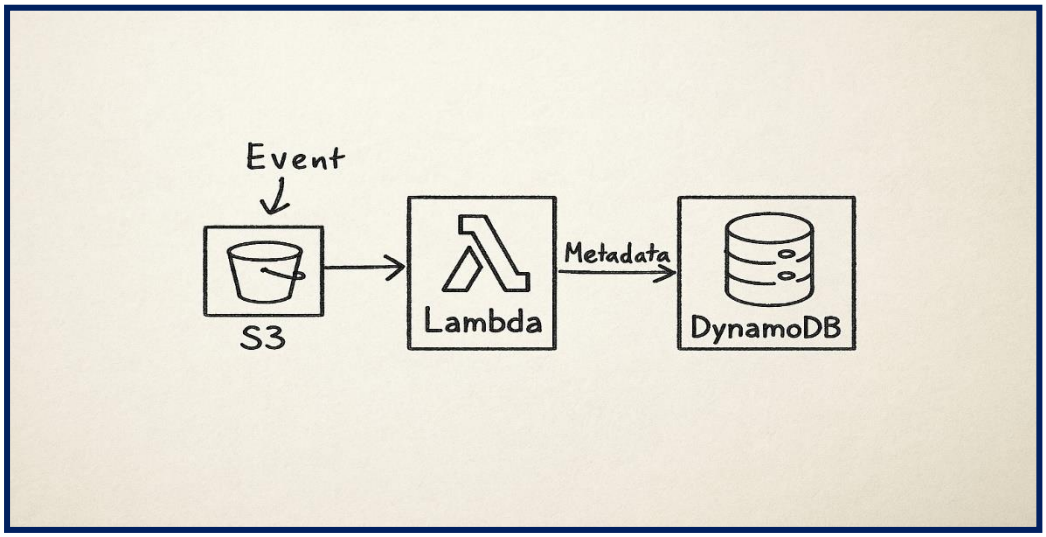
Automating Metadata Flow from S3 to DynamoDB with AWS Lambda.

This project automates the workflow of capturing metadata from newly uploaded objects in an S3 bucket. Upon each upload, an AWS Lambda function is triggered to extract the object's metadata and store it in a DynamoDB table—enabling seamless, serverless metadata tracking.

