

Problem Statement: Create a lambda function which will analyze the image stored in S3 bucket and return matching results using the amazon service Rekognition.

Scenario 1: The first scenario is just uploading the images and test the code manually

Scenario 2: Automatically detect labels from images uploaded to a specific S3 **input folder** and write the label results to a **CSV file** in an **output folder**.

Amazon Service:

1. Amazon Rekognition

Pre-requisites:

1. Amazon Free Tier Account.
2. Knowledge on how to create IAM users and add permissions.
3. Python Code
4. Image in S3 bucket

Architecture Overview:

1. S3 Bucket stores the image
 - a. Input Folder Name – input
 - b. Output Folder Name - output
2. Lambda function is triggered with the image details (manually)
3. Amazon Rekognition to analyze the image
4. Output is a list of detected labels in csv format

Steps:

1. Create an S3 bucket
2. Select General Purpose
3. Provide a unique bucket name aws_rekognition
4. Object ownership – Select ACLs disabled
5. Uncheck Block public access settings for this bucket
6. Check the box “I acknowledge”
7. Leave other settings as default
8. Click on create bucket
9. Will use .jpg or .jpeg images for the demo

10. Upload the rose image into the bucket
11. The image will be available in the bucket
12. Create a role with the following permissions and name the role
 - a. **AmazonRekognitionFullAccess**
 - b. **AmazonS3ReadOnlyAccess**
13. After IAM role creation, create a lambda function
14. Click on Create a function
15. Select Author from Scratch
16. Enter a function name
17. Select runtime as Python
18. Click on Use an Execution Role
19. Select the IAM Role Created
20. Leave other settings as default
21. Click on Create Function
22. In the code tab, paste the python code
23. Refer the below screenshots

S3 Bucket Creation

Create bucket Bucket are containers for data stored in S3.

General configuration

AWS Region: US East (N. Virginia) aws-us-east-1

Bucket type: Standard

General purpose buckets are the original S3 buckets type. They allow all AWS services to access your data.

Directory Recommended for inconsistency-free access. These buckets use only the S3 Express File Sync storage class, which provides faster processing of file uploads and higher durability.

Bucket name: Bucket names must start longer and end with a letter or number. Valid characters are a-z, 0-9, periods (.), underscores (_), and hyphens (-).

Copy settings from existing bucket - optimus This will copy the bucket's existing configuration over.

Choose bucket Choose a different bucket to copy its configuration over.

Versioning: Enable versioning

Object Ownership Control ownership of objects written to this bucket. Errors occur: AWS accounts and the use of access-control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (Recommended) All objects in this bucket are owned by this account. Access to this bucket and its objects is restricted using policy statements.

ACLs enabled Objects in this bucket can be owned by other AWS accounts. Access to this bucket and the objects can be specified using ACLs.

Block Public Access settings for this bucket Block public access settings control who can access your objects. By default, you can block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require more granular access control, use AWS Identity and Access Management (IAM) policies.

Block all public access Turns off public access to all objects in this bucket. Turn on the following settings if you want to use static website hosting:

- Block public access to buckets and objects granted through new access control lists (ACLA)
- Block public access to buckets and objects granted through legacy public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACAs.
- Block public access to buckets and objects granted through new access control lists (ACLA)
- Block public access to buckets and objects granted through legacy public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACAs.
- Block public and cross-account access to buckets and objects through static public buckets or static website host policies.
- Block public and cross-account access to buckets and objects through static public buckets or static website host policies.

Turning off block all public access might result in this bucket and the objects within becoming public. We recommend that you use AWS Identity and Access Management (IAM) policies to restrict access to your objects such as static website hosting.

I understand that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning Versioning is a feature of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from accidental deletes, edits, and application failures.

Enabled Enabled

Tags - optional You can use Bucket tags to track storage costs and compliance buckets.

No tags associated with this bucket.

Add tag

Default encryption Encryption is automatically applied to new objects stored in this bucket.

Encryption type: Server-side encryption with Amazon S3-managed keys (SSE-S3)

Server-side encryption with AWS KMS-managed keys (SSE-KMS)

Client-side encryption with AWS KMS-managed keys (SSE-KMS)

Dual-layer server-side encryption with AWS Key Management Service keys (DKE-SSE-KMS)

Important: AWS Lambda functions with two execution roles of an account. You receive an warning, and double-clicking on this message will take you to the AWS Lambda console.

Bucket Key Only one standard key for SSE-AES-based encryption can be assigned at a time. SSE standard keys aren't supported for DKE-SSE-KMS.

Disable Disable

> Advanced settings

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

S3 Bucket – Image Upload

The screenshot shows the AWS S3 'Files and folders' interface. It displays a single file entry: 'rose.jpg' (image/jpeg, 1.1 MB). There are buttons for 'Remove', 'Add files', and 'Add folder'. A search bar at the top says 'Find by name'.

IAM Role Creation – Lambda Function

The screenshot shows the 'Select trusted entity' step of the IAM role creation wizard. It includes sections for 'Trusted entity type' (AWS Service selected), 'Use case' (Lambda selected), and 'Service or use case' (Lambda selected). A 'Next Step' button is visible.

The screenshot shows the 'Name, review, and create' step of the IAM role creation wizard. It includes sections for 'Role details' (Role name: 'AmazonrekognitionRole'), 'Step 1: Select trusted entities' (JSON policy document), 'Step 2: Add permissions' (Permissions policy summary: 'AmazonrekognitionFullAccess' and 'AmazonSShareDocuments'), and 'Step 3: Add tags' (Add tags optional). A 'Create role' button is at the bottom right.

Create a Lambda Function

Create function info

Create one of the following options to create your function.

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presents for common use cases.

Container image
Select a container image to deploy for your function.

Basic information

Function name
 lambda-rekognition-demo
Function name must be 3 to 128 characters, must be unique in the Region, and can't include spaces. Valid characters are a-zA-Z, 0-9, hyphen (-), and underscore (_).

Runtime info
The runtime language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
 Python 3.15

Architecture info
The execution environment architecture you want for your function code:
 x86_64
 arm64

Permissions info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose the role that defines the permissions of your function. To create a custom role, go to the IAM console [\[IAM\]](#).

Create a new role with AWS Lambda permissions

Create a new role from AWS policy templates

Existing role
Select the role that you created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
 View the Wlsh_Policy_Role (2016472340052) role [View] on the IAM console.

Additional Configurations

Enable Code signing info
AWS Lambda automatically signs your Lambda function's code before it runs. This prevents anyone from injecting malicious code into your Lambda function.

Enable Lambda layers info
AWS Lambda layers are reusable components for your Lambda functions.

Enable KMS encryption with AWS Lambda customer managed key info
By default, Lambda encrypts the file that archive using an AWS owned key.

Enable function URL info
A function URL provides a direct endpoint for your Lambda function.

Enable tags info
Add tags to your Lambda function to identify it in AWS resources. Each tag consists of a key and an optional value. You can use tags for search and filter your resources, track your AWS costs, and enforce attribute-based access control.

Enable VPC info
Connect your function to a VPC to access private resources using an endpoint.

Create Function Cancel

Scenario 1 - Python Code – Upload the code and click test the output

```
import json # module is used to work with JSON data

import boto3 # Python SDK - allows Python to talk to AWS Services such as S3, Lambda

def lambda_handler(event, context): # function manually triggered

    bucket_name = "Enter your bucket name" # Enter the bucket name

    image_obj_name = "rose.jpeg" # Enter the image name along with the extension exact filename / This is manual method

    try:
        rkClient = boto3.client("rekognition", region_name="Enter your region") # Creates a connection with the AWS Service and calls the service to analyze the image and detect objects
    except:
        rkResponse = rkClient.detect_labels(
            Image={

                'S3Object':{

                    'Bucket': bucket_name,
                    'Name': image_obj_name
                }
            },
        )

        print(rkResponse['Labels']) # contains the list of things the services see such as Flower, Nature , Plant

        return rkResponse['Labels'] # Prints the labels

    except Exception as e:
        print("Get labels failed because ",e)

    except Exception as e:
        print("Client connection to Rekognition failed because ",e)
```

Scenario 2: Automatically detect labels from images uploaded to a specific S3 **input folder** and write the label results to a **CSV file** in an **output folder**.

rekognition.07042025 [Info](#)

Objects | Metadata | Properties | Permissions | Metrics | Management | Access Points

Objects (2)

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

Name	Type	Last modified	Size	Storage class
input/	Folder	-	-	-
output/	Folder	-	-	-

input/ [Copy S3 URI](#)

Objects (2)

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

Name	Type	Last modified	Size	Storage class
sunflower.jpeg	jpeg	April 7, 2025, 18:53:16 (UTC+05:30)	12.5 KB	Standard
tulip.jpeg	jpeg	April 7, 2025, 18:53:16 (UTC+05:30)	5.4 KB	Standard

output/ [Copy S3 URI](#)

Objects (1/0)

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

Name	Type	Last modified	Size	Storage class
------	------	---------------	------	---------------

No objects
You don't have any objects in this folder.

[Upload](#)

Paste the following code in the lambda code editor and deploy the code

```
import boto3

import csv

from io import StringIO

# Initialize AWS clients
s3 = boto3.client('s3')
rekognition = boto3.client('rekognition')

def lambda_handler(event, context):
    # Hardcoded values for simplicity
    bucket = "rekognition.07042025"      # Replace with your actual bucket name
```

```

input_folder = "input/"

output_key = "output/labels.csv"

print("Bucket:", bucket)
print("Input folder:", input_folder)
print("Output file:", output_key)

# Step 1: List image files in the input folder

image_keys = []

response = s3.list_objects_v2(Bucket=bucket, Prefix=input_folder)

if 'Contents' not in response:
    return {
        'statusCode': 404,
        'body': 'No files found in input folder.'
    }

for obj in response['Contents']:
    key = obj['Key']
    if key.endswith('.jpg', '.jpeg', '.png') and not key.endswith('/'):
        image_keys.append(key)

if not image_keys:
    return {
        'statusCode': 404,
        'body': 'No supported image files found in input folder.'
    }

print(f"Found {len(image_keys)} image(s): {image_keys}")

# Step 2: Prepare CSV output in memory

csv_buffer = StringIO()

writer = csv.writer(csv_buffer)

writer.writerow(['Image Name', 'Label', 'Confidence (%)'])

# Step 3: Detect labels for each image using Rekognition

for image_key in image_keys:
    try:
        rekog_response = rekognition.detect_labels(
            Image={'S3Object': {'Bucket': bucket, 'Name': image_key}},
            MaxLabels=10,
            MinConfidence=75
    
```

```

    )

for label in rekog_response['Labels']:

    writer.writerow([
        image_key.split('/')[-1],
        label['Name'],
        round(label['Confidence'], 2)
    ])

except Exception as e:

    print(f"Error processing {image_key}: {str(e)}")

# Step 4: Upload CSV to the output folder

try:

    s3.put_object(
        Bucket=bucket,
        Key=output_key,
        Body=csv_buffer.getvalue(),
        ContentType='text/csv'
    )

    print(f"CSV uploaded to s3://{bucket}/{output_key}")

except Exception as e:

    print(f"Error uploading CSV: {str(e)}")

    return {
        'statusCode': 500,
        'body': f"Failed to upload CSV: {str(e)}"
    }

return {
    'statusCode': 200,
    'body': f"CSV file with labels saved at s3://{bucket}/{output_key}"
}

```

After deploying, test the code and in the output folder a csv file will be created.

Event JSON

```
1 [{}  
2   "bucket": "rekognition.07042025",  
3   "input_folder": "input",  
4   "output_key": "output/labels.csv"  
5 ]  
6
```

✓ Executing function: succeeded ([logs](#))

▼ Details

```
{  
  "statusCode": 200,  
  "body": "CSV file with labels saved at s3://rekognition.07042025/output/labels.csv"  
}
```

Objects (1)						
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						
Name		Type	Last modified	Size	Storage class	Actions
labels.csv	labels.csv	csv	April 7, 2025, 19:14:58 (UTC+05:30)	266.0 B	Standard	Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

A	B	C	D
Image Name	Label	Confidence (%)	
sunflower.jpeg	Flower	100	
sunflower.jpeg	Plant	100	
sunflower.jpeg	Sunflower	100	
tulip.jpeg	Flower	100	
tulip.jpeg	Petal	100	
tulip.jpeg	Plant	100	
tulip.jpeg	Tulip	98.47	
tulip.jpeg	Food	94.9	
tulip.jpeg	Ketchup	94.9	