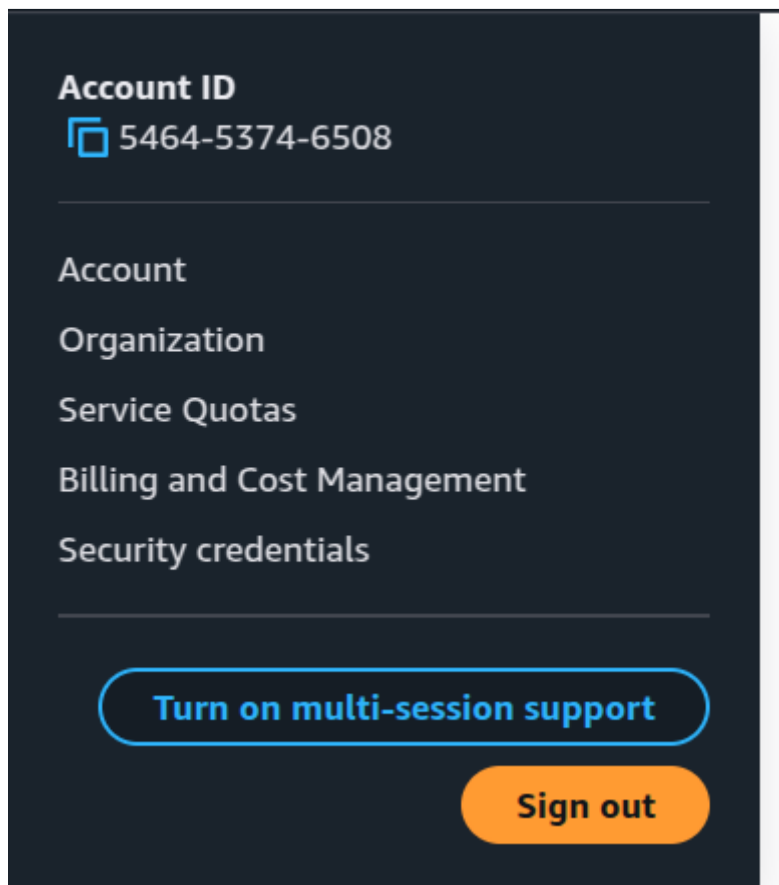


**Q) Create a serverless file content-type detector using S3 and AWS Lambda. Create an S3 bucket and configure S3 Event Notifications to trigger a Lambda function upon file upload. Write a Lambda function in Python or Node.js to retrieve the uploaded file's metadata and log the content type to CloudWatch. 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

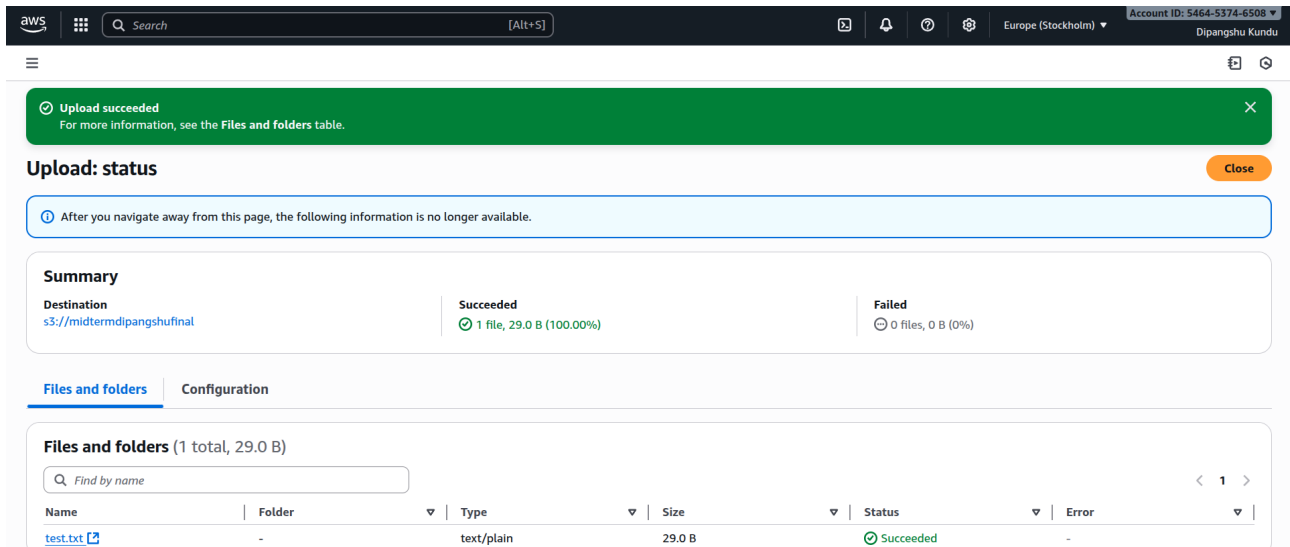
1. Go to **AWS Console** → **S3** → **Create Bucket**.
2. Give it a unique name (e.g., my-lambda-demo-bucket).
3. Choose a region (keep it the same as Lambda).
4. Leave defaults, but **uncheck "Block all public access"** if you need public access.
5. **Create the bucket.**

