Project 1 (Part II): IaaS

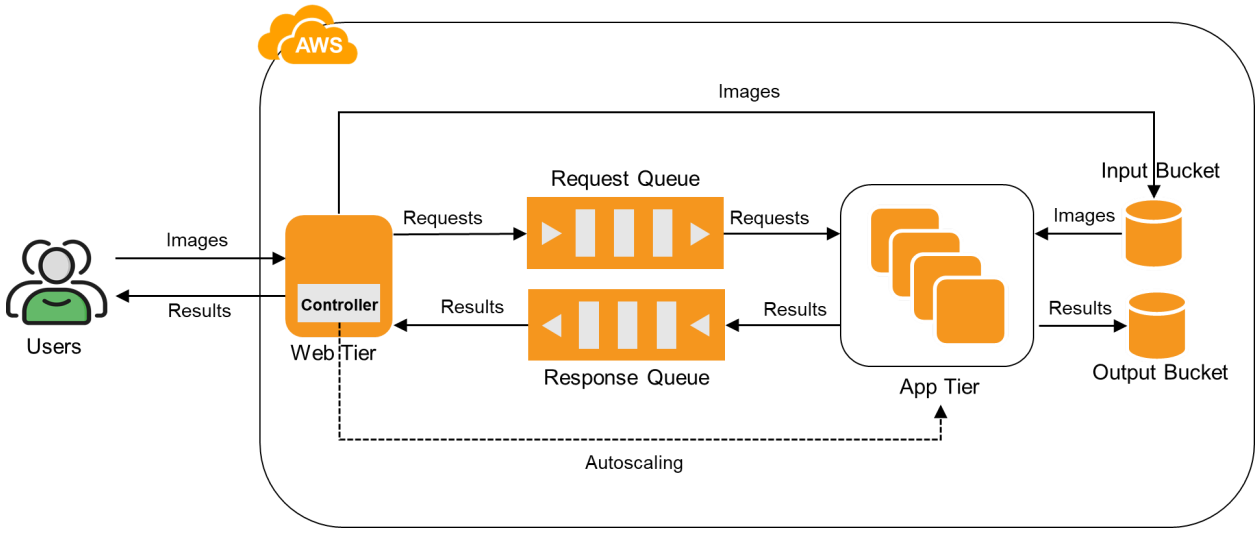
Due by **10/26/2025 11:59PM**

Summary

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

Description

We will strictly follow the architecture below to develop the application. You MUST use Python to implement all your programs.



Web Tier

The web tier will receive face recognition requests from users, forward the recognition requests to the app tier, receive the recognition results from the app tier and then return the results to the users. Name your web tier program “server.py”. The web tier runs on a single EC2 instance.

The web tier should implement the following (Steps 1 and 2 are already implemented by your Part I; Steps 3 and 4 are the new requirements of Part II):

1. Extract images received from HTTP requests

- The web tier MUST handle HTTP POST requests to the root endpoint ("/") on port 8000.
- 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_000.jpg”.

2. Store the received images in S3
    - Store all the input images in an S3 bucket for persistence.
    - 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_000.jpg, and the value is the image file.
    - You MUST name your S3 input bucket : <ASU ID>-in-bucket

      For example, if your ASU ID is “1225754101”, your input bucket name will be 1225754101-in-bucket.
3. Forward the face recognition requests to the application tier
    - The web tier sends the request from the users to the request SQS queue.
    - You MUST name your SQS request queue: <ASU ID>-req-queue. For example, if your ASU ID is “1225754101”, your request queue’s name will be 1225754101-req-queue.
    - You are NOT allowed to send the input images to the request queue. To make sure this does not happen, you need to change the maximum message size for the SQS queue to 1KB.
4. Return the results to the HTTP requests
    - The app tier will pass the face recognition results back to the web tier through the SQS response queue.
    - You MUST name your response queue: <ASU ID>-resp-queue.
      For example, if your ASU ID is “1225754101”, your response queue name will be 1225754101-resp-queue
    - The web tier will return the recognition result as the response to the original HTTP request. You are not allowed to use additional HTTP requests to fetch the results.
    - The output MUST be in plain text and in the format of <filename>:<prediction_results>.

      For the request “test_000.jpg”, the output should be “test_000:Paul” in plain text.

Application Tier

The app tier will use the provided deep learning model for model inference. Name your app tier program "backend.py".

The first step is to create an AMI for launching the application 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#>".

