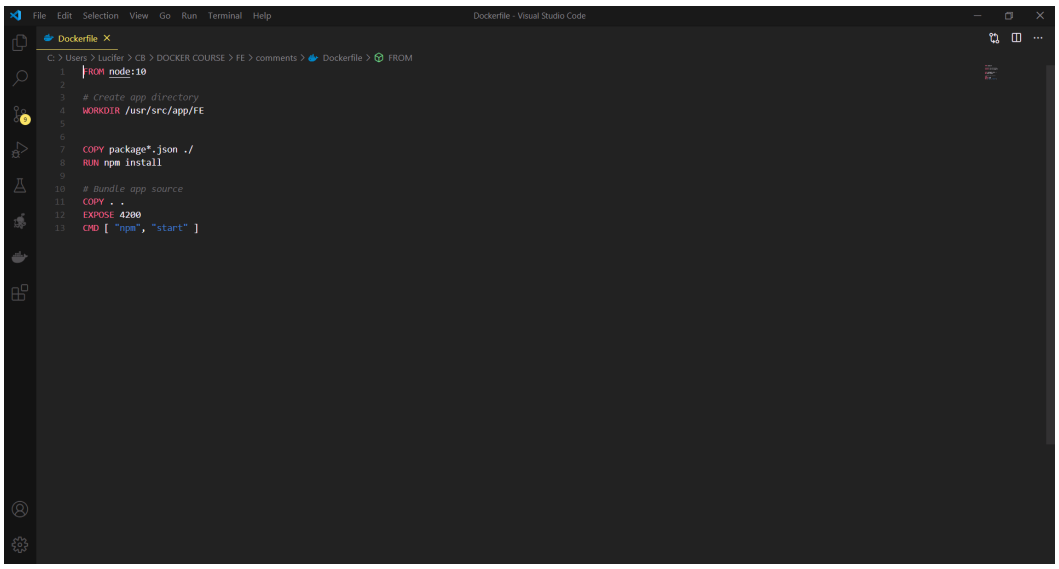
For this Docker 201 Project I have created a simple MEAN Application which takes the Comment and Username about a movie as Input and stores it in a MongoDB collection. On Application Startup, All the existing comments from the database are fetched and displayed.

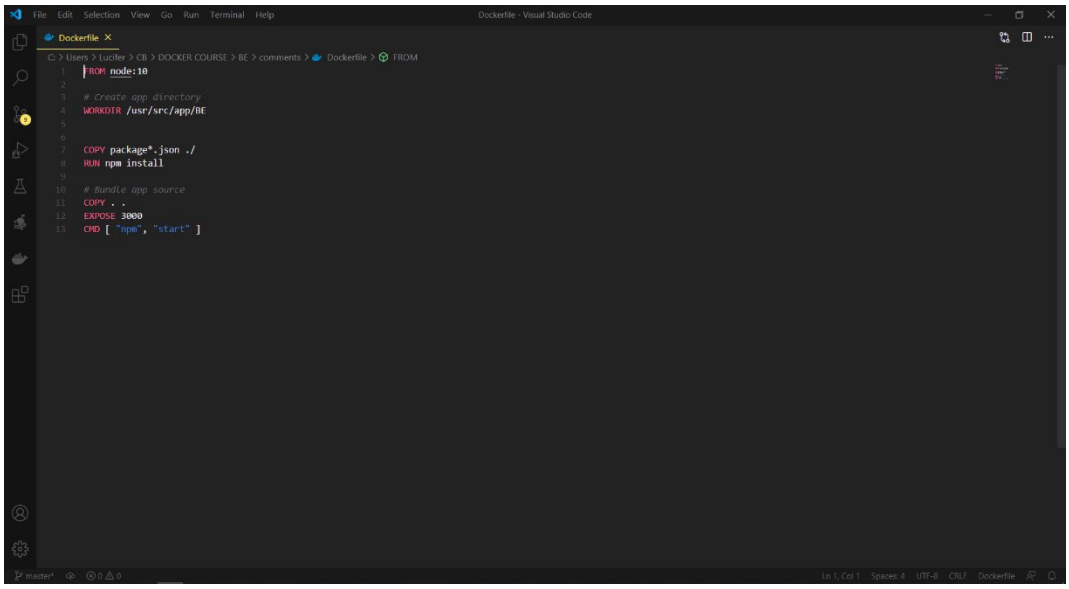
- The Application is containerized into 3 containers. One for Angular Front-End. One for NodeJS Back-End and one for MongoDB Database.

