Serverless Project Documentation

Prepared by: Reesa Susan Sabu
Date: December 22, 2024
Email: reesasusan@gmail.com

Introduction

This document outlines the implementation details of a serverless project leveraging AWS services. The project demonstrates a fully functional application featuring file uploads, notifications, and REST APIs, adhering to modern serverless architecture principles.

Table of Contents

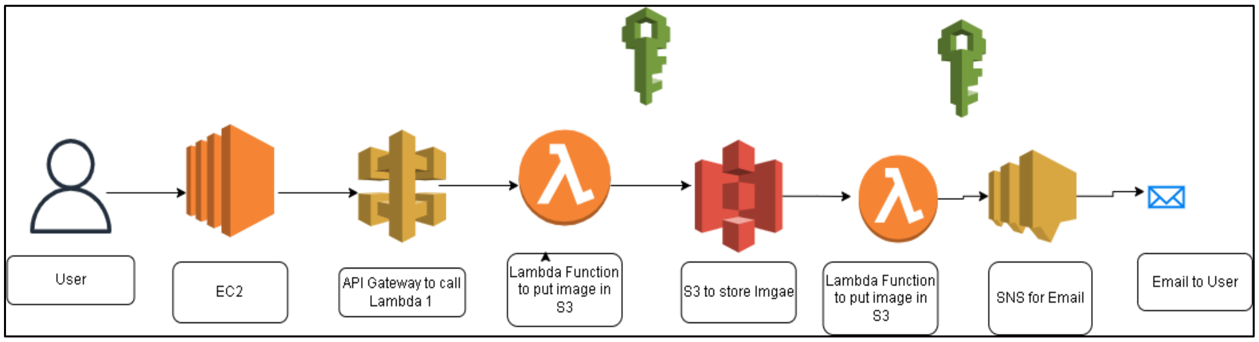
1. Project Overview

This project is a serverless application implemented using:
- AWS Lambda for backend logic.
- Amazon S3 for storage.
- API Gateway to expose the Lambda function as a REST API.
- Amazon SNS for notifications.

The primary objective is to enable efficient file uploads and real-time notifications.

2. System Architecture

Below is the architecture diagram for the project:


User
EC2
API Gateway to call Lambda 1
Lambda Function to put image in S3
S3 to store Imgae
Lambda Function to put image in S3
SNS for Email
Email to User

3. Implementation Steps

1. Environment Setup

- **Provision EC2 Instance**:
  - Use AWS Management Console or CLI to create an EC2 instance.
- **Install Git**:

```
sudo yum install git
```

- **Clone Repository**:

```
git clone https://github.com/bhatiasimarjeet/serverless_repo.git
cd serverless_repo
```

- **Install Node.js**:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
 . ~/.nvm/nvm.sh
 nvm install 16.8.0
node -e "console.log('Running Node.js ' + process.version)"
```

- **Install Dependencies**:

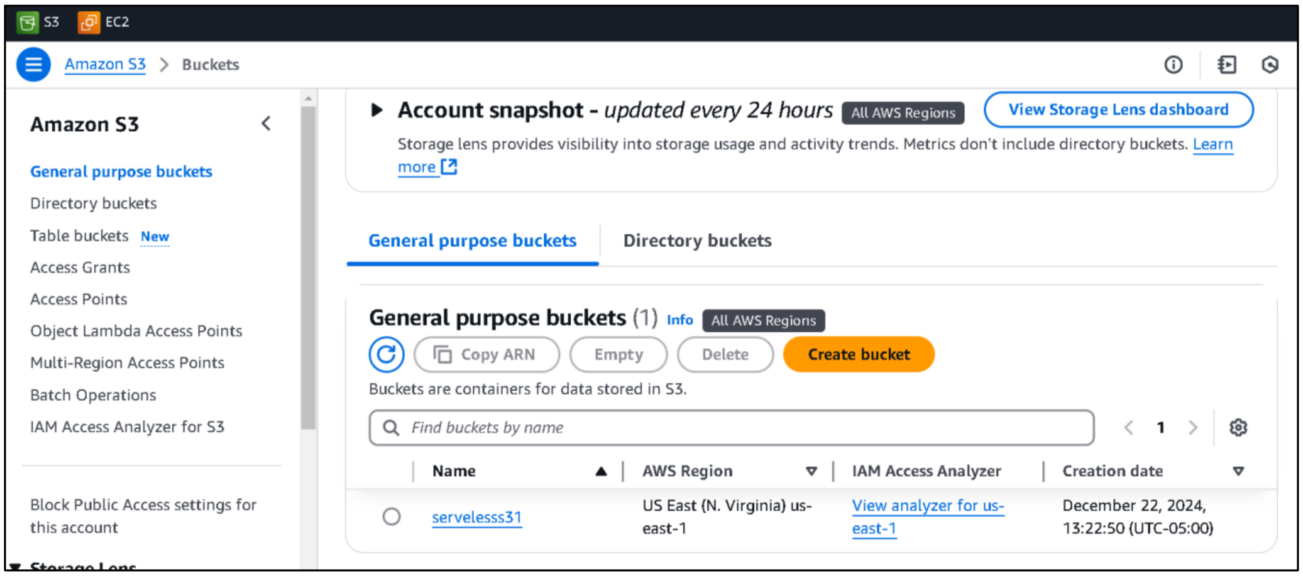
```
npm install
npm run serve
```

- **Security Group Update**:
  - Add a custom TCP rule for port 8080 if connection times out.

2. S3 Bucket Configuration

- **Create an S3 Bucket**:
- Ensure “Block All Public Access” is disabled.
- **Configure CORS**:

```
[
 {
  "AllowedHeaders": ["*"],
  "AllowedMethods": ["PUT", "POST", "GET",
"HEAD"],
  "AllowedOrigins": ["*"],
  "ExposeHeaders": []
 }
]
```



Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. Learn more

General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
serveless31	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 22, 2024, 13:22:50 (UTC-05:00)

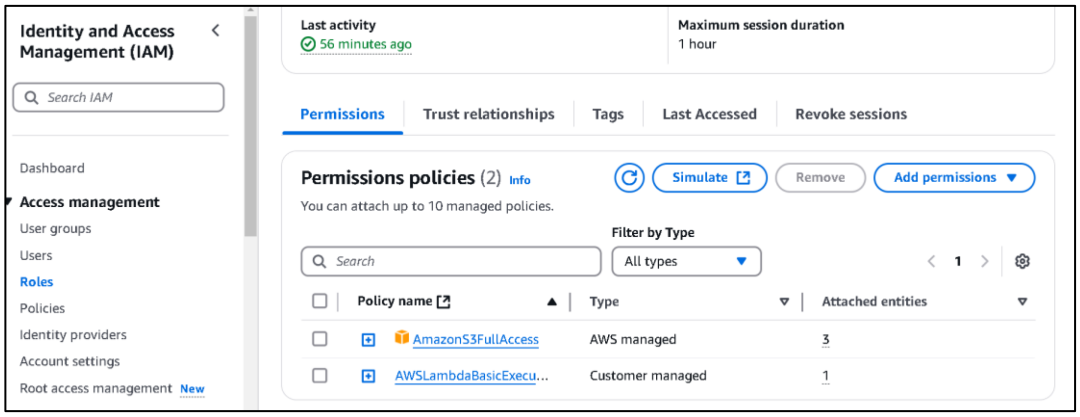
3. Lambda Functions

3.1 Lambda Function 1 (Node.js)

- **Create Lambda Function**:
  - Use Node.js 16 runtime.
- Code**:**

```
import { S3Client } from '@aws-sdk/client-s3';

import { getSignedUrl } from '@aws-sdk/s3-request-presigner';

import { PutObjectCommand } from '@aws-sdk/client-s3';



const s3 = new S3Client({ region: process.env.REGION });

const uploadBucket = 'liveproject4testing';



export const handler = async (event) => {

  const result = await getUploadURL();

  console.log('Result: ', result);

  return result;

};



