



AT83.03 Cloud Computing

CLOUD-BASED DOCUMENT SUBMISSION SYSTEM WITH REAL-TIME NOTIFICATION

By : Tatiya S. 124875



TABLE OF CONTENT

- 01** INTRODUCTION
- 02** PROJECT OVERVIEW
- 03** METHODOLOGY
- 04** E2E TESTING
- 05** CHALLENGES AND LIMITATIONS
- 06** CONCLUSION





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





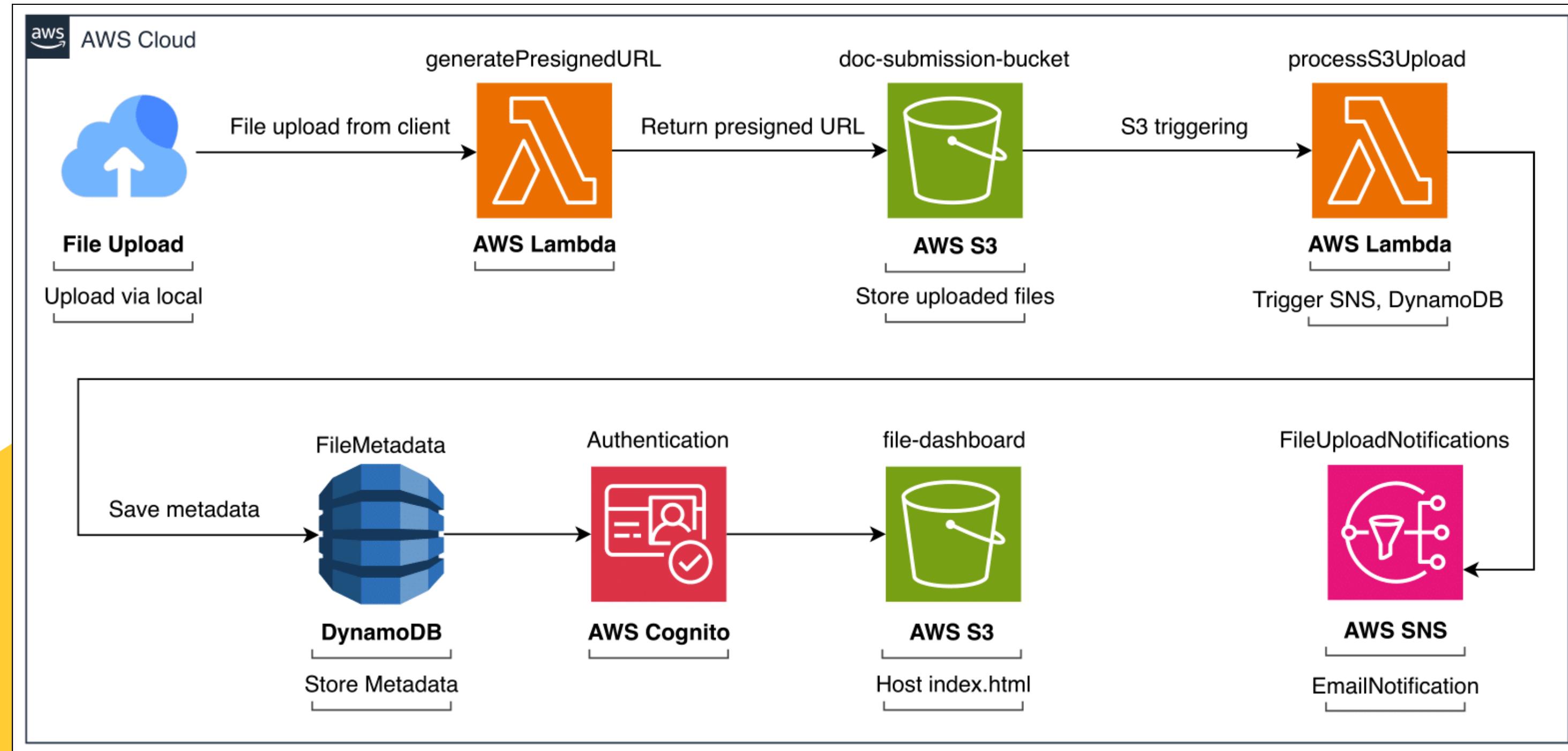
02

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

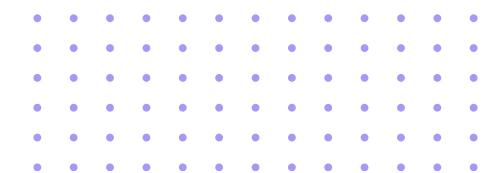


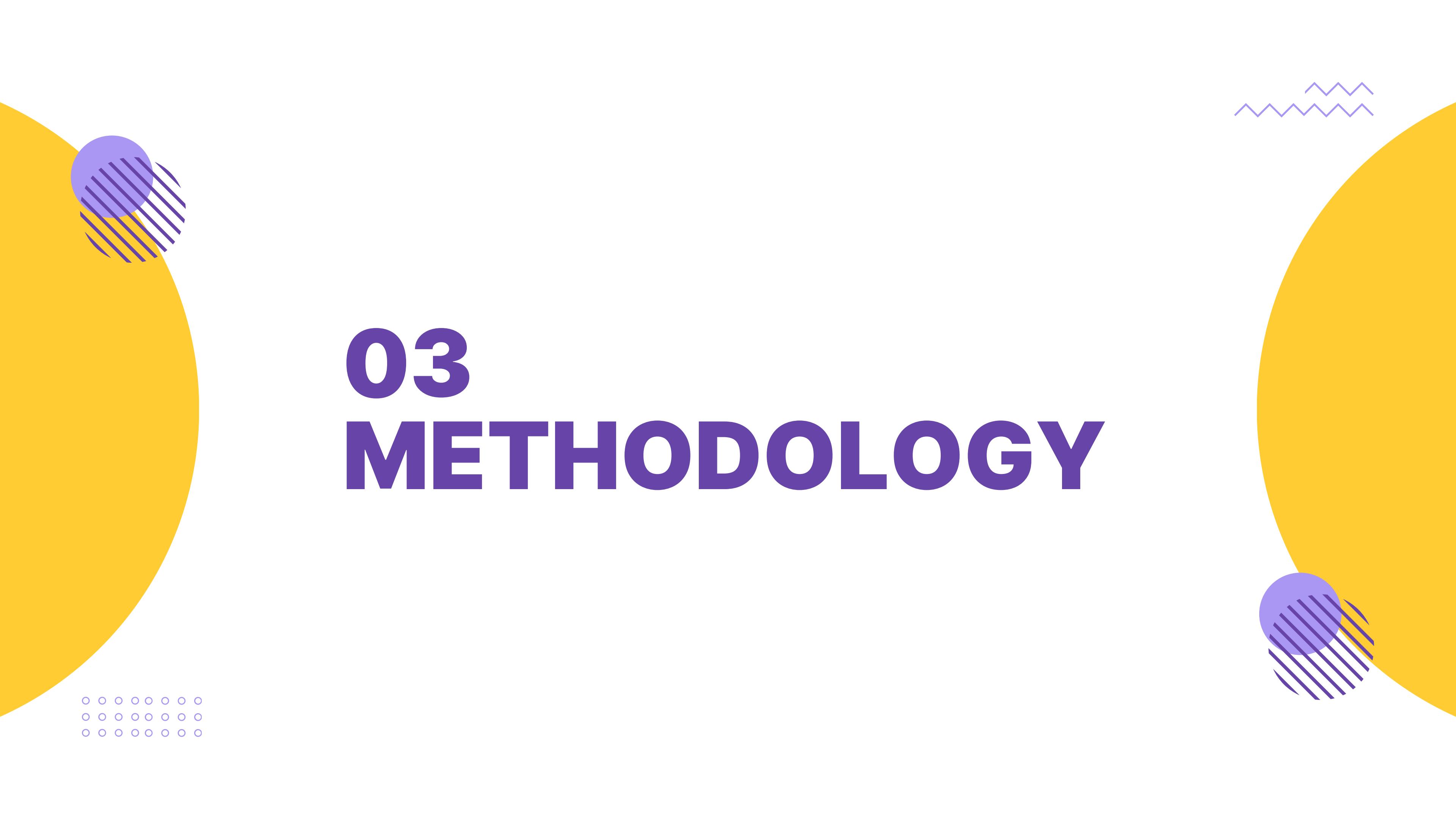
**AUTHEN -
TICATION**

06



**IAM
ROLES**



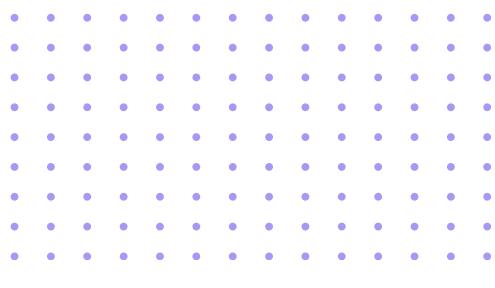


03

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.

General purpose buckets (2) [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

Find buckets by name

[C](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Name	AWS Region	IAM Access Analyzer	Creation date
doc-submission-bucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	April 7, 2025, 16:02:13 (UTC+07:00)
