AT83.03 Cloud Computing

CLOUD-BASED DOCUMENT SUBMISSION SYSTEM WITH REAL-TIME NOTIFICATION

By : Tatiya S. 124875

TABLE OF CONTENT

01
INTRODUCTION

INTRODUCTION

- Cloud adoption is increasing for secure digital workflows
- Document submission systems benefit from flexibility and scalability
- Goal: Design a cost-effective, real-time, cloud-native solution
- AWS services used to create a serverless event-driven pipeline

OBJECTIVES

1. Build a fully serverless submission system using AWS
2. Use pre-signed URLs for secure document uploads
3. Trigger backend processing with Lambda upon file upload
4. Send real-time email notifications using Amazon SNS
5. Store metadata in DynamoDB
6. Create a dashboard using Amazon S3, Cognito, and HTML

02
PROJECT OVERVIEW

PROJECT OVERVIEW

01 SECURE DOCUMENT UPLOADS USING PRE-SIGNED URLS TO AMAZON S3

02 AUTOMATIC FILE PROCESSING VIA AWS LAMBDA

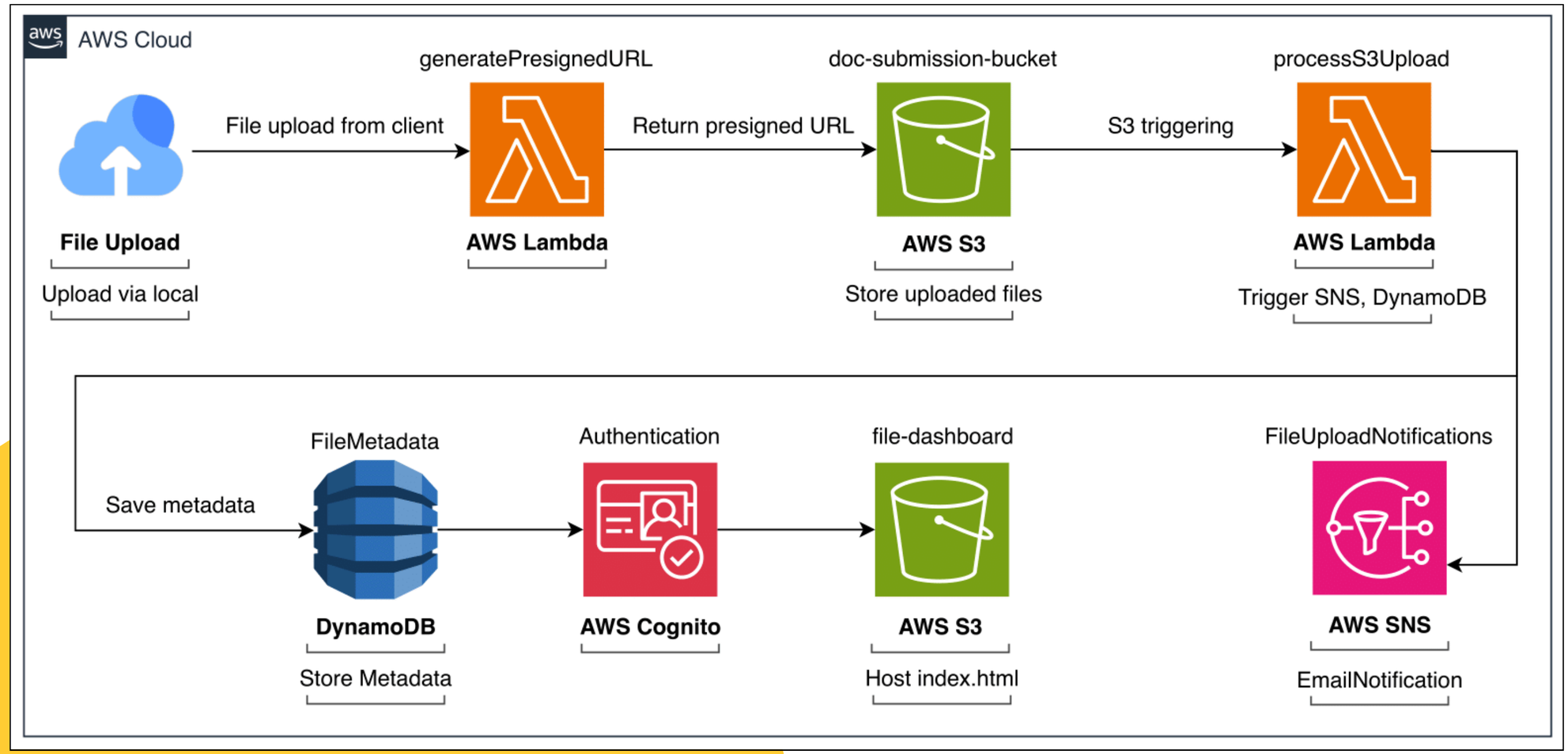
03 REAL-TIME NOTIFICATIONS USING AMAZON SNS

04 METADATA STORAGE IN AMAZON DYNAMODB

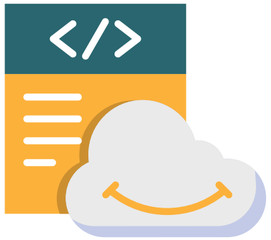
04 COFIG IAM ROLES FOR EACH SERVICE

05 WEB DASHBOARD HOSTED ON S3 WITH COGNITO AUTHENTICATION

SYSTEM ARCHITECTURE



AWS SERVICES



01

STORAGE, STATIC WEB

02

EVENT-BASED FUNCTIONS

03

REAL-TIME NOTIFICATION

04

METADATA STORAGE

05

AUTHEN - TICATION

06

IAM ROLES

03
METHODOLOGY

METHODOLOGY

01 CREATE S3 BUCKET FOR STORING FILES

02 CREATE LAMBDA FUNCTIONS,
CONFIGURE S3 EVENT TRIGGER

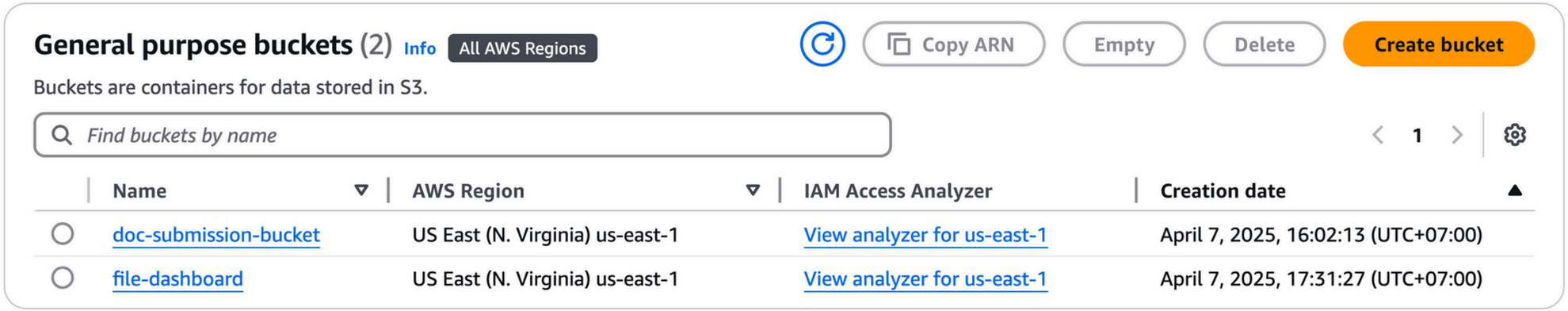
03 SET UP SNS TOPIC AND SUBSCRIPTION

04 CREATE DYNAMODB TABLE

05 BUILD AND HOST DASHBOARD ON S3

1. doc-submission-bucket

- primary storage location for uploaded files.

