

# **Serverless File Processing & Monitoring System using AWS**

## **Abstract**

This project demonstrates a simple yet powerful serverless architecture using Amazon Web Services (AWS). The system automatically detects file uploads to an Amazon S3 bucket, processes events using AWS Lambda, logs details to Amazon CloudWatch, and sends real-time notifications through Amazon SNS. The project showcases essential AWS concepts, including serverless computing, event-driven automation, cloud monitoring, and secure resource management.

## **1. Introduction**

AWS provides a scalable and cost-effective cloud platform for deploying applications without managing servers. Serverless architecture allows functions to run only when triggered, reducing operational overhead. In this project, a file monitoring system is built using AWS S3, Lambda, CloudWatch, and SNS. Whenever a user uploads a file, the system automatically triggers a workflow: processing, logging, and alerting.

This project aligns with AWS Cloud Practitioner concepts and demonstrates hands-on cloud implementation using AWS core services.

## **2. Aim**

To build a fully serverless file processing and monitoring system using AWS services that automatically responds to S3 file uploads and provides real-time visibility through monitoring and notifications.

## **3. Objectives**

1. To implement a serverless event-driven workflow using AWS Lambda.
2. To store and manage files using Amazon S3.
3. To enable real-time logs and monitoring using Amazon CloudWatch.
4. To configure Amazon SNS for alert notifications.
5. To demonstrate basic cloud security using IAM roles and permissions.

#### **4. AWS Services Used**

- **Amazon S3 – File storage and event trigger**
- **AWS Lambda – Event-driven compute function**
- **Amazon SNS – Email notifications to the user**
- **Amazon CloudWatch – Logs, monitoring, and dashboards**
- **AWS IAM – Secure access and execution roles**

#### **5. System Architecture**

##### **Workflow:**

1. User uploads a file to Amazon S3
2. S3 sends an event to AWS Lambda
3. Lambda processes the event and logs details
4. Logs appear in CloudWatch
5. Lambda sends notification through Amazon SNS
6. User receives an email alert

#### **6. Implementation Steps**

##### **Step 1 — S3 BUCKET SETUP (Storage Lead)**

###### **Steps to Perform**

###### **1. Create S3 Bucket**

1. Go to **AWS Console → S3**
2. Click **Create bucket**
3. Enter bucket name → serverless-file-bucket-yourname
4. Region → **us-east-1**
5. Keep **Block all public access = ON**
6. Click **Create bucket**

###### **2. Create Folder Structure**

Inside the bucket:

- Create folder: uploads/
- Create folder: processed/

### 3. Upload Test Files

1. Go to folder uploads/
2. Click **Upload → Add files**
3. Upload test1.txt, sample2.txt

### 4. Verify Bucket

- Check that files show up
- Confirm bucket name for other members

## Step 2 — IAM SETUP (Security Lead)

### Steps to Perform

#### 1. Create IAM Role for Lambda

1. Go to **IAM → Roles → Create role**
2. Select **AWS Service → Lambda**
3. Attach policies:
  - AmazonS3FullAccess
  - CloudWatchLogsFullAccess
  - AmazonSNSFullAccess
4. Name the role:  
**LambdaS3ExecutionRole**

#### 2. Create IAM User (optional for demo)

- Username → project-user
- Permission → S3FullAccess
- Enable **MFA**
- Download credentials

#### 3. Security Verification

- Ensure no public access to bucket

- Ensure Lambda uses the created role

## Step 3 — LAMBDA FUNCTION (Automation Lead)

### Steps to Perform

#### 1. Create Lambda Function

1. Go to **Lambda**
2. Click **Create function**
3. Choose **Author from scratch**
4. Name → FileEventProcessor
5. Runtime → **Python 3.9**
6. Execution role → **Use existing role** → **LambdaS3ExecutionRole**

#### 2. Add S3 Trigger

1. Go to Lambda → **Triggers**
2. Add Trigger → **S3**
3. Select bucket: serverless-file-bucket-yourname
4. Event type: **All object create events**
5. Prefix: uploads/

#### 3. Add Code

Paste this code:

```
import json
```

```
import boto3
```

```
sns = boto3.client('sns')
```

```
def lambda_handler(event, context):
```

```
    bucket = event['Records'][0]['s3']['bucket']['name']
```

```
    file_name = event['Records'][0]['s3']['object']['key']
```

```

message = f"New file uploaded: {file_name} in bucket: {bucket}"
print(message)

sns.publish(
    TopicArn='YOUR_SNS_TOPIC_ARN',
    Message=message,
    Subject='New File Upload Alert'
)

return {
    'statusCode': 200,
    'body': json.dumps('File processed successfully!')
}

```

Click **Deploy**.

