

# Project 1: IaaS

CSE 546: Cloud Computing

## Summary

In the first project, we will build an elastic application that can automatically scale out and in on demand and cost-effectively by using IaaS cloud. Specifically, we will build this application using the IaaS resources from Amazon Web Services (AWS). AWS is the most widely used IaaS provider and offers a variety of compute, storage, and message services. Our application will 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 future.

## Description

Our cloud app will provide an image recognition service to users, by using cloud resources to perform deep learning on images provided by the users. The deep learning model will be provided to you, but you need to build the application that uses this model to provide the service, and meet the following typical requirements for a cloud application:

- 1) The app should take images received from users as input and perform image recognition on these images using the provided deep learning model. It should also return the recognition result (the top 1 result from the provided model) as output to the users. The input is a .png file, and the output is the prediction result. For example, the user uploads an image named "1\_ship.png". For the above request, the output should be "ocean liner" in plain text.

To facilitate the testing, a standard image dataset is provided to you at:

<http://visa.lab.asu.edu/cifar-10/>

To send images to the web tier of your cloud app, you can refer to the following document:

<https://docs.google.com/document/d/1q4nbhtviiMykzwVIEXnpGa9BYXADyiPrrXfiUGjAJHg/edit?usp=sharing>

- 2) The deep learning model is provided to you in an AWS image (ID: ami-0ee8cf7b8a34448a6, Name: app-tier, Region: us-east-1). Use the following command to invoke the program in the home directory:  

```
cd /home/ubuntu/classifier/  
python3 image_classification.py path_to_the_image
```
- 3) Each image input is a request to your application and your application should be able to handle multiple requests concurrently. It should automatically scale out when the request

Copyright © Ming Zhao, PhD

demand increases, and automatically scale in when the demand drops. Because we have limited resources from the free tier, the application should use no more than 20 instances, and it should queue all the pending requests when it reaches this limit. When there is no request, the application should use the minimum number of instances. Give your instances meaningful names, for example, a web-tier instance should be named in the fashion of “web-instance1” and an app-tier instance should be named “app-instance1”.

- 4) All the inputs (images) and outputs (recognition results) should be stored in S3 for persistency. The inputs should be stored in one bucket in the form of .png file, e.g., 1\_ship.png, and the outputs should be stored in another bucket in the form of (input, output) pairs, e.g., (1\_ship, ocean liner). Specify your bucket names in your project README file.
- 5) The application should handle all the requests as fast as possible, and it should not miss any request. The recognition requests should all be correct.
- 6) For the sake of easy testing, use the resources from only the us-east-1 region for all your app's needs.

## Submission

The submission includes two steps:

- 1) You need to submit your implementation on Canvas by **March 12th 11:59:59pm**. 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 for running your application. It should also contain a simple README file that lists your group member names and the public IP and S3 bucket name 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. A template will be provided to you for writing the report. Do not include any binary file in your zip file. Only one submission is needed per group.  
Failure to follow the above submission instructions will cause penalty to your grade. The submission link will be closed immediately after the deadline.
- 2) You need to give a live demo to the TAs on **March 12th** on Zoom. We will try the code that you submit to Canvas. You have only five minutes for the demo, and there will be no second chance if you fail the demo.

## 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 for your work than to submit late for no credit.
- 2) 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, 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.