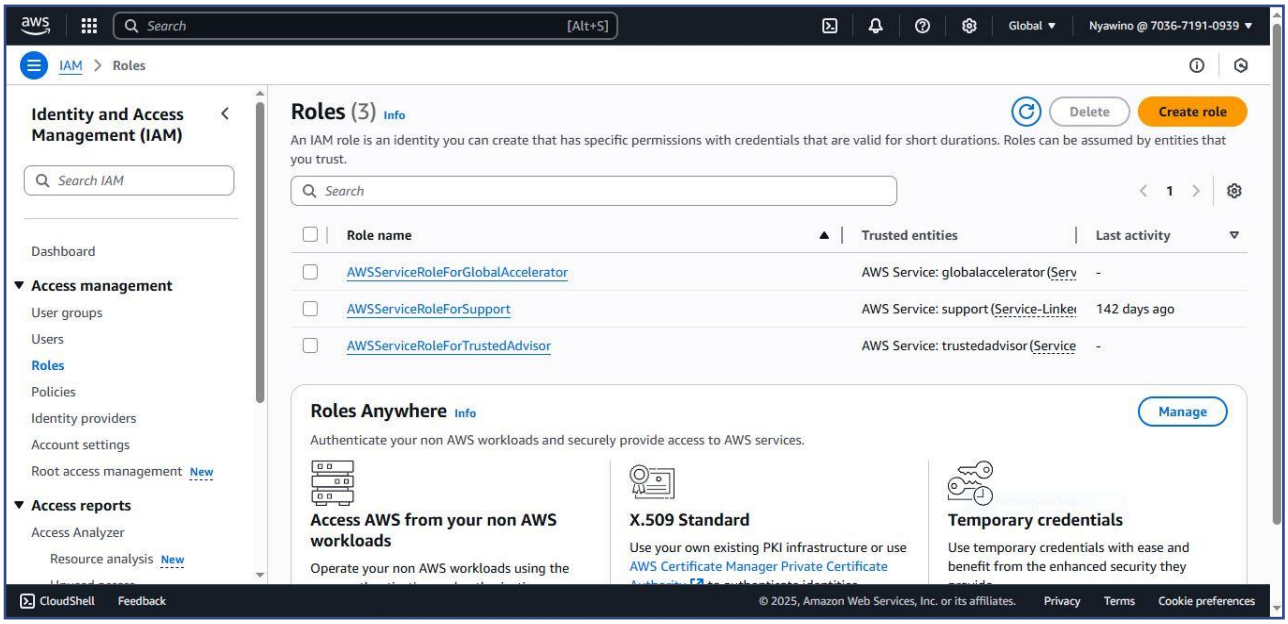


Architecture Overview

1. **Trigger**: S3 Event (e.g., `s3:ObjectCreated:*`)
2. **Action**: Lambda Function (Python)
3. **Target**: DynamoDB Table