const getUploadURL = async () => {

  const actionId = Date.now();
const command = new PutObjectCommand({
    Bucket: uploadBucket,
    Key: `${actionId}.jpg`,
    ContentType: 'image/jpeg',
  });
  const uploadURL = await getSignedUrl(s3, command, { expiresIn: 3600 });
// URL expires in 1 hour
  return {
    statusCode: 200,
    isBase64Encoded: false,
    headers: {
      'Access-Control-Allow-Origin': '*',
    },
    body: JSON.stringify({
      uploadURL,
      photoFilename: `${actionId}.jpg`,
    }),
  };
};
```

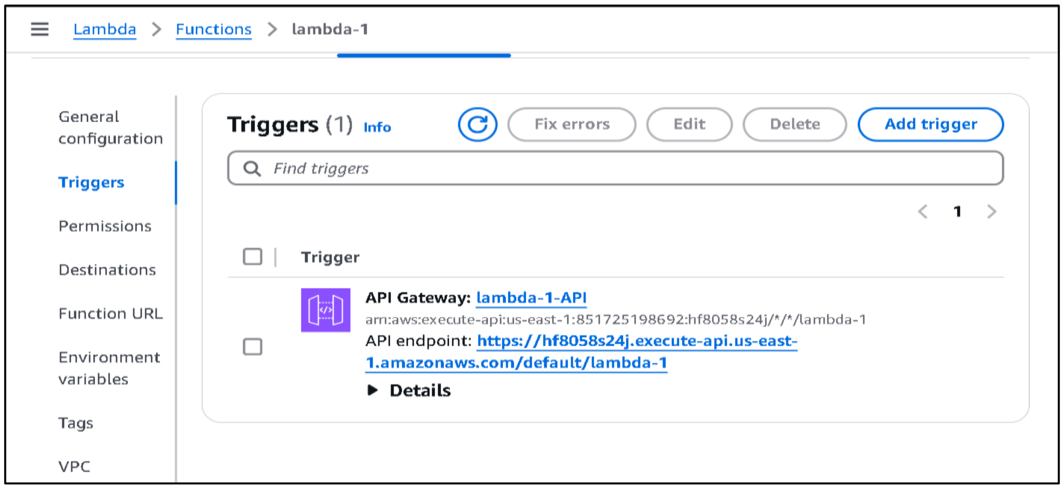
- **IAM Role**:
  - Assign "S3 Full Access" permissions to the Lambda function.



Permissions policies (2) Info
You can attach up to 10 managed policies.

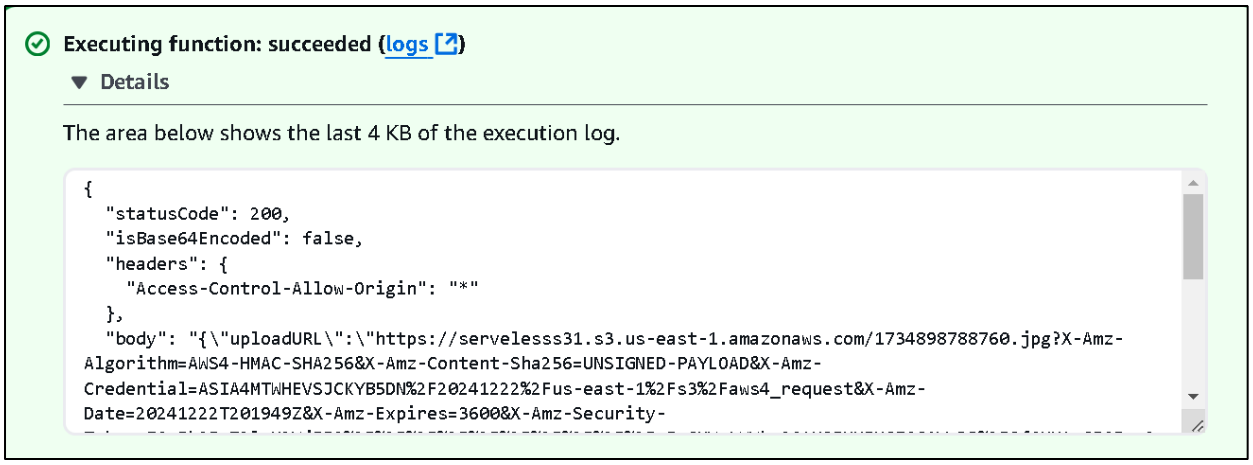
Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	3
AWSLambdaBasicExecu...	Customer managed	1

- **Add API Gateway Trigger**:
    - Configure REST API with Open Access.



