Cloud and Serverless Computing Project

T. SIDDHARTHA
2100030543

Cloud-Based Assignment Evaluator: Achieving Precision through Similarity Index Analysis

In this project we will be using the services like AWS Lambda, Amazon S3, AWS Textract, AWS IAM, AWS Cloud Watch.

AWS Lambda:

It is a serverless service provided by AWS.

There is no need to provision the resources and servers.

It is an event-driven architecture that is when the only when the event is triggered the function gets executed. It uses pay-as-you-go pricing .

It has automatic scaling.

Amazon S3:

It is an object storage service.

It is a storage service where the data can be stored and retrieved anytime and anywhere.

It has got the best durability 99.999999999 .

It has unlimited storage and where each object should comprise the size of 0 bytes to 5 Tb.

AWS Textract:

It is a machine learning tool that can automate the printed text and handwriting.

Extract text, forms, and tables from documents with structured data, using the Amazon Textract Document Analysis API.

AWS IAM:

A web service that helps you securely control access to AWS resources.

In this project we will be using **Roles.**

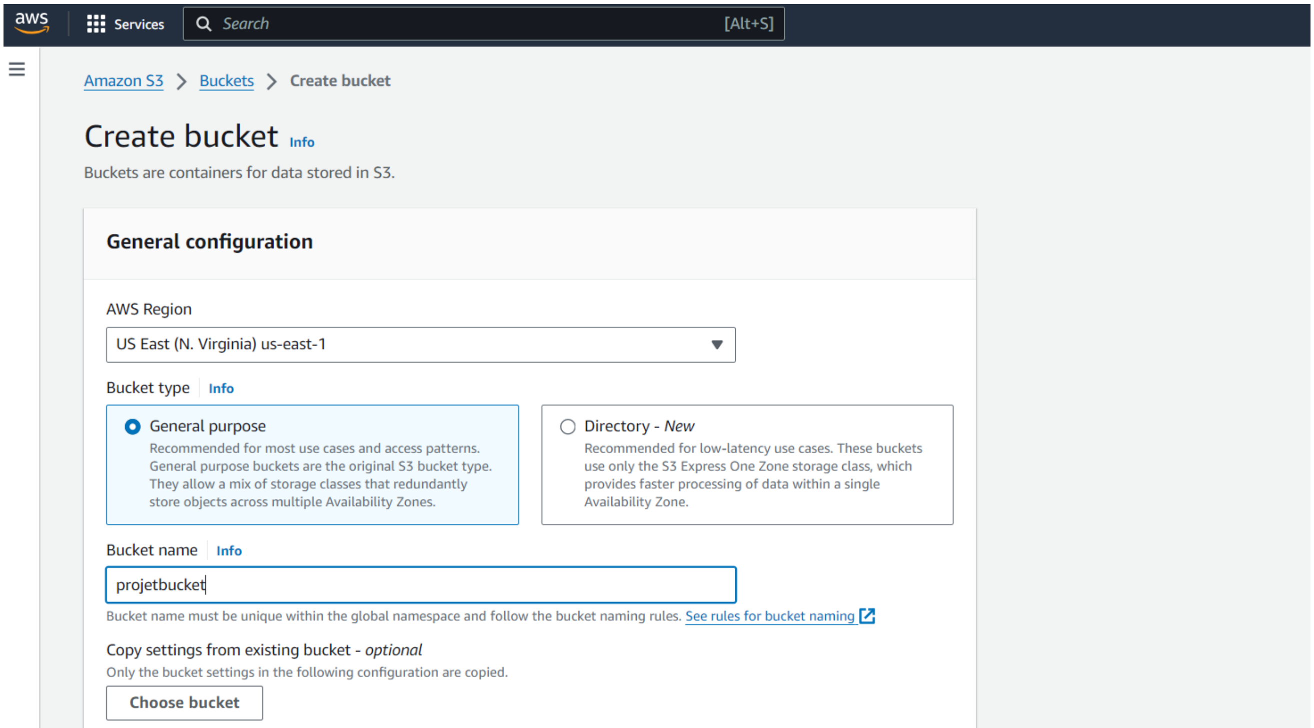
Roles: They provide temporary permission for accessing the service in AWS.

AWS CloudWatch:

Observe and monitor resources and applications on AWS, on premises, and on other clouds.
It helps you to view the history of events that have been performed by the users.

Steps of Project :

1. Create an S3 bucket.



Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type Info

● General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

○ Directory - *New*
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

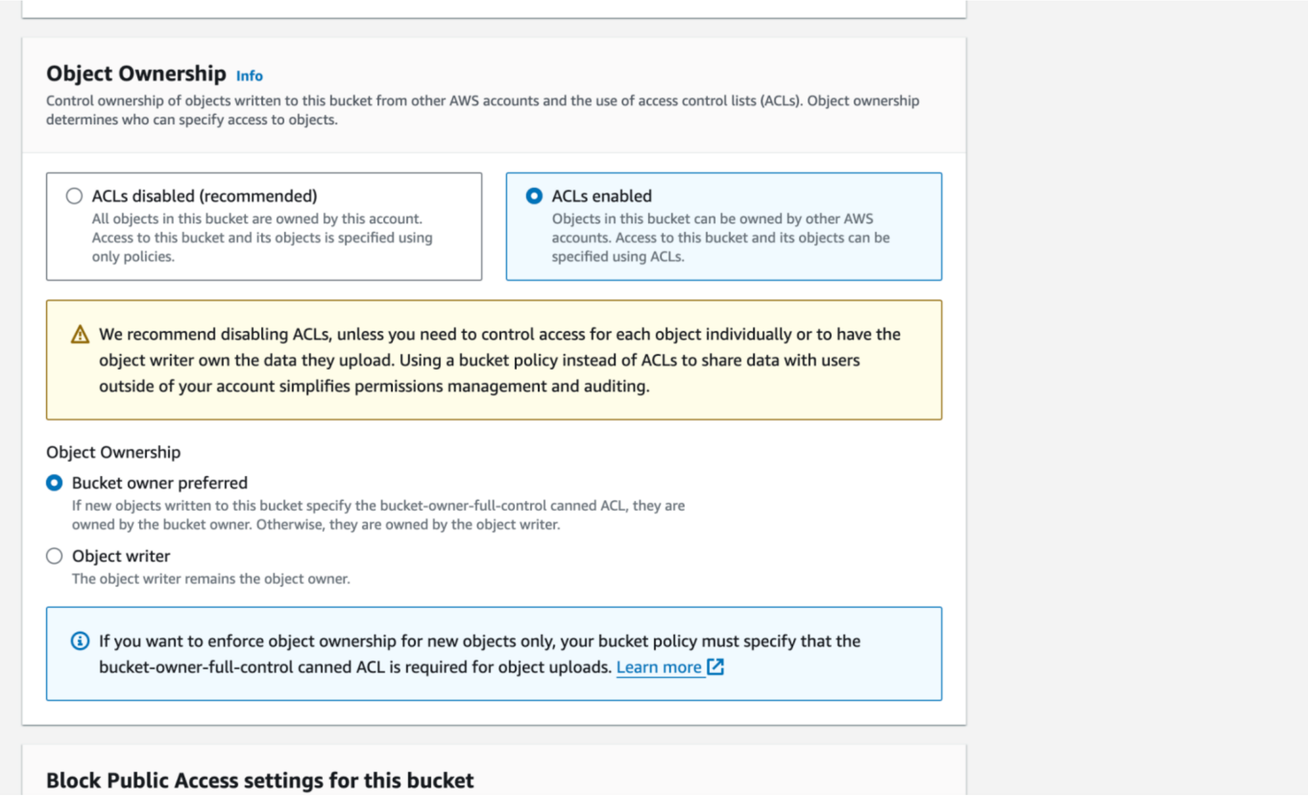
projetbucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