#### **4. Test Lambda**

- Upload a new file to S3 → uploads/test3.txt
- Check logs in CloudWatch

### **Step 4 — CLOUDWATCH MONITORING (Observability Lead)**

#### **Steps to Perform**

##### **1. View Lambda Logs**

1. Go to **CloudWatch → Logs**
2. Click log group: /aws/lambda/FileEventProcessor
3. Open latest log stream
4. Verify log:  
“New file uploaded: uploads/test3.txt”

##### **2. Create CloudWatch Dashboard**

1. Go to **Dashboards → Create Dashboard**

2. Name: ServerlessMonitoringDashboard

3. Add widgets:

- Lambda Invocations
- Lambda Errors
- S3 Metrics
- SNS Delivery Status

### **3. Create Alarm (Optional)**

1. Go to **Alarms** → **Create alarm**
2. Select Errors metric for Lambda
3. Condition → Errors ≥ 1
4. Notification → SNS topic

## **Step 5: SNS NOTIFICATIONS (Alerting Lead)**

### **Steps to Perform**

#### **1. Create SNS Topic**

1. Go to **SNS** → **Topics** → **Create topic**
2. Type → **Standard**
3. Name → FileUploadAlerts

#### **2. Add Email Subscription**

1. SNS → Subscriptions → Create subscription
2. Protocol → Email
3. Enter team email IDs
4. Each member must click **Confirm Subscription** in email

#### **3. Connect SNS to Lambda**

Replace in code:

TopicArn='YOUR\_SNS\_TOPIC\_ARN',

With your actual SNS topic ARN.

#### **4. Test Email Alert**

- Upload AlertTest.txt to S3
- All team members should receive email:  
**“New file uploaded: uploads/AlertTest.txt”**

## 7. OUTPUT

- S3 successfully stored uploaded files
- Lambda function triggered automatically
- CloudWatch generated real-time logs
- SNS sent email notifications instantly
- Full serverless workflow executed without manual intervention

## 8. Screenshot:

### Step 1 — S3 BUCKET SETUP (Storage Lead)

The image consists of three vertically stacked screenshots of the AWS S3 console interface.

**Screenshot 1: Summary**

This screenshot shows the summary page for an S3 bucket named "serverless-file-bucket-sharing". It displays two successful uploads: "SAMPLE 1.txt" and "TEST1.txt", both of which are text/plain files with 0B size and a status of "Succeeded".

Name	Folder	Type	Size	Status
SAMPLE 1.txt	-	text/plain	0 B	Succeeded
TEST1.txt	-	text/plain	0 B	Succeeded

**Screenshot 2: Upload Page**

This screenshot shows the "Upload" page for the same bucket. It allows users to drag and drop files or add them via a file browser. A table lists the files being uploaded: "SAMPLE 1.txt" and "TEST1.txt", both of which are text/plain files with 0B size.

Name	Folder	Type	Size
SAMPLE 1.txt	-	text/plain	0 B
TEST1.txt	-	text/plain	0 B

**Screenshot 3: Bucket Details**

This screenshot shows the main bucket details page for "serverless-file-bucket-sharing". A green success message at the top states "Successfully created folder 'processed'." The "Objects" tab is selected, showing two folders: "processed/" and "uploads/".

Name	Type	Last modified	Size	Storage class
processed/	Folder	-	-	-
uploads/	Folder	-	-	-

aws | Search [Alt+S] | Account ID: 0376-8884-8471 | Europe (Stockholm) | Santhosh

Amazon S3

**Amazon S3**

General purpose buckets

Directory buckets

Table buckets

Vector buckets

Access Grants

Access Points (General Purpose Buckets, Fsx file systems)

Access Points (Directory Buckets)

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this bucket

General purpose buckets (1) [Info](#)

Buckets are containers for data stored in S3.

Find buckets by name

Name | AWS Region | Creation date

Name	AWS Region	Creation date
<a href="#">serverless-file-bucket-sharing</a>	US East (N. Virginia) us-east-1	November 24, 2025, 15:07:24 (UTC+05:30)

[Create bucket](#)

**Account snapshot** [Info](#) [View dashboard](#) Updated daily

Storage Lens provides visibility into storage usage and activity trends.

**External access summary - new** [Info](#) Updated daily

External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws | Search [Alt+S] | Account ID: 0376-8884-8471 | United States (N. Virginia) | Santhosh

Amazon S3 > Buckets > Create bucket

Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the Storage tab or the [Amazon S3 pricing page](#).

