26 April 2023

Using AWS for Image Recognition

Michael Garcia
Daniel Garza

CIS 4355-01
Spring 2023

EXECUTIVE SUMMARY

Objective

Our goal is to create a system that can accurately generate a series of tags about an image that is given, and send it to a mobile phone via email. Additionally, we would like to use this project to explore the various services that AWS has to offer.

Background

Image generation/recognition is becoming increasingly popular with AI models such as ChatGPT/DALL-E/HuggingGPT. With this project, we wanted to attempt to create a pseudo-build of an AI image recognition model solely utilizing various services within AWS, as it has its own model that can recognize images and characteristics.

AWS Services

- IAM - sets permissions for Lambda
- S3 - stores images to be labeled by Rekognition
- Rekognition - identifies labels to associate to image
- Lambda - hosts core python code
- SQS - works in conjunction with SNS to deliver notifications
- SNS - sends notification with labels via email
- DynamoDB - stores labels produced by system

Challenges & Limitations

- Learning to integrate multiple AWS services into python code
- Unable to use SNS for text messaging due to law requiring usage of toll-free number
- Code deployment required before testing, resulting in constant rollbacks
- DynamoDB caused many issues with integrating into our system
    - Lack of experience with NoSQL

Conclusion

While our image recognition system can only provide basic labels from the images it is given, it could be improved to recognize more specific details with AutoML, such as the make and model of a car. In addition, it was somewhat difficult to work within the bounds of the AWS Free Tier, as a lot of the services required payment for resource use/extra features.

Project Milestones:

1. Research various AWS services to incorporate into the image recognition system.
2. Link the various services to create a rough draft of the system through coding them together.
3. Create a working prototype that can recognize images and return descriptors.
4. Employ the Amazon SQS service to send the descriptors to a mobile device.

Deliverables:

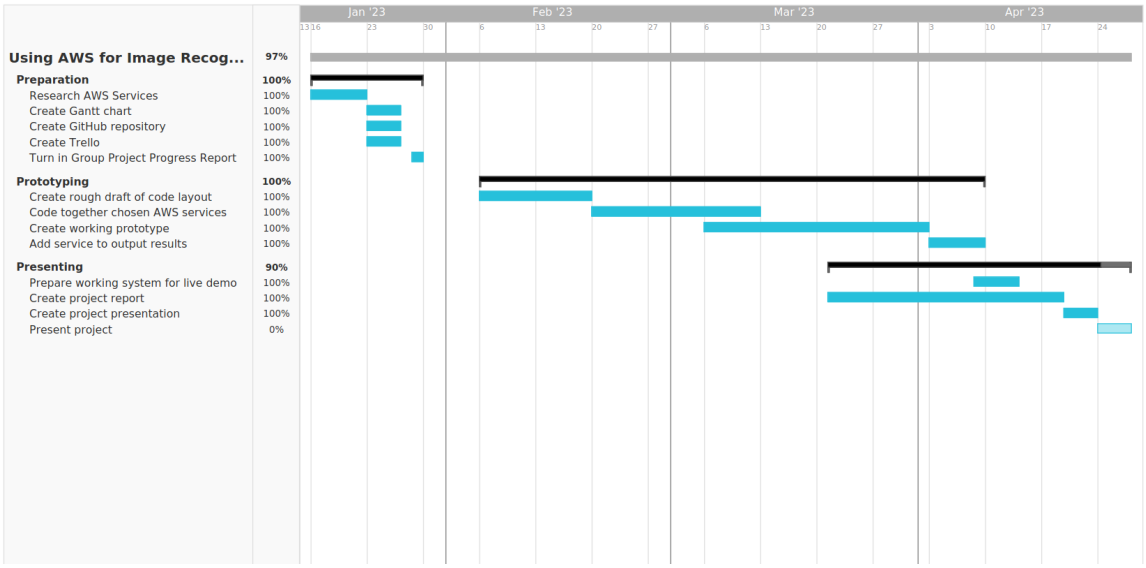
1. Project Progress Report
2. Functional Code
3. Final Report & Presentation

Professional Accomplishments:

1. Knowledge of AWS service integration with python into Lambda.
2. Knowledge of AWS service applications.
3. Understanding of cloud infrastructure overall.