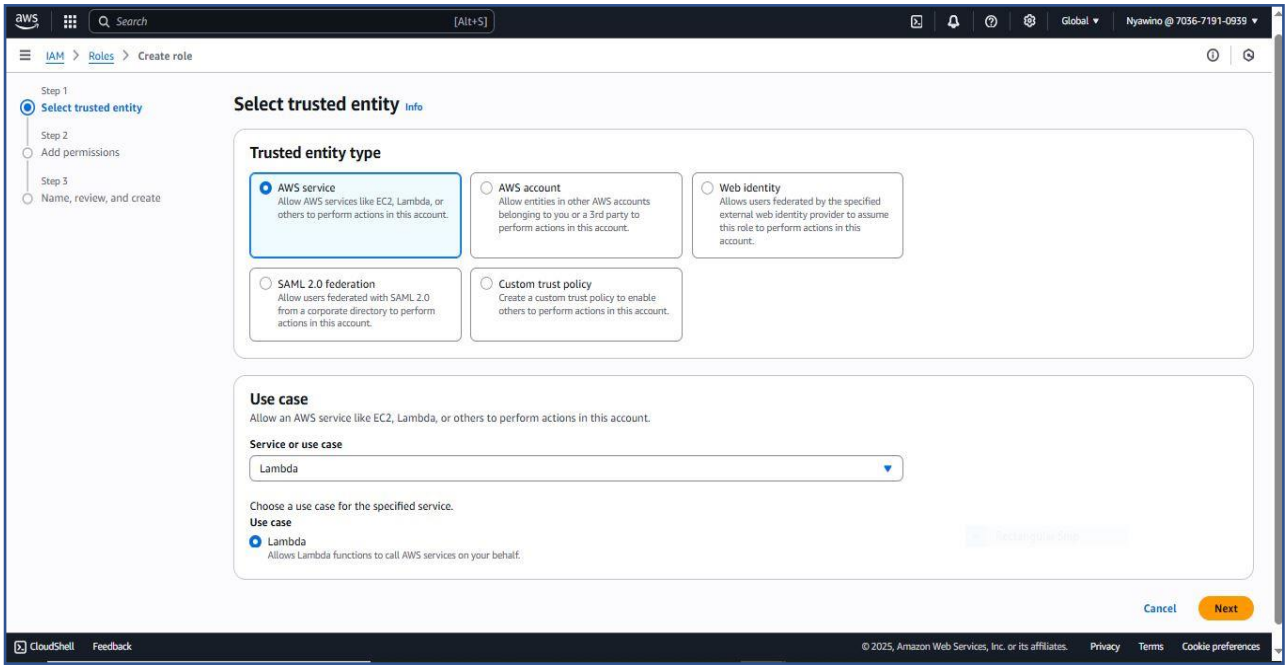
Step 1: Create an IAM Role for Lambda

- Go to the **IAM Console**
- Click **Roles** → **Create Role**

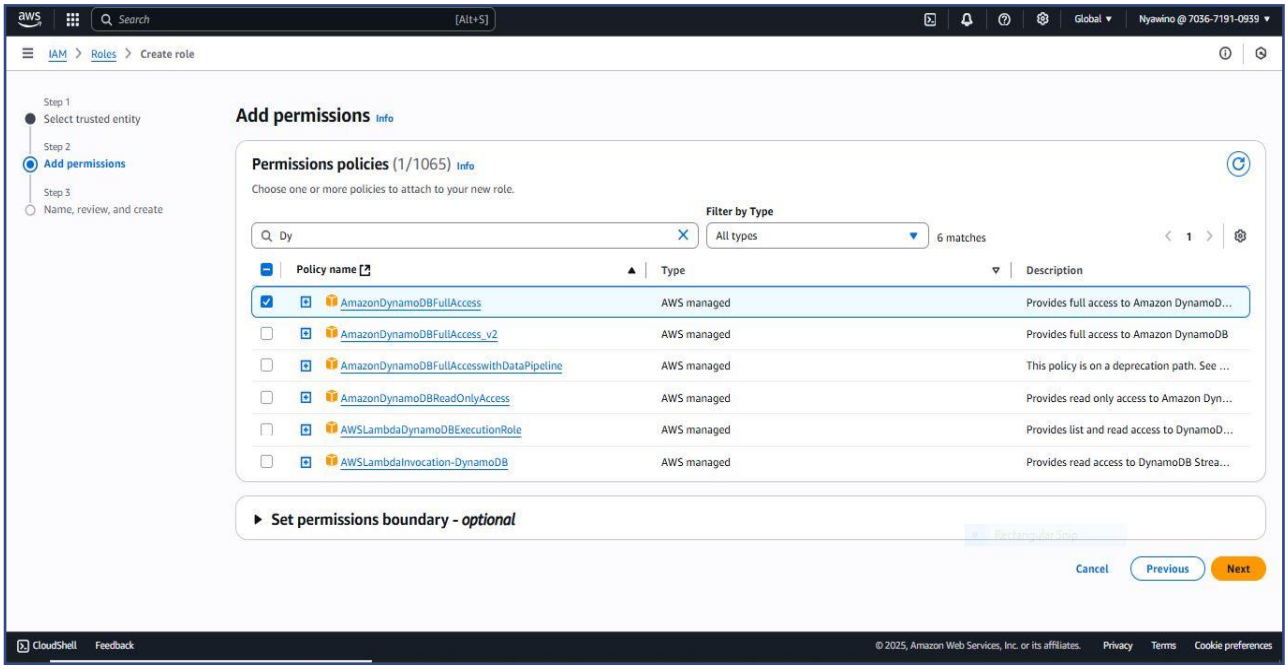


Documented By: Rhoda Ndege

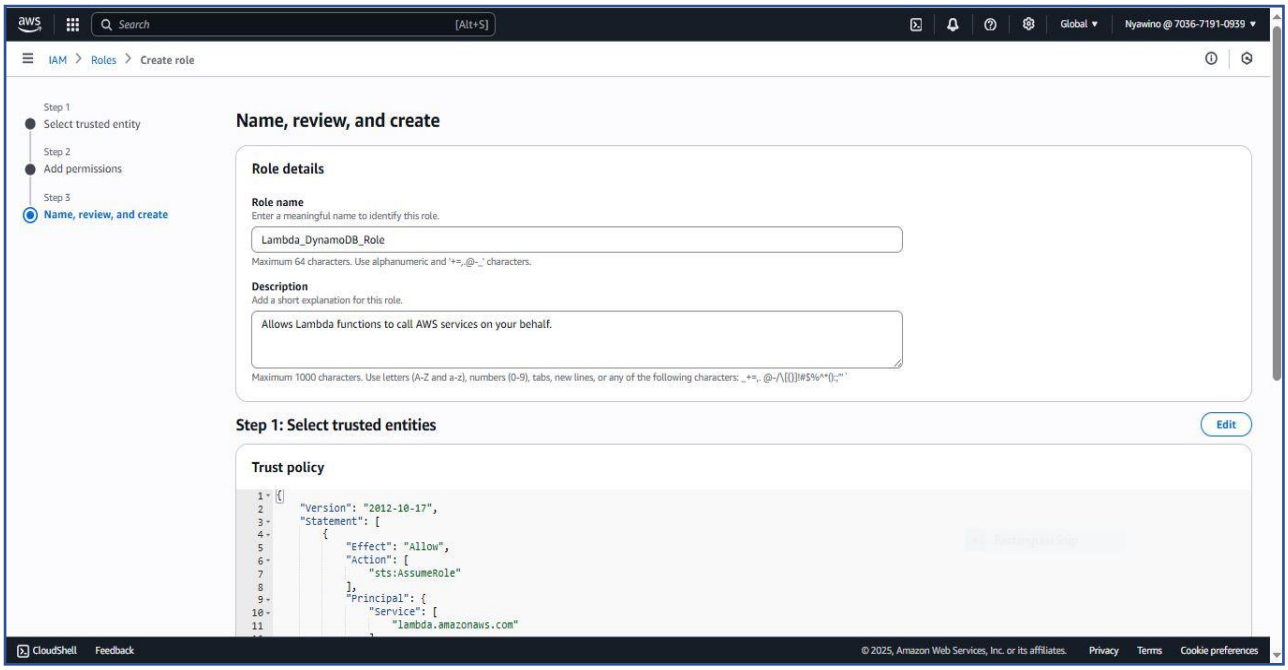
- Use Case ⟶ **Lambda**. Click **Next**



- **Add Permissions:**
  - ❖ *AmazonDynamoDBFullAccess*
  - ❖ *AmazonS3ReadOnlyAccess*
- Click **Next**



