UE17CS352: Cloud Computing

Class Project: Rideshare

Date of Evaluation: 16th May, 2020
Evaluator(s): Prof. K. Srinivas
Submission ID: 1409
Automated submission score: 10

SNo	Name	USN	Class/Section
1	Sarthak Gupta	PES1201700077	6G
2	Dev Bhartra	PES1201700186	6G
3	Dhruv Vohra	PES1201700281	6G

INTRODUCTION

We have created the backend for a cloud based RideShare application , that can be used to pool rides. Using this, the user can add themselves, delete a user, create a new ride, search for rides, join existing rides, and delete rides.

STUDY MATERIAL

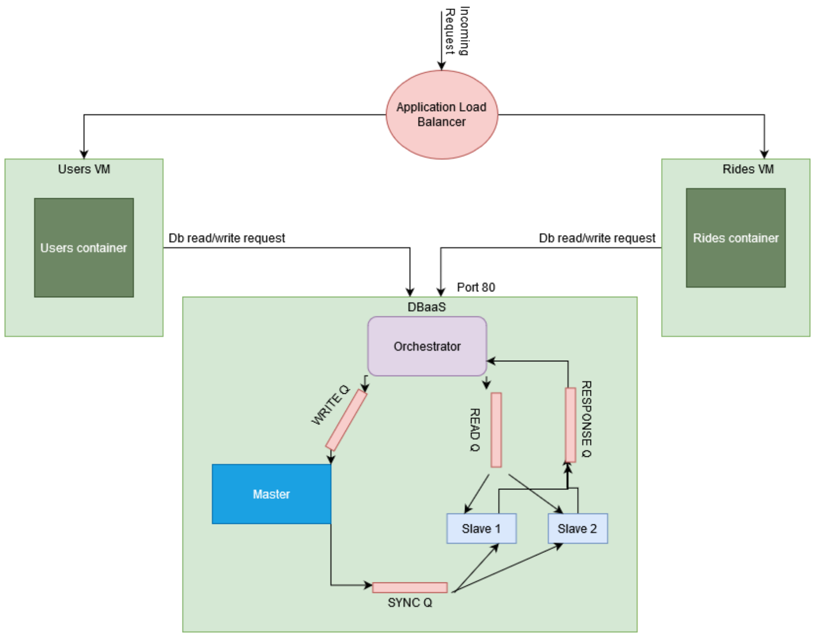
Study material includes Flask, Mongodb, Docker, RabbitMQ, Nginx, Redis, Gunicorn and Zookeeper documentations along with the links provided in the assignment specifications.

DESIGN

The core technologies used to build this project were

- **Flask RESTful**, for creating the API's in a RESTful architecture,
- **MongoDB**, as the database,
- **RabbitMQ**, as a message broker,
- and **AWS** as the host cloud platform.

The API's are categorized based on functionality and are containerized in separate virtual machines as **microservices**. An AWS ELB load balancer is used to control traffic flow to the EC2 instances and route the requests based on the microservice. **Nginx** is used as the web server, with **Gunicorn** acting as the Web Server Gateway Interface (WSGI). Database operations are managed by a custom highly available, fault tolerant DBaaS. **Redis** is used as an in-memory database for internal operations. **Docker** is used for container orchestration, and the **Docker SDK** is used to programmatically bring up containers for scaling purposes.



TESTING

- Making sure that the data did not get lost in the pipeline, and was delivered on time and in the correct format.
- Catching Exceptions, handling edge cases, while always returning correct status codes.
- Ensuring AWS services were configured correctly.

CHALLENGES

- Deciding on the right kind of architecture for our stack and use case.
- Syncing old data to newly spawned worker slaves.
- Scaling without breaking or dropping any requests.
- Learning new technologies and implementing them to build an error free system.
- Writing decoupled generalised code so that code can be maintained or changed on change in specifications.

CONTRIBUTIONS

- **Sarthak** - Core API logic, DB operations and sync, Debugging, Fault Tolerance, AWS
- **Dev** - Orchestrator using RabbitMQ, Docker, Debugging, AWS config
- **Dhruv** - Orchestrator using RabbitMQ, Scaling using Docker SDK, AWS config

CHECKLIST

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