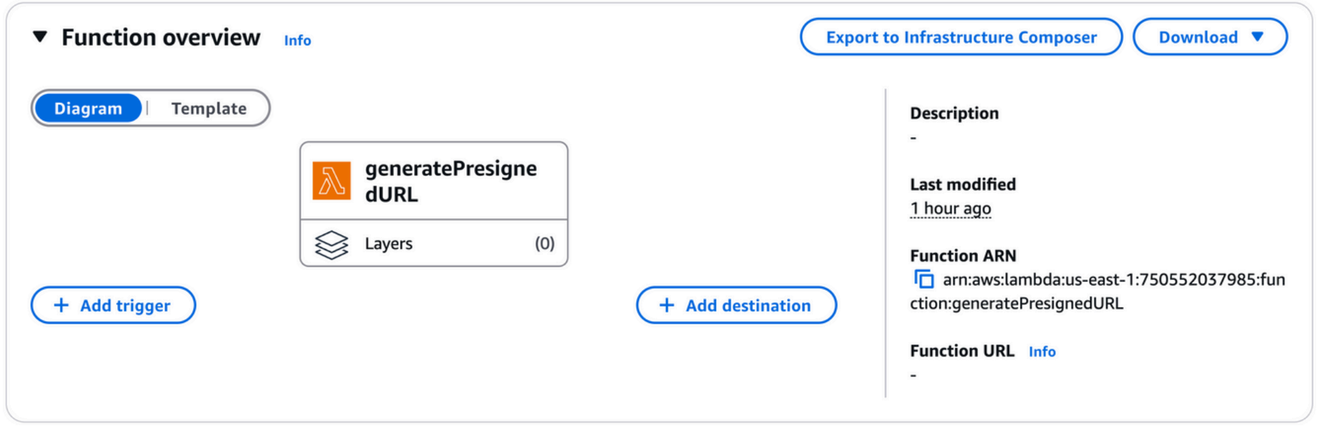


1. doc-submission-bucket

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadForWebsite",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::file-dashboard/*"
        }
    ]
}
```

2. generate PresignedURL

- Generates a pre-signed URL for secure file uploads to S3.
- Receives the filename and username from the query.
- A unique file key is created using the username and a UUID.
- Generates a PUT URL that expires in 7 days.
- Returns the upload URL and file key in a JSON response.




```
import json
import boto3
import os
import uuid
from datetime import datetime, timedelta

s3_client = boto3.client('s3')
BUCKET_NAME = 'doc-submission-bucket'

def lambda_handler(event, context):
 file_name = event['queryStringParameters']['filename']
 fake_user = event['queryStringParameters'].get('user', 'anonymous')

 key = f"{fake_user}/{uuid.uuid4()}_{file_name}"

 presigned_url = s3_client.generate_presigned_url(
 'put_object',
 Params={'Bucket': BUCKET_NAME, 'Key': key},
 ExpiresIn=604800 # 7 days in seconds
 )

 return {
 'statusCode': 200,
 'body': json.dumps({'upload_url': presigned_url, 'file_key': key}),
 'headers': {'Content-Type': 'application/json'}
 }
```

2. generate PresignedURL

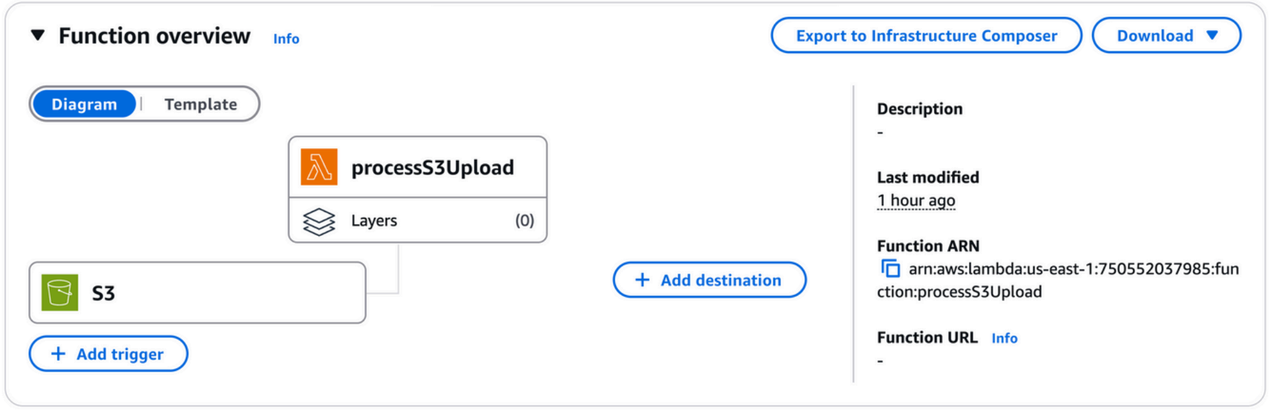
AllowUploadToDemoUserFolder

Copy JSON | Edit

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUploadToDemoUserFolder",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::doc-submission-bucket/demo-user/*"
        }
    ]
}
```

2. process S3Upload

- Triggered when a file is uploaded to Amazon S3.
- Extracts the bucket name, file key, file size, and uploader from the S3 event.
- Sends an email notification using Amazon SNS.
- Stores file metadata (file key, uploader, size, timestamp) in the DynamoDB table FileMetadata.
- Automate real-time notifications and metadata storage.




```
import json
import boto3
import urllib.parse
from datetime import datetime

sns_client = boto3.client('sns')
dynamodb = boto3.resource('dynamodb')

SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:750552037985:FileUploadNotifications'
DDB_TABLE = 'FileMetadata'

def lambda_handler(event, context):
 for record in event['Records']:
 bucket = record['s3']['bucket']['name']
 key = urllib.parse.unquote_plus(record['s3']['object']['key'])
 size = record['s3']['object']['size']
 timestamp = datetime.utcnow().isoformat()
 uploader = key.split('/')[0] # 'demo-user'

 # 1. Send SNS email
 message = f"A file was uploaded.\n\nUploader: {uploader}\nFile: {key}\nSize:
{size} bytes\nTime: {timestamp}"
 sns_client.publish(
 TopicArn=SNS_TOPIC_ARN,
 Subject="New File Uploaded",
 Message=message
 )

 # 2. Save to DynamoDB
 table = dynamodb.Table(DDB_TABLE)
 table.put_item(Item={
 'file_key': key,
 'uploader': uploader,
 'size': size,
 'timestamp': timestamp
 })

 return {'statusCode': 200}
```

