Dhirubhai Ambani University
Technology

Formerly DA-IICT

# IT457 Cloud Computing

# Final Report

**Date:** December 2, 2025

## CloudLens

Serverless Content Moderation & Processing Pipeline


**Group 34**

- 202412015 Swara Chokshi
- 202412083 Nitesh Sachade
- 202412109 Stuti Shah
- 202412117 Kishan Thanki

# 1. ABSTRACT

This project implements a fully serverless, event-driven cloud application on Amazon Web Services (AWS) that automates image moderation and processing. Using Amazon S3, AWS Lambda, Amazon Rekognition, and API Gateway, the solution enables users to upload images through a static web interface and select the desired processing action: Thumbnail generation or image watermarking. The system validates images for inappropriate content using Rekognition by default for every image uploaded, ensures secure uploads via presigned URLs, and processes images using the Pillow library. Output images are stored in dedicated S3 buckets depending on the action performed and if any inappropriate image is uploaded so a warning is shown to the user and image is moved to quarantine bucket and user gets option to download processed image and if multiple images then user gets option to download as zip folder. The architecture highlights cloud-native principles such as scalability, resource elasticity, least-privilege security, and integrated monitoring through Amazon CloudWatch. Designed to operate within the AWS Free Tier or beginner cloud credits, the project provides an efficient, cost-effective demonstration of serverless computing and event-driven workflows.

# 2. PROBLEM STATEMENT

With the increasing volume of user-generated content on modern platforms, organizations require automated, scalable, and cost-efficient systems to moderate and process images in real time. Traditional server-based systems often suffer from high operational costs, limited scalability, and security challenges, especially when handling untrusted uploads.

There is a need for a **serverless cloud solution** that:

- Automatically moderates images for unsafe or inappropriate content.
- Supports image processing tasks like thumbnailing and watermarking.
- Enables secure upload mechanisms for untrusted public clients.
- Eliminates infrastructure management overhead.
- Stays cost-efficient and ideally operates within AWS Free Tier limits.

This project addresses these challenges by leveraging AWS serverless components to build an event-driven, highly scalable, secure, and low-cost image moderation and processing pipeline.

# 3. PROJECT OVERVIEW

## 3.1 PROJECT GOALS

### 3.1.1 PRIMARY GOAL

To design, implement, test, package, deploy, and monitor a **serverless cloud application** on AWS that automatically **moderates** uploaded images for inappropriate content using Amazon Rekognition, and performs image **processing** (thumbnailing, watermarking) based on user-specified metadata, ensuring all operations stay within the **AWS Free Tier** limits or utilize initial account credits.

### 3.1.2 SECONDARY GOALS

- Demonstrate a practical understanding of event-driven architectures using S3 triggers and Lambda.
- Implement secure cloud practices including least-privilege IAM roles, S3 presigned URLs for uploads, and appropriate S3 bucket policies.
- Utilize serverless compute (AWS Lambda) to showcase elastic, scalable, and cost-effective processing.
- Successfully integrate multiple AWS services: S3, Lambda, Rekognition, and API Gateway.
- Implement application monitoring using a custom Amazon CloudWatch Dashboard.
- Gain hands-on experience in deploying and testing a multi-component cloud application.

## 3.2 PROJECT SCOPE

### 3.2.1 In Scope

- **Frontend:** A basic HTML/CSS/JavaScript single-page application hosted via S3 Static Website Hosting.
  - Allows user selection of image files (.jpg, .png).
  - Allows users to select multiple images at once for batch processing.
  - Allows user selection of processing action (Thumbnail, Watermark), to be passed as S3 object metadata.
  - Allows users to access processed image(s) and also download option for image and if multiple images are there then user can download zip file

containing all processed images.
    - The safety section shows if any of the image(s) were found inappropriate and removed automatically.
- **API Layer:** An Amazon API Gateway HTTP API endpoint.
    - Triggers a Lambda function to generate upload URLs.
    - Requires CORS configuration.
- **Upload Mechanism:** An AWS Lambda function that generates time-limited S3 presigned URLs for direct browser uploads.
- **Core Infrastructure:**
    - An S3 bucket configured to receive uploads via presigned URLs and trigger the main Lambda function. Requires CORS configuration.
    - An S3 Event Notification on the uploads bucket triggering the ModerationProcessor Lambda.
- **Backend Processing:**
    - An AWS Lambda function containing the primary application logic.
    - Reads action metadata from the triggering S3 object.
        - If unsafe: Moves original image to a dedicated S3 quarantine bucket.
        - If safe: Proceeds to complete selected action.
    - **Thumbnailing:** Uses the Pillow library to create a custom size thumbnail. Saves thumbnails to a dedicated S3 public-thumbnails bucket and also provides processed images to users on a website which is downloadable .
    - **Watermarking:** Uses the Pillow library to add a custom text watermark which we take as text input from the user. Saves watermarked images to a dedicated S3 watermarked-output bucket and also provides downloadable images.
- **Security:** Appropriate IAM Roles and Policies for both Lambda functions ensuring least-privilege access.
- **Monitoring:** A custom Amazon CloudWatch Dashboard displaying key Lambda metrics.
- **Constraint:** All AWS services utilized must operate within the limits of the AWS Free Tier or be covered by the initial $100-$200 AWS credits provided to new accounts.

### 3.2.2 Out of Scope

- User authentication or authorization system for the frontend.
- Processing of video files or non-image file formats.
- Advanced image editing features (e.g., filters, cropping beyond thumbnails).
- Persistent storage of moderation results or image metadata in a database (e.g.,

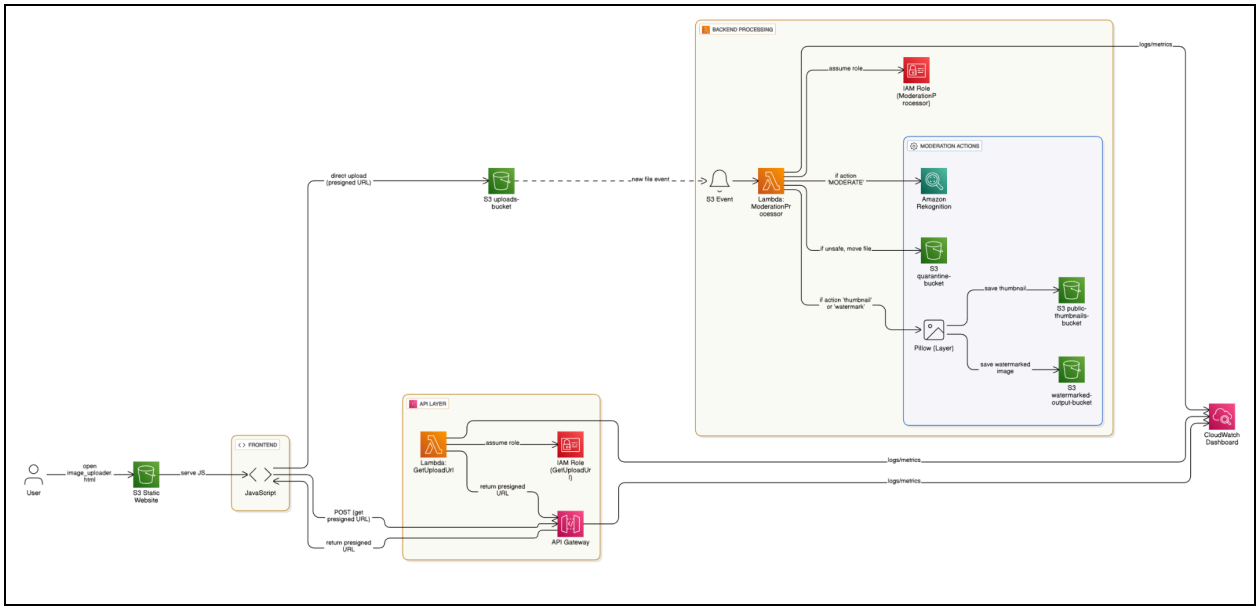
DynamoDB).

- Complex, granular error handling and retry logic for all potential failure points.
- Automated CI/CD pipeline for deployments. Manual deployment via AWS Console or CLI is acceptable.
- Advanced security measures (e.g., AWS WAF, detailed S3 Bucket Policies beyond basic access control and CORS).

# 4. CLOUDLENS ARCHITECTURE

The application is a **Serverless Event-Driven Pipeline**. It consists of two distinct, decoupled flows: the frontend upload (Flow 1) and the backend processing (Flow 2).

## 4.1 (3 - Tier) Architecture




## 4.2 Flow 1: Frontend Upload (User Interaction & Secure Upload)

This flow securely gets the file from the user's browser into S3 without exposing any credentials.

1. **User (Browser):** The user opens the image_uploader.html page, which is hosted publicly from an **S3 Static Website Hosting** bucket.
2. **Frontend (JavaScript):** The user selects a file and an action. The JS code then makes a POST request to our **API Gateway** endpoint.
3. **API Gateway:** This service acts as the secure front door. It receives the POST

request and triggers **Lambda Function 1**.

4. **Lambda 1 (GetUploadUrlLambda):** This function's only job is security. It uses its **IAM Role** (which has s3:PutObject permission) to generate a secure, temporary **S3 Presigned URL**.
5. **Return Flow:** The Presigned URL is sent back through API Gateway to the frontend JavaScript.
6. **Direct Upload:** The frontend JavaScript uses this unique URL to upload the image file directly to the private **S3 uploads-bucket**. The browser checks the bucket's **CORS policy** (which we configured to allow uploads from our S3 website URL) and completes the upload.

### 4.3 Flow 2: Backend Processing (Event-Driven Pipeline)

This flow starts automatically the moment the upload in Flow 1 finishes.

1. **S3 (uploads-bucket):** The new file upload is detected.
2. **S3 Event Notification:** An event is immediately sent to trigger **Lambda Function 2 (ModerationProcessor)**.
3. Lambda 2 (ModerationProcessor): This is the "brain" of the application. It uses its IAM Role to:

➢ **By default**

1. Call Amazon Rekognition (DetectModerationLabels).
2. If Unsafe: Move the original file to the S3 quarantine-bucket and return the respective respone.
3. If Safe: Proceed to the selected action.

- **If action == 'thumbnail':**
  1. Use the Pillow (Layer) to resize the image according to the provided size.
  2. Save the new thumbnail to the S3 public-thumbnails-bucket and show and provide a download option on the website to the user.
- **If action == 'watermark':**
  1. Use the Pillow (Layer) to add a watermark which was entered by the user.
  2. Save the new image to the S3 watermarked-output-bucket and show and provide a download option on the website to the user.

4. **CloudWatch (Monitoring):** Throughout this entire process, both Lambda

functions and API Gateway are continuously sending logs and metrics, which are collected in CloudWatch and displayed on our **CloudWatch Dashboard**.

# 5. PLATFORMS & TECHNOLOGIES USED

The CloudLens application is built entirely on AWS using a serverless-first approach. Each platform and technology was selected based on clear technical requirements, cost considerations, scalability needs, and the nature of our problem (untrusted uploads, real-time moderation, low maintenance).

This section explains **why** each service was chosen, the **benefits it provides**, and the **problem it solves** in our architecture.

## 5.1 Serverless Architecture

We adopted a **serverless architecture** because the application workload is fully event-driven and does not require long-running servers. Manual provisioning, configuration, patching, and scaling of traditional virtual machines (EC2/VMs) would add unnecessary complexity and cost.

### Why Serverless?

- **Automatic Scaling:** Lambda automatically adjusts capacity based on the number of uploads, ideal for unpredictable traffic.
- **Zero Server Maintenance:** No operating system updates, patching, or uptime management required.
- **Pay-per-Use:** We only pay when a function is executed or when storage/API calls are made.
- **Built for Event-Driven Workflows:** S3 upload → triggers → moderation → processing flow fits perfectly into serverless design.

### Need for Serverless

The workload consists of short, image-based tasks that run only when triggered. Thus, a fully serverless model ensures maximum efficiency and zero idle costs.

## 5.2 Amazon S3 – Object Storage + Static Website Hosting

### Why S3 for Storage?

- S3 is a reliable, scalable, and cost-effective storage service for large volumes of images.
- Supports **direct browser uploads using presigned URLs**, making it secure for untrusted public users.
- Provides event notifications that trigger downstream processing automatically.
- Stores original uploads, processed images (thumbnails/watermarked), and unsafe images in separate buckets.

### Why S3 Static Website Hosting for Frontend?

- Simple HTML/CSS/JS frontend does not require a backend server.
- Fast global access with S3 + CloudFront (optional).
- Extremely low cost (only storage charges).
- Fully serverless hosting with zero maintenance.

### Benefit

S3 centralizes all static content and user uploads under the same serverless environment and integrates seamlessly with other AWS services.

## 5.3 AWS Lambda – Compute Engine for Core Processing

### Why Lambda instead of EC2?

| Lambda | EC2 |
|---|---|
| Auto-scaled | Must manually scale |
| Pay only per request | Pay 24/7 even when idle |

| | |
|---|---|
| No server management | Requires OS maintenance |
| Fast deployment | Longer setup time |
| Event-driven | Not natively event-triggered |

Because our functions only run when an image arrives, Lambda is an ideal choice.

