Group Project 2: PaaS

CSE 546: Cloud Computing

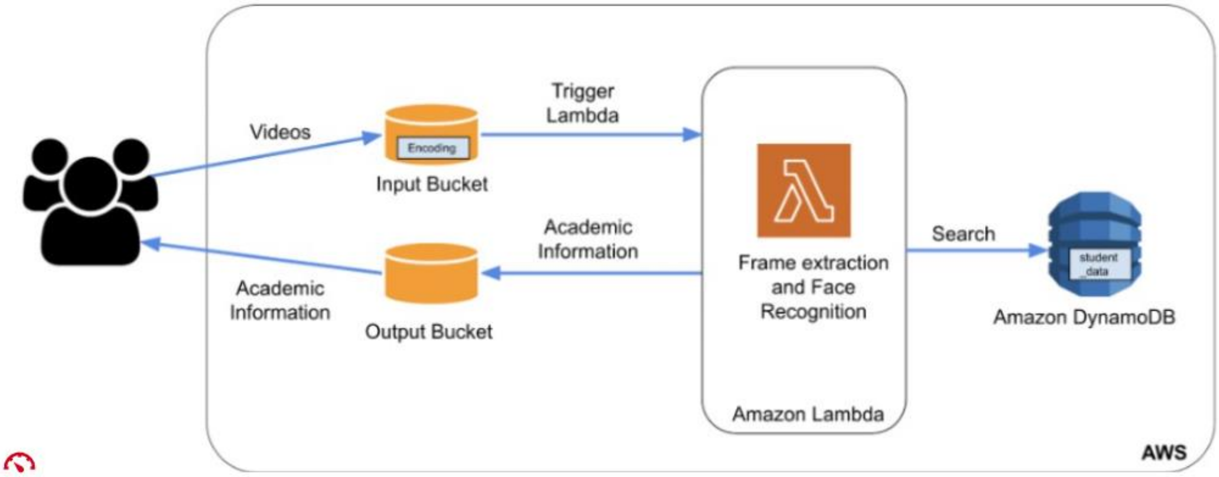
Due Oct 30th 11:59:59PM Firm!

Summary

In the second project, we will build an elastic application that can automatically scale out and in on demand and cost-effectively by using the PaaS cloud. Specifically, we will build this application using AWS Lambda and other supporting services from AWS. AWS Lambda is the first and currently the most widely used function-based serverless computing service. We will develop an application that is more sophisticated than Group Project 1, as we are now more experienced with cloud programming and PaaS also makes it easier for us to develop in the cloud. Our application will still offer a meaningful cloud service to users, and the technologies and techniques that we learn will be useful for us to build many others in the future.

Description

Our cloud app will implement a smart classroom assistant for educators. This assistant takes videos from the user's classroom, performs face recognition on the collected videos, looks up the recognized students in the database, and returns the relevant academic information of each student back to the user. The following diagram shows the architecture of this app.



1. Users upload videos to your input bucket stored in S3.
2. When a new video becomes available in the input bucket, it triggers the Lambda function to process the video.

The Lambda function uses a multimedia framework, *ffmpeg,* to extract frames from the video. The sample command in Python is as below:

```
path = "/tmp/"

os.system("ffmpeg -i " + str(video_file_path) + " -r 1 " + str(path) + "image-%3d.jpeg")
```

The Lambda function then uses the Python face_recognition library to recognize faces from the frames. To simplify your program, you need to classify only the first detected face in the video and ignore the rest. A tutorial on how to use this library is in the

following link: https://pypi.org/project/face-recognition/ Links to an external

site.. Links to an external site.You can find the encoding file which stores the names of the known faces here:

https://github.com/nehavadnere/cse546-project-lambda/blob/master/encoding
Links to an external site.

The Lambda function uses the name of the first recognized face to search in DynamoDB for this person's academic information. You need to preload the following student data into DynamoDB:

https://github.com/nehavadnere/cse546-project-lambda/blob/master/student_data.json

Links to an external site.

You will create your Lambda function using a customized container image which is preinstalled with ffmpeg and the face_recognition library. The Dockerfile for building this container image is at the following link:

https://github.com/nehavadnere/cse546-project-lambda/blob/master/Dockerfile
Links to an external site.

1. Finally, the Lambda function stores the student's academic information as a file in the output bucket in S3. The name of the file should be the video name (e.g., "test_0") and the content should be a CSV file containing these three fields: name, major, and year (e.g., "president_trump, physics, junior,").

To facilitate testing, use the sample videos provided here:

https://github.com/nehavadnere/cse546-project-lambda/tree/master/test_cases
Links to an external site.

Use this workload generator to create requests to your app (the TAs will use the same workload generator to grade your submission.)

https://github.com/nehavadnere/cse546-project-lambda/blob/master/workload.py
Links to an external site.
The workload generator will use this mapping file to check the correctness of your app's output.

https://github.com/nehavadnere/cse546-project-lambda/blob/master/mapping
Links to an external site.

Test your code thoroughly using the provided workload generator. Check the following:

- The output of face recognition is correct;
- The contents in the S3 input and output buckets are correct;
- All the requests are processed within a reasonable amount of time. As a reference point, for a workload of 100 requests (using the provided sample videos), the TAs' implementation of the project can complete it within seven minutes.

Your project will be graded according to the above criteria.

Submission

You need to submit your implementation on Canvas by due date. Your submission should be a single zip file that is named by the full names of your team members. The zip file should contain all your source code and the AWS credentials(aws_access_key_id, aws_secret_access_key) for programmatically accessing your AWS resources (Lambda functions, S3 buckets, and DynamoDB). It should also contain a simple README file that lists your group member names, the AWS credentials, and S3 bucket names of your app; and a detailed PDF file that describes the design of your application and any additional information that can help the instructor understand your code and run your application. Use the provided template to prepare your report. Do not include any binary file in your zip file. Only one submission is needed per group.

You need to keep your app running until the grading (demo session) is done. The TAs will use the above information (S3 bucket names and AWS credentials) to check your submission.

Failure to follow the above submission instructions will cause a penalty to your grade. The submission link will be closed immediately after the deadline.

Policies

- Late submissions will absolutely not be graded (unless you have verifiable proof of emergency). It is much better to submit partial work on time and get partial credit for your work than to submit late for no credit.

- Each group needs to work independently on this exercise. We encourage high-level discussions among groups to help each other understand the concepts and principles. However, the code-level discussion is prohibited and plagiarism will directly lead to failure of this course. We will use anti-plagiarism tools to detect violations of this policy.