Server-side encryption with Amazon S3 managed keys (SSE-S3)

Server-side encryption with AWS Key Management Service keys (SSE-KMS)

Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

**Bucket Key**

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable

Enable

**Advanced settings**

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Create bucket](#)

aws | Search [Alt+S] | Account ID: 0376-8884-8471 | United States (N. Virginia) | Santhosh

Amazon S3 > Buckets > Create bucket

**Create DUCKET** [Info](#)

Buckets are containers for data stored in S3.

**General configuration**

**AWS Region**  
US East (N. Virginia) us-east-1

**Bucket type** [Info](#)

General purpose  
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.

Directory  
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.

**Bucket name** [Info](#)

serverless-file-bucket-sharing

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](#)

**Copy settings from existing bucket - optional**

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Step 2 — IAM SETUP (Security Lead)

The screenshots illustrate the process of creating an IAM role:

- Step 1: Select trusted entity**

The first screenshot shows the "Select trusted entity" step. The "AWS service" option is selected, which is described as allowing AWS services like EC2, Lambda, or others to perform actions in the account.
- Name, review, and create**

The second screenshot shows the "Name, review, and create" step. The role name is set to "LambdaS3ExecutionRole". The description is "Allows Lambda functions to call AWS services on your behalf".
- Permissions**

The third screenshot shows the "Permissions" tab of the role's configuration page. It lists three attached managed policies: "AmazonS3FullAccess", "AmazonSNSFullAccess", and "CloudWatchLogsFullAccess", all of which are AWS managed policies.

## Step 3 — LAMBDA FUNCTION (Automation Lead)

The screenshot shows the 'Create function' configuration page for AWS Lambda. In the 'Execution role' section, 'Use an existing role' is selected, and 'LambdaS3ExecutionRole' is chosen from a dropdown. Below this, under 'Additional configurations', there is a note about setting up networking, security, and governance.

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).  
 Create a new role with basic Lambda permissions  
 Use an existing role  
 Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
LambdaS3ExecutionRole

**Additional configurations**  
Use additional configurations to set up networking, security, and governance for your function. These settings help secure and customize your Lambda function deployment.

**Add trigger**

**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/serverless-file-bucket-sharing

**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

**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.  
uploads/

**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.

**Add**

Screenshot of the AWS Lambda console showing the successful update of the function FileEventProcessor.

**FileEventProcessor**

**Description**

- Last modified: 21 seconds ago
- Function ARN: arn:aws:lambda:us-east-1:03768848471:function:FileEventProcessor
- Function URL: [Info](#)

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

**Diagram** | [Template](#)

**Trigger**

+ Add destination | + Add trigger

**CloudShell** | [Feedback](#) | [Console Mobile App](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Screenshot of the AWS Lambda console showing the successful update of the function FileEventProcessor.

**FileEventProcessor**

**Code Editor**

**lambda\_function.py**

```

1 import json
2
3 def lambda_handler(event, context):
4     bucket_name = event['Records'][0]['s3']['bucket']['name']
5     file_name = event['Records'][0]['s3']['object']['key']
6
7     print(f"New note uploaded: {file_name} in bucket: {bucket_name}")
8
9     return {
10         'statusCode': 200,
11         'body': json.dumps('Note upload logged successfully!')
12     }

```

Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)

**EXPLORER**

**FILEEVENTPROCESSOR**

lambda\_function.py

**DEPLOY** △ Undeployed

Deploy (Ctrl+Shift+U) | Test (Ctrl+Shift+I)

**TEST EVENTS [NONE SELECTED]**

[FileEventProcessor](#)

[CloudShell](#) | [Feedback](#) | [Console Mobile App](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Screenshot of the AWS CloudWatch logs for the FileEventProcessor function.

**Log events**

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

**Display**

Timestamp	Message
2025-11-24T09:55:34.308Z	INIT_START Runtime Version: python:3.9.v125 Runtime Version ARN: arn:aws:lambda:us-east-1::run...
2025-11-24T09:55:34.389Z	START RequestId: 5bbfbfa02-d75a-49d5-a223-3a043e9ce0ed Version: \$LATEST
2025-11-24T09:55:34.390Z	New note uploaded: uploads/SAMPLE+2.txt in bucket: serverless-file-bucket-sharing
2025-11-24T09:55:34.391Z	END RequestId: 5bbfbfa02-d75a-49d5-a223-3a043e9ce0ed
2025-11-24T09:55:34.392Z	REPORT RequestId: 5bbfbfa02-d75a-49d5-a223-3a043e9ce0ed Duration: 1.34 ms Billed Duration: 79 ms...