- The DockerFile to build the Front-End and Back-End images from code is shown below

Front-End DockerFile (Located in “DOCKER COURSE/FE/comments”)

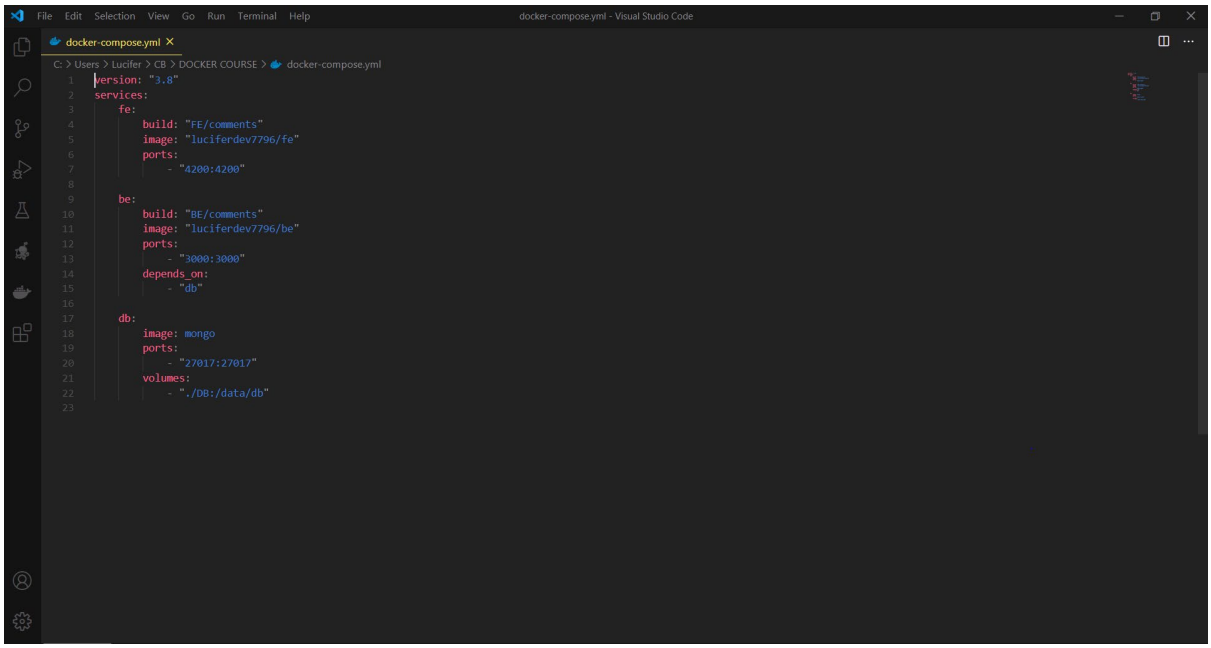


Back-End DockerFile (Located in “DOCKER COURSE/BE/comments”)



- Both Folders contain a ".dockerignore" to avoid adding the node_modules folder to the image.

