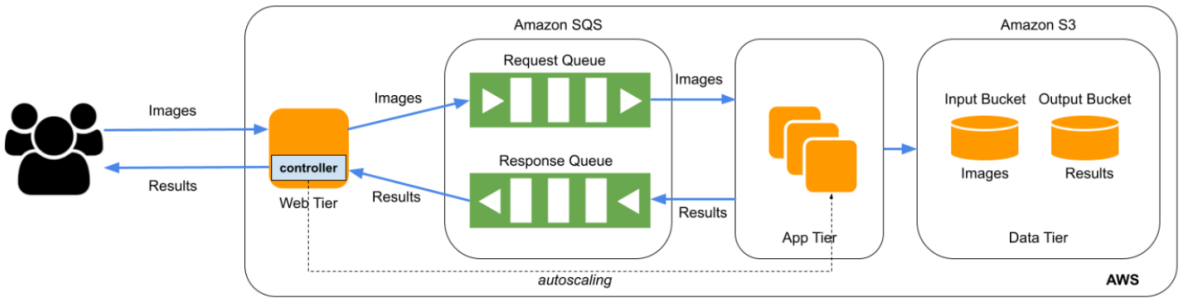
Project 1 (Part 2): IaaS

Summary

In Part 2 of Project 1, we will complete the development of the elastic face recognition application using the IaaS resources from AWS. We will implement the App Tier and Data Tier of our multi-tiered cloud application, use a machine learning model to perform face recognition and implement autoscaling to allow the App Tier to dynamically scale on demand.

Description

We will strictly follow the architecture below to develop the application.



Web Tier

1. You **MUST** use the same web tier EC2 instance from Part 1 of the project. The web tier **Must** listen for requests on port number **8000**.
2. It should take images received from users as input and forward it to the App Tier for model inference. It should also return the recognition result from the App Tier as output to the users. The input from each request is a .jpg file, and the output in each response is the recognition result. Find more details below.

Input:

- The key to the HTTP payload **MUST** be defined as "**inputFile**" and should be used as the same. In the case of a Python backend, it denotes a standard Python file object.
- For example, the user uploads an image named "test_00.jpg".
- Use the provided workload generator to generate requests to your Web Tier.

Output:

- The Web Tier will handle HTTP POST requests to the root endpoint ("/").
- The output **MUST** be in plain text, and the format <filename>:<classification_results>
- For the above example request, the output should be "test_00:Paul" in plain text.
- You need to implement the handling of concurrent requests in your Web Tier.

To facilitate the testing, a standard face dataset and the expected recognition output of each image are provided to you at

- **Input**: visa-lab/CSE546-Cloud-Computing/face_images_1000.zip
- **Output**: visa-lab/CSE546-Cloud Computing/classification_face_images_1000.csv

3. The Web Tier uses only **one** instance. You **MUST** name your web-tier instance "**web-instance**"

4. The Web Tier sends requests to the App Tier and receives results using two **SQS Queues**. You **MUST** follow the following naming convention in naming your S3 Buckets:
   a. **Request Queue**: <ID>-req-queue
   b. **Response Queue**: <ID>-resp-queue

   For example, if your ID is "12345678910", your queue names will be 12345678910-req-queue and 12345678910-resp-queue

5. The Web Tier also runs the autoscaling **controller,** which determines how to scale the App Tier. You are **not** allowed to use AWS features for autoscaling. You **MUST** implement your own auto scaling algorithm.

App Tier

1. The App Tier will use the provided deep learning model for model inference. The deep learning model code is available here. The first step is to create an AMI, which will be used to launch App Tier instances.
   a. Launch a base EC2 instance. You can use the AWS Linux or Ubuntu AMI.
   b. Install the required package on the instance using the following command.

```
pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cpu
```

   c. Copy the provided deep learning model code and model weights folder to the EC2 instance using scp.
   d. Refer to the README.md on how to use the deep learning model code.
   e. Create an AMI using this EC2 instance, following the instructions here. Now, you can use this AMI to create your App Tier instances. You **MUST** name your app-tier instances "**app-tier-instance-<instance#>**"
2. The App Tier should automatically scale out when the request demand increases and automatically scale in when the demand drops. The number of App Tier instances should be 0 when there are no requests being processed or waiting to be processed. The number of App Tier instances can scale to **at most 20** because we have limited resources from the free tier.

Data Tier

1. All the inputs (images) and outputs (recognition results) should be stored in separate buckets on S3 for persistence.
2. S3 stores all the objects as key-value pairs. For the Input bucket, each object's key is the input file's name, e.g., test_00.jpg, and the value is the image file. For the Output bucket, the key is the image name (e.g., test_00), and the value is the classification result
   (e.g., "Paul").
3. You **MUST** follow the following naming convention in naming your S3 Buckets
   a. **Input Bucket**: <ID>-in-bucket
   b. **Output Bucket**: <ID>-out-bucket

   For example, if your ID is "12345678910", your bucket names will be 12345678910-in-bucket and 12345678910-out-bucket