- **Role Name** for example: *Lambda_DyanmoDB_Role*
- Review the Role.
- Click **Create Role**

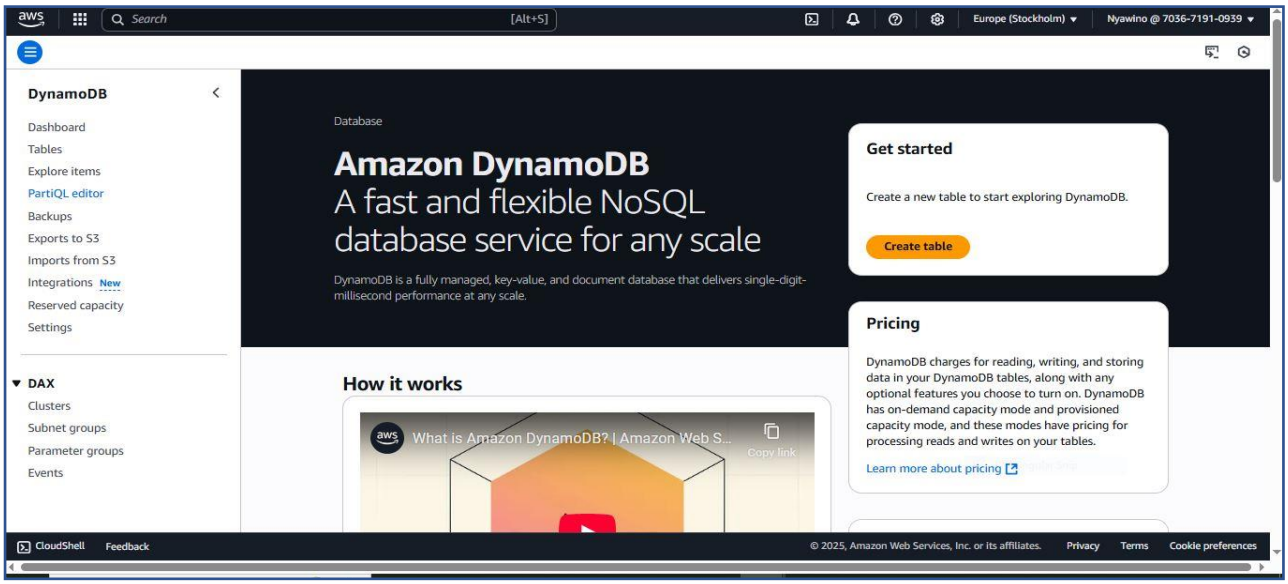


Step 2: Create an S3 Bucket.

- On the AWS Console: Search for **S3**
- Click **Create a Bucket.**
- Fill in:
  - ❖ Bucket Name: for example, *mys3bucket*
  - ❖ Object Ownership: ACL enabled
  - ❖ Disable Block Public Access
- Click **Create a Bucket.**

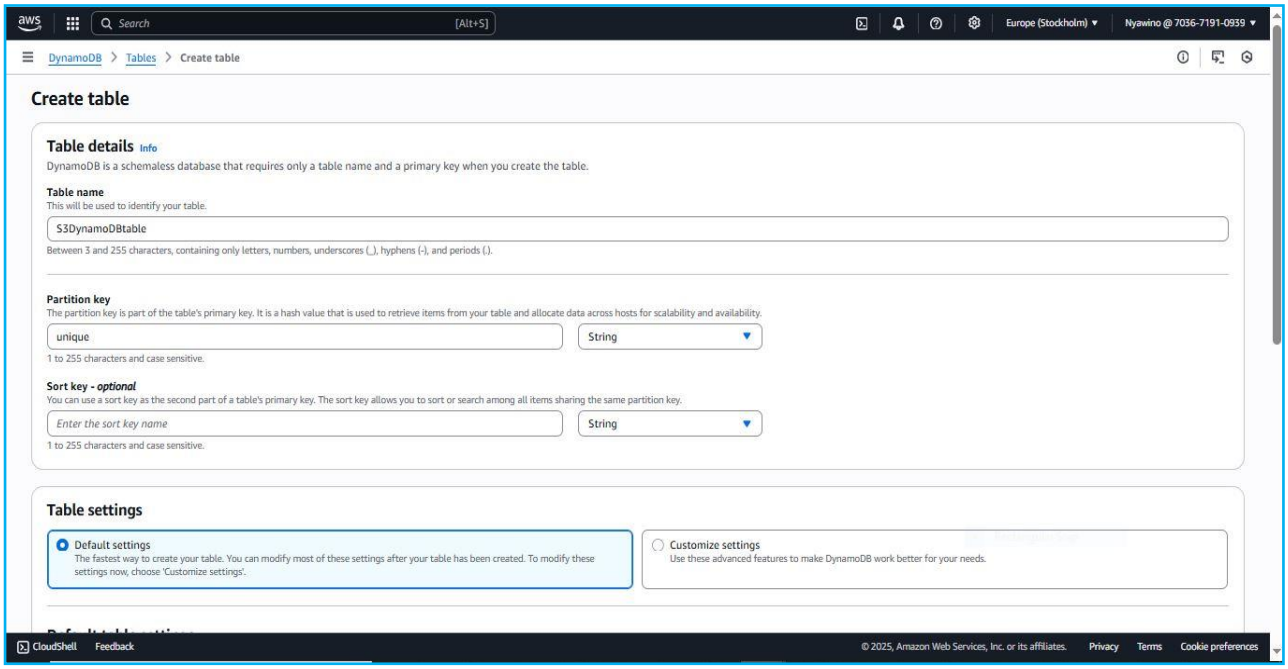
Step 3: Create a DynamoDB Table

- On the AWS Console, search for **AWS DynamoDB**
- Click **Create Table**



Documented By: Rhoda Ndege

- Create a table and use default settings
- Fill in:
    - ❖ Table Name: e.g., *Newdynamodbtable*
    - ❖ Partition Key: unique
    - ❖ Table Settings: Leave as Default

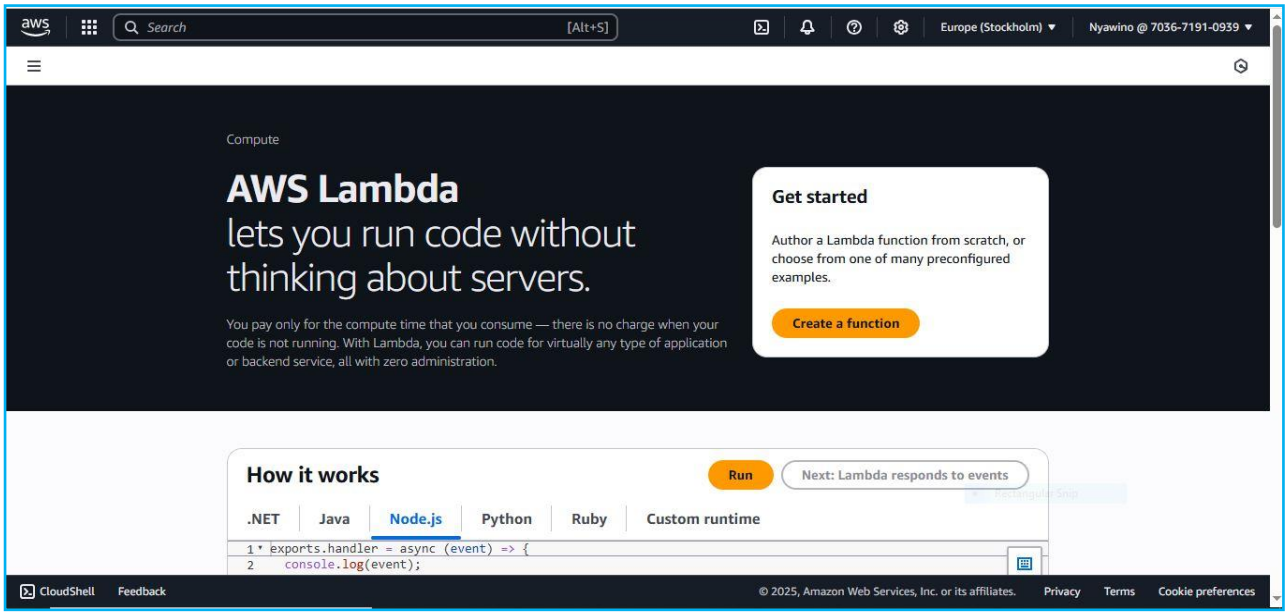


- Click **Create Table**

Step 4: Create a Lambda Function.

AWS Lambda is a serverless computing services that allows you to run code without provisioning or managing services.

- On the AWS Console, search for **AWS Lambda**
- Click **Create Function**



Documented By: Rhoda Ndege

- Choose:
  - ❖ Author from Scratch
  - ❖ Name: `S3ToDynamoFunction`
  - ❖ Runtime: Python 3.10/3.9/3.11)
  - ❖ Permissions: Use existing role: select `Lambda_DynamoDB_Role`

- Click **Create function**

Step 5: Add the Lambda Code.

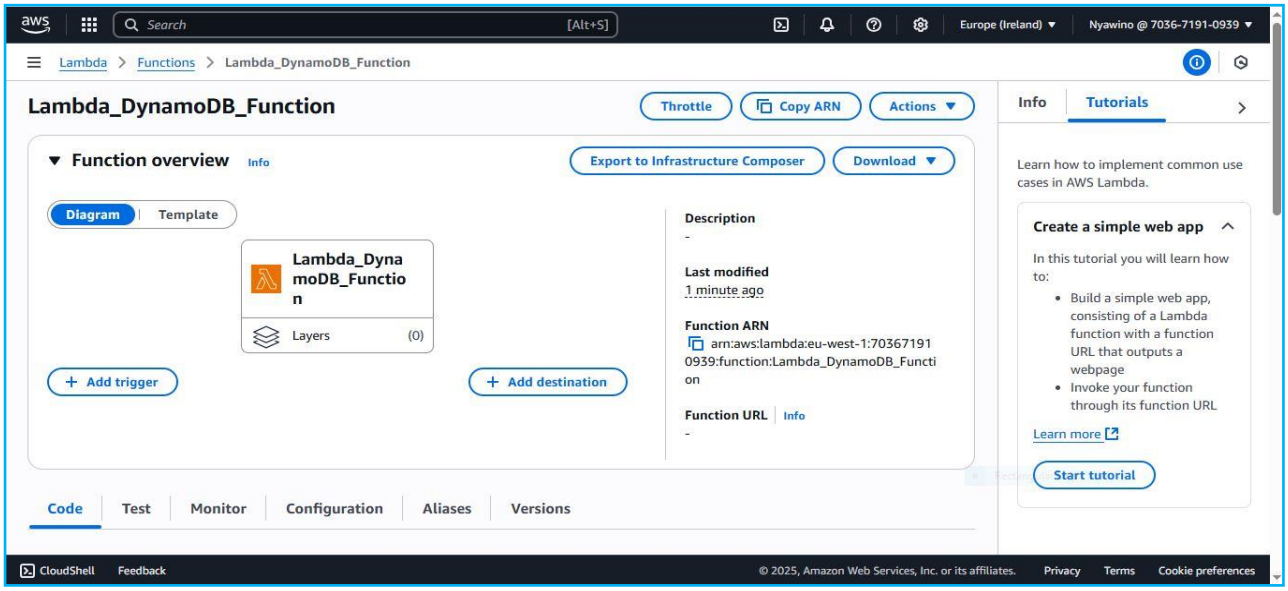
In the Lambda editor, **paste this code:**

```
import boto3
from uuid import uuid4

s3 = boto3.client('s3')
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('S3DynamoDBtable')

def lambda_handler(event, context):
    for record in event['Records']:
        bucket_name = record['s3']['bucket']['name']
        object_key = record['s3']['object']['key']
        event_name = record['eventName']
        event_time = record['eventTime']

        # Get object metadata
        try:
            meta = s3.head_object(Bucket=bucket_name, Key=object_key)
            size = meta['ContentLength']
        except Exception as e:
            print(f"Failed to get metadata: {e}")
            size = -1

        table.put_item(Item={
            'unique': str(uuid4()),
            'Bucket': bucket_name,
            'Object': object_key,
            'Size': size,
            'Event_name': event_name,
            'Event_time': event_time
        })
```

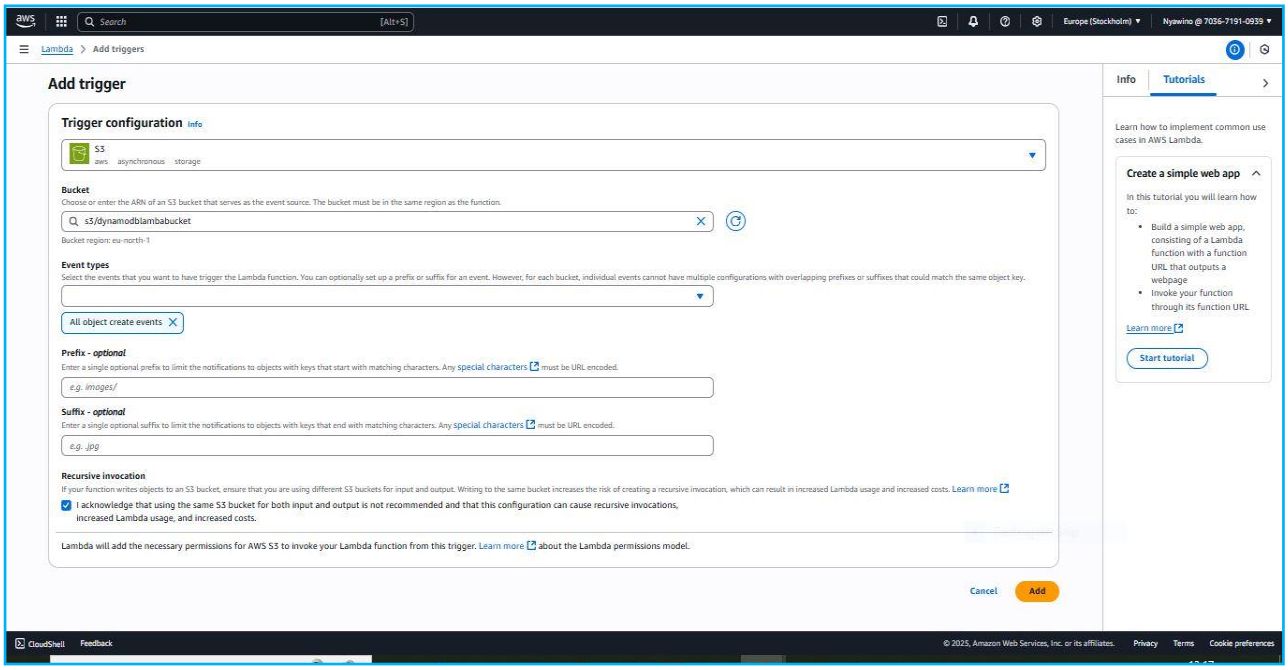
- Click **Deploy**

Step 6: Add an S3 Trigger

- On the same **Lambda Function page**, go to **"Configuration → Triggers"**
- Click **"Add trigger"**



- Choose:
  - **Trigger type**: S3
  - **Bucket**: your bucket name
  - **Event type**: PUT or All object create events
  - **Prefix/suffix**: leave blank unless filtering



Step 8: Test the Automation.

- Go to your S3 bucket
- Upload any test file (e.g., `test.txt`)
- Go to **CloudWatch Logs > /aws/lambda/S3ToDynamoFunction**
  - ❖ Confirm your function was triggered
  - ❖ Check for errors or success messages
- Go to **DynamoDB → Tables → S3DynamoDBtable → Explore Table Items**
- You should see a new row with the metadata.