

AWS Lambda with S3 Trigger and DynamoDB Integration

Introduction

AWS Lambda is a **serverless compute** service that lets you run code without provisioning or managing servers. It automatically scales and runs your code in response to triggers such as changes in data, system state, or user actions.

In this hands-on session, we will:

- Create an **S3 bucket**.
- Create a **Lambda function** triggered on file upload.
- Write a Lambda function in **Python 3.10**.
- Store metadata in a **DynamoDB table**.
- Clean up all AWS resources afterward.

Technologies Used

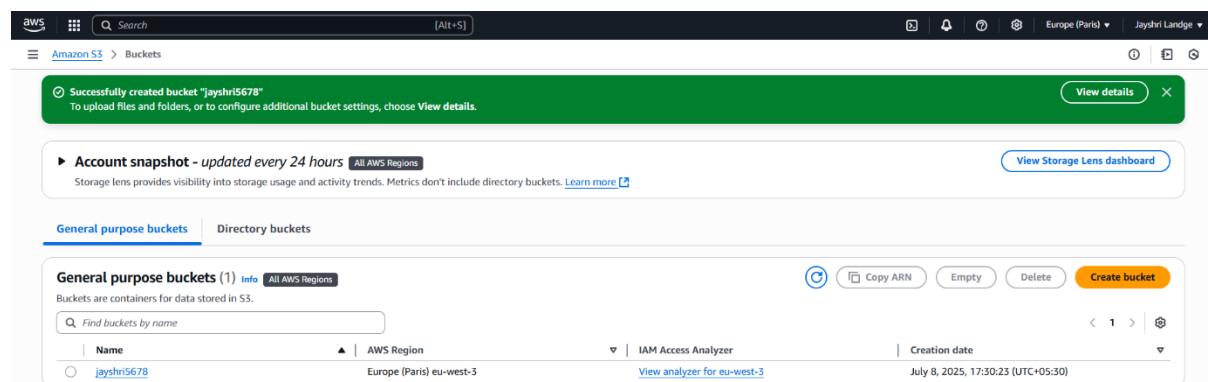
- AWS Lambda
- Amazon S3
- Amazon DynamoDB
- AWS IAM (Identity & Access Management)
- Python 3.10
- Boto3 (AWS SDK for Python)

Step-by-Step Implementation

Step 1: Create an IAM Role

◆ Go to **IAM Console** → **Roles** → **Create Role**

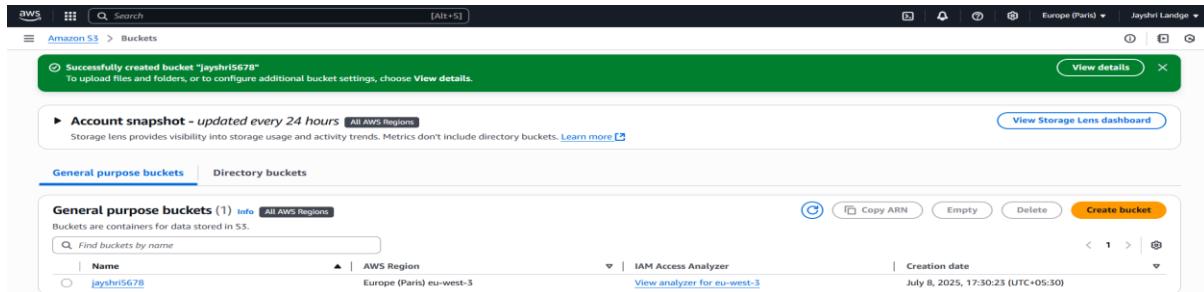
- **Trusted Entity:** AWS Service → Lambda
- **Permissions:** Attach `AmazonDynamoDBFullAccess` and `AmazonS3FullAccess`
- **Role Name:** `lambda-dynamodb-s3-role`



Screenshot 1: IAM Role creation screen

Step 2: Create an S3 Bucket

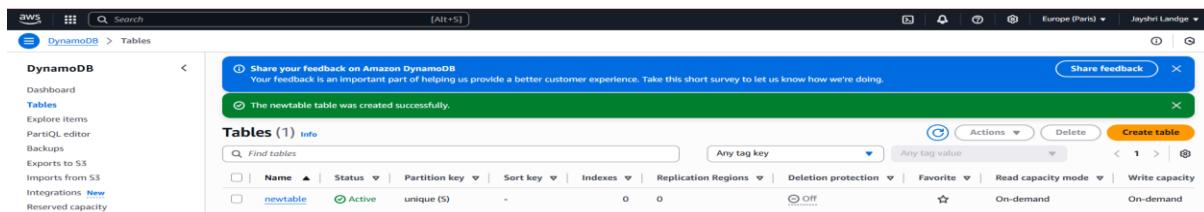
- ◆ Go to **S3 Console** → **Create bucket**
 - **Bucket Name:** my-upload-bucket-[your-unique-suffix]
 - Leave other settings as default
 - Disable public access



Screenshot 2: S3 bucket creation

Step 3: Create a DynamoDB Table

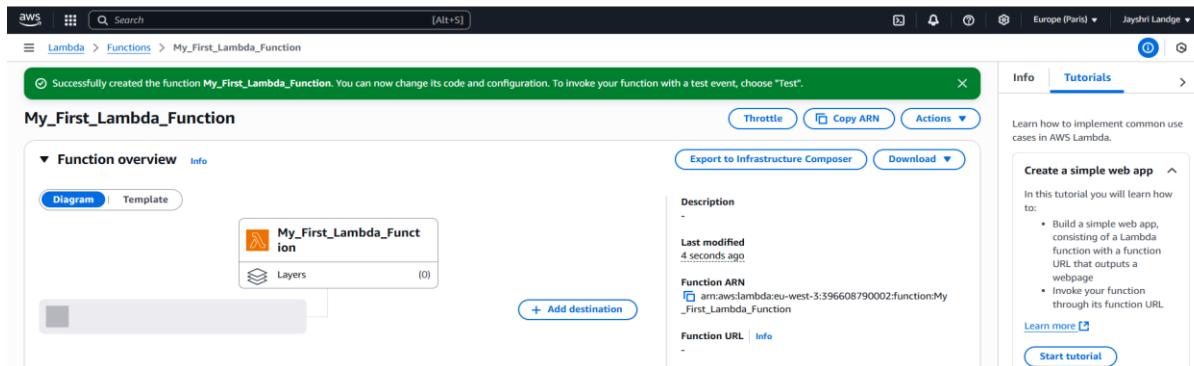
- ◆ Go to **DynamoDB Console** → **Create table**
 - **Table Name:** newtable
 - **Primary key:** unique (String)
 - Leave other settings default



Screenshot 3: DynamoDB table setup

Step 4: Create the Lambda Function

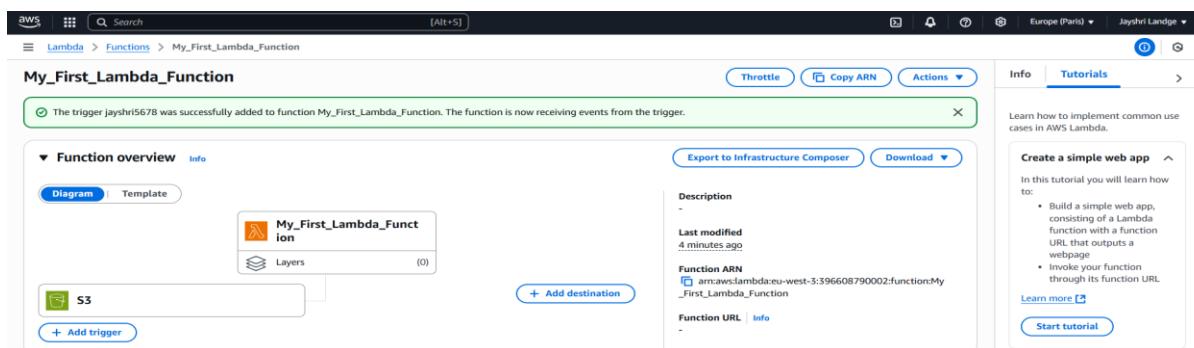
- ◆ Go to **AWS Lambda Console** → **Create Function**
 - **Author from scratch**
 - Function name: S3DynamoFunction
 - Runtime: Python 3.10
 - Execution role: Use existing role → lambda-dynamodb-s3-role



Screenshot 4: Lambda function configuration

Step 5: Add S3 Trigger to Lambda

- ◆ Inside Lambda → Triggers → Add Trigger:
- **Select Trigger:** S3
- **Bucket:** Choose the one created in Step 2
- **Event Type:** PUT (Object Created)
- **Enable trigger**



Screenshot 5: Adding S3 trigger to Lambda

Step 6: Add Python Code to Lambda

Paste the following code in the Lambda function editor:

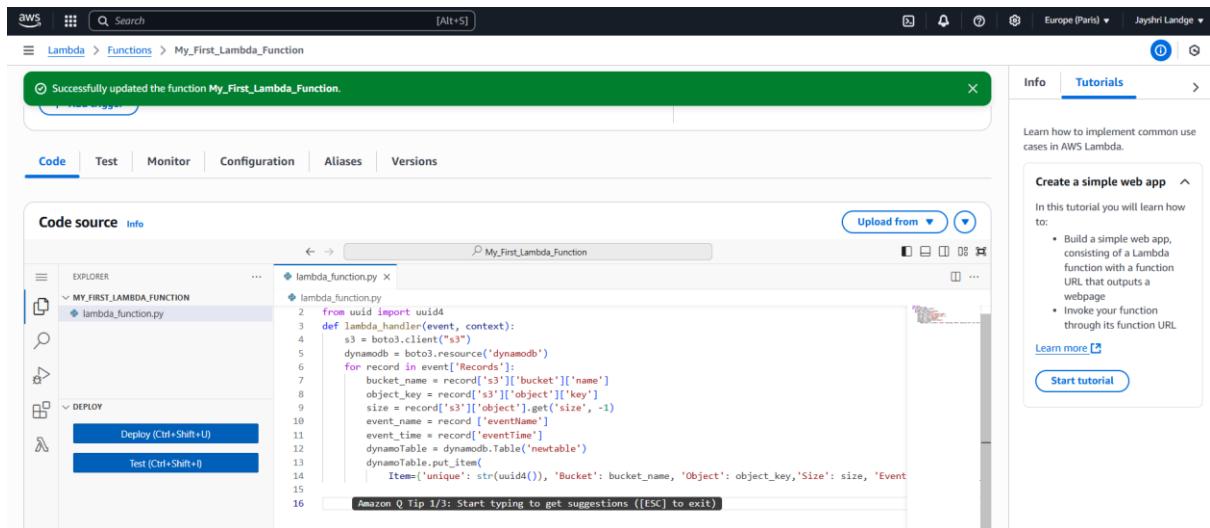
```
import boto3
from uuid import uuid4
def lambda_handler(event, context):
    s3 = boto3.client("s3")
    dynamodb = boto3.resource('dynamodb')

    for record in event['Records']:
        bucket_name = record['s3']['bucket']['name']
```

```

object_key = record['s3']['object']['key']
size = record['s3']['object'].get('size', -1)
event_name = record['eventName']
event_time = record['eventTime']
dynamoTable = dynamodb.Table('newtable')
dynamoTable.put_item(
    Item={
        'unique': str(uuid4()),
        'Bucket': bucket_name,
        'Object': object_key,
        'Size': size,
        'Event': event_name,
        'EventTime': event_time
    }
)

```



Screenshot 6: Lambda code editor Deploy code

Step 7: Test the Setup

- ◆ Go to **S3 bucket** → Upload any sample file (e.g., .txt, .jpg)
- ◆ After uploading:
 - Go to **DynamoDB Console**
 - Open the newtable
 - Click on **Explore Table Items**
 - You should see a new item inserted with metadata

The screenshot shows the AWS S3 'Upload: status' page. At the top, a green banner indicates 'Upload succeeded'. Below it, a message says 'For more information, see the Files and folders table.' A 'Close' button is in the top right. A note below states: 'After you navigate away from this page, the following information is no longer available.' The 'Summary' section shows 'Destination s3://jayshri5678' with 'Succeeded' (1 file, 14.2 KB) and 'Failed' (0 files, 0 B). The 'Files and folders' tab is selected, showing a table with one item: EC2.txt (text/plain, 14.2 KB, Succeeded).

Screenshot 7: Uploaded file in S3

The screenshot shows the AWS DynamoDB 'Explore items' interface for the 'newtable' table. On the left, a sidebar lists 'DynamoDB' sections like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Under 'Explore items', 'Tables (1)' is selected, showing 'newtable' highlighted. The main panel shows the 'newtable' table details. It includes a 'Scan or query items' section with 'Scan' selected, a 'Select attribute projection' dropdown set to 'All attributes', and a 'Completed - Items returned: 1' message. Below is a table titled 'Table: newtable - Items returned (1)' with one row of data: unique (String), Bucket (jayshri5678), Event (ObjectCreated:Put), EventTime (2025-07-08T12:20:58.960Z), Object (EC2.txt), and Size (14520).

Screenshot 8: Record entry in DynamoDB table

Conclusion

In this hands-on session, I successfully:

- Created a **serverless workflow** using AWS Lambda
- Triggered a Lambda function on **S3 file upload**
- Stored file metadata into **DynamoDB** using **Python + Boto3**

This practical illustrates how AWS services can be combined for **event-driven applications** using **minimal infrastructure setup**, ideal for scalable and cost-effective architectures.