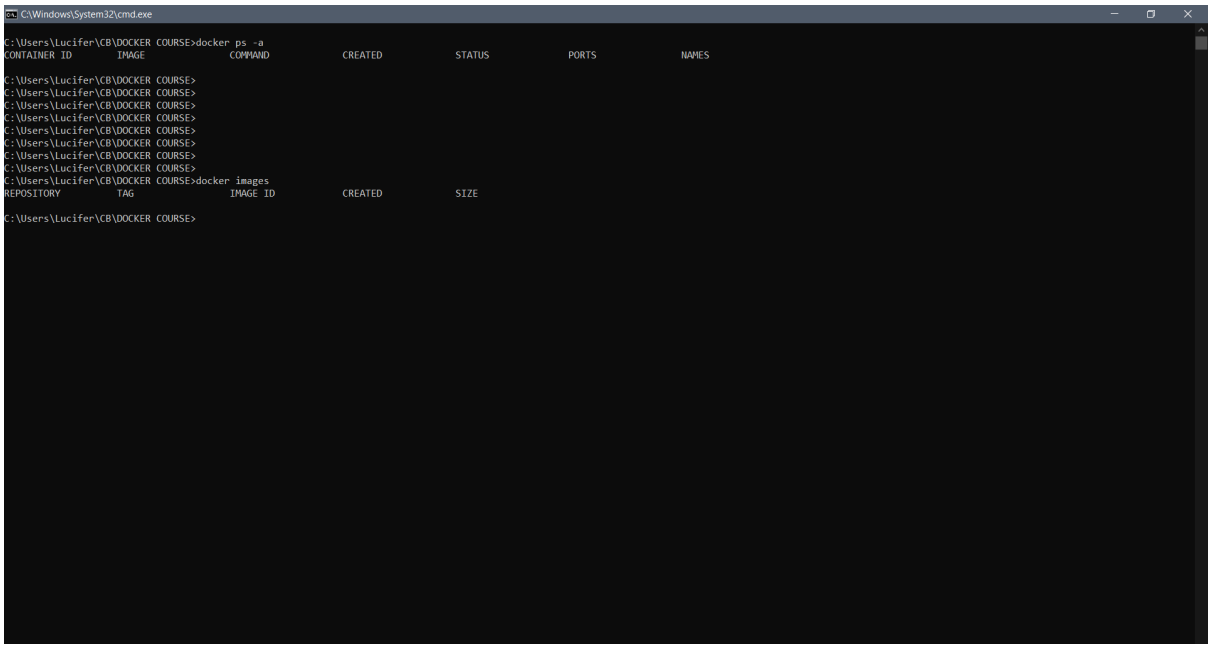
Shown below is the docker-compose file written to build, run and start server for the 3 containers.



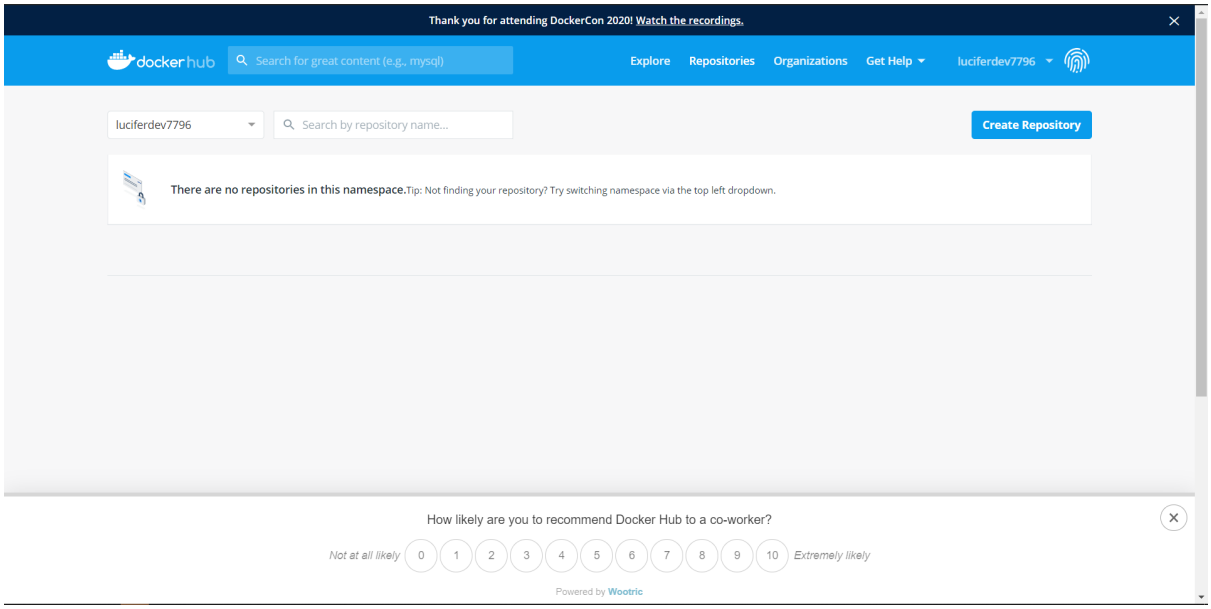
As seen above there are 3 services, one each for Front-End, Back-End and Database.

Execution:-

1. Initially there are no containers or images on the machine. if there are, they can be removed using
   “docker rm <container name>” to remove container and
   “docker rmi <image name>” to remove image respectively.



2. There are no images on our personal Docker Hub Registry either.

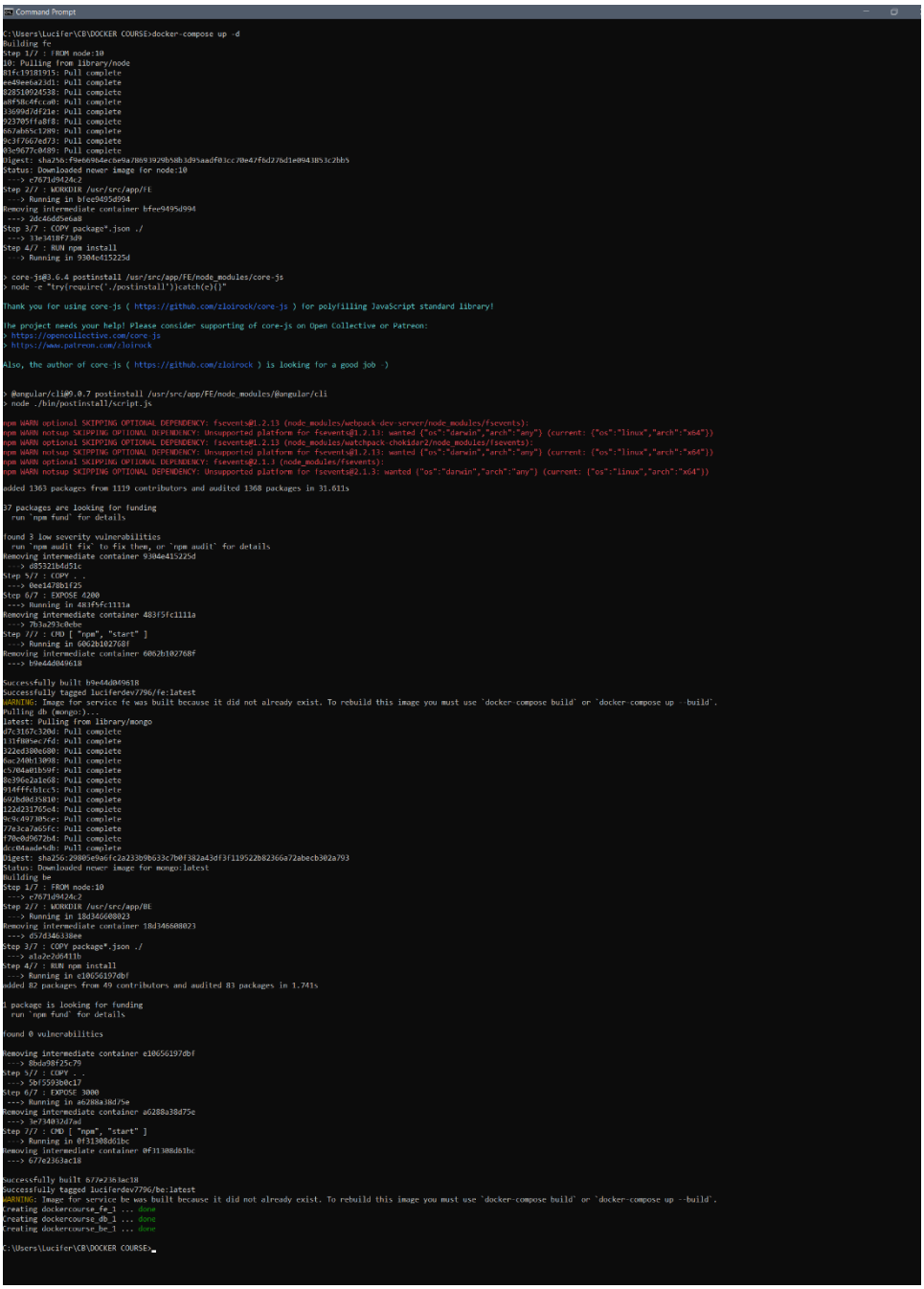


3. Now we will open a cmd where the docker-compose file is present and run the command

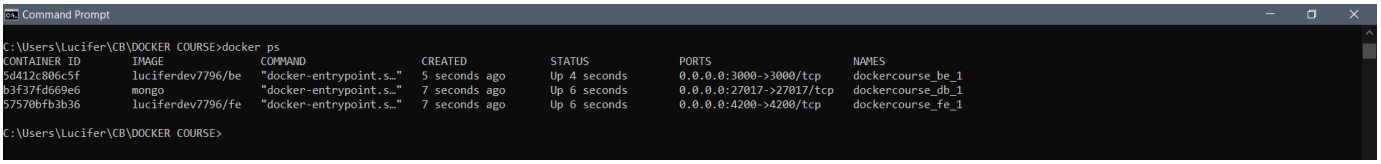
   “docker-compose up -d”

   This will build all the images and spin up containers for the same and start the servers of the applications for us.