2. processS3Upload

AllowPublishToSNS

Copy JSON | Edit

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-1:750552037985:FileUploadNotifications"
        }
    ]
}
```

S3UploadPermissions-SNS-DynamoDB

Copy JSON | Edit

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:033321897760:FileUploadNotifications"
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem"
            ],
            "Resource": "arn:aws:dynamodb:us-east-1:*:table/FileMetadata"
        }
    ]
}
```

3. SNS

- SNS topic for file uploads
- Email subscription
- Lambda sends formatted message with File name, Timestamp, Upload status
- Users receive instant confirmation of submission

Details

Name
FileUploadNotifications

Display name
-

ARN
arn:aws:sns:us-east-1:750552037985:FileUploadNotifications

Topic owner
750552037985

Type
Standard

Subscriptions (1)

Edit | Delete | Request confirmation | Confirm subscription | Create subscription

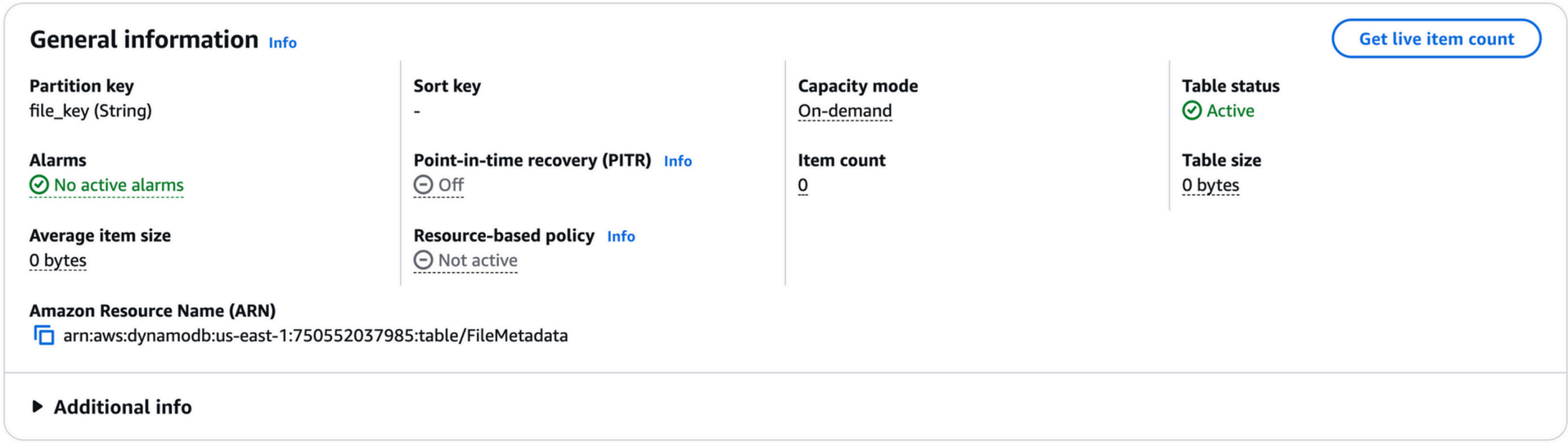
Search

1

ID	Endpoint	Status
977d7ea9-548a-4227-a2a4-9dbd752465da	swbt.tatiya@gmail.com	Confirmed

4. DynamoDB

- Table: FileMetadata
- Fields stored: file_key, uploader, timestamp
- Real-time insertion by Lambda
- Dashboard reads data from this table