No newer events at this moment. Auto retrying... [Pause](#)

**CloudWatch**

Favorites and recents

Alarms

AI Operations

GenAI Observability

Application Signals

Infrastructure Monitoring

Logs

Log groups

Log Anomalies

Live Tail

[CloudShell](#) | [Feedback](#) | [Console Mobile App](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

## Step 4 – CLOUDWATCH MONITORING (Observability Lead)

The screenshot shows the AWS CloudWatch Metrics interface. At the top, there is a search bar and navigation links for 'CloudWatch' and 'Dashboards'. Below this, a modal window titled 'Create new dashboard' is open, showing a 'Dashboard name' input field with 'ServerlessMonitoringDashboard' typed in. There is also a note about valid characters and a 'Create dashboard' button. The main dashboard area shows a list of metrics under 'Graphed metrics (1)'. One metric, 'FileEventProcessor Errors', is selected and highlighted with a blue border. The 'Create widget' button is visible at the bottom right of the modal. The bottom part of the screenshot shows the final state of the dashboard, which now contains a single metric graph for 'Errors' over time.

**Create new dashboard**

Dashboard name: ServerlessMonitoringDashboard

Valid characters in dashboard names include "0-9A-Za-z-\_".

Create dashboard

**Add metric graph**

Errors

Persist time range: 3h

Graphed metrics (1): FileEventProcessor Errors

Create widget

**ServerlessMonitoringDashboard**

Count: 1

0.5

0 07:05 10:05

Errors

Autosave: Off

Actions: Save

## Step 5: SNS NOTIFICATIONS (Alerting Lead)

The screenshots illustrate the process of setting up SNS notifications for file uploads.

**Screenshot 1: Create topic**

**Details**

Type: **Info** (Standard selected)

Topic type cannot be modified after topic is created

**FIFO (first-in, first-out)**

- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

**Standard**

- Best-effort message ordering
- At-least once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

**Name**: FileUploadAlerts

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (\_).

**Display name - optional** (**Info**)

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

**My Topic**

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Account ID: 0376-8884-8471 Santhosh

**Screenshot 2: Create subscription**

**Details**

**Topic ARN**: arn:aws:sns:us-east-1:037688848471:FileUploadAlerts

**Protocol**: Email

**Endpoint**: An email address that can receive notifications from Amazon SNS. jupg24mca23173@jainuniversity.ac.in

After your subscription is created, you must confirm it. **Info**

**Subscription filter policy - optional** (**Info**)

**Screenshot 3: Create subscription**

**Details**

**Topic ARN**: arn:aws:sns:us-east-1:037688848471:FileUploadAlerts

**Protocol**: Email

**Endpoint**: An email address that can receive notifications from Amazon SNS. santhosh.govindan0932@gmail.com

After your subscription is created, you must confirm it. **Info**

us-east-1.console.aws.amazon.com/console/home?region=us-east-1 © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Account ID: 0376-8884-8471 Santhosh

Screenshot of the AWS SNS console showing the creation of an SNS topic named "FileUploadAlerts". The topic is of type "Standard" and has an ARN of "arn:aws:sns:us-east-1:037688848471:FileUploadAlerts". It is owned by the user with ID "037688848471".

The "Subscriptions" tab is selected, showing one subscription:

ID	Endpoint	Status	Protocol
Deleted	santhosh.govindan0932...	Confirmed	EMAIL

Screenshot of the AWS Lambda console showing the successful update of a Lambda function named "FileEventProcessor". The function code is as follows:

```
def lambda_handler(event, context):
    key = event['Records'][0]['s3']['object']['key']
    file_name = event['Records'][0]['s3']['object']['name']

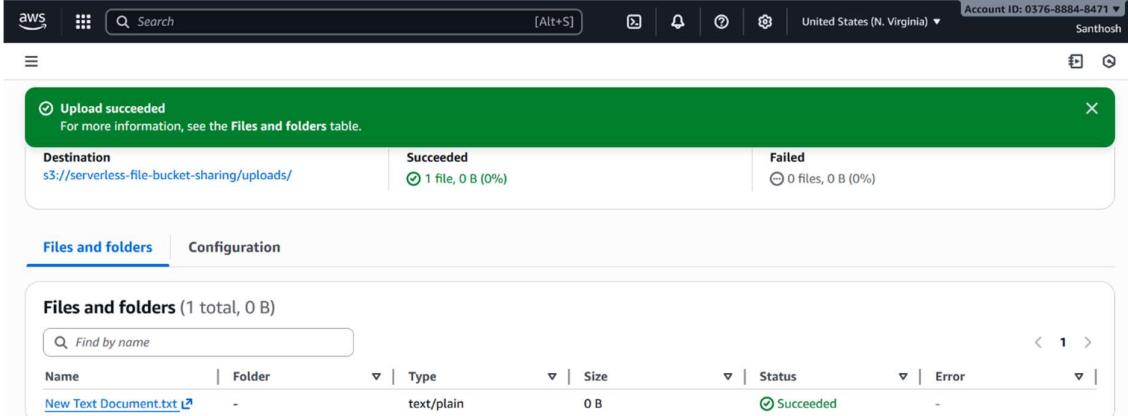
    message = f"New file uploaded: {file_name} in bucket: {bucket}"
    print(message)

    sns.publish(
        TopicArn='arn:aws:sns:us-east-1:037688848471:FileUploadAlerts',
        Message=message,
        Subject='New File Upload Alert'
    )

    return {
        'statusCode': 200,
        'body': json.dumps('File processed successfully!')
    }
```

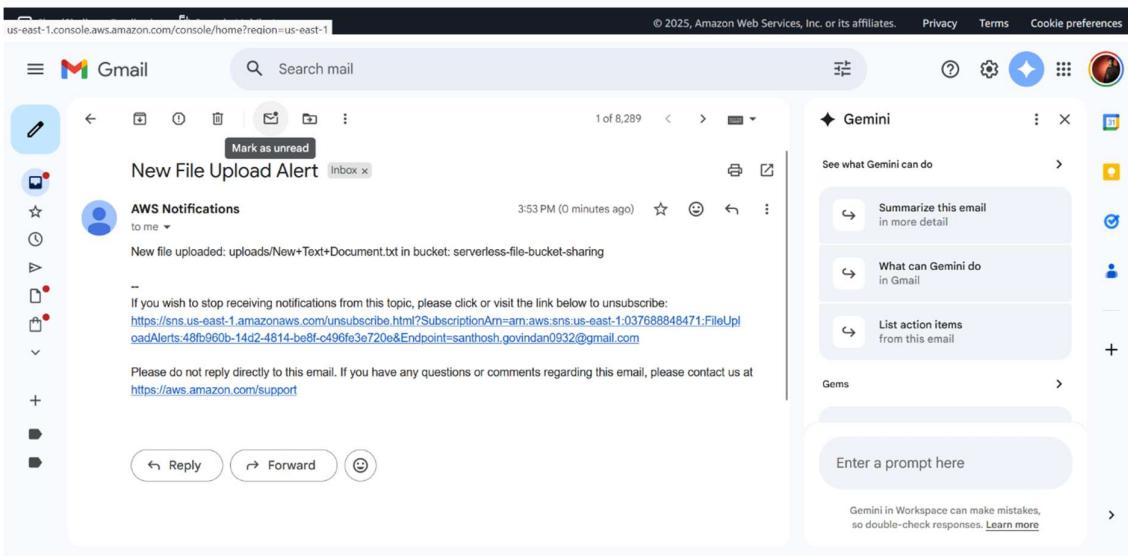
A success message "Successfully updated the function FileEventProcessor." is displayed at the bottom.

## Final verification:



The screenshot shows the AWS CloudWatch Metrics interface. A green notification bar at the top indicates "Upload succeeded" with the message "For more information, see the Files and folders table." Below this, there are two sections: "Succeeded" (1 file, 0 B (0%)) and "Failed" (0 files, 0 B (0%)).

Below the metrics, the "Files and folders" tab is selected, showing a table with one item: "New Text Document.txt" (text/plain, 0 B, Succeeded).

The screenshot shows the AWS Lambda function execution log. It displays the logs for a successful file upload. The log message reads:

```
2023-10-12T14:53:48.000Z - INFO - New file uploaded: uploads/New+Text+Document.txt in bucket: serverless-file-bucket-sharing
```

## 9. Conclusion

This project demonstrates a complete serverless event-driven workflow using AWS cloud services. With S3 for storage, Lambda for processing, CloudWatch for monitoring, and SNS for notifications, the system showcases a practical real-world cloud solution. The implementation proves how AWS serverless computing can simplify development and automate workflows with minimal setup and zero server management.

### ⭐ If you want, I can now generate:

- 📌 A downloadable Word (.docx) file of this report
- 📌 A matching PPT
- 📌 A GitHub README file

Just tell me: "**Generate Word file now**" and I will create it.