API Gateway: lambda-1-API
arn:aws:execute-api:us-east-1:851725198692:hf8058s24j/*/*/lambda-1
API endpoint: https://hf8058s24j.execute-api.us-east-1.amazonaws.com/default/lambda-1

- **Test the Function**



Executing function: succeeded (logs)

The area below shows the last 4 KB of the execution log.

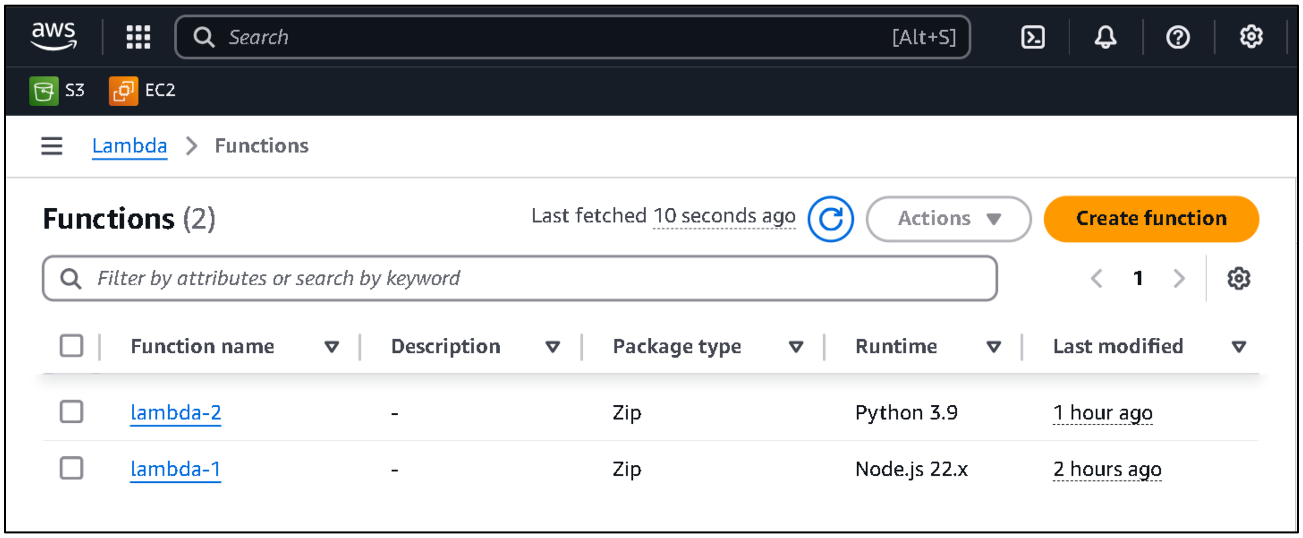
```
{
  "statusCode": 200,
  "isBase64Encoded": false,
  "headers": {
    "Access-Control-Allow-Origin": "*"
  },
  "body": "{\"uploadURL\":\"https://serveless31.s3.us-east-1.amazonaws.com/1734898788760.jpg?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-Content-Sha256=UNSIGNED-PAYLOAD&X-Amz-
Credential=ASIA4MTWHEVSJCKYB5DN%2F20241222%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Date=20241222T201949Z&X-Amz-Expires=3600&X-Amz-Security-
```

3.2 Lambda Function 2 (Python)

- **Create Lambda Function**:
    - Use Python 3.9 runtime.

- **Code**:

```
import boto3

def lambda_handler(event, context):

  MY_SNS_TOPIC_ARN = 'arn:aws:sns:us-east-
1:388333323558:topicforserverlessproject'

  sns_client = boto3.client('sns')

  sns_client.publish(

    TopicArn=MY_SNS_TOPIC_ARN,

    Subject='Request Submitted',

    Message='Request Submitted'

  )
```

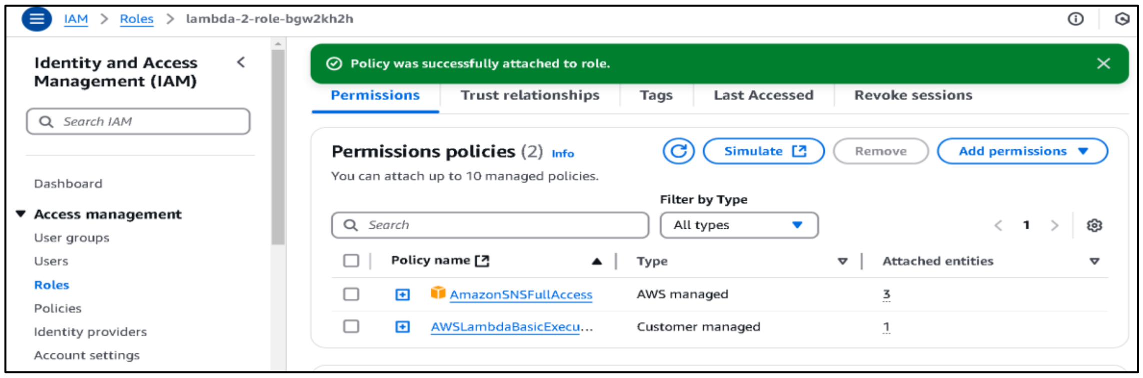


Lambda > Functions

Functions (2) — Last fetched 10 seconds ago

Function name	Description	Package type	Runtime	Last modified
lambda-2	-	Zip	Python 3.9	1 hour ago
lambda-1	-	Zip	Node.js 22.x	2 hours ago

- **IAM Role**:
  - Assign "SNS Full Access" permissions.



IAM > Roles > lambda-2-role-bgw2kh2h

Policy was successfully attached to role.

Permissions policies (2)

You can attach up to 10 managed policies.

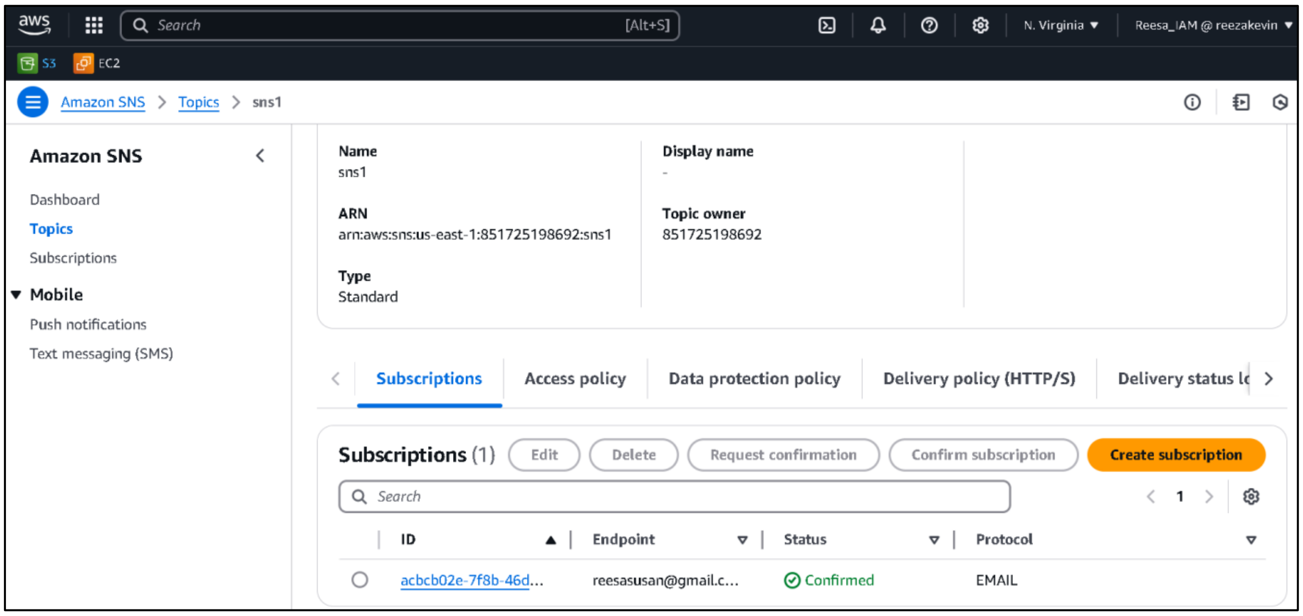
Filter by Type: All types

Policy name	Type	Attached entities
AmazonSNSFullAccess	AWS managed	3
AWSLambdaBasicExecu...	Customer managed	1

- **Add S3 Trigger**:
    - Configure Lambda to trigger on object creation.

4. SNS Notifications

- **Create SNS Topic**:
    - Confirm email subscription.



Name
sns1

ARN
arn:aws:sns:us-east-1:851725198692:sns1

Type
Standard

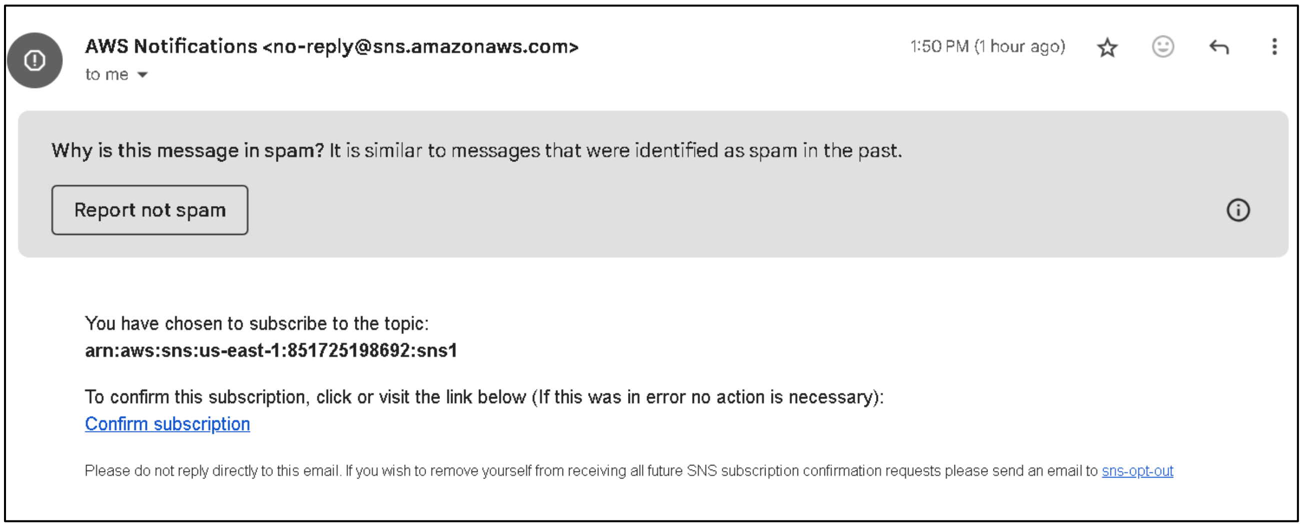
Display name
-

Topic owner
851725198692

Subscriptions (1)

ID	Endpoint	Status	Protocol
acbcb02e-7f8b-46d...	reesasusan@gmail.c...	Confirmed	EMAIL

- **Test Notifications**:
    - Validate email receipt after Lambda execution.



AWS Notifications <no-reply@sns.amazonaws.com>
to me
1:50 PM (1 hour ago)

Why is this message in spam? It is similar to messages that were identified as spam in the past.

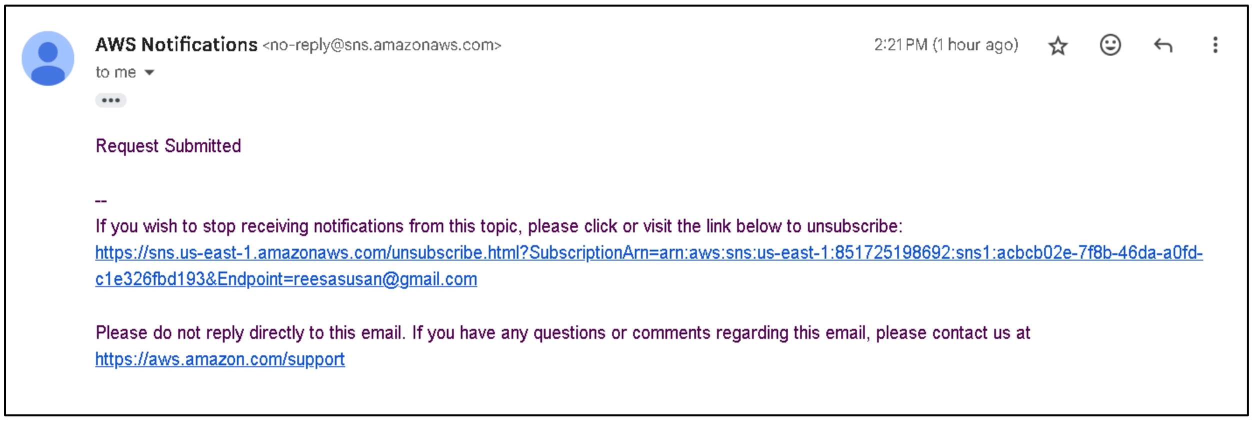
You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:851725198692:sns1

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

5. Testing and Deployment

- **End-to-End Testing**:
- Verify file uploads, API responses, and email notifications



AWS Notifications <no-reply@sns.amazonaws.com>
to me
2:21 PM (1 hour ago)

Request Submitted

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:851725198692:sns1:acbcb02e-7f8b-46da-a0fd-c1e326fbd193&Endpoint=reesasusan@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

- **Clean-Up**:
    - Remove resources to avoid unnecessary charges.

4. Challenges and Resolutions

- **CORS Errors**:
    - Resolved by configuring CORS policy for S3 bucket.
- **IAM Permission Issues**:
    - Addressed by assigning correct roles to Lambda functions.

5. Conclusion

This project highlights my ability to design and deploy serverless solutions using AWS. The implementation reflects a strong grasp of modern cloud architecture principles.