PROJECT SCHEDULE MANAGEMENT

Gantt Chart:



Trello Board:
[redacted]

GitHub Repository:
[redacted]

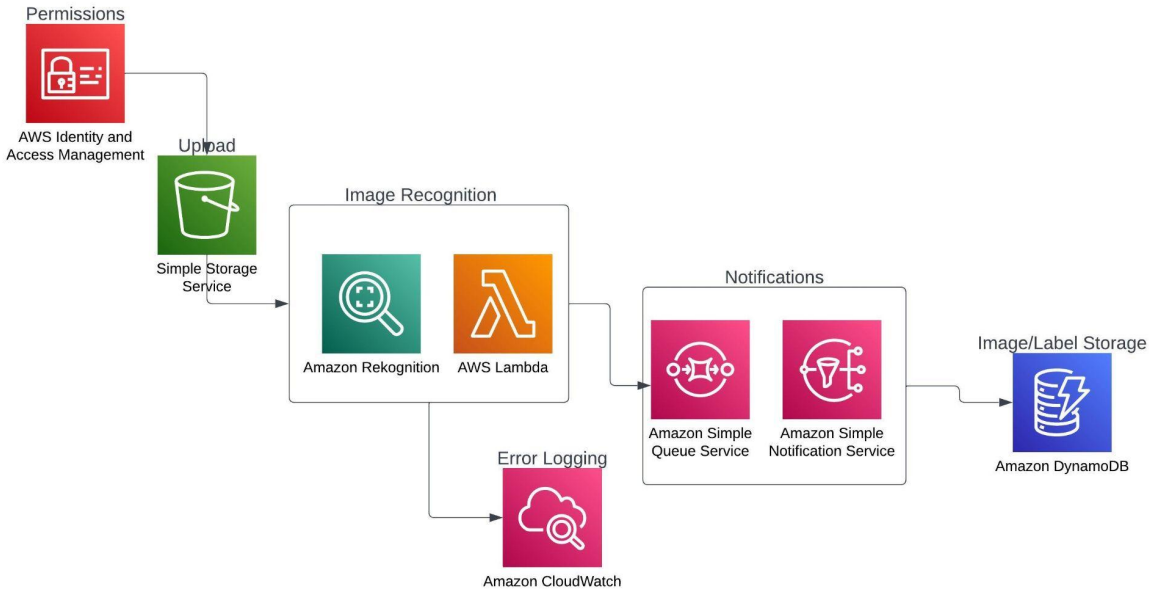
AWS Architecture

Overview

For this project, we utilized several different AWS services. Before we started, we set permissions for our user account to be used with the project via the IAM service. We then created a S3 bucket to upload photos to be temporarily stored beforebeing scanned by our Lamda/Rekognition python environment. Once this scan was completed, SNS and SQS worked in conjunction to send an email to a subscribed email address with prediction labels as to what the image contains. After this system completes its cycle, DynamoDB stores each of these labels produced by this system and categorizes them by the image name.

Cloud Architecture

Below is a diagram outlining the flow of data within our image recognition system. Each service is labeled and grouped according to how it was utilized in our AWS enviroment.



Service Integration

IAM Role Permissions

The following is a screenshot of our initial IAM role permissions. As you can see, it is insecure because it allows full access to every service used in our project. This was created as a proof of concept and is not indicative of what a role would look like in a production environment.

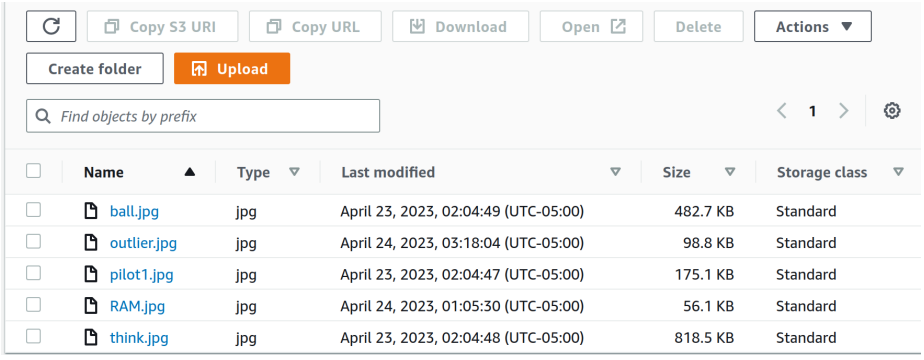
Policy name	Type	Description
AmazonSQSFullAccess	AWS managed	Provides full access to Amazon SQS via the AWS Ma…
AmazonS3FullAccess	AWS managed	Provides full access to all buckets via the AWS Mana…
CloudWatchFullAccess	AWS managed	Provides full access to CloudWatch.
AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon DynamoDB via the A…
AmazonRekognitionFullAccess	AWS managed	Access to all Amazon Rekognition APIs
AWSLambdaBasicExecutionR…	AWS managed	Provides write permissions to CloudWatch Logs.
AmazonSNSFullAccess	AWS managed	Provides full access to Amazon SNS via the AWS Ma…

