Group Project 1: IaaS

Due by Sep 25th

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 the 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 massage 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 the 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 "test_0.JPEG". For the above request, the output should be "bathtub" in plain text.

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

   Links to an external site. imagenet-100-updated.zip Download imagenet-100-updated.zip

   The expected output of each image is provided to you at Classification

   Results on ImageNet.xlsx Download Classification Results on ImageNet.xlsx

   Use the provided workload generator to create concurrent requests to your app. The TAs will use the same workload generator to grade your submission. In this code repo, you can also find examples of the server-side code for handling concurrent

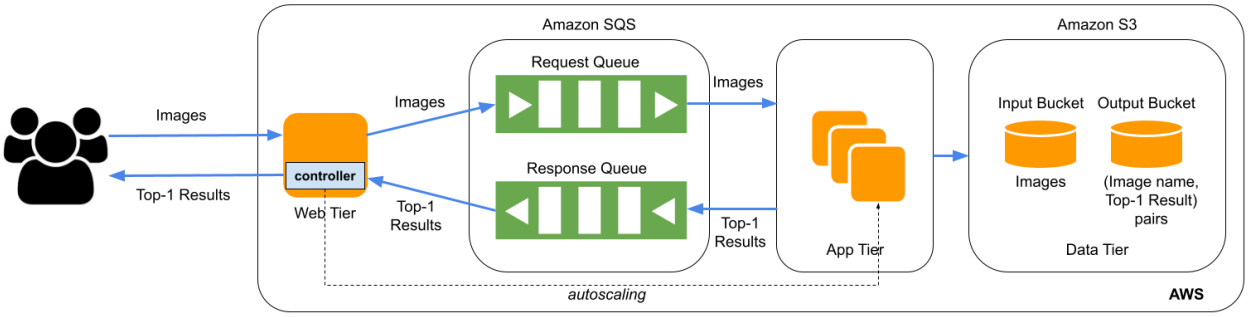
requests. https://github.com/nehavadnere/CSE546_Sum22_workload_generator Links to an external site.

2. The deep learning model is provided to you in an AWS image (ID: ami-0bb1040fdb5a076bc Links to an external site., Name: app-tier, Region: us-east-1). Please login/connect as user "***ubuntu***", not "***root***", then 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. Your app should be able to handle multiple requests concurrently. It should automatically scale out when the request demand increases, and automatically scale in when the demand drops. Because we have limited resources from the free tier, the app 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 app 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 persistence. S3 stores all the objects in a form of key-value pair. The inputs should be stored in one bucket. The key of each object in this bucket is in the form of .JPEG files, e.g., test_0.JPEG. and the value in this bucket is the image file. The outputs should be stored in another bucket. The key is the image name (e.g., test_0), and the value is in the form of (image name, top-1 result) pairs, e.g., (test_0, bathtub). Specify your bucket names in your project README file.
5. The app should handle all the requests as fast as possible, and it should not miss any requests. 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.

Follow the reference architecture below to implement your app.



Note: to avoid the web tier times out before the app tier finishes processing a request, you can remove the web tier timeout limit. For PHP backends, you can set set_time_limit to 0 to remove the limit.

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

1. The output of the workload generator is correct;
2. The contents in the S3 buckets are correct;
3. While processing the workload, the number of EC2 instances is correct.
4. 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 completes within seven minutes.

Your project will be graded according to the above criteria.

Submission

You need to submit your implementation on Canvas by Sep 25th at 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 (aws_access_key_id, aws_secret_access_key) for programmatically accessing your AWS resources (EC2 instances, SQS queue, and S3 buckets). It should also contain a simple README file that lists your group member names, each members' task, the AWS credentials, PEM key for web-tier SSH access, the web tier's URL, EIP, SQS queue names, 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 template on the bottom of this page to prepare your PDF file (group 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 is done. We will host a demo session (TIME TBD) Demo will weight a significate percentage of the project grade, so please prepare well. The TAs may also use the above information (web tier URL, SQS queue names, S3 bucket names, and AWS credentials) to check your submission independently.

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

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 the failure of this course. We will use anti-plagiarism tools to detect violations of this policy.