

Name- Dipangshu Kundu

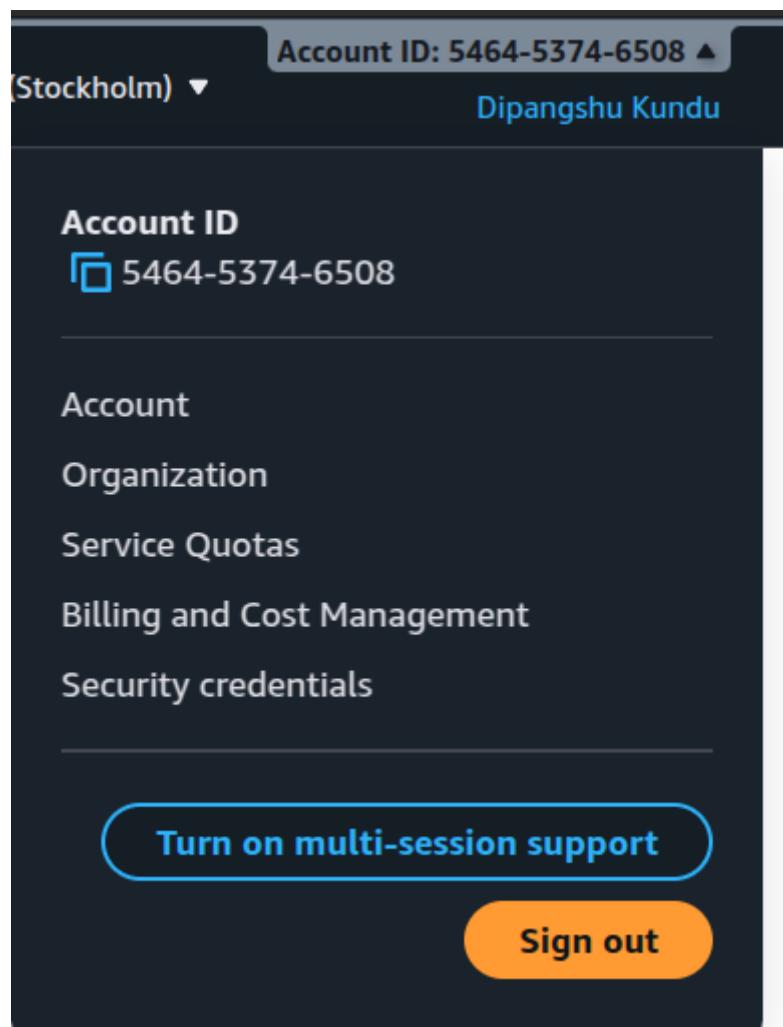
Reg No- 22BBS0148

AWS MID TERM

DATE:- 10/9/2025

Q) Create a serverless file content-type detector using S3 and AWS Lambda. Create an S3 bucket and configure S3 Event Notifications to trigger a Lambda function upon file upload. Write a Lambda function in Python or Node.js to retrieve the uploaded file's metadata and log the content type to CloudWatch. Test by uploading at least three different file types and verify that the logs display the detected content types. Document all configurations, code snippets, and test results with screenshots.

Solution:-



1. Create an S3 Bucket

1. Go to **AWS Console** → **S3** → **Create Bucket**.
2. Give it a unique name (e.g., my-lambda-demo-bucket).
3. Choose a region (keep it the same as Lambda).
4. Leave defaults, but **uncheck "Block all public access"** if you need public access.
5. **Create the bucket.**

The screenshot shows the 'Create bucket' configuration page in the AWS S3 console. It includes sections for 'General configuration', 'Object Ownership', and a summary message at the bottom.

General configuration:

- AWS Region:** Europe (Stockholm) eu-north-1
- Bucket type:** General purpose (selected)
- Bucket name:** midtermdipangshufinal
- Copy settings from existing bucket - optional:** Choose bucket
- Format:** s3://bucket/prefix

Object Ownership:

- Object Ownership:** Bucket owner enforced
- ACLs:** ACLs disabled (recommended)
- 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.

