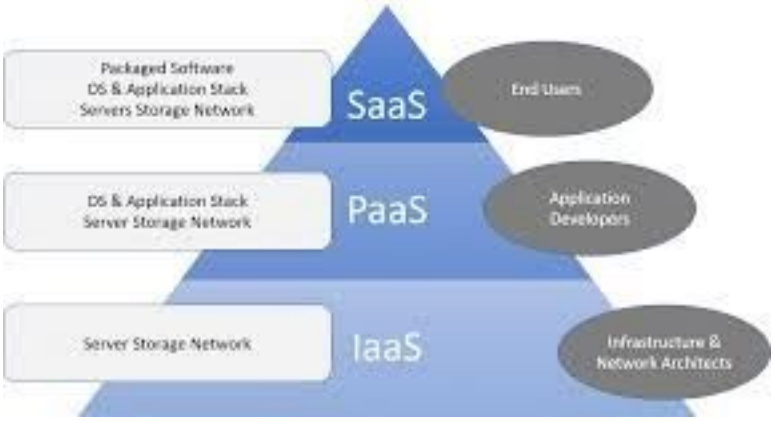


CS352: Cloud Computing Jan-May 2019

Container Orchestrator

CS352 | 18/05/2019

SNo	Name	USN	Class/Section
1	B Shashank	01FB16ECS086	B
2	Bhavani Shankar	01FB16ECS089	B
3	Chandresh L	01FB16ECS095	B

Introduction

Our orchestrator has three major features – load balancing, fault tolerance and auto scaling. It divides the incoming HTTP requests equally among the running containers in a round-robin fashion. It checks for unhealthy containers, kills it and replaces it with a healthy container. It also deploys containers as the number of HTTP requests crosses the threshold.

Related work

Docker SDK for python - https://docker-py.readthedocs.io/en/stable/

Docker tutorial series - https://rominirani.com/docker-tutorial-series-a7e6ff90a023

Flask docs - http://flask.pocoo.org/docs/1.0/

EXPERIMENTS/DESIGN

We have created two APIs in addition to the 11 APIs – Health check API and Crash API.

Health check API checks if the container is healthy by checking if it is able to access the database. We do this by creating a dummy file in our volume. It returns 200 if the container is healthy and 500 if it is not.

The crash API crashes the container and returns 200 on success.

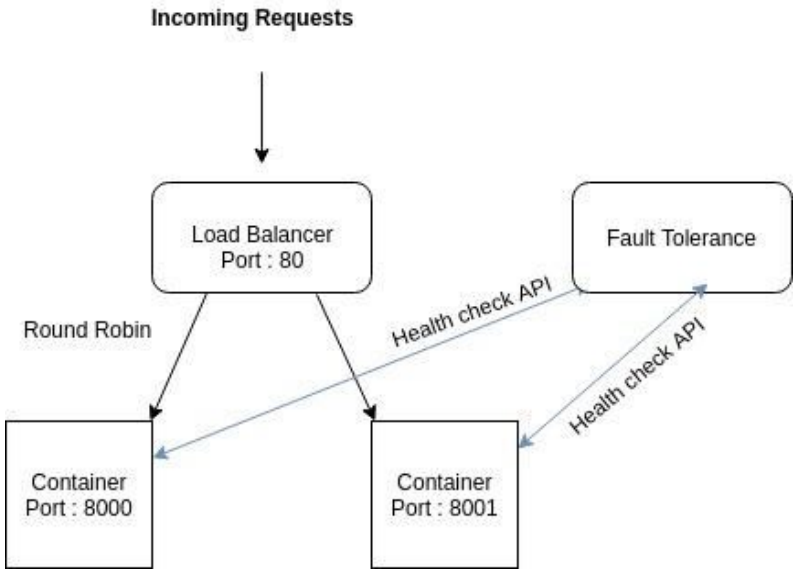
The load balancer feature of the orchestrator divides the incoming HTTP requests among running containers equally. This ensures that there is lesser load on the containers.

The fault tolerance feature ensures that there are no unhealthy containers. It does this by polling the health check API of each running containers every one second. Once it finds the unhealthy container it kills it and creates another container using the same act image , at the same port.

The auto scaling feature increases the number of containers when the requests crosses a certain threshold. Every 2 minutes it checks for the requests and deploys container accordingly. If the number of requests falls down then it kills the container.

