

CSE 546 Project - 2 Report

Arizona State University

Aditya Reddy Mali - 1224165813

Paromita Roy - 1224708511

Tejesh Reddy Sigineni - 1222183339

1. Problem Statement

This project aims to build and implement an elastic application that performs face recognition on videos which are input by the user. This application could be used in a smart classroom setting which would enable educators to look up academic information of students recognised from the video, whose details are stored in a database.

It is built using the Platform as a Service functionality provided by AWS, using features such as AWS Lambda, DynamoDB and S3.

2. Design and Implementation

2.1. Architecture

In our implementation, a user is asked to upload an MP4 file. The video is uploaded into an S3 bucket (input-bucket-video). Once the video is uploaded successfully, a Lambda function is triggered. This function is created out of an image containing the face recognition functionality, also known as the Lambda handler function. It extracts a frame from the video and stores it as an image. A facial recognition module is run on this image, which creates an encoding. The encoding is matched with a known encoding file to retrieve the name of the student. Finally, we use this name to query a DynamoDB, which contains all the relevant information of every student. Based on the query, the details retrieved are stored in CSV format and sent to another S3 bucket. This bucket serves as the output bucket (output-bucket-vid).

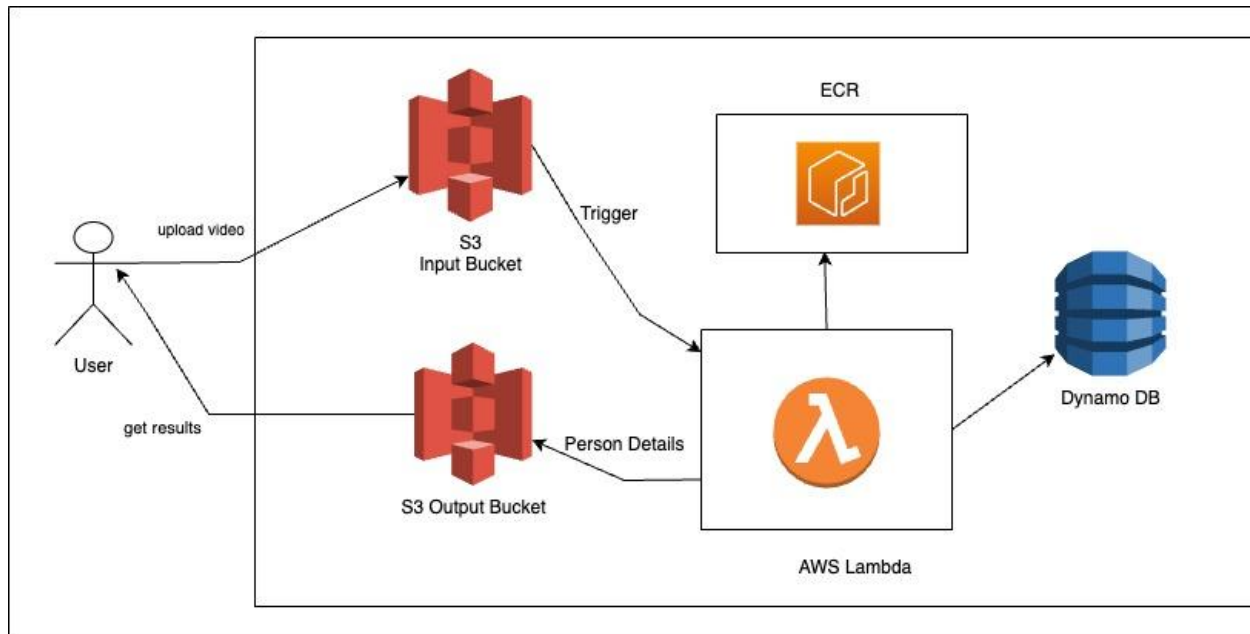


Figure 1. Architecture Diagram

2.2. Autoscaling

AWS Lambda allows you to run code without managing servers. It is a serverless computing solution. One of AWS Lambda's primary features is to automatically modify the number of instances running based on the volume of incoming requests. When autoscaling is set up for a Lambda function, AWS Lambda automatically tracks the volume of requests and modifies the number of instances running, to account for variations in traffic. In other words, as traffic grows, Lambda will automatically add more instances to handle the load, and when it reduces, Lambda will automatically delete instances to conserve resources and cut expenses.

2.3. Member Tasks

1. **Aditya Reddy Mali:** Worked on writing Python code on the Lambda function to get student records from DynamoDB. Processed these results into CSV format and stored it in the output bucket. Configured the IAM users and policies. Worked on end to end testing. Wrote the architecture section in the document.
2. **Paromita Roy:** Loaded data onto the DynamoDB with the data given to us. Set up and configured Docker. Worked on building the Docker image and pushing it to Amazon ECR. Wrote code to query the DynamoDB to compare and retrieve results. Compared the results to check if outputs are appropriate. Worked on the Testing and Evaluation part of the document.
3. **Tejesh Reddy Sigineni:** Worked on the facial recognition service. Used `face_recognition` library to detect the face in each frame and compared the

detected face's encoding to the known face encodings to determine if a match existed. Used OpenCV library to read the video file and display the output. It took the videos from the input S3 bucket and processed them. Compared it with the encoding file given to us to check results. Wrote on the Code and Outputs section in the document.

3. Testing and Evaluation

To test this application, we used the workload generator provided to us. These are the steps followed:

1. Verify that both S3 buckets are initially empty.
2. Run the following command to upload the MP4 files:

```
python3 workload.py
```

3. Verify that the number of files in the input bucket are equal to the number of videos uploaded by the user.
4. Check the logs to ensure that the Lambda function is being triggered as soon as the input bucket starts getting populated.
5. Check the output S3 bucket to see if results are being populated.
6. Once the execution of the program has been completed, verify that the number of objects in the output bucket is equal to the number of videos initially uploaded by the user.
7. Check each object in the output bucket to confirm that the CSV files contain the right details.

4. Code

The files involved in this application are as follows:

1. **Dockerfile:** The dockerfile given to us required some changes. We included the upload of encoding in this file.
2. **Entry.sh:** This file ensures that the program runs in the appropriate directory.
3. **Handler.py:** This is the entry point when the lambda function kicks in. It contains code responsible for the facial recognition on the file uploaded to the S3 bucket. It also queries the DynamoDB to fetch results. It runs an ffmpeg command which extracts a frame from the video and converts it to jpeg format. Then, an encoding is created and is compared with the encoding file given to us. This returns the name associated with the person in the image. Finally, the output is converted to CSV format and is pushed into the output S3 bucket.
4. **Mapping:** This file contains the mapping for every MP4 file. It gives us information about every tag associated with each video.

5. **Workload.py**: This file has functions which contain code to clear both the input and output buckets when the program initially starts. It also pushes the user uploaded MP4 files into the input S3 bucket.
6. **ecr_push.sh**: Created a script to run all commands necessary to log into, build and tag the created image, and push it to the ECR repository. This avoids running multiple commands repeatedly.

Steps to create the environment and run the program:

1. Set up docker on your system.
2. Retrieve an authentication token and authenticate your Docker client to your registry.

```
aws ecr get-login-password --region us-east-1 | docker login --username
AWS --password-stdin 704676190155.dkr.ecr.us-east-1.amazonaws.com
```

3. Create an image containing using the Docker file to include files from the directory.

```
docker build -t smart-classroom .
docker tag smart-classroom:latest
704676190155.dkr.ecr.us-east-1.amazonaws.com/smart-classroom:latest
```

4. Push the image to Amazon ECR.

```
docker push
704676190155.dkr.ecr.us-east-1.amazonaws.com/smart-classroom:latest
```

5. Update the image URI on the Lambda function to reflect the uploaded *handler.py*.
6. Run workload generator with the following command:

```
python3 workload.py
```

7. Check the input bucket and verify that the videos are being uploaded successfully. Verify that the number of objects in the bucket is equal to the number of videos being uploaded.
8. Check the output bucket and verify the results are being accurately labelled.

5. Outputs and Screenshots

```
(base) → smart-classroom-face-recognition git:(main) x
'Contents'
Nothing to clear in input bucket
Nothing to clear in output bucket
Running Test Case 1
Uploading to input bucket.. name: test_0.mp4
Uploading to input bucket.. name: test_1.mp4
Uploading to input bucket.. name: test_2.mp4
Uploading to input bucket.. name: test_6.mp4
Uploading to input bucket.. name: test_7.mp4
Uploading to input bucket.. name: test_5.mp4
Uploading to input bucket.. name: test_4.mp4
Uploading to input bucket.. name: test_8.mp4
Running Test Case 2
Uploading to input bucket.. name: test_36.mp4
Uploading to input bucket.. name: test_22.mp4
Uploading to input bucket.. name: test_0.mp4
Uploading to input bucket.. name: test_1.mp4
Uploading to input bucket.. name: test_23.mp4
Uploading to input bucket.. name: test_37.mp4
Uploading to input bucket.. name: test_21.mp4
Uploading to input bucket.. name: test_35.mp4
Uploading to input bucket.. name: test_2.mp4
Uploading to input bucket.. name: test_34.mp4
Uploading to input bucket.. name: test_20.mp4
Uploading to input bucket.. name: test_18.mp4
Uploading to input bucket.. name: test_24.mp4
Uploading to input bucket.. name: test_30.mp4
Uploading to input bucket.. name: test_6.mp4
Uploading to input bucket.. name: test_7.mp4
Uploading to input bucket.. name: test_31.mp4
Uploading to input bucket.. name: test_25.mp4
Uploading to input bucket.. name: test_19.mp4
Uploading to input bucket.. name: test_33.mp4
```

Figure 2. Running the workload generator

Amazon S3

Buckets

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

AWS Organizations settings

Feature spotlight

AWS Marketplace for S3

Amazon S3 > Buckets > input-bucket-video

input-bucket-video

Publicly accessible

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (38)

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](#)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	test_0.mp4	mp4	March 25, 2023, 15:32:14 (UTC-07:00)	315.0 KB	Standard
<input type="checkbox"/>	test_1.mp4	mp4	March 25, 2023, 15:32:15 (UTC-07:00)	3.4 MB	Standard
<input type="checkbox"/>	test_18.mp4	mp4	March 25, 2023, 15:32:36 (UTC-07:00)	3.4 MB	Standard
<input type="checkbox"/>	test_19.mp4	mp4	March 25, 2023, 15:32:46 (UTC-07:00)	408.5 KB	Standard
<input type="checkbox"/>	test_2.mp4	mp4	March 25, 2023, 15:32:29 (UTC-07:00)	408.5 KB	Standard
<input type="checkbox"/>	test_20.mp4	mp4	March 25, 2023, 15:32:34 (UTC-07:00)	345.7 KB	Standard
<input type="checkbox"/>	test_21.mp4	mp4	March 25, 2023, 15:32:25 (UTC-07:00)	1.6 MB	Standard
<input type="checkbox"/>	test_22.mp4	mp4	March 25, 2023, 15:32:11 (UTC-07:00)	739.0 KB	Standard
<input type="checkbox"/>	test_23.mp4	mp4	March 25, 2023, 15:32:20 (UTC-07:00)	224.9 KB	Standard
<input type="checkbox"/>	test_24.mp4	mp4	March 25, 2023, 15:32:37 (UTC-07:00)	609.5 KB	Standard

Figure 3. Objects in the input-bucket

Amazon S3

Buckets

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

AWS Organizations settings

Feature spotlight

AWS Marketplace for S3

Amazon S3 > Buckets > output-bucket-vid

output-bucket-vid

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (47)

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](#)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	test_0	-	March 25, 2023, 15:32:17 (UTC-07:00)	30.0 B	Standard
<input type="checkbox"/>	test_1	-	March 25, 2023, 15:32:24 (UTC-07:00)	33.0 B	Standard
<input type="checkbox"/>	test_18	-	March 25, 2023, 15:32:41 (UTC-07:00)	33.0 B	Standard
<input type="checkbox"/>	test_19	-	March 25, 2023, 15:32:51 (UTC-07:00)	23.0 B	Standard
<input type="checkbox"/>	test_2	-	March 25, 2023, 15:32:32 (UTC-07:00)	23.0 B	Standard
<input type="checkbox"/>	test_20	-	March 25, 2023, 15:32:38 (UTC-07:00)	34.0 B	Standard
<input type="checkbox"/>	test_21	-	March 25, 2023, 15:32:43 (UTC-07:00)	37.0 B	Standard
<input type="checkbox"/>	test_22	-	March 25, 2023, 15:32:16 (UTC-07:00)	20.0 B	Standard
<input type="checkbox"/>	test_23	-	March 25, 2023, 15:32:24 (UTC-07:00)	26.0 B	Standard
<input type="checkbox"/>	test_24	-	March 25, 2023, 15:32:43 (UTC-07:00)	45.0 B	Standard

Figure 4. Objects in the output-bucket

AWS Console Home

Dashboard

Tables

Update settings

Explore items

PartiQL editor

Backups

Exports to S3

Imports from S3

Reserved capacity

Settings

DAX

Clusters

Subnet groups

Parameter groups

Events

Completed. Read capacity units consumed: 0.5

Items returned (8)

Actions

Create item

< 1 >

<input type="checkbox"/>	name	id	major	year
<input type="checkbox"/>	president_biden	2	history	sophomore
<input type="checkbox"/>	floki	4	history	junior
<input type="checkbox"/>	president_obama	7	electrical_e...	senior
<input type="checkbox"/>	mr_bean	1	lawyer	freshmen
<input type="checkbox"/>	president_trump	5	physics	junior
<input type="checkbox"/>	johnny_dep	8	computer_s...	senior
<input type="checkbox"/>	vin_diesel	3	computer_s...	sophomore
<input type="checkbox"/>	morgan_freeman	6	math	senior

Figure 5. Returned results

CloudWatch

Favorites and recents

Dashboards

Alarms 2 6 0

In alarm

All alarms

Billing

Logs

Log groups

Logs Insights

Metrics

All metrics

Explorer

Streams

X-Ray traces

Events

CloudWatch > Log groups > /aws/lambda/smart-classroom > 2023/03/25/[\$LATEST]9dab17a693d342038f4283fc6139e94f

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

Create metric filter

Filter events

Clear 1m 30m 1h 12h Custom

Display

	Timestamp	Message
		No older events at this moment. Retry
	2023-03-25T15:32:05.441-07:00	Imports done
	2023-03-25T15:32:05.442-07:00	loaded encoding
	2023-03-25T15:32:05.444-07:00	START RequestId: be72ec9c-7838-4768-b4db-679fa9b22c9f Version: \$LATEST
	2023-03-25T15:32:05.444-07:00	In event
	2023-03-25T15:32:05.444-07:00	ObjectCreated:Put
	2023-03-25T15:32:05.444-07:00	https://input-bucket-video.s3.amazonaws.com/test_7.mp4
	2023-03-25T15:32:07.302-07:00	test_7 uploaded to s3
	2023-03-25T15:32:07.305-07:00	END RequestId: be72ec9c-7838-4768-b4db-679fa9b22c9f
	2023-03-25T15:32:07.305-07:00	REPORT RequestId: be72ec9c-7838-4768-b4db-679fa9b22c9f Duration: 1860.64 ms Billed Duration: 4062 ms ...
	2023-03-25T15:32:07.754-07:00	START RequestId: a2fc3624-a7a0-4b00-babe-5b6dc6b18f17 Version: \$LATEST
	2023-03-25T15:32:07.755-07:00	In event
	2023-03-25T15:32:07.755-07:00	ObjectCreated:Put

Figure 6. Logs returned by the Lambda function

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
<input type="checkbox"/>	latest	Image	March 20, 2023, 14:15:13 (UTC-07)	984.99	Copy URI	sha256:dd0253bf833c58...	Complete	1 Critical + 376 others (details)
<input type="checkbox"/>	<untagged>	Image	March 19, 2023, 22:28:11 (UTC-07)	984.99	Copy URI	sha256:b8c356ae137aa...	Complete	1 Critical + 376 others (details)
<input type="checkbox"/>	<untagged>	Image	March 19, 2023, 22:20:34 (UTC-07)	984.99	Copy URI	sha256:61091dd2eda6f7...	Complete	1 Critical + 376 others (details)
<input type="checkbox"/>	<untagged>	Image	March 19, 2023, 22:16:42 (UTC-07)	984.99	Copy URI	sha256:82cf314e4dadf2d...	Complete	1 Critical + 376 others (details)
<input type="checkbox"/>	<untagged>	Image	March 19, 2023, 21:49:52 (UTC-07)	984.99	Copy URI	sha256:f2704279cb67c2...	Complete	1 Critical + 376 others (details)
<input type="checkbox"/>	<untagged>	Image	March 19, 2023, 21:45:03 (UTC-07)	984.99	Copy URI	sha256:73d6822700bcc9...	Complete	1 Critical + 376 others (details)
<input type="checkbox"/>	<untagged>	Image	March 17, 2023, 08:02:55 (UTC-07)	984.99	Copy URI	sha256:f31b880f6e72aa3...	Complete	1 Critical + 376 others (details)
<input type="checkbox"/>	<untagged>	Image	March 16, 2023, 18:59:02 (UTC-07)	984.98	Copy URI	sha256:f18bec26c783e45...	Complete	1 Critical + 376 others (details)

Figure 7. ECR Images

6. References:

1. <https://docs.aws.amazon.com/AmazonECR/latest/userguide/getting-started-cli.html>
2. <https://docs.aws.amazon.com/lambda/latest/dg/images-create.html>