The application tier should implement the following for every request in the request queue:

1. Retrieve the request from SQS request queue (<ASU ID>-req-queue).
2. Fetch the corresponding image from the S3 input bucket.
3. Perform model inference for face recognition.
4. Store the recognition result in the S3 output bucket.
   a. For the output bucket, the key is the image_image (e.g., test_000), and the value is the classification results (e.g., "Paul").
   b. You MUST name your output bucket: <ASU ID>-out-bucket

      For example, if your ASU ID is "1225754101", your output bucket name will be 1225754101-out-bucket.
5. Push the recognition result to the response queue.

Autoscaling

The web tier will also implement the autoscaling controller (named "controller.py") which determines how to scale the application tier. You are NOT allowed to use the AWS Auto Scaling service. You MUST implement your own auto scaling algorithm according to the following policies:

1. The number of the application tier instances should be 0 when there are no requests being processed or waiting to be processed.
2. The number of the application tier instances can scale up to 15 because we have limited resources from the free tier.

3. Each application tier instance fetches and processes only 1 request at a time.
4. After processing the request, an application tier instance is stopped immediately if there are no more pending requests; otherwise, it continues to process the next request.
5. To reduce the startup overhead, all application tier instances can be initialized to the "stopped" state.

Additional Requirements

- Make sure to monitor your AWS usage and stay within the free tier limits for EC2, S3, and SimpleDB. If you incur any charge by accident, you can always request AWS to waive it.
- To facilitate testing, you MUST use only the resources from the US-East-1 region, and follow all the naming conventions as described in the document.
- Assign your single web instance a static IP address so it does not change during testing. You MUST name your EC2 instance "web-instance". Note that you can use a dynamic IP address during development to not incur any cost, but you MUST use Elastic IP to assign a static IP address to your web tier instance before you submit your project for grading.

Testing

Make sure that you use the provided autograder and follow the instructions below to test your project submission. Failure to do so may cause you to lose all the project points and there will be absolutely no second chance. Use a Linux-based system to test your project.

1. Download the zip file you submitted from Canvas.
2. Download the autograder from GitHub:
   https://github.com/nehavadnere/CSE546-FALL-2025.git
   In order to clone the Github repository follow the below steps:
   a. git clone https://github.com/nehavadnere/CSE546-FALL-2025.git
   b. cd CSE546-FALL-2025/
   c. git checkout project-1-part-2
   d. Create a directory "submissions" in the CSE546-FALL-2025 directory and move your zip file to the submissions directory.
2. To facilitate the testing, a standard face dataset and the expected recognition output of each image are provided to you at:
   - Input: dataset/face_images_1000.zip
   - Output: dataset/classification_face_images_1000.csv

