Reliability Project
Auto-scaling for Cloud Microservices

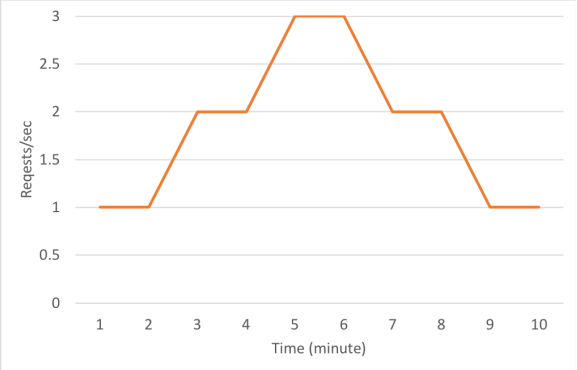
ECE 422, Winter 2018

Due Date: 11:59pm Thursday, April 12, 2018.

Your goal in this project is to implement a reactive auto-scaling engine for a cloud microservice application. You deploy the application on the Cybera infrastructure using Docker microservices. For this project a starter kit has been provided for you:

- https://github.com/hamzehkhazaei/ECE422-Proj2-StartKit

The auto-scaler should scale out or in (ie, horizontal scalability) the application according to the workload. To do so, the auto-scalar needs to monitor the response time of users' requests and check if the response times are in the acceptable range or not; acceptable range is indicated by an upper and lower threshold. If response times are longer than the upper bound then it needs to scale out the application to maintain the reliability and performance. And if the response times are shorter than the lower threshold, your auto-scaler should scale in the application to optimize the operation cost on the cloud. This way you will have a self-adaptive application that optimizes the performance and cost at the same time. The best way to test your application is to apply a normal shape (aka bell shape) workload such as the following to your application.



Hints:

1. This application comprises of two microservices, namely **web** and the datastore (ie **Redis**). The web microservices is the bottleneck; so, your auto-scaler only needs to manage the web microservice.
2. In order to avoid oscillation (aka ping-pong effect), your auto-scaler is better to monitor response times for a period of time (eg 10-20 seconds) and then based on the average value decide what to do next.

The final result of this project should be an auto-scalable application that reactively adjusts itself with respect to the workload. You can implement the auto-scaler in any language you want, including Python, Go, Java or C++.

Design Document

You need to submit a report of your project that includes the followings:

1. A high level architectural view of your application in which auto-scalability features has been shown.
2. A state diagram that shows the state, events and actions in the auto-scaler.
3. The pseudocode of the auto-scaling algorithm along with reasons behind the parameter settings in the algorithm; important parameters may include lower and higher thresholds, the length of the monitoring interval, scaling policies, etc.

Submission

There are 2 phases to complete your project:

1. You need to host your project on a Gitlab/GitHub private project and add me (id: hamzehkhazaei) and the TA Mojtaba Yeganejou (id: mojtabayeganejou) as project members. Your project should include source codes, design documents and the readme. You may not modify or change anything after the deadline.

   ### Hint:

   Or you can alternatively create a subproject within your first project so that there is no need to manage two separate repositories.

2. Finally, you must demo your auto-scalable application for the TA within one week after the deadline; you will demo your project in Software Engineering Laboratory, ETLC 5-005. You should arrange your demo time with the TA (yeganejo@ualberta.ca). Each demo should take no longer than 30 minutes.

Grading

Design	30%
Correct operations	70%