Summary message: Successfully created bucket "midtermdipangshufinal". To upload files and folders, or to configure additional bucket settings, choose View details.

The screenshot shows the 'Buckets' page in the AWS S3 console, displaying the newly created bucket 'midtermdipangshufinal'.

General purpose buckets (5)

- Actions:** Copy ARN, Empty, Delete, Create bucket
- Search:** Find buckets by name

Account snapshot: Updated daily

Storage Lens provides visibility into storage usage and activity trends.

2. Upload a Test File to S3

- After bucket creation → **Upload** → choose a test file (e.g., test.txt).
- **We'll use this file to trigger Lambda.**

The screenshot shows the AWS S3 console interface. At the top, there's a green success message: "Upload succeeded. For more information, see the Files and folders table." Below this, a summary table shows one file uploaded successfully (S3://midtermdipangshufinal/test.txt) and zero files failed. Under the "Files and folders" tab, a table lists the uploaded file: test.txt, which is 29.0 B in size and has a status of "Succeeded".

Name	Folder	Type	Size	Status	Error
test.txt	-	text/plain	29.0 B	Succeeded	-

3. Create a Lambda Function

1. Go to **AWS Console** → **Lambda** → **Create Function**.
2. Choose **Author from scratch**.
3. Name: MyS3LambdaFunction.
4. Runtime: **Python 3.x** (or Node.js if you prefer).
5. Permissions: Create or use an existing role with **S3 and CloudWatch access**.
6. **Click Create Function.**

Create function Info

Choose one of the following options to create your function.

- Author from scratch

Start with a simple Hello World example.
- Use a blueprint

Build a Lambda application from sample code and configuration presets for common use cases.
- Container image

Select a container image to deploy for your function.

Basic information

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

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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 3.13`

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

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.

Additional configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

[Cancel](#) [Create function](#)

4. Add S3 Trigger to Lambda

1. In your Lambda function → **Configuration** → **Triggers** → **Add Trigger**.
2. Select **S3**.
3. Choose your bucket (my-lambda-demo-bucket).
4. Event type: **PUT (All object create events)**.
5. Check "Enable trigger".
6. **Save**.

MyS3LambdaFunction

Function overview Info

Diagram **Template**

MyS3LambdaFunction

Layers (0)

S3

+ Add destination

+ Add trigger

Throttle **Copy ARN** **Actions**

Description
-

Last modified
2 hours ago

Function ARN
`arn:aws:lambda:eu-north-1:546453746508:function:MyS3LambdaFunction`

Function URL Info
-

Code **Test** **Monitor** **Configuration** **Aliases** **Versions**

Code source Info

lambda_function.py

Open in Visual Studio Code **Upload from**

EXPLORER

5. Write lambda Code

```

import json
import boto3

# Initialize the S3 client outside the handler to reuse the connection
s3 = boto3.client('s3')

def lambda_handler(event, context):
    print("Received event:", json.dumps(event))

    for record in event['Records']:
        # Extract bucket and object key from the event record
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']

        try:
            # Use head_object to get object metadata without downloading the file
            response = s3.head_object(Bucket=bucket, Key=key)
            content_type = response['ContentType']

            # Log the content type to CloudWatch logs
            print(f"Content-Type: {content_type} for object {key} in bucket {bucket}")
        except Exception as e:
            print(f"Error processing object {key} in bucket {bucket}: {e}")

```

**import json
import boto3**

**# Initialize the S3 client outside the handler to reuse the connection
s3 = boto3.client('s3')**

```

def lambda_handler(event, context):  
    print("Received event:", json.dumps(event))

```

```
for record in event['Records']:
    # Extract bucket and object key from the event record
    bucket = record['s3']['bucket']['name']
    key = record['s3']['object']['key']

    try:
        # Use head_object to get object metadata without downloading
        # the file
        response = s3.head_object(Bucket=bucket, Key=key)
        content_type = response['ContentType']

        # Log the content type to CloudWatch
        print(f"File '{key}' in bucket '{bucket}' has a Content-Type of:
{content_type}")

    except Exception as e:
        print(f"Error getting metadata for file '{key}': {e}")

    return {
        'statusCode': 200,
        'body': json.dumps('Content-type detection complete.')
    }
```

6. Test the Integration

1. We Go to our S3 bucket.
2. Upload a new file (e.g., hello.txt).

The screenshot shows the AWS S3 console interface. At the top, there's a green success message: "Upload succeeded" with a link to "See files and folders table". Below this, the bucket name "midtermdipangshufinal" is displayed. The "Objects" tab is selected, showing a table of uploaded files:

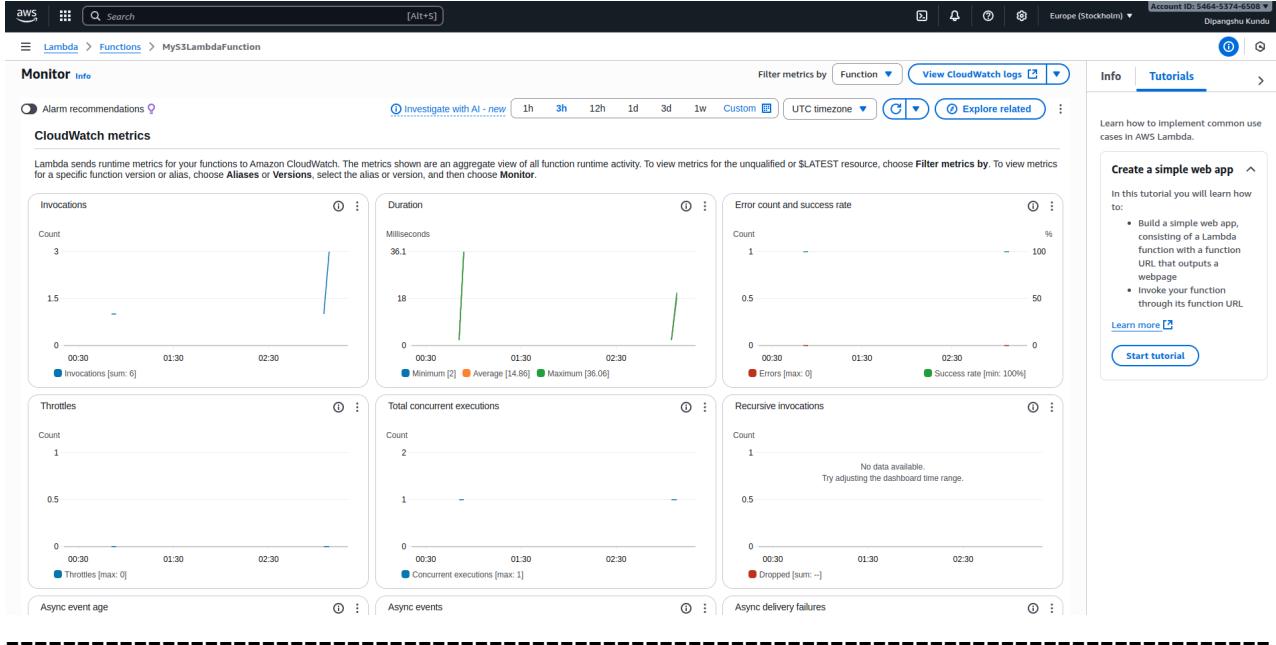
Name	Type	Last modified	Size	Storage class
hello.txt	txt	September 10, 2025, 08:35:11 (UTC+05:30)	57.0 B	Standard
test.txt	txt	September 10, 2025, 08:24:55 (UTC+05:30)	29.0 B	Standard

