

# Cloud and Serverless Computing Project

D. Jaswanth  
2100030129

## 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**.

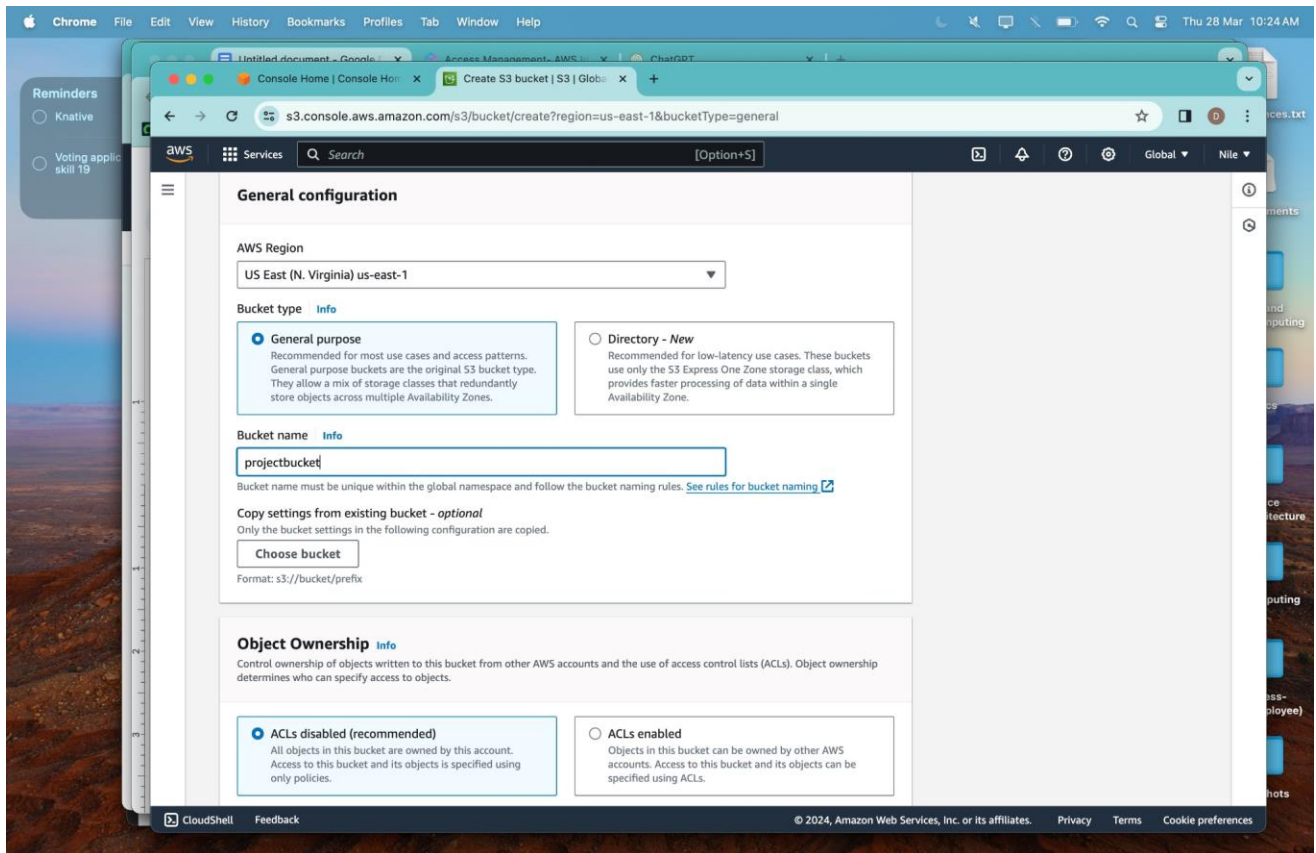
**Roles:** They provide temporary permission for accessing the service in AWS.

### 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.



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 Texttract, AWS CloudWatch, AWS

**Function name**  
Enter a name that describes the purpose of your function.

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.

▼

↺

[View the project-role role](#) on the IAM console.

Lambda and also give full access to all the services.

## Entity and Access management (IAM)

Search IAM

Dashboard

Groups management

Groups

Users

Roles

Groups

Identity providers

Account settings

Groups reports

Groups Analyzer

External access

Managed access

Analyzer settings

Entitlement report

Initialization activity

Service control policies (SCPs)

### Permissions policies (4) [Info](#)

You can attach up to 10 managed policies.



Simulate [↗](#)

Remove

Add permissions [↕](#)

Filter by Type

All types

< 1 >

<input type="checkbox"/>	Policy name <a href="#">↗</a>	Type	Attached entities
<input type="checkbox"/>	<a href="#">AmazonS3FullAccess</a>	AWS managed	4
<input type="checkbox"/>	<a href="#">AmazonTextractFullAccess</a>	AWS managed	2
<input type="checkbox"/>	<a href="#">AWSLambda_FullAccess</a>	AWS managed	7
<input type="checkbox"/>	<a href="#">CloudWatchLogsFullAccess</a>	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](#) [↗](#)

Generate policy

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
- Properties
- Permissions
- Metrics
- Management
- Access Points

## Objects (3) [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](#)

Show versions

< 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	responses/	Folder	-	-	-
<input type="checkbox"/>	t1.pdf	pdf	March 25, 2024, 14:48:02 (UTC+05:30)	15.0 KB	Standard
<input type="checkbox"/>	test.pdf	pdf	March 25, 2024, 14:48:04 (UTC+05:30)	2.0 KB	Standard

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

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 centers globally. Millions of customers -- including the fastest-growing startups, largest enterprises and leading government agencies -- are using AWS to lower cost, become more agile, and innovate faster.

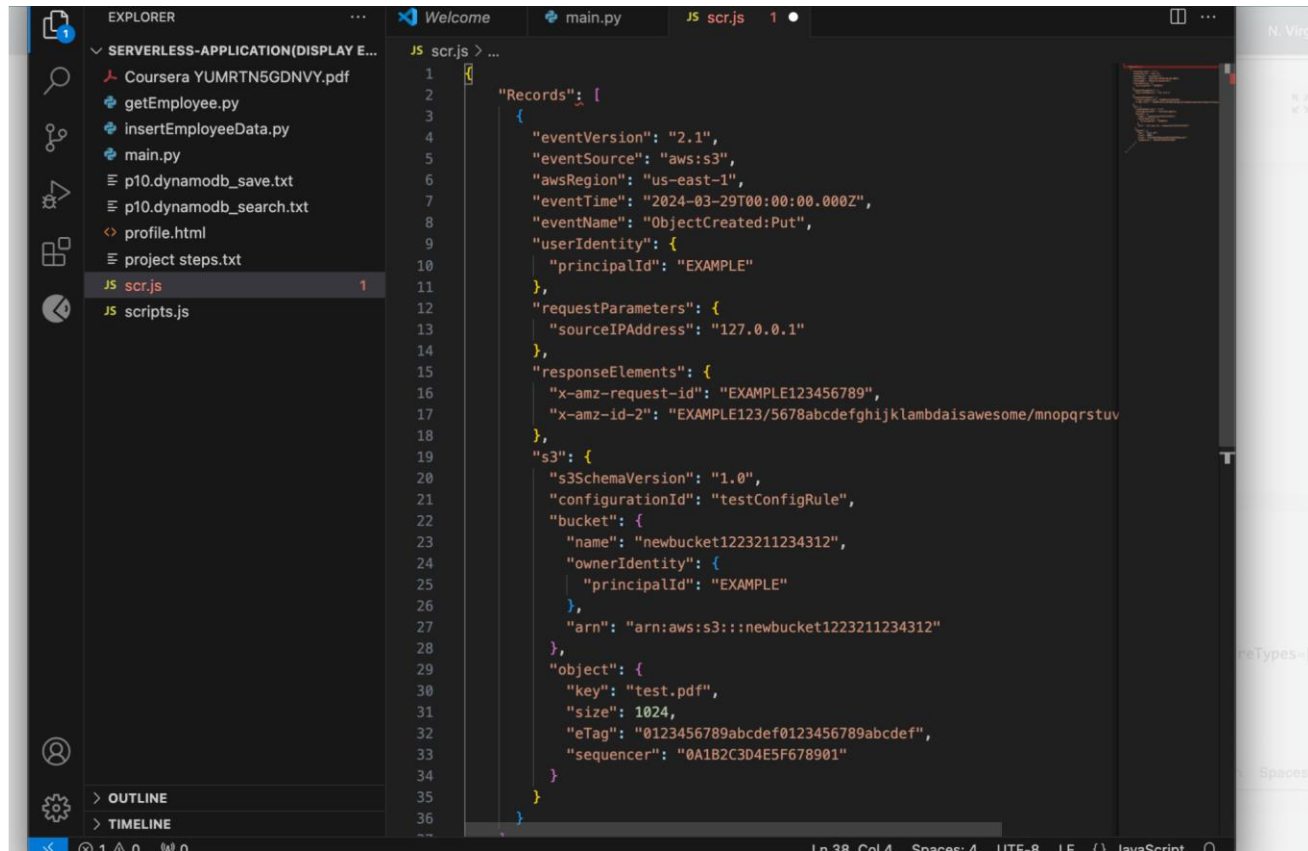
And now add the code in the lambda function to check the similarity index of both the files:

The image shows a VS Code editor window with a dark theme. The Explorer sidebar on the left displays a project structure for 'SERVERLESS-APPLICATION(DISPLAY E...'. The main editor area shows the 'main.py' file with the following Python code:

```
1 import boto3
2 from difflib import SequenceMatcher
3
4 # Set up AWS services
5 s3 = boto3.client('s3')
6 textextract = boto3.client('textextract')
7
8 # Set up similarity threshold
9 SIMILARITY_THRESHOLD = 0.0 # Adjust threshold as needed
10
11 def lambda_handler(event, context):
12     # Get text from main file (test.pdf)
13     main_file_bucket = 'newbucket1223211234312'
14     main_file_key = 'test.pdf'
15     main_file_text = extract_text_from_pdf(main_file_bucket, main_file_key)
16
17     # Get text from sub file (t1.pdf)
18     sub_file_bucket = 'newbucket1223211234312'
19     sub_file_key = 't1.pdf'
20     sub_file_text = extract_text_from_pdf(sub_file_bucket, sub_file_key)
21
22     # Compare similarity of the two files
23     similarity_score = calculate_similarity(main_file_text, sub_file_text)
24     print("Similarity Score:", similarity_score)
25
26     # Create response
27     response = create_response(similarity_score)
28
29     return response
30
31 def extract_text_from_pdf(bucket, key):
32     response = textextract.analyze_document(Document={'S3Object': {'Bucket': buc
33
34     text = ''
35     # Extract text from Textextract response
36     for block in response['Blocks']:
```

Create an Event to trigger the function:





Services

Search

File Edit Find

Go to Anything

Environment

mainfunction

lambda\_fu

Code properties

Event name

test

Refresh

Delete

Event JSON

Format JSON

```
1 {
2   "Records": [
3     {
4       "eventVersion": "2.1",
5       "eventSource": "aws:s3",
6       "awsRegion": "us-east-1",
7       "eventTime": "2024-03-29T00:00:00.000Z",
8       "eventName": "ObjectCreated:Put",
9       "userIdentity": {
10        "principalId": "EXAMPLE"
11      },
12      "requestParameters": {
13        "sourceIPAddress": "127.0.0.1"
14      },
15      "responseElements": {
16        "x-amz-request-id": "EXAMPLE123456789",
17        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklmbdaisawesom/mnopqrstuvwxyzABCDEFGH
18      },
19    },
20    "s3": {
21      "s3SchemaVersion": "1.0",
22      "configurationId": "testConfigRule",
23      "bucket": {
24        "name": "newbucket1223211234312",
25        "ownerIdentity": {
26          "principalId": "EXAMPLE"
27        }
28      }
29    }
30  ]
31 }
```

FeatureTypes=

Python Spaces=

And now test the function:

Services Search [Option+S]

File Edit Find View Go Tools Window Test Deploy

Go to Anything (% P)

Environment

- mainfunction - /
  - lambda\_function.py

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 Size: 128 MB

Request ID

d407004c-741e-4588-8d31-c0e570120771

Code properties info

The status code is 200 and the function worked !!

The similarity index between both the files is 98.18%.

aws Services Search [Option+S] N. Virginia Nile

**CloudWatch** X

Favorites and recents ▶

Dashboards

▼ **Alarms** 0 0 0 0

In alarm

All alarms

Billing

▼ **Logs**

**Log groups**

Log Anomalies

Live Tail

Logs Insights

▶ **Metrics**

▶ **X-Ray traces**

▶ **Events**

▶ **Application Signals**

▶ **Network monitoring**

▶ **Insights**

Settings

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

**Log events** [Refresh] [Actions ▼] [Start tailing] [Create metric filter]

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](#)

Q Filter events - press enter to search 1m 1h [Grid] Local timezone ▼ Display ▼ [Settings]

▶	Timestamp	Message
		No older events at this moment. <a href="#">Retry</a>
▶	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. <i>Auto retry paused.</i> <a href="#">Resume</a>

And check the results in the Cloud Watch log: 98.18%.

GitHub Profile : <https://github.com/Nilesh-27/Home-Assignment-Similarity-Index-AWS>