file-dashboard	US East (N. Virginia) us-east-1	View analyzer for us-east-1	April 7, 2025, 17:31:27 (UTC+07:00)

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.

Function overview [Info](#)

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

generatePresigne dURL

 Layers (0)

+ Add trigger + Add destination

Export to Infrastructure Composer Download ▾

Description -

Last modified 1 hour ago

Function ARN [arn:aws:lambda:us-east-1:750552037985:function:generatePresignedURL](#)

Function URL [Info](#)



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

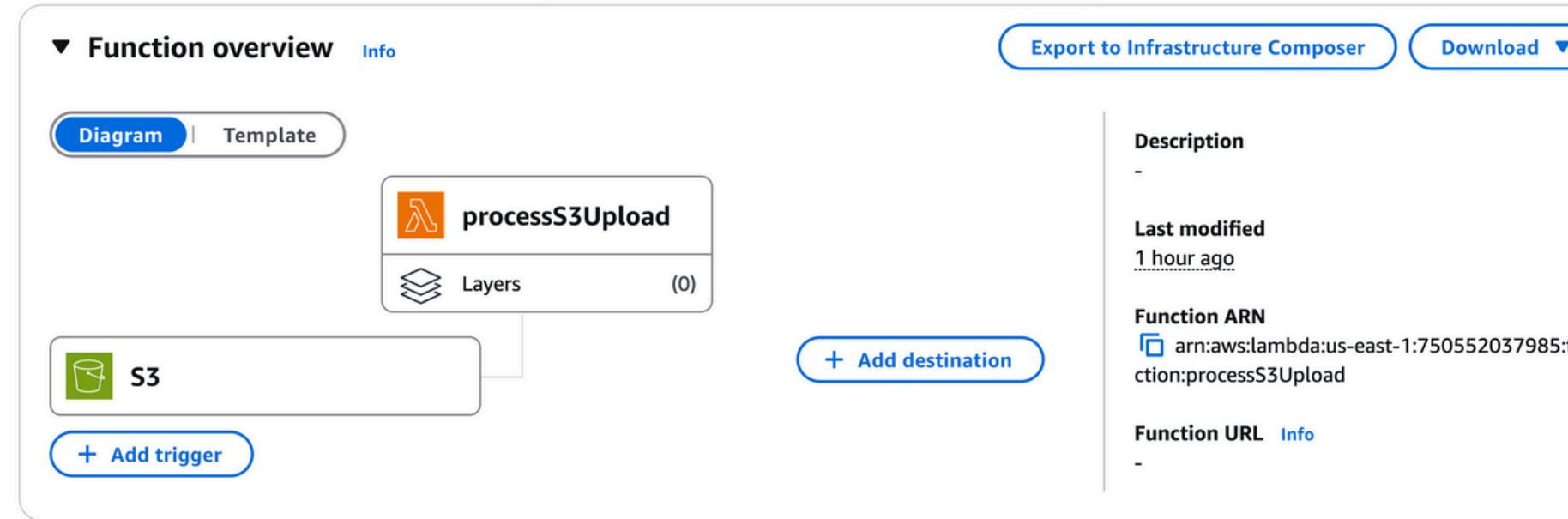
[Copy JSON](#) [Edit](#)

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Sid": "AllowUploadToDemoUserFolder",  
6       "Effect": "Allow",  
7       "Action": "s3:PutObject",  
8       "Resource": "arn:aws:s3:::doc-submission-bucket/demo-user/*"  
9     }  
10    ]  
11 }
```

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

```
1 [{}  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": "sns:Publish",  
7       "Resource": "arn:aws:sns:us-east-1:750552037985:FileUploadNotifications"  
8     }  
9   ]  
10 ]
```

S3UploadPermissions-SNS-DynamoDB

```
1 [{}  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": [  
7         "sns:Publish"  
8       ],  
9       "Resource": "arn:aws:sns:us-east-1:033321897760:FileUploadNotifications"  
10     },  
11     {  
12       "Effect": "Allow",  
13       "Action": [  
14         "dynamodb:PutItem"  
15       ],  
16       "Resource": "arn:aws:dynamodb:us-east-1:*:table/FileMetadata"  
17     }  
18   ]  
19 ]
```

[Copy JSON](#) [Edit](#)

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

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



General information [Info](#)

Partition key
file_key (String)

Sort key
-

Alarms
[No active alarms](#)

Average item size
0 bytes

Point-in-time recovery (PITR) [Info](#)
[Off](#)

Resource-based policy [Info](#)
[Not active](#)

Capacity mode
On-demand

Item count
0

Table status
[Active](#)

Table size
0 bytes

[Get live item count](#)

[Amazon Resource Name \(ARN\)](#)
[arn:aws:dynamodb:us-east-1:750552037985:table/FileMetadata](#)

▶ Additional info

4. DynamoDB



IdentityPoolRole

[Copy JSON](#)[Edit](#)

```
1 {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Action": [  
7                 "dynamodb:Scan"  
8             ],  
9             "Resource": "arn:aws:dynamodb:us-east-1:*:table/FileMetadata"  
10        }  
11    ]  
12 }
```

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

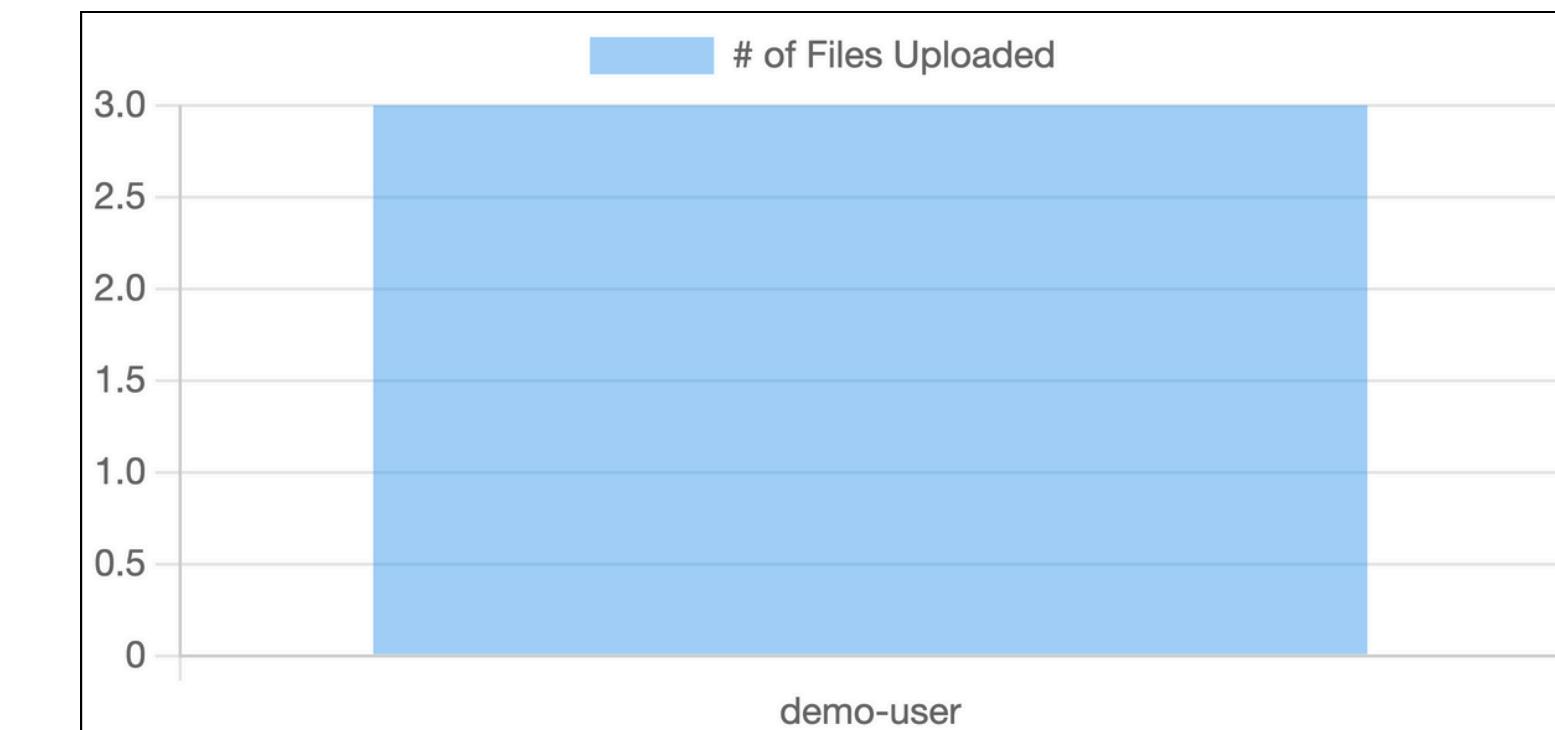
- Dashboard:
- HTML + JavaScript (with Chart.js)
- Hosted via S3 static hosting
- Dynamically shows table and bar chart of uploaded files



 **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

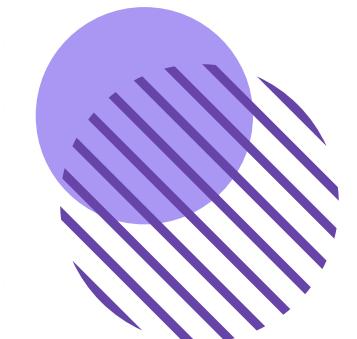


EXPECTED OUTPUT

- 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

```
curl -X PUT "https://doc-submission-bucket.s3.amazonaws.com/demo-user/f103e3b-9ab8-4245-88f4-e93af5d963ac_sample.pdf?AWSAccessKeyId=ASIA25QDDHJQ4APSZSD7&Signature=Q170DQWeILWrctkh%2F3wR3RFd2Z8%3D&x-amz-security-token=IQoJb3JpZ2luX2VjEOP%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJIMEYCIQDjMrQjtoy9tQzgM2XHVujHZoofWioMK6I3m3Q5Kh13AIhAMQ6dEVG2n%2BqF%2BT19%2FPBM8dwi%2BP8za160xYgUxCPaOlmAkwCCFwQABoMNzUwNTUyMDM30Tg1IgwAlVybs8JPr%2FFsASEq2QIcRqCOy97NeKynlaP7iIJRk5L5xCSkcGLL6hFevKScgUCLVPbIL7jyrU%2BCLebexRPxoxCeIhUXMNQFOL7GmMBFoS4pdHKK%2BKd6enebqiuh9G%2Fs%2B2G4NvwWSQyO1fKh%2Bk8p135mZIsMBwIxuiNW271o0Sg0qhFPIdp49cXKk1K%2FCEjoM%2FLwYnzd438y7y2nHN9HFEvvY2%2Ft62YoC3Do0Dwqij0R2mPp6hW77XSCFhTCCgkJImoE3Jc3FXr8c0P%2B4HOT%2ByNMPO9LCKWlojwPXQmKPXCN150GSAko36Iwmg4%2FQMRuWgFfcufsVk%2FK4YEQ0aaSK%2FG%2FJnL5BuAqEx6Dm7eLX67%2FHLDxGnIIXao0Oyw79uY1LVqsi9JJ1ihPuPjF34id3bwu%2BEGhimIAYyUbc78TfQHEf%2Fc%2BEV53ENnAfeLqtQfVH%2B2BBYkYqPx%2BzeI%2B2mAYR16VsdBQwy9b0vwY6nQEekHM9uznQoloFdMf%2BTJD9s0T%2BKgytl6sao5oS%2FKa3GhqRhIsRI9x3NNH1PFCHU7rTV7gXi8V8t%2BIe1xMYTh3Wdd%2BBwzU19mG4e0ZLPzv7nEASj%2BFNEz71jU8Gko%2Bqnt33LP8jZoQo%2BRkDwHYu6on2NgDI69MtYFjD6rkGNOX7RU4T5eya0v8H30BPz%2FW0SzanzFQJCRkiVvfsG&Expires=1744627148" \
--upload-file "$HOME/Downloads/ADA_Assignment_3.pdf"
```



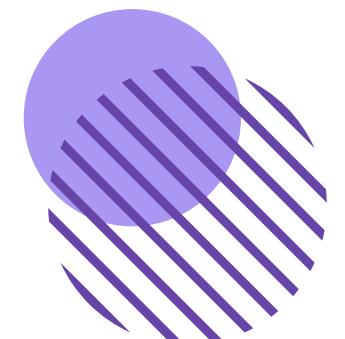
2. FILES STORED IN S3 BUCKET

Objects (3)

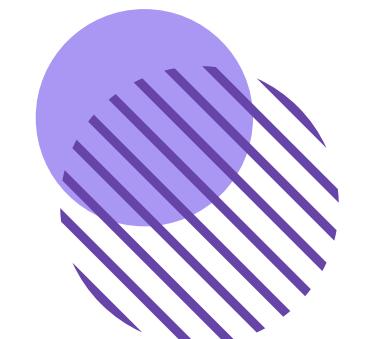
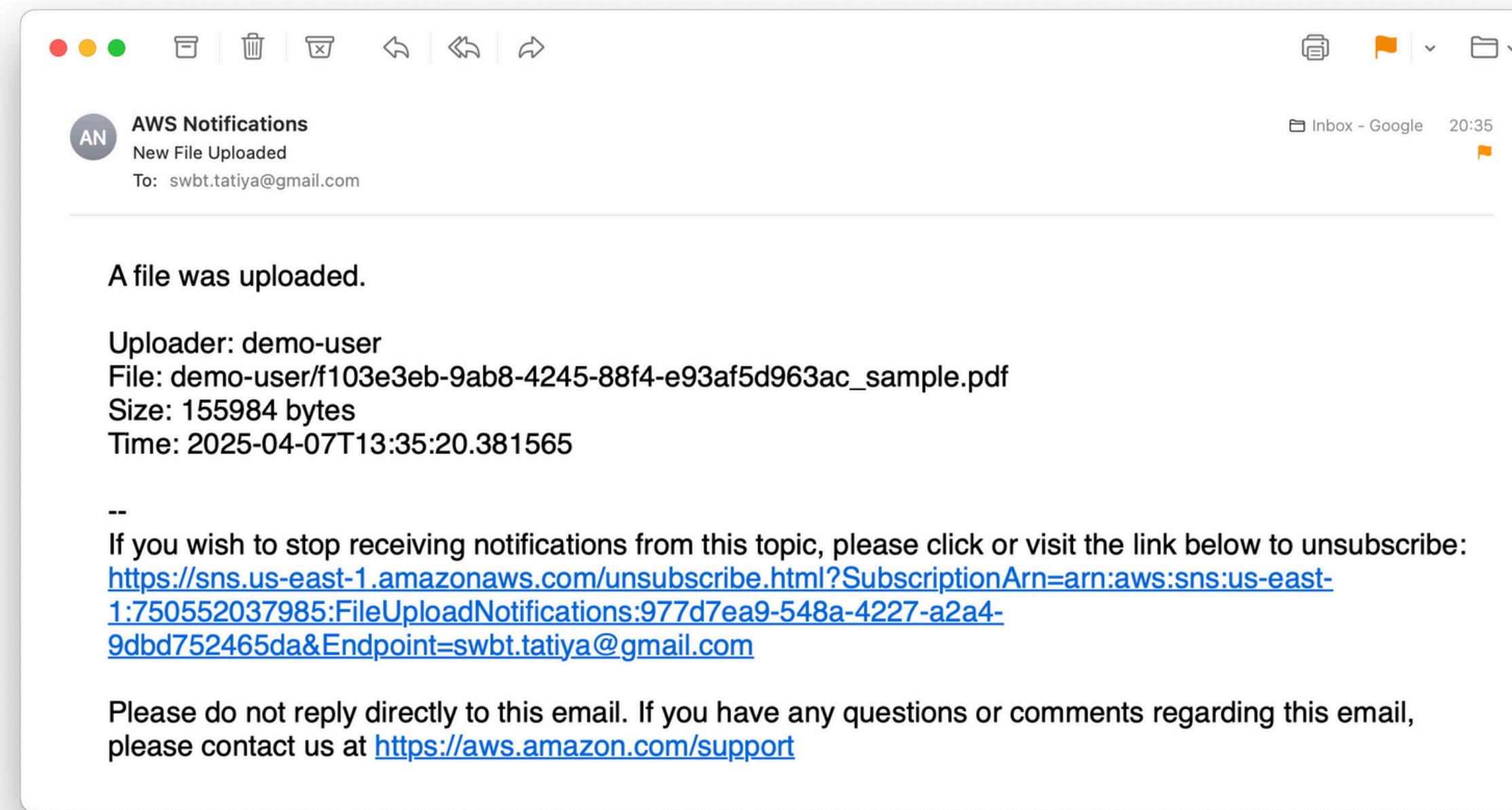
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	2f69e69a-b22b-4d0d-8ac3-8c2ad2ec037b_sample.pdf	pdf	April 7, 2025, 16:51:47 (UTC+07:00)	152.3 KB	Standard
<input type="checkbox"/>	3d02c37b-fe43-4994-bbf4-ead033b01e13_sample.pdf	pdf	April 7, 2025, 16:36:36 (UTC+07:00)	152.3 KB	Standard
<input type="checkbox"/>	f103e3eb-9ab8-4245-88f4-e93af5d963ac_sample.pdf	pdf	April 7, 2025, 20:35:19 (UTC+07:00)	152.3 KB	Standard



3. SNS EMAIL NOTIFICATION



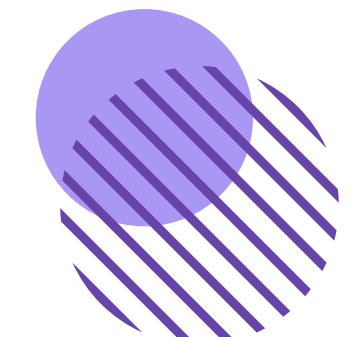
4. FILES STORED IN DynamoDB

Items returned (3)

(C)Actions ▾Create item

<1>|⚙️☒

<input type="checkbox"/>	file_key (String) ▾	size ▾	timestamp ▾	uploader ▾
<input type="checkbox"/>	demo-user/f103e...	155984	2025-04-07T20:16:28.963457	demo-user
<input type="checkbox"/>	demo-user/2f69e...	155984	2025-04-07T09:51:47.649537	demo-user
<input type="checkbox"/>	demo-user/test-a...	123456	2025-04-07T09:50:53.140253	demo-user

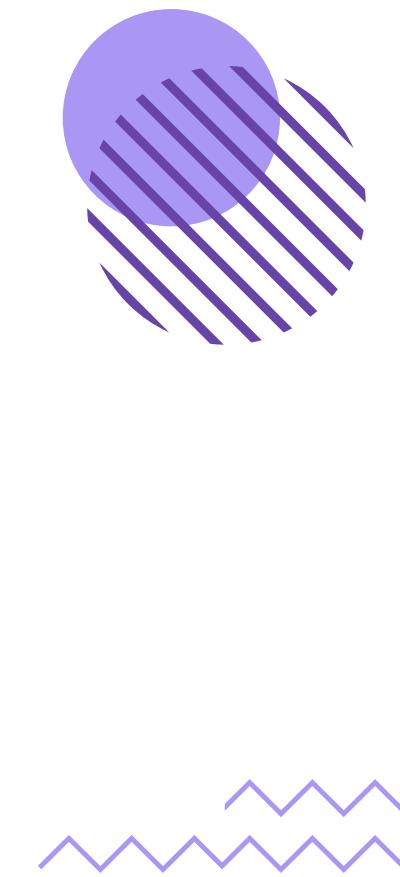
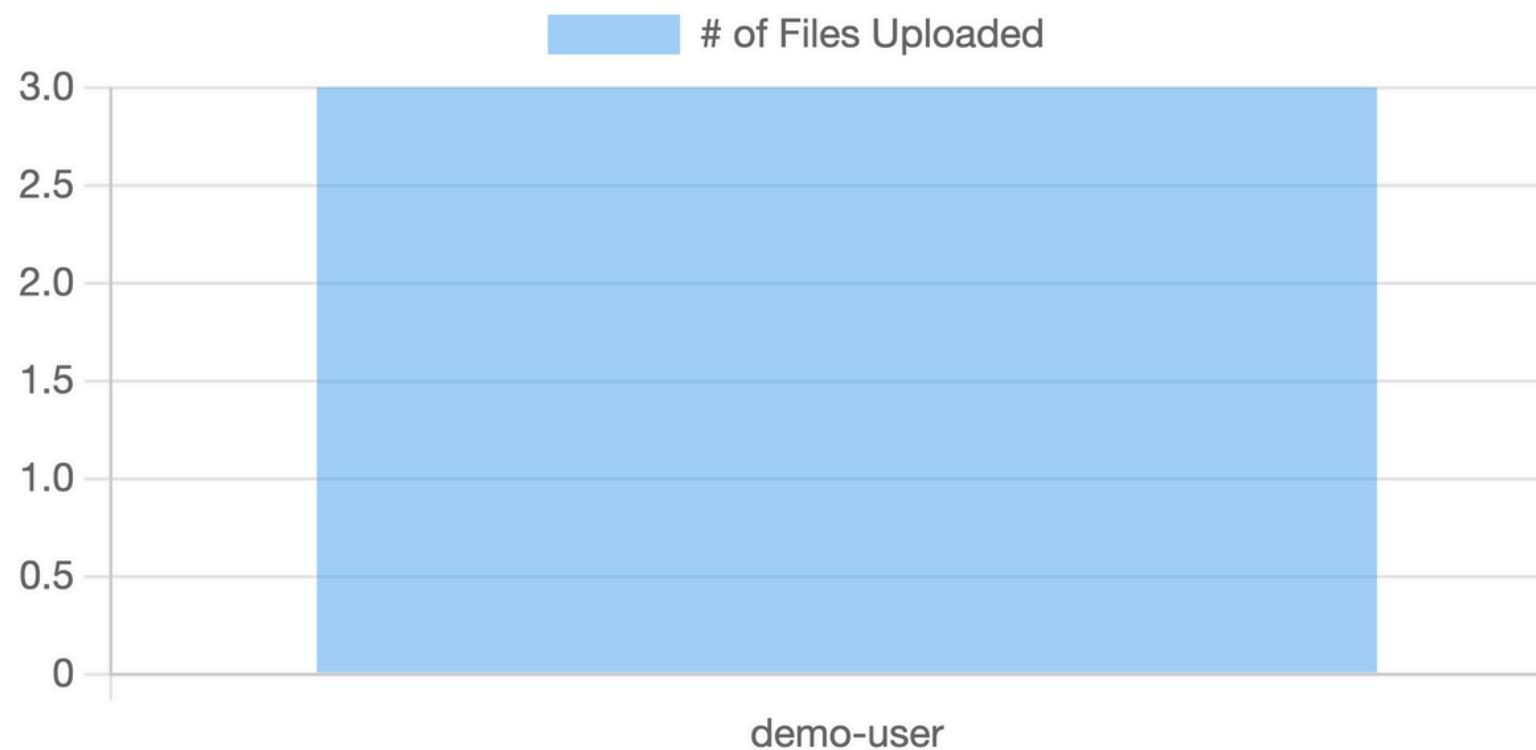


5. DASHBOARD

File Metadata Dashboard

Displaying file upload records from DynamoDB

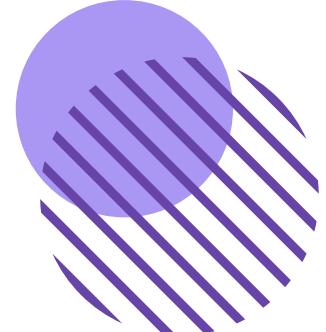
File Key	Uploader	Timestamp
demo-user/f103e3eb-9ab8-4245-88f4-e93af5d963ac_sample.pdf	demo-user	2025-04-07T13:35:20.381565
demo-user/2f69e69a-b22b-4d0d-8ac3-8c2ad2ec037b_sample.pdf	demo-user	2025-04-07T09:51:47.649537
demo-user/test-assignment.pdf	demo-user	2025-04-07T09:50:53.140253



6. CLOUDWATCH

Log streams - generatePresignedURL

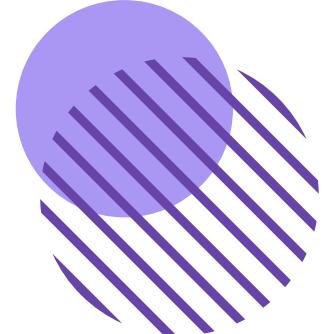
Log streams (7)		 Delete	Create log stream	Search all log streams
<input type="text"/> Filter log streams or try prefix search		<input type="checkbox"/> Exact match	<input type="checkbox"/> Show expired  Info	< 1 > 
<input type="checkbox"/> Log stream	Last event time			
<input type="checkbox"/> 2025/04/09/[\$LATEST]ca8757d422394ac4af25989ca76af640	2025-04-09 10:14:24 (UTC)			
<input type="checkbox"/> 2025/04/09/[\$LATEST]33a9617da6fc48b094e0f4825bc46c13	2025-04-09 10:00:42 (UTC)			
<input type="checkbox"/> 2025/04/09/[\$LATEST]9afb2f6c0d3a4bdea0db7d05fb844832	2025-04-09 09:47:08 (UTC)			
<input type="checkbox"/> 2025/04/07/[\$LATEST]3dba58fccdaf40c5a54c36da695159ee	2025-04-07 10:39:08 (UTC)			
<input type="checkbox"/> 2025/04/07/[\$LATEST]100d96a36a334537ae19df6253ee9dfa	2025-04-07 09:51:30 (UTC)			
<input type="checkbox"/> 2025/04/07/[\$LATEST]0e5d4de3ee03464da046123c1544477f	2025-04-07 09:31:45 (UTC)			
<input type="checkbox"/> 2025/04/07/[\$LATEST]4b6fa1fef04347ff993f8fd9b59c9e50	2025-04-07 09:05:05 (UTC)			



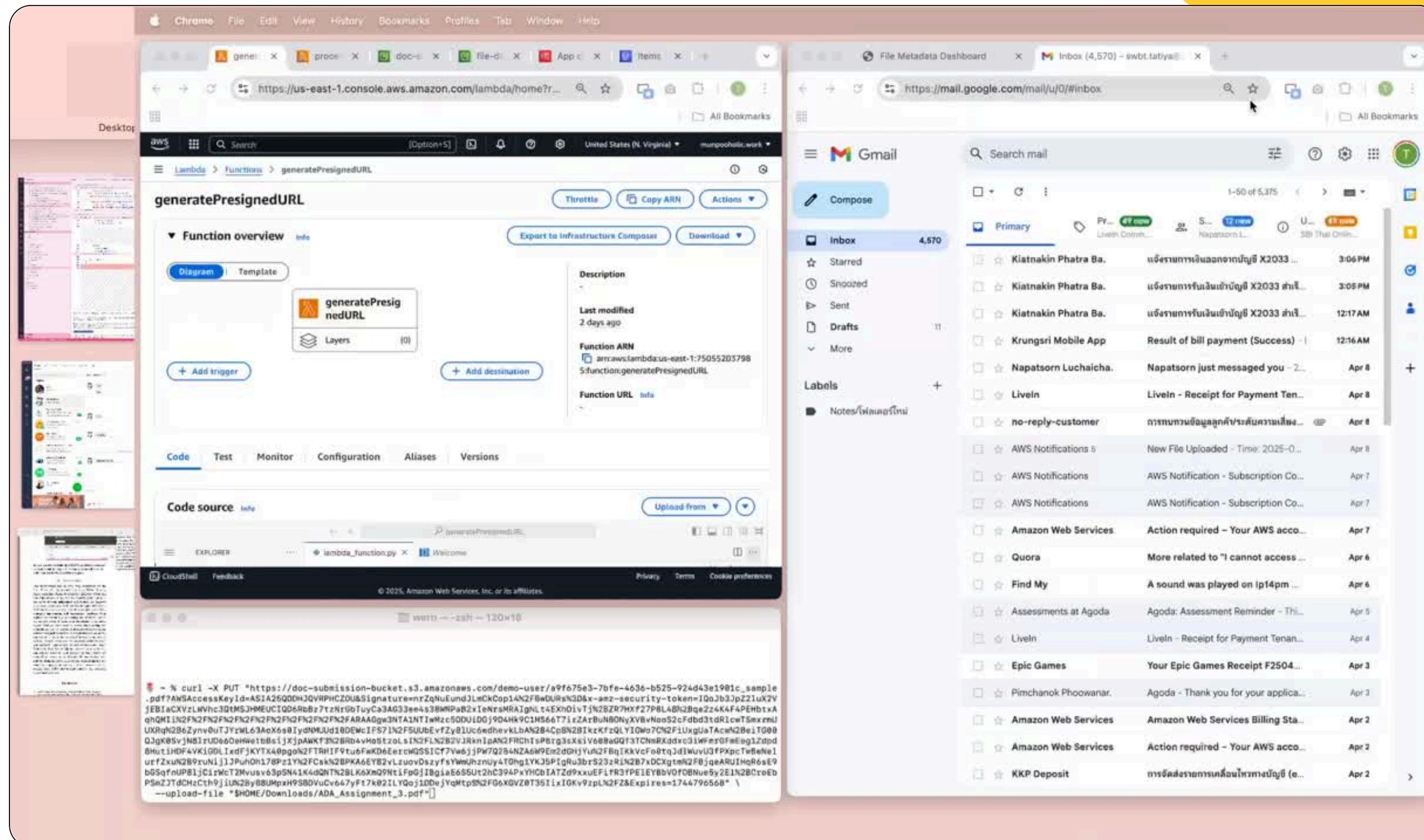
6. CLOUDWATCH

Log streams - processS3Upload

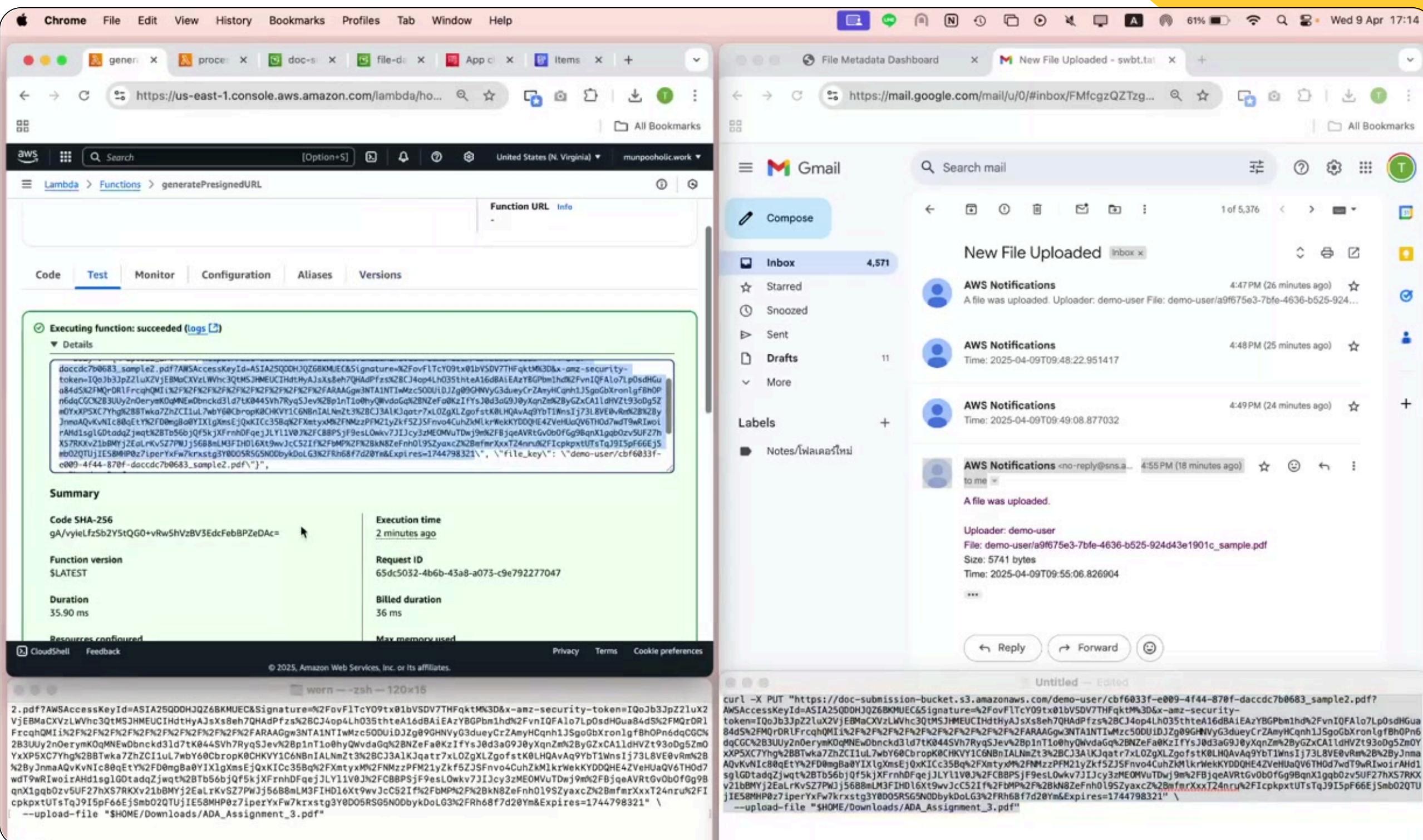
Log streams (8)		 Delete	Create log stream	Search all log streams
<input type="text"/> Filter log streams or try prefix search		<input type="checkbox"/> Exact match	<input type="checkbox"/> Show expired  Info	< 1 > 
<input type="checkbox"/> Log stream	Last event time			
<input type="checkbox"/> 2025/04/09/[LATEST]41ada10dc8c84a48a44033aaa7003604	2025-04-09 10:12:47 (UTC)			
<input type="checkbox"/> 2025/04/09/[LATEST]fc42ab0530a740dfa0b4d512158a614a	2025-04-09 09:55:07 (UTC)			
<input type="checkbox"/> 2025/04/09/[LATEST]34868a61eea34436a7b7e0bd00a2dfab	2025-04-09 09:49:09 (UTC)			
<input type="checkbox"/> 2025/04/07/[LATEST]dac69d63188a45f497aa3ab1f0eabdcc	2025-04-07 20:16:29 (UTC)			
<input type="checkbox"/> 2025/04/07/[LATEST]ce15b1db440c4adefbfde24db55c73bb5	2025-04-07 13:35:21 (UTC)			
<input type="checkbox"/> 2025/04/07/[LATEST]599f689c739e418f91d9f92fc5daa870	2025-04-07 10:39:26 (UTC)			
<input type="checkbox"/> 2025/04/07/[LATEST]519f28d38866445e95453d0238ec7d39	2025-04-07 09:51:47 (UTC)			
<input type="checkbox"/> 2025/04/07/[LATEST]fe5b1dd1eeee4e1ca8f793b3505266a4	2025-04-07 09:37:32 (UTC)			



DEMO - PART1



DEMO - PART2





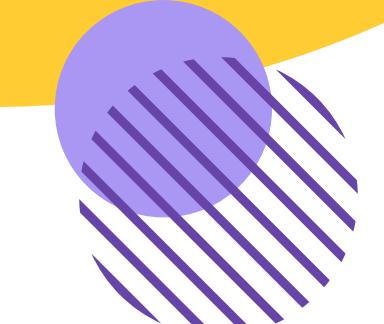
05 CHALLENGES



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



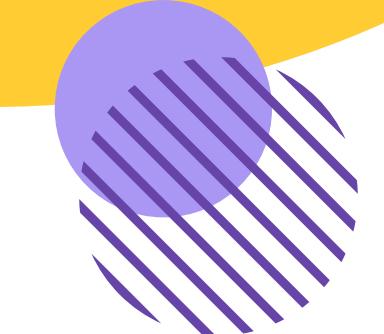


06 CONCLUSION

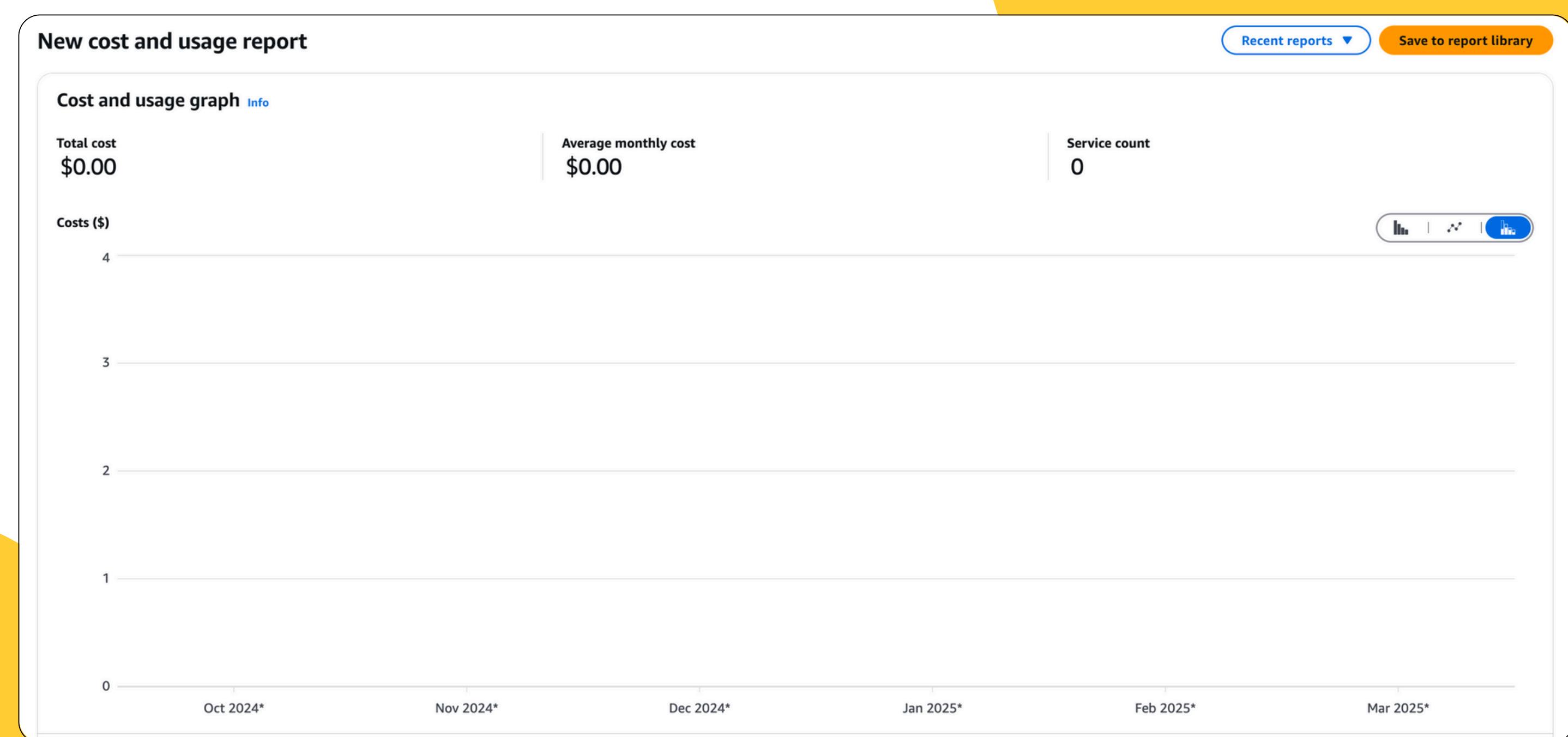


AWS WELL-ARCHITECTED FRAMEWORK

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.

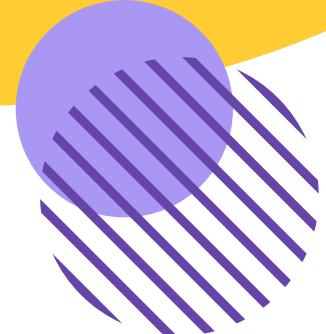


COST-OPTIMIZATION



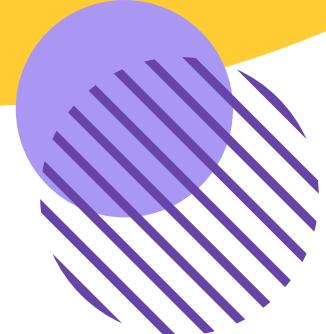
FUTURE WORK

- 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