2. Enable the ACL (Access Control Lists):



Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

○ ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

● ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

⚠ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Object Ownership

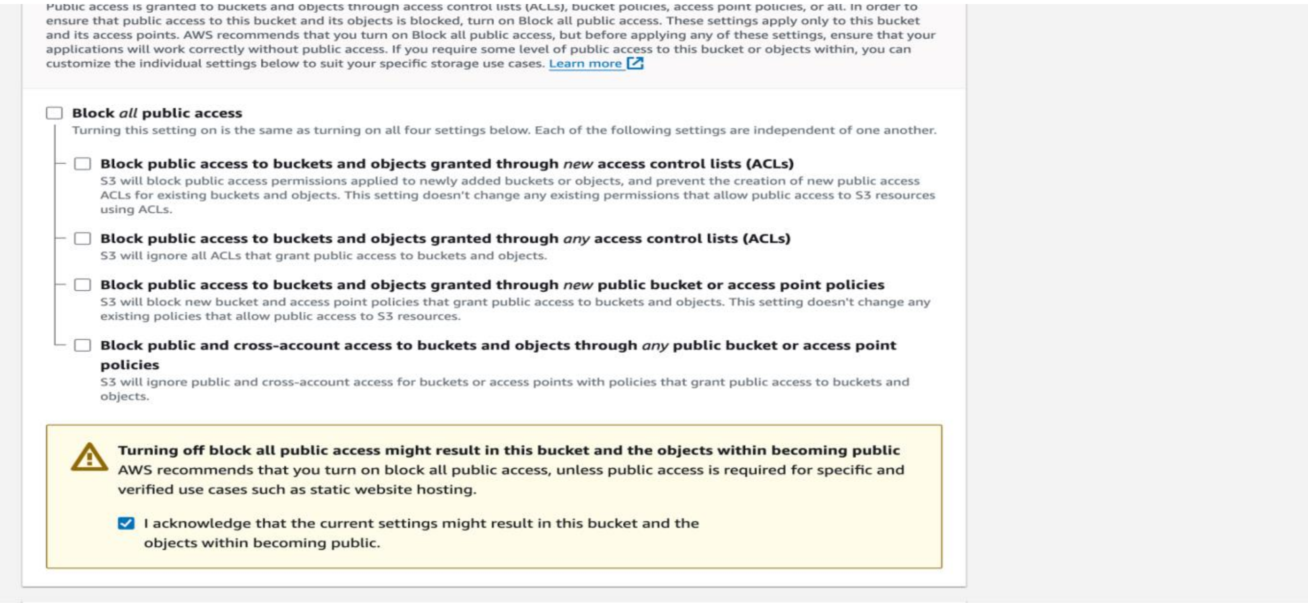
● Bucket owner preferred
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

○ Object writer
The object writer remains the object owner.

ⓘ If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. Learn more

Block Public Access settings for this bucket

3. Enable public access.



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.

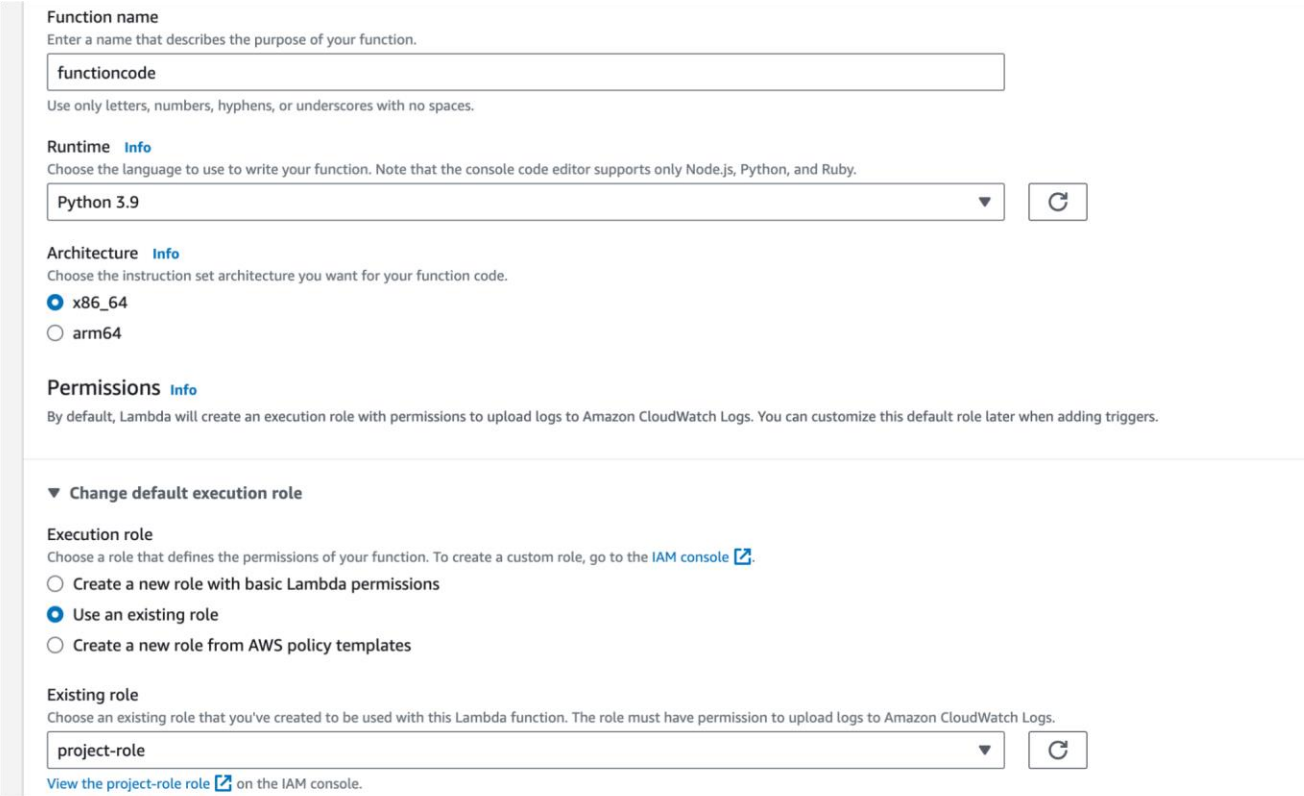
⚠ **Turning off block all public access might result in this bucket and the objects within becoming public**
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

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

4. Create the bucket.

Create a Lambda Function:

1. Create a function: Provide a name for the function
2. Choose the RunTime as : Python 3.9
3. And create a role: which includes the permissions Amazon S3, AWS Textract, AWS CloudWatch, AWS Lambda and also give full access to all the services.



Function name
Enter a name that describes the purpose of your function.
functioncode
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

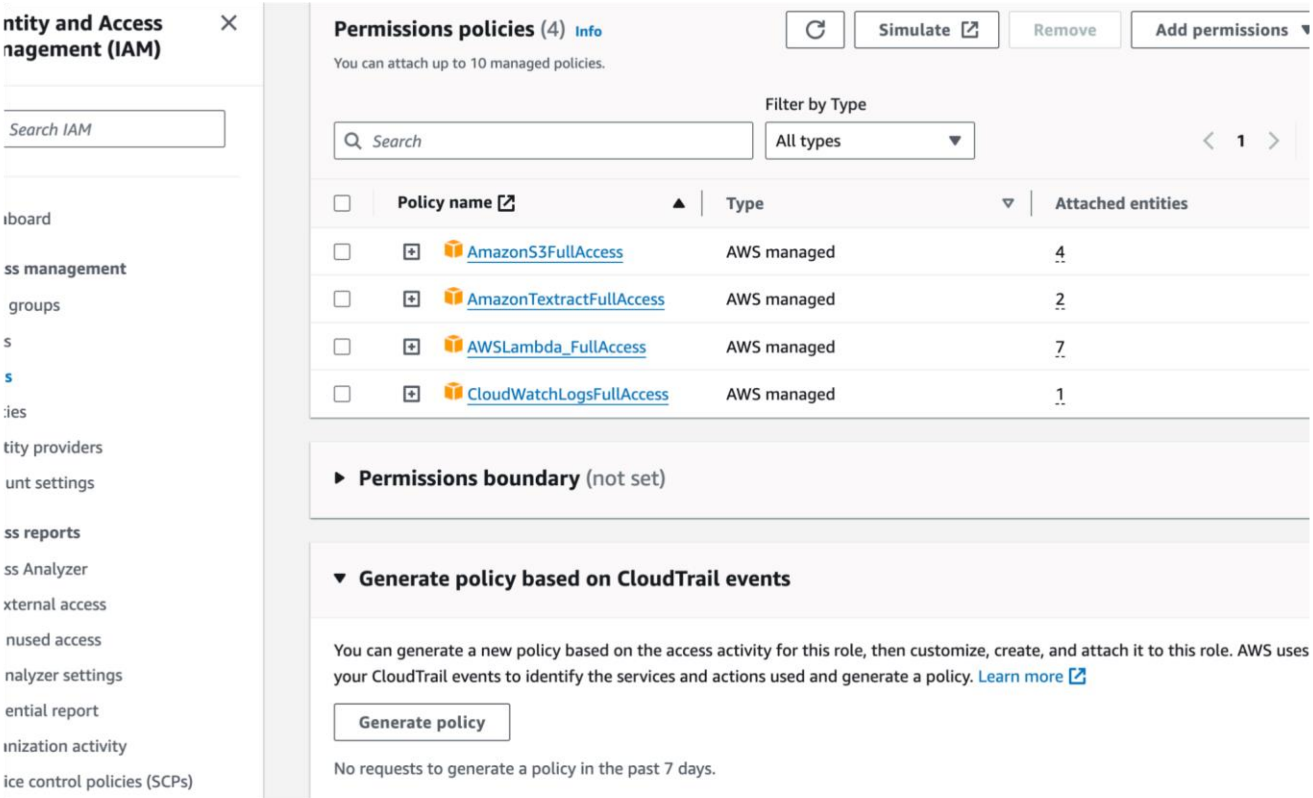
Architecture Info
Choose the instruction set architecture you want for your function code.
● x86_64
○ arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
○ Create a new role with basic Lambda permissions
● Use an existing role
○ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
project-role
View the project-role role on the IAM console.



ntity and Access
nagement (IAM)

Search IAM

board

ss management

groups

s

s

ies

tity providers

unt settings

ss reports

ss Analyzer

xternal access

nused access

nalyzer settings

ential report

nization activity

ice control policies (SCPs)

Permissions policies (4) Info

You can attach up to 10 managed policies.

Filter by Type

Search

All types

□	Policy name	Type	Attached entities
□	AmazonS3FullAccess	AWS managed	4
□	AmazonTextractFullAccess	AWS managed	2
□	AWSLambda_FullAccess	AWS managed	7
□	CloudWatchLogsFullAccess	AWS managed	1

