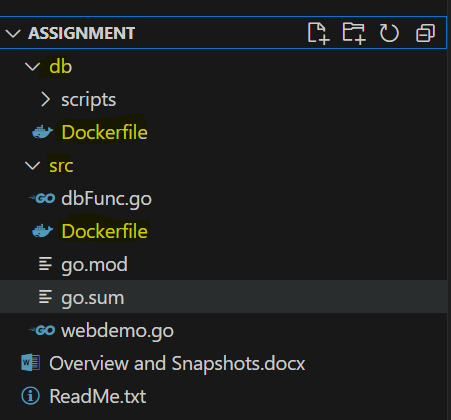
I am developing a small multiservice application, where there will be one webapp service (web APIs) and another one will be database service.

To achieve this, I have written a small webapp in Golang and using MS SQL Server as database to store and fetch data through webapp api. WebApp has API like check\_db, execSql etc.

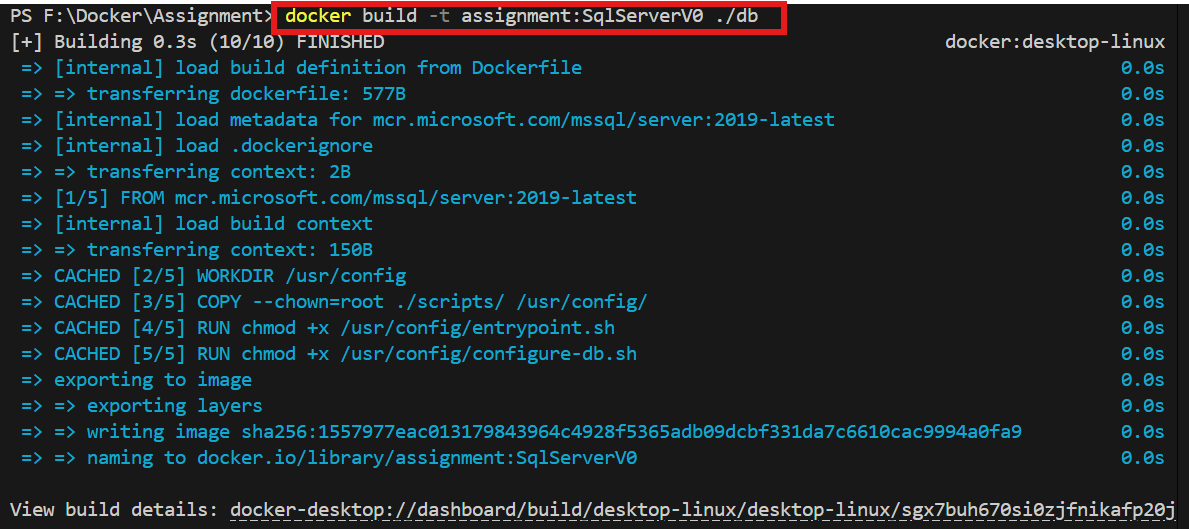
* Below is the source code of this assignment:

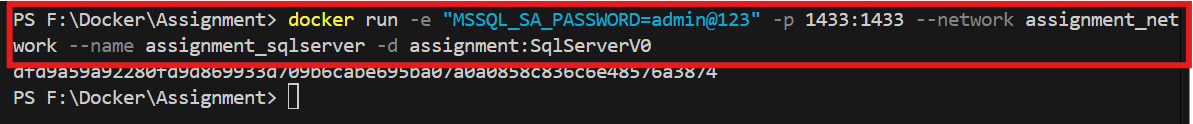


Inside “db” directory, there is Dockerfile for database docker image and the scripts, which will be executed, when container will run.

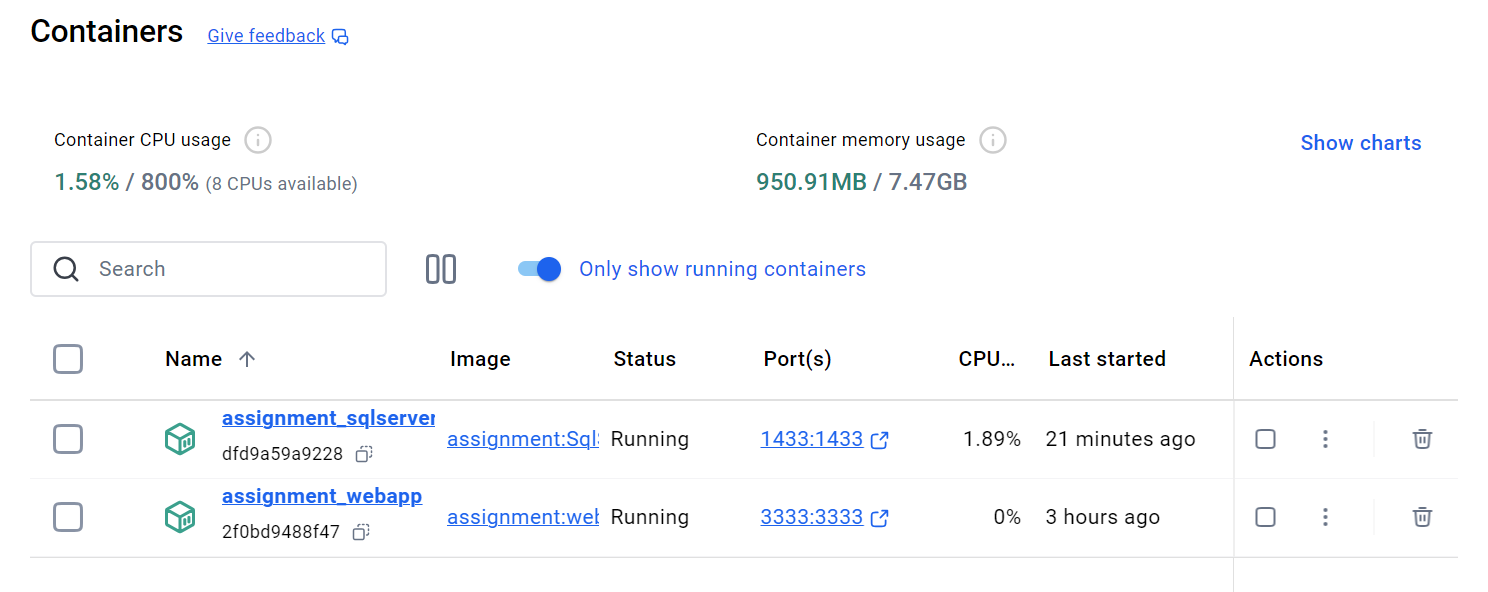
Likewise, inside “src” directory, there is code for webapp along with its Dockerfile for webapp’s docker image.

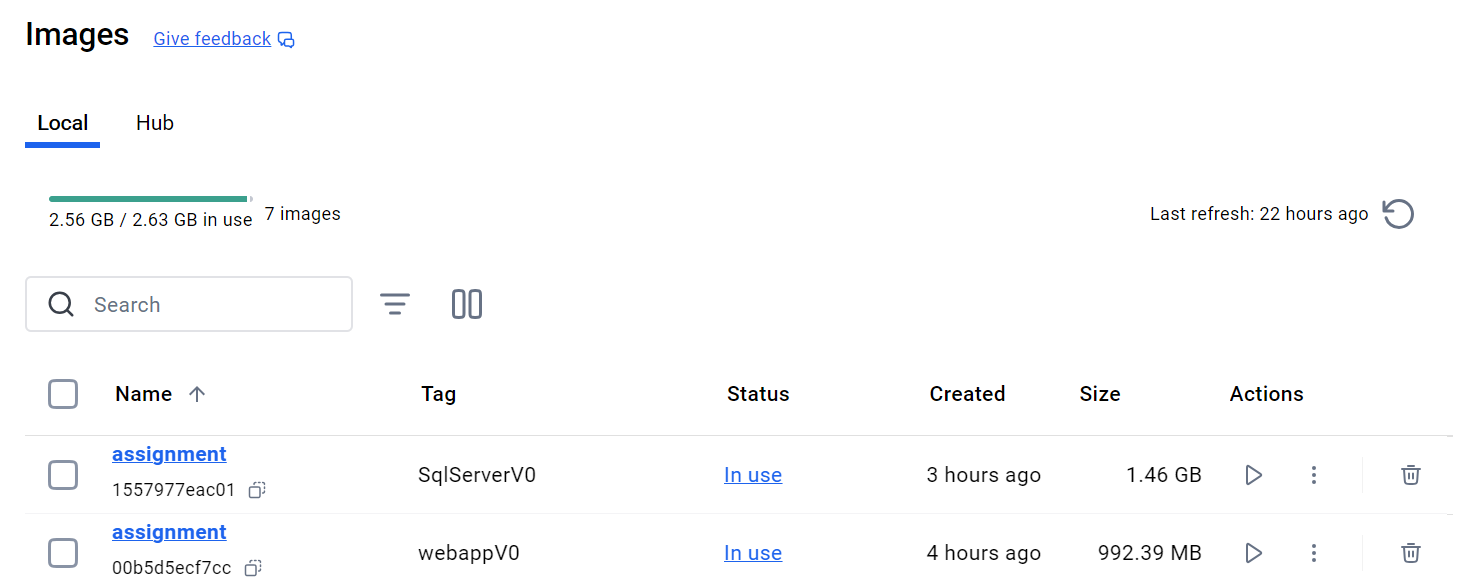
“ReadMe.txt” file has all the command to setup both the containers along with the webapp api execution instructions.



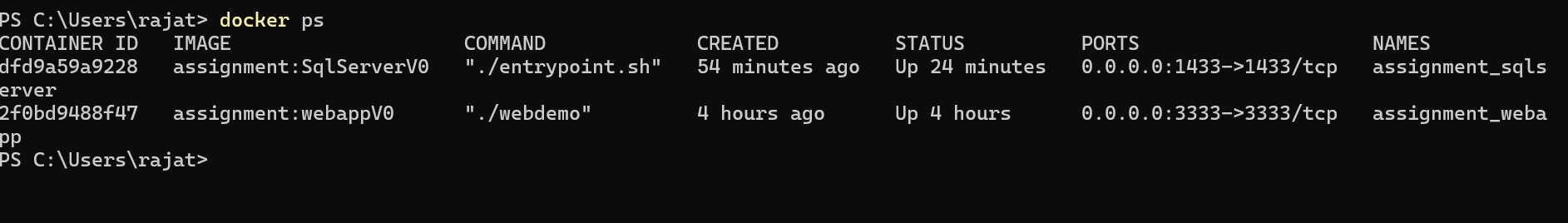


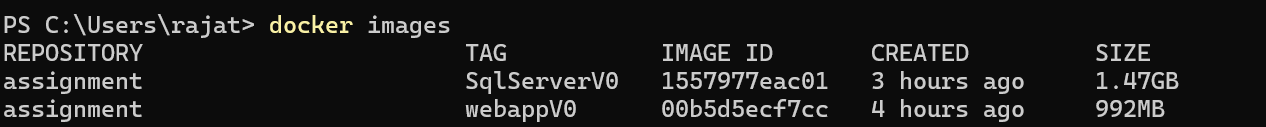
* After executing the commands specified in ReadMe.txt, below are the containers and images created:



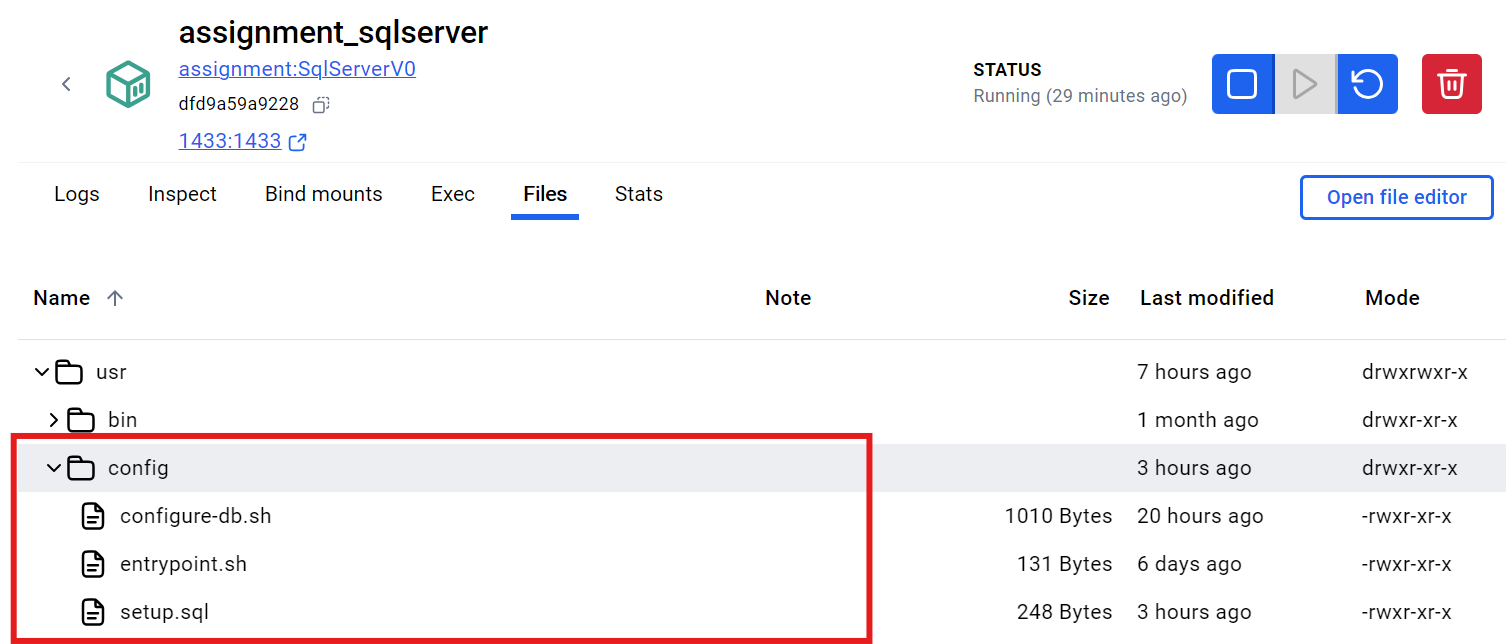


* The same can be seen through docker commands on terminal:

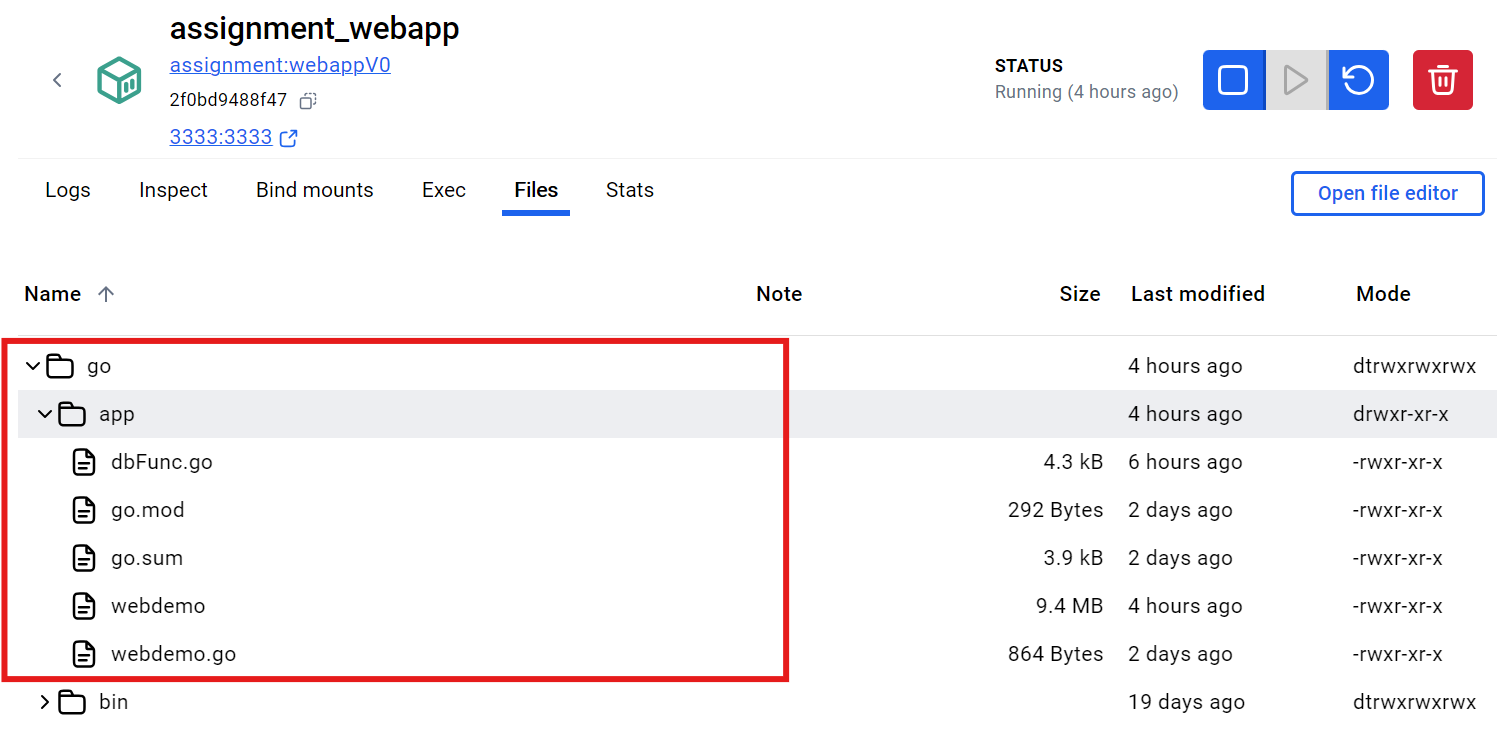




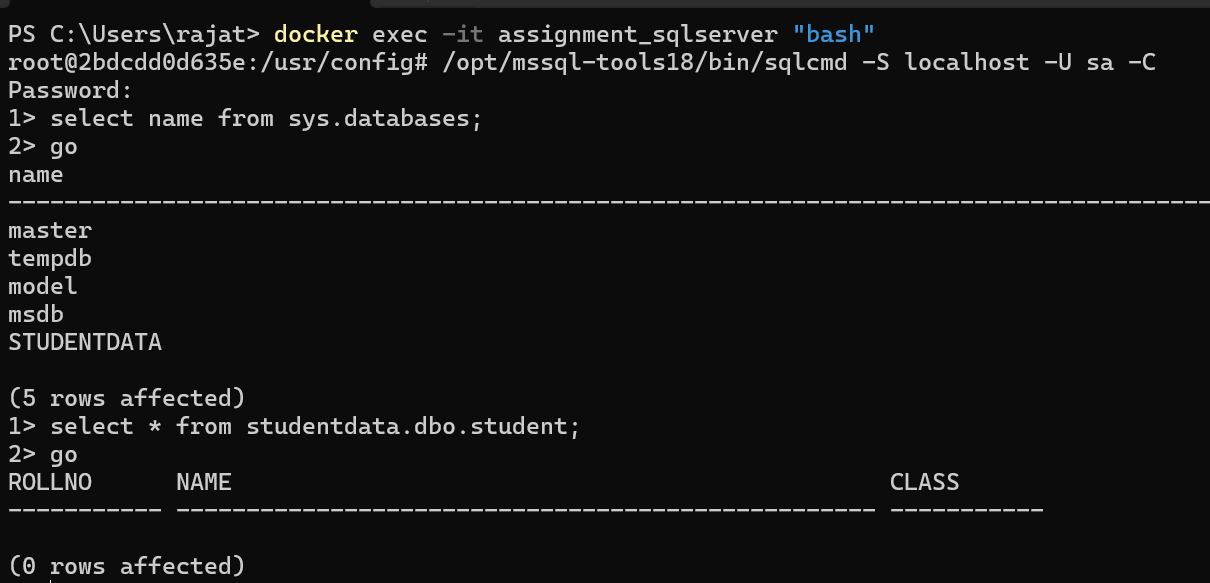
* Below are the scripts that has been copied into the mssql database container -



* Below is the webapp codebase which we have copied into the webapp container –

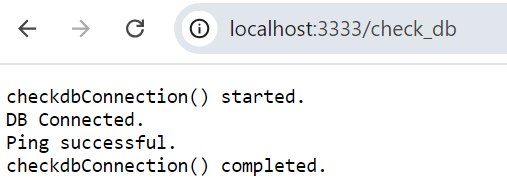


* To check, if database container is successfully created and running (STUDENTDATA is the database that we have created through script) – [Note: pass is ‘admin@1234’]

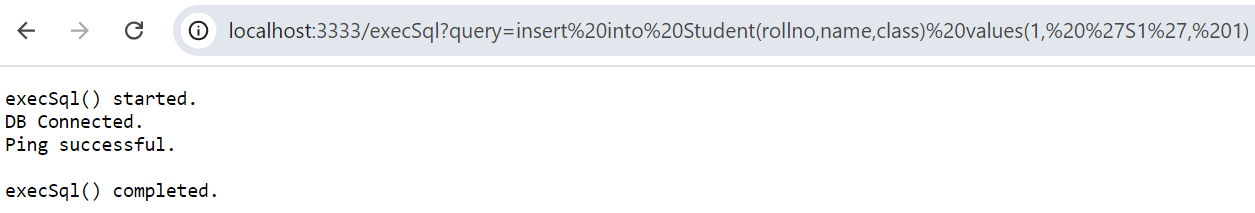


* Below are the snapshots of few web Api running –

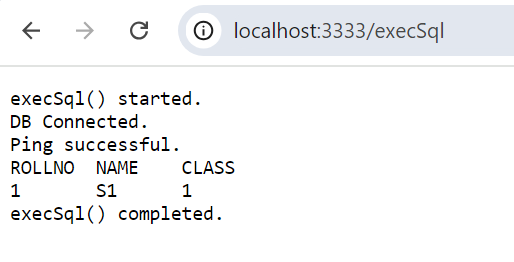
**http://localhost:3333/check\_db**



**http://localhost:3333/execSql?query=insert into Student(rollno,name,class) values(1, 'S1', 1)**

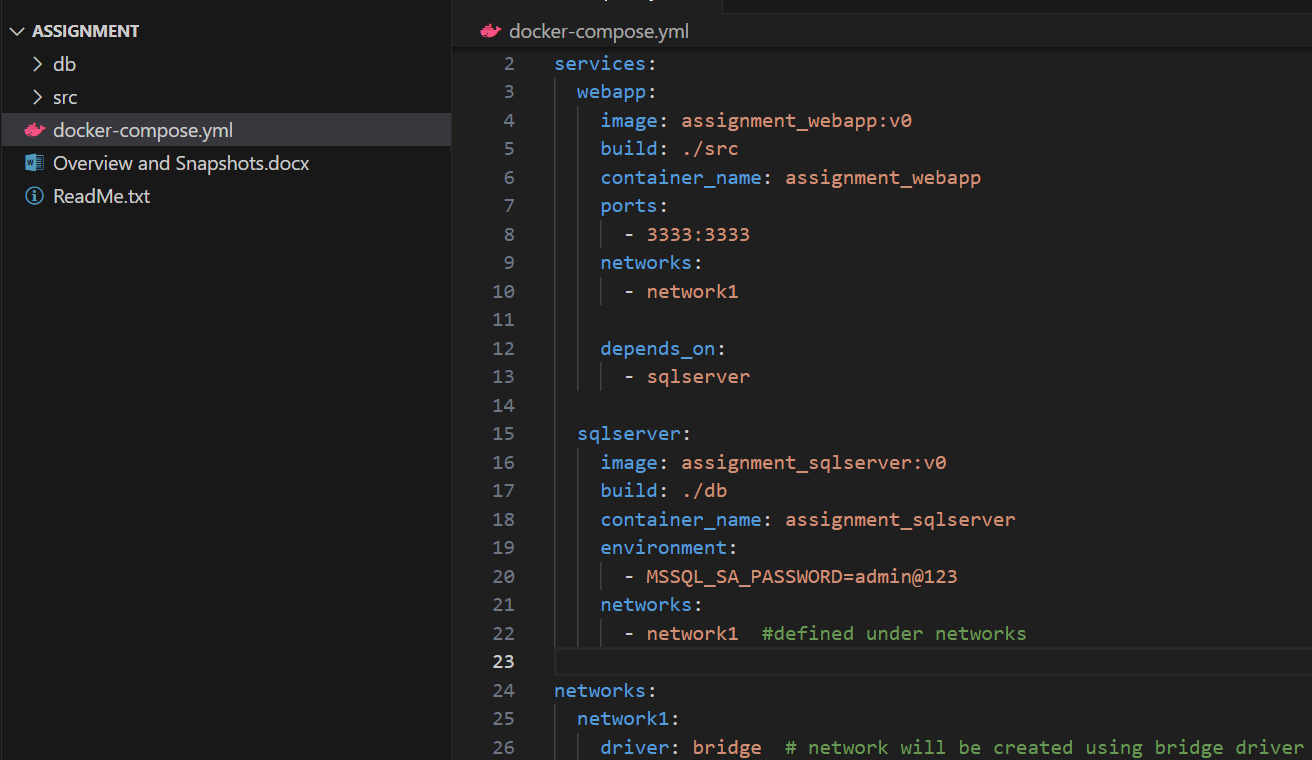


**http://localhost:3333/execSql**



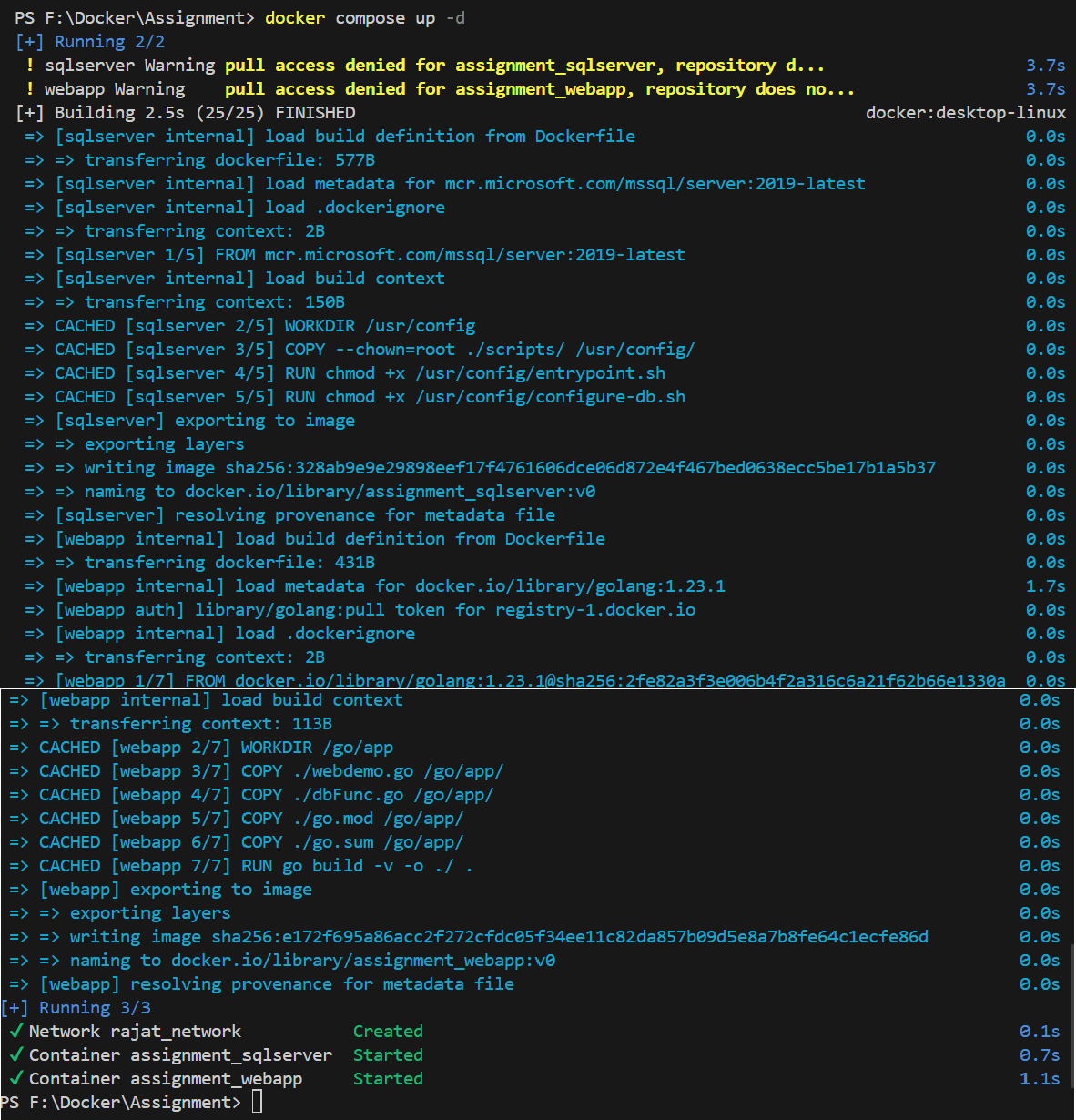
**Using Docker compose command –**

I have added a “docker-compose.yml” file in the project and write the instructions in it, as shown below:

****

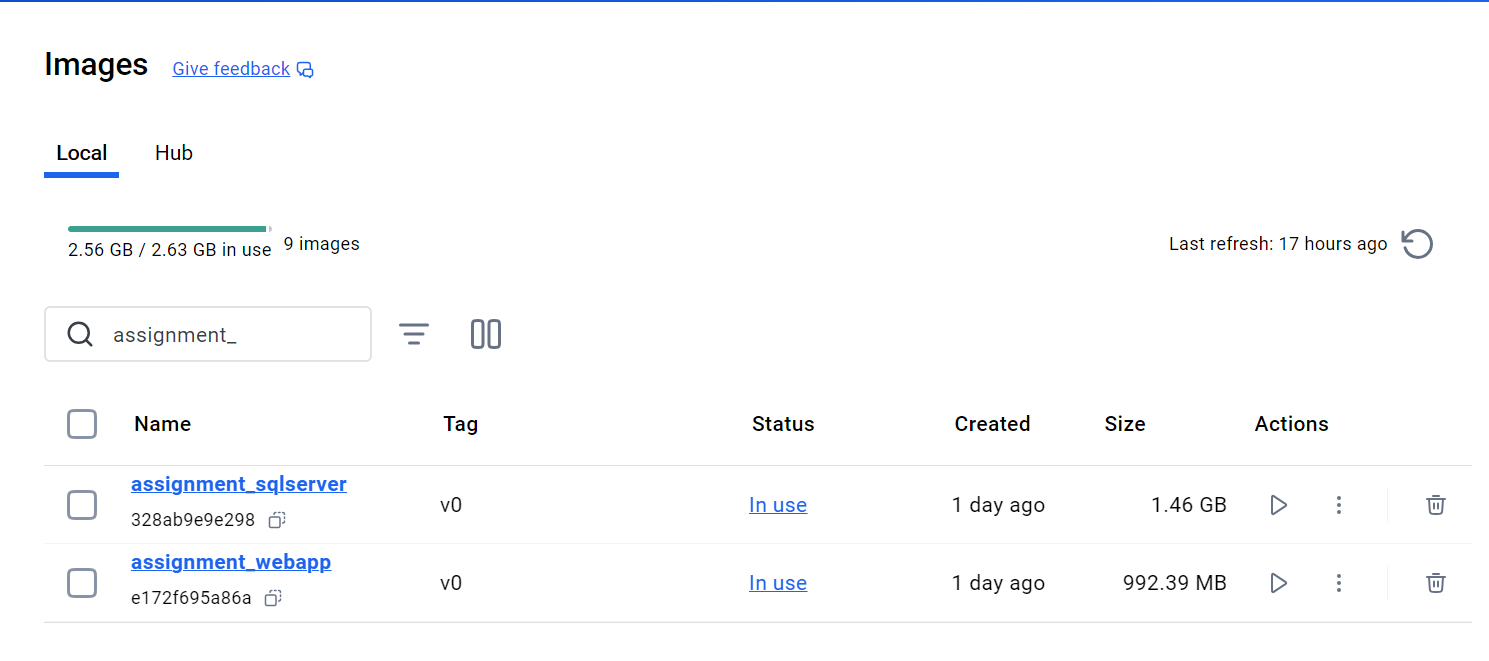
Then, from terminal, go to the Assignment directory and run the below command –

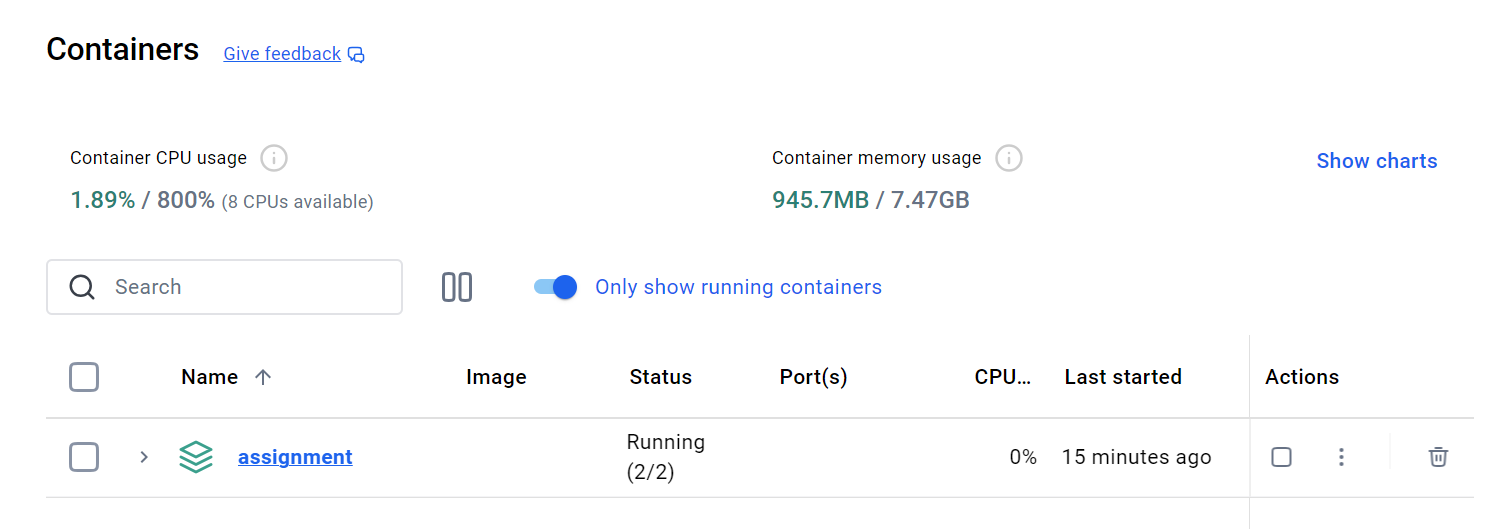
* Docker compose up -d

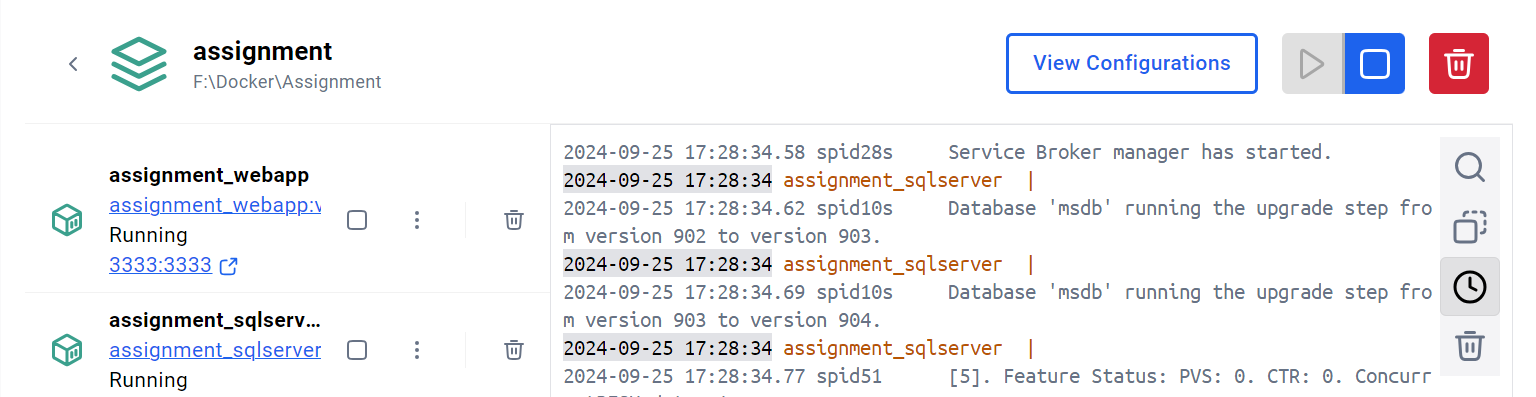


As we can see, there is one network and two containers have been created.

Same can be seen in docker desktop –







Now go to browser and run all the APIs as shown earlier.

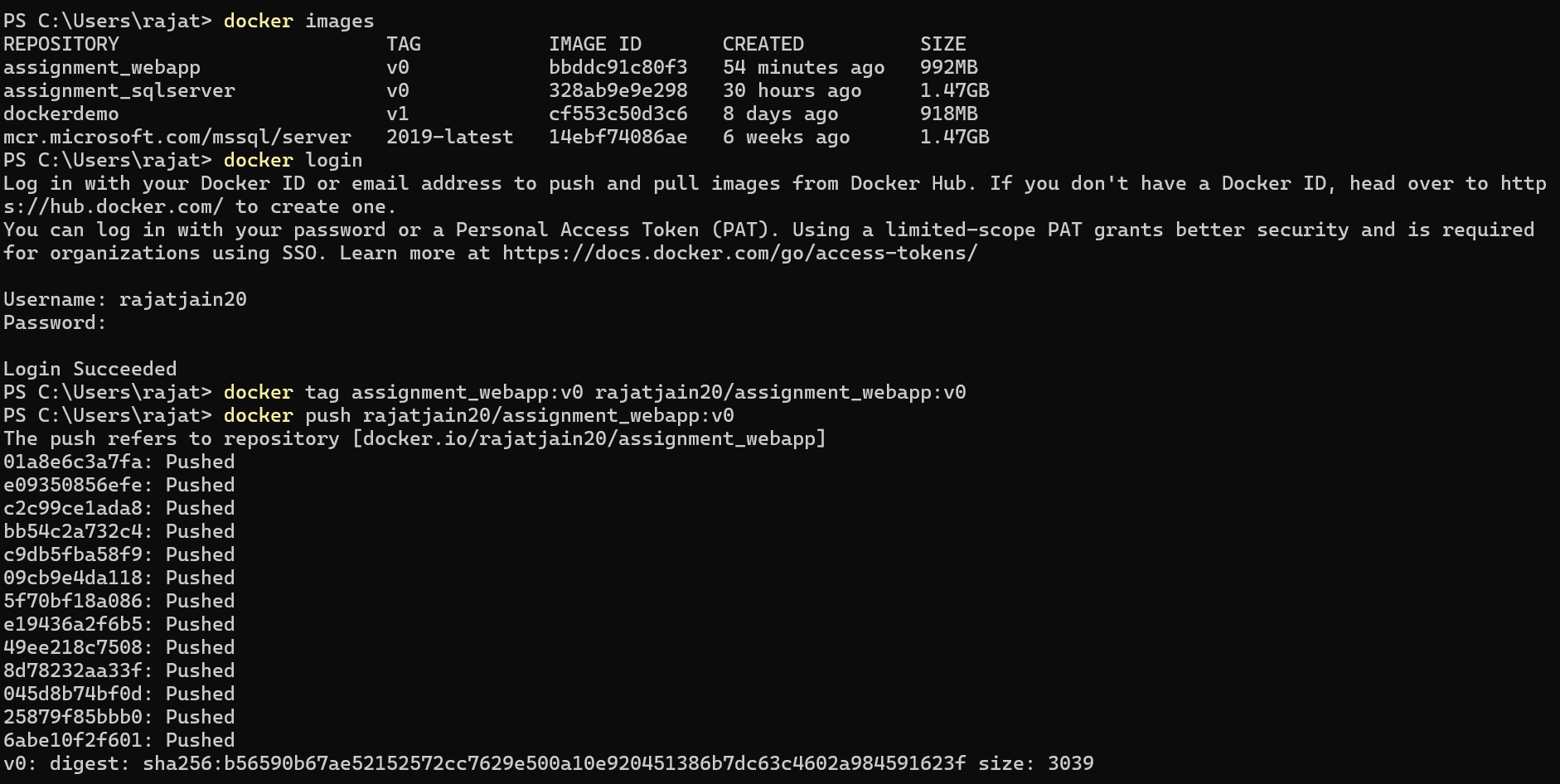
**Pushing Docker Images to Docker Registry :**

Commands:

* docker login

Pushing webapp image:

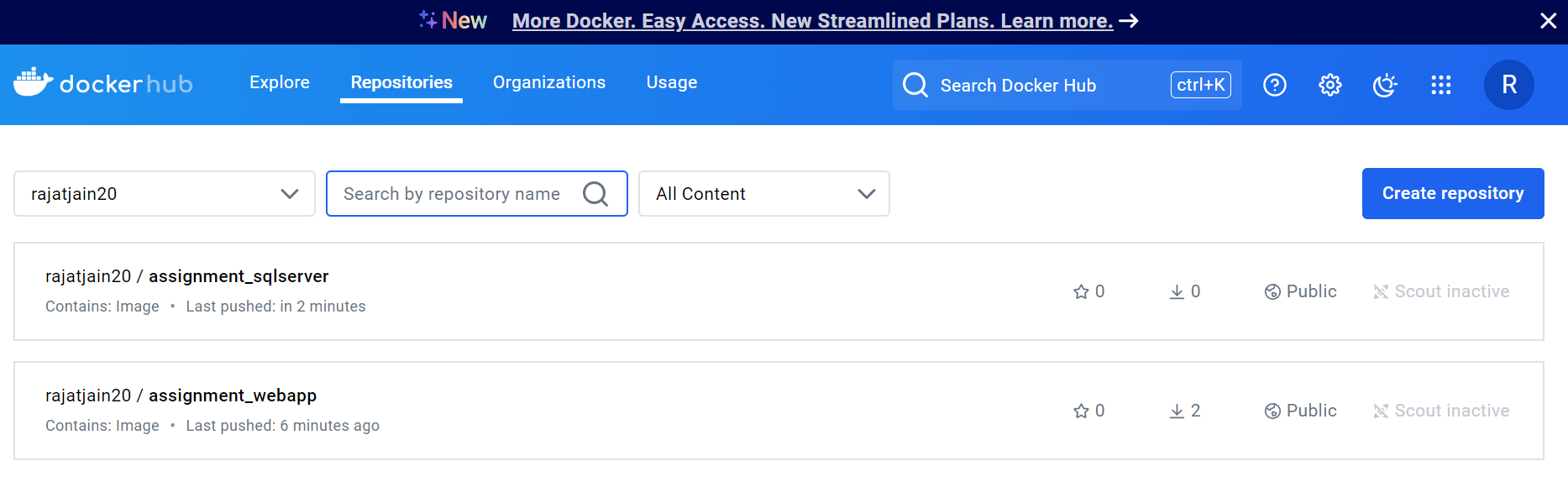
* docker tag assignment\_webapp:v0 rajatjain20/assignment\_webapp:v0
* docker push rajatjain20/assignment\_webapp:v0



Pushing sqlserver image:

* docker tag assignment\_sqlserver:v0 rajatjain20/assignment\_sqlserver:v0
* docker push rajatjain20/assignment\_sqlserver:v0

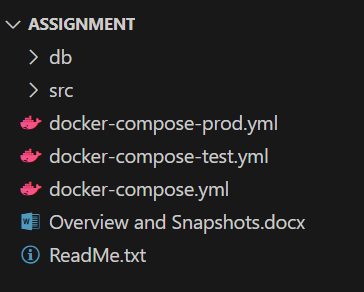




**Bonus Ques** –

To implement environment variable management in your Docker Compose file to handle different environments (development, testing, production), I have created three docker compose yml files –

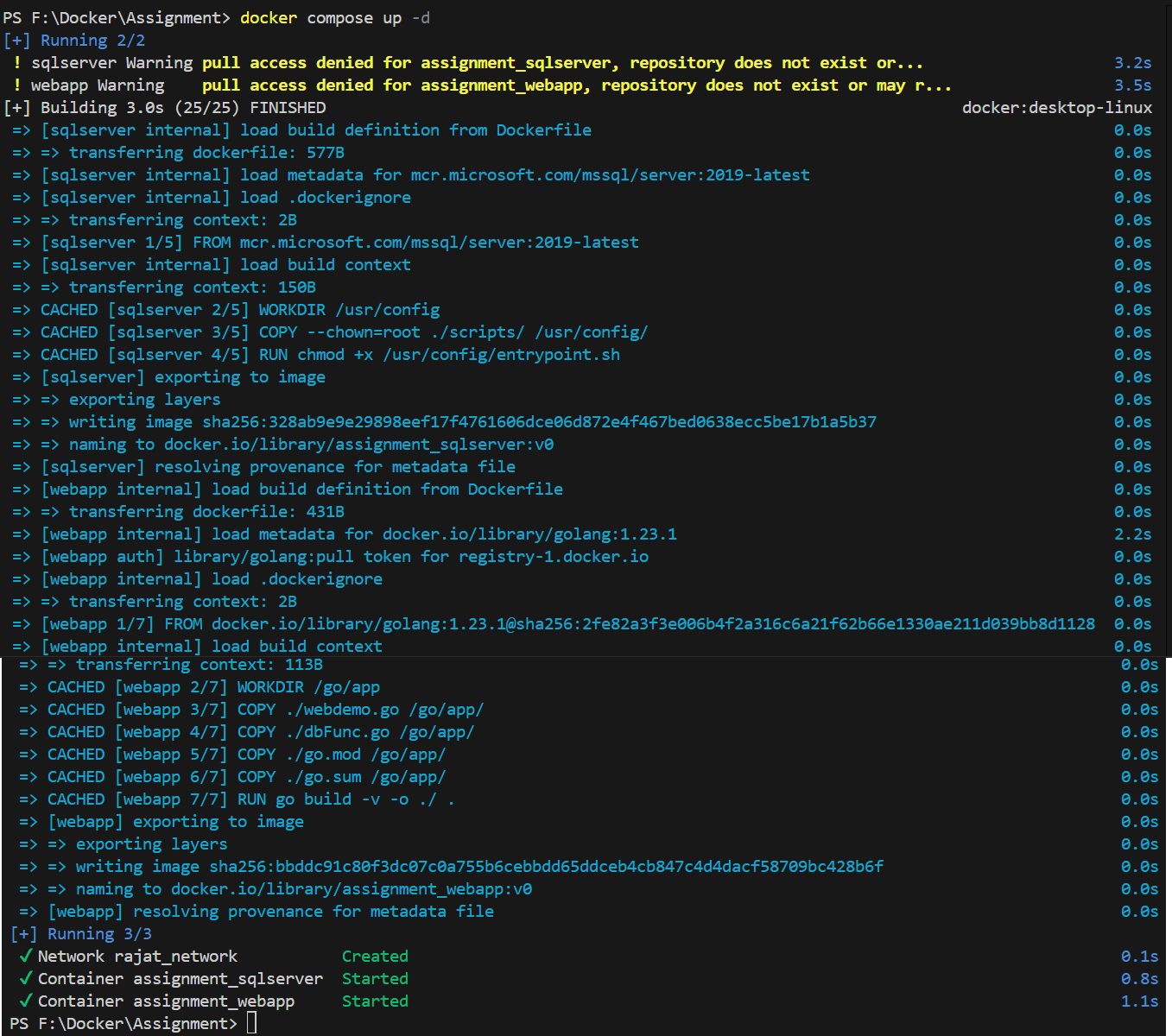
* docker-compose.yml (default – DEVELOPMENT)
* docker-compose-test.yml (TESTING)
* docker-compose-prod.yml (PRODUCTION)



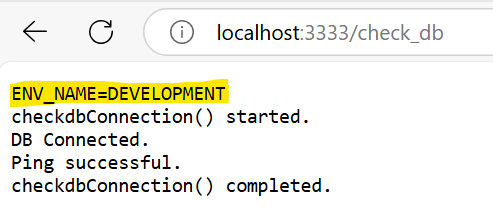
Commands –

1. Development:

> docker compose up -d

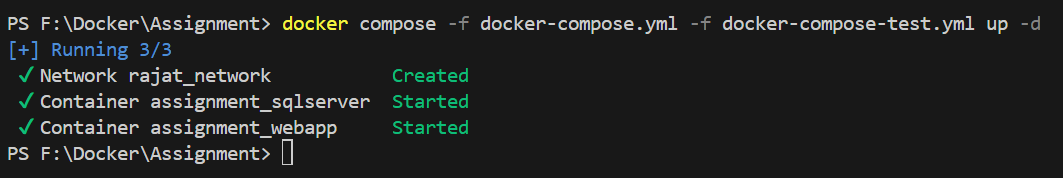


Now when you run webapp API:

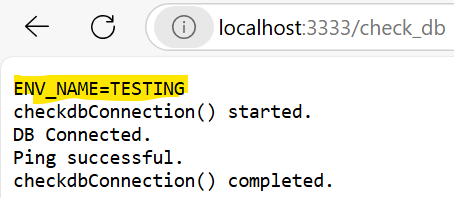


1. Testing: (first execute command: > docker compose down)

> docker compose -f docker-compose.yml -f docker-compose-test.yml up -d

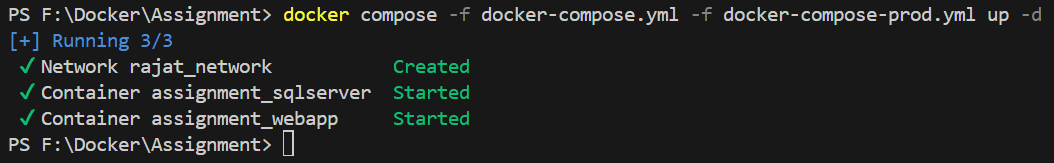


Now when you run webapp API:



1. Production: (first execute command: > docker compose down)

> docker compose -f docker-compose.yml -f docker-compose-prod.yml up -d



Now when you run webapp API:

