[Challenge] AWS Lambda Exercise

In this challenge lab, you create an AWS Lambda function to count the number of words in a text file.

After completing this lab, you will be able to do the following:

- Create a Lambda function to count the number of words in a text file.
- Configure an Amazon Simple Storage Service (Amazon S3) bucket to invoke a Lambda function when a text file is uploaded to the S3 bucket.
- Create an Amazon Simple Notification Service (Amazon SNS) topic to report the word count in an email.

Your challenge

3. Create a Lambda function to count the number of words in a text file. The general steps are as follows:
    - Use the AWS Management Console to develop a Lambda function in Python and create the function's required resources.
    - Report the word count in an email by using an SNS topic. Optionally, also send the result in an SMS (text) message.
    - Format the response message as follows:

      ```
      The word count in the <textFileName> file is nnn.
      ```

      Replace *<textFileName>* with the name of the file.
    - Enter the following text as the email subject: `Word Count Result`
        - Automatically invoke the function when the text file is uploaded to an S3 bucket.
4. Test the function by uploading a few sample text files with different word counts to the S3 bucket.
5. Forward the email that one of your tests produces and a screenshot of your Lambda function to your instructor.

Hints

- Create all of your resources in the same AWS Region.
- You need an AWS Identity and Access Management (IAM) role for the Lambda function to access other AWS services. Because the lab policy does not permit the creation of an IAM role, use the **LambdaAccessRole** role. The LambdaAccessRole role provides the following permissions:
    - AWSLambdaBasicExecutionRole

      This is an AWS managed policy that provides write permissions to Amazon CloudWatch Logs.
    - AmazonSNSFullAccess

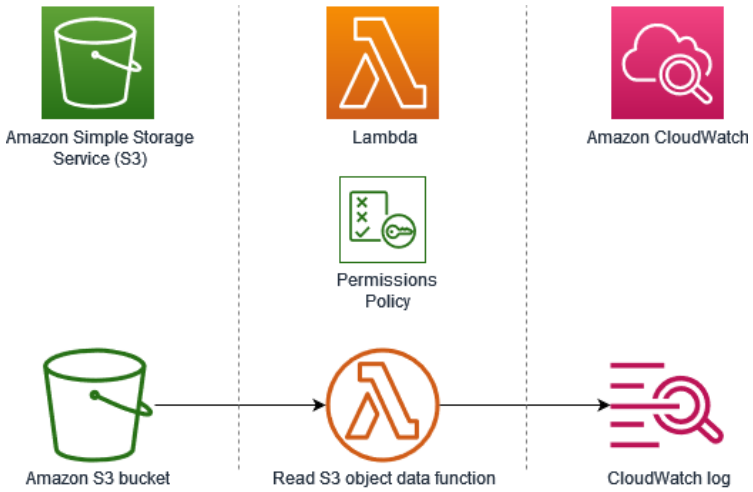
      This is an AWS managed policy that provides full access to Amazon SNS via the AWS Management Console.
    - AmazonS3FullAccess

      This is an AWS managed policy that provides full access to all buckets via the AWS Management Console.
    - CloudWatchFullAccess

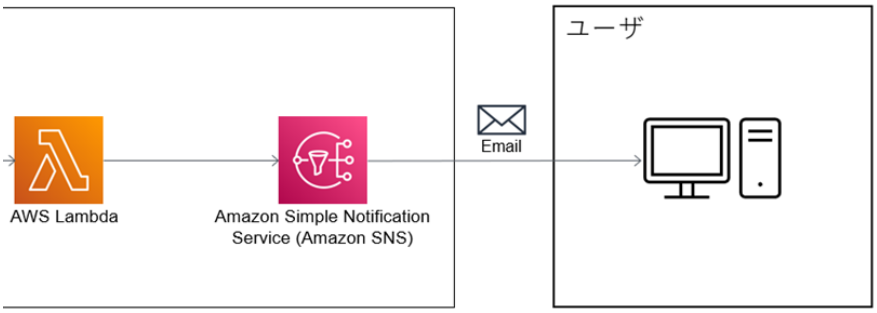
      This is an AWS managed policy that provides full access to Amazon CloudWatch.
- Refer to the following lab for additional guidance:
    - Working with AWS Lambda

Step 1: Configure an Amazon S3 bucket

สร้าง Lambda function และกำหนดค่า trigger สำหรับ S3, ทุกครั้งที่เพิ่มอ็อบเจกต์ลงใน S3 ฟังก์ชันคุณจะทำงานและส่งผลลัพธ์ประเภทของอ็อบเจกต์นั้นไปยัง CloudWatch Logs

