

17CS352:Cloud Computing

Class Project: Ride share

Date of Evaluation : 16/05/2020

Evaluator(s) : Pushpa,Nitin

Submission ID : 1360

Automated submission score : 10

|  |  |  |  |
| --- | --- | --- | --- |
| SNo | Name | USN | Class/Section |
| 1 | Syed Zubair | PES 1201701395 | D |
| 2 | Laxman M | PES 1201701632 | C |

# Introduction

We have tried to implement the DataBase as a service model for our REST API’s. Our API’s send the read and write request to the orchestrator for which read and write request is divided. Read request is completed by the slave worker and write is done by the master worker ( they both run in a separate container ).

The Orchestrator puts the request in readQ . Slaves reads the request from the readQ in round robin fashion , reads the data from DB and puts the data back in responseQ.

When the write request arrives the request is put in writeQ and the master reads the requests form the queue and writes to its database and gives it to syncQ . All the Slaves take data from there and write to their own Db’s

## Related work

We have referred below links:

1.Docker SDK for creating container at run time

<https://docker-py.readthedocs.io/en/stable/containers.html>

2. used Kazoo to binding with zookeeper

<https://kazoo.readthedocs.io/en/latest/async_usage.htm>

3. To cover the basics of creating messaging applications using RabbitMQ.Â

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

4. Copying the database asynchronously

<https://docs.mongodb.com/manual/reference/method/db.copyDatabase/>

https://stackoverflow.com/questions/42912755/how-to-create-a-db-for-mongodb-container-on-start-up/42917632#42917632

## ALGORITHM/DESIGN

We have used three threads in Orchestrator. One working for all the Queues.Other working for zookeeper and the last one for the Auto scaling by counting the read requests.

Used the fanout exchange method in syncQ . For making the database consistent among slave workers. That run separately and add the contents to its database when the data arrives in the queue.

## TESTING

We have tested all the API’s through the postman.we successfully got the response with status code.

We checked the test cases that were required to get marks, the inputs that we're being tested. It came to notice that scaling was failing due to a bit delay. So it was put nearly to 2 min 30 sec which eventually passed all the test cases.

## CHALLENGES

When auto scaling we had to give names to the container for slaves and DB’s.Random names could be given. But we were getting errors due to restart policy.The error was that the same name could not be there so we used docker.api.errors. Due to this whenever the same name error would come we pruned that name of the container and rebuilt it.It worked well.

We used mongodb as a Database. It required authentication prior before its usage, when a new container of its image was created.Before we did it manually as no new images were built at run time. When auto scaling it was a problem. We got a javascript file (mongo-init.js file) for automated authentication at the run time.

Another was to replicate the database. We thought of copying the data from the master as it was the complete one at any time of execution. So we used the db.copy command which was available in mongo 4.0 version.

When achieving fault tolerant slave workers.it was very difficult to differentiate scaling slaves and crashed slaves .we have used exit state of the slave to differentiate them.if exit state of the slave is 137 then it is crashed slave.otherwise it is scaling slave.

## Contributions

As we were a team of two it was all done by both of us.

Rabbitmq was focused by Syed zubair

Zookeeper was focused by Laxman M

## 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 |

Github link:

<https://github.com/syedzubr/Cloud-Computing>