Project 2 (Part 1): PaaS

[CSE 546] [SPRING_2024] Cloud Computing

*Due by **04/05/2024 at 11:59:59pm***

Summary

In the second project, we will develop another cloud application by using PaaS resources. 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 a more sophisticated application than 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 PaaS application will provide face recognition as a service on videos streamed from the clients (e.g., security cameras). This is an important cloud service to many users, and the technologies and techniques that we learn will be useful for us to build many others in the future.

Description

The project is divided into two parts. In the first part, we will focus on implementing the video-splitting function. The clients upload videos to the Input bucket. When a new video is uploaded, it triggers a video-splitting function that splits the video into frames and stores the Group-of-Pictures (GoP) in another bucket called Stage-1. The second part of the project will include the development of the other Lambda functions and S3 buckets needed for this application.

The architecture of the cloud application in Part 1 is shown in Figure 1.

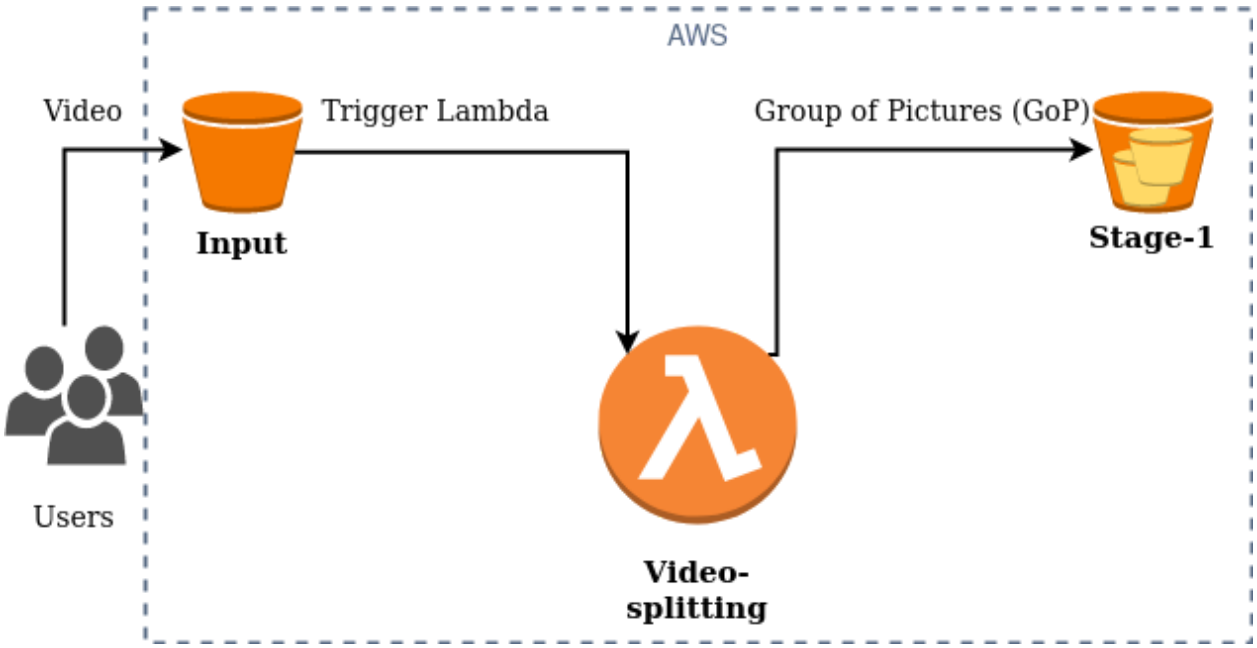

Fig 1: Architecture diagram of Project 2 (Part 1)

Input Bucket

- This bucket stores the videos uploaded by the workload generator.
- The name of the input bucket MUST be **<ASU ID>-input**. Each student submission MUST have only one input bucket with the exact naming. For example, if your ASU ID is "12345678910", your bucket name will be 12345678910-input.
- The input bucket must contain only .mp4 video files sent by the workload generator. (100-video dataset link)
- Each new video upload should trigger an invocation of the **video-splitting** Lambda function.

The video-splitting Function

- The name of this Lambda function MUST be "**video-splitting**".
- The function is triggered whenever a new video is uploaded to the "**<ASU ID>-input**" bucket.
- The function gets the video file and splits the video in Group-of-Pictures (GoP) using the ffmpeg library.
  The function splits the given input in frames using the following command:

```
ffmpeg -ss 0 -r 1 -i <input_video>.mp4 -vf fps=1/10 -start_number 0
-vframes 10 /tmp/test_0/output-%02d.jpg -y
```

- The function stores this group of pictures (GoP) in a single folder with the same name as the input video in the "**<ASU ID>-stage-1**" bucket. E.g. If the <input_video> is test_00.mp4, then its corresponding group of pictures (GoP) is stored in a folder named test_00 in the "**<ASU ID>-stage-1**" bucket.
- You can use the provided video-splitting code to implement your Lambda function.
- You can also use the provided templates for the Dockerfile and handler code. You need to add the required code/packages in the template code to run and compile the video-splitting function.

Stage-1 bucket

- This bucket stores the result of the video-splitting Lambda function.
- The name of the input bucket MUST be **<ASU ID>-stage-1**. Each student submission MUST have only one "**<ASU ID>-stage-1**" bucket with the exact naming.
- The <ASU ID>-stage-1 bucket should have a folder with the exact name as the input video name without any extension. E.g. If the input bucket has 5 videos with the names test_0.mp4, test_1.mp4, test_2.mp4, test_3.mp4, test_4.mp4, then the stage-1 bucket should have 5 folders with the names exactly as the input video files: test-0, test_1, test_2, test_3, test_4.
- Each folder in <ASU ID>-stage-1 contains images with the extension .jpg. It follows the naming convention "**output_xx.jpg**", where xx is the frame number.

Testing & Grading

- Create an IAM user with the following permissions to facilitate automated grading using our grading script. You can create policies from AWS UI -> IAM -> Policies.
  - **"s3:Get*",**
  - **s3:PutObject**
  - **s3:List***
  - **lambda:GetFunction**
  - **cloudwatch:GetMetricData**
- Provide access key and secret key for the IAM user created for grading using this Google Form.
- Use the provided workload generator to test your app thoroughly.
- The grading will be done using automated grading script and following the rubrics provided below.
- The Dockerfile and video-splitting function code are tested on Ubuntu 20.04.4.
- You must use AWS region 'us-east-1' for all the S3 buckets and lambda.

	Test Case	Test Criteria		Total Points
1	Validate the Lambda function	Check if there exists a Lambda function with the name video-splitting.	There is a Lambda function with the name- "video-splitting" that runs successfully (http response 200) (5)	5
2	Validate the names and initial states of S3 buckets	1) Check if there exist 2 S3 buckets with names - <ASU ID>-input, <ASU ID>-stage-1. 2) Check if both the S3 buckets are empty initially.	<10-digit-number>-input and <10-digit-number>-stage-1 buckets exist and are empty before the 100-videos test (5)	5
3	Validate 100 folders in the stage-1 bucket for 100-videos test with names the same as input videos	Run workload generator script on the provided URL by the students with 100 requests. Check there are 100 folders with names- test_0, …, test_99 in bucket stage-1.	Deduct 1 point for each wrong/missing folder in the stage-1 bucket	20
4	Validate the number of images in each folder of the stage-1 bucket	Check there are 10 jpg images with names - output_00.jpg, … output_10.jpg in each folder.	Deduct 1 point for each wrong/missing image in every folder in the stage-1 bucket	20
5	Check correctness of the output images	Sample 5 folders randomly from the stage-1 bucket and check if the images match with the input video	Deduct 1 point for each wrong image	20
6	Check average duration of the function	Check average duration of video-splitting Lambda function execution	Average duration is 10 .0k ms or below (20)	20
7	Check the concurrency of video-splitting function	Check maximum concurrency of video-splitting Lambda execution	Concurrency > 5 (10)	10

Submission

The submission requires three components; all must be done by ***04/05/2024 at 11:59:59pm***.

1. Provide both the bucket names and your grading user credentials using a Google Form.
2. Upload your source code to Canvas as a single zip file named by your full name: <lastname><firstname>.zip. Submit only the source code that you have created/modified: **handler.py, Dockerfile,** and **requirements.txt**. Do not include any code not developed by you. Do not include any binary files.

IMPORTANT:

- Failure to follow the submission instructions will cause a penalty to your grade.
- Do not change your code after the submission deadline. The grader will compare the code you have submitted on Canvas and the code you run on AWS. If any discrepancy is detected, it will be considered as an academic integrity violation.

Policies

1) 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 than to submit late for no credit.

2) You need to work independently on this project. We encourage high-level group discussions to help others understand the concepts and principles. However, code-level discussion is prohibited, and plagiarism will directly lead to failure in this course. We will use anti-plagiarism tools to detect violations of this policy.