**Project Overview**

I have built a multiservice application, where there will be three services -

1. **DataInputApp:**

This is a Golang based web application, through which, user can add text data (.text file) in a shared directory (using webapi) from where the webapp will pick that data.

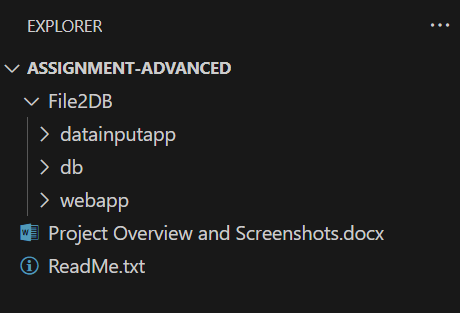
1. **Webapp:**

This is a Golang based web application, which will access the data stored in the shared directory from "DataInputApp" and store that data into Database (using webapi). User will be able to view that data stored in DB through a webapi.

1. **Database:**

This is a MSSQL database. In this, we have 'FILE2DB' database and a table named "FILE2DBDATA". Data will be stored in this table only by webapp.

**Project:**



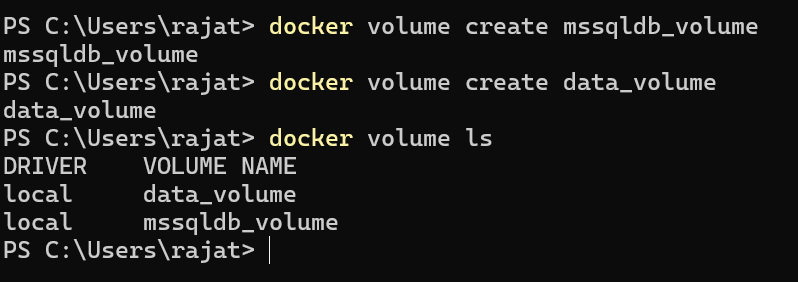
**Setting up volumes, networks, images and container** -

1. Create Volume for database persistence:

> docker volume create mssqldb\_volume

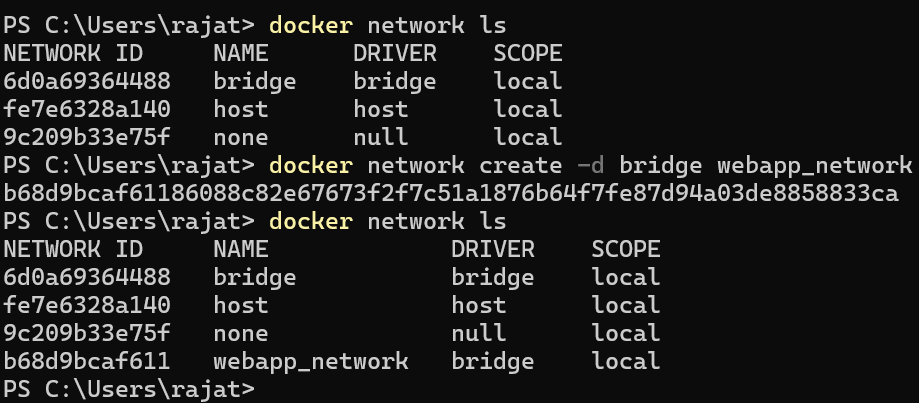
1. Create Volume for sharing data between ‘dataInputapp’ and ‘webapp’:

> docker volume create data\_volume



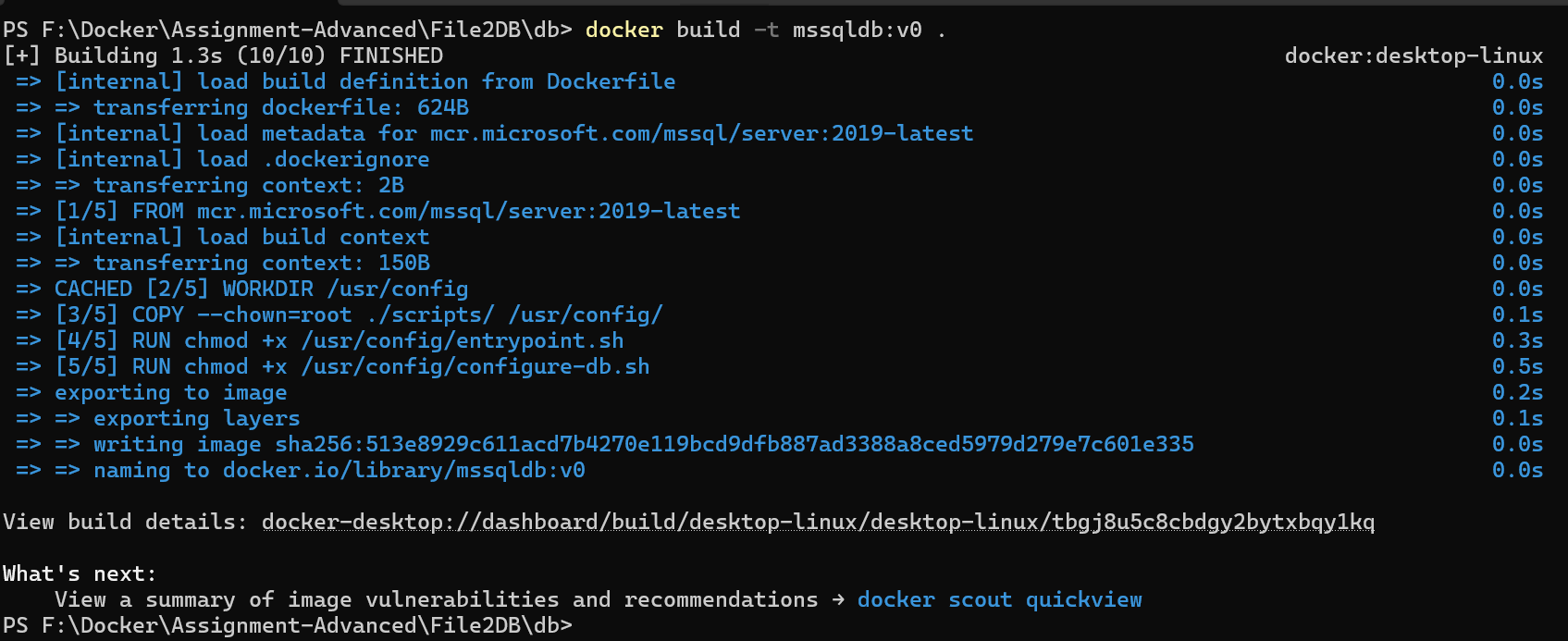
1. Create Network on which ‘webapp’ container can communicate with ‘mssqldb’ container:

> docker network create -d bridge webapp\_network

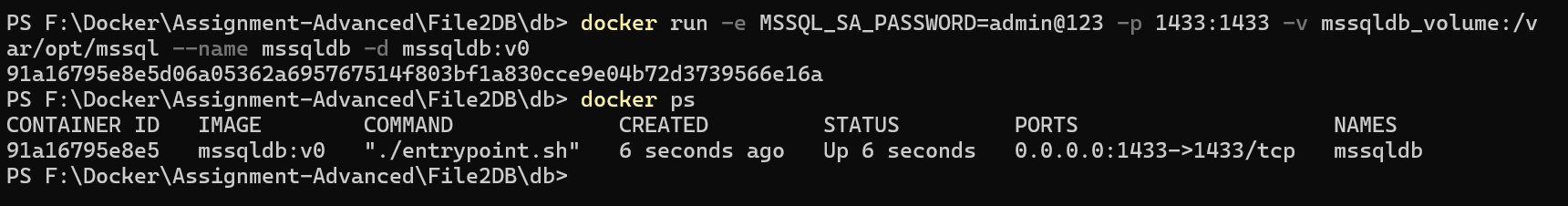


1. Create images and run containers –
2. db image and container : (go to db directory in project on terminal)

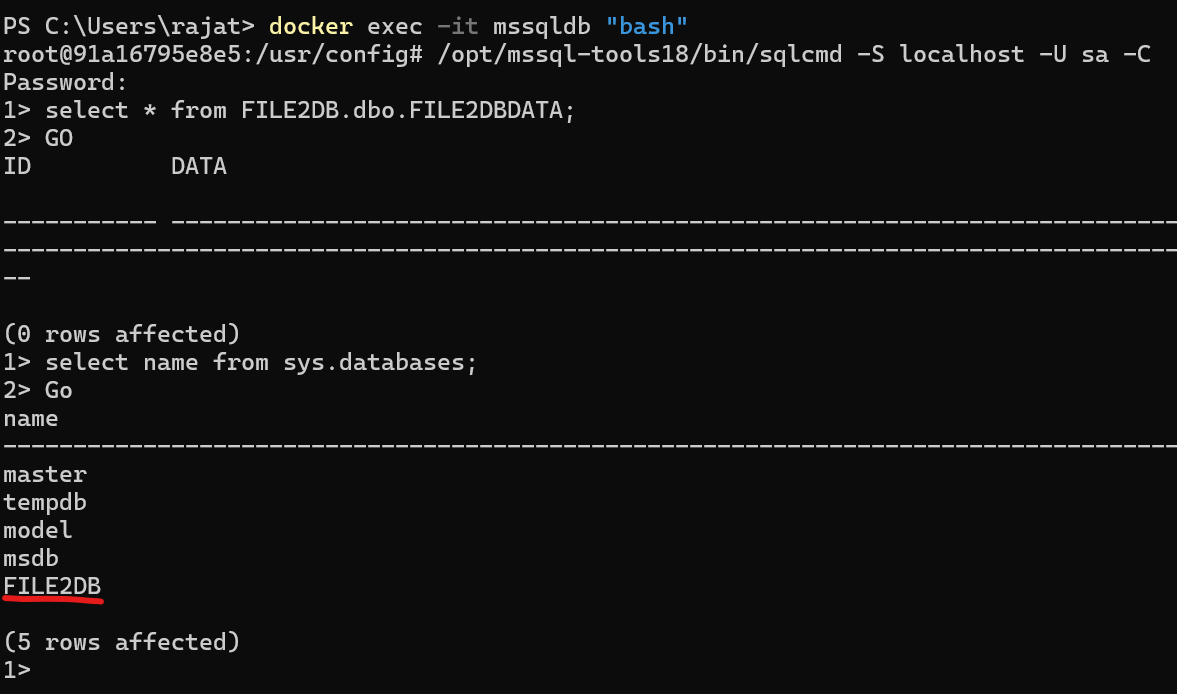
> docker build -t mssqldb:v0 .



> docker run -e MSSQL\_SA\_PASSWORD=admin@123 -p 1433:1433 -v mssqldb\_volume:/var/opt/mssql --name mssqldb -d mssqldb:v0



Check if mssqldb container is running fine –



Connect DB container to network for communication with webapp:

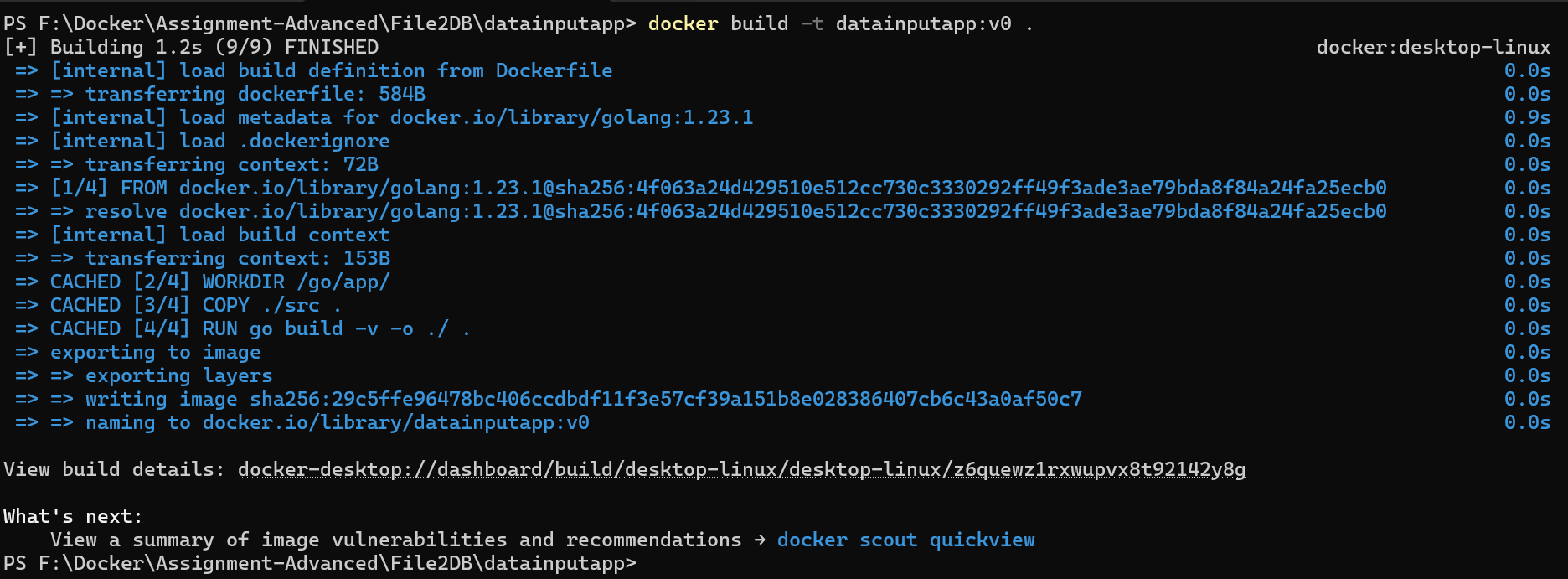
> docker network connect --alias mssqldb webapp\_network mssqldb

> docker inspect mssqldb

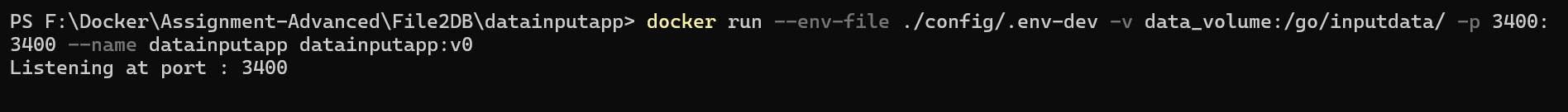


1. datainputapp image and container: (go to datainputapp directory in project on terminal)

> docker build -t datainputapp:v0 .

