Lab 14: Triggering Lambda from S3 to Update a DynamoDB Table

Objective:

Create a serverless workflow where uploading a file to an S3 bucket triggers a Lambda function, which extracts metadata and stores it into a DynamoDB table.

Estimated Duration: 60 minutes

Prerequisites:

- AWS Free Tier account
- IAM user with permissions to access S3, Lambda, and DynamoDB

Part A: Create a DynamoDB Table

1. Go to **DynamoDB > Tables > Create Table**
2. Table name: `S3Uploads`
3. Partition key: `FileName` (String)
4. Leave other settings as default
5. Click **Create Table**

Part B: Create an S3 Bucket

1. Navigate to **S3** > **Create bucket**
2. Bucket name: `lambda-s3-dynamo-demo` (must be globally unique)
3. Region: e.g., `ap-south-1`
4. Disable "Block all public access"
5. Click **Create bucket**

Part C: Create a Lambda Function

1. Go to **Lambda** > **Create function**
2. Name: `S3ToDynamoDBLogger`
3. Runtime: **Python 3.10**
4. Execution role:
    - Select "Create new role with basic Lambda permissions"

After creation, go to the IAM role and attach the following policies:

- `AmazonS3ReadOnlyAccess`
- `AmazonDynamoDBFullAccess`

Part D: Add S3 Trigger to Lambda

1. Go to the Lambda function > **Configuration > Triggers**
2. Add trigger:
   - Source: `S3`
   - Bucket: `lambda-s3-dynamo-demo`
   - Event type: `PUT`
3. Click **Add**

Part E: Lambda Function Code

1. Open **Code** section of the Lambda function
2. Replace the default code with the following:

```
import json
import boto3
import time

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('S3Uploads')

def lambda_handler(event, context):
    records = event.get('Records', [])
    if not records:
        return {"statusCode": 400, "body": "No records found."}

    s3_info = records[0]['s3']
    bucket = s3_info['bucket']['name']
    file_key = s3_info['object']['key']
    size = s3_info['object'].get('size', 0)

    timestamp = int(time.time())

    # Insert metadata into DynamoDB
    table.put_item(
        Item={
            'FileName': file_key,
            'Bucket': bucket,
            'Size': size,
            'Timestamp': timestamp
        }
    )

    return {
        'statusCode': 200,
        'body': json.dumps(f"File '{file_key}' metadata inserted into 
DynamoDB.")
    }
```

3. Click **Deploy** to save and apply the changes

Part F: Test the Integration

1. Go to **S3** > **lambda-s3-dynamo-demo** > **Upload**

2. Upload any file (e.g., `sample.txt`)
3. Wait a few seconds
4. Go to **DynamoDB > Tables > S3Uploads > Explore Table Items**
5. You should see a new record:
    - FileName: `sample.txt`
    - Bucket: `lambda-s3-dynamo-demo`
    - Size: `...`
    - Timestamp: `...`

Part G: Cleanup (Optional)

- Delete Lambda function
- Delete DynamoDB table
- Delete S3 bucket

Student Assignment

- Extend Lambda to log file type or uploader IP (if available)
- Log the operation in CloudWatch
- Validate file types before inserting into DynamoDB