The screenshot shows the 'Create bucket' page in the AWS S3 console. The top navigation bar includes the AWS logo, a search bar, and the user's account information (Europe (Stockholm), Account ID: 5464-5374-6508, Dipangshu Kundu). The breadcrumb trail is 'Amazon S3 > Buckets > Create bucket'. The main content area is titled 'General configuration'. Under 'AWS Region', it shows 'Europe (Stockholm) eu-north-1'. The 'Bucket type' section has two options: 'General purpose' (selected) and 'Directory'. The 'General purpose' option is described as 'Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.' The 'Directory' option is described as 'Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.' The 'Bucket name' field contains 'midtermdipangshufinal'. Below this, it states 'Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). Learn More'. The 'Copy settings from existing bucket - optional' section has a 'Choose bucket' button. The 'Object Ownership' section has two options: 'ACLs disabled (recommended)' (selected) and 'ACLs enabled'. The 'ACLs disabled' option is described as 'All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.' The 'ACLs enabled' option is described as 'Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.' At the bottom, it says 'Object Ownership: Bucket owner enforced'.

The screenshot shows the 'Buckets' page in the AWS S3 console. The top navigation bar is the same as the previous screenshot. The breadcrumb trail is 'Amazon S3 > Buckets'. A green success message banner at the top says 'Successfully created bucket "midtermdipangshufinal"'. Below this, it says 'To upload files and folders, or to configure additional bucket settings, choose View details.' The main content area is titled 'General purpose buckets (5)'. It includes a search bar with the placeholder 'Find buckets by name' and a list of actions: 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. On the right, there is an 'Account snapshot' section with a 'View dashboard' button. The 'Account snapshot' section says 'Updated daily' and '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.**



## 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.

[Change default execution role](#)

[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** [Throttle](#) [Copy ARN](#) [Actions](#)

[Export to Infrastructure Composer](#) [Download](#)

**Function overview** [Info](#)

[Diagram](#) [Template](#)

**MyS3LambdaFunction**

**Layers** (0)

**S3**

[+ Add trigger](#)

[+ Add destination](#)

**Description**  
-

**Last modified**  
2 hours ago

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

**Function URL** [Info](#)  
-

**Code source** [Info](#)

[Open in Visual Studio Code](#) [Upload from](#)

**Code source**

EXPLORER

lambda\_function.py

**Trigger configuration** [Info](#)

**S3**  
aws asynchronous storage

**Bucket**  
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.  
s3/intermediaryshuffinal  
Bucket region: eu-north-1

**Event types**  
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.  
All object create events X PUT X

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.  
e.g. images/

**Suffix - optional**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.  
e.g. .jpg

**Recursive invocation**  
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)  
☒ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

[Cancel](#) [Add](#)

## 5. Write lambda Code

**Successfully updated the function MyS3LambdaFunction.**

**MyS3LambdaFunction**  
Layers (0)

**S3** (2)  
[+ Add trigger](#)

[+ Add destination](#)

**Last modified**  
18 minutes ago

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

**Function URL** [Info](#)

**Code source** [Info](#)

[Open in Visual Studio Code](#) [Upload from](#)

**EXPLORER**  
MY\_S3\_LAMBDA\_FUNCTION  
lambda\_function.py

```

1  def lambda_handler(event, context):
2      print("Received event:", json.dumps(event))
3
4      for record in event['Records']:
5          # Extract bucket and object key from the event record
6          bucket = record['s3']['bucket']['name']
7          key = record['s3']['object']['key']
8
9          try:
10             # Use head object to get object metadata without downloading the file
11             response = s3.head_object(Bucket=bucket, Key=key)
12             content_type = response['ContentType']
13
14             # Log the content type to CloudWatch
15             print(f'Result for bucket {bucket}: has a Content Type of: {content_type}')
16
17
18
19
20
21

```

**DEPLOY**  
[Deploy \(Ctrl+Shift+U\)](#)  
[Test \(Ctrl+Shift+I\)](#)

```
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}")

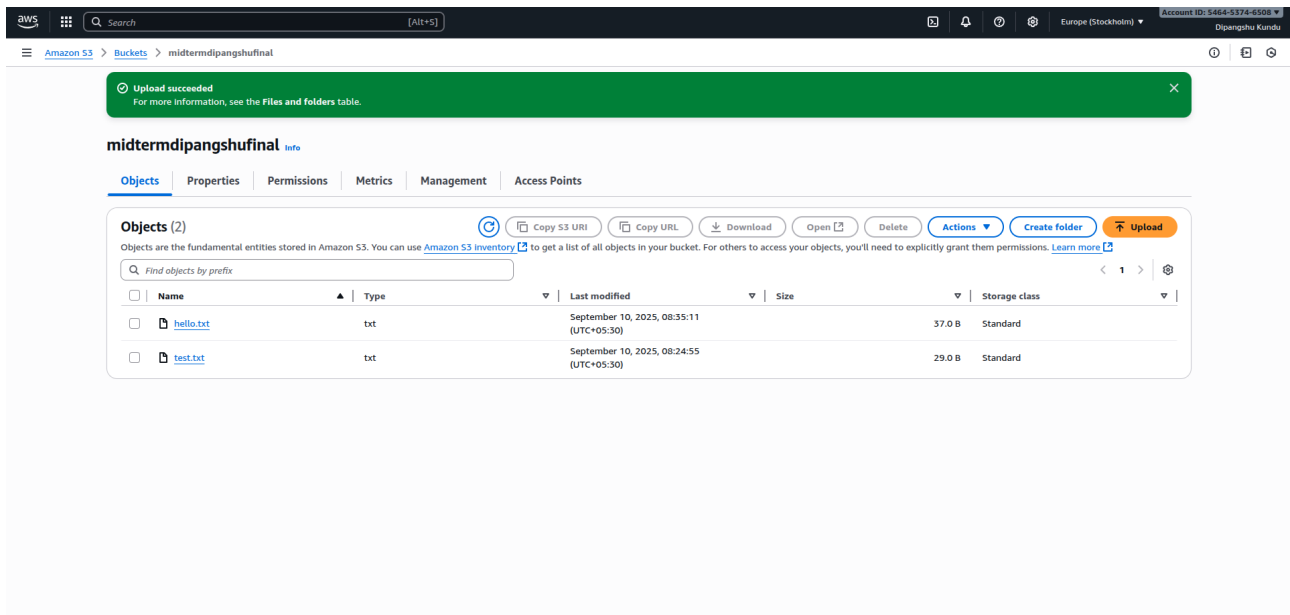
    except Exception as e:
        print(f"Error getting metadata for file '{key}': {e}")

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

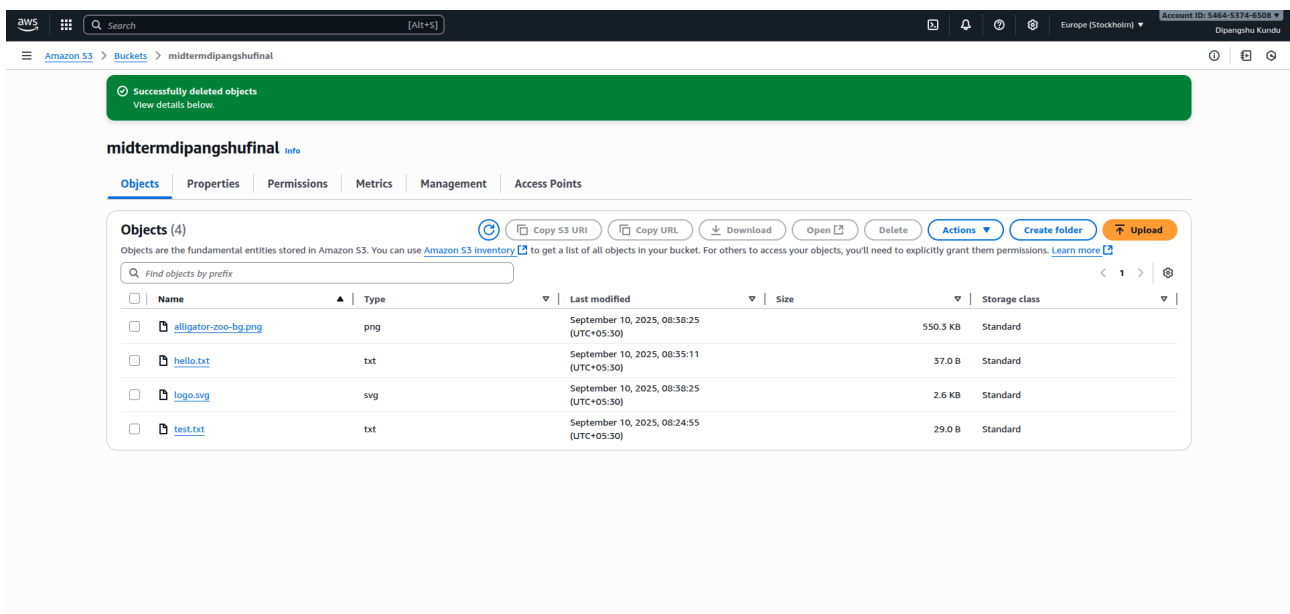
---

## 6. Test the Integration

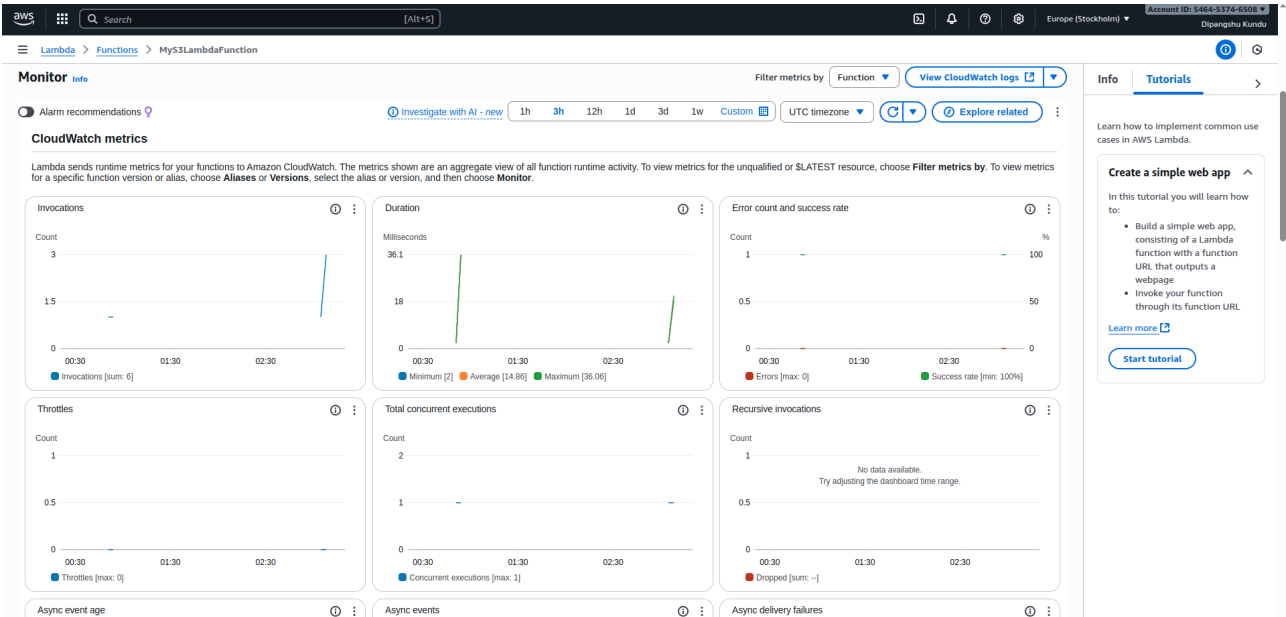
1. We Go to our **S3 bucket**.
2. Upload a new file (e.g., hello.txt).



## 7. UPLOADING THREE DIFFERENT TYPES OF FILES



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	▶ 2025-09-10T03:05:11.588Z	START RequestId: f8624282-5e11-4a4d-bd7c-9c360310ed9d Version: \$LATEST
Log Anomalies	▶ 2025-09-10T03:05:11.581Z	Event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "eu-north-1", "eventTime": "2025-09-10T03:05:10.375Z", "eventName": "ObjectCreated:Put", "u...
Live Tail	▶ 2025-09-10T03:05:11.581Z	New file uploaded: hello.txt (37 bytes) in bucket midtermdipangshufinal
Logs Insights	▶ 2025-09-10T03:05:11.583Z	END RequestId: f8624282-5e11-4a4d-bd7c-9c360310ed9d
Contributor Insights	▶ 2025-09-10T03:05:11.583Z	REPORT RequestId: f8624282-5e11-4a4d-bd7c-9c360310ed9d 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	▶ 2025-09-10T03:08:24.666Z	START RequestId: cd681c3e-6c61-4f3d-9b49-f0d1124f65fb Version: \$LATEST