4. DynamoDB

IdentityPoolRole

Copy JSON Edit

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:Scan"
            ],
            "Resource": "arn:aws:dynamodb:us-east-1:*:table/FileMetadata"
        }
    ]
}
```

5. Cognito

- Authentication
- Secure access to DynamoDB from frontend

Identity pool overview

Identity pool name
FileDashboardPool

Identity pool ID
us-east-1:de2c242e-6ade-4224-ad4d-d44e80445622

ARN
arn:aws:cognito-identity:us-east-1:750552037985:identitypool/us-east-1:de2c242e-6ade-4224-ad4d-d44e80445622

User access
Guest

Created time
April 7, 2025 at 17:09 GMT+7

Last updated time
April 7, 2025 at 17:09 GMT+7

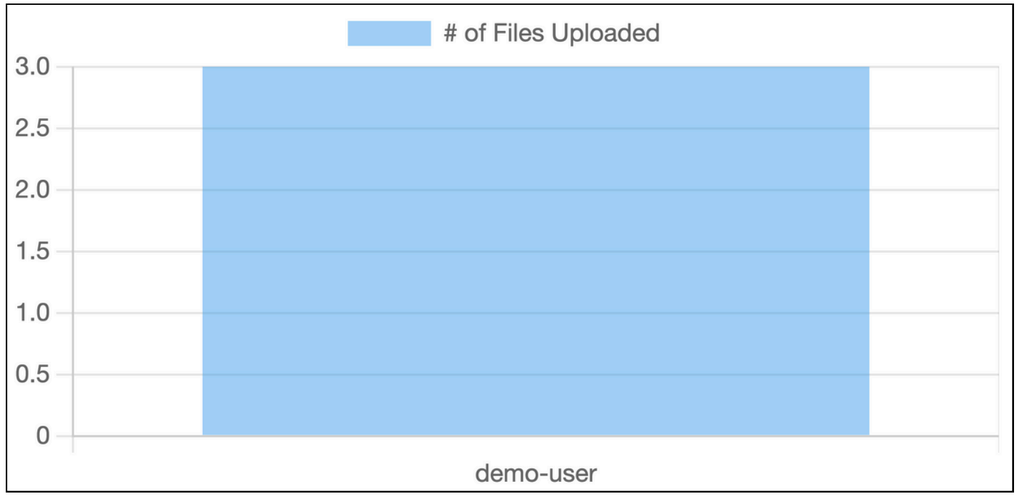
5. Dashboard

- Dashboard:
- HTML + JavaScript (with Chart.js)
- Hosted via S3 static hosting
- Dynamically shows table and bar chart of uploaded files

File Metadata Dashboard

Displaying file upload records from DynamoDB

File Key	Uploader	Timestamp
demo-user/f103e3eb-9ab8-4245-88f4-e93af5d963ac_sample.pdf	demo-user	2025-04-07T13:35:20.381565