**Lambda's Roles in CloudLens**

- Generate presigned S3 upload URLs
- Moderate images using Rekognition
- Process images using Pillow (Lambda Layer)

## Motivation

Lambda dramatically simplifies architecture, reduces cost, and eliminates server configuration overhead.

## 5.4 Amazon Rekognition – AI-Based Image Moderation

**Why Rekognition Instead of Building a Custom ML Model?**

Building a custom CV/ML model would require:

- Dataset collection
- Model training
- Infrastructure for inference
- Continuous tuning and retraining

Rekognition provides:

- Pre-trained moderation models
- High accuracy for explicit, unsafe, or inappropriate content
- Near real-time inference
- Fully managed service

**Need**

The goal of the project is not ML development but efficient moderation. Rekognition allows us to integrate industry-grade AI moderation instantly.

## 5.5 Amazon API Gateway – Secure API Front Door

**Why API Gateway?**

- Acts as a secure interface between frontend and Lambda.
- Prevents exposing Lambda publicly.
- Enforces HTTP method validation and CORS restrictions.
- Handles throttling, access control, and request routing.
- Integrates natively with Lambda.

**Why Needed for This Project?**

The frontend must request a **temporary presigned URL** for secure uploads. Exposing S3 directly would be unsafe.
API Gateway ensures:

- No user ever receives AWS credentials
- Only signed, time-limited upload URLs are issued
- Frontend can safely upload images without exposing private buckets

## 5.6 AWS IAM – Identity and Access Management

**Why IAM?**

- Enforces **least privilege** permissions for Lambda and S3.
- Prevents unauthorized access to buckets.
- Ensures presigned URLs only allow controlled uploads.

**Need**

Because the application deals with public uploads, strict security boundaries are essential.

### 5.7 Amazon CloudWatch – Logging, Monitoring & Dashboarding

#### Why CloudWatch?

CloudWatch gives complete visibility into:

- Lambda invocation metrics
- Errors, throttles, and cold starts
- Rekognition API call logs
- API Gateway traffic

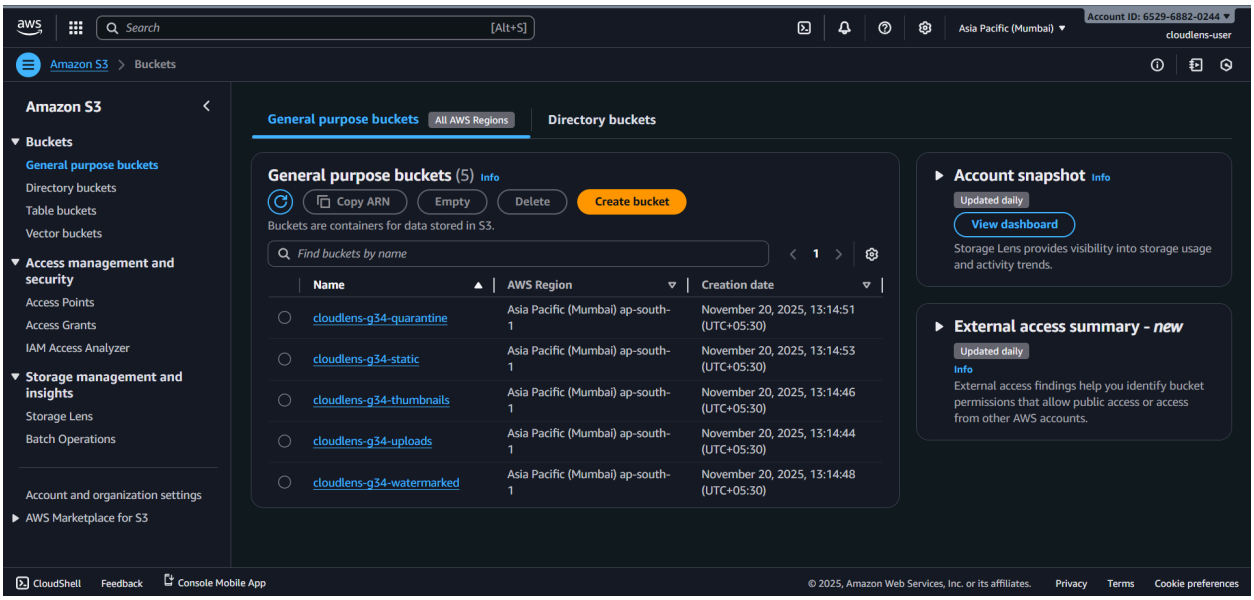
#### Need

Real-time monitoring helps verify correct system behavior, detect failures, and demonstrate operational insights for the project evaluation.

## 6. AWS SERVICES USED

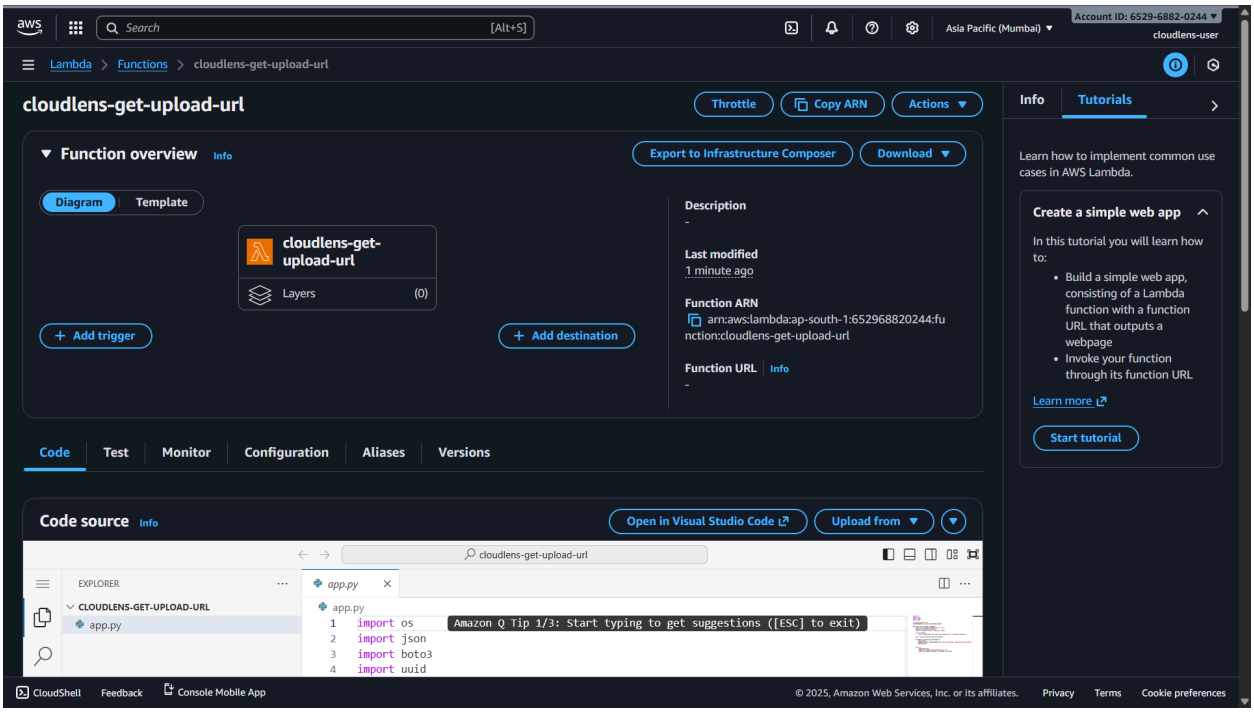
### 6.1 Amazon S3 (Simple Storage Service)

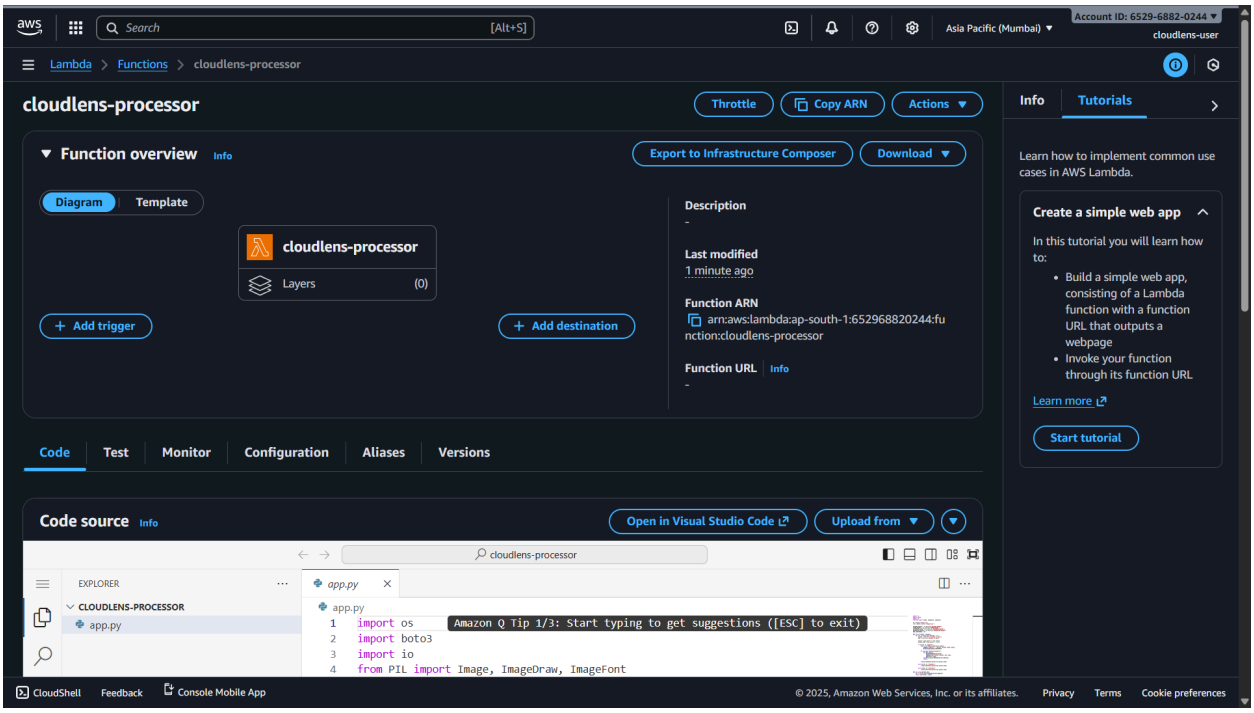
- Used to store uploaded images, processed outputs (thumbnails/watermarked), and unsafe images in separate dedicated buckets.



## 6.2 AWS Lambda

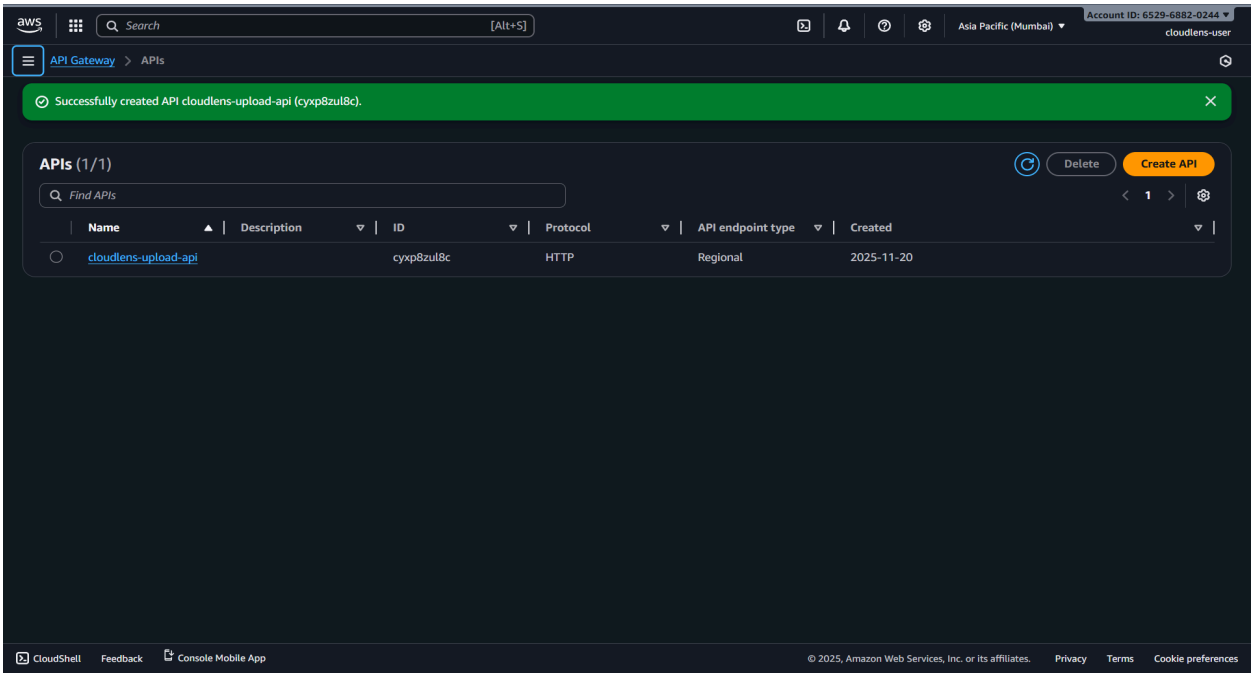
- Serverless computer used for generating presigned URLs and processing uploaded images (moderation + editing).