Due to security concerns, we updated the role to only allow the required access for our system to function. This improved role allows our project to run without giving it undue access and increases overall security.

Policy name	Type	Description
AmazonRekognitionReadOnlyAccess	AWS managed	Access to all Read rekognition
AmazonSQSReadOnlyAccess	AWS managed	Provides read only access to A
AmazonS3ReadOnlyAccess	AWS managed	Provides read only access to a
AWSLambdaSQSQueueExecutionRole	AWS managed	Provides receive message, del
AWSLambdaBasicExecutionRole	AWS managed	Provides write permissions to C
DynamoProjectPolicy	Customer inline	
SNSProjectPolicy	Customer inline	

S3 Image Upload

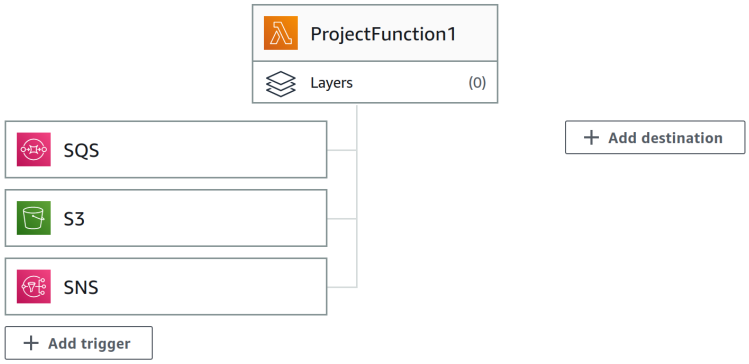
For this project, we used an S3 bucket as a repository for images to be processed by the Lambda function and Rekognition service.



Name	Type	Last modified	Size	Storage class
ball.jpg	jpg	April 23, 2023, 02:04:49 (UTC-05:00)	482.7 KB	Standard
outlier.jpg	jpg	April 24, 2023, 03:18:04 (UTC-05:00)	98.8 KB	Standard
pilot1.jpg	jpg	April 23, 2023, 02:04:47 (UTC-05:00)	175.1 KB	Standard
RAM.jpg	jpg	April 24, 2023, 01:05:30 (UTC-05:00)	56.1 KB	Standard
think.jpg	jpg	April 23, 2023, 02:04:48 (UTC-05:00)	818.5 KB	Standard

Lambda & Rekognition

We added triggers to our Lambda as seen below. Once an object is uploaded to our S3 bucket, the function is triggered and the image is processed through Rekognition. The SQS and SNS triggers work in conjunction to notify the user that image processing has been completed.

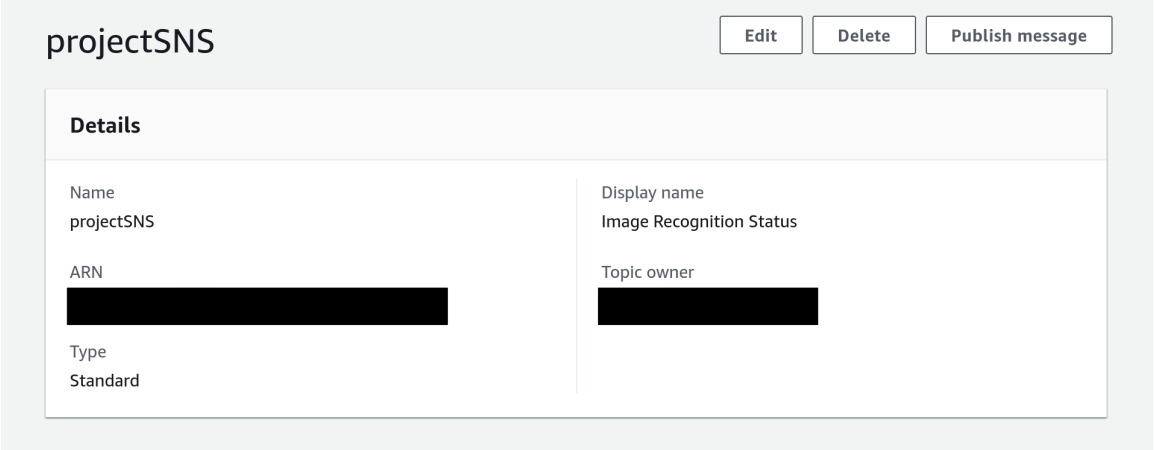


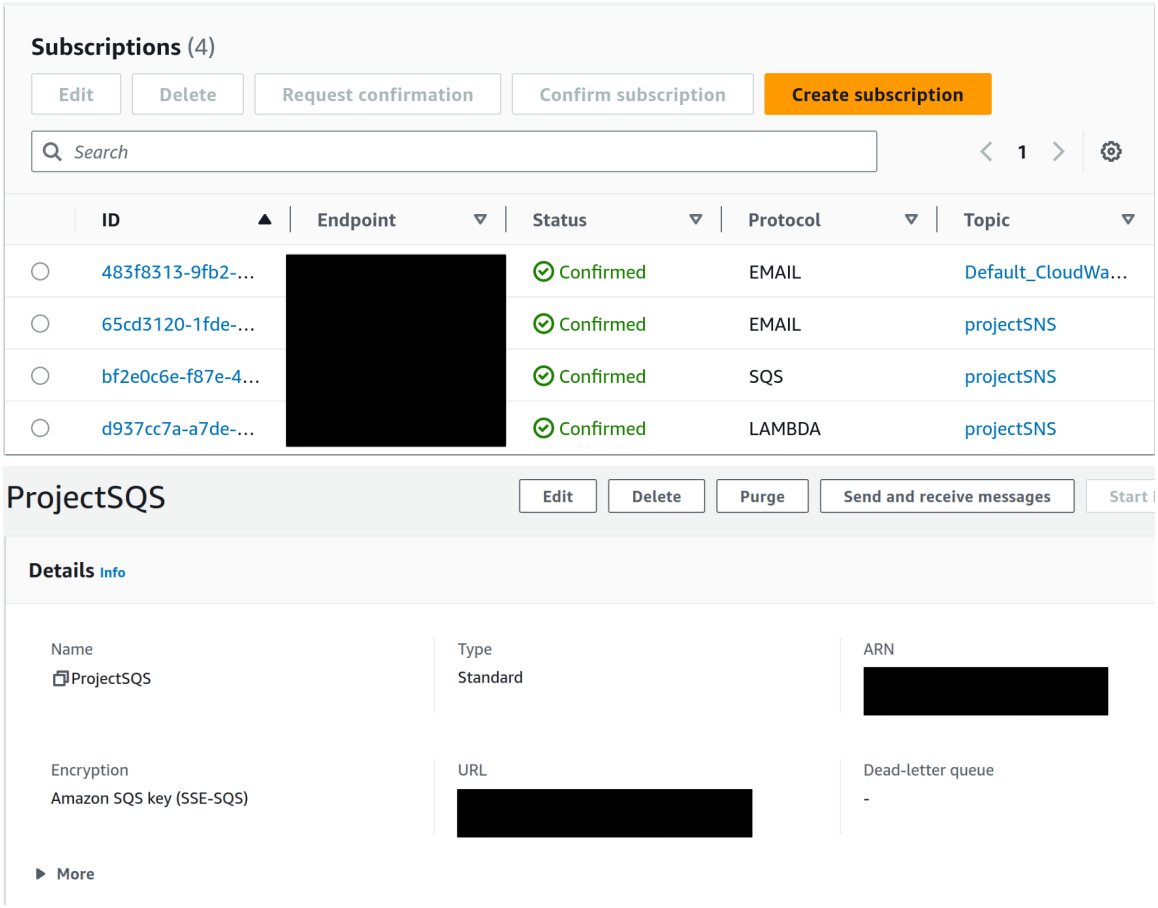
In our code, we used the boto3 and uuid libraries. UUID is used specifically to generate a random image ID for the DynamoDB. This is required to denote each image.

```
import boto3
import uuid

s3_client = boto3.client('s3')
rekognition_client = boto3.client('rekognition')
sqs_client = boto3.client('sqs')
sns_client = boto3.client('sns')
dynamodb_client = boto3.client('dynamodb')

def lambda_handler(event, context):
    # Retrieve S3 bucket and object information from the event
    s3_bucket = event['Records'][0]['s3']['bucket']['name']
    s3_object_key = event['Records'][0]['s3']['object']['key']

    try:
        image = get_s3_image(s3_bucket, s3_object_key)

        labels = detect_labels(image)

        process_labels(labels, s3_object_key)

        message = '\n'
        for label in labels:
            message += f'Label: {label["Name"]}, Confidence: {label["Confidence"]}%\n'
```

SNS & SQS

For our notification delivery process, we utilized both SNS and SQS topics. Additionally, they show how the subscriptions work to send an email notification.





DynamoDB Storage

The following is a screenshot of our DynamoDB table. The table receives the output from the image processing and contains the labels created by the Rekognition service. The primary key is “image_id” and is randomly generated by the code.

image_id	confidence	label	object_key
d64cb3fb-9af3-47ab-...	99.9999847...	Truck	RAM.jpg
62ab538e-e4fb-4383...	99.9999847...	Pickup Truck	RAM.jpg
7439524a-5d35-408c...	95.6109008...	American F...	ball.jpg
4848d1ae-0c84-4b1a...	95.6109008...	Sport	ball.jpg
a2afaa7a-40ad-4cea-...	95.6109008...	Football	ball.jpg
48eae29a-bf83-46f1-...	99.9999847...	Vehicle	RAM.jpg

Final Thoughts

Although there are various types of image recognition models being used in public domain, our model's significance is that we were able to create it using free resources from AWS. While it only provides basic labels for images, an improvement we could make would be to enhance it to identify more precise details within an image, such as the make and model of a car, using additional services such as AutoML.

Despite challenges such as integrating multiple AWS services into our python code, learning about legal restrictions on using SNS for text messaging, and the constant need for code deployment and rollbacks to test it, we were able to build a functioning system. Not to mention the many issues we encountered with integrating DynamoDB into our system, mostly due to our lack of experience with NoSQL.

Working within the limits of the AWS Free Tier proved challenging as many services required payment for extra features and resource usage. However, the knowledge we are taking away from this project about cloud infrastructure makes the hardships we faced valuable in the end.

References

Amazon DynamoDB. (n.d.). Retrieved March 28, 2023, from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStartedDynamoDB.html

Marquez, E. (2021, November 10). *How to create an AWS lambda function*. Cloud Computing. Retrieved February 6, 2023, from https://www.techtarget.com/searchcloudcomputing/tip/How-to-create-an-AWS-Lambda-function

Musgrave, D. (2022). *Lambda*. Amazon. Retrieved February 8, 2023, from https://docs.aws.amazon.com/lambda/latest/dg/API_Reference.html

North, F. (1998). *SNS*. Amazon. Retrieved April 4, 2023, from https://aws.amazon.com/getting-started/hands-on/send-fanout-event-notifications/

North, F. (1998). *SQS*. Amazon. Retrieved April 9, 2023, from https://aws.amazon.com/getting-started/hands-on/send-messages-distributed-applications/

Prerequisite: Setting up Amazon S3 - Amazon Simple Storage Service. (n.d.). Retrieved March 2, 2023, from https://docs.aws.amazon.com/AmazonS3/latest/userguide/setting-up-s3.html

Schütz Julia. (2011). *Amazon Rekognition: Detecting Labels*. Amazon. Retrieved March 17, 2023, from https://docs.aws.amazon.com/rekognition/latest/dg/labels.html?pg=ln&sec=ft

Whitehouse-Grant-Christ, I. H. V. (2011). *IAM*. Amazon. Retrieved March 8, 2023, from https://docs.aws.amazon.com/IAM/latest/UserGuide/access.html