demo-user/2f69e69a-b22b-4d0d-8ac3-8c2ad2ec037b_sample.pdf	demo-user	2025-04-07T09:51:47.649537
demo-user/test-assignment.pdf	demo-user	2025-04-07T09:50:53.140253



04
E2E TESTING

E2E TESTING

01 UPLOAD FILE WITH PRE-SIGNED URL 

02 FILES STORED IN S3 BUCKET 

03 SNS EMAIL NOTIFICATIONS 

04 FILES STORED IN DYNAMODB 

05 DASHBOARD SHOWS UPLOADED FILE IN REAL TIME 

EXPECTED OUTPUT

01 S3 RECEIVES UPLOADED FILES

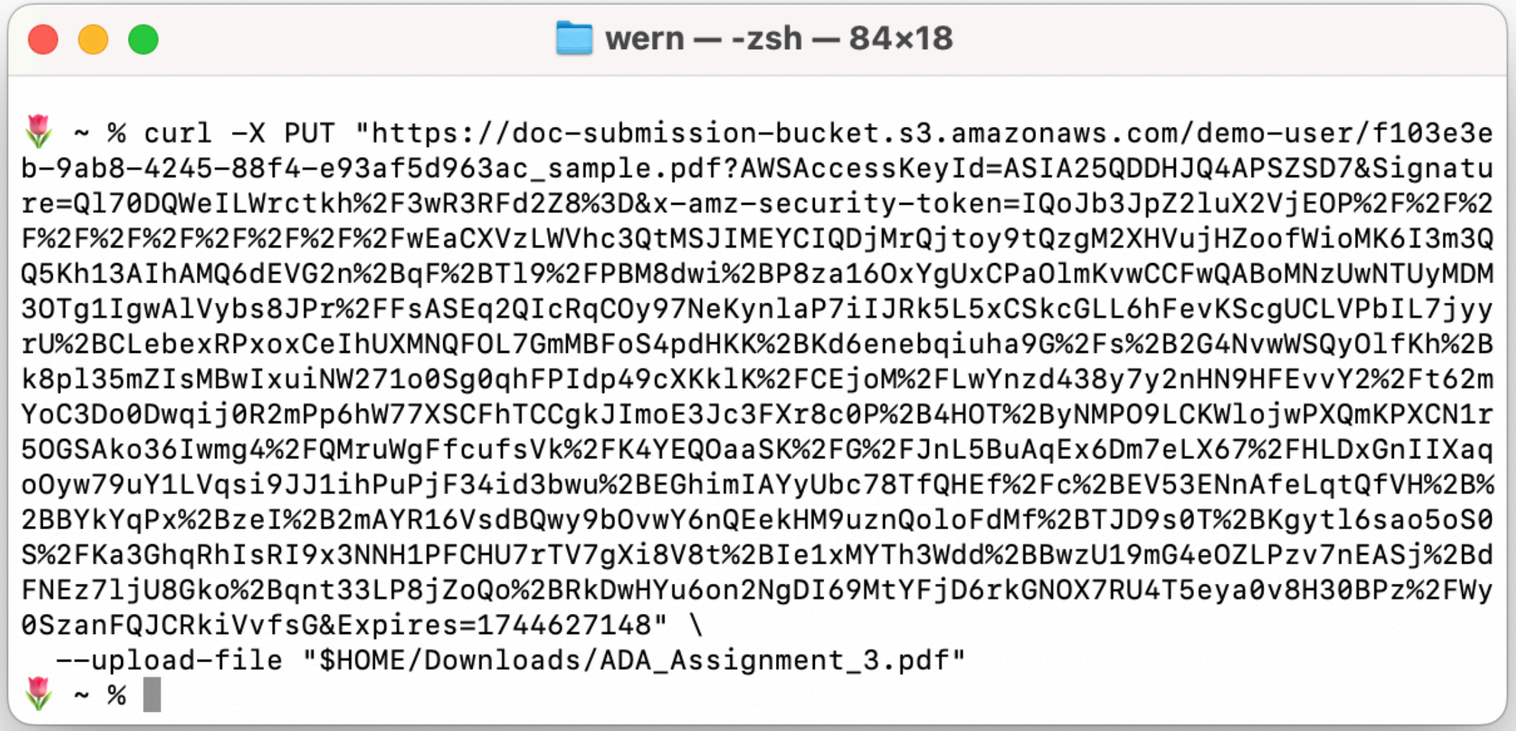
02 SNS SENDS AN EMAIL UPON SUCCESSFUL UPLOAD

03 DYNAMODB LOGS METADATA

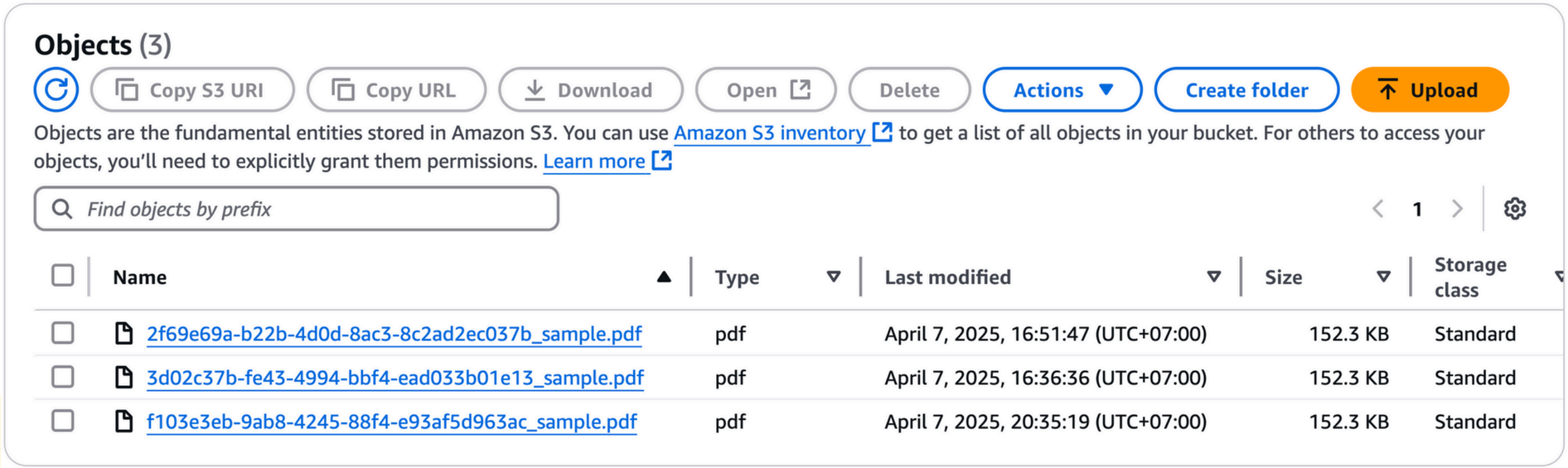
04 DASHBOARD DISPLAYS REAL-TIME FILE INFO

05 CLOUDWATCH TRACKS ALL ACTIVITIES

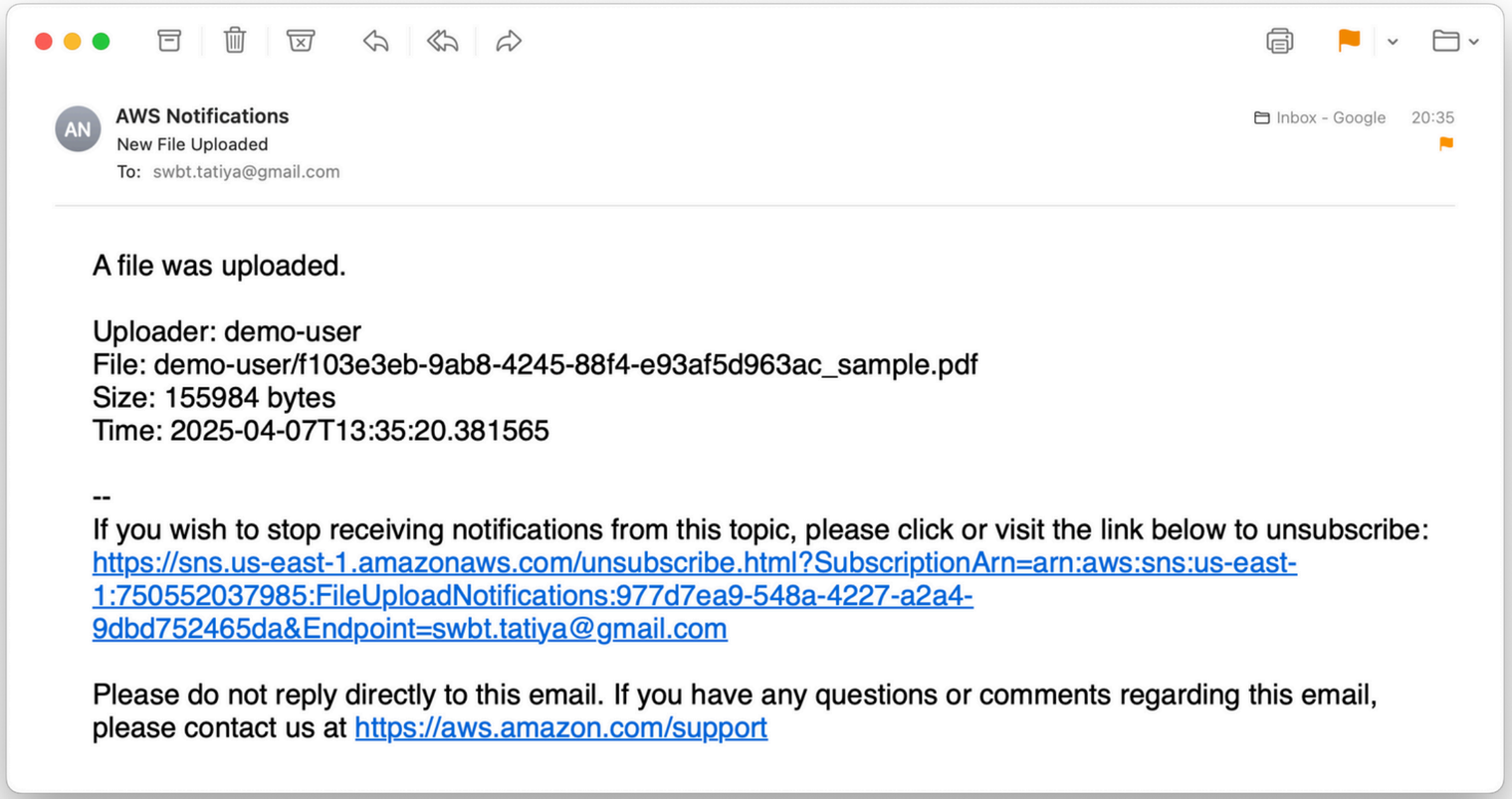
1. UPLOAD FILE



2. FILES STORED IN S3 BUCKET



3. SNS EMAIL NOTIFICATION



AN **AWS Notifications**
New File Uploaded
To: swbt.tatiya@gmail.com

Inbox - Google 20:35

A file was uploaded.

Uploader: demo-user
File: demo-user/f103e3eb-9ab8-4245-88f4-e93af5d963ac_sample.pdf
Size: 155984 bytes
Time: 2025-04-07T13:35:20.381565

--