3. Prepare to run the autograder
    a. Install Python: sudo apt install python3
    b. Populate the class_roster.csv
        i. If you are a student; replace the given template only with your details.
        ii. If you are a grader; use the class roster for the entire class.
    c. Clean up your S3 buckets and SQS queues.
4. Run the autograder
    - To run the autograder: python3 autograder.py --num_requests 100 --img_folder="<dataset folder path>" --pred_file="<output classification csv file path>"
    - The autograder will look for submissions for each entry present in the class_roster.csv
    - For each submission the autograder will
        - The autograder extracts the credentials.txt from the submission and parses the entries.
        - Use the Grader IAM credentials to test the project as per the grading rubrics and allocate grade points.
    - The autograder has a workload generator component to generate requests to your web tier.
5. Use the provided autograder to test your app thoroughly. Check the following:
    a. The face recognition results are correct.
    b. The contents in the S3 buckets are correct.
    c. While processing the workload, the number of EC2 instances is correct.
    d. The number of messages in request and response queues should increase from 0 and then reduce back to 0.
    e. All the requests are processed within a reasonable amount of time. As a reference point, for a workload of 100 concurrent requests using the TAs' implementation of the project, it completed within 96 seconds.
    f. After all requests are processed, the application tier should scale back to 0 within 5 seconds after the workload is over.
6. To facilitate testing, you MUST use only the resources from the US-East-1 region.
7. Note: In the past we have observed a rate limit on the ASU Wi-Fi. Test using your own phone's hotspot or home network if you experience high latency when using ASU Wi-Fi.
8. Sample Output

```
++++++++++++++++++++++++++++++ CSE546 Autograder  ++++++++++++++++++++++++++++++
- 1) Extract the credentials from the credentials.txt
- 2) Execute the test cases as per the Grading Rubrics
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++ Autograder Configurations ++++++++++++++++++++++++++++
Project Path: /home/local/ASUAD/kjha9/git/GTA-CSE546-FALL-2025/Project-1/part-2/grader
Grade Project: Project-1
Class Roster: class_roster.csv
Zip folder path: /home/local/ASUAD/kjha9/git/GTA-CSE546-FALL-2025/Project-1/part-2/grader/submissions
Grading script: /home/local/ASUAD/kjha9/git/GTA-CSE546-FALL-2025/Project-1/part-2/grader/grade_project1_p2.py
Test Image folder path: ../web-tier/upload_images/
Classification results file: ../../Classification Results on Face Dataset (1000 images).csv
Autograder Results: Project-1-grades.csv
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++ Grading for Doe John ASUID: 1225754101 ++++++++++++++++++++
Extracted
/home/local/ASUAD/kjha9/git/GTA-CSE546-FALL-2025/Project-1/part-2/grader/submissions/Project1-1225754101.zip to
extracted
File: extracted/credentials/credentials.txt has values (‘XXXXXXXXXXXXXXXXX’, ‘XXXXXXXXXXXXXXXXX’, 'XXXXXX')
Credentials parsing complete.
----------------------------------------------------------------
IAM ACCESS KEY ID: XXXXXXXXXXXXXXXXX
IAM SECRET ACCESS KEY:XXXXXXXXXXXXXXXXX
----------------------------------------------------------------
Following policies are attached with IAM user:cse546-AutoGrader: ['AmazonEC2ReadOnlyAccess', 'IAMReadOnlyAccess',
'AmazonSQSFullAccess', 'AmazonS3FullAccess']
[IAM-log] AmazonEC2ReadOnlyAccess policy attached with grading IAM
[IAM-log] AmazonS3FullAccess policy attached with grading IAM
[IAM-log] AmazonSQSFullAccess policy attached with grading IAM
[Cloudwatch-log] CAUTION !! You do not have a Cloudwatch alarm set. Kindly refer to the Project-0 document and learn
how to set a billing alarm
-------------- CSE546 Cloud Computing Grading Console -----------
IAM ACCESS KEY ID: XXXXXXXXXXXXXXXXX
IAM SECRET ACCESS KEY:XXXXXXXXXXXXXXXXX
Web-Instance IP Address: XXXXXX
----------------------------------------------------------------
----------------- Executing Test-Case:1 ----------------
[EC2-log] AmazonEC2ReadOnlyAccess policy attached with grading IAM
[EC2-log] Found 1 web-tier instances in running state.
[EC2-log] Found 0 app-tier instances in running state
[EC2-log] EC2-state validation Pass. Found 1 web-tier instances in running state. Found 0 app-tier instances in
running state.Points deducted: 0
[S3-log] AmazonS3FullAccess policy attached with grading IAM
[S3-log] - WARN: If there are objects in the S3 buckets; they will be deleted
[S3-log] ------------------------------------------------------
[S3-log] S3 Bucket:1225754101-in-bucket has 0 object(s).
[S3-log] S3 Bucket:1225754101-out-bucket has 0 object(s).
[S3-log] Points deducted:0
[SQS-log] The expectation is that both the Request and Response Queues should exist with max message size set to 1KB
and be EMPTY
[SQS-log] - WARN: This will purge any messages available in the SQS
[SQS-log] ------------------------------------------------------
[SQS-log] AmazonSQSFullAccess policy attached with grading IAM
[SQS-log] SQS Request Queue:1225754101-req-queue has 0 pending messages with max message size set to 1 KB.
```

```
[SQS-log] SQS Response Queue:1225754101-resp-queue has 0 pending messages.
[SQS-log] Points deducted:0
----------------- Executing Test-Case:2 ----------------
[AS-log] - Autoscaling validation starts ..
[AS-log] - The expectation is as follows:
[AS-log]  -- # of app tier instances should gradually scale and eventually reduce back to 0
[AS-log]  -- # of SQS messages should gradually increase and eventually reduce back to 0
```

# of messages in SQS Request Queue	# of messages in SQS Response Queue	# of app-tier EC2 instances in running	# of objects in S3 Input Bucket	# of objects in S3 Output Bucket
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
29	0	0	47	0
29	0	0	100	0
78	0	0	100	0
78	0	0	100	0
100	0	2	100	0
0	0	6	100	0
100	0	12	100	0
29	0	15	100	0
100	0	15	100	0
100	0	15	100	0
100	0	15	100	0
99	0	15	100	0
100	0	15	100	0
87	0	15	100	2
97	0	15	100	6
91	0	15	100	10

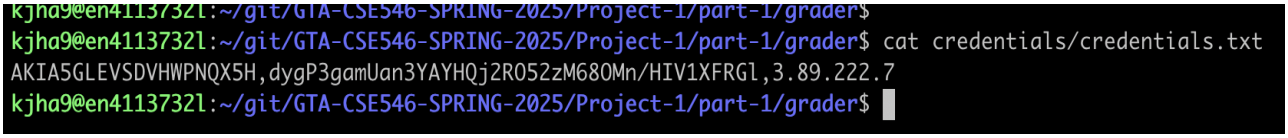
```
|         73          |         0          |         15         |        100         |         16         |
-------------------------------------------------------------------------------------------------------
|         75          |         2          |         15         |        100         |         25         |
-------------------------------------------------------------------------------------------------------
|         87          |         0          |         15         |        100         |         33         |
-------------------------------------------------------------------------------------------------------
|         54          |         4          |         15         |        100         |         43         |
-------------------------------------------------------------------------------------------------------
|         39          |         4          |         15         |        100         |         51         |
-------------------------------------------------------------------------------------------------------
|         70          |         12         |         15         |        100         |         62         |
-------------------------------------------------------------------------------------------------------
|         54          |         0          |         15         |        100         |         71         |
-------------------------------------------------------------------------------------------------------
|         30          |         0          |         15         |        100         |         81         |
-------------------------------------------------------------------------------------------------------
|         16          |         14         |         15         |        100         |         91         |
-------------------------------------------------------------------------------------------------------
|         0           |         12         |         15         |        100         |         97         |
-------------------------------------------------------------------------------------------------------
|         61          |         9          |         15         |        100         |         97         |
-------------------------------------------------------------------------------------------------------
|         73          |         10         |         15         |        100         |         98         |
-------------------------------------------------------------------------------------------------------
|         5           |         12         |         15         |        100         |         98         |
-------------------------------------------------------------------------------------------------------
|         0           |         10         |         15         |        100         |         98         |
-------------------------------------------------------------------------------------------------------
|         0           |         12         |         12         |        100         |         98         |
-------------------------------------------------------------------------------------------------------
|         0           |         17         |         4          |        100         |        100         |
-------------------------------------------------------------------------------------------------------
[Workload-gen] ----- Workload Generator Statistics -----
[Workload-gen] Total number of requests: 100
[Workload-gen] Total number of requests completed successfully: 100
[Workload-gen] Total number of failed requests: 0
[Workload-gen] Total number of correct predictions : 100
[Workload-gen] Total number of wrong predictions: 0
[Workload-gen] Total response time: 96.1222038269043 (seconds)
[Workload-gen] ----------------------------------

|         0           |         0          |         0          |        100         |        100         |
-------------------------------------------------------------------------------------------------------
|         0           |         0          |         0          |        100         |        100         |
-------------------------------------------------------------------------------------------------------
|         0           |         0          |         0          |        100         |        100         |
-------------------------------------------------------------------------------------------------------
[Test-Case-3-log] Waiting for 5sec for the resources to scale in ...
[AS-log] Time to scale in to 0 instances: 0.24 seconds.Points:[10/10]
[Test-Case-3-log] Stop event set. Waiting for autoscaling thread to finish.
|         0           |         0          |         0          |        100         |        100         |
```

```
----------------------------------------------------------------------------------------------------------------
[Test-Case-3-log] 100/100 entries in S3 bucket:1225754101-in-bucket.Points:[5.0/5]
[Test-Case-3-log] 100/100 entries in S3 bucket:1225754101-out-bucket.Points:[5.0/5]
[Test-Case-3-log] 100/100 correct predictions.Points:[10.0/10]
[Test-Case-3-log] Test average Latency: 0.961222038269043 sec. `avg latency<1.2s`.Points:[40/40]
[Test-Case-3-log] ------------------------------------------------------
[AS-log] EC2 instances scale out as expected.Points:[15/15]
[AS-log] EC2 instances scale back to 0 as expected.Points:[5/5]
[AS-log] SQS messages in 1225754101-req-queue increased from 0 and reduced back to 0. [5/5]
[AS-log] SQS messages in 1225754101-resp-queue increased from 0 and reduced back to 0. [5/5]
[AS-log] S3 bucket:1225754101-in-bucket objects increased from 0 to 100.
[S3-log] Bucket:1225754101-in-bucket is now EMPTY !!
[AS-log] S3 bucket:1225754101-out-bucket objects increased from 0 to 100.
[S3-log] Bucket:1225754101-out-bucket is now EMPTY !!
[AS-log] ------------------------------------------------------
Total Grade Points: 100.0
Removed extracted folder: extracted
Total time taken to grade for Doe John ASUID: 1225754101: 115.00062108039856 seconds
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Grading complete for Project-1. Check the Project-1-grades.csv file.
```

Submission

To facilitate grading, DO NOT split server.py, controller.py, or backend.py into multiple source code files.

1. Create a Grading IAM User. The TA will use this only for grading.
    a. You MUST name the grading IAM user as “cse546-AutoGrader”
    b. For Project-1**,** the Grading IAM requires only these permission
        i. IAMReadOnlyAccess
        ii. AmazonEC2ReadOnlyAccess
        iii. AmazonS3FullAccess
        iv. AmazonSQSFullAccess
2. Submit the following in a zip file to Canvas:
    a. Credentials: Make a directory named “credentials”, and include your txt file named ‘credentials**.**txt**’**
        i. In the ‘credentials’ folder, create a txt file “credentials**.**txt” with following parameter values separated by commas in the order as mentioned below
            1. ACCESS KEY ID of the grading IAM user
            2. SECRET ACCESS KEY of the grading IAM user



3. Elastic IPv4 address of the web tier.

b. Web tier code: Make a directory named 'web-tier' and include your web tier code file named 'server.py' and 'controller.py' for the controller code.
Disclaimer: You are not allowed to submit any additional Python files apart from 'server.py' and 'controller.py'.

c. App tier code: Make a directory named 'app-tier' and include your app tier code file named 'backend.py'

2. You MUST name the zip file following the below naming convention: "Project1-<Your ASU ID>.zip", e.g., Project1-1225754101.zip
You can use the following command to create your zip file:
zip -r Project1-1225754101.zip credentials/ web-tier/ app-tier/
Note: Extensions automatically generated by Canvas due to multiple submission attempts will be managed by the grading script.
3. Do not submit any other file.
4. Keep your web tier instance running until you are informed that grading is done. If your web tier is not reachable when your project is being graded, you will receive zero point for the entire project.
5. Make sure your S3 buckets and SQS queues are all empty.
6. Do not change your code after the submission deadline. We 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.
7. Do not do any kind of testing on your web tier after the submission deadline; otherwise, you may lose CPU credit, causing your instance to be slow during the grading and deductions to your project grade.

Rubrics

Note: The grader may use a workload different from the one provided to test your project during grading. Passing all the test cases does not guarantee all the points. Your code will be checked for correctness, plagiarism, and use of AI-generated code.

#	Test Objective	Test Criteria	Pass	Points
1	Validate the initial state of the resources (EC2, S3 and SQS)	To check if 1) There is 1 web tier instance named "web-instance" and in "running" state. 2) There are 0 app tier instances in "running" state. 2) The input and output buckets must be empty. 3) The SQS request and response queues must be empty 4) The request queue's max message size must be 1KB	1) Deduct 100 points if the web tier is not accessible. 2) Deduct 70 points if there are any app tier instances in the running state 2) Deduct 5 points each if any bucket or queue is not empty 3) Deduct 20 points if the request queue's max message size is not set to 1KB	
2	Validate the completeness, correctness and the latency of the requests	Run workload generator script on the provided IP address by the students with 100 requests 1) The HTTP response code must be 200 for all the requests 3) All the classification results must be correct 4) The S3 input and output buckets should each have 100 objects 5) The number of messages in request queue and response queue gradually increase and reduce back to 0 6) Validate autoscaling 7) Response time must be reasonable	For each of the 100 requests 1) +0.1 correct classification result (10) 2) +0.05 storing the input in the S3 bucket (5) 3) +0.05 storing the output in the S3 bucket (5) 4) The # of app tier instances should gradually scale out from 0 to 15 (15) 5) The # of app tier instances scales in to 0 (5) 6) Total time for app tier instances to scale in to 0 after the workload is over (10) - less than 2 secs (10) - between 2 secs - 5 secs (5) - greater than 5 secs (0) 7) The # of SQS messages in request queue should increase from 0 and then reduce back to 0 (5) 8) The # of SQS messages in response queue should increase from 0 and then reduce back to 0 (5) 9) Average latency for 100 requests (40) - < 1.2 secs (40) - >= 1.2 secs and < 2.1 secs (20) - between >= 2.1 secs < 3 secs (10) - >= 4.2 secs (0)	100

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.
- Every student needs to **work independently** on this exercise. We encourage high-level discussions among students to help each other understand the concepts and principles. However, a 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.
- **The use of generative AI tools is not allowed** to complete any portion of the assignment.