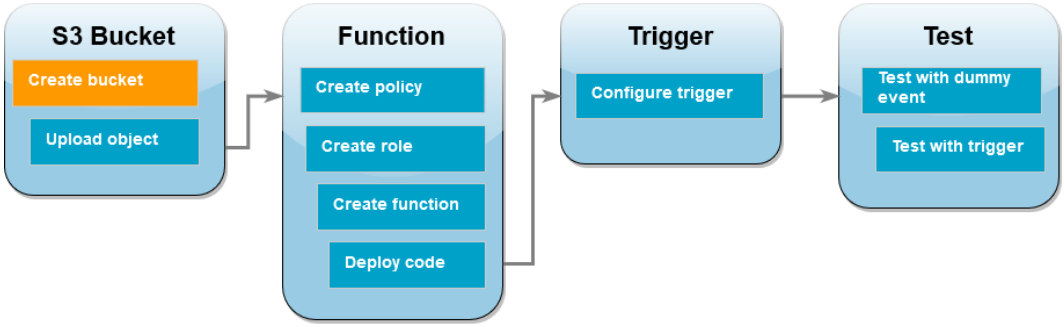


สามารถ Lambda function ร่วมกับ Amazon S3 trigger เพื่อทำงานประมวลผลไฟล์หลากหลายประเภทได้ เช่น สามารถใช้ Lambda function เพื่อสร้างไฟล์ภาพขนาดย่อ (Thumbnail) ทุกครั้งที่มีการอัปโหลดไฟล์ภาพไปยัง S3 หรือแปลงไฟล์เอกสารที่ถูกอัปโหลดให้อยู่ในรูปแบบอื่น



Lambda function ทำหน้าที่ประมวลผลข้อความก่อนส่งต่อข้อความไปยัง SNS Topic เพื่อให้ SNS กระจายข้อมูลไปยังผู้รับ (Subscribers) ตามที่กำหนดไว้

Step 1.1: Create an S3 bucket



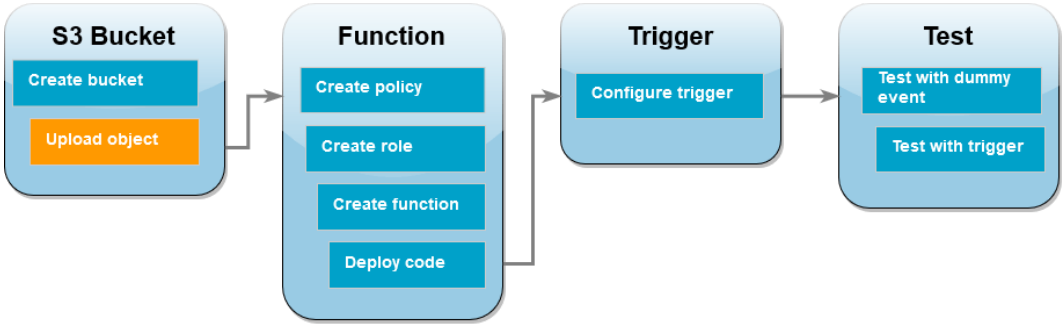
1) S3 > Create bucket
2) General configuration กำหนดค่าดังนี้

   Bucket name: lab.wordcount

Bucket name ต้องเป็น globally unique name โดยที่ S3 Bucket naming rules กำหนดไว้ว่า Bucket names ประกอบด้วยเงื่อนไขต่อไปนี้เท่านั้น: ตัวพิมพ์เล็ก ตัวเลข จุด (.) และขีด (-)

3) ตัวเลือกอื่นๆ กำหนดให้เป็นค่าเริ่มต้นตามเดิม

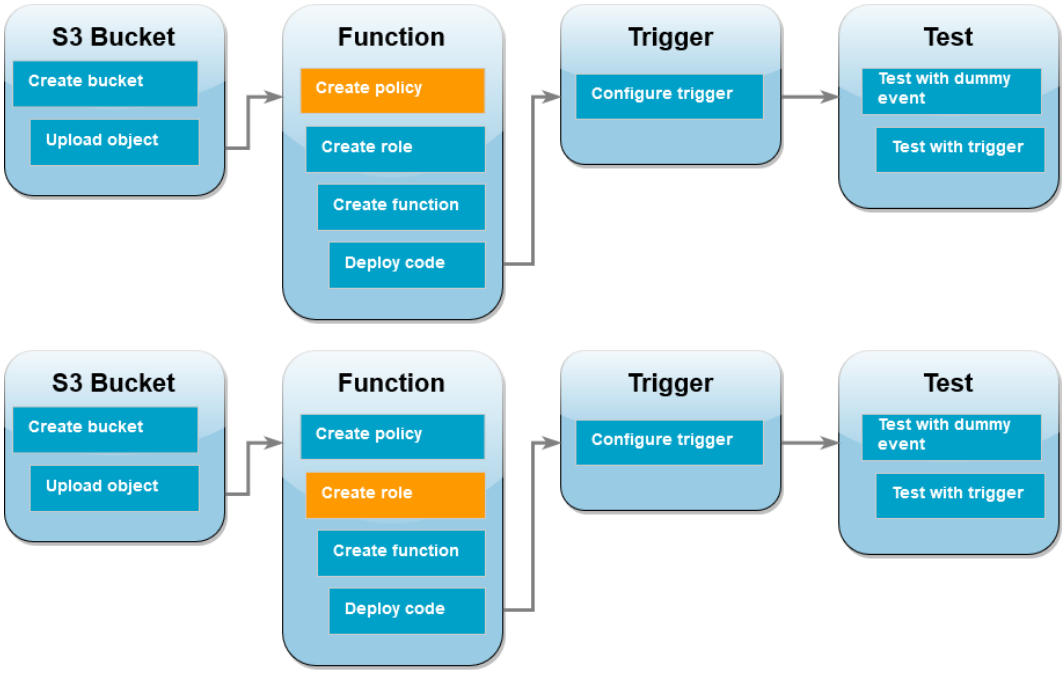
Step 1.2: Upload a test object to your bucket



ขั้นตอนหลังจากนี้ จะต้องทำการทดสอบ Lambda function เพื่อยืนยันว่าโค้ดของฟังก์ชันทำงานอย่างถูกต้อง, S3 จึงจำเป็นต้องมีอ็อบเจกต์สำหรับการทดสอบ อ็อบเจกต์นี้สามารถเป็นไฟล์ใดก็ได้ เช่น jpg, txt, docx, pdf แต่ในแล็บแนะนำเป็นไฟล์ txt เนื่องจากโค้ดที่เขียนขึ้นมาสามารถนับจำนวนคำใน txt ได้อย่างถูกต้อง แต่ในไฟล์ docx นับได้ไม่ถูกต้อง ซึ่งต้องหาวิธีแก้ต่อไปในภายหลัง

4) Buckets > Upload > Add files > Open > Upload

Step 2: Observing the IAM role settings



สำหรับแล็บนี้ ได้จัดเตรียม IAM role ไว้ให้แล้ว เนื่องจากนโยบายของแล็บไม่อนุญาตให้สร้าง IAM role

Case สร้าง custom permissions policy: สร้าง policy > สร้าง role เพื่อ attach policy นั้นๆ เข้าไปใน service (trusted entity) ที่ต้องการ

Case ใช้ default policy ผ่าน console: สร้าง role เพื่อกำหนด service ที่ต้องการเป็น trusted entity เท่านั้น > เลือก policy

5) **IAM role created by AWS:** Services > Security, Identity, & Compliance > IAM > Roles > LambdaAccessRole
6) เลือก Trust Relationships และจะเห็นว่า lambda.amazonaws.com อยู่ในรายการของ trusted entity ซึ่งหมายความว่า Lambda service สามารถใช้ IAM role นี้ได้

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

การกำหนด Role ให้ Lambda service สามารถใช้ได้ คลิกที่ตัว code > Edit trust policy > AssumeRole

7) **Default IAM Policies by AWS:** เลือก Permissions และสังเกตสิทธิ์ที่ได้รับใน role นี้:

- AWSLambdaBasicExecutionRole ให้สิทธิ์ในการเขียนลงใน CloudWatch Logs (เหมือนกับ AWSLambdaBasicRunRole)
- AmazonSNSFullAccess ให้สิทธิ์เต็มรูปแบบในการเข้าถึงทรัพยากร SNS
- AmazonS3FullAccess ให้สิทธิ์เต็มรูปแบบในการเข้าถึงทรัพยากร S3
- CloudWatchFullAccess ให้สิทธิ์เต็มรูปแบบในการเข้าถึงทรัพยากร CloudWatch
- code การให้สิทธิ์ในแต่ละ role อ่านเพิ่มเติมได้ใน Additional file

ฟังก์ชัน Lambda ที่จะถูกสร้างขึ้นในภายหลัง จะใช้ role: LambdaAccessRole

Step 3: Configuring notifications

สร้าง SNS (Simple Notification Service) topic และ subscribe อีเมลกับ topic นั้น

Step 3.1: Creating an SNS topic

สร้าง SNS topic สำหรับเผยแพร่รายงานการวิเคราะห์ยอดขาย และ subscribe อีเมลกับ topic นั้น และ topic ดังกล่าวมีหน้าที่ส่งข้อความทั้งหมดที่ได้รับไปยังสมาชิกทุกคน เราจะใช้ Amazon SNS console เพื่อดำเนินงานนี้

8) Services > Application Integration > Simple Notification Service
9) Topics > Create topic
10) กำหนดค่าดังนี้

- Type: Standard
- Name: ReadS3ObjectTopic
- Display name: RSOTopic

11) Create topic
12) คัดลอกและวางค่า ARN ลงใน text editor document

จะต้องระบุค่า ARN นี้ เมื่อทำการกำหนดค่า Lambda function ในลำดับต่อไป

Step 3.2: Subscribing to the SNS topic

13) Create subscription และกำหนดค่าดังนี้

Protocol: Email

Endpoint: email address

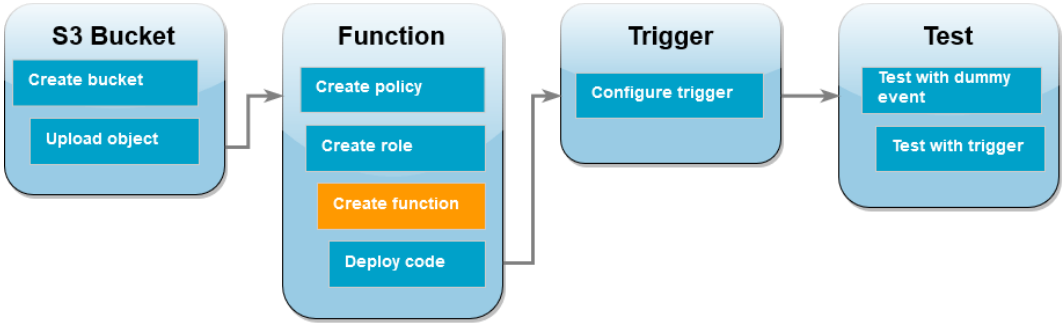
14) Create subscription > subscription จะถูกสร้างขึ้นและมีสถานะเป็น Pending confirmation
15) ตรวจสอบกล่องจดหมายของที่อยู่อีเมลที่ subscribe ไว้

จะเห็นอีเมลจาก RSOTopic มีหัวข้อ "AWS Notification - Subscription Confirmation."

16) เปิดอีเมลและเลือก Confirm subscription

แท็บเบราว์เซอร์ใหม่จะเปิดขึ้นมาและแสดงหน้าเว็บพร้อมข้อความ "Subscription confirmed!"

Step 4: Create the ReadS3Object Lambda function

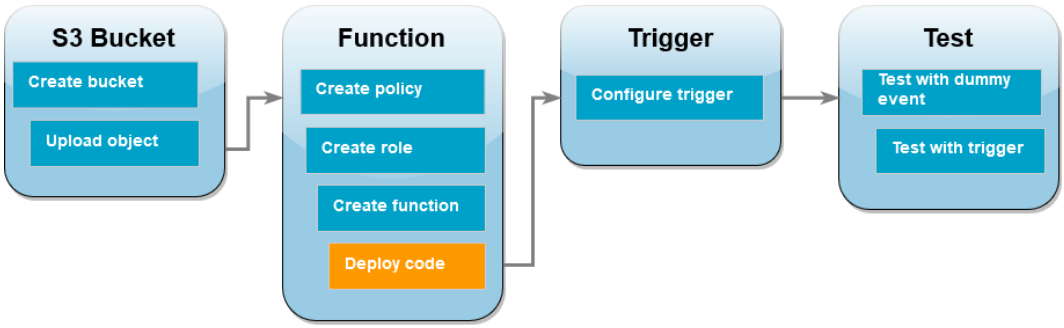


17) Lambda > Functions > Create function และกำหนดค่าดังนี้

- เลือก Author from scratch
- Function name: ReadS3Object
- Runtime: Python 3.12
- Change default execution role กำหนดการตั้งค่าดังนี้
    - Execution role: Use an existing role.
    - Existing role: LambdaAccessRole

18) Create function

Step 5: Deploy the function code



Lambda function จะดึงข้อมูล key name ของอ็อบเจกต์ที่อัพโหลดไว้ และ bucket name จาก event parameter ที่ได้รับจาก S3

โค้ดนี้เป็น Lambda function เขียนด้วยภาษา Python ที่มีการใช้งาน AWS SDK for Python (Boto3) เพื่อทำงานกับบริการของ Amazon, ทำงานโดยการนับจำนวนคำในไฟล์ข้อความที่ถูกอัปโหลดไปยัง S3 bucket และส่งรายงานจำนวนคำไปยัง SNS topic ผ่านทางอีเมล ดังนี้

```
import boto3
def lambda_handler(event, context):
    # Get the S3 bucket and file key from the event
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']
    # Download the file from S3
    s3 = boto3.client('s3')
    response = s3.get_object(Bucket=bucket, Key=key)
    text = response['Body'].read()
    # Count the number of words
    word_count = len(text.split())
    # Create an SNS client and publish the word count
    sns = boto3.client('sns')
    sns.publish(
```

```
        TopicArn='YOUR_SNS_TOPIC_ARN',
        Message=f'The word count for in the {key} is {word_count}.',
        Subject='Word Count Result'
    )
    return {
        'statusCode': 200,
        'body': f'Word count: {word_count}'
    }
```

ขั้นตอนการทำงานของโค้ดนี้

1. ดึงข้อมูลชื่อ S3 bucket และ key ของไฟล์ที่ถูกอัปโหลดจาก event object
2. ดาวน์โหลดเนื้อหาไฟล์จาก S3 bucket ดังกล่าว
3. นับจำนวนคำในไฟล์
4. ส่งข้อความรายงานจำนวนคำผ่าน SNS topic ตาม TopicArn ที่ระบุไว้

อธิบายโค้ดเพิ่มเติมใน Additional file

19) Lambda > Functions > ReadS3Object

20) Runtime settings > Edit

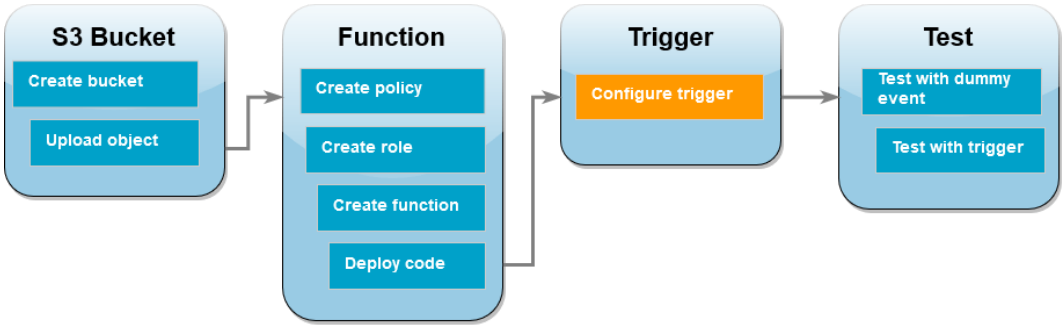
21) Handler: ReadS3Object.lambda_handler

22) Save

23) Code source > New File > Save As ReadS3Object.py

24) Deploy

Step 6: Adding a trigger to the ReadS3Object Lambda function



หลังจากทำการ deploy function code แล้ว ขั้นตอนนี้จะสร้าง Amazon S3 trigger เพื่อเรียกใช้งานฟังก์ชันดังกล่าว

25) Function overview > Add trigger

26) กำหนดค่า ดังนี้

Trigger configuration: S3

Bucket: lab.wordcount

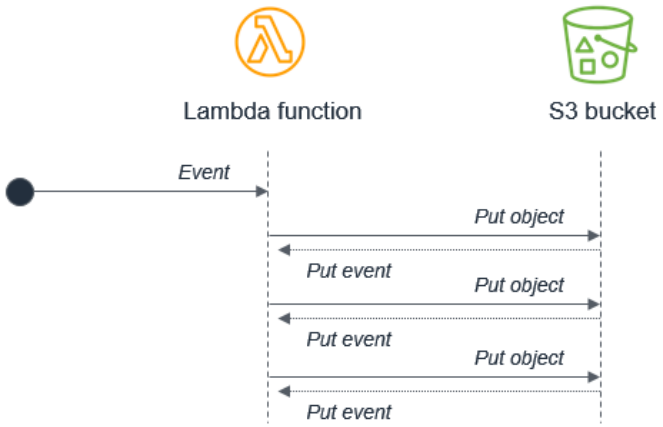
Event types: All object create events

27) Recursive invocation > check box

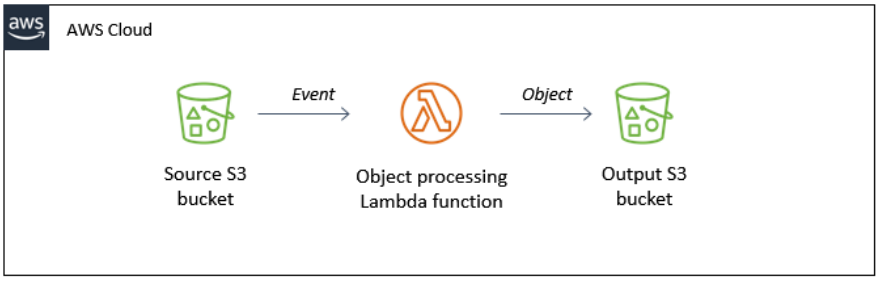
เพื่อรับทราบว่าการใช้ Amazon S3 bucket เดียวกันสำหรับทั้งการรับข้อมูลเข้า (input) และส่งข้อมูลออก (output) เป็นวิธีที่ไม่แนะนำ

หมายเหตุ:

- วิธีดังกล่าวอาจก่อให้เกิดปัญหาการเรียกใช้ฟังก์ชันแบบวนซ้ำ (recursive invocation) หรือเหตุการณ์ที่ Lambda function ทริกเกอร์ตัวเองซ้ำๆ อธิบายโดยละเอียด คือ Lambda function เขียน object ส่งกลับไปยัง source bucket ทำให้เกิดการเรียกใช้ฟังก์ชันเดิมอีกครั้งผ่านการ put event และเขียน object ที่สองลงใน bucket เดิม ส่งผลให้เกิดการเรียกใช้ Lambda function ซ้ำๆ วนไปเรื่อยๆ นำไปสู่การใช้ทรัพยากรที่มากเกินไปและอาจก่อให้เกิดปัญหาค่าใช้จ่ายที่สูงขึ้น
- ในแล็บนี้จะไม่มีการเขียนหรือส่งข้อมูลกลับไปยัง source bucket แต่อย่างใด

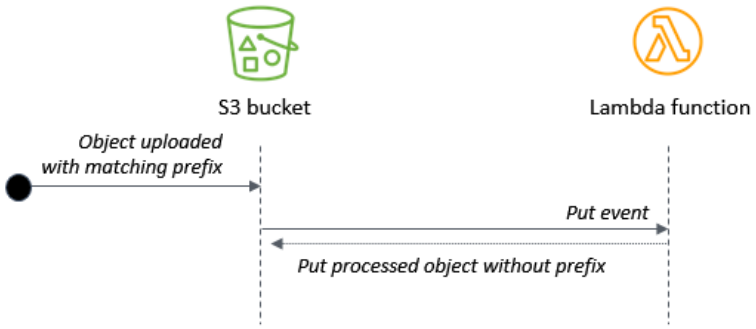


Best practice เพื่อหลีกเลี่ยงปัญหานี้ คือ การใช้ S3 bucket สองถังแยกกัน เนื่องจากการเขียน output object ไปยังถังที่สองเป็นวิธีที่ง่ายและปลอดภัยที่สุดในการป้องกันปัญหานี้ แต่หากจำเป็นต้องใช้ bucket เดียวกัน ก็ต้องมีการดำเนินการควบคุมที่เหมาะสม รูปแบบ two-bucket เป็นสถาปัตยกรรมที่แนะนำสำหรับ object processing workloads ใน S3

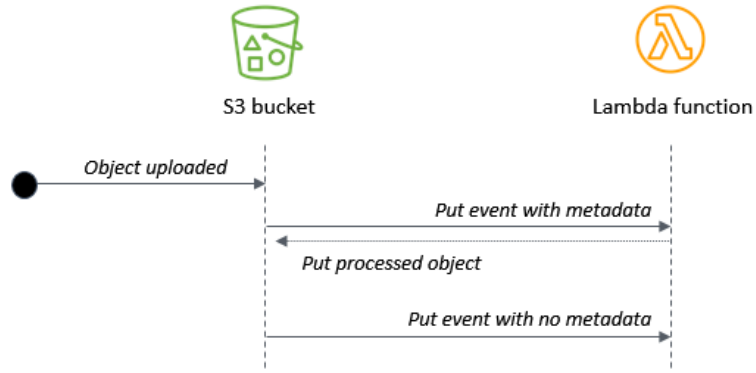


หากจำเป็นต้องเขียนวัตถุที่ประมวลผลแล้วกลับไปยัง source bucket, นี่เป็น 3 รูปแบบของสถาปัตยกรรมทางเลือกเพื่อลดความเสี่ยงในการเรียกใช้ฟังก์ชันแบบวนซ้ำได้

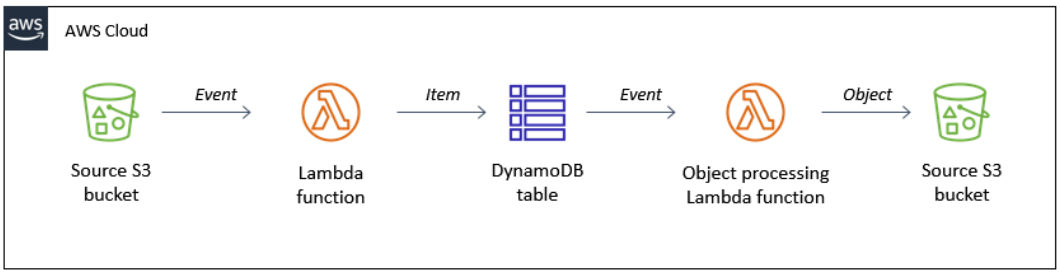
1. Using a prefix or suffix in the S3 event notification



2. Using object metadata to identify the original S3 object



3. Using an Amazon DynamoDB table to filter duplicate events



สามารถอ่านเพิ่มเติมได้ที่: https://aws.amazon.com/blogs/compute/avoiding-recursive-invocation-with-amazon-s3-and-aws-lambda/

28) Add

Step 7: Adding a destination to the ReadS3Object Lambda function

29) Function overview > Add destination

30) Destination configuration กำหนดค่าดังนี้

Source: Asynchronous invocation

- Lambda function จะทำงานโดยไม่ต้องรอผลลัพธ์จากปลายทาง (destination) หรือ Lambda function จะทำงานเสร็จสิ้นทันทีโดยไม่ต้องรอให้ SNS Topic ประมวลผลข้อความ

Condition: On success

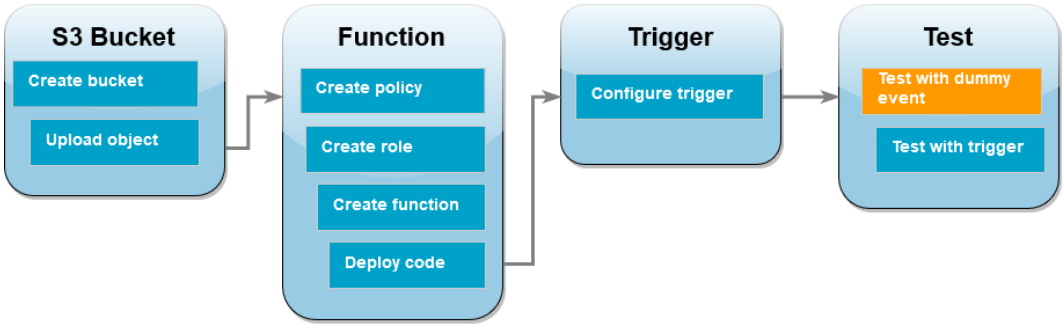
- Lambda function จะส่งข้อความไปยัง SNS Topic เฉพาะเมื่อ function ทำงานสำเร็จ

  Destination type: SNS topic

  Destination: RSOTopic

31) Save

Step 8: Test Lambda function with a dummy event



เริ่มต้นการทดสอบโดยการส่ง dummy S3 event ไปยังฟังก์ชัน เพื่อยืนยันว่าฟังก์ชันทำงานได้อย่างถูกต้อง

32) Test และกำหนดค่าดังนี้

Test event action: Create new event

Even name: MyTestEvent

Template: S3 Put

Event JSON และกำหนดค่าดังนี้

awsRegion: us-west-2 (ภูมิภาคที่ S3 bucket ถูกสร้างขึ้น)

name: lab.wordcount

arn: arn ของ lab.wordcount

key: ชื่อไฟล์ที่ใช้ในการทดสอบ เช่น Script.txt

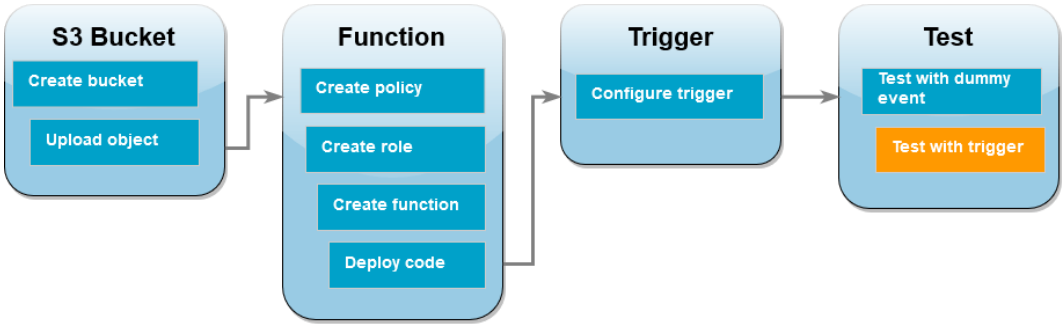
33) Save

34) Test

เคล็ดลับ: หากได้รับข้อผิดพลาดเกี่ยวกับ timeout error ให้เลือกปุ่ม test อีกครั้ง บางครั้งเมื่อรันฟังก์ชันครั้งแรก มันจะใช้เวลานานกว่าปกติในการเตรียมความพร้อม และค่า default timeout ของ Lambda (3 วินาที) จะเกินไปกว่านี้เล็กน้อย โดยทั่วไปจะสามารถรันฟังก์ชันอีกครั้งได้ แล้วข้อผิดพลาดจะหายไป หรือในทางเลือกอื่นคือเพิ่มค่า timeout โดยทำตามขั้นตอนเหล่านี้:

Configuration > General configuration > Edit > ปรับเปลี่ยน Timeout ตามความเหมาะสม > Save

Step 9: Test the Lambda function with the Amazon S3 trigger



เพื่อทดสอบฟังก์ชันด้วยการใช้ trigger ที่กำหนดค่าไว้ จะต้องอัปโหลดอ็อบเจกต์ไปยัง S3 bucket เมื่อตรวจสอบแล้วว่า Lambda function ถูกเรียกใช้งานอย่างถูกต้อง จากนั้นจึงใช้ CloudWatch Logs เพื่อดูผลลัพธ์ที่ได้จากฟังก์ชันดังกล่าว

การตรวจสอบการทำงานของฟังก์ชันโดยใช้ CloudWatch Logs

35) CloudWatch > Logs > Log groups > /aws/lambda/ReadS3Object

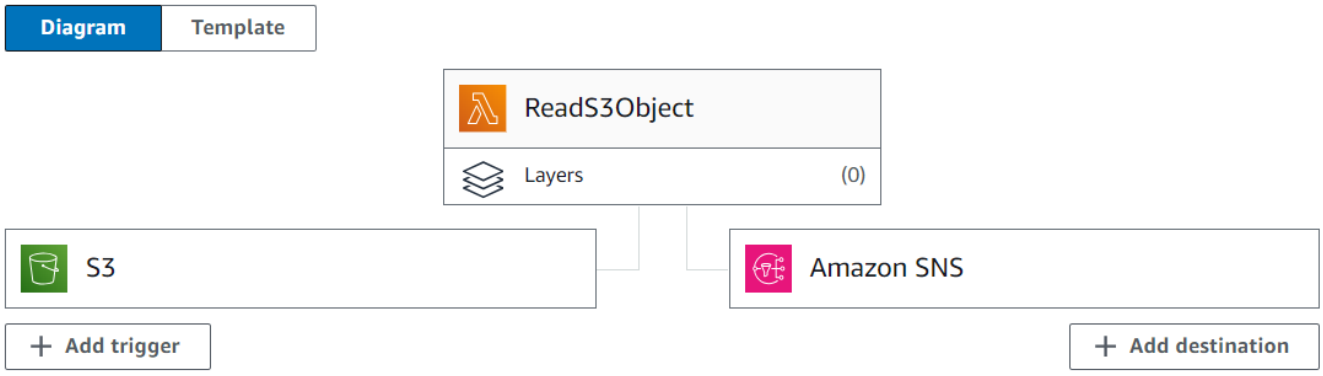
36) Log streams > เลือก Log events ที่ต้องการเข้าไปดู

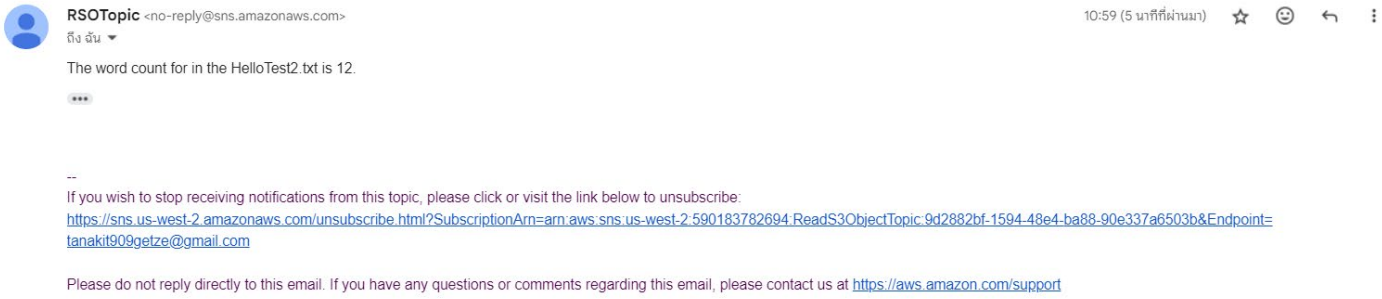
37) หากฟังก์ชันถูกเรียกใช้งานอย่างถูกต้องโดยตอบสนองต่อทริกเกอร์ของ Amazon S3 จะเห็นเอาท์พุตคล้ายกับด้านล่างนี้ CONTENT TYPE ที่เห็นจะขึ้นอยู่กับประเภทของไฟล์ที่ถูกอัปโหลดไปยัง bucket

```
2024-03-27T10:46:51.336+07:00	START RequestId: e721d242-a932-4616-8fb3-7d5fd2c032b0 Version: $LATEST
2024-03-27T10:46:52.511+07:00	END RequestId: e721d242-a932-4616-8fb3-7d5fd2c032b0
2024-03-27T10:46:52.511+07:00	REPORT RequestId: e721d242-a932-4616-8fb3-7d5fd2c032b0 Duration: 1173.94 ms Billed Duration: 1174 ms Memory Size: 128 MB Max Memory Used: 36 MB
```

Lab Results

ผลลัพธ์แบบที่ 1 เมื่อ**มี**การกำหนด Destination ให้กับ Lambda Function





หากต้องการส่งบันทึกการเรียกใช้งาน (Invocation Logs) แบบ Asynchronous ที่**สำเร็จ**หรือ**ล้มเหลว**ไปยัง destination resource เมื่อเงื่อนไขสำหรับการเรียกใช้งาน destination เกิดขึ้น, Lambda จะส่งไฟล์ JSON ที่มีรายละเอียดของการเรียกใช้ไปยัง destination resource