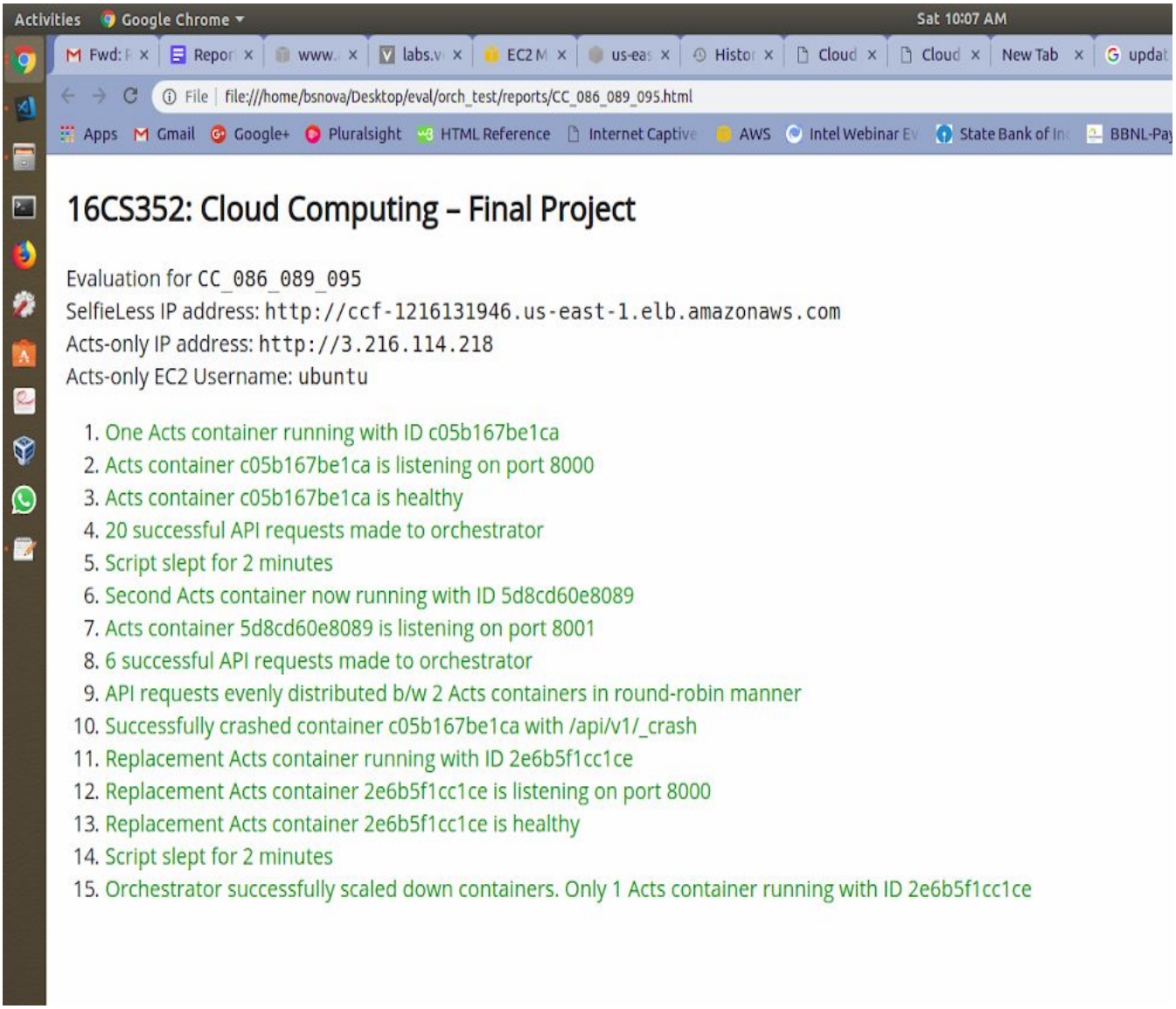
Additionally we have also maintained a log file which keeps tracks of the events that take place in the orchestrator.

DESIGN :



TESTING/RESULTS

Tested against the test.pyc file shared in the drive. We were able to integrate the backend with the app and upload images into it.



REFERENCES

Docker SDK for python - https://docker-py.readthedocs.io/en/stable/

Docker tutorial series - https://rominirani.com/docker-tutorial-series-a7e6ff90a023

Flask docs - http://flask.pocoo.org/docs/1.0/

.

EVALUATIONS (Leave this for the faculty)

Date	Evaluator	Comments	Score

CHECKLIST

SNo	Item	Status
1.	Source code documented	
2	Source code uploaded to CCBD server	
3	Source code in GitLab. Please do not upload your source code to github where it can be seen by everyone.	