7. UPLOADING THREE DIFFERENT TYPES OF FILES

The screenshot shows the AWS S3 console interface. At the top, there's a green success message: "Successfully deleted objects" with a link to "View details below.". Below this, the bucket name "midtermdipangshufinal" is displayed. The "Objects" tab is selected, showing a table of uploaded files:

Name	Type	Last modified	Size	Storage class
alligator-zoo-bg.png	png	September 10, 2025, 08:38:25 (UTC+05:30)	550.5 KB	Standard
hello.txt	txt	September 10, 2025, 08:35:11 (UTC+05:30)	57.0 B	Standard
logo.svg	svg	September 10, 2025, 08:38:25 (UTC+05:30)	2.6 KB	Standard
test.txt	txt	September 10, 2025, 08:24:55 (UTC+05:30)	29.0 B	Standard

8. → Go back to Lambda → Monitor → Logs (CloudWatch).
→ We should see logs showing the file name, size, and bucket



→ VERIFYING THAT THE LOGS ARE OF CORRECT DETECTED TYPES

hello.txt

Log groups	<ul style="list-style-type: none"> ▶ 2025-09-10T03:05:11.589Z START RequestId: f8624282-5e11-4a4d-bd7c-9c360310ed0d Version: \$LATEST ▶ 2025-09-10T03:05:11.589Z Event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "eu-north-1", "eventTime": "2025-09-10T03:05:10.375Z", "eventName": "ObjectCreated:Put", "u..."} ▶ 2025-09-10T03:05:11.589Z New file uploaded: hello.txt (37 bytes) in bucket midtermmidpangshufinal ▶ 2025-09-10T03:05:11.589Z END RequestId: f8624282-5e11-4a4d-bd7c-9c360310ed0d ▶ 2025-09-10T03:05:11.589Z REPORT RequestId: f8624282-5e11-4a4d-bd7c-9c360310ed0d Duration: 2.15 ms Billed Duration: 101 ms Memory Size: 128 MB Max Memory Used: 36 MB Init Duration: 98.43 ms
------------	--

alligator-zoo-bg.png

Metrics	<ul style="list-style-type: none"> ▶ 2025-09-10T03:08:24.666Z START RequestId: cd681c3e-6c61-4f3d-9b49-f0d1124f65fb Version: \$LATEST ▶ 2025-09-10T03:08:24.666Z Event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "eu-north-1", "eventTime": "2025-09-10T03:08:24.139Z", "eventName": "ObjectCreated:Put", "u..."} ▶ 2025-09-10T03:08:24.666Z New file uploaded: alligator-zoo-bg.png (563551 bytes) in bucket midtermmidpangshufinal ▶ 2025-09-10T03:08:24.666Z END RequestId: cd681c3e-6c61-4f3d-9b49-f0d1124f65fb ▶ 2025-09-10T03:08:24.666Z REPORT RequestId: cd681c3e-6c61-4f3d-9b49-f0d1124f65fb Duration: 19.21 ms Billed Duration: 20 ms Memory Size: 128 MB Max Memory Used: 36 MB
---------	---

logo.svg

Insights	<ul style="list-style-type: none"> ▶ 2025-09-10T03:08:25.466Z START RequestId: 3d2d9620-1518-4d5d-a3ce-3ad7af99c763 Version: \$LATEST ▶ 2025-09-10T03:08:25.466Z Event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "eu-north-1", "eventTime": "2025-09-10T03:08:24.950Z", "eventName": "ObjectCreated:Put", "u..."} ▶ 2025-09-10T03:08:25.466Z New file uploaded: logo.svg (2632 bytes) in bucket midtermmidpangshufinal ▶ 2025-09-10T03:08:25.486Z END RequestId: 3d2d9620-1518-4d5d-a3ce-3ad7af99c763 ▶ 2025-09-10T03:08:25.486Z REPORT RequestId: 3d2d9620-1518-4d5d-a3ce-3ad7af99c763 Duration: 20.25 ms Billed Duration: 21 ms Memory Size: 128 MB Max Memory Used: 36 MB
----------	---