   Output:-

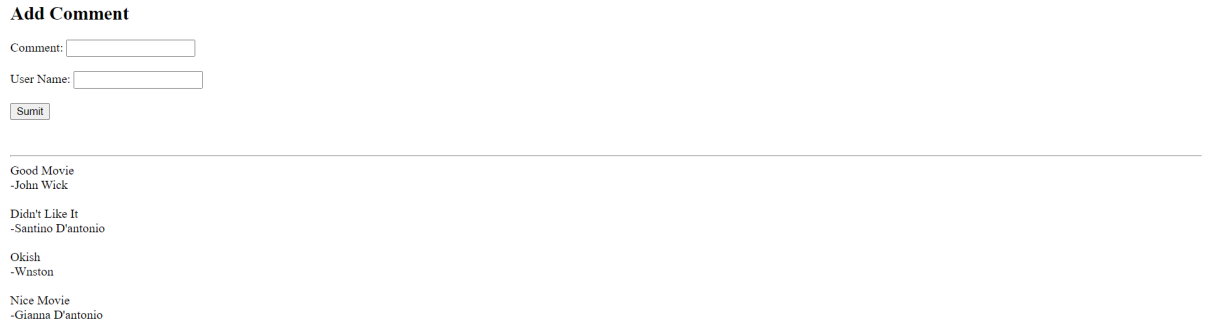


4. Now if we do
   “docker ps”
   We should see our 3 containers running.

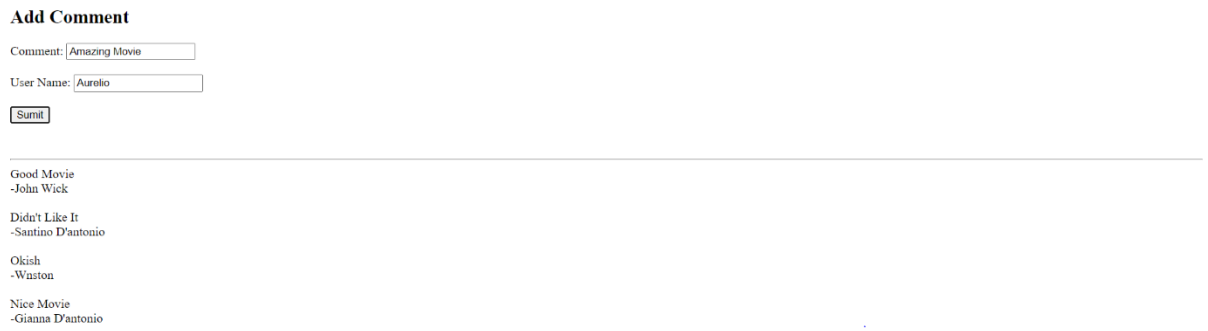


5. We can access our Application at the Shown ports
   (4200 for Front-End, 3000 for Back-End and 27017 for database)

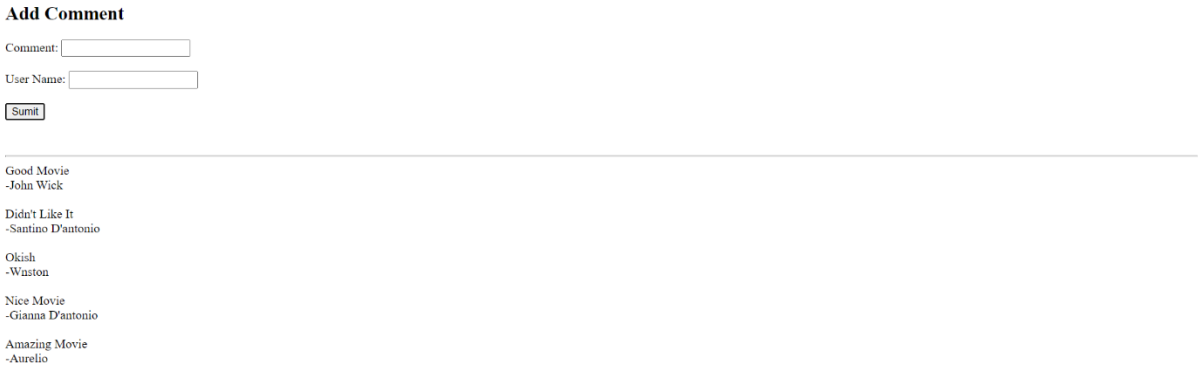
6. Opening localhost:4200 gives us the app



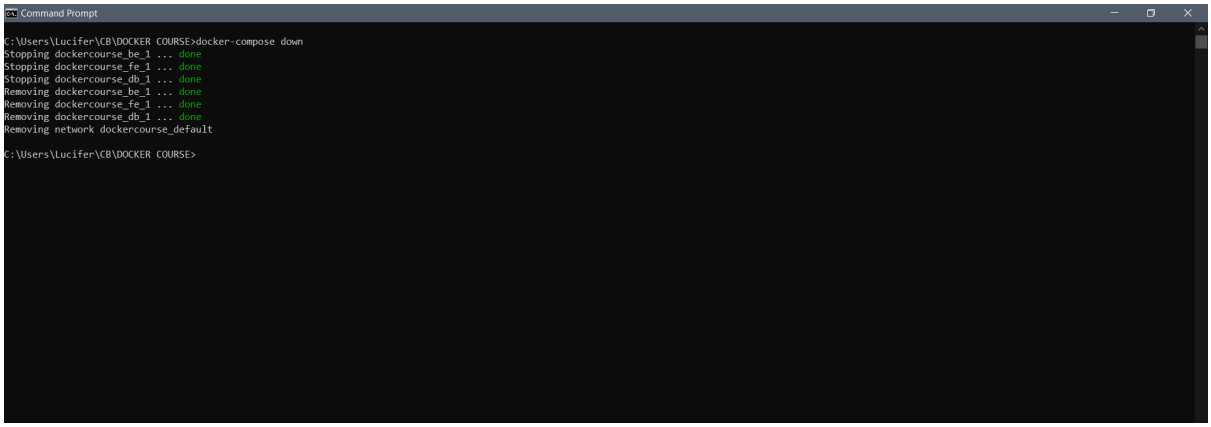
7. Now we'll add a new comment



8. The comment gets saved below in the list of comments



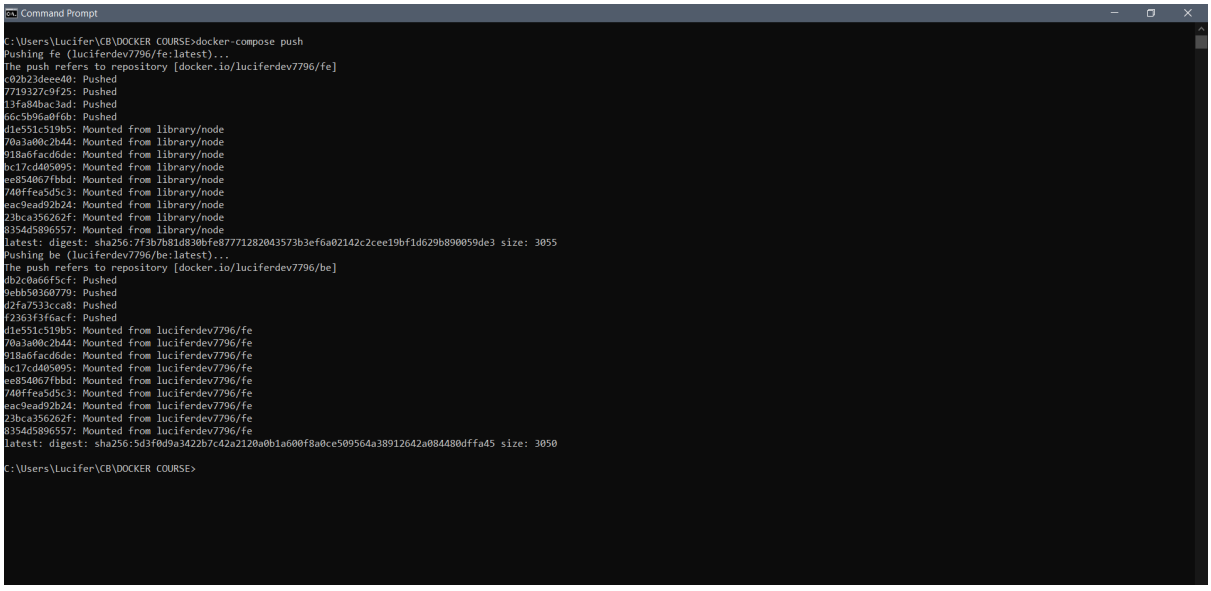
9. To stop all the containers we do
   “docker-compose down”



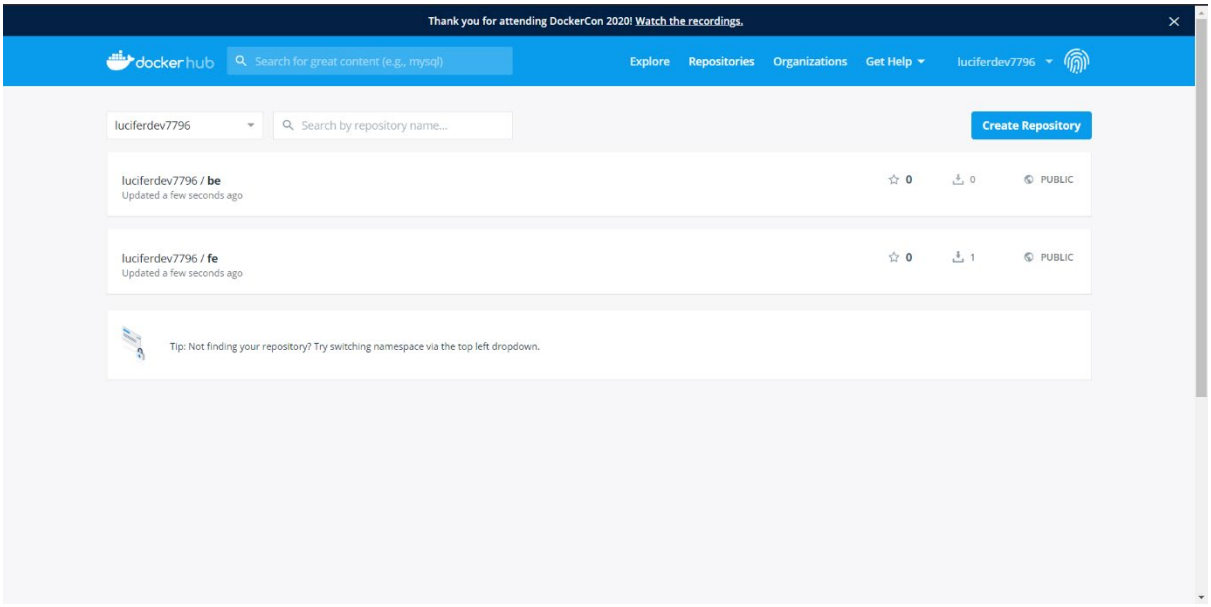
10. Now the containers are stopped. If we again do “docker-compose up -d” and open the app again, We see that the data is not lost and is preserved because in the docker-compose.yml, We have specified the volume configuration which maps a host folder to the container folder so the database is stored in the “DOCKER COURSE\DB” folder rather than in the container preventing loss on restart.

11. Finally we can push our Front-End and Back-End images to our registry on docker hub using the command

    “docker-compose push”



12. If we browse to our personal Docker Hub Registry on the web, we can find our newly pushed images there.



13. “docker-compose push” is equivalent to running the “docker push” command on both the images independently one after the other

NOTE: We do not need to push the database image as we ourselves have pulled the mongo image from docker hub (done by the docker-compose file for us). So no point pushing an image we had pulled in the first place as it's already present in the Registry.