17CS352:Cloud Computing

Class Project: Rideshare

Taxi Application

Date of Evaluation:16/05/2020
Evaluator(s):Venkatesh Prasad
Submission ID:1262
Automated submission score: 5

SNo	Name	USN	Class/Section
1	Jayanth B	PES1201802374	C
2	Prithvi V	PES1201700716	C
3	Hrushikesh HS	PES1201802394	C
4	Tejas Sirigiri	PES1201802376	C

Introduction

This project runs along the same lines of applications like ola and uber.The RideShare allows the users to create a new ride if they are travelling from point A to point B.It has been implemented using flask i.e, each application program interface that was created was created using flask.Then,for the storage of information,mongodb was used.In the second phase of the project docker was been used,in which 2 containers were created,one for rides and another for users.In the third phase a load balancer was created in which it helps in sending requests to it's appropriate container whenever a request is made .In the final project a DbaaS service was used,in which it consists of an orchestrator and master and slave workers.Rabbitmq was used for the requests to be queued in 4 different queues,i.e,Read queue,Write queue,Sync queue and Response queue.Three containers were created ,one for users,another for rides and another for the DbaaS service.

Related work

We made use of the materials provided by the faculty on piazza and the links provided.

We also made use of resources from the web like stackoverflow for the correction of multiple errors that were causing problems and the mongo website for the creation of schemaless database.We also reffered to the rabbitmq and zookeper website for the setting up of the orchestrator.

Reference links-

https://www.rabbitmq.com/getstarted.html

https://zookeeper.apache.org/doc/r3.4.11/zookeeperTutorial.html

https://docs.mongodb.com/

https://stackoverflow.com/questions/23013220/max-retries-exceeded-with-url-in-requests

ALGORITHM/DESIGN

The algorithm we made use of was the round robin fashion algorithm.This algorithm is been used to store and send messages/requests to the read queue,write queue,sync queue and response queue.For eg:whenever a write request from the users container is sent ,the request is stored in the write queue and then sent to the master workers and then the requests is sent to the sync queue in which requests are inturn sent to the slave workers and then sent back to the orchestrator.Similarly whenever there's a read request it's sent to the read queue and the response is sent from the slave worker back to the response queue,which is then sent to the orchestrator.With respect to scaling whenever a slave crashes, the zookeeper calls the crash api to create a new slave worker.

TESTING

We had tested the following-

1)To check if the API's are responding or failing .

2)To check if the API's respond with the correct output or not.

3)To check if the API's respond with the appropriate response code.

4)To check if the containers that were created was working.

5)To check if the Orchestrator was working or not

6) To check if the requests are going to it's respective queues.

7)To check if zookeeper was running without any hassle.

CHALLENGES

The challenges we faced -

1)Getting rid of errors caused because of unmatched port numbers.

Solution-could change and match the port numbers eventually

2) An error in the scaling.py file which kept showing us network not found.

Solution- corrected it by giving the correct network name.

3)Load balancer not working due to wrong availability zones with respect to the instances created for each target groups.

Solution- matched the availability zones of target groups with respect to its instances.

4)Failed create new ride and failed get upcoming ride for source and destination and failed to get ride id.

Solution-this was due to an error in code ,which was eventually fixed

5)Failed getting request on aws ec2-instance .

Solution-this was due to not providing proper inbound rules.

Contributions

Jayanth B(PES1201802734) -Load Balancer and RabbitMQ(channel exchange,queue creation and message exchange) .Master and slave DB for writes.

Prithvi V(PES1201700716) -Slave Replication ,Slave Fault Tolerence,RabbitMQ(channel exchange,queue creation and message exchange).

Hrushikesh HS(PES1201802394) - API's

Tejas S(PES1201802376) - API's

CHECKLIST

SNo	Item	Status
1.	Source code documented	Done
2	Source code uploaded to private github repository	Done
3	Instructions for building and running the code. Your code must be usable out of the box.	Done