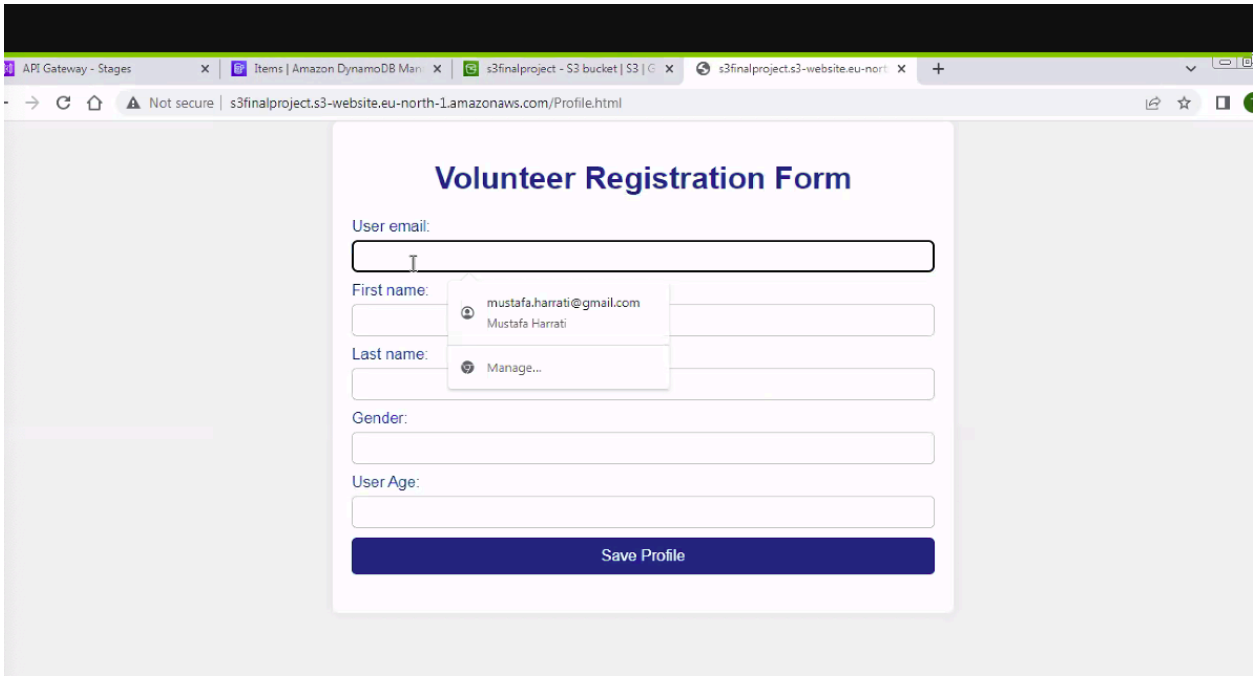


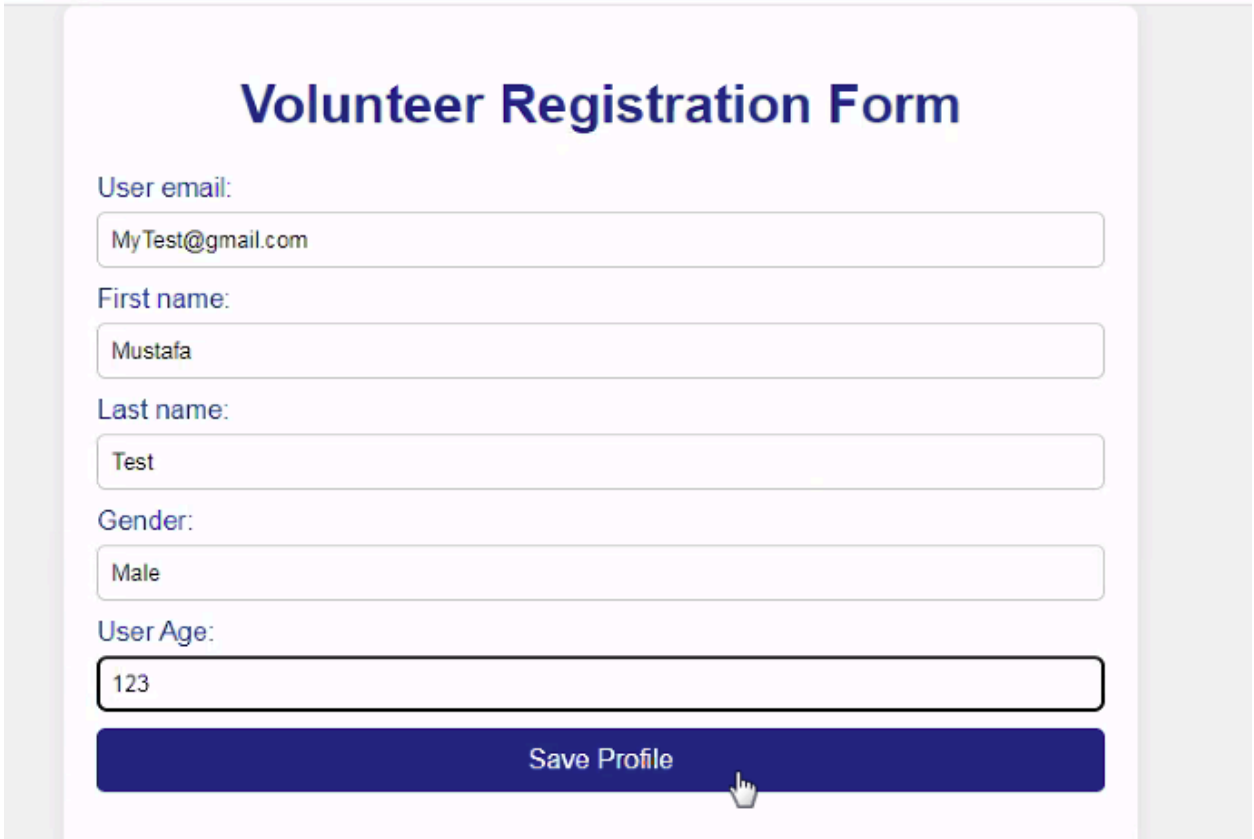
54. Back on the website, scroll down to click on that same link 'Register by Submitting the Form'



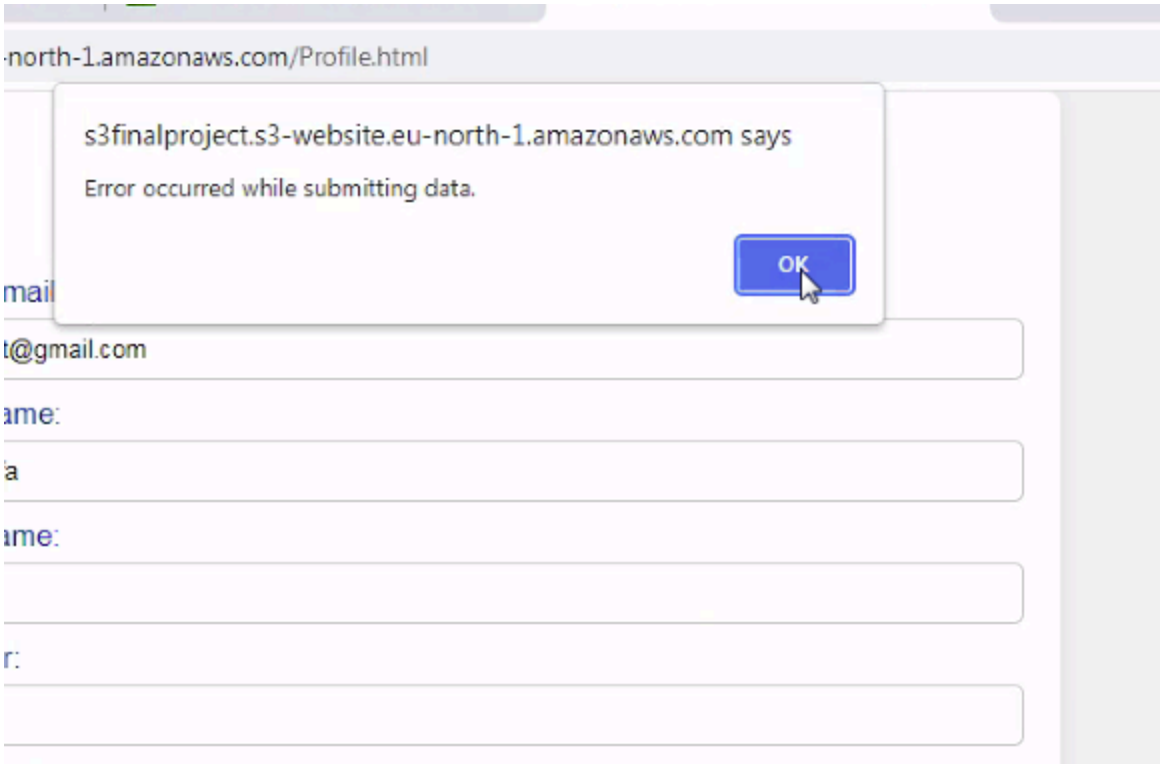
55. Success

56. Now enter any details in the registration form

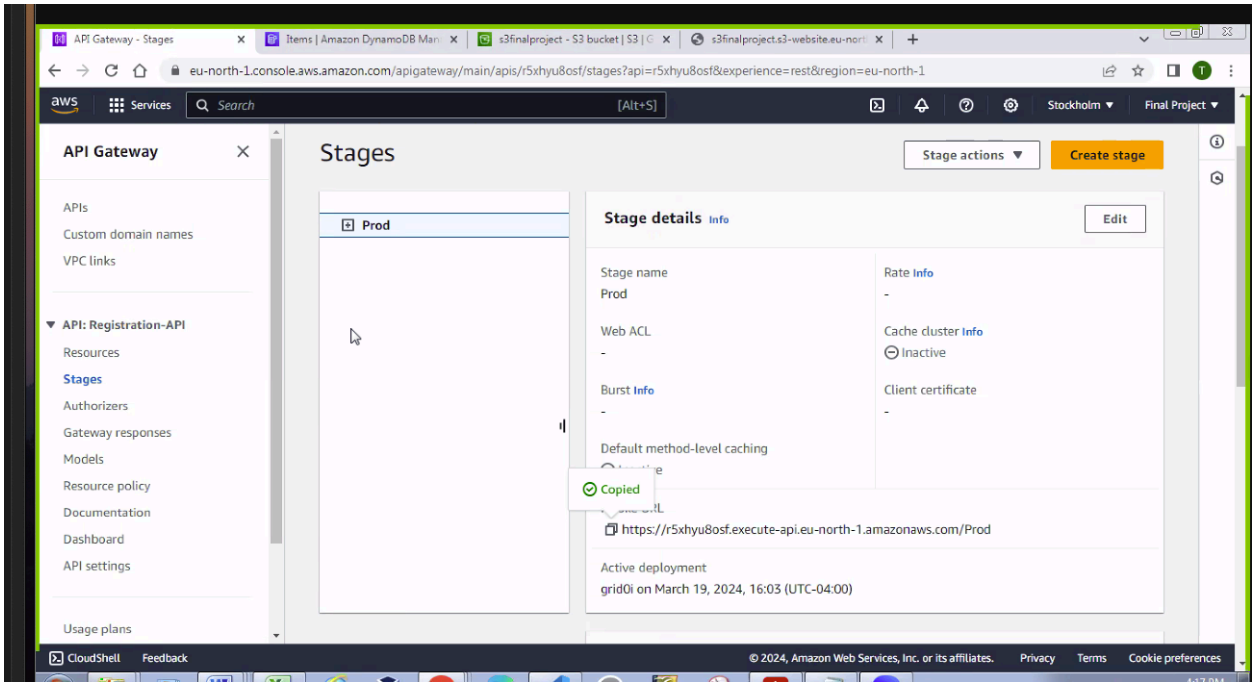




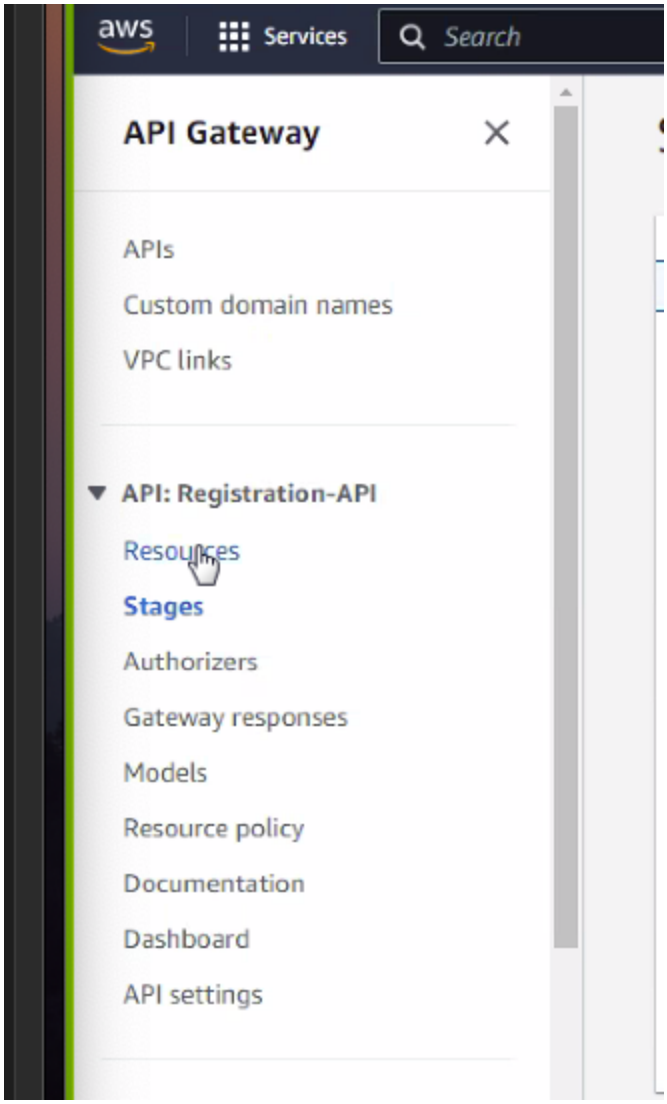
57. [ERROR!]



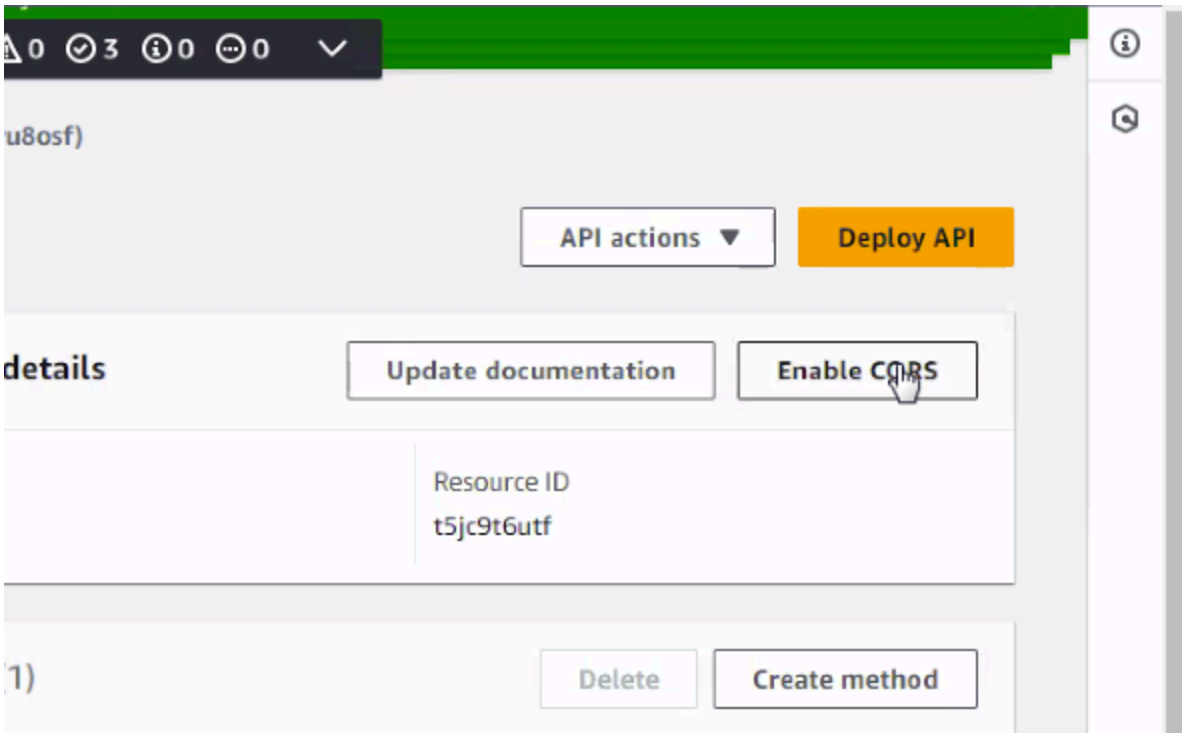
58. Go back to the API Gateway



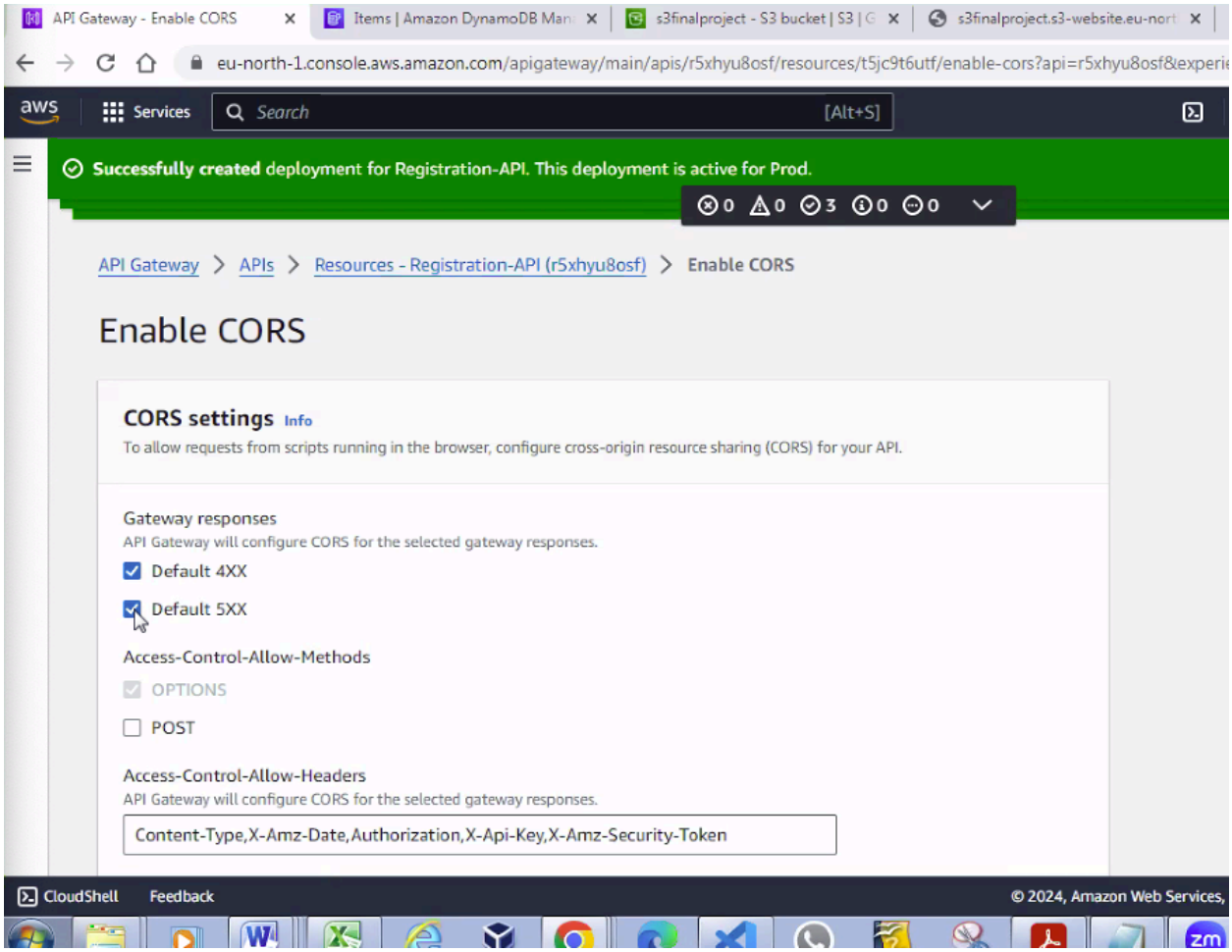
59. On the left side, click on ‘Resources’



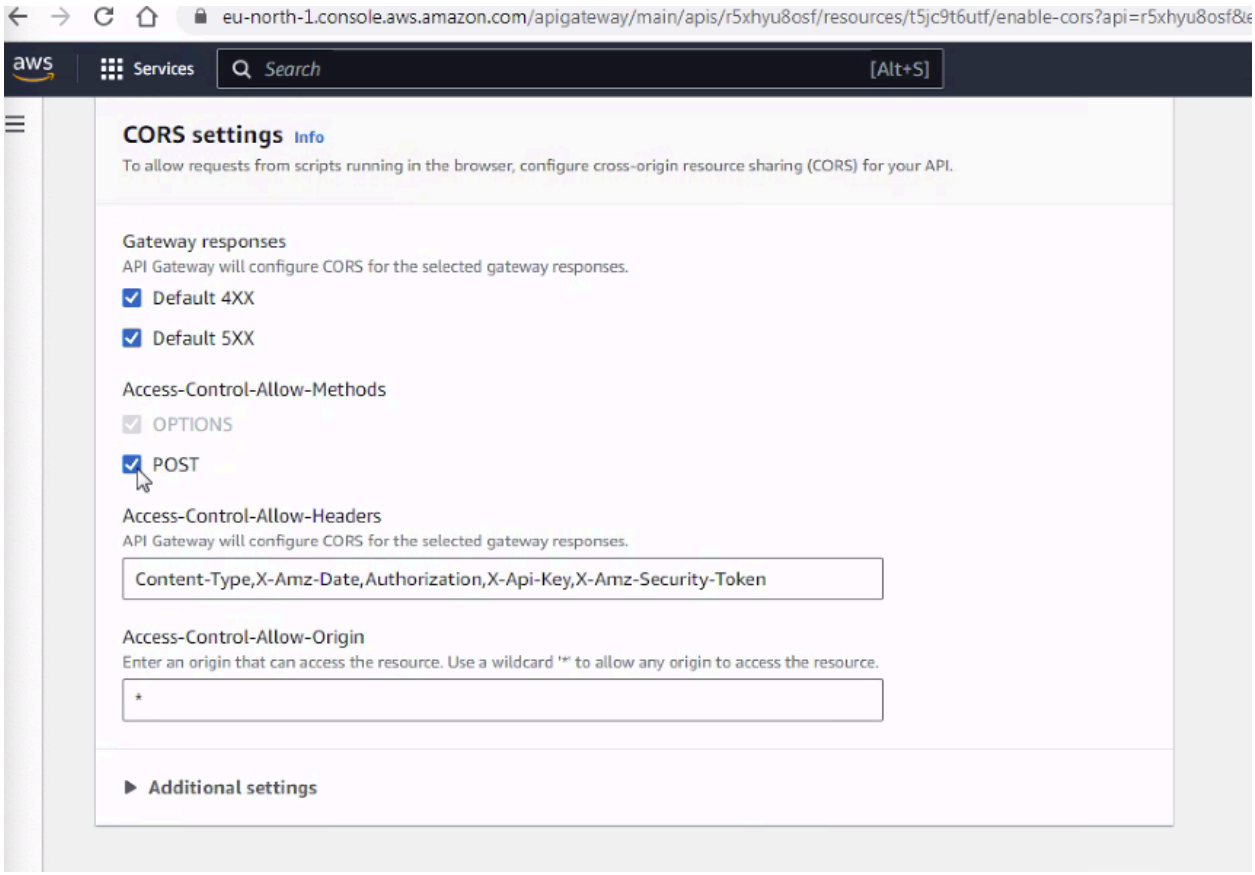
60. Enable CORS by clicking on the button



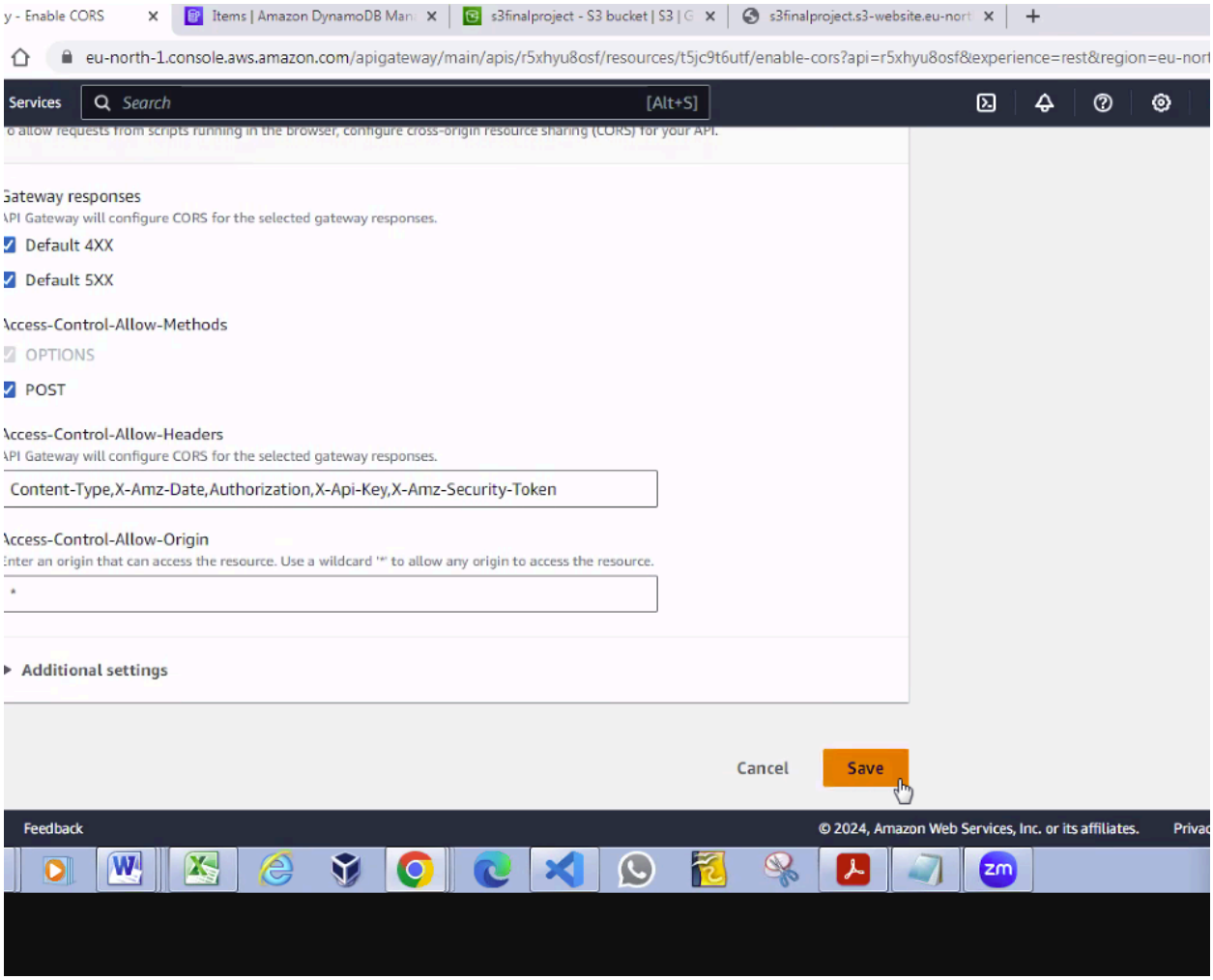
61. In the ‘Enable CORS’ section, select both “Default 4XX” and “Default 5XX”



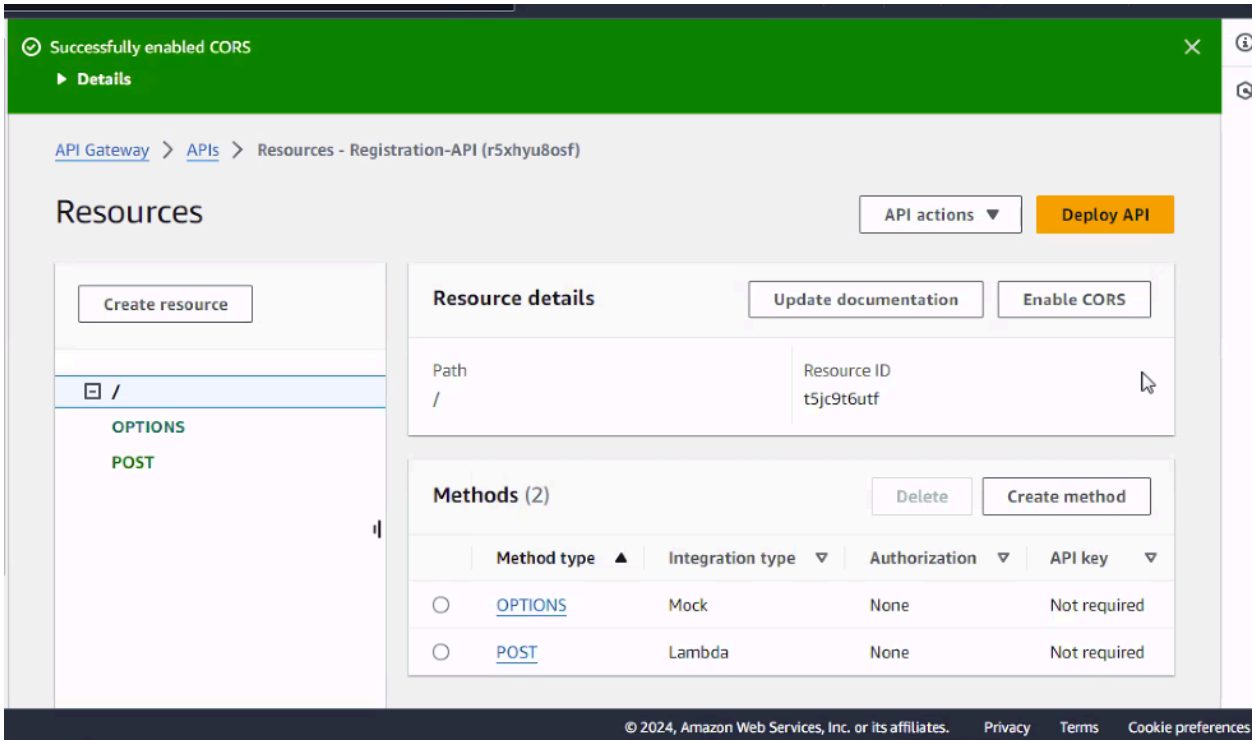
62. Select ‘POST’ option too

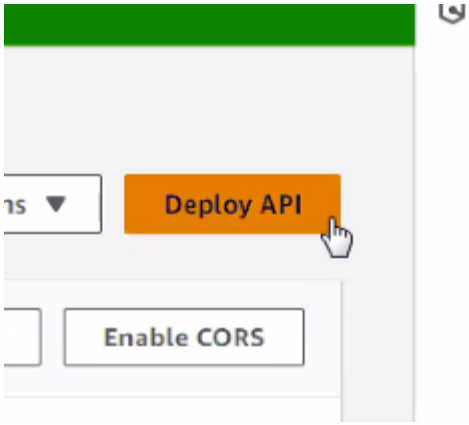


63. Scroll down and click on “Save”



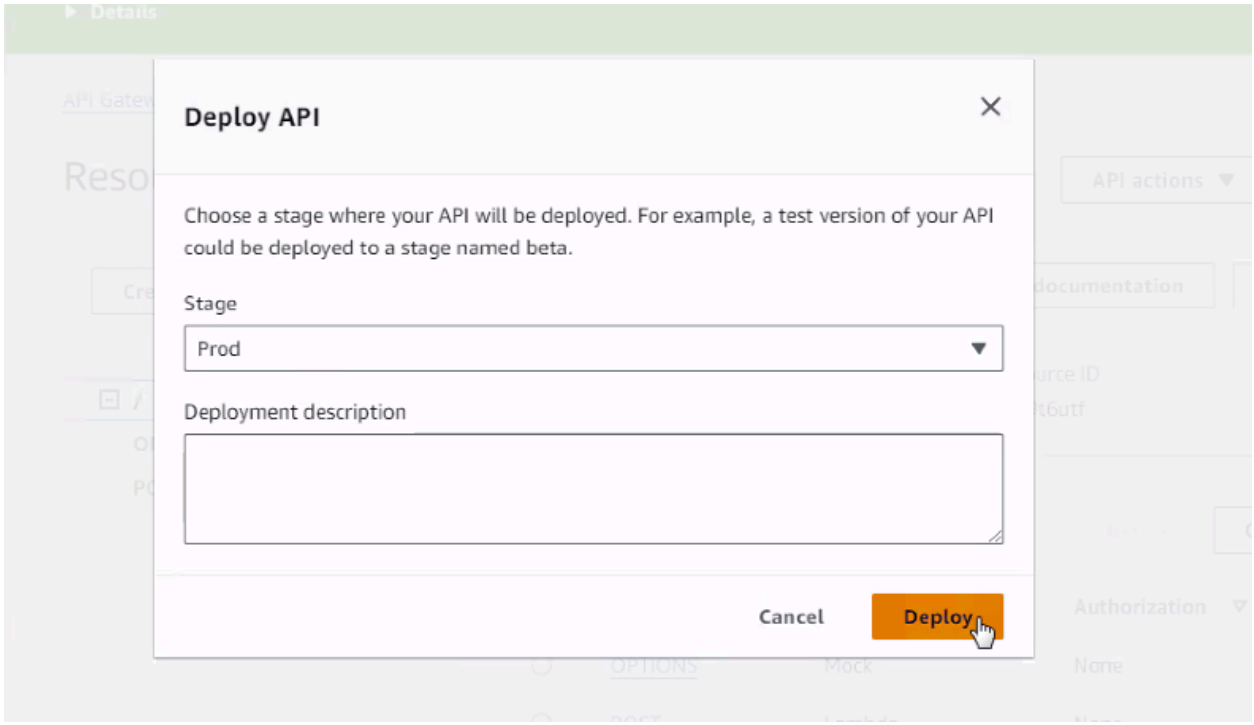
64. Success



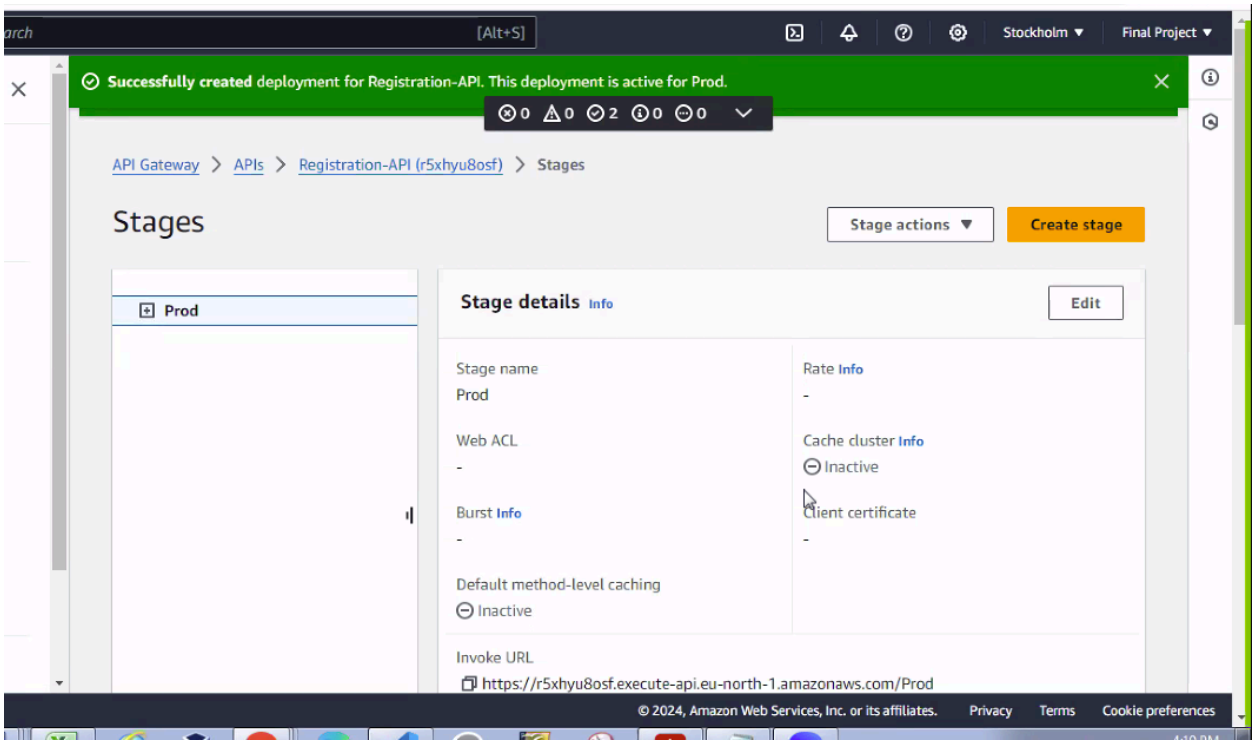


65. Click on 'Deploy API'
66. Select the 'Stage' as the name you gave it earlier "Prod"

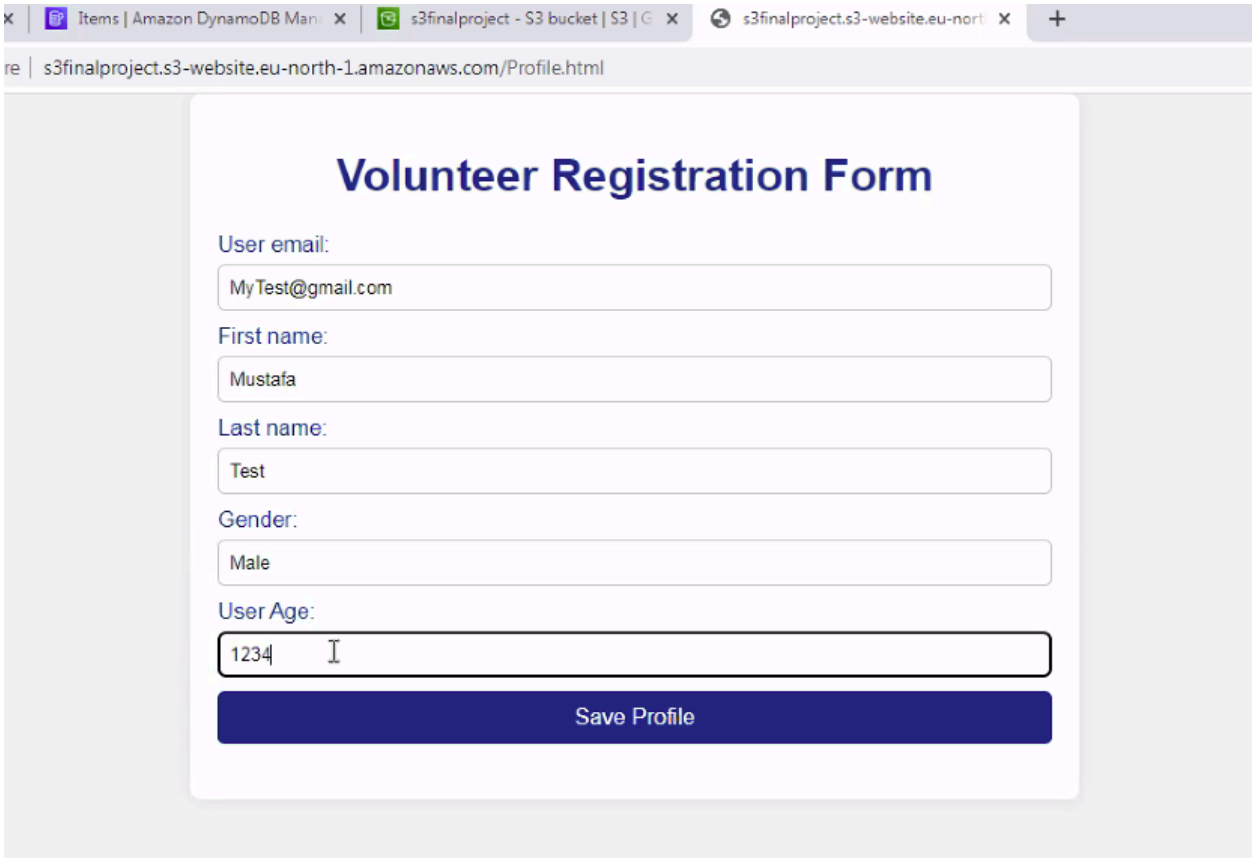
67. Click ‘Deploy ‘ button in orange at the bottom right corner



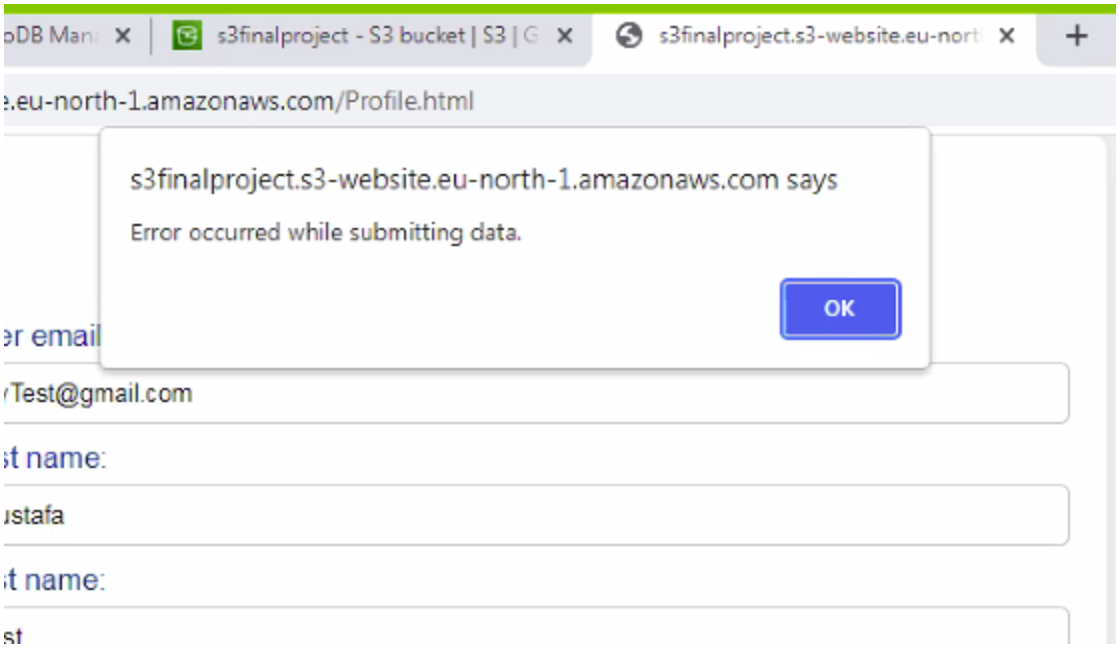
68. Success



69. Go back to website and see if you can register…



70. [Error!]



71.