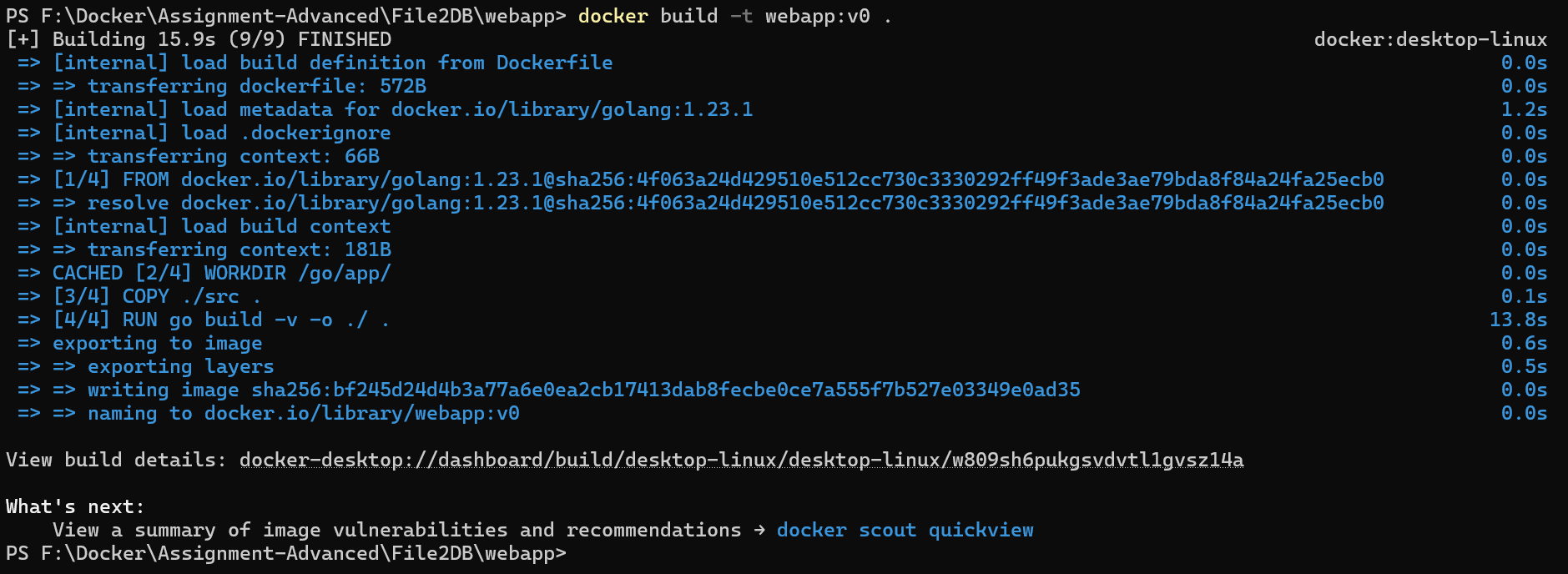


> docker run --env-file ./config/.env-dev -v data\_volume:/go/inputdata/ -p 3400:3400 --name datainputapp datainputapp:v0

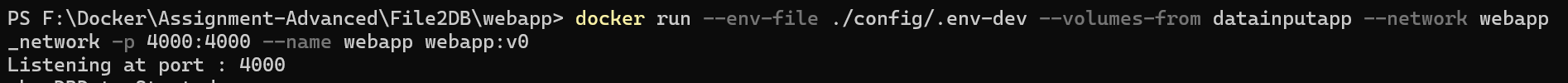


1. webapp image and container: (go to webapp directory in project on terminal)

> docker build -t webapp:v0 .



> docker run --env-file ./config/.env-dev --volumes-from datainputapp --network webapp\_network -p 4000:4000 --name webapp webapp:v0

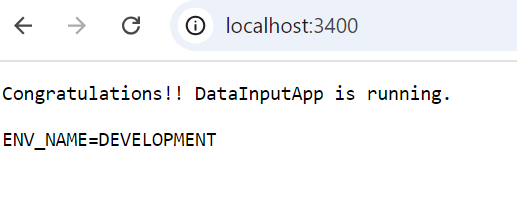


So now all three containers are running and they are connected through network and required volumes are mounted.

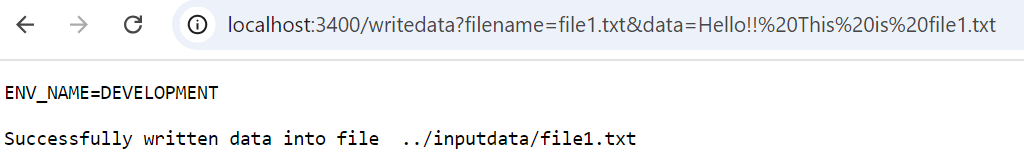
Let's execute and test the application.

Below are the webapi:

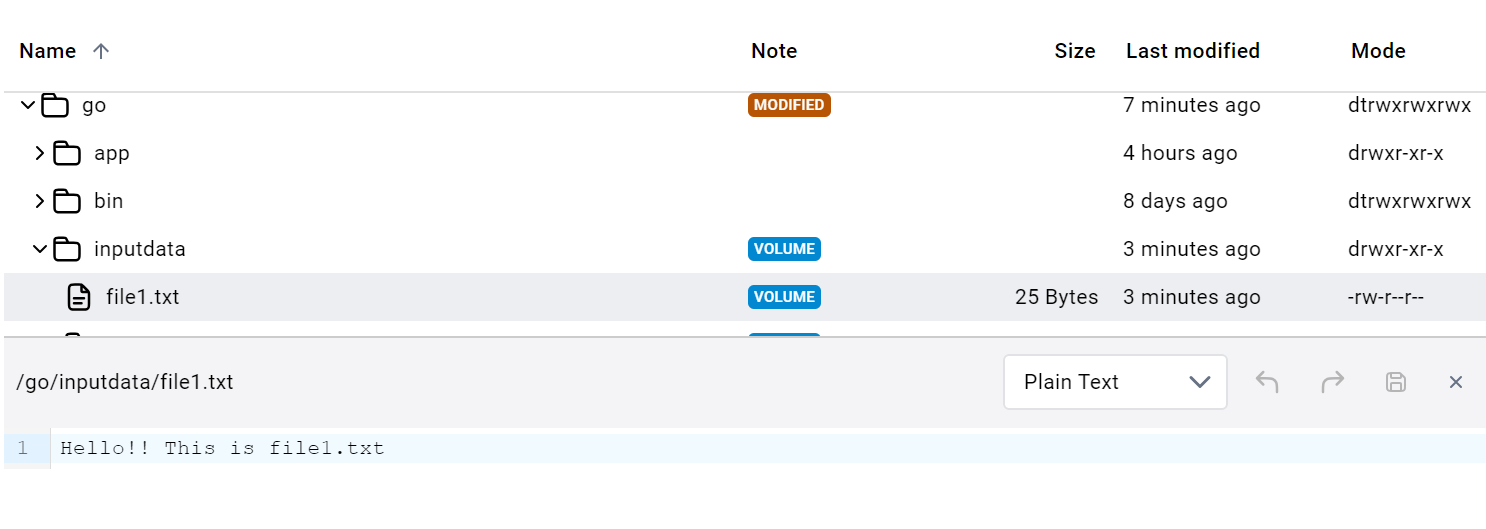
1. datainputapp: (running on port-3400)
2. <http://localhost:3400/>



1. http://localhost:3400/writedata?filename=file1.txt&data=Hello!! This is file1.txt

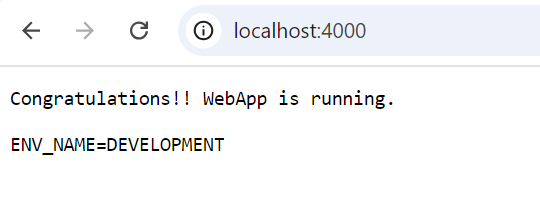


We can check the file in specified directory in docker desktop –



As this volume and same directory are being used by webapp as well. We can access this file and can process it through webapp and insert data into database.

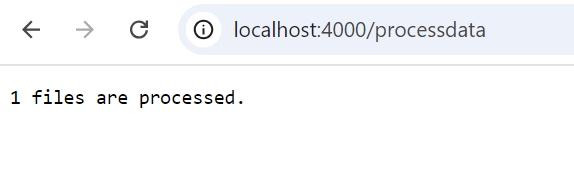
1. webapp: (running on port 4000)
2. <http://localhost:4000/>



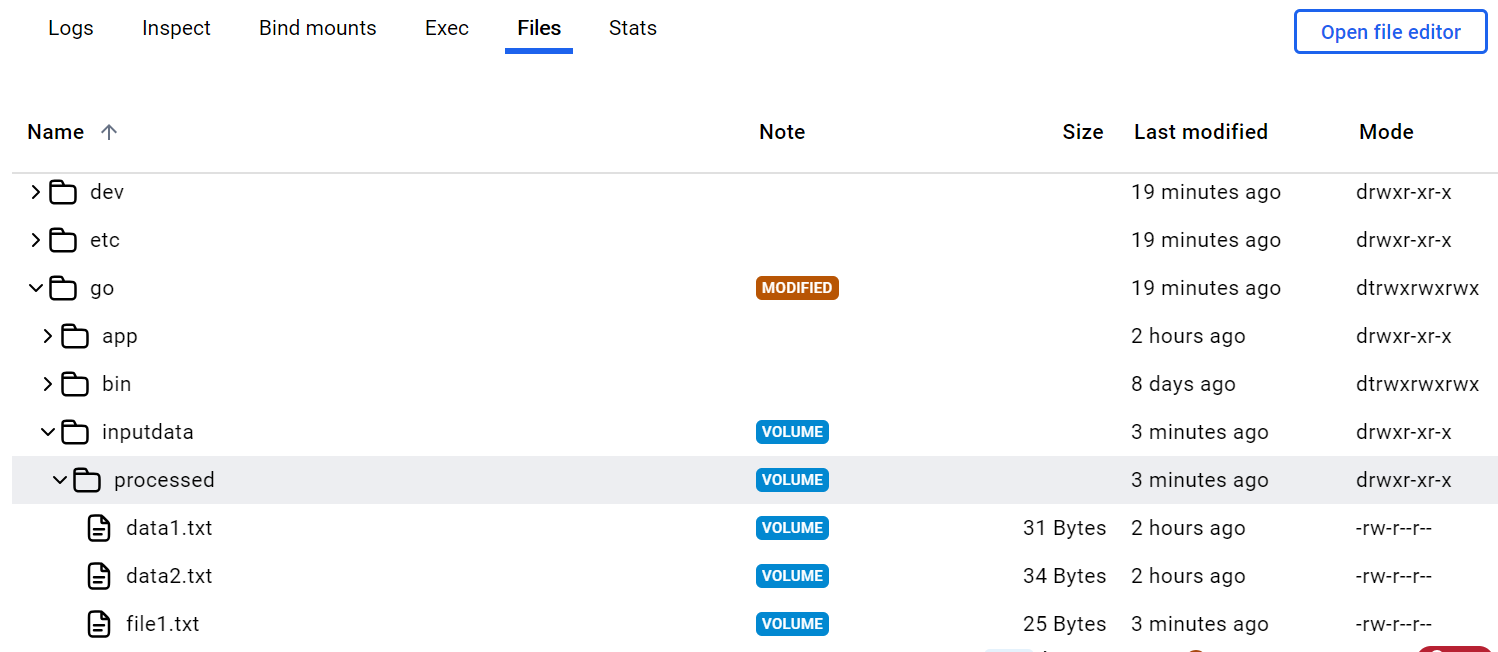
1. <http://localhost:4000/showdbdata>

As there is nothing in database table. So just an empty json is returned.

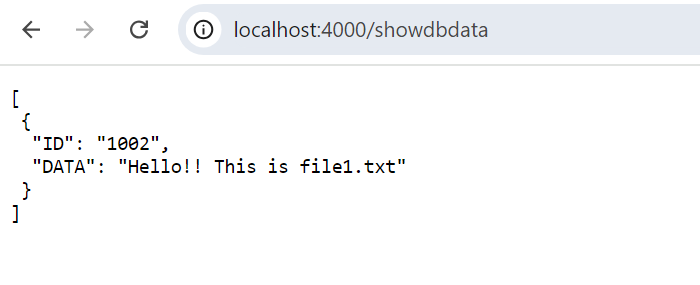
1. <http://localhost:4000/processdata>



This API picks all the files in the specified directory (inputdata) and insert its data into database table and also move the files into ‘inpitdata/processed/’ directory.



1. <http://localhost:4000/showdbdata>



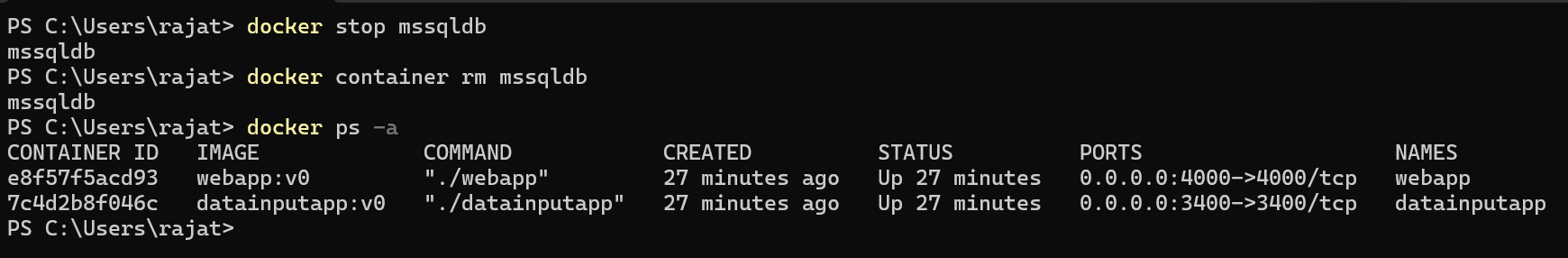
**To Check data persistence**:

We will stop ‘mssqldb’ container and remove it and then create and run it again.

> docker stop mssqldb

> docker container rm mssqldb

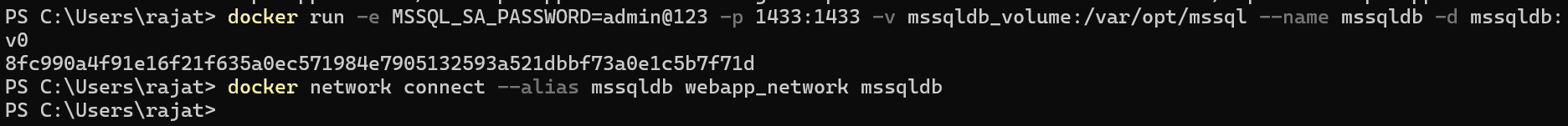
> docker ps -a



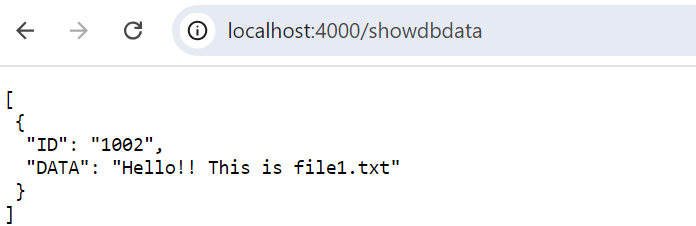
Now run the container again and also remember that we have to add network to it as we did before:

> docker run -e MSSQL\_SA\_PASSWORD=admin@123 -p 1433:1433 -v mssqldb\_volume:/var/opt/mssql --name mssqldb -d mssqldb:v0

> docker network connect --alias mssqldb webapp\_network mssqldb

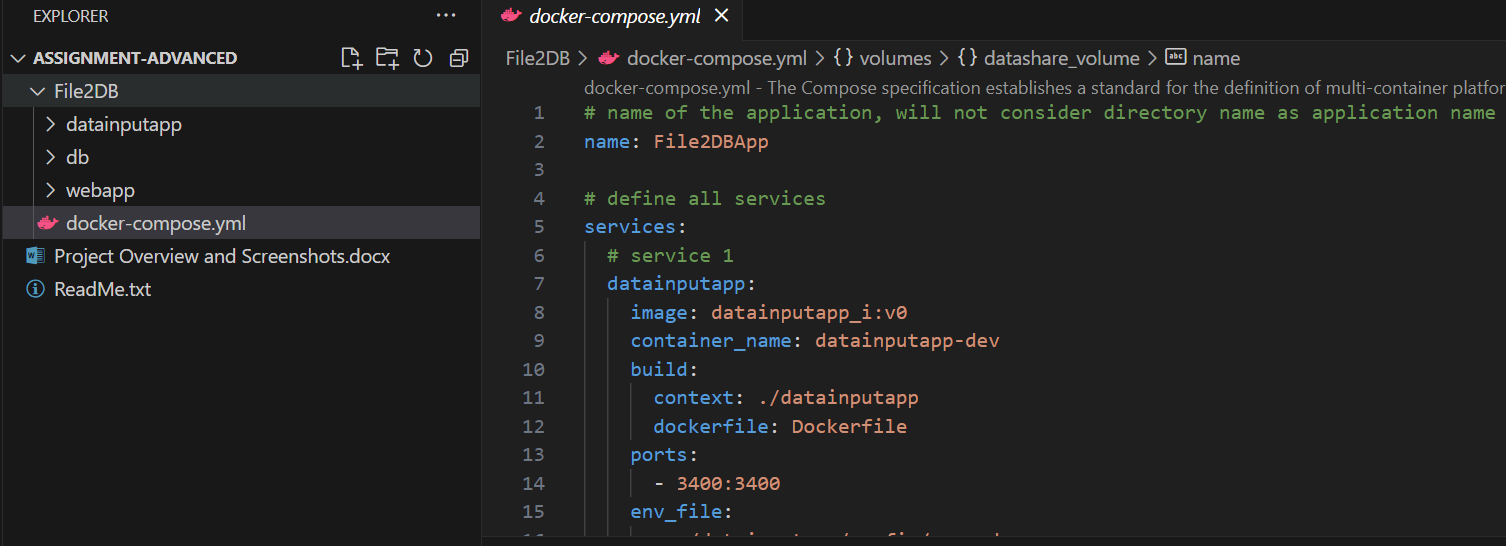


Now if we run below url, the data should persist:



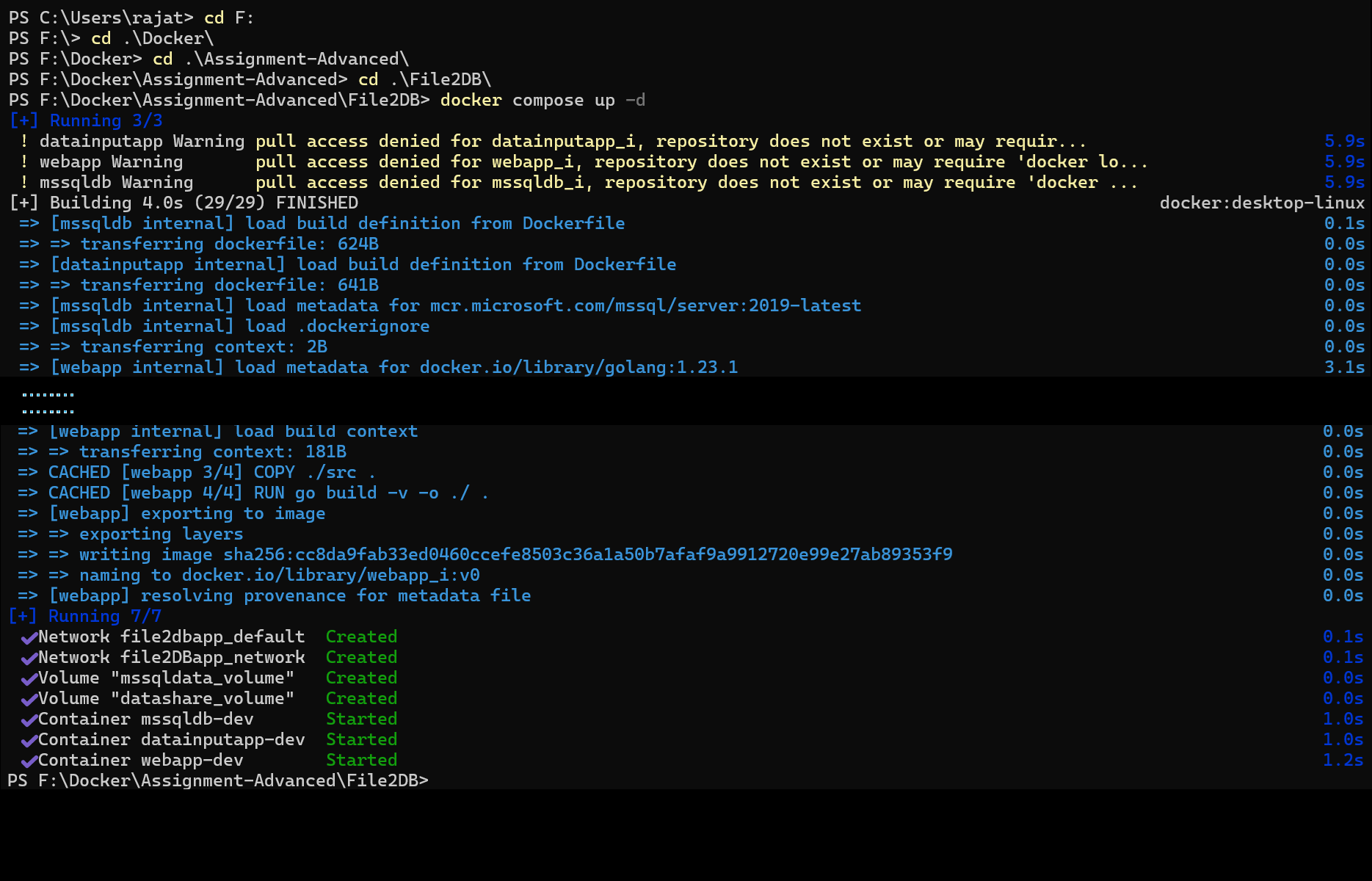
**Docker compose:**

There is one ‘docker-compose.yml’ file has been created in the project.



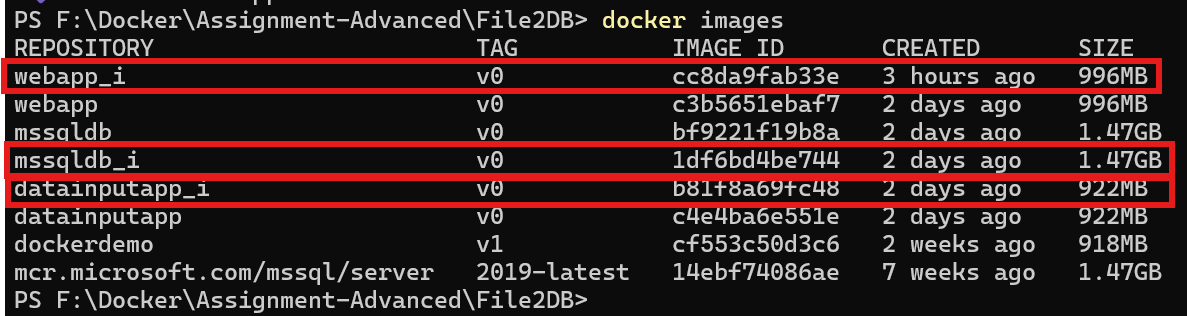
On the terminal, go to File2DB directory and run below command:

> docker compose up -d



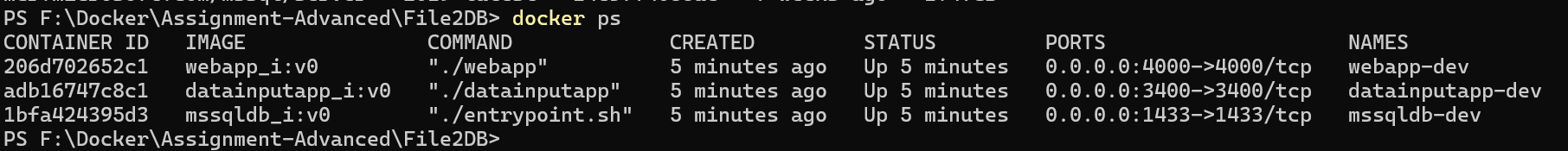
Lets look at the images:

> docker images

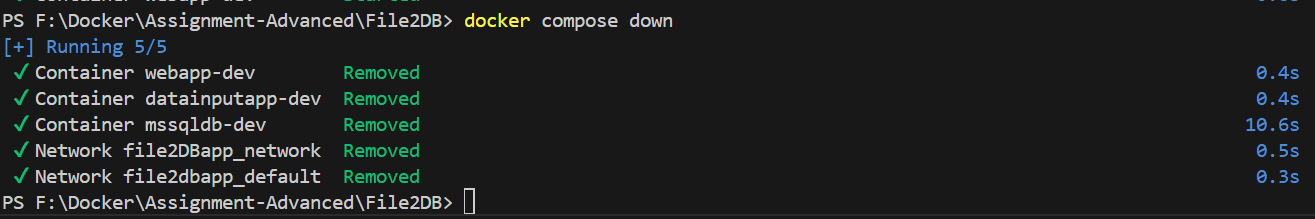


Container for this compose file:

> docker ps



> docker compose down



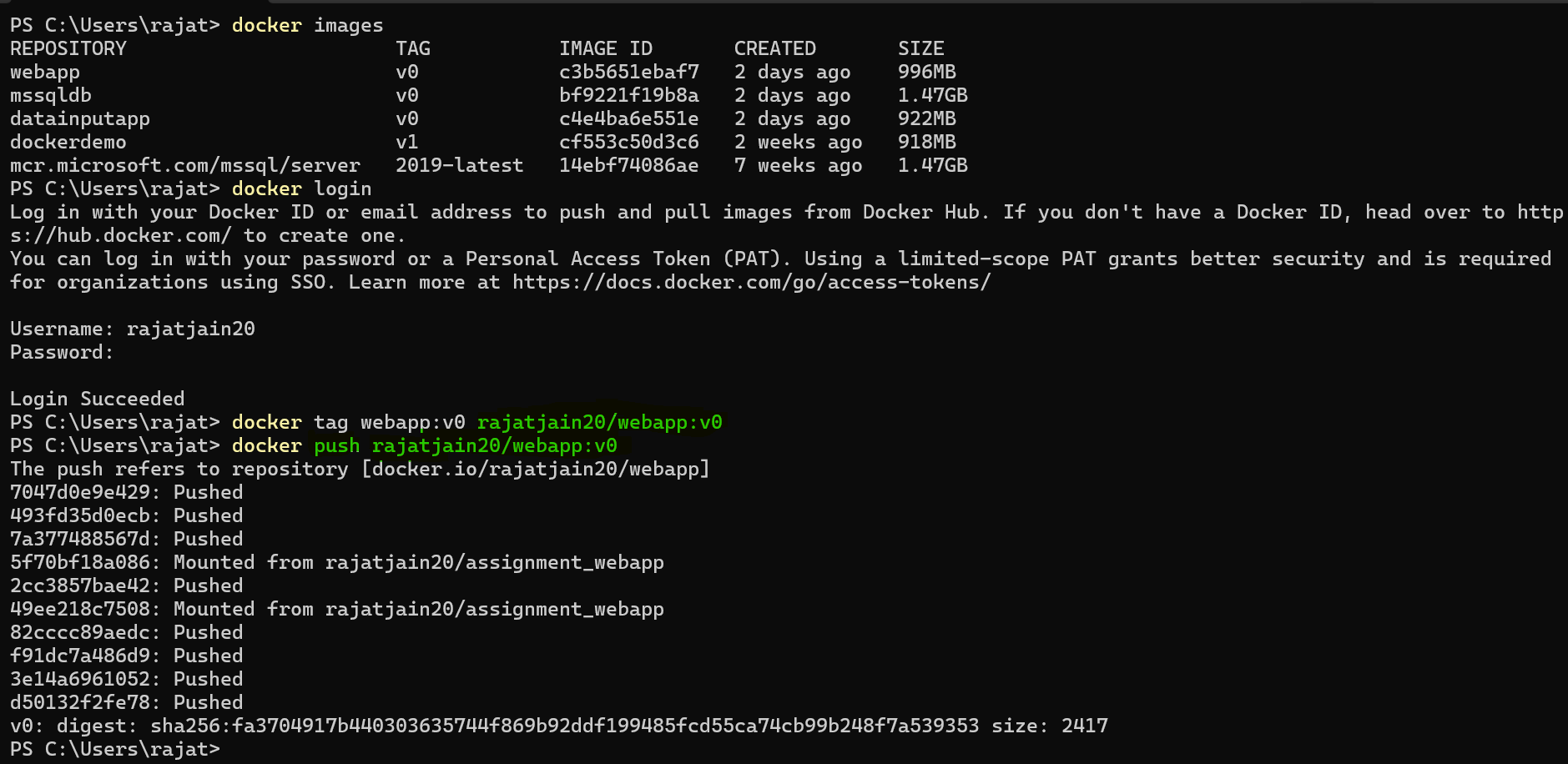
**Push images to docker registry (Docker Hub):**

Commands:

> docker login

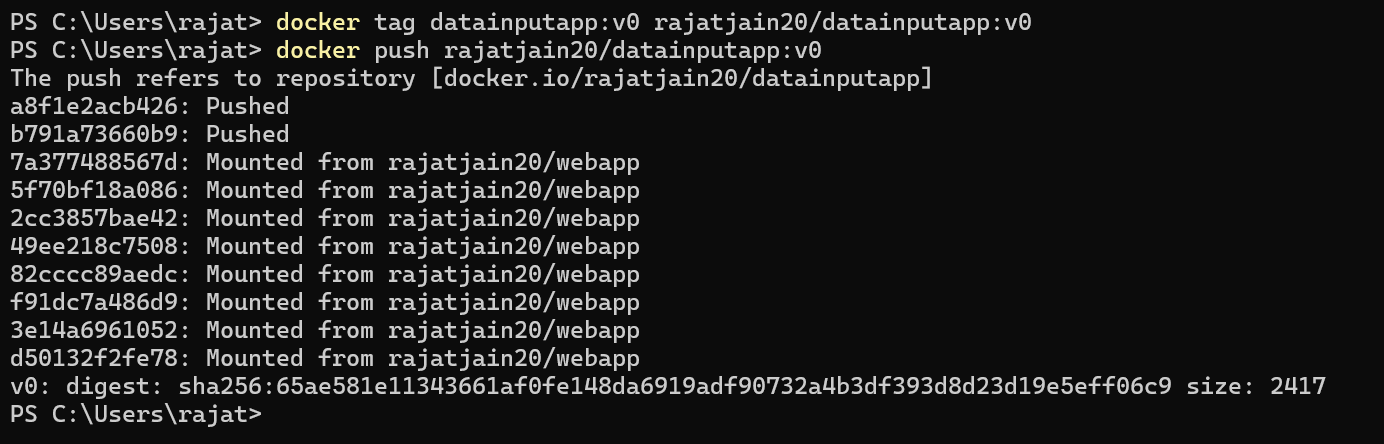
> docker tag webapp:v0 rajatjain20/webapp:v0

> docker push rajatjain20/webapp:v0



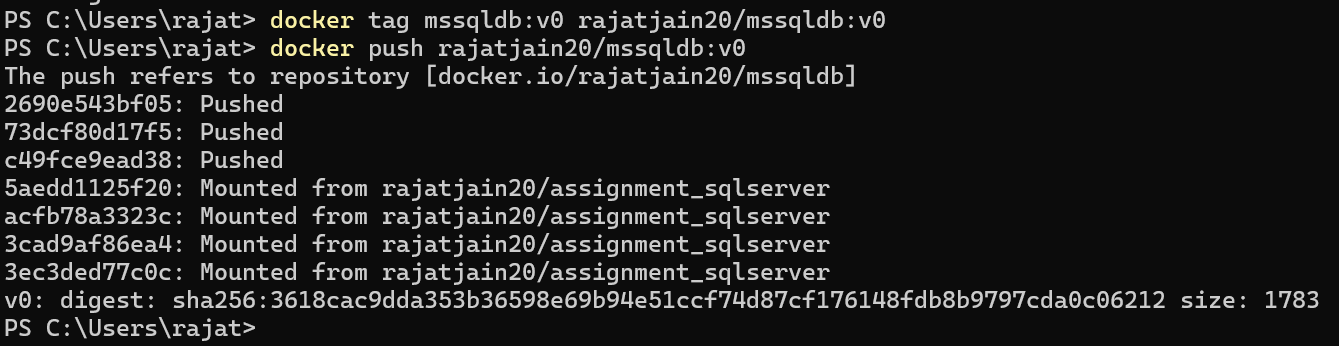
> docker tag datainputapp:v0 rajatjain20/datainputapp:v0

> docker push rajatjain20/datainputapp:v0

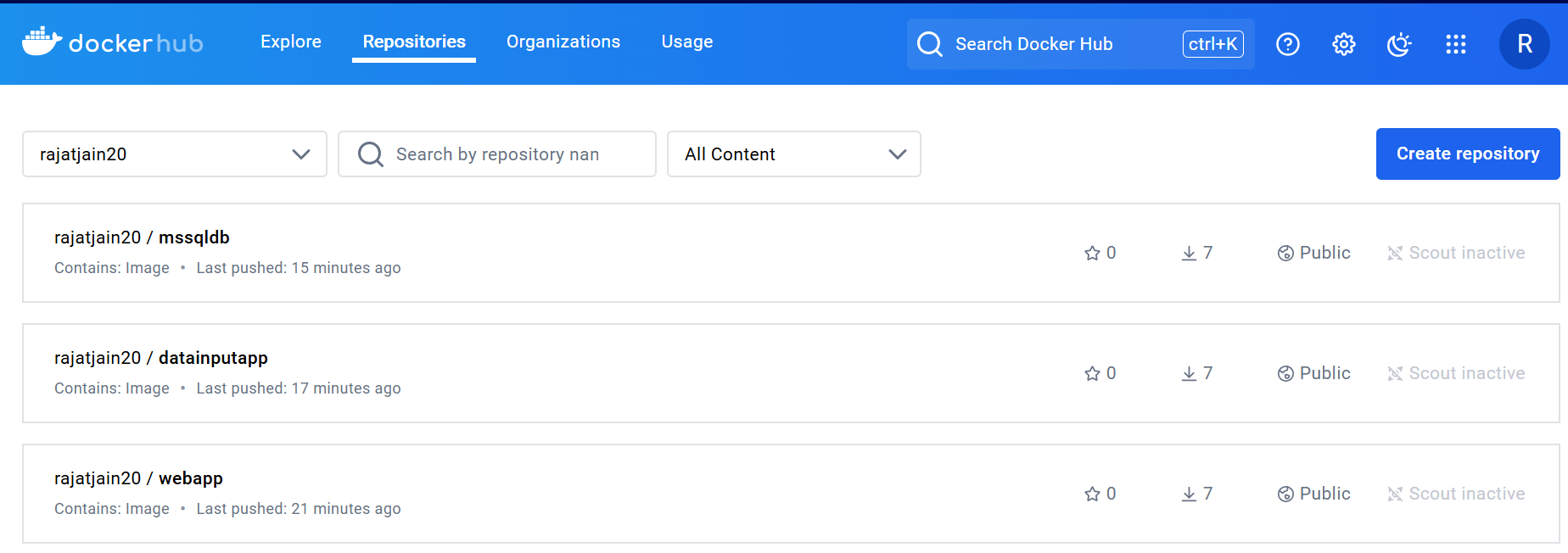


> docker tag mssqldb:v0 rajatjain20/mssqldb:v0

> docker push rajatjain20/mssqldb:v0



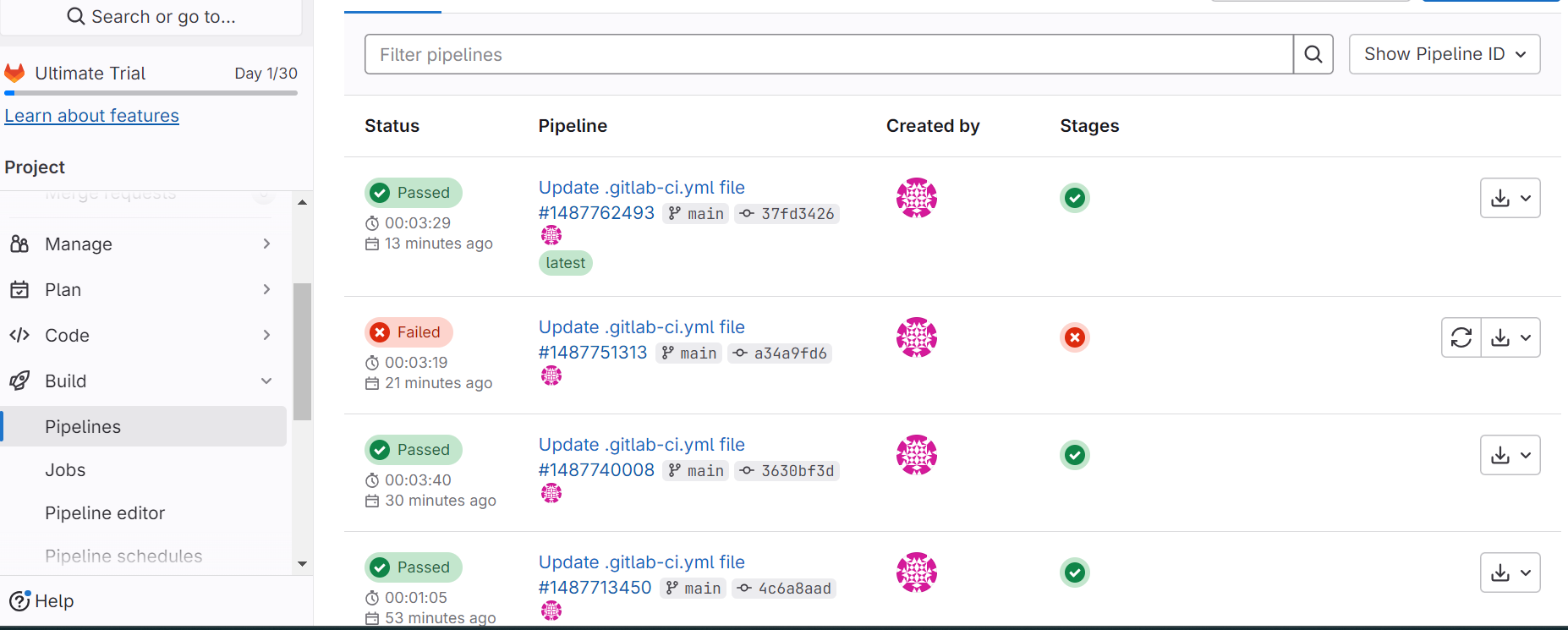
We can check, if images are present in docker registry:



**CI/CD pipeline:**

I have used GitLab for CI/CD of this application. I have created a file ".gitlab-ci.yml", which builds the images of the applications and saves them into GitLab registry.

Pipelines:



Images pushed into gitlab’s Container Registry:

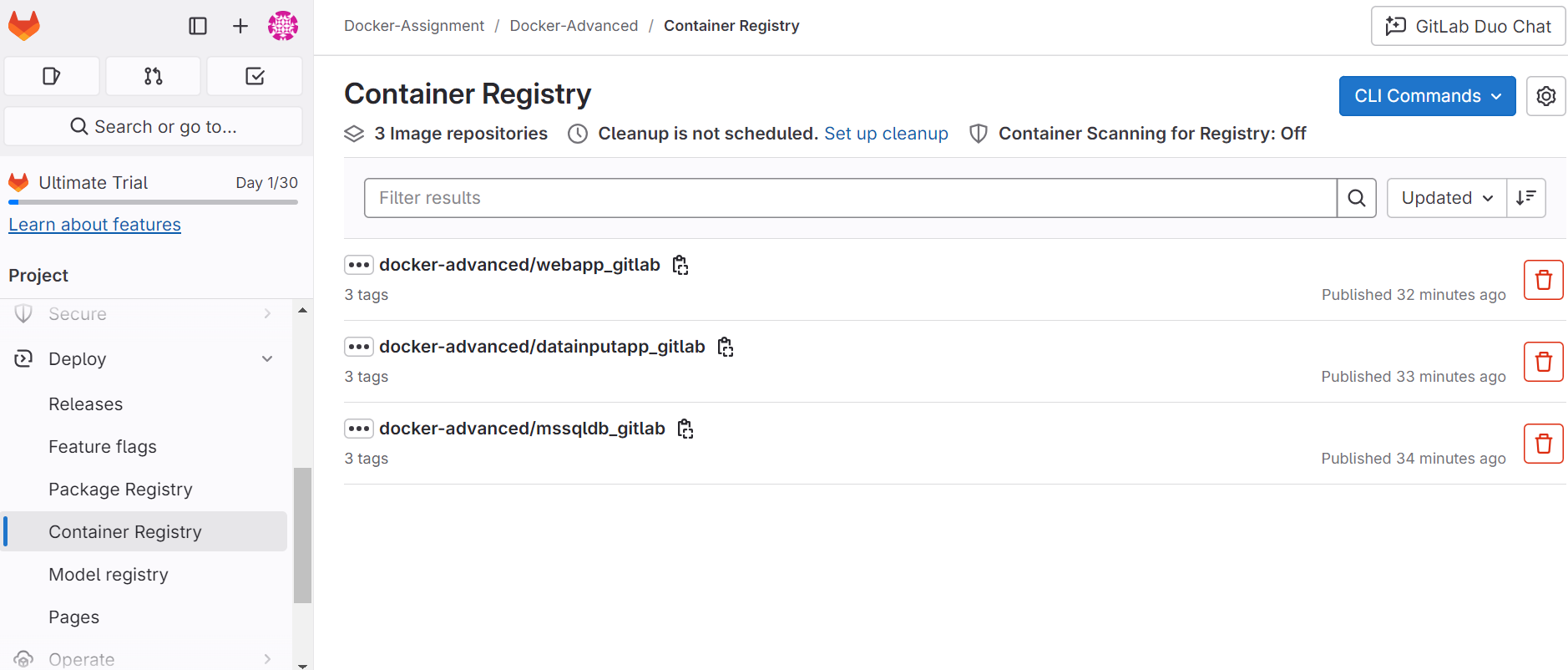


Image tags can be seen by clicking on any of the image:

