CSE 546 Cloud Computing

Individual Contribution MSCS portfolio report

Rohan Vipulkumar Dvivedi
ASU id 1224063958

This document consists of individual contributions of Rohan Dvivedi for the project 1 of the course CSE 546 Cloud computing. This project consisted of 3 major tasks, the app tier, web tier and the auto scaling. I, being a member of the group lambda, was responsible for implementing, testing, debugging and deployment of the webtier application on an EC2 instance.

The webtier (in this project) is an HTTP web service, accessible over the HTTP path "/image/push" (specific to my application), on which image can be uploaded in multipart/form-data format, and the webtier will then return a HTTP response, with body in text/plain format holding the classification result of the image.

I first started with an empty Django web application. I added a push controller to which an image can be uploaded. I then started to implement utility functions to write and read messages from SQS queues.

I required a thread that would continuously poll messages from the response SQS queue, update the global ReceivedMessages data structure and notify any other threads that are waiting for updates in this data structure. Here the ReceivedMessages data structure is nothing but a python dictionary that is protected by a lock (ReceivedMessagesLock) and we can wait for changes to be done in this data structure on a condition variable (ReceivedMessagesConditionVariable).

Our logic in webtier flows as follows: A request comes with an image. Then we mark the global data structure with a flag keyed by the image name, marking that we are waiting for a response for this image. We push the image in the SQS queue in the format [Image_name + ":" + base64(image_data)]. And then we go to wait on the global condition variable, so that any other thread can wake us up, when there are any changes in the ReceivedMessages data structure (This is done by the additional thread that we created before, as explained earlier).

When any message is read from the response SQS queue by our additional thread (of format [Image_name + ":" + classification_result]). We store the result (keyed by image name) into the global results data structure.

I was also the one who deployed the webtier Django app on the EC2 instance along with an apache2 webserver. Doing this I learned how to configure/tune a production grade web server like Apache2 server for any web application.