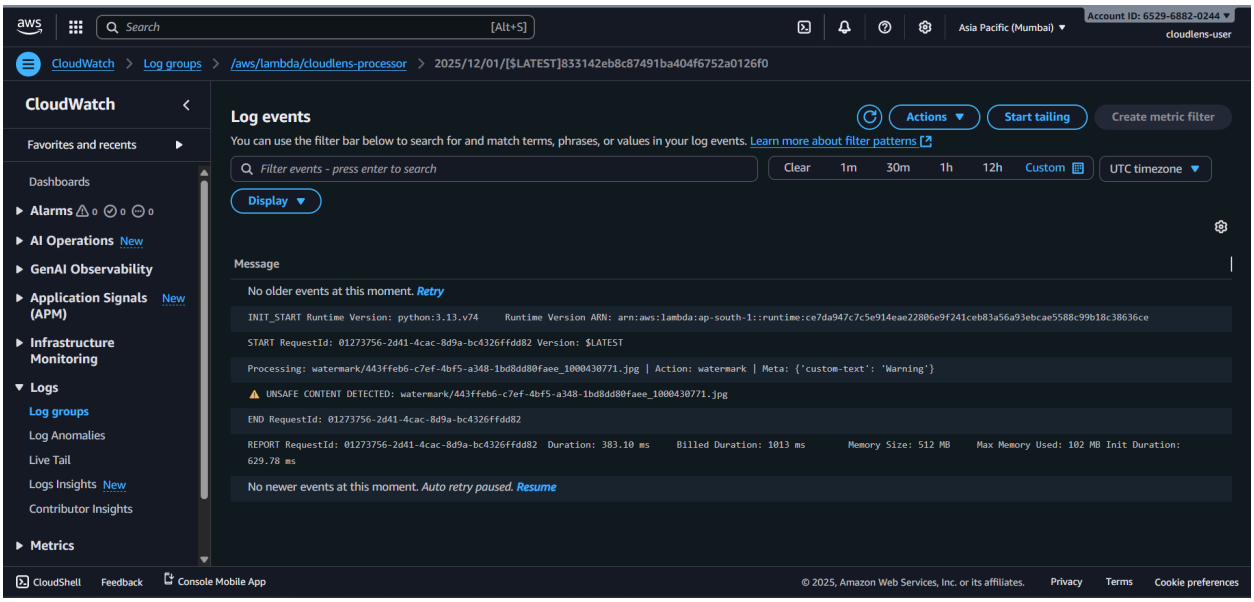
## 6.3 Amazon API Gateway (HTTP API)

- Provides a secure public API endpoint that triggers the Lambda function to generate presigned S3 upload URLs.



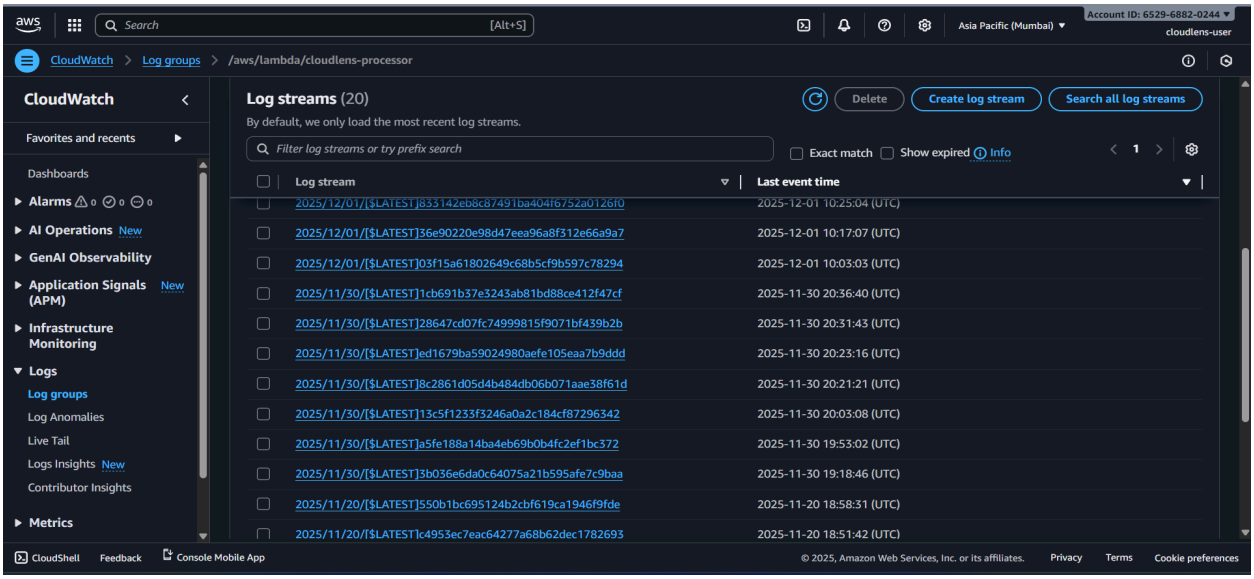

## 6.4 Amazon Rekognition

- Performs automated image moderation to detect unsafe or inappropriate content.



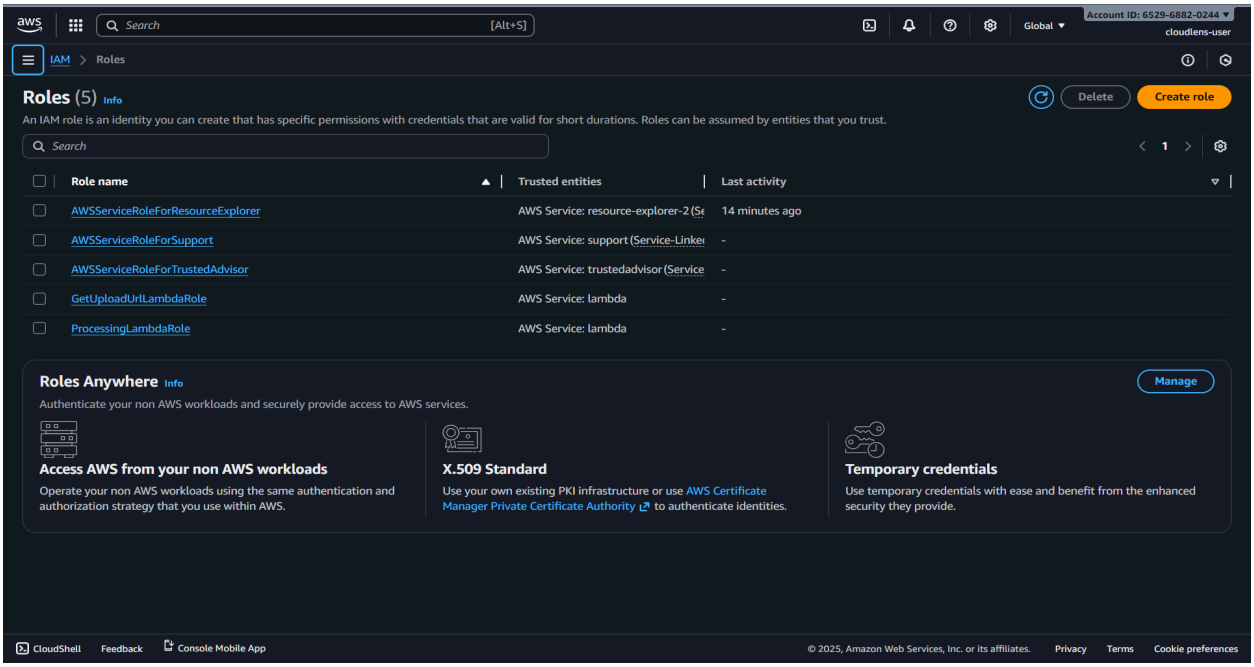
## 6.5 Amazon CloudWatch

- Collects logs and metrics from Lambda and API Gateway, and displays them through a custom monitoring dashboard.



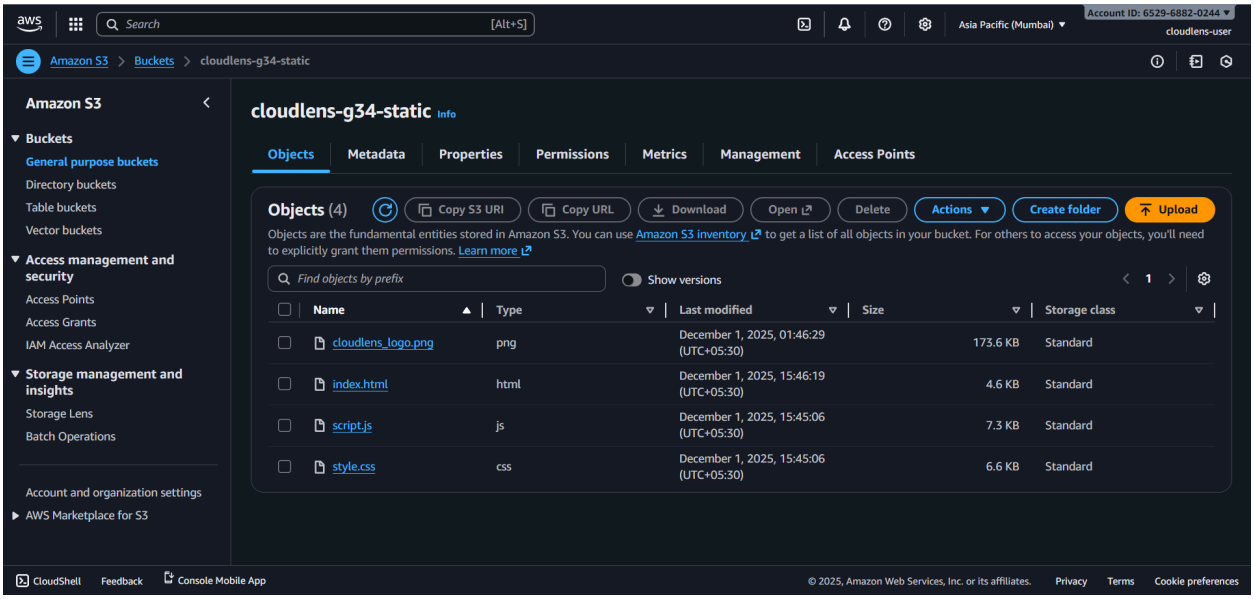

## 6.6 AWS IAM (Identity and Access Management)

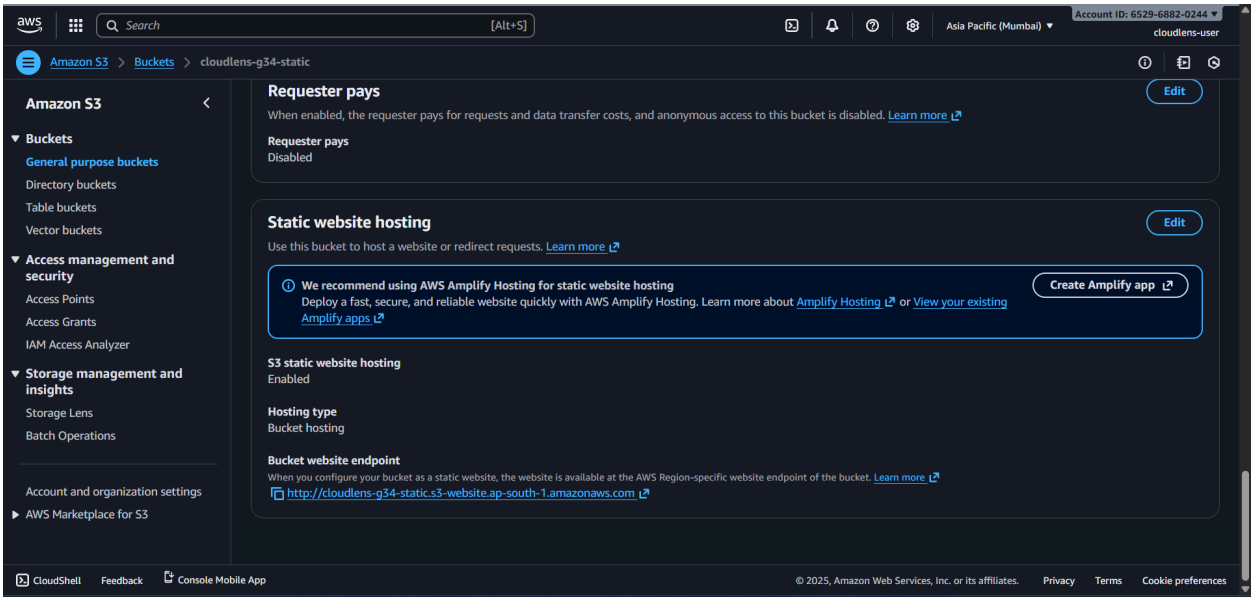
- Provides fine-grained least-privilege permissions for Lambdas, S3 buckets, and API Gateway.