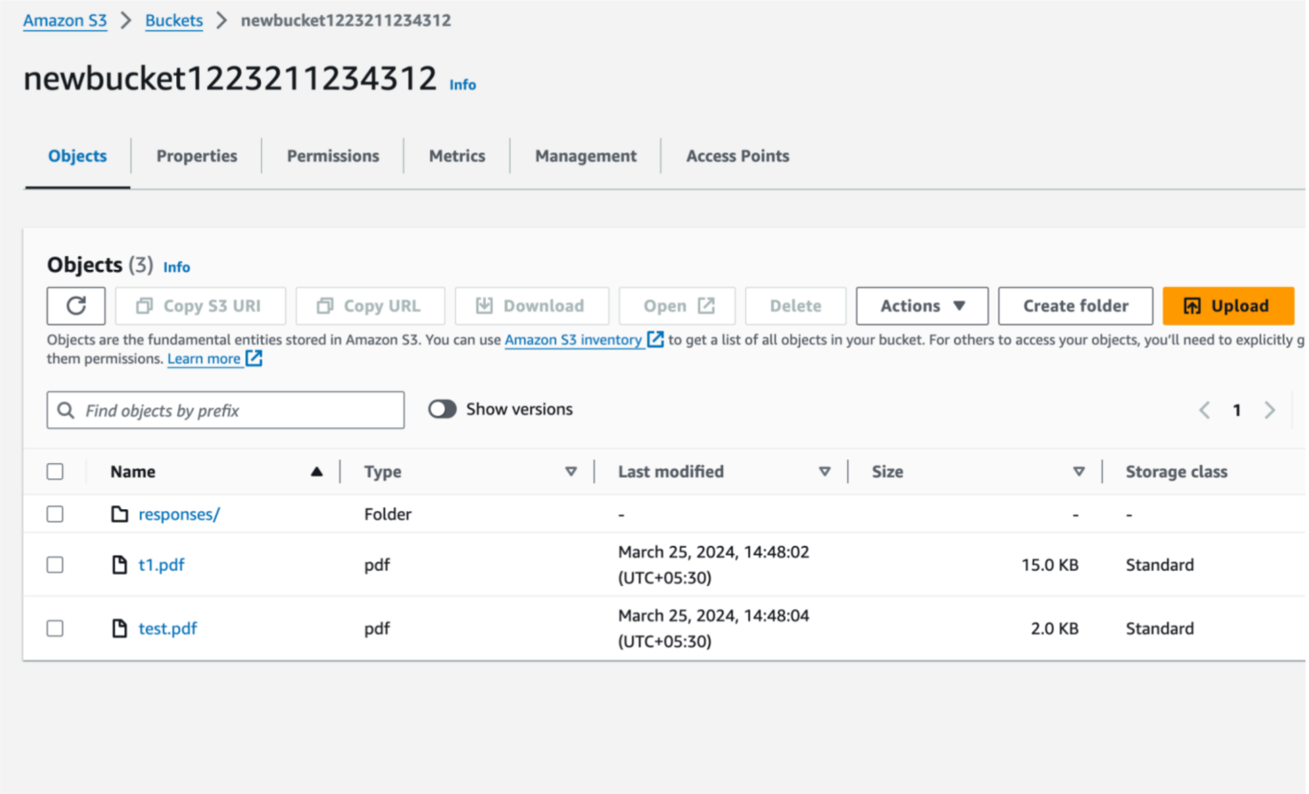
▶ **Permissions boundary** (not set)

▼ **Generate policy based on CloudTrail events**

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

No requests to generate a policy in the past 7 days.

And now add two files one as the Main file and the other as sub file in the S3 to find the similarity index between the files.



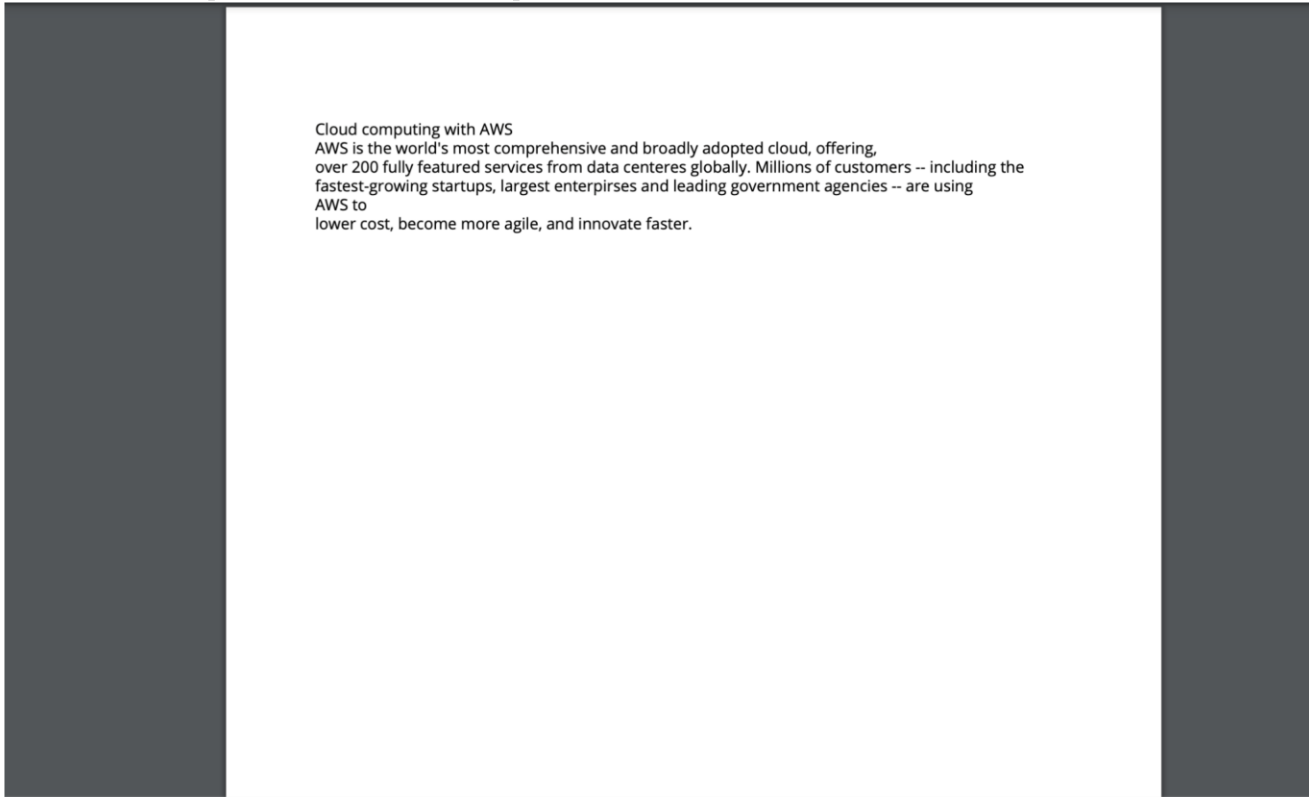
newbucket1223211234312 Info

Objects (3) Info

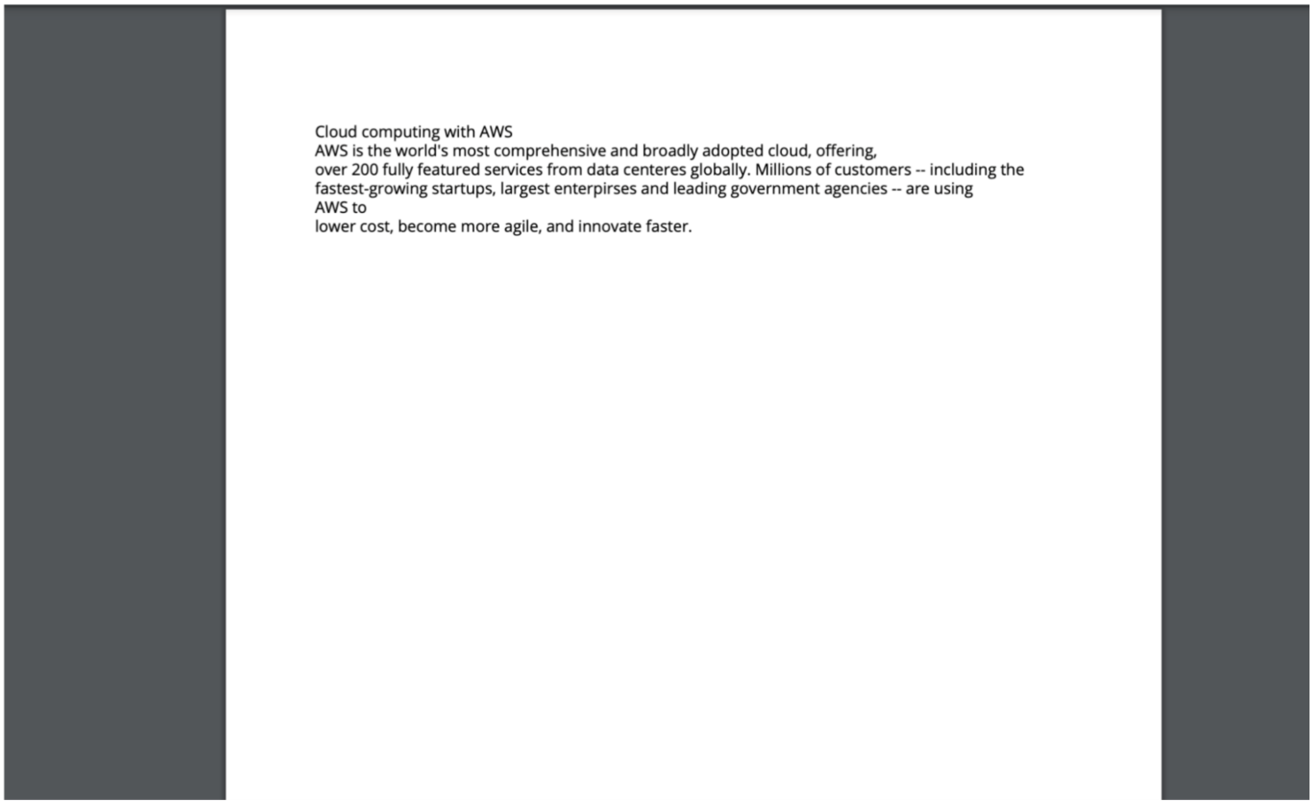
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly g
them permissions. Learn more

□	Name ▲	Type	Last modified	Size	Storage class
□	responses/	Folder	-	-	-
□	t1.pdf	pdf	March 25, 2024, 14:48:02 (UTC+05:30)	15.0 KB	Standard
□	test.pdf	pdf	March 25, 2024, 14:48:04 (UTC+05:30)	2.0 KB	Standard

Test.pdf: This is the main file T1.pdf: This is the sub file.

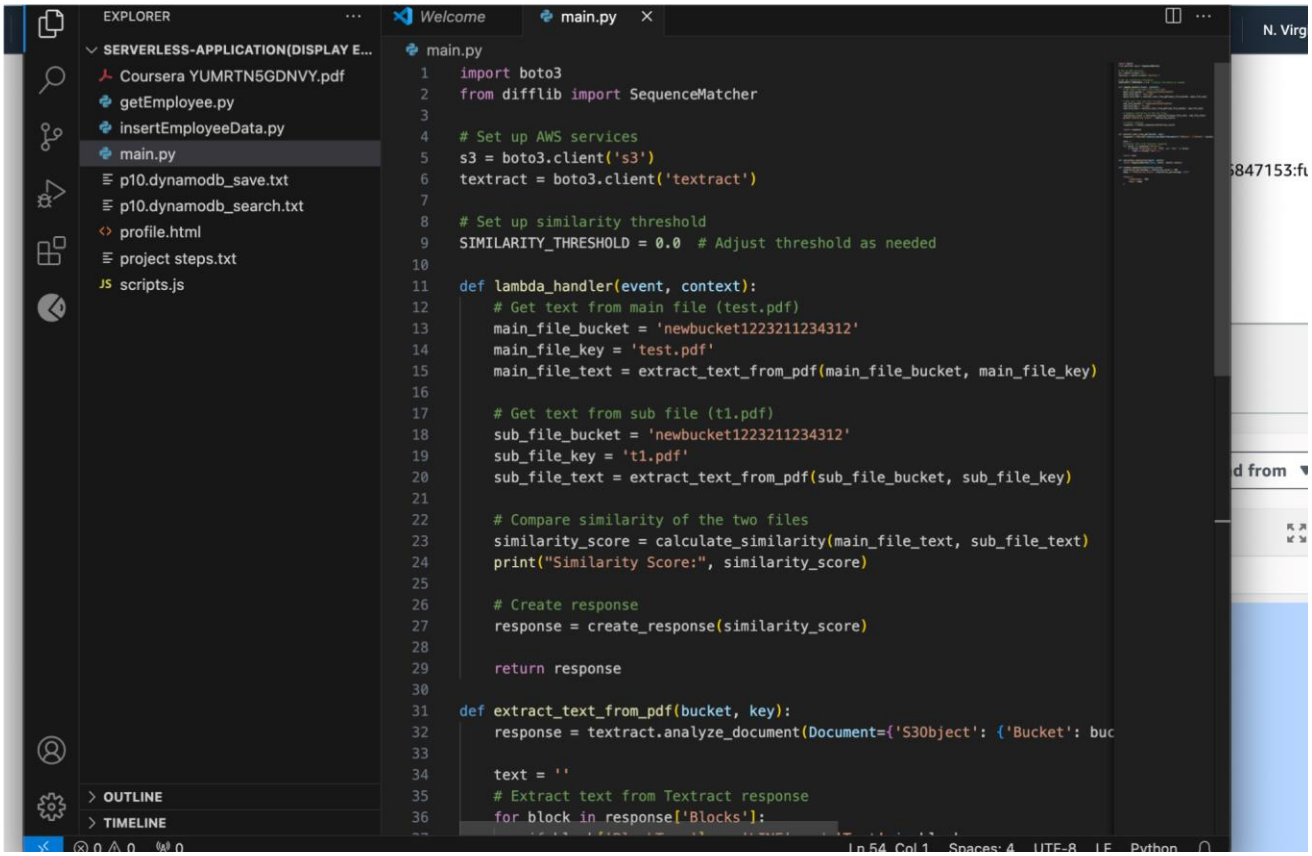


Cloud computing with AWS
AWS is the world's most comprehensive and broadly adopted cloud, offering,
over 200 fully featured services from data centeres globally. Millions of customers -- including the
fastest-growing startups, largest enterpirses and leading government agencies -- are using
AWS to
lower cost, become more agile, and innovate faster.