▶ Application Signals (APM)	▶ 2025-09-10T03:08:24.668Z	Event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "eu-north-1", "eventTime": "2025-09-10T03:08:24.139Z", "eventName": "ObjectCreated:Put", "u...
▶ Network Monitoring	▶ 2025-09-10T03:08:24.668Z	New file uploaded: alligator-zoo-bg.png (563551 bytes) in bucket midtermdipangshufinal
	▶ 2025-09-10T03:08:24.686Z	END RequestId: cd681c3e-6c61-4f3d-9b49-f0d1124f65fb
	▶ 2025-09-10T03:08:24.686Z	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	▶ 2025-09-10T03:08:25.466Z	START RequestId: 3d2d9628-1518-4d5d-a3ce-3ad7af99c763 Version: \$LATEST
Settings	▶ 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...
Telemetry config	▶ 2025-09-10T03:08:25.466Z	New file uploaded: logo.svg (2632 bytes) in bucket midtermdipangshufinal
Getting Started	▶ 2025-09-10T03:08:25.486Z	END RequestId: 3d2d9628-1518-4d5d-a3ce-3ad7af99c763
What's new	▶ 2025-09-10T03:08:25.486Z	REPORT RequestId: 3d2d9628-1518-4d5d-a3ce-3ad7af99c763 Duration: 29.25 ms Billed Duration: 21 ms Memory Size: 128 MB Max Memory Used: 36 MB