**Development of todo app**

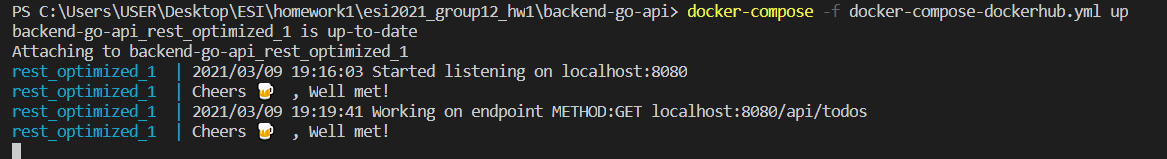
Request for tester:

1. Please create the network using docker network **docker network create external-example**
2. Please inspect the network and put the ip of backend network to the **API\_URL** of **main.go** file in **frontend app.**

Task 1:

The task 1 consisted of making the endpoint for todo api and exposing the endpoints so that it can be consumed in frontend.

There is a use of emoji feature in log of backend api. Emoji will be shown when we run the application using **go run main.go** as shown below:



The exposed api endpoint are given below:

|  |  |  |
| --- | --- | --- |
| Feature | Endpoint | Request type |
| List todos | localhost:8080/api/todos | GET |
| Add todos | localhost:8080/api/todos | POST |
| Delete todos | localhost:8080/api/todos/id | DELETE |
| Mark to do as complete | localhost:8080/api/todos/complete/id | POST |
| List individual to do | localhost:8080/api/todos/id | GET |

There is emoji feature that makes our TODO app unique.

**Task 4:**

In task 4 we should setup the docker registry in cloud machine(for this we choose dockerhub). The image has been uploaded to the dockerhub. The image name are **rabindradahal/frontendv1.0** and **rabindradahal/gorestv1.0.**

Since my team mate is assigned the task 3, but he could not complete the task because he is not able to learn the go language and docker in the pace of lecture. So, as a requirement of lab 4, I have made the dockerized image for frontend and backend.

There are mainly three docker compose file in each frontend and backend app.

1. docker-compose.yml file which consists of image of bigger size. Its size is more than 350MB.
2. docker-compose-dockerhub.yml which consists of image which is pushed to docker hub. Its size is also more than 350MB.
3. Docker-compose-optimized.yml file consists of optimized size of 12MB to 13MB.

Inorder to run these docker compose file, firstly we should build it using following command:

**docker-compose -f docker-compose.yml up --build -d --remove-orphans**

**docker-compose -f docker-compose-dockerhub.yml up --build -d --remove-orphans**

**docker-compose -f docker-compose-optimized.yml up --build -d --remove-orphans**

To run the docker compose file, we should use following command for backend app:

**docker-compose -f docker-compose.yml up**

Other are also written in similar manner.

To run the docker compose file, we should use following command for frontend app:

**docker-compose run frontend**

A network has to be created using **docker network create external-example**

**Time spent for the completion of task:**

It took almost 4 days to learn the go language and docker. Still I am not comfortable in writing docker networking stuff.

Inorder to complete the task1, it took almost 8 hours(4hr(first day) + 4hr(second day)).

Inorder to complete task 4, it took more than 12 hours. For writing dockerization of frontend and backend (around 6 - 7 hour)( I donot have good grasp of networking between docker container and it took me long time). There was some issue with frontend app dockerization and got answer via asking in #hw-disscussion and stackoverflow(<https://stackoverflow.com/questions/66544508/golang-infinite-for-loop-problem-with-docker-run>). It took almost more than 5-6 hours. To make the optimized build it took more than 1 hour for both frontend and backend.

Gain:

I learned GO language constructs, rest methodology using GO and docker command and making docker-compose file.

Pain:

Learning GO language and docker from scratch is really a pain. Docker networking is very hard to learn.

Note: Please use the <http://172.25.0.2:8080/api/todos> when testing with docker in frontend app.

When testing in local environment donot forget to change the api url with http://localhost:8080/api/todos