## 6.7 S3 Static Website Hosting

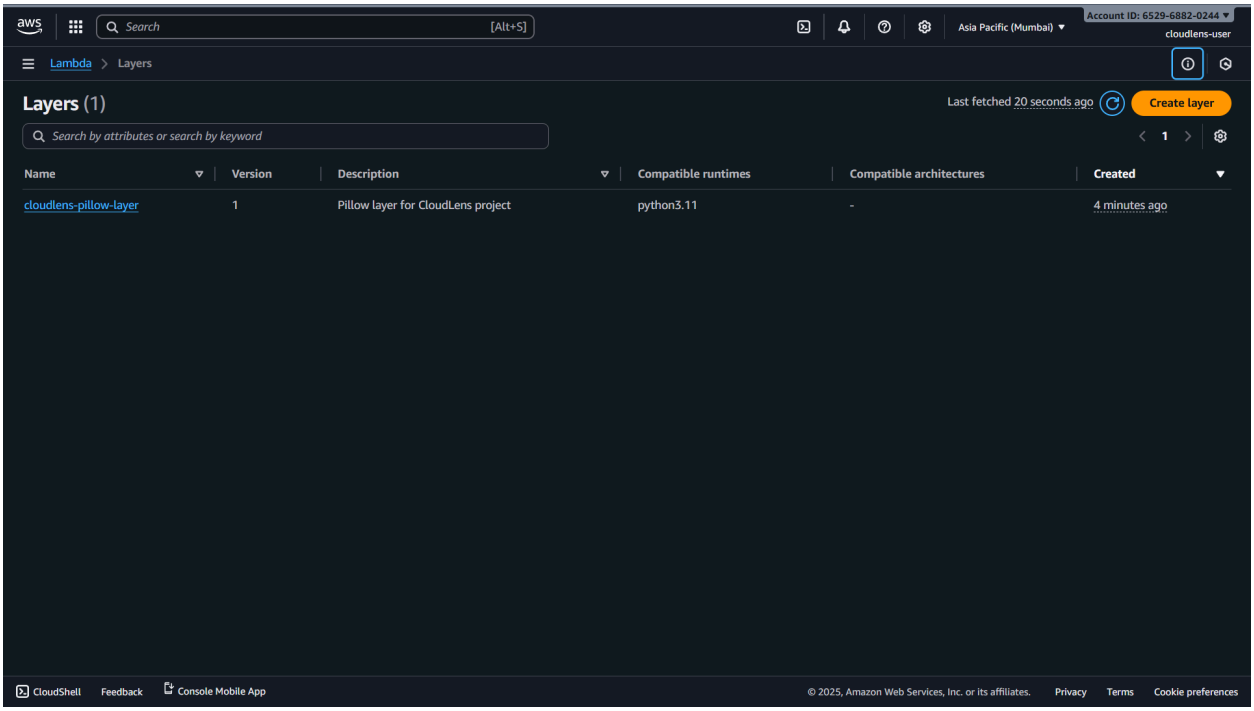
- Hosts the frontend web interface (HTML/CSS/JS) for user uploads and interactions.

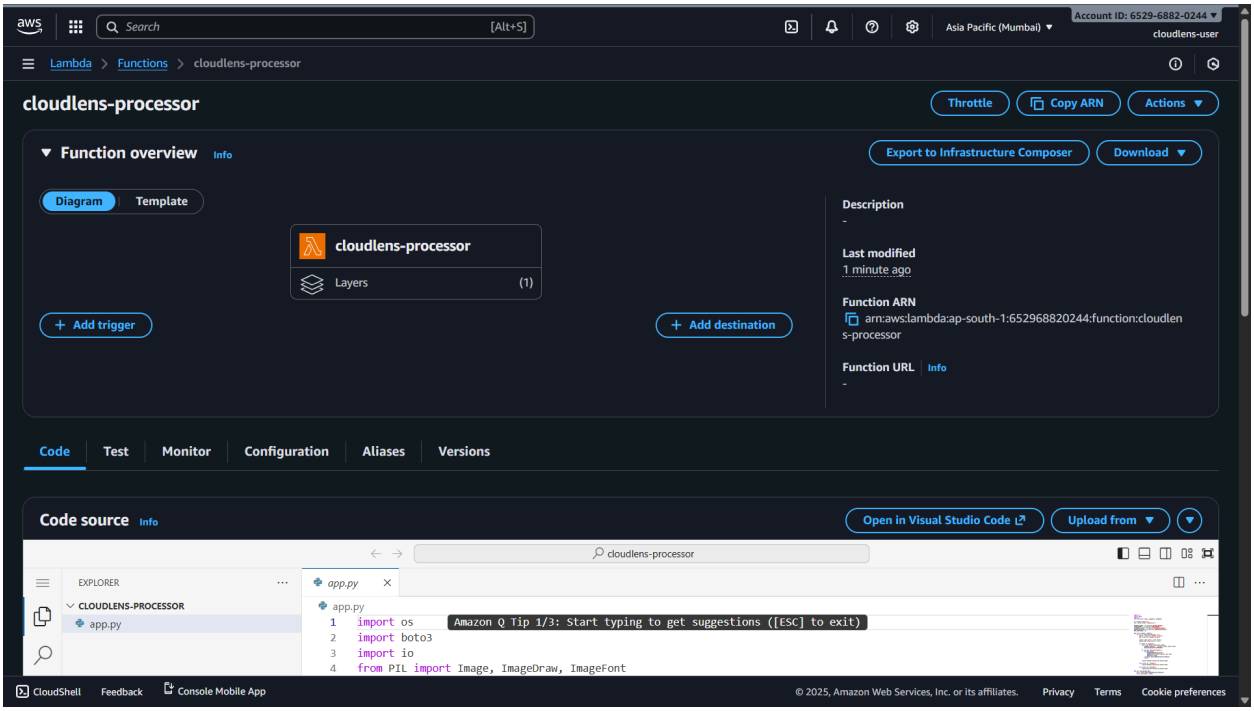


## 6.8 AWS Lambda Layers

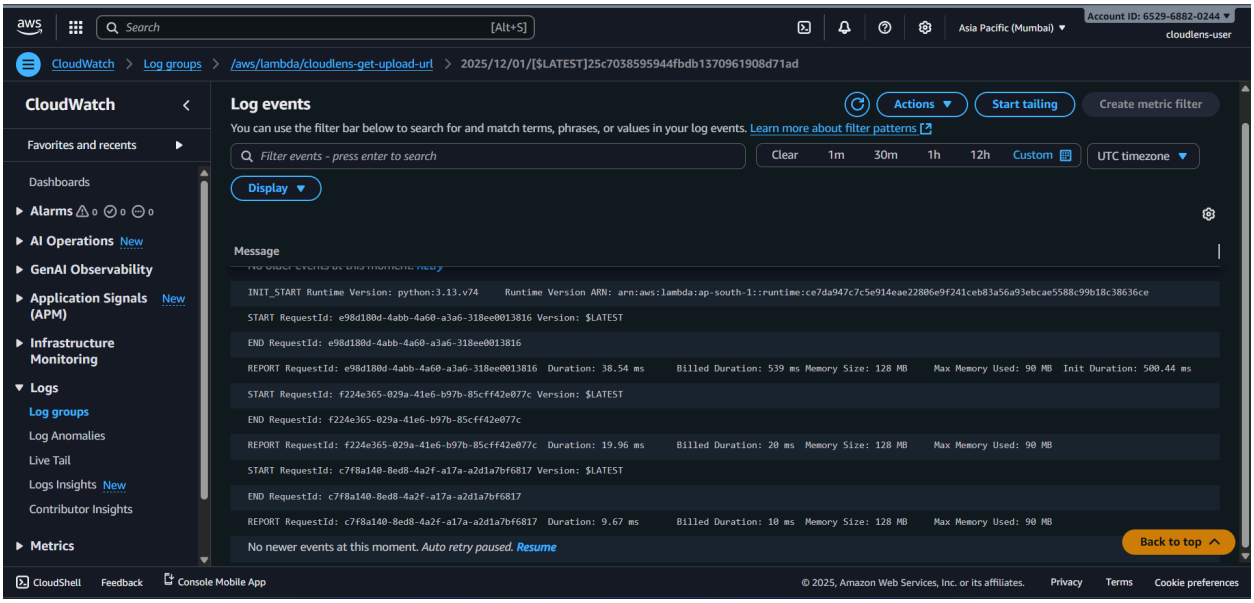
- Used to package and attach the Pillow image-processing library for thumbnailing and watermarking.



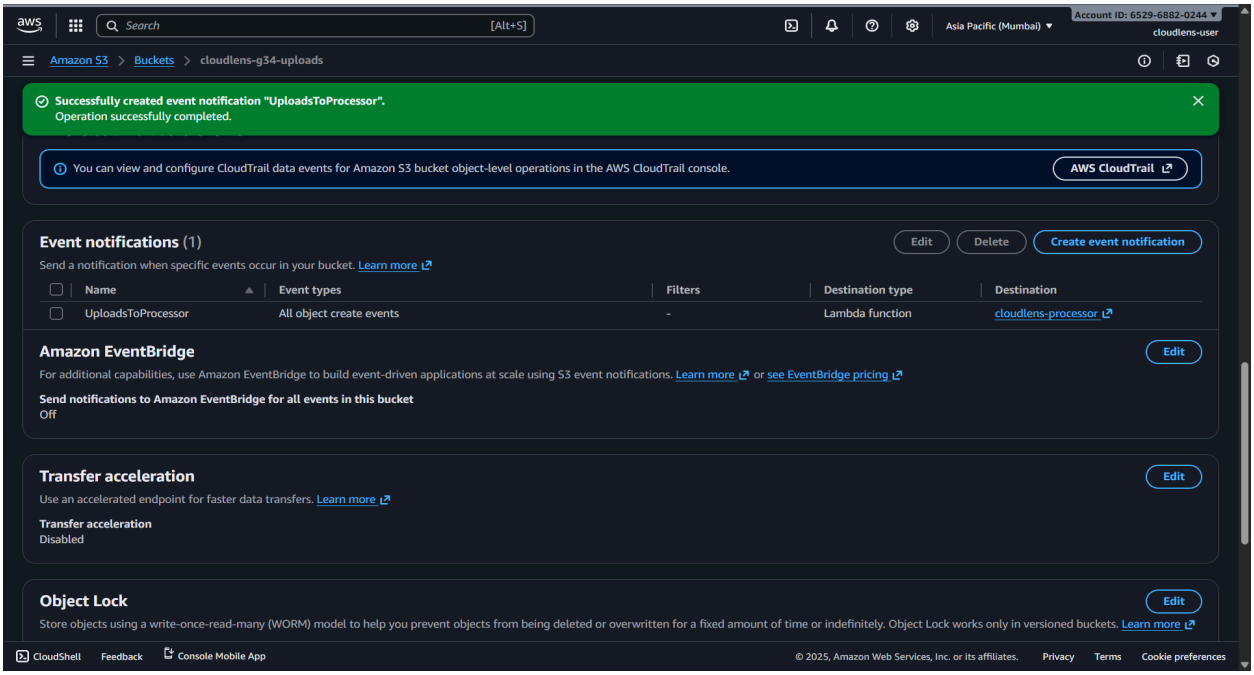
## 6.9 Amazon CloudWatch Logs

- Stores detailed execution logs from both Lambda functions for debugging and monitoring.



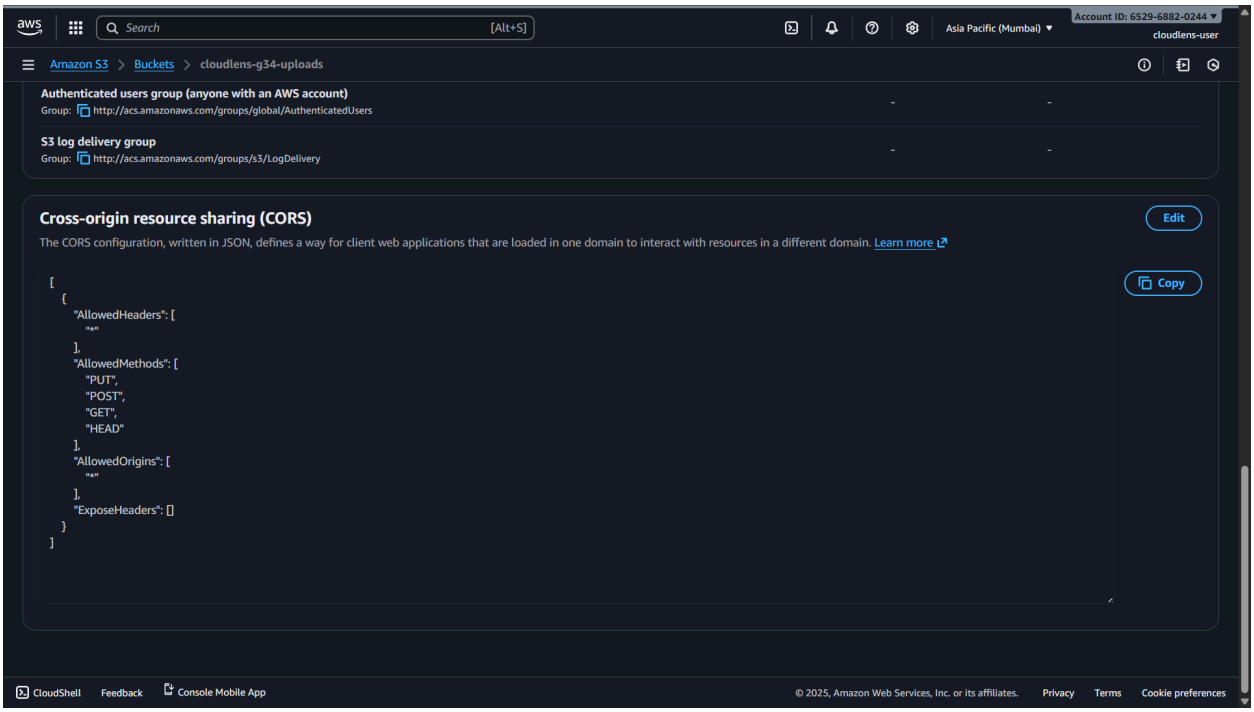

## 6.10 S3 Event Notifications

- Automatically triggers the ModerationProcessor Lambda when a new file is uploaded.



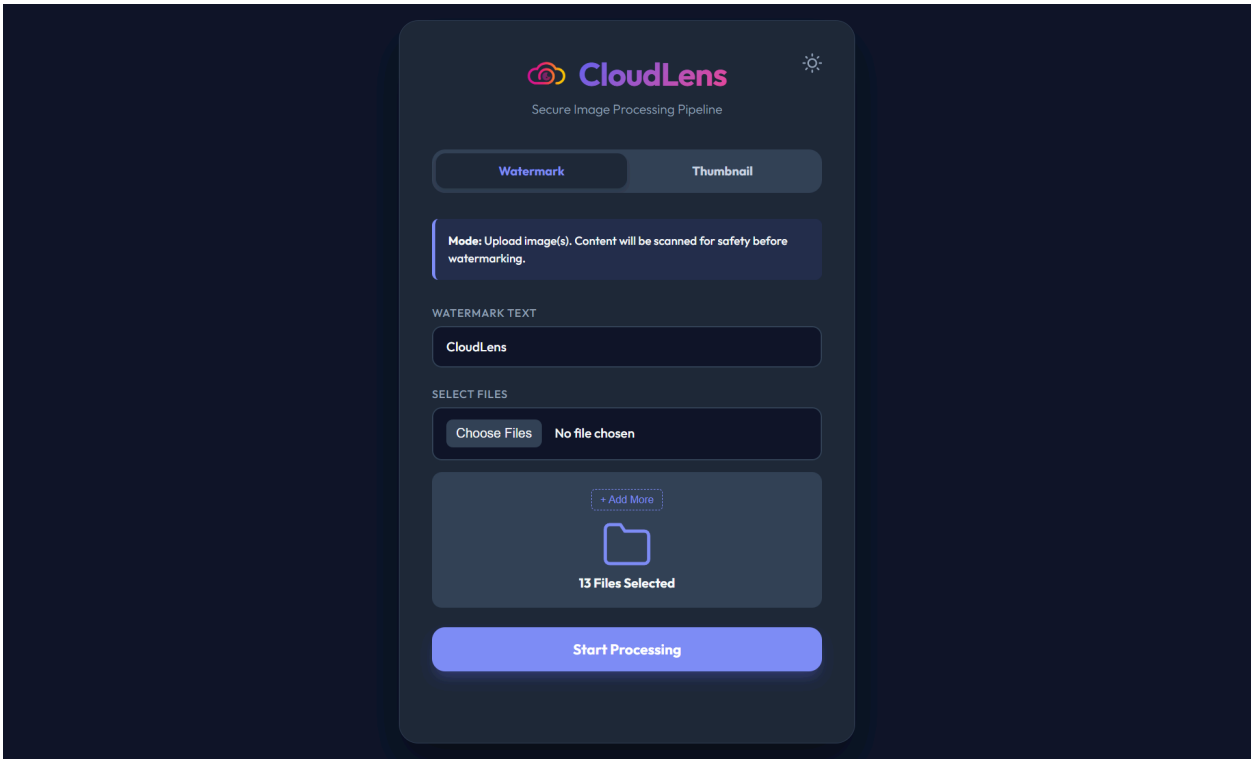
## 6.11 CORS Configuration (S3 + API Gateway)

- Ensures secure cross-origin requests between the static website and API endpoints.

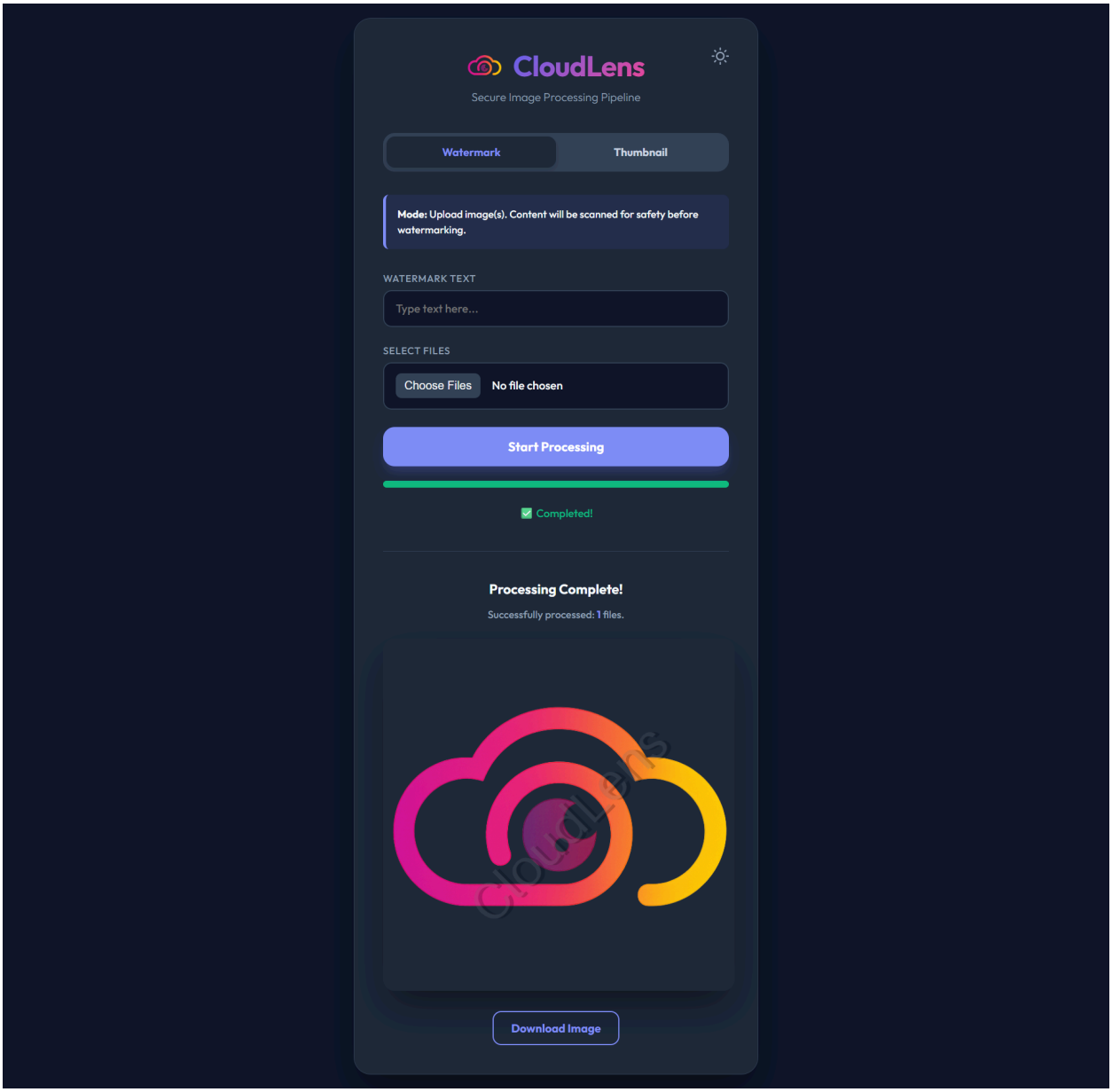


# 7. FRONTEND

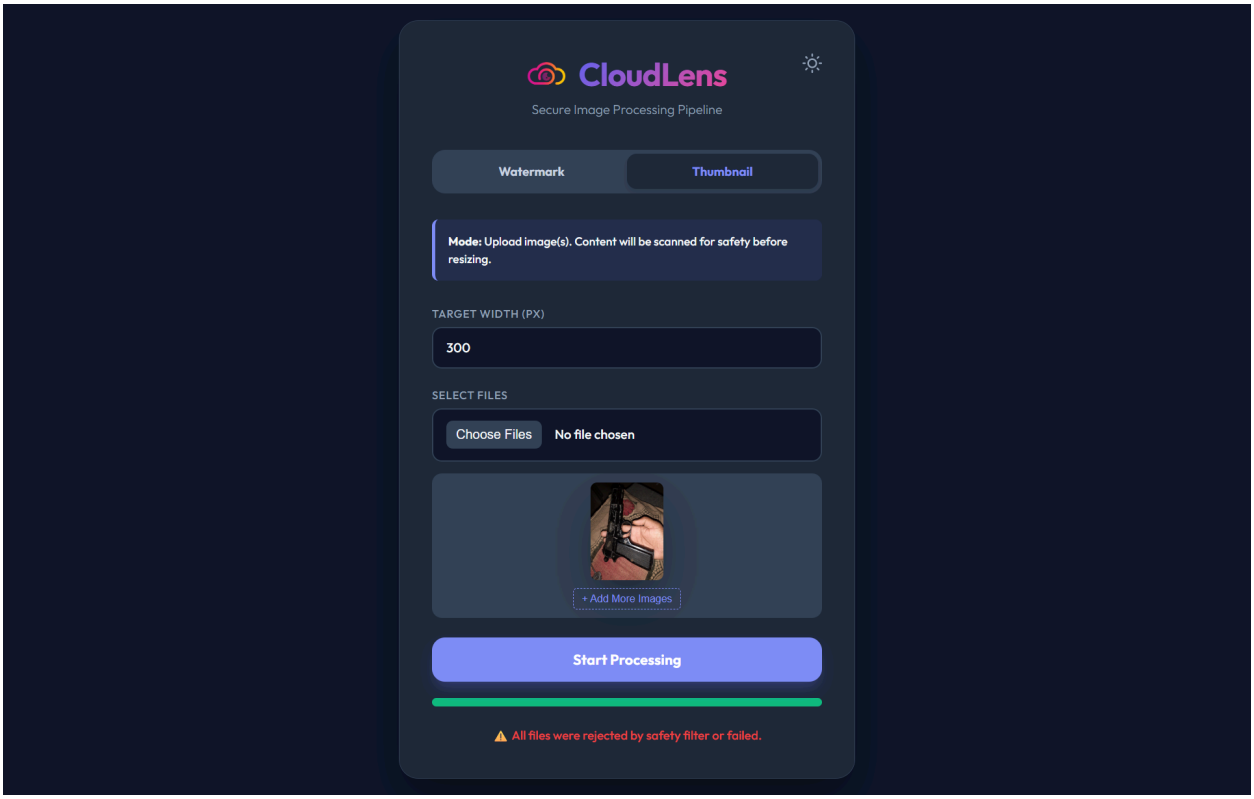
## 7.1 Upload Image(s)



## 7.2 Watermarked Image

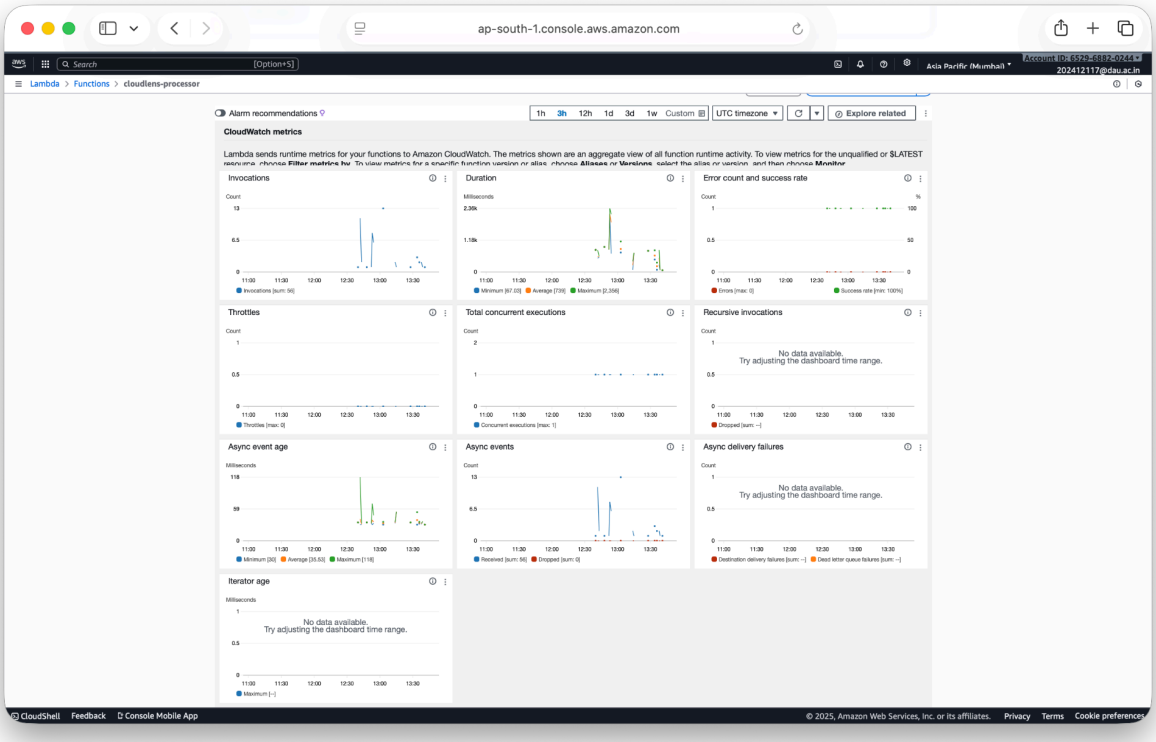


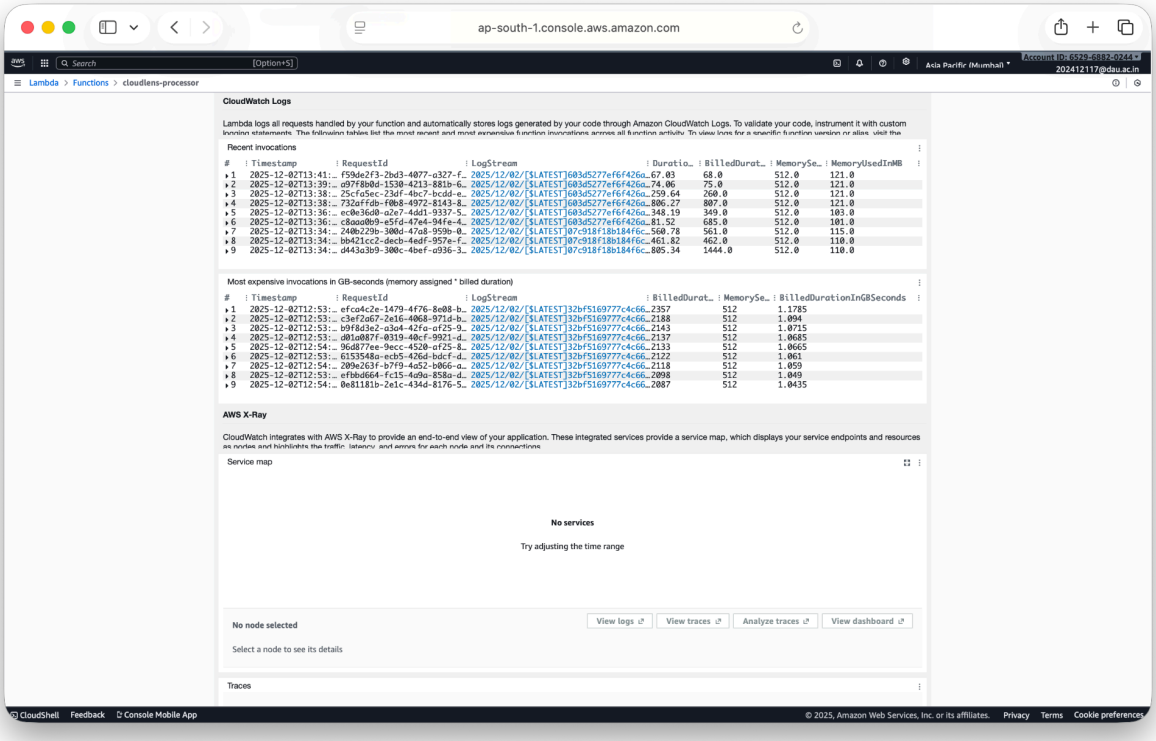
## 7.3 Uploaded Image for Thumbnail is Inappropriate



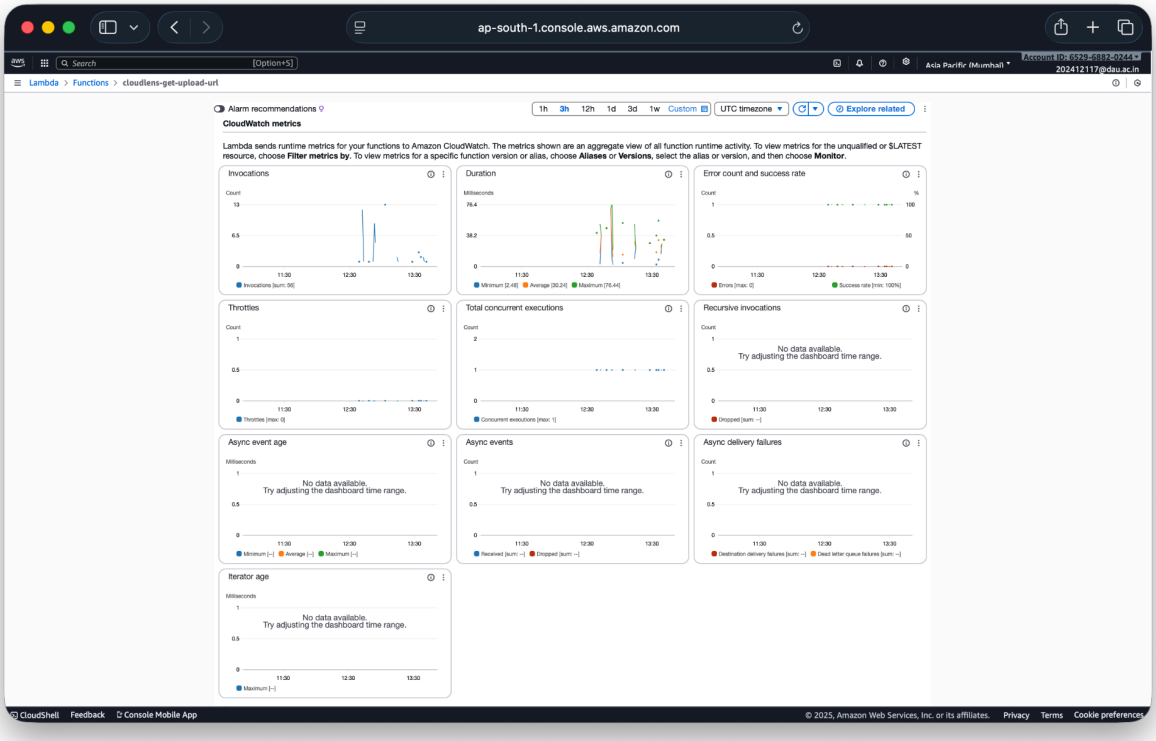
# 8. DASHBOARD SCREENSHOTS

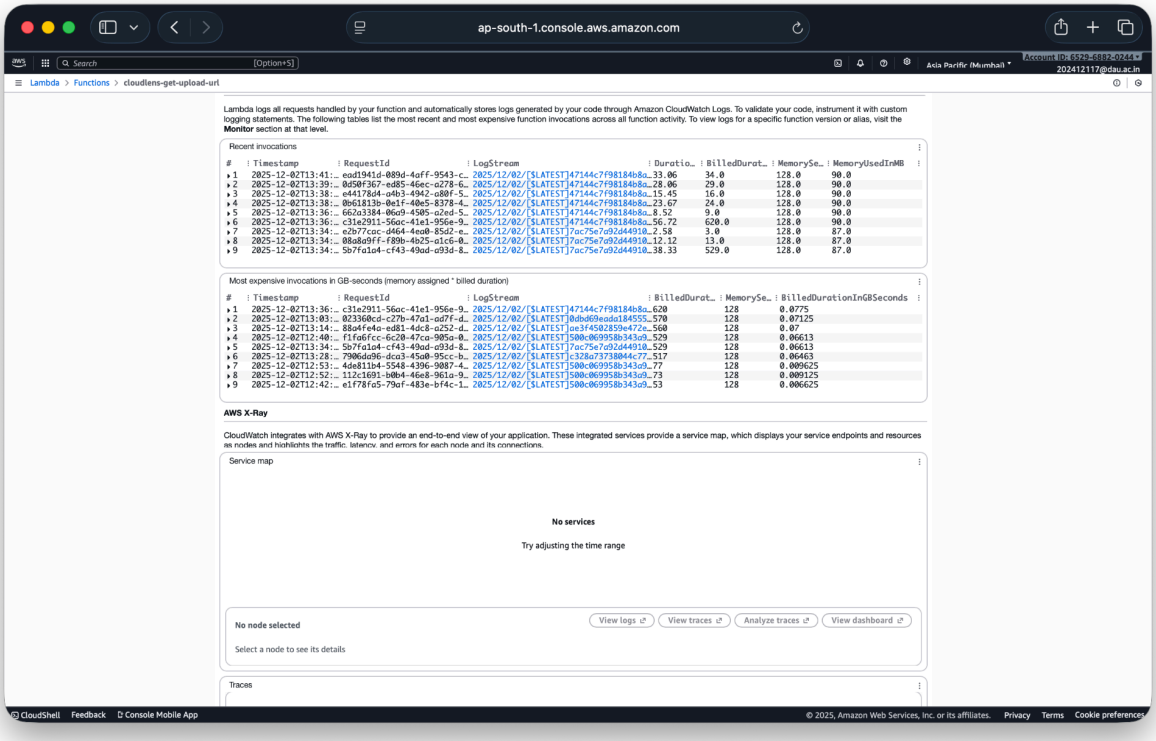
## 8.1 Lambda Function cloudlens-processor



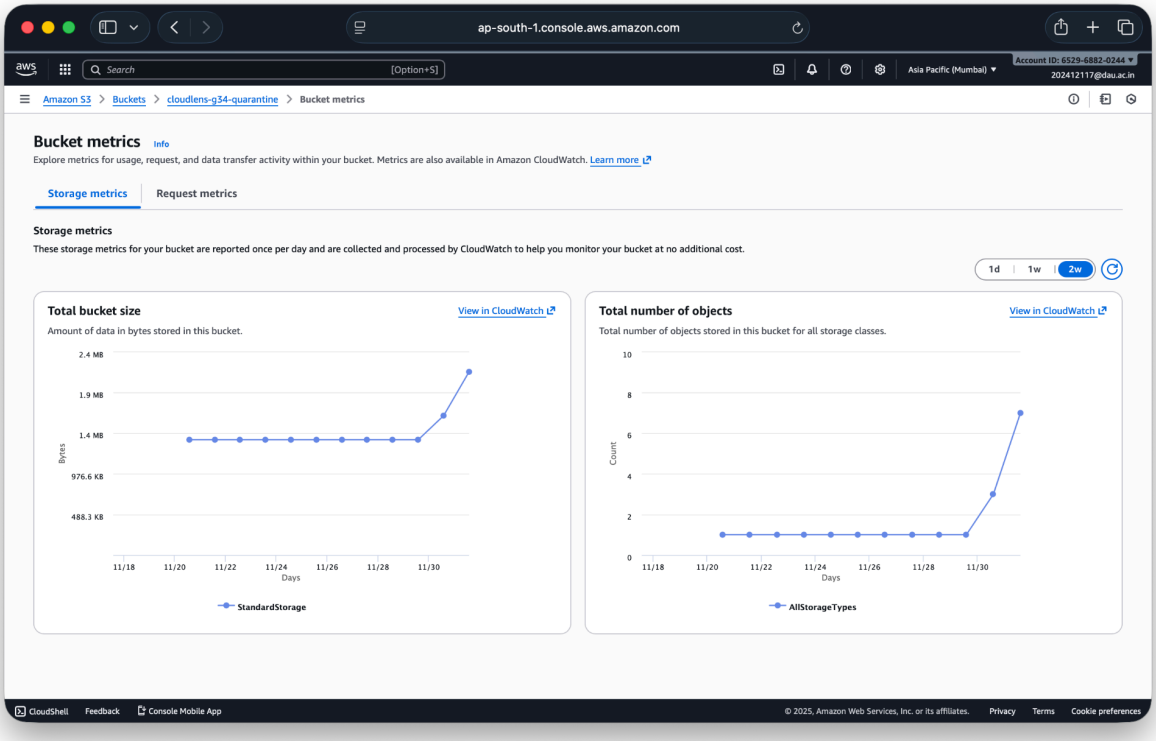

ap-south-1.console.aws.amazon.com
Search
[Option+S]
Asia Pacific (Mumbai)
202412117@dau.ac.in
Lambda > Functions > cloudlens-processor
CloudWatch Logs
Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom
Recent invocations
# Timestamp RequestId LogStream Duratio… BilledDurat… MemorySe… MemoryUsedInMB
1 2025-12-02T13:41:… f59de2f3-2bd3-4077-a327-f… 2025/12/02/[$LATEST]603d5277ef6f426a… 67.03 68.0 512.0 121.0
2 2025-12-02T13:39:… a97f8b0d-1530-4213-881b-6… 2025/12/02/[$LATEST]603d5277ef6f426a… 74.06 75.0 512.0 121.0
3 2025-12-02T13:38:… 25cfa5ec-23df-4bc7-bcdd-e… 2025/12/02/[$LATEST]603d5277ef6f426a… 259.64 260.0 512.0 121.0
4 2025-12-02T13:38:… 732effdb-f0b8-4972-8143-8… 2025/12/02/[$LATEST]603d5277ef6f426a… 806.27 807.0 512.0 121.0
5 2025-12-02T13:36:… ec0e36d0-a2e7-4dd1-9337-5… 2025/12/02/[$LATEST]603d5277ef6f426a… 348.19 349.0 512.0 103.0
6 2025-12-02T13:36:… c8aaa0b9-e5fd-47e4-94fe-4… 2025/12/02/[$LATEST]603d5277ef6f426a… 81.52 685.0 512.0 101.0
7 2025-12-02T13:34:… 240b229b-300d-47a8-959b-0… 2025/12/02/[$LATEST]07c918f18b184f6c… 560.78 561.0 512.0 115.0
8 2025-12-02T13:34:… bb421cc2-decb-4edf-957e-f… 2025/12/02/[$LATEST]07c918f18b184f6c… 461.82 462.0 512.0 110.0
9 2025-12-02T13:34:… d443a3b9-300c-4bef-a936-3… 2025/12/02/[$LATEST]07c918f18b184f6c… 805.34 1444.0 512.0 110.0
Most expensive invocations in GB-seconds (memory assigned * billed duration)
# Timestamp RequestId LogStream BilledDurat… MemorySe… BilledDurationInGBSeconds
1 2025-12-02T12:53:… efca4c2e-1479-4f76-8e08-b… 2025/12/02/[$LATEST]32bf5169777c4c66… 2357 512 1.1785
2 2025-12-02T12:53:… c3ef2a67-2e16-4068-971d-b… 2025/12/02/[$LATEST]32bf5169777c4c66… 2188 512 1.094
3 2025-12-02T12:53:… b9f8d3e2-a3a4-42fa-af25-9… 2025/12/02/[$LATEST]32bf5169777c4c66… 2143 512 1.0715
4 2025-12-02T12:53:… d01a087f-0319-40cf-9921-d… 2025/12/02/[$LATEST]32bf5169777c4c66… 2137 512 1.0685
5 2025-12-02T12:54:… 96d877ee-9ecc-4520-af25-8… 2025/12/02/[$LATEST]32bf5169777c4c66… 2133 512 1.0665
6 2025-12-02T12:53:… 6153548a-ecb5-426d-bdcf-d… 2025/12/02/[$LATEST]32bf5169777c4c66… 2122 512 1.061
7 2025-12-02T12:54:… 209e263f-b7f9-4a52-b066-a… 2025/12/02/[$LATEST]32bf5169777c4c66… 2118 512 1.059
8 2025-12-02T12:53:… efbbd664-fc15-4a9a-858a-d… 2025/12/02/[$LATEST]32bf5169777c4c66… 2098 512 1.049
9 2025-12-02T12:54:… 0e81181b-2e1c-434d-8176-5… 2025/12/02/[$LATEST]32bf5169777c4c66… 2087 512 1.0435
AWS X-Ray
CloudWatch integrates with AWS X-Ray to provide an end-to-end view of your application. These integrated services provide a service map, which displays your service endpoints and resources
Service map
No services
Try adjusting the time range
No node selected
Select a node to see its details
View logs
View traces
Analyze traces
View dashboard
Traces
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

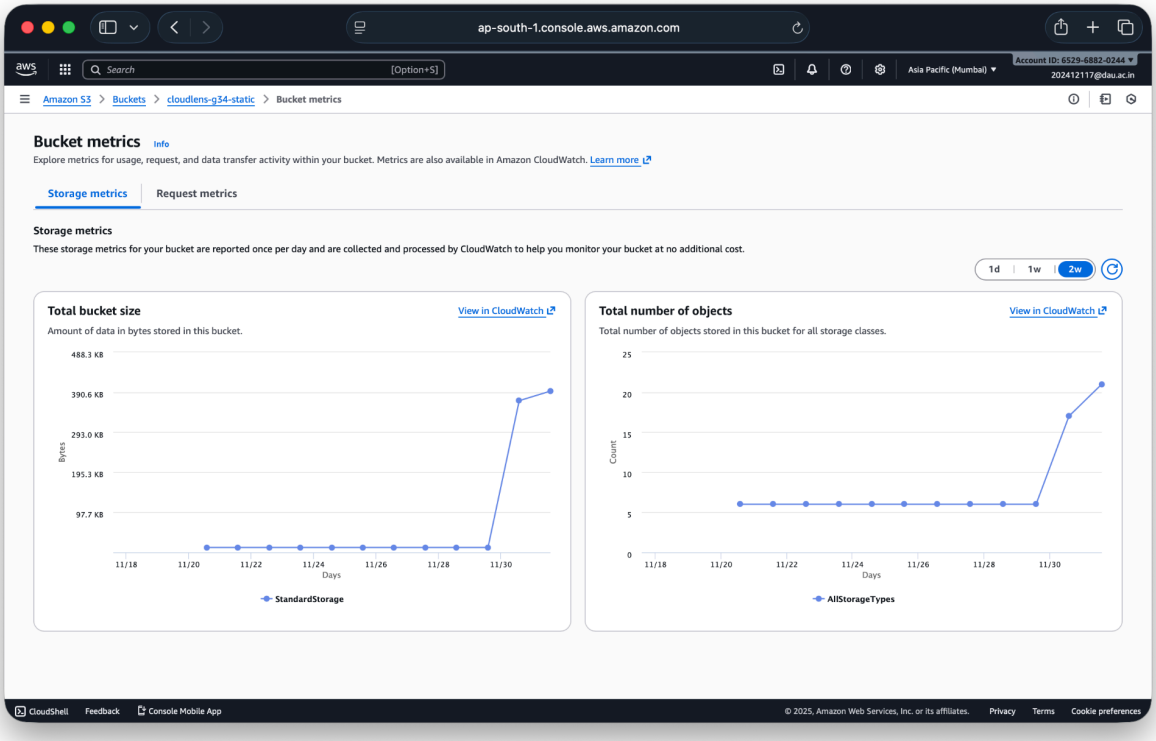
# 8.2 Lambda Function cloudlens-get-upload-url

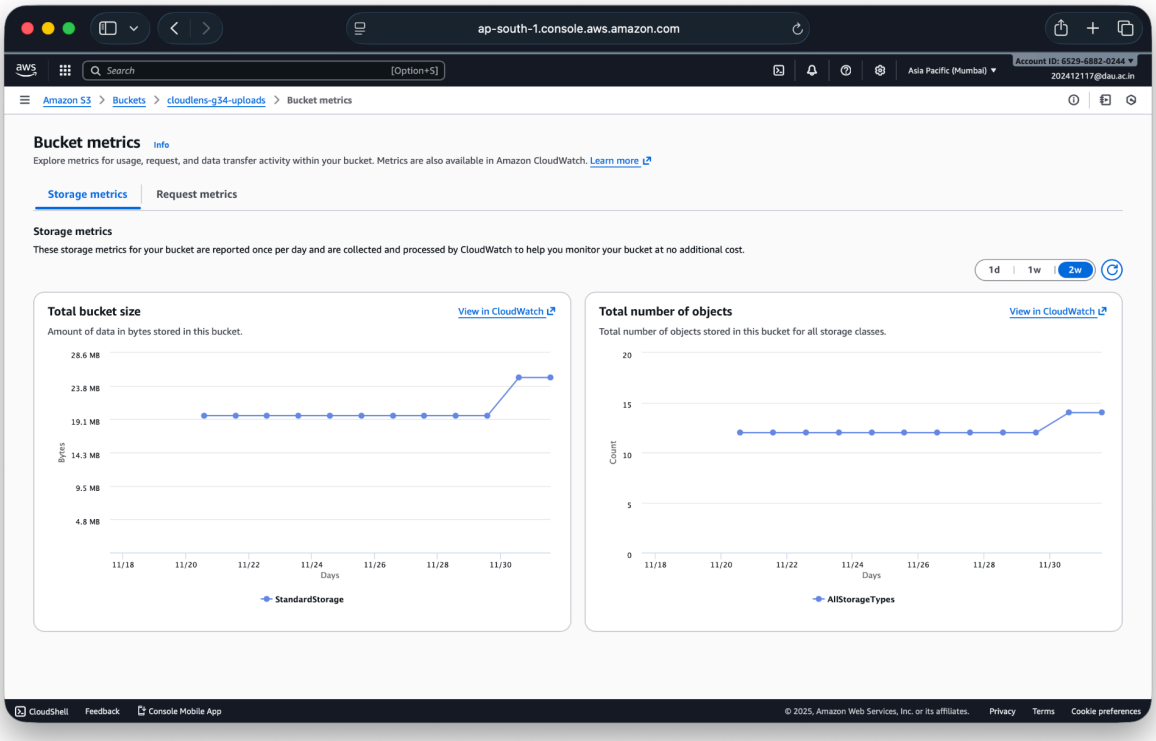


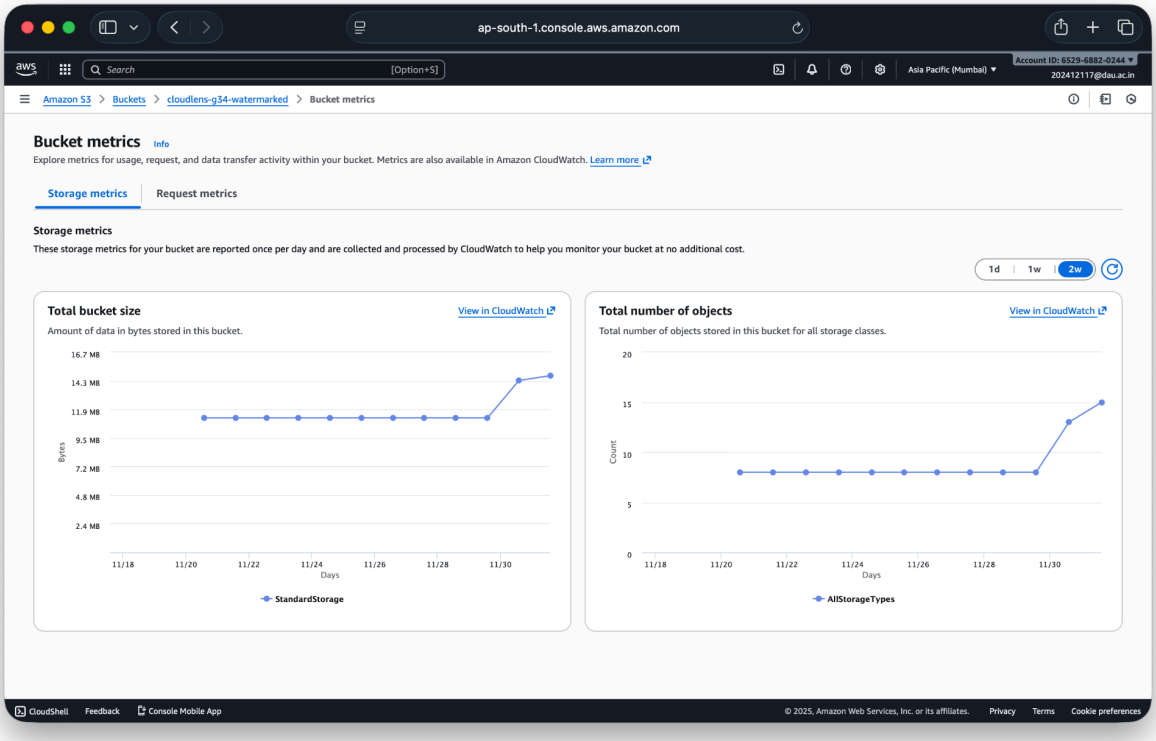



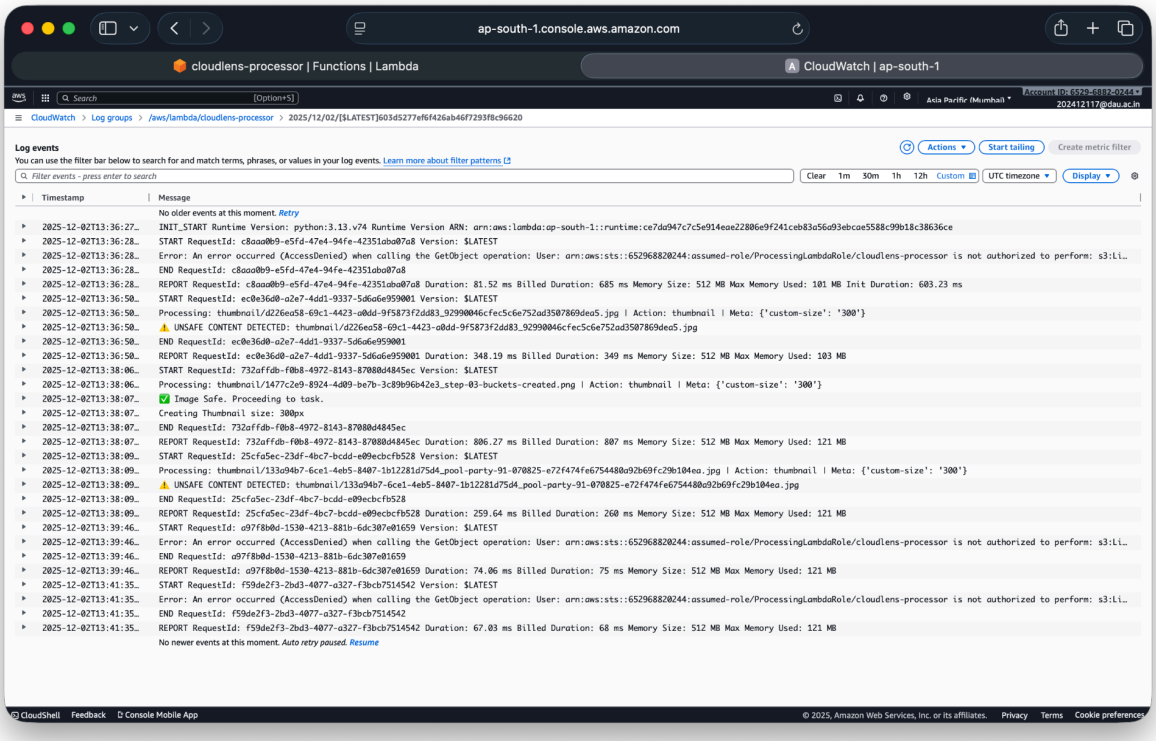
# 8.3 Buckets Metrics & Logs




ap-south-1.console.aws.amazon.com
Search
[Option+S]
Asia Pacific (Mumbai)
Account ID: 6529-6882-0244
202412117@dau.ac.in
Amazon S3 > Buckets > cloudlens-g34-uploads > Bucket metrics
Bucket metrics
Info
Explore metrics for usage, request, and data transfer activity within your bucket. Metrics are also available in Amazon CloudWatch. Learn more
Storage metrics
Request metrics
Storage metrics
These storage metrics for your bucket are reported once per day and are collected and processed by CloudWatch to help you monitor your bucket at no additional cost.
1d
1w
2w
Total bucket size
View in CloudWatch
Amount of data in bytes stored in this bucket.
28.6 MB
23.8 MB
19.1 MB
14.3 MB
9.5 MB
4.8 MB
Bytes
11/18
11/20
11/22
11/24
11/26
11/28
11/30
Days
StandardStorage
Total number of objects
View in CloudWatch
Total number of objects stored in this bucket for all storage classes.
20
15
10
5
0
Count
11/18
11/20
11/22
11/24
11/26
11/28
11/30
Days
AllStorageTypes
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


ap-south-1.console.aws.amazon.com
Search
[Option+S]
Asia Pacific (Mumbai)
Account ID: 6529-6882-0244
202412117@dau.ac.in
Amazon S3 > Buckets > cloudlens-g34-watermarked > Bucket metrics
Bucket metrics
Info
Explore metrics for usage, request, and data transfer activity within your bucket. Metrics are also available in Amazon CloudWatch. Learn more
Storage metrics
Request metrics
Storage metrics
These storage metrics for your bucket are reported once per day and are collected and processed by CloudWatch to help you monitor your bucket at no additional cost.
1d
1w
2w
Total bucket size
View in CloudWatch
Amount of data in bytes stored in this bucket.
16.7 MB
14.3 MB
11.9 MB
9.5 MB
7.2 MB
4.8 MB
2.4 MB
Bytes
11/18
11/20
11/22
11/24
11/26
11/28
11/30
Days
StandardStorage
Total number of objects
View in CloudWatch
Total number of objects stored in this bucket for all storage classes.
20
15
10
5
0
Count
11/18
11/20
11/22
11/24
11/26
11/28
11/30
Days
AllStorageTypes
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


ap-south-1.console.aws.amazon.com
cloudlens-processor | Functions | Lambda
CloudWatch | ap-south-1
CloudWatch > Log groups > /aws/lambda/cloudlens-processor > 2025/12/02/[$LATEST]603d5277ef6f426ab46f7293f8c96620
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
Actions
Start tailing
Create metric filter
Filter events - press enter to search
Clear 1m 30m 1h 12h Custom UTC timezone Display
Timestamp
Message
No older events at this moment. Retry
2025-12-02T13:36:27… INIT_START Runtime Version: python:3.13.v74 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:ce7da947c7c5e914eae2280 6e9f241ceb83a56a93ebcae5588c99b18c38636ce
2025-12-02T13:36:28… START RequestId: c8aaa0b9-e5fd-47e4-94fe-42351aba07a8 Version: $LATEST
2025-12-02T13:36:28… Error: An error occurred (AccessDenied) when calling the GetObject operation: User: arn:aws:sts::652968820244:assumed-role/ProcessingLambdaRole/cloudlens-processor is not authorized to perform: s3:Li…
2025-12-02T13:36:28… END RequestId: c8aaa0b9-e5fd-47e4-94fe-42351aba07a8
2025-12-02T13:36:28… REPORT RequestId: c8aaa0b9-e5fd-47e4-94fe-42351aba07a8 Duration: 81.52 ms Billed Duration: 685 ms Memory Size: 512 MB Max Memory Used: 101 MB Init Duration: 603.23 ms
2025-12-02T13:36:50… START RequestId: ec0e36d0-a2e7-4dd1-9337-5d6a6e959001 Version: $LATEST
2025-12-02T13:36:50… Processing: thumbnail/d226ea58-69c1-4423-a0dd-9f5873f2dd83_92990046cfec5c6e752ad3507869dea5.jpg | Action: thumbnail | Meta: {'custom-size': '300'}
2025-12-02T13:36:50… ⚠️ UNSAFE CONTENT DETECTED: thumbnail/d226ea58-69c1-4423-a0dd-9f5873f2dd83_92990046cfec5c6e752ad3507869dea5.jpg
2025-12-02T13:36:50… END RequestId: ec0e36d0-a2e7-4dd1-9337-5d6a6e959001
2025-12-02T13:36:50… REPORT RequestId: ec0e36d0-a2e7-4dd1-9337-5d6a6e959001 Duration: 348.19 ms Billed Duration: 349 ms Memory Size: 512 MB Max Memory Used: 103 MB
2025-12-02T13:38:06… START RequestId: 732affdb-f0b8-4972-8143-87080d4845ec Version: $LATEST
2025-12-02T13:38:06… Processing: thumbnail/1477c2e9-8924-4d09-be7b-3c89b96b42e3_step-03-buckets-created.png | Action: thumbnail | Meta: {'custom-size': '300'}
2025-12-02T13:38:07… ✅ Image Safe. Proceeding to task.
2025-12-02T13:38:07… Creating Thumbnail size: 300px
2025-12-02T13:38:07… END RequestId: 732affdb-f0b8-4972-8143-87080d4845ec
2025-12-02T13:38:07… REPORT RequestId: 732affdb-f0b8-4972-8143-87080d4845ec Duration: 806.27 ms Billed Duration: 807 ms Memory Size: 512 MB Max Memory Used: 121 MB
2025-12-02T13:38:09… START RequestId: 25cfa5ec-23df-4bc7-bcdd-e09ecbcfb528 Version: $LATEST
2025-12-02T13:38:09… Processing: thumbnail/133a94b7-6ce1-4eb5-8407-1b12281d75d4_pool-party-91-070825-e72f474fe6754480a92b69fc29b104ea.jpg | Action: thumbnail | Meta: {'custom-size': '300'}
2025-12-02T13:38:09… ⚠️ UNSAFE CONTENT DETECTED: thumbnail/133a94b7-6ce1-4eb5-8407-1b12281d75d4_pool-party-91-070825-e72f474fe6754480a92b69fc29b104ea.jpg
2025-12-02T13:38:09… END RequestId: 25cfa5ec-23df-4bc7-bcdd-e09ecbcfb528
2025-12-02T13:38:09… REPORT RequestId: 25cfa5ec-23df-4bc7-bcdd-e09ecbcfb528 Duration: 260 ms Billed Duration: 260 ms Memory Size: 512 MB Max Memory Used: 121 MB
2025-12-02T13:39:46… START RequestId: a97f8b0d-1530-4213-881b-6dc307e01659 Version: $LATEST
2025-12-02T13:39:46… Error: An error occurred (AccessDenied) when calling the GetObject operation: User: arn:aws:sts::652968820244:assumed-role/ProcessingLambdaRole/cloudlens-processor is not authorized to perform: s3:Li…
2025-12-02T13:39:46… END RequestId: a97f8b0d-1530-4213-881b-6dc307e01659
2025-12-02T13:39:46… REPORT RequestId: a97f8b0d-1530-4213-881b-6dc307e01659 Duration: 74.06 ms Billed Duration: 75 ms Memory Size: 512 MB Max Memory Used: 121 MB
2025-12-02T13:41:35… START RequestId: f59de2f3-2bd3-4077-a327-f3bcb7514542 Version: $LATEST
2025-12-02T13:41:35… Error: An error occurred (AccessDenied) when calling the GetObject operation: User: arn:aws:sts::652968820244:assumed-role/ProcessingLambdaRole/cloudlens-processor is not authorized to perform: s3:Li…
2025-12-02T13:41:35… END RequestId: f59de2f3-2bd3-4077-a327-f3bcb7514542
2025-12-02T13:41:35… REPORT RequestId: f59de2f3-2bd3-4077-a327-f3bcb7514542 Duration: 67.03 ms Billed Duration: 68 ms Memory Size: 512 MB Max Memory Used: 121 MB
No newer events at this moment. Auto retry paused. Resume
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

# 9. MEMBER CONTRIBUTIONS

- **Member 1 (Nitesh): Backend: Moderation & Logic**
  - Primary Responsibilities:
    - Develop ModerationProcessor Lambda (Python)
    - Integrate Amazon Rekognition (DetectModerationLabels)
    - Implement safe/unsafe decision logic
    - Log all moderation results to CloudWatch Logs

- **Member 2 (Swara): Infrastructure, Security & Monitoring**
  - Primary Responsibilities:
    - Define all IAM Roles & Policies
    - Create all S3 Buckets
    - Build CloudWatch Dashboard for metrics
    - Compile final report & demo video

- **Member 3 (Kishan): Backend: Image Processing**
  - Primary Responsibilities:
    - Build and attach the Pillow Lambda Layer
    - Implement the thumbnail generation logic
    - Implement the watermark generation logic
    - Save processed images to appropriate S3 buckets

- **Member 4 (Stuti): Cloud-Integrated Frontend & API**
  - Primary Responsibilities:
    - Configure S3 Static Website Hosting
    - Set up API Gateway endpoint & GetUploadUrlLambda
    - Implement all CORS configuration (S3 & API-G)

# 10. CONCLUSION

The CloudLens project successfully demonstrates how serverless technologies on AWS can be combined to build a secure, scalable, and cost-efficient image moderation and processing pipeline. By leveraging services such as Amazon S3, AWS Lambda, Amazon Rekognition, and API Gateway, the system eliminates the need for traditional server management while providing real-time, automated content moderation and image processing. The integration of presigned URLs ensures secure uploads from untrusted clients, and the use of Pillow within a Lambda Layer enables efficient generation of thumbnails and watermarked images.

Throughout the implementation, core cloud principles including event-driven design, least-privilege IAM security, automatic scaling, and detailed monitoring via CloudWatch were applied effectively. The project remained within AWS Free Tier constraints, reinforcing its cost-optimized architecture.

Overall, CloudLens not only fulfills the problem requirement of moderating and transforming user-generated content but also showcases practical industry-level cloud design patterns. The experience enabled the team to gain hands-on understanding of serverless workflows, distributed cloud components, security enforcement, and operational monitoring. This project lays a solid foundation for future enhancements such as user authentication, advanced image processing, database integration, and automated CI/CD pipelines.

CLICK HERE TO GO TO **CLOUDLENS**