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-1:750552037985:FileUploadNotifications:977d7ea9-548a-4227-a2a4-9dbd752465da&Endpoint=swbt.tatiya@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

4. FILES STORED IN DynamoDB

Items returned (3)

Actions | Create item

1

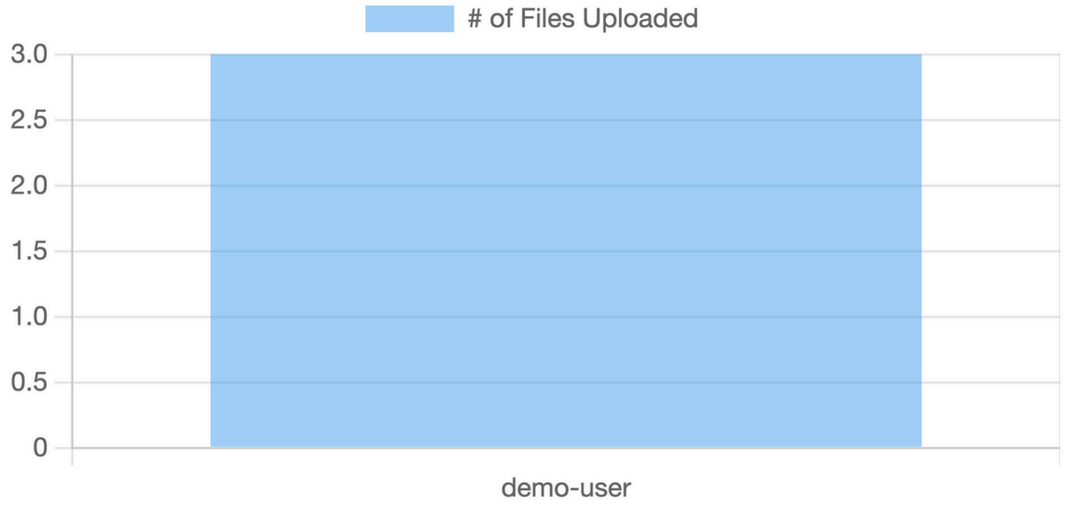
file_key *(String)*	size	timestamp	uploader
demo-user/f103e...	155984	2025-04-07T20:16:28.963457	demo-user
demo-user/2f69e...	155984	2025-04-07T09:51:47.649537	demo-user
demo-user/test-a...	123456	2025-04-07T09:50:53.140253	demo-user

5. DASHBOARD

File Metadata Dashboard

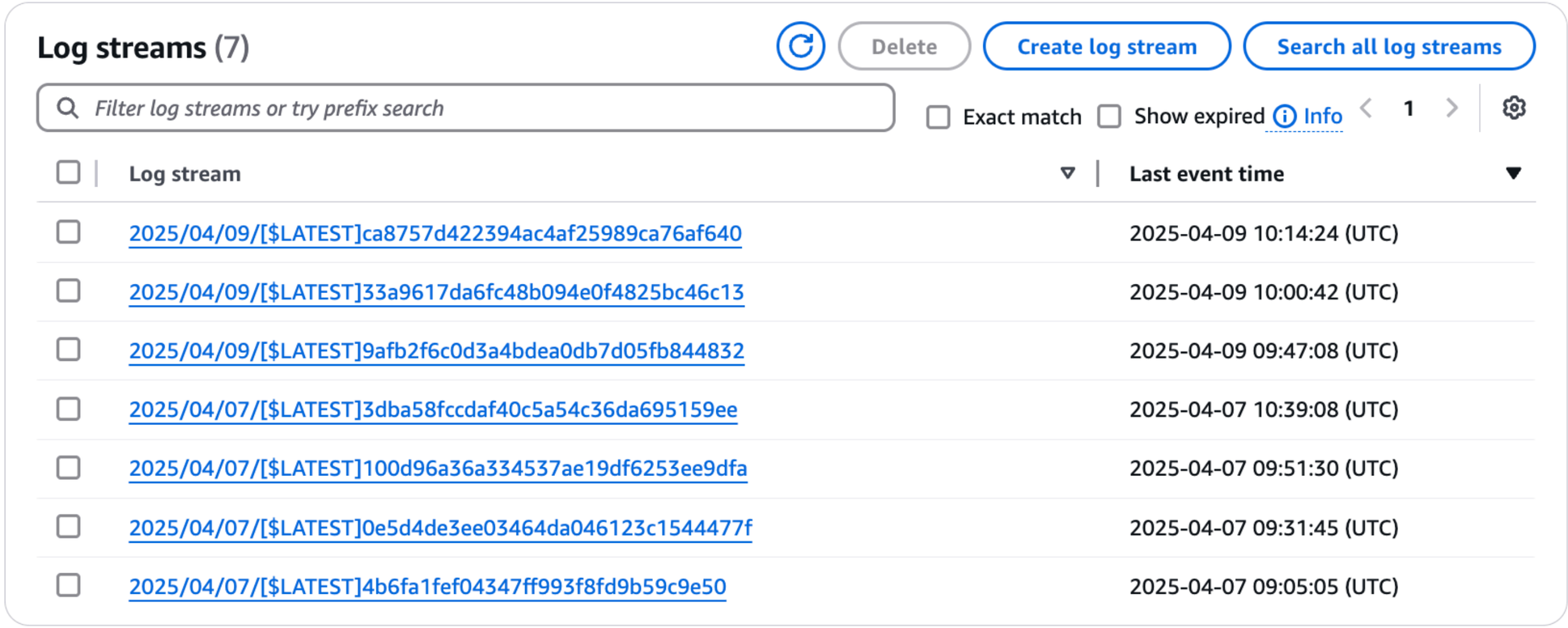
Displaying file upload records from DynamoDB

File Key	Uploader	Timestamp
demo-user/f103e3eb-9ab8-4245-88f4-e93af5d963ac_sample.pdf	demo-user	2025-04-07T13:35:20.381565
demo-user/2f69e69a-b22b-4d0d-8ac3-8c2ad2ec037b_sample.pdf	demo-user	2025-04-07T09:51:47.649537
demo-user/test-assignment.pdf	demo-user	2025-04-07T09:50:53.140253



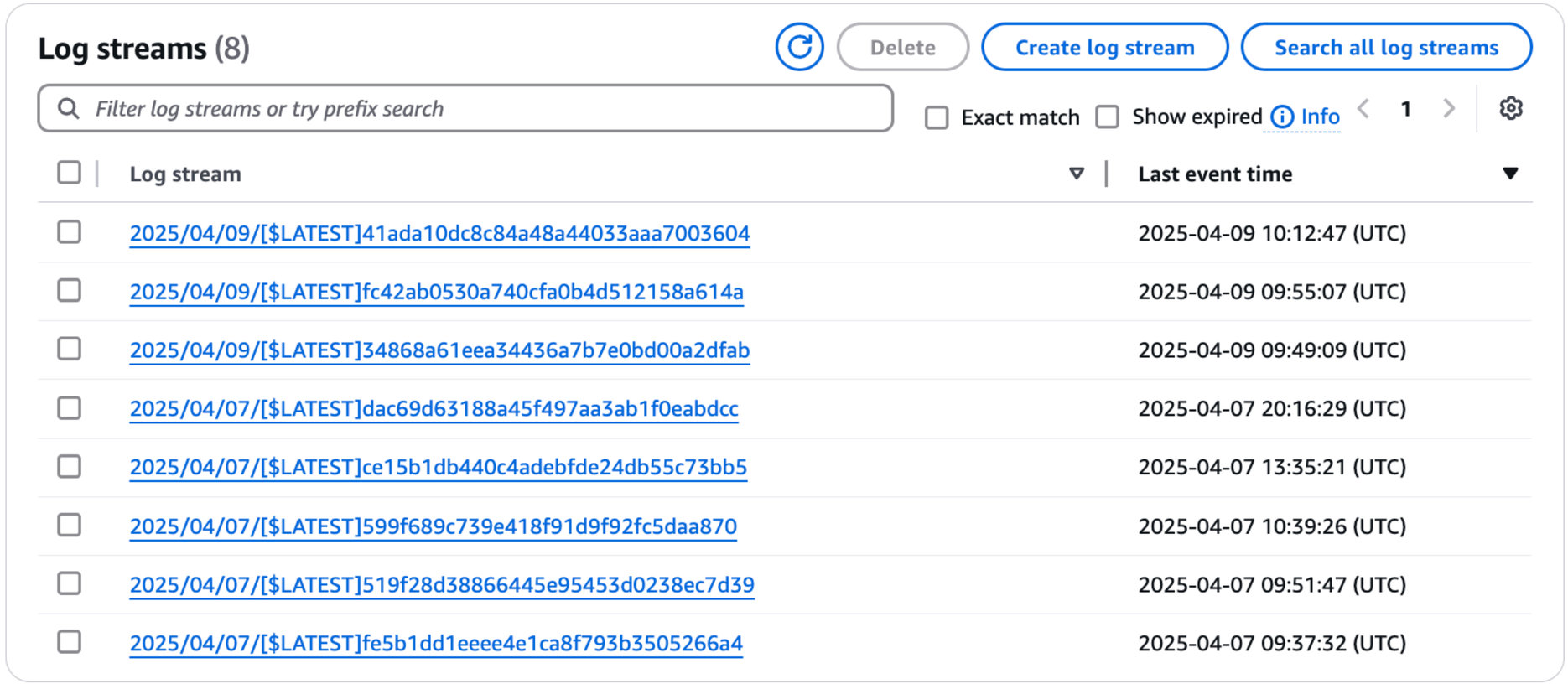
6. CLOUDWATCH

Log streams - generatePresignedURL

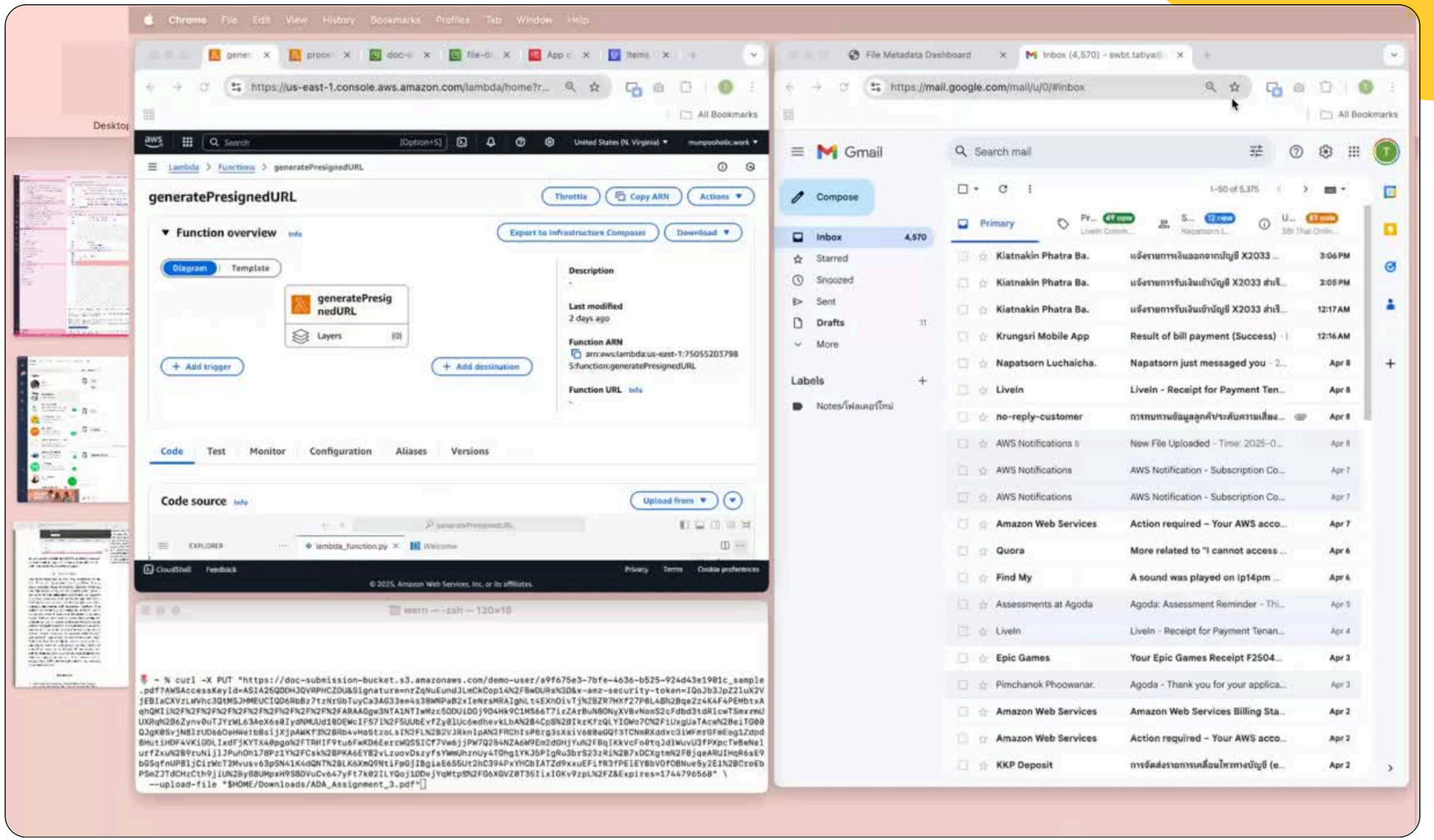


6. CLOUDWATCH

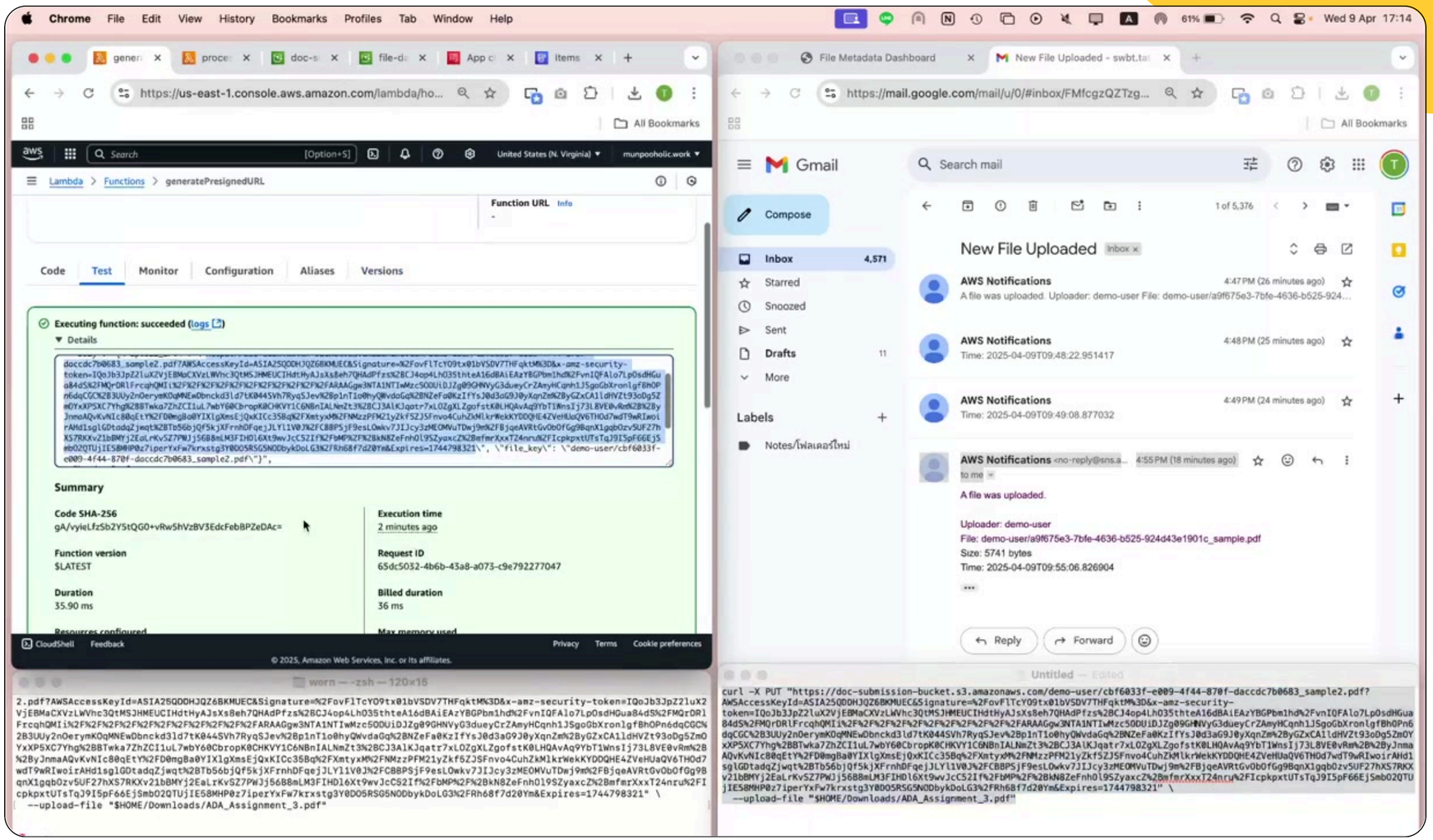
Log streams - processS3Upload



DEMO - PART1


generatePresignedURL
Function overview
Code
Test
Monitor
Configuration
Aliases
Versions
Code source
Gmail
Compose
Inbox
Starred
Snoozed
Sent
Drafts
More
Labels
Primary

DEMO - PART2


Executing function: succeeded (logs)
Summary
Code SHA-256
Execution time
Function version
Request ID
Duration
Billed duration
AWS Notifications
New File Uploaded
A file was uploaded.
curl -X PUT "https://doc-submission-bucket.s3.amazonaws.com/demo-user/cbf6033f-e009-4f44-870f-daccdc7b0683_sample2.pdf?
--upload-file "$HOME/Downloads/ADA_Assignment_3.pdf"

05
CHALLENGES

SANDBOX LIMITATIONS

1. Cognito Domain/Hosted UI
2. SNS Email Notifications
3. DynamoDB Write Limitations
4. Custom IAM Role Restrictions

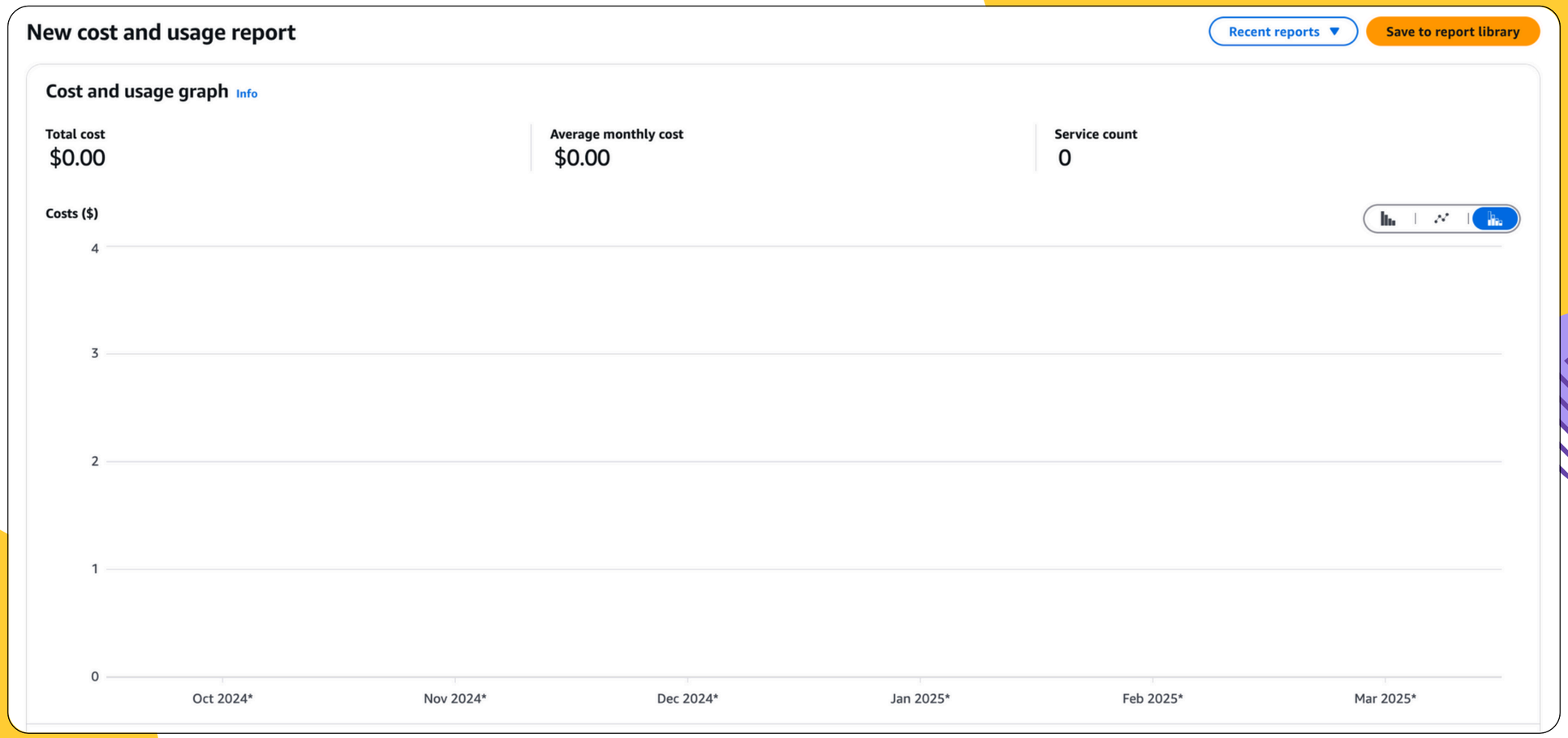
Solution : Move to Free Tier

06
CONCLUSION

AWS WELL-ARCHITECTED FRAMEWORK

1. Operational Excellence: Automated workflows using Lambda and CloudWatch; S3 triggers eliminate manual polling.
2. Security: Secure access via Amazon Cognito and pre-signed URLs; IAM follows least privilege (limited in AWS Academy).
3. Reliability: High durability via S3 (11 9’s); retry-enabled Lambda functions; fault-tolerant DynamoDB.
4. Performance Efficiency: Serverless, event-driven design scales automatically; pre-signed URLs reduce auth overhead.
5. Cost Optimization: Pay-per-use model; avoided EC2 by using S3 static hosting; all components fit within AWS Free Tier for cost-effective deployment.

COST-OPTMIZATION



FUTURE WORK

- Enhance authentication UI & File uploads
- Implement role-based access (admin / user)
- Add metadata fields like tags, file type
- Enable analytics with Amazon QuickSight or OpenSearch.
- Support file versioning with S3 to track document updates.
- Integrate AI services like Textract or Comprehend for content extraction.

CONCLUSION

- Successfully built a fully serverless document submission system
- Secure file upload with pre-signed URLs
- Event-based automation with Lambda
- Real-time dashboard for monitoring submissions
- No VPC or inbound/outbound rules needed — all components use fully managed AWS services with public endpoints

THANK YOU