

# Assignment 5

## Aws Lab work

22BRS1117

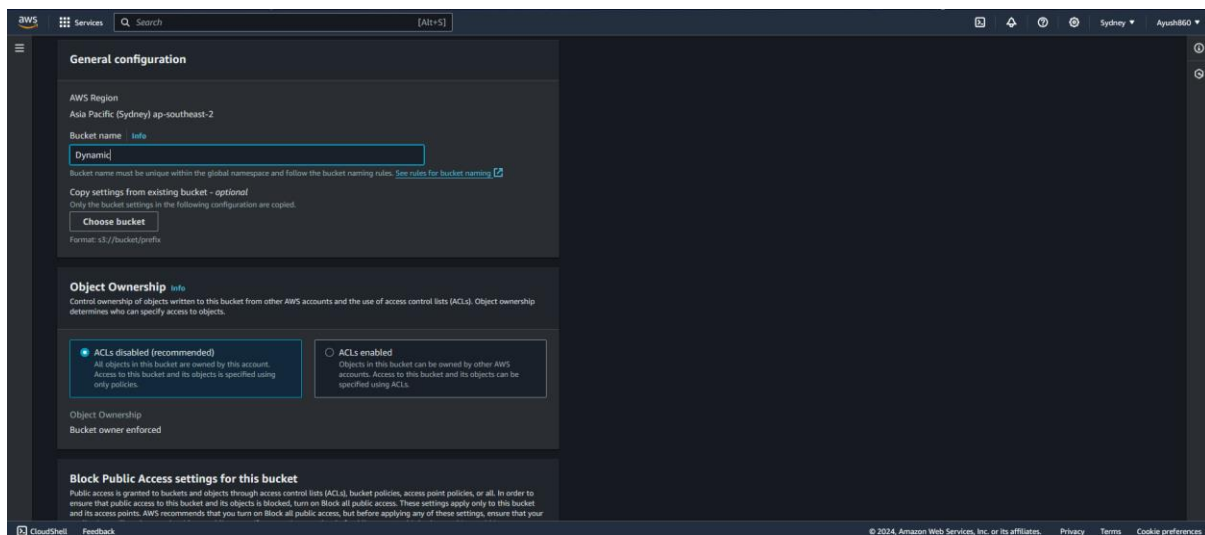
Ayush Raj

### Instructions

Create a Dynamic website and deploy it in AWS Lambda

Follow the same standards and naming conventions while uploading

#### 1. Creating an s3 bucket



#### 2. Give cross account access

## Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, 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 public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

### ☐ Block *all* public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

#### ☐ Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

#### ☐ Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

#### ☐ Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

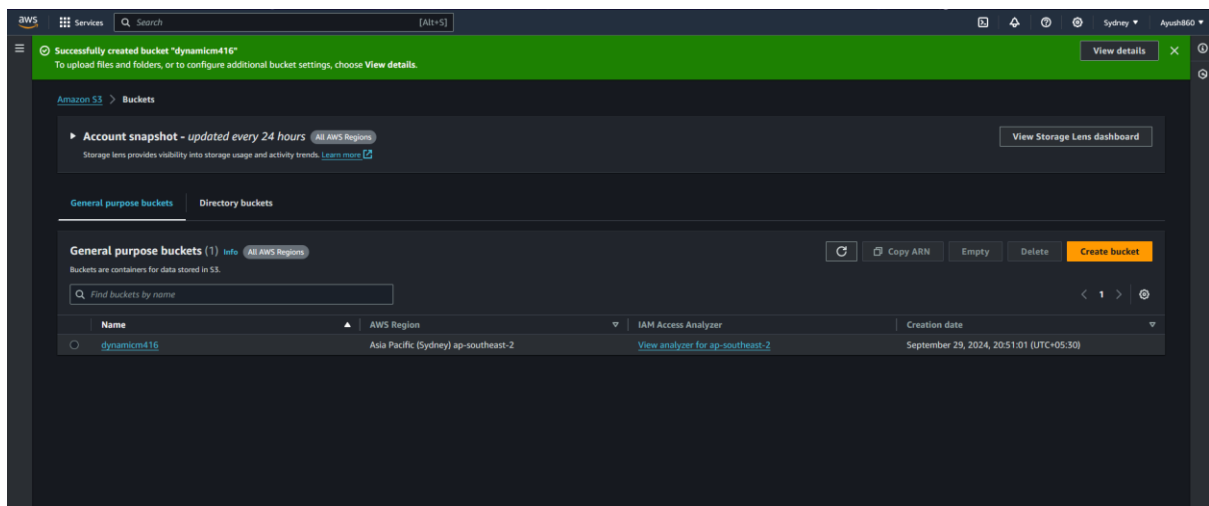
#### ☒ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



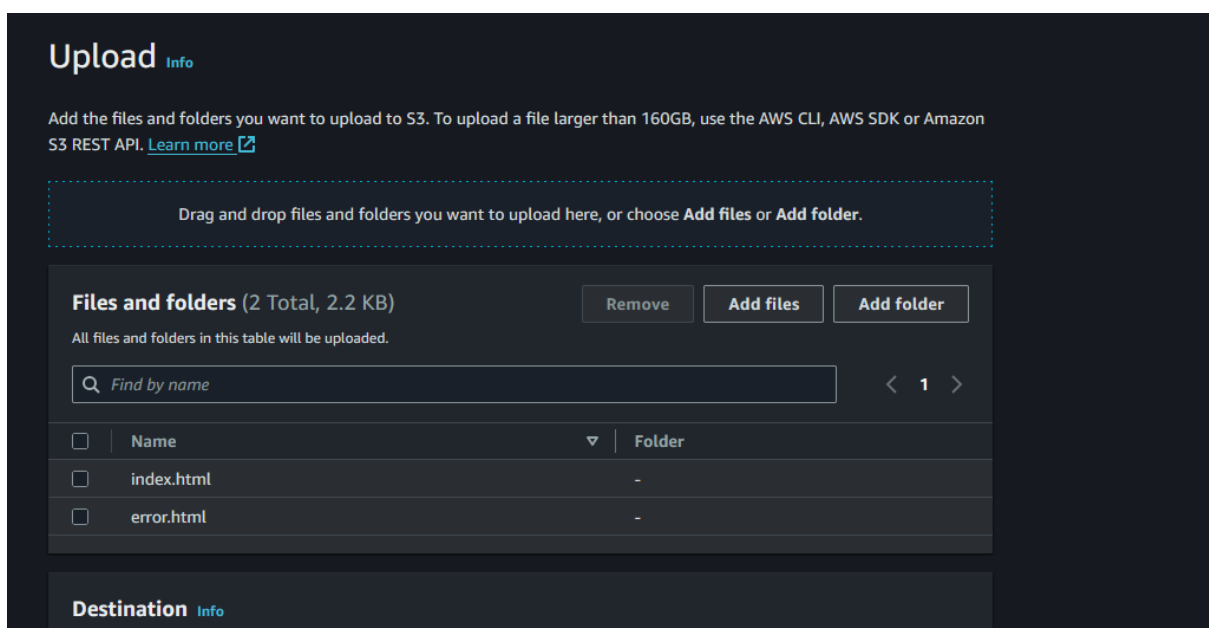
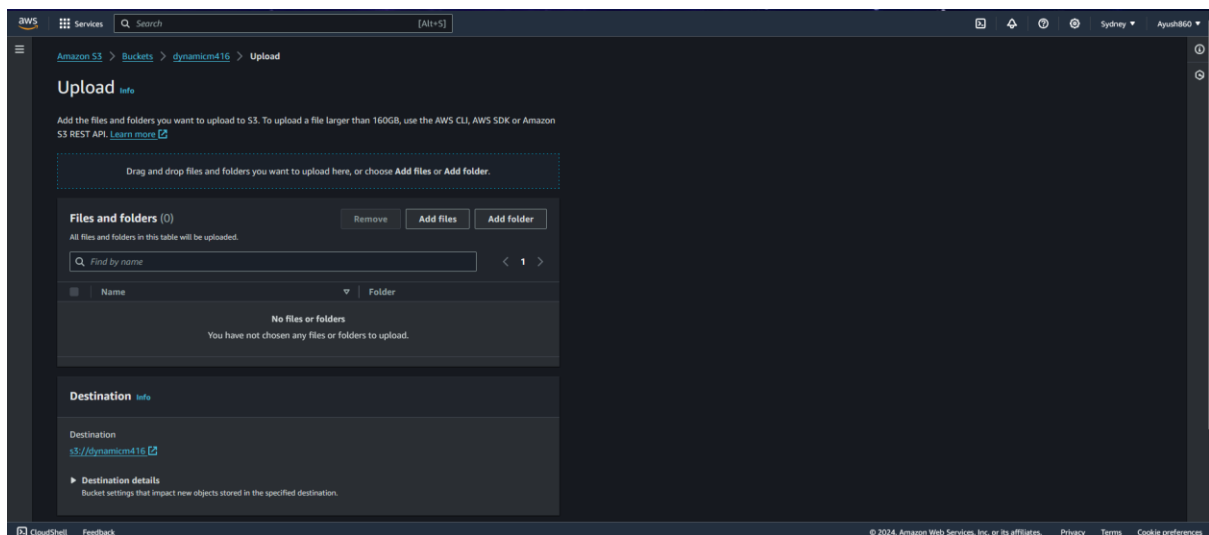
**Turning off block all public access might result in this bucket and the objects within becoming public**  
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

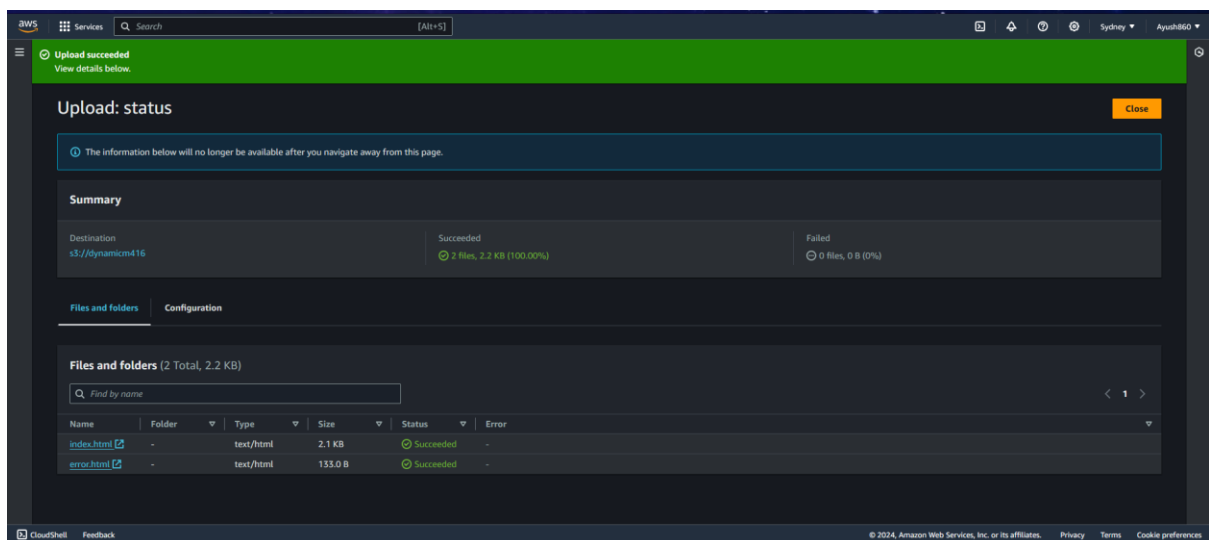


3. Click on the bucket, next step is to upload the files 2 files index.html and error.html has already been created

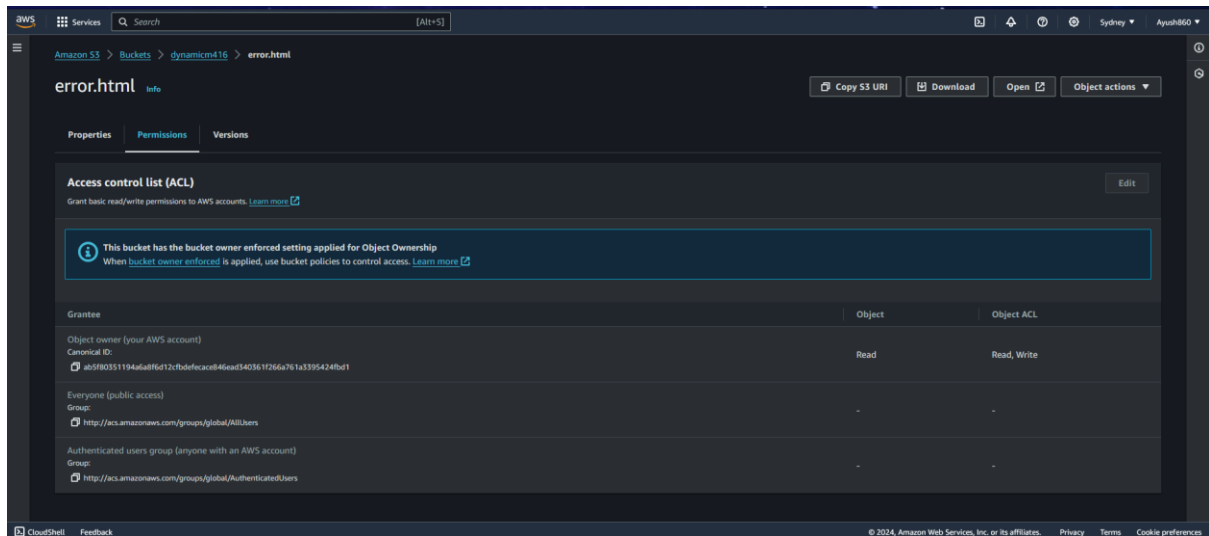
Click on upload and upload the files from the local desktop



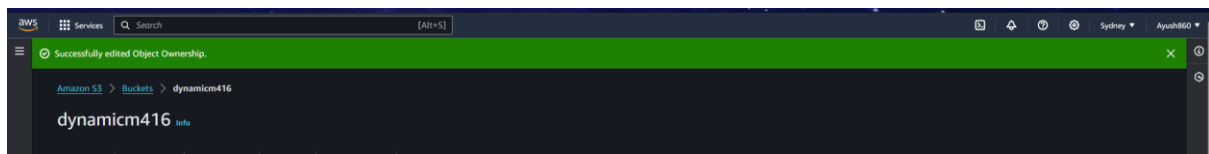
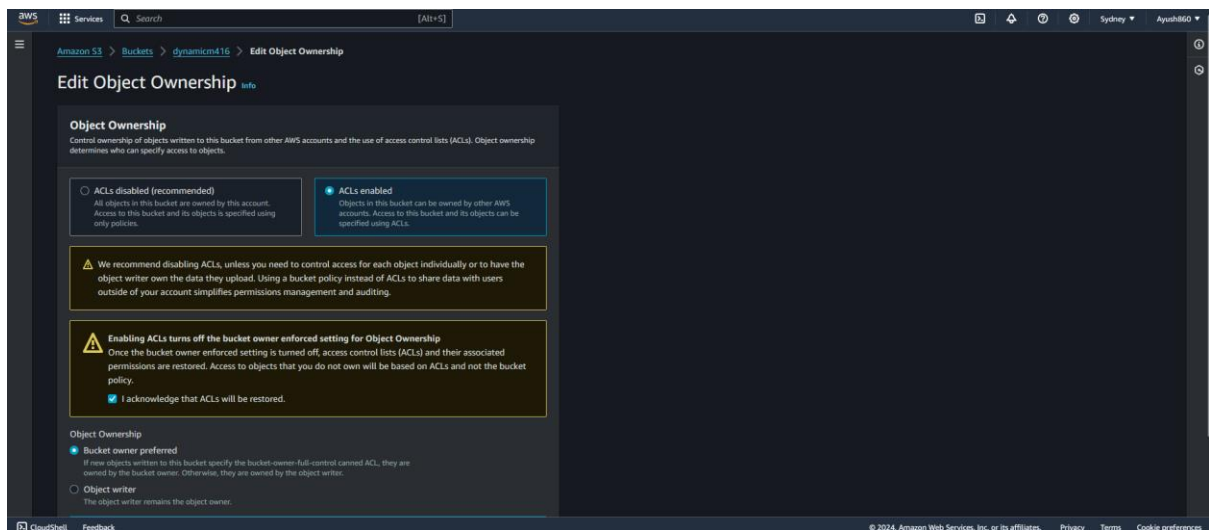
We can see upload is successful



- Now we have to make it public for that first we need to make object ownership and acl permissions to enabled



Do these settings and save changes



Similarly do this for the other object as well

- Now click on make public using acl

Services

Search

[Alt+S]

Sydney

Ayush860

Amazon S3

Buckets

dynamicm416

dynamicm416

info

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (2)

info

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

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

Find objects by prefix

	Name	Type	Last modified	Size
<input checked="" type="checkbox"/>	error.html	html	September 29, 2024, 20:57:37 (UTC+05:30)	13
<input checked="" type="checkbox"/>	index.html	html	September 29, 2024, 20:57:36 (UTC+05:30)	2

Download as

Share with a pre-signed URL

Calculate total size

Copy

Move

Initiate restore

Query with S3 Select

Edit actions

Rename object

Edit storage class

Edit server-side encryption

Edit metadata

Edit tags

Make public using ACL

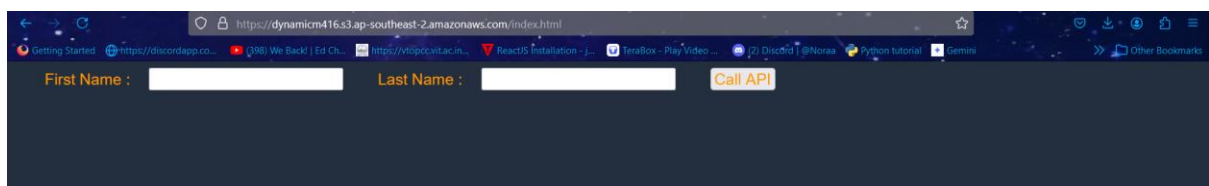


6. We can see the object urls

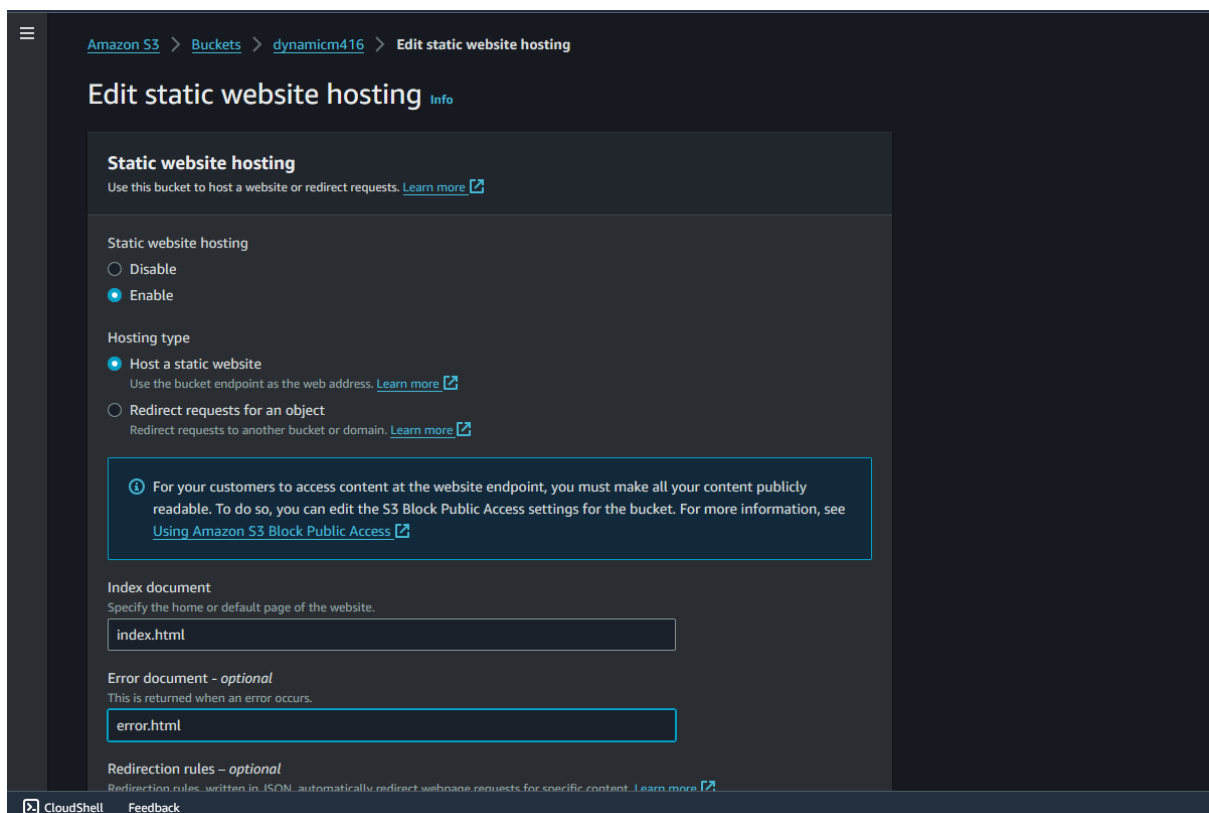
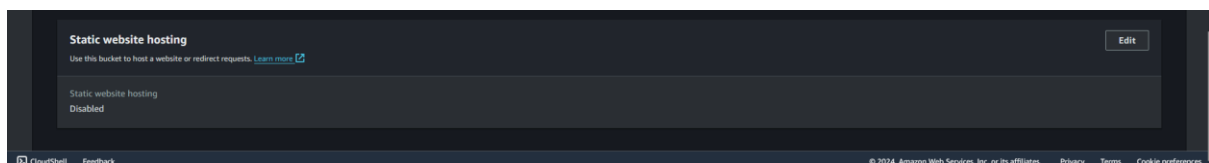
<https://dynamicm416.s3.ap-southeast-2.amazonaws.com/error.html>

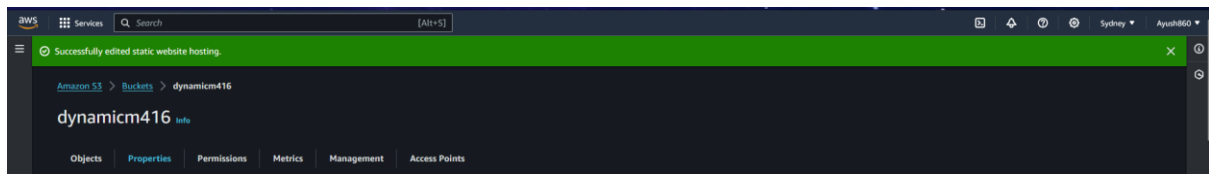


<https://dynamicm416.s3.ap-southeast-2.amazonaws.com/index.html>

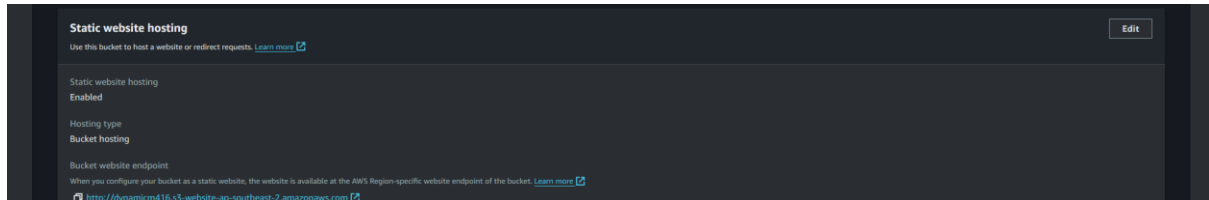


7. After this now do static website hosting



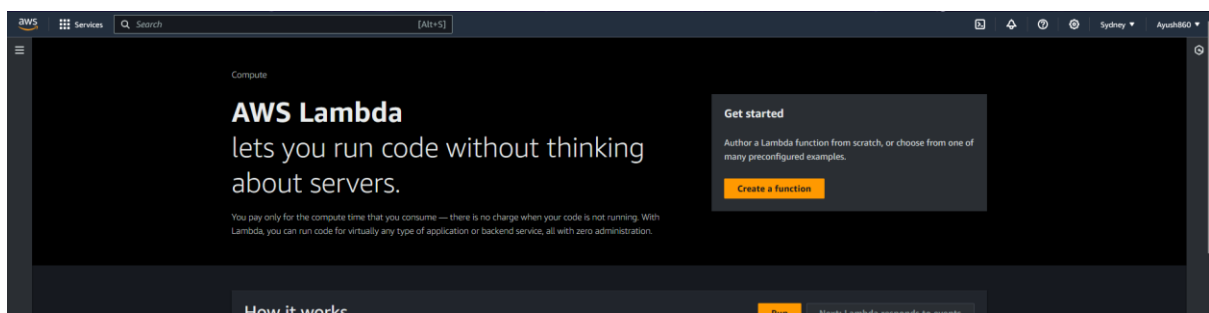
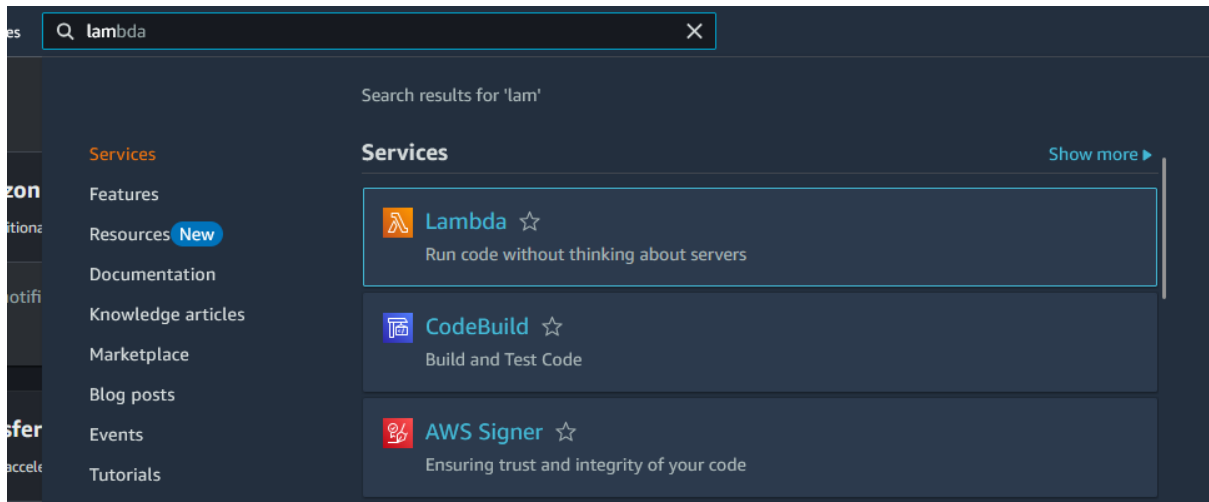


8. We got the website link

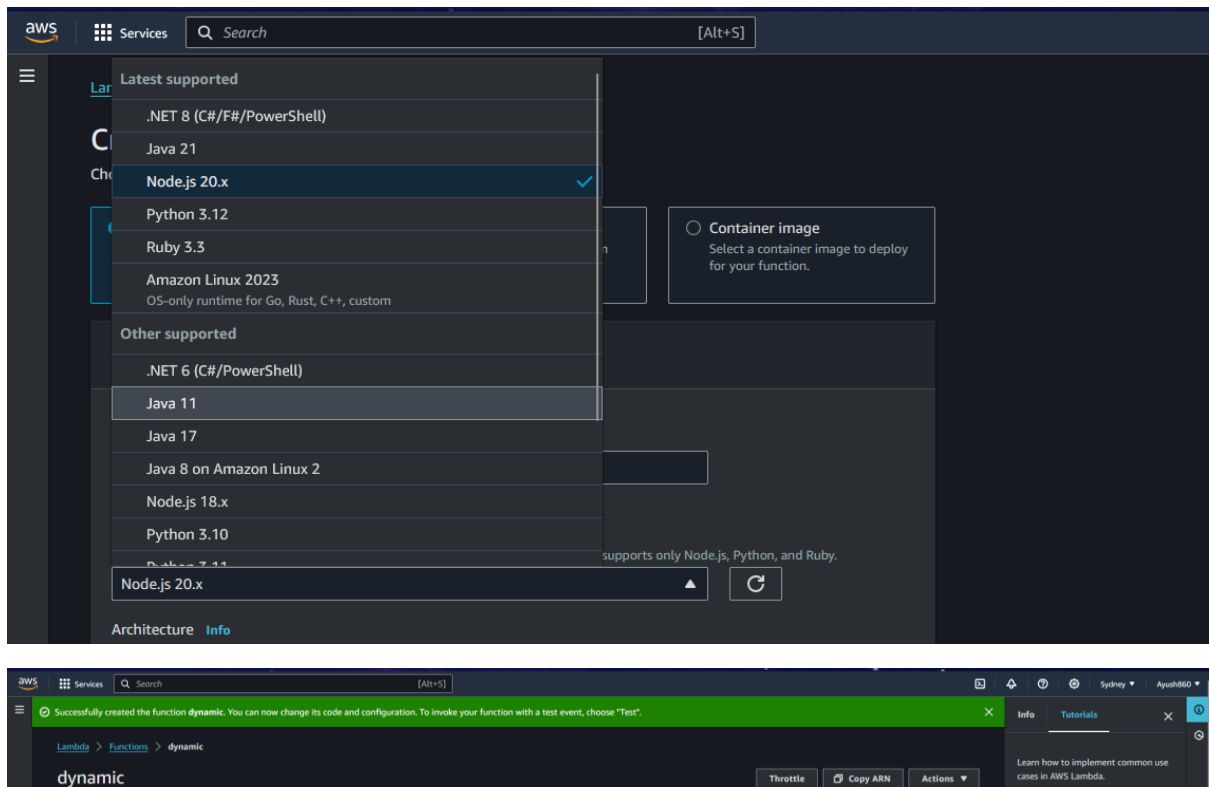


<http://dynamiccm416.s3-website-ap-southeast-2.amazonaws.com>

9. Now search for lambda and create function



10. Keep settings default only thing is change from node js to python latest version



## 11. Create lambda function handler

Code for the lambda handler

# import the json utility package since we will be working with a JSON object

import json

# import the AWS SDK (for Python the package name is boto3)

import boto3

# import two packages to help us with dates and date formatting

from time import gmtime, strftime

# create a DynamoDB object using the AWS SDK

dynamodb = boto3.resource('dynamodb')

# use the DynamoDB object to select our table

table = dynamodb.Table('HelloWorldDatabase')

# store the current time in a human readable format in a variable

now = strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

# define the handler function that the Lambda service will use as an entry point



```
def lambda_handler(event, context):

# extract values from the event object we got from the Lambda service and store in a variable

    name = event['firstName'] + ' ' + event['lastName']

# write name and time to the DynamoDB table using the object we instantiated and save response in
a variable

    response = table.put_item(

        Item={

            'ID': name,

            'LatestGreetingTime':now

        })

# return a properly formatted JSON object

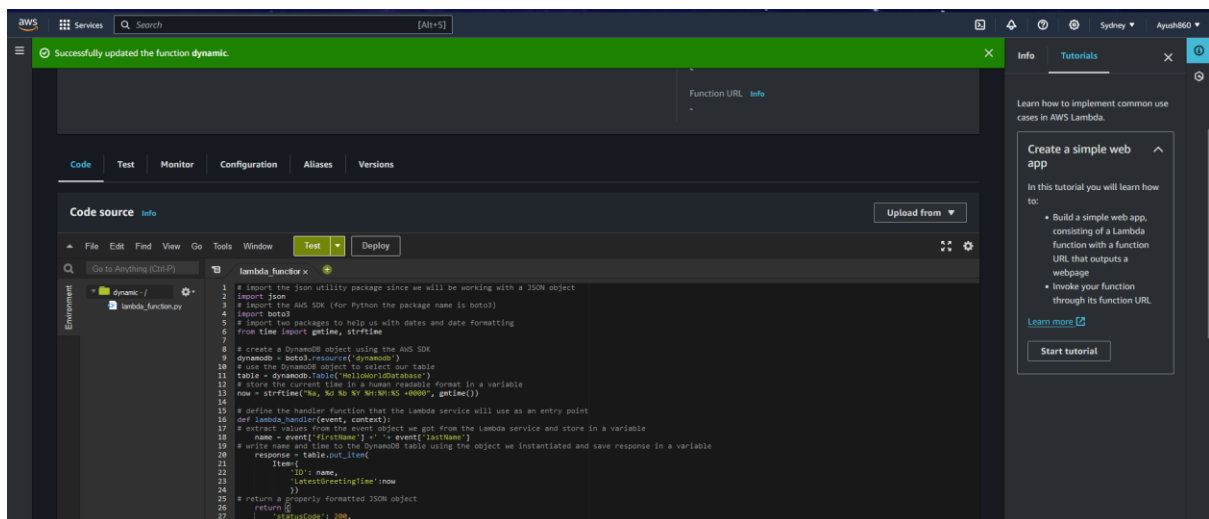
    return {

        'statusCode': 200,

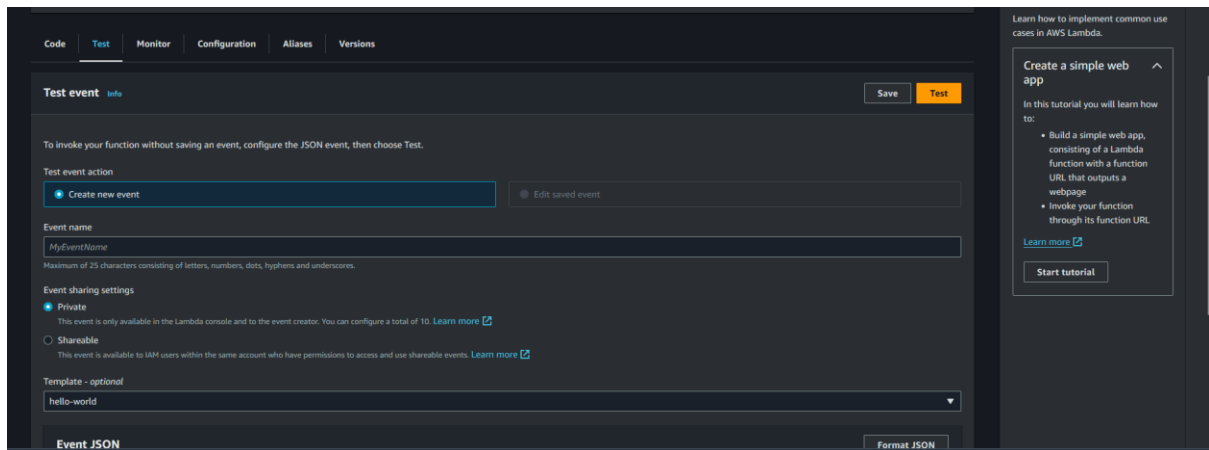
        'body': json.dumps('Hello from Lambda, ' + name)

    }
```

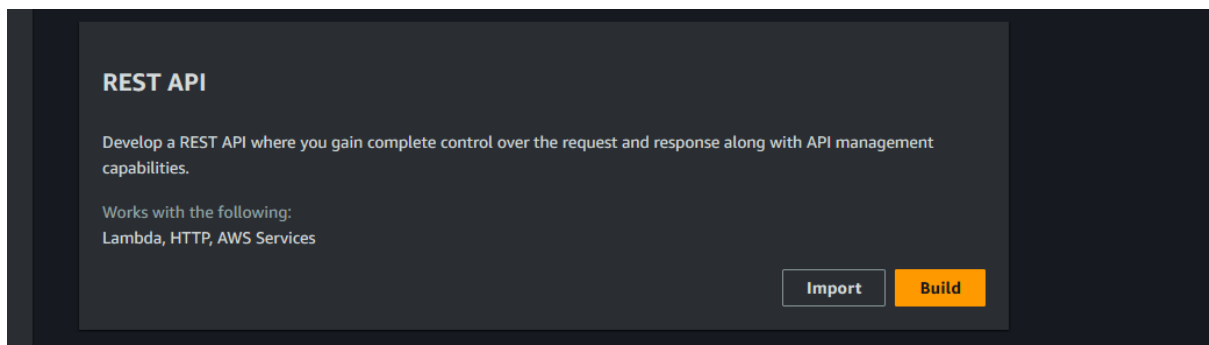
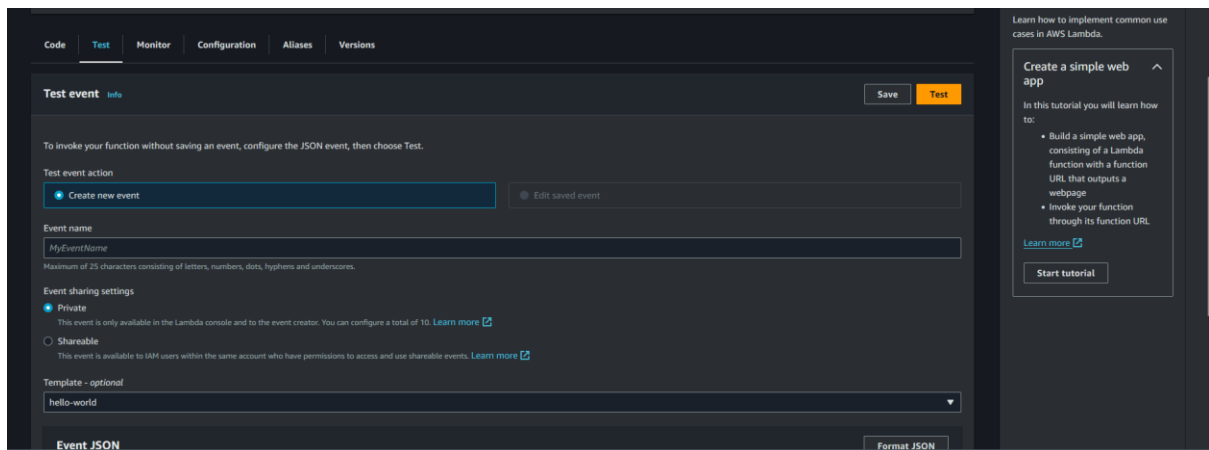
Deploy the changes

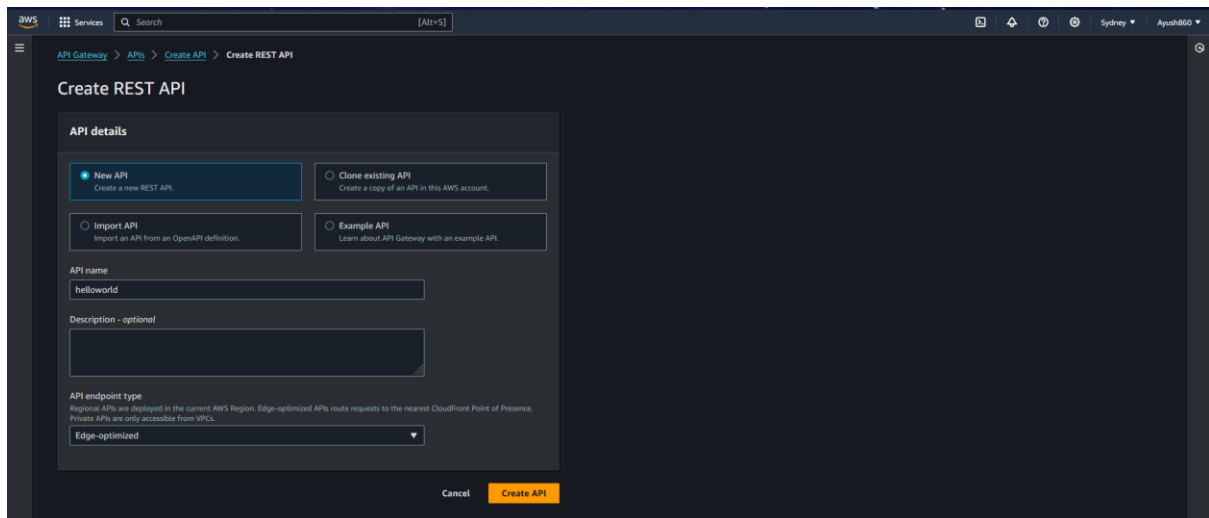


## 12. Create a test event

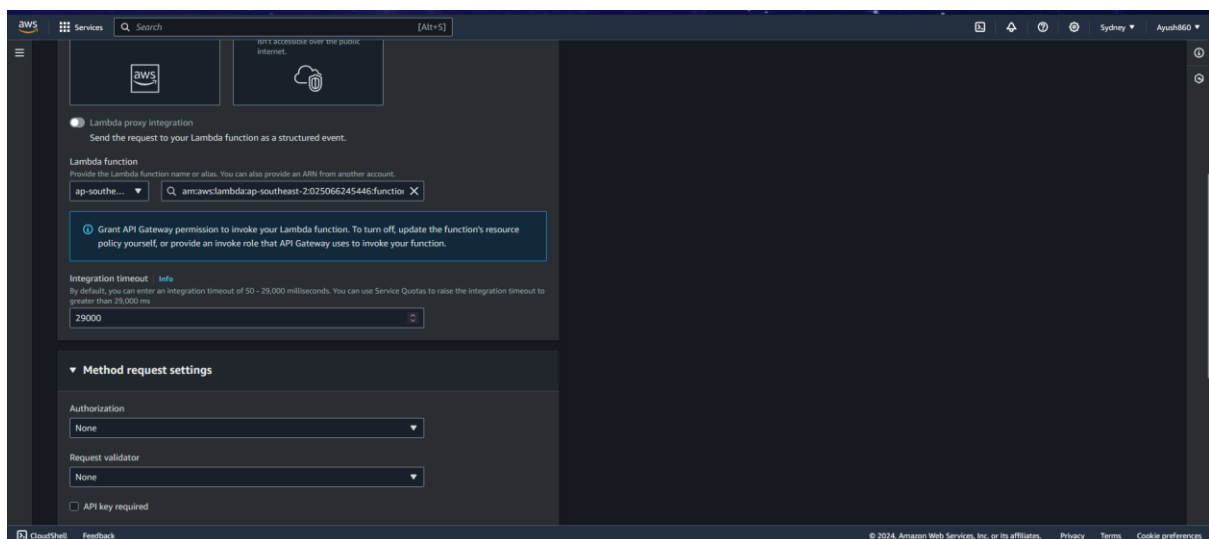
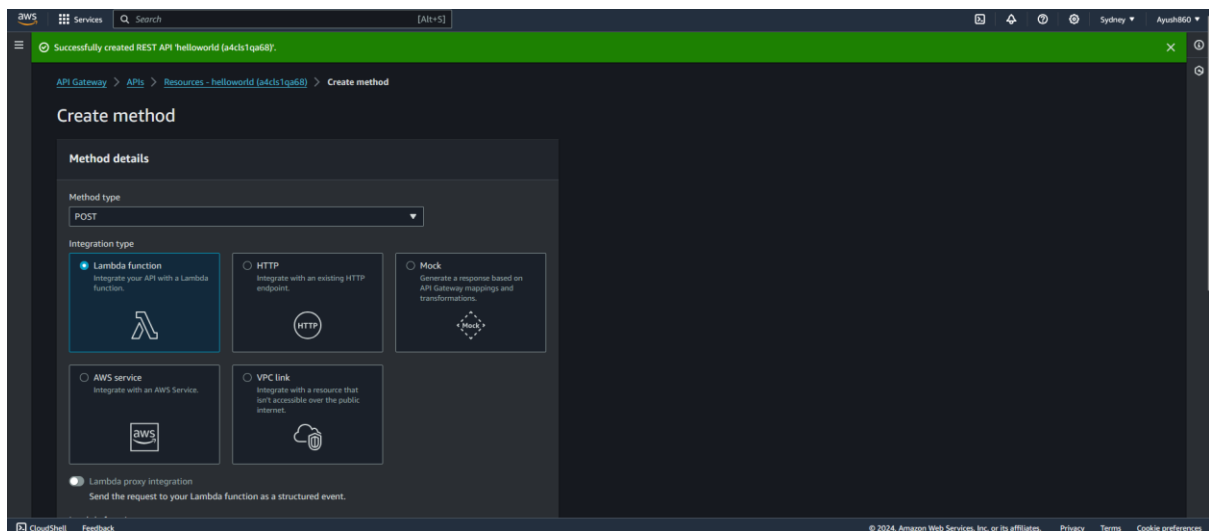


13. Now next we will look for api gateways

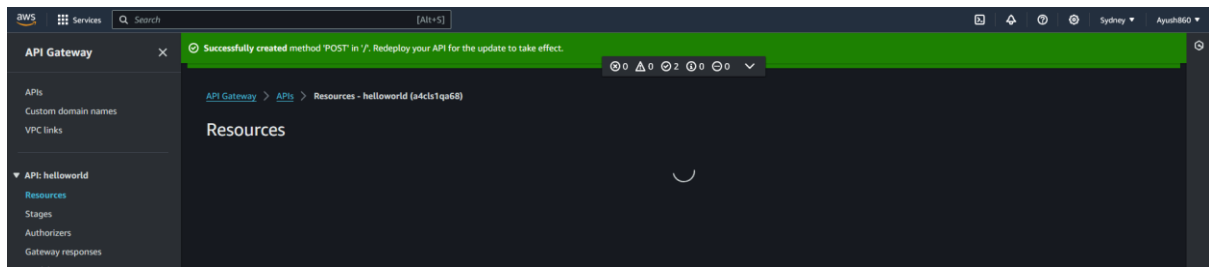




Now create a methods in rest api

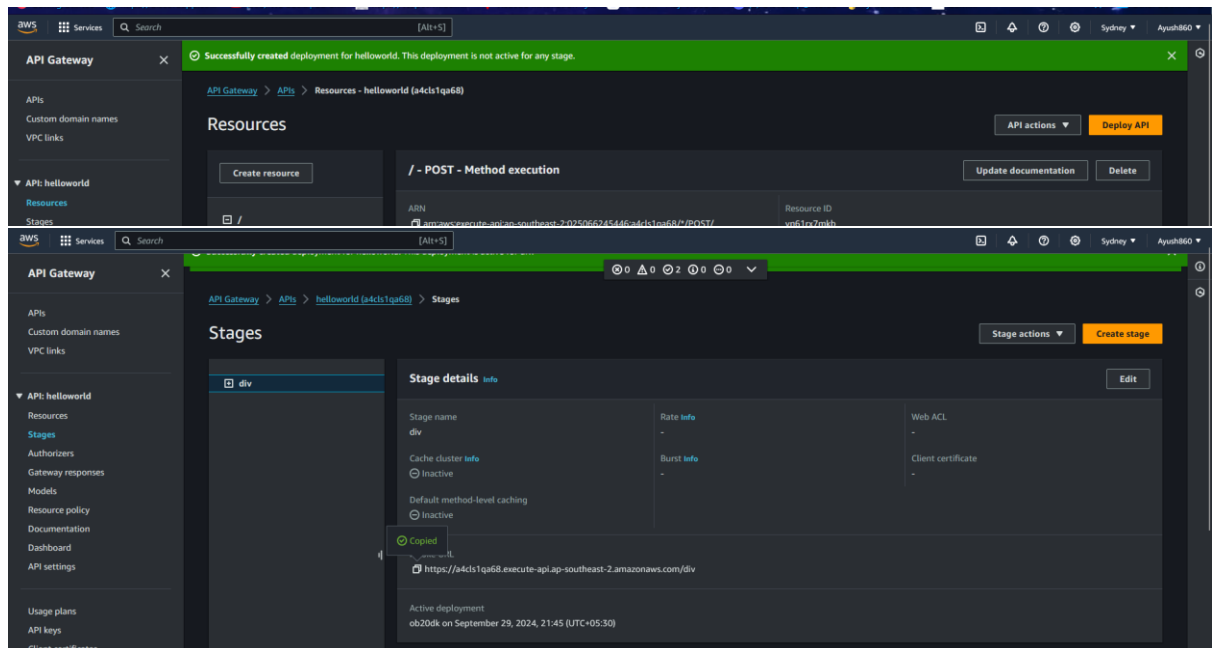


Select the lambda function that we created



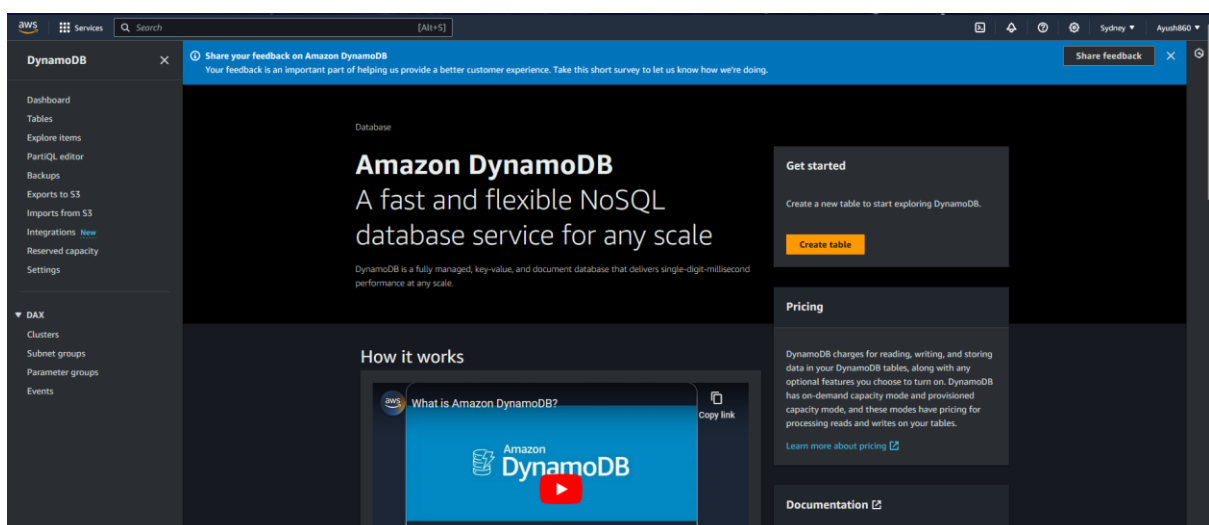
As we can see the method was successfully created

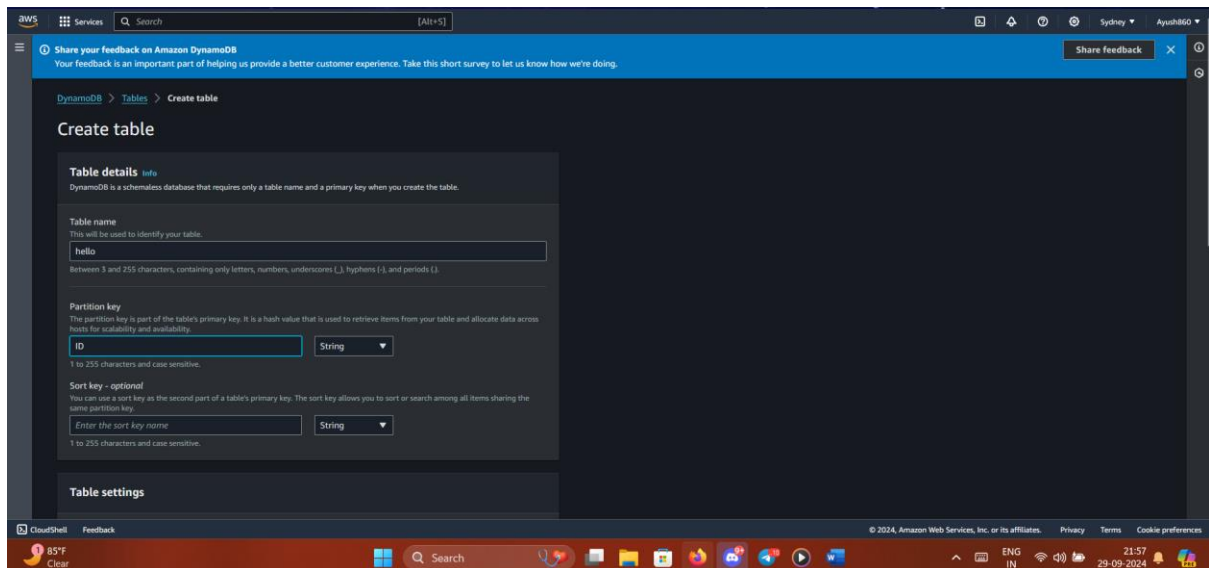
14.



15.

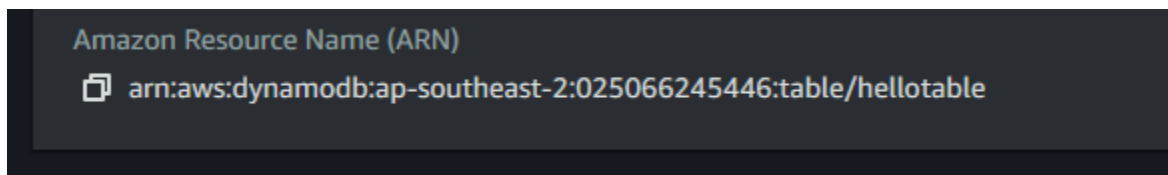
16. We need to create a dynamo db





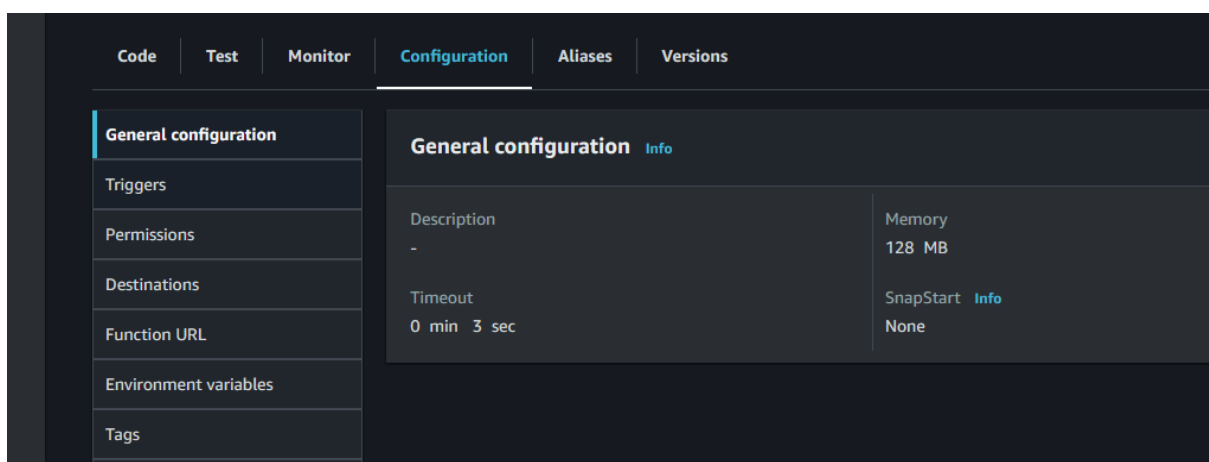
17. Create the db and copy the arn address

arn:aws:dynamodb:ap-southeast-2:025066245446:table/hellotable

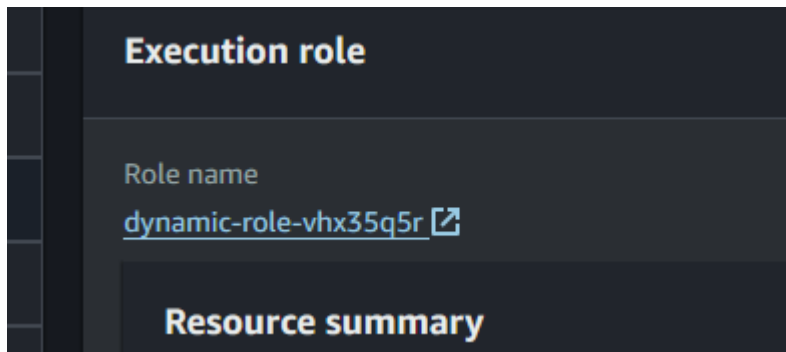


18. Now go back to lamda function

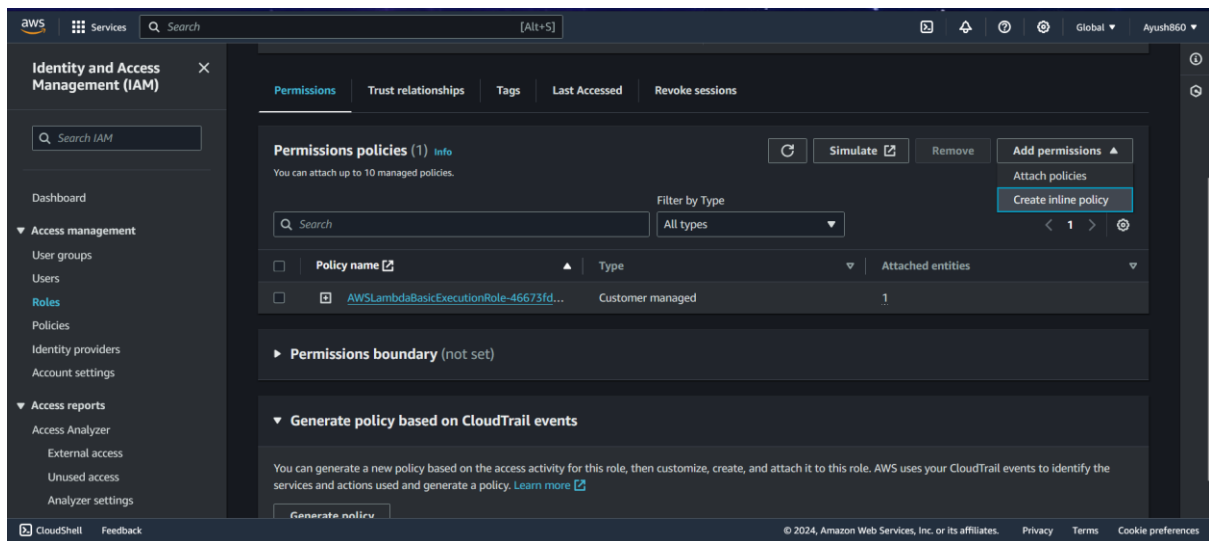
19. Click on configuration > permissions



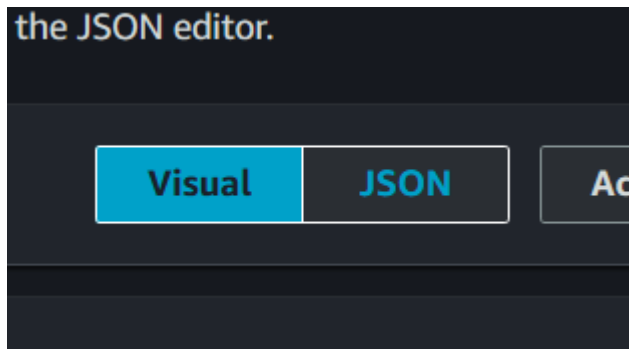
After permissions



Select the role >create inline policy



Choose Json



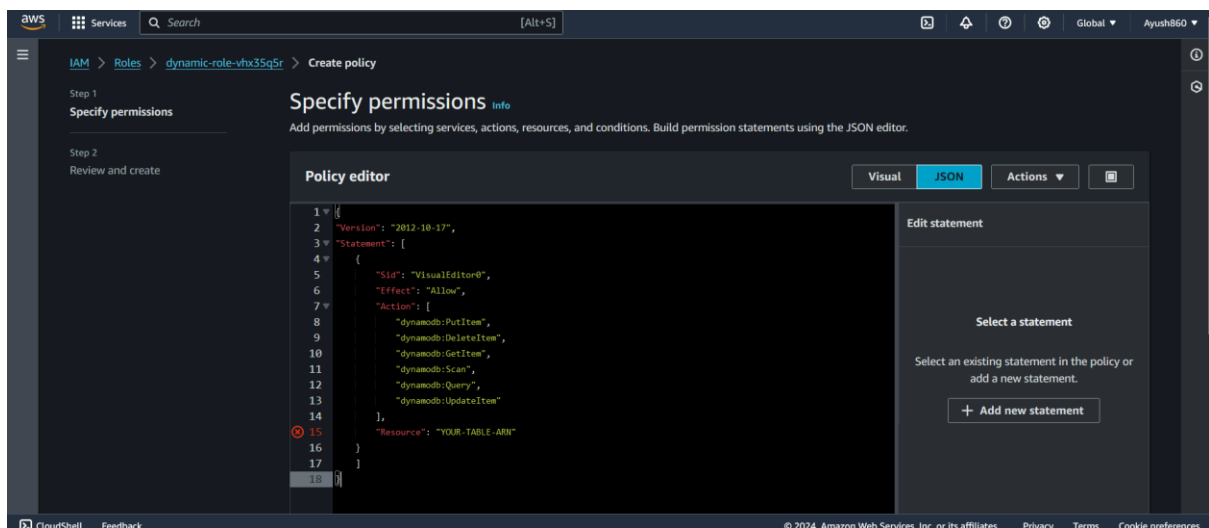
Write this code

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "VisualEditor0",  
      "Effect": "Allow",
```

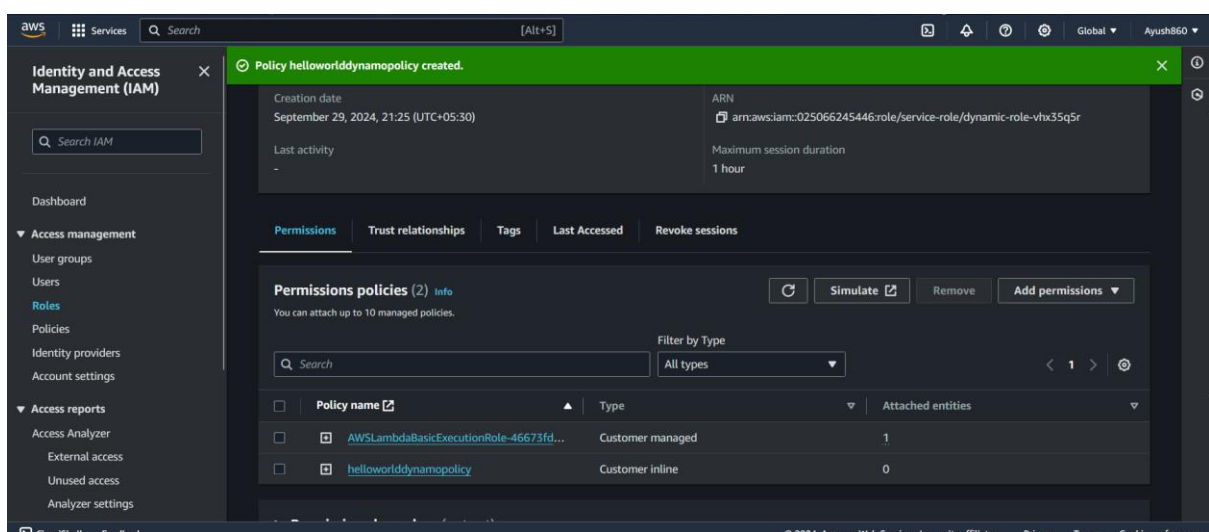
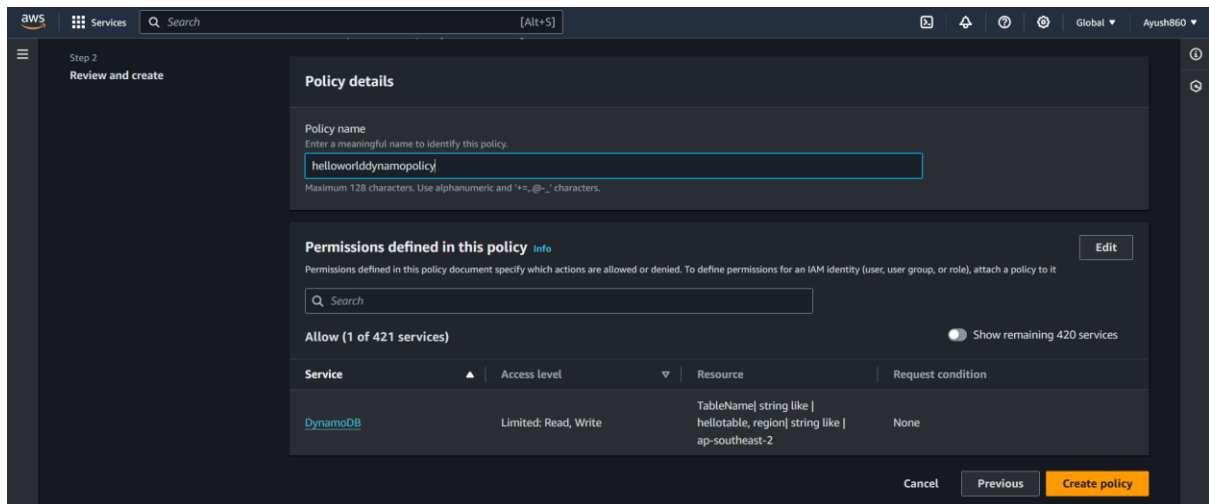
```

    "Action": [
        "dynamodb:PutItem",
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:Scan",
        "dynamodb:Query",
        "dynamodb:UpdateItem"
    ],
    "Resource": "YOUR-TABLE-ARN"
}
]
}

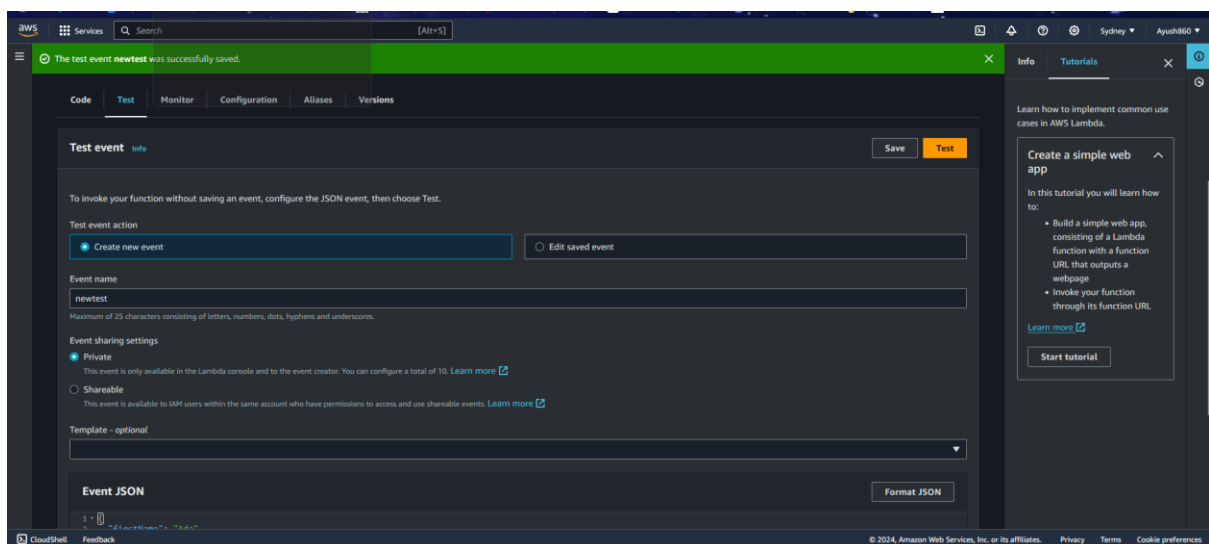
```



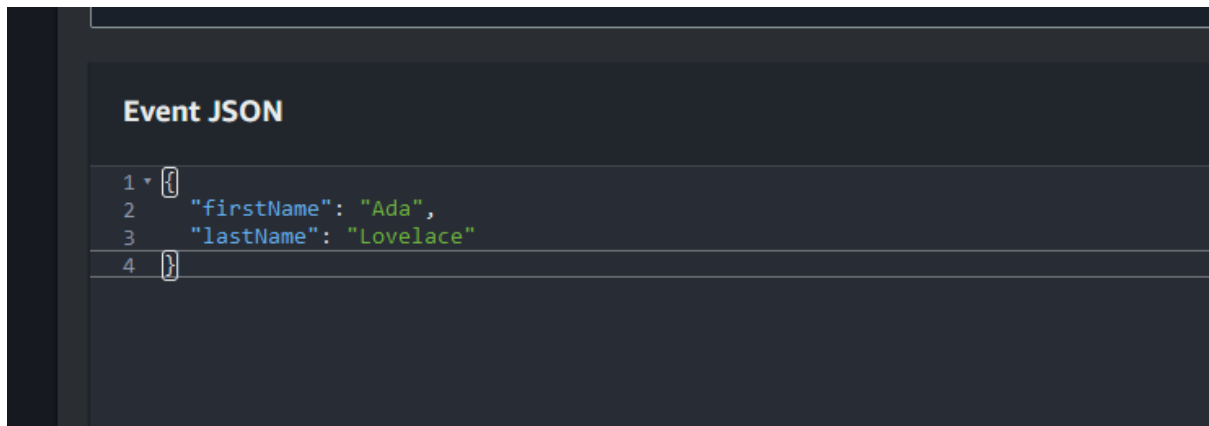
20. Click next > Now name the policy



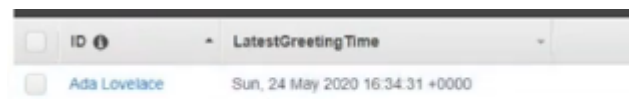
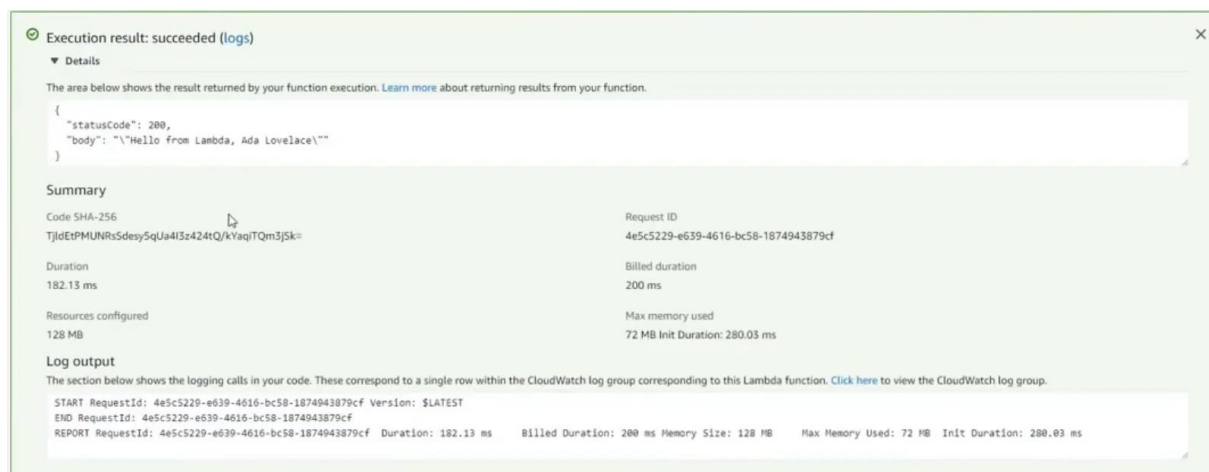
## 21. Create a test event







## Test results



We can see the results the dynamo db is adding these items

Now we can run our website



We added this

And we can see the item is getting added to our dynamo db table

<input type="checkbox"/>	ID ⓘ	LatestGreetingTime	▼
<input type="checkbox"/>	Ada Lovelace	Sun, 24 May 2020 16:34:31 +0000	
<input type="checkbox"/>	niraj kumar	Sun, 24 May 2020 16:49:17 +0000	