Ollscoil Teicneolaíochta an Oirdheiscirt

South East Technological University

Department of Computing

Bachelor of Science (Hons) in Software Development

Cloud Data Centres - Y4

