# Automated Image Processing and Content Deliver on AWS

August 10, 2025

## Architecture Overview

- **Raw Images S3 Bucket** – Stores original uploaded images
- **AWS Lambda** – Resizes and watermarks images
- **Processed Images S3 Bucket** – Stores processed images
- **DynamoDB** – Stores metadata (filename, size, timestamp)
- **CloudFront** – Distributes processed images globally
- **API Gateway** – Optional trigger for manual processing

## Architecture Diagram

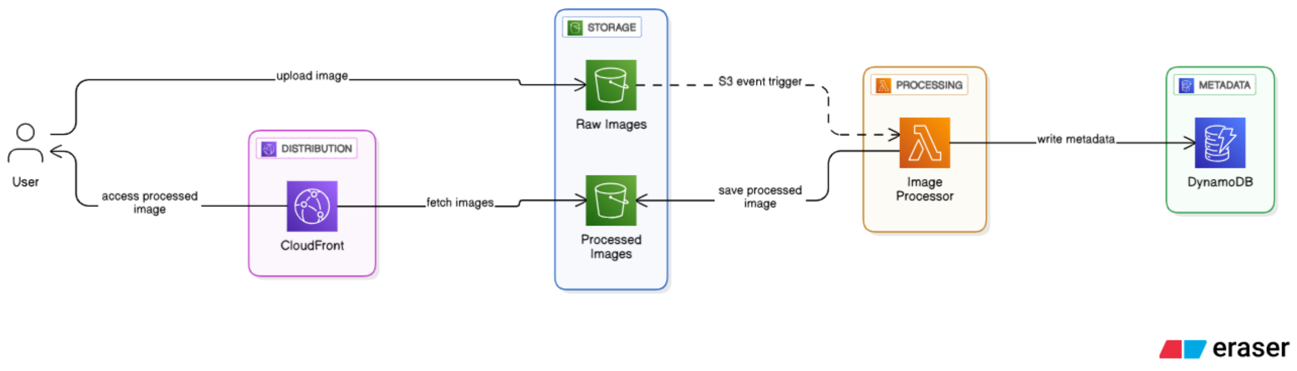

Figure 1: Automated Image Processing and Content Delivery on AWS

## Implementation Steps

### 1. Create S3 Buckets

- **raw-image-bucket** for source Bucket and **processed-image-bucket** for Destination Bucket
- **Enable** Block All Public Access
- Configure bucket policies in destination bucket for CloudFront(OAI) access

### 2. Create DynamoDB Table

- go to Aws console → Search DynamoDB → Create Table
- Table name: `ImageMetadata`
- Partition key: `filename` (String)
- Create Table

### 3. IAM Role Configuration

| **Component** | **Purpose** |
|---|---|
| Go to AWS Console | Search IAM ROLE → Create Role |
| Attached Policies | • AmazonS3FullAccess (read/write both buckets)<br>• AmazonDynamoDBFullAccess (metadata storage)<br>• CloudWatchLogsFullAccess (debugging) |
| Role Name | LambdaImageProcessingRole |
| Critical Need | Without this role, Lambda cannot access S3, DynamoDB, or logs |

### 3. Lambda Function Setup

- Go to AWS Console → Search Lambda → Create Lambda
- Name: `ImageProcessingFunction`
- Select Runtime: Python 3.12
- Execution role: `LambdaImageProcessingRole`
- Scroll down in Lambda Function Attach Pillow Layer (ARN for us-east-1):

```
arn:aws:lambda:us-east-1:770693421928:layer:Klayers-p312-Pillow:6
```

- Attach a Python Code in Lambda Code Section
- Go to Test Section In Lambda Function And add a Json Policy and the click Test.

### 4. Configure Triggers & Distribution

- S3 Event Trigger: PUT on `raw-image-bucket`
- raw-image-bucket = your source bucket name
- processed-image-bucket = your destination bucket name

### 5. Configure CloudFront

- Go to Aws Console → Search CloudFront → Create distribution
- Attach a distributiton name
- Select Origin of Destination Bucket: **processed-image-bucket.s3.amazonaws.com**
- Set Redirect HTTP to HTTPS
- Create Distribution Go to S3 destination Bucket and Upload an **index.html** file in the destination bucket for testing

## Testing Pipeline

1. Upload `photo.jpg` to raw bucket
2. Lambda processes → creates `processed-photo.jpg`
3. Metadata stored in DynamoDB
4. Access via CloudFront URL

✓ **Result:** Fully functional serverless pipeline with automatic resizing, watermarking, metadata storage and global distribution.