Project 1: IaaS—Amazon Web Services

CSE 546: Cloud Computing

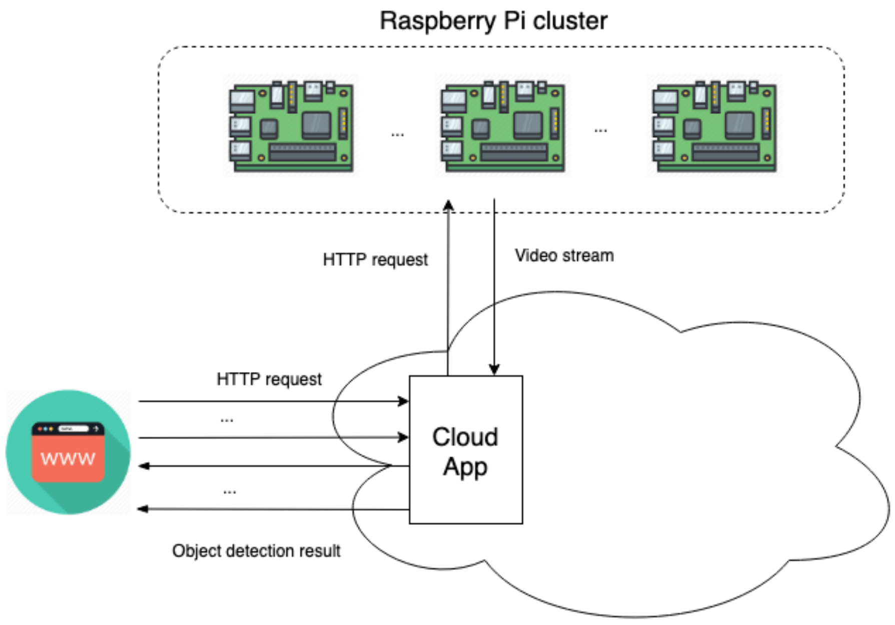
Bookmark this page (instead of printing it out). We will update it with more info and tips as needed.

1. 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 cloud resources. Specifically, we will build this application using cloud resources from Amazon Web Services (AWS). AWS is the most widely used IaaS provider and offers a variety of compute, storage, and message cloud 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.

2. Description

Our cloud app will provide a video surveillance service to users, by using cloud resources to perform deep learning on videos collected from Internet of Things (IoT) and to detect objects in the videos in real time.



The deep learning model and IoT video feeds will be provided to you. You need to build a cloud app that uses this model to provide the service and meet the following typical requirements for a cloud app:

1) Your app should service requests from clients via HTTP using a URL such as http://X.X.X.X (replace X.X.X.X with the IP address of the app). After receiving a request from a client, the app should send a video request to the surveillance system (built using a cluster of raspberry Pis deployed at Dr. Z's lab), using the URL: http://206.207.50.7/getvideo. The Pi cluster will start recording and return the video back to the app. After receiving the video, the app should perform object detection on the videos using the provided deep learning model, and return the detection result (e.g., "person") back to the client via HTTP.

2) The deep learning model will be provided to you in an AWS image(ami-0e355297545de2f82). Your app should invoke this model to perform object detection on the received videos. The URL of the video surveillance system will be provided to you in 1). To detect the objects in the video, you should run *Xvfb :1 & export DISPLAY=:1* first, then run *./darknet detector demo cfg/coco.data cfg/yolov3-tiny.cfg + weight_path + video_path -dont_show > result* to invoke the model. Last, run *python darknet_test.py* to parse the output from darknet.

3) The app should be able to handle multiple requests from clients 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 (video names) and outputs (detection results) should be stored in S3 for persistence. They should be stored in a bucket in the form of (input, output) pairs, e.g., ("video-1550538579.h264", "person").

5) The app should handle all the requests as fast as possible, and it should not miss any request. The object detection requests should all be correct.

6) For the sake of easy testing, use the resources from only the us-west-1a region for all your app's needs.

3. Submission

The submission includes two steps:

1) You need to submit your implementation on Canvas by **March 22nd 11:59:59pm**.

    a) 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.

    b) 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.

    c) Do not include any binary file in your zip file. Only one submission is needed per group.

    d) Failure to follow the above submission instructions will cause a penalty to your grade. The submission link will be closed immediately after the deadline.

2) You need to give a live demo to the TA by **March 22nd 5pm**. We will try the code that you submitted to Canvas. You have only five minutes for the demo, and there will be no second chance if you fail the demo.

4. 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, 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.