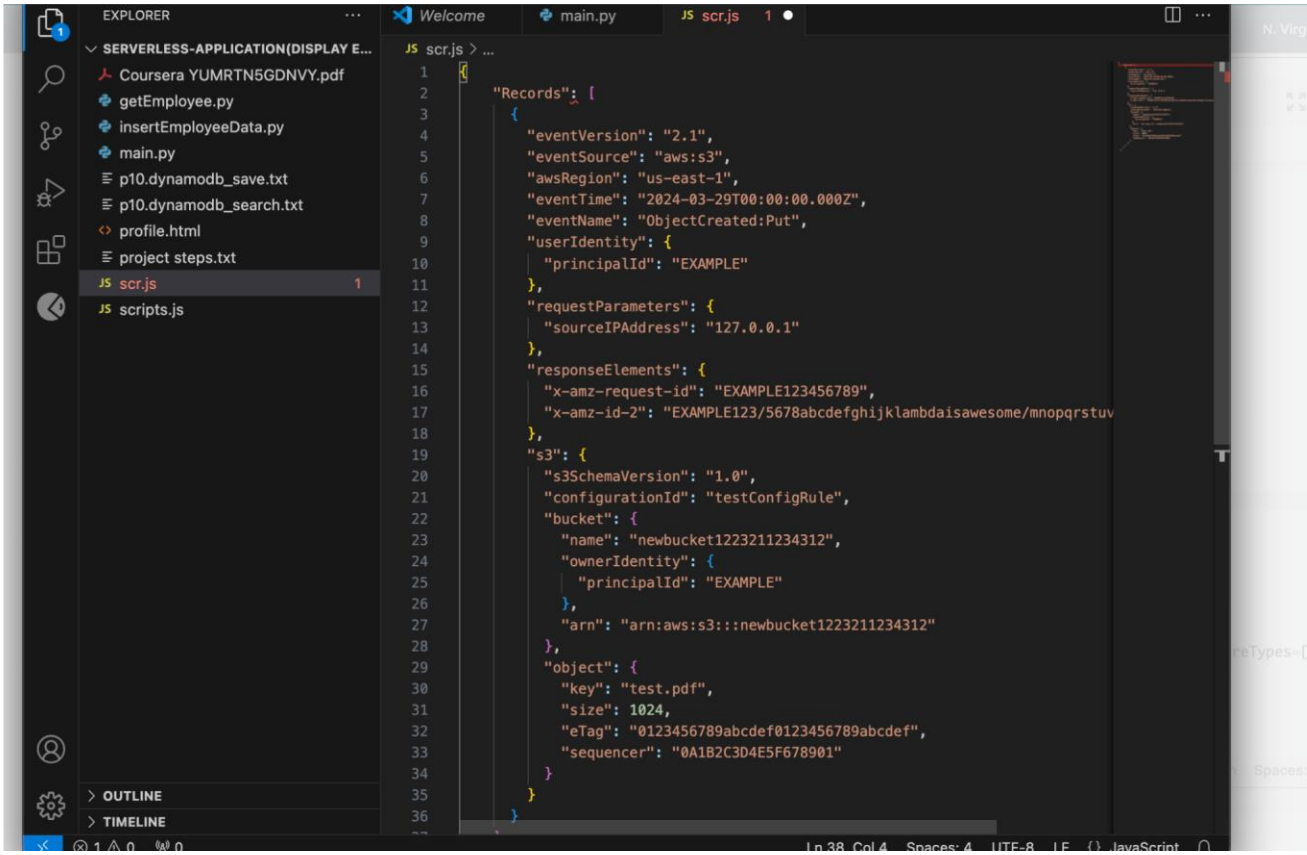
Cloud computing with AWS
AWS is the world's most comprehensive and broadly adopted cloud, offering,
over 200 fully featured services from data centeres globally. Millions of customers -- including the
fastest-growing startups, largest enterpirses and leading government agencies -- are using
AWS to
lower cost, become more agile, and innovate faster.

And now add the code in the lambda function to check the similarity index of both the files:

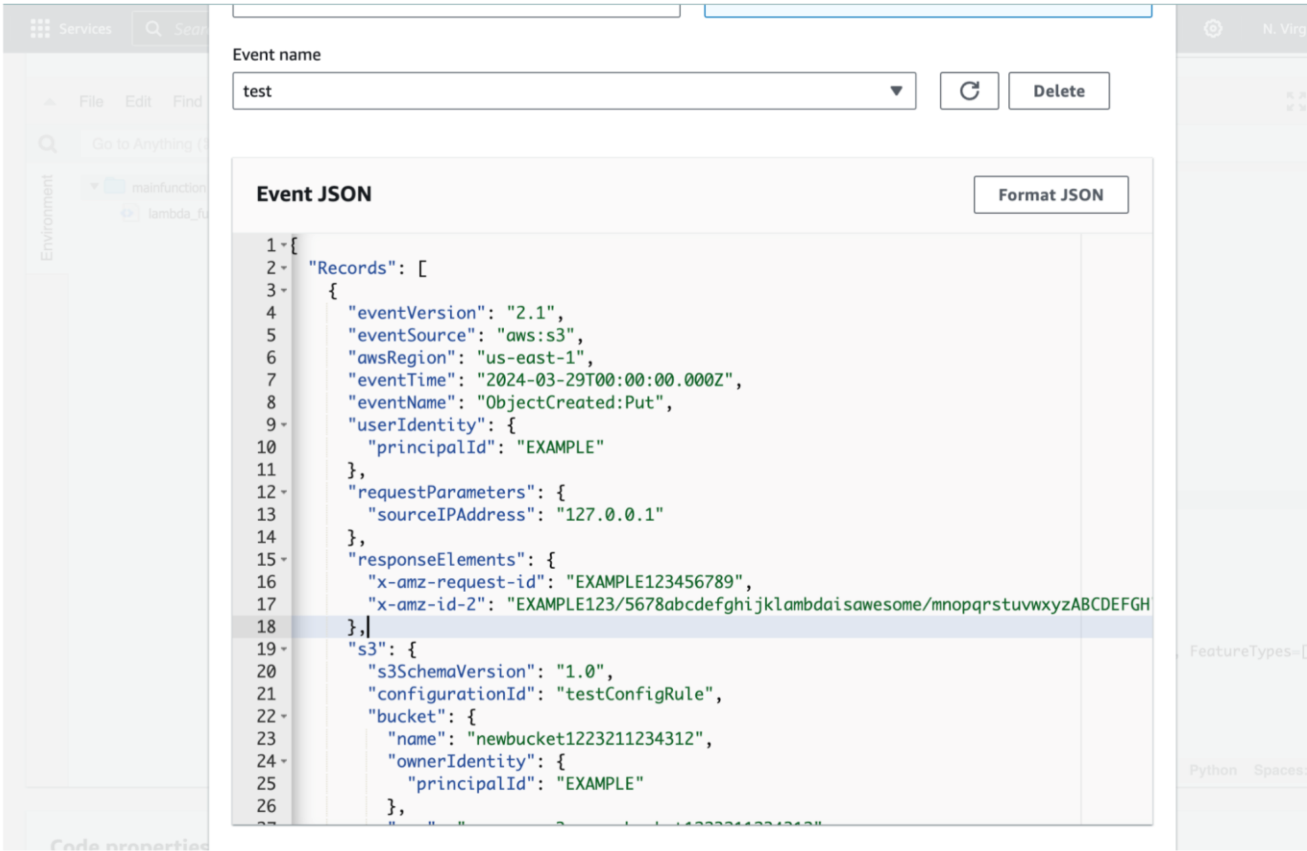


```
import boto3
from difflib import SequenceMatcher

# Set up AWS services
s3 = boto3.client('s3')
textract = boto3.client('textract')

# Set up similarity threshold
SIMILARITY_THRESHOLD = 0.0  # Adjust threshold as needed

def lambda_handler(event, context):
    # Get text from main file (test.pdf)
    main_file_bucket = 'newbucket1223211234312'
    main_file_key = 'test.pdf'
    main_file_text = extract_text_from_pdf(main_file_bucket, main_file_key)

    # Get text from sub file (t1.pdf)
    sub_file_bucket = 'newbucket1223211234312'
    sub_file_key = 't1.pdf'
    sub_file_text = extract_text_from_pdf(sub_file_bucket, sub_file_key)

    # Compare similarity of the two files
    similarity_score = calculate_similarity(main_file_text, sub_file_text)
    print("Similarity Score:", similarity_score)

    # Create response
    response = create_response(similarity_score)

    return response

def extract_text_from_pdf(bucket, key):
    response = textract.analyze_document(Document={'S3Object': {'Bucket': buc

    text = ''
    # Extract text from Textract response
    for block in response['Blocks']:
```

Create an Event to trigger the function:



```
{
    "Records": [
      {
        "eventVersion": "2.1",
        "eventSource": "aws:s3",
        "awsRegion": "us-east-1",
        "eventTime": "2024-03-29T00:00:00.000Z",
        "eventName": "ObjectCreated:Put",
        "userIdentity": {
          "principalId": "EXAMPLE"
        },
        "requestParameters": {
          "sourceIPAddress": "127.0.0.1"
        },
        "responseElements": {
          "x-amz-request-id": "EXAMPLE123456789",
          "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuv
        },
        "s3": {
          "s3SchemaVersion": "1.0",
          "configurationId": "testConfigRule",
          "bucket": {
            "name": "newbucket1223211234312",
            "ownerIdentity": {
              "principalId": "EXAMPLE"
            },
            "arn": "arn:aws:s3:::newbucket1223211234312"
          },
          "object": {
            "key": "test.pdf",
            "size": 1024,
            "eTag": "0123456789abcdef0123456789abcdef",
            "sequencer": "0A1B2C3D4E5F678901"
          }
        }
      }
```



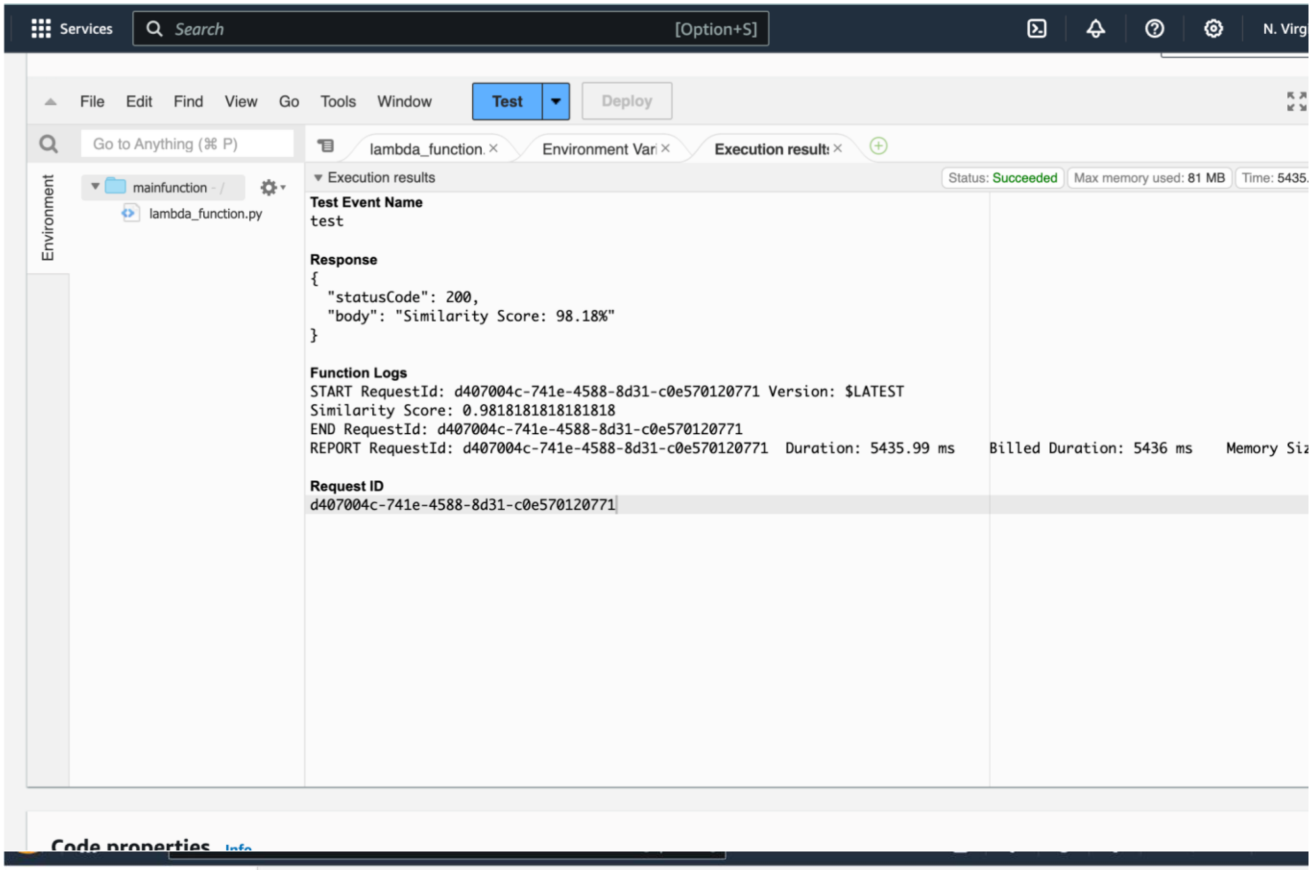
Event name

test

Event JSON

```json
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-03-29T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "newbucket1223211234312",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
```

And now test the function:



Execution results

Status: Succeeded | Max memory used: 81 MB | Time: 5435.

Test Event Name

```
test
```

Response

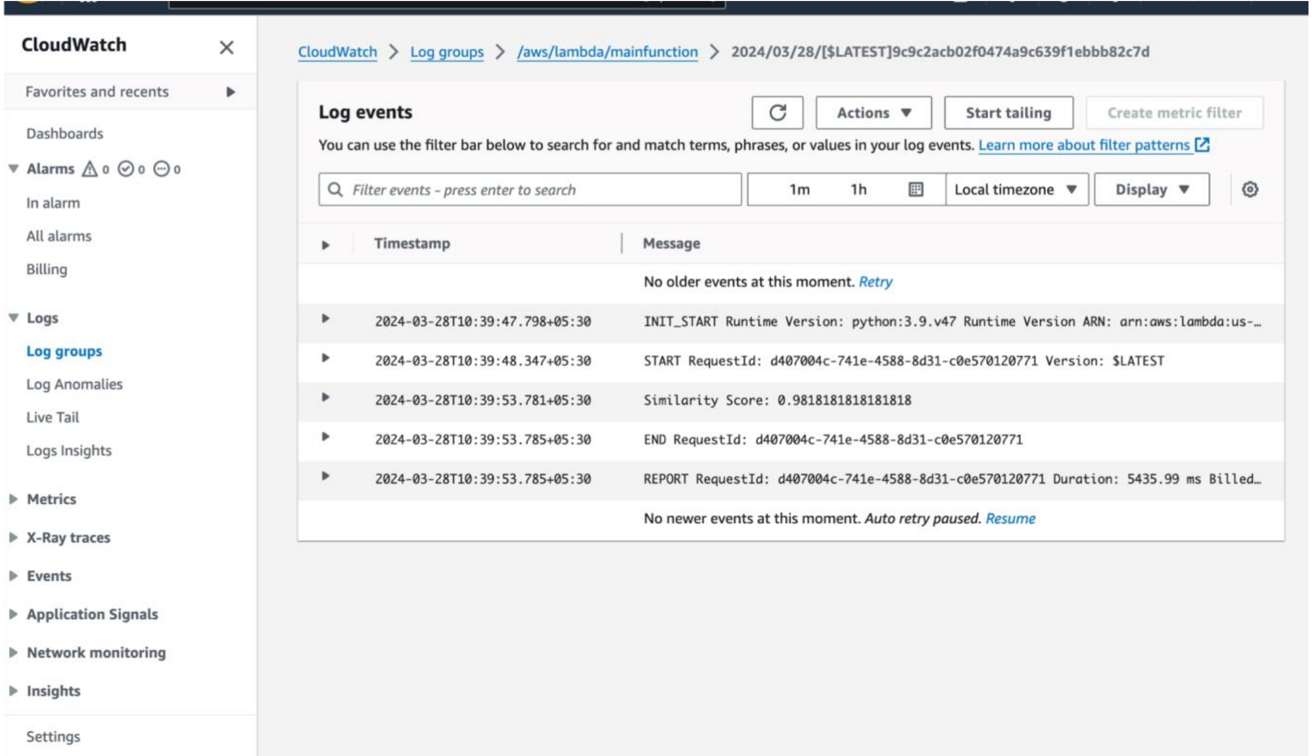
```
{
  "statusCode": 200,
  "body": "Similarity Score: 98.18%"
}
```

Function Logs

```
START RequestId: d407004c-741e-4588-8d31-c0e570120771 Version: $LATEST
Similarity Score: 0.9818181818181818
END RequestId: d407004c-741e-4588-8d31-c0e570120771
REPORT RequestId: d407004c-741e-4588-8d31-c0e570120771  Duration: 5435.99 ms    Billed Duration: 5436 ms     Memory Siz
```

Request ID

```
d407004c-741e-4588-8d31-c0e570120771
```



CloudWatch > Log groups > /aws/lambda/mainfunction > 2024/03/28/[$LATEST]9c9c2acb02f0474a9c639f1ebbb82c7d

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                        Message
                                 No older events at this moment. Retry
2024-03-28T10:39:47.798+05:30    INIT_START Runtime Version: python:3.9.v47 Runtime Version ARN: arn:aws:lambda:us-…
2024-03-28T10:39:48.347+05:30    START RequestId: d407004c-741e-4588-8d31-c0e570120771 Version: $LATEST
2024-03-28T10:39:53.781+05:30    Similarity Score: 0.9818181818181818
2024-03-28T10:39:53.785+05:30    END RequestId: d407004c-741e-4588-8d31-c0e570120771
2024-03-28T10:39:53.785+05:30    REPORT RequestId: d407004c-741e-4588-8d31-c0e570120771 Duration: 5435.99 ms Billed…
                                 No newer events at this moment. Auto retry paused. Resume
```

The status code is 200 and the function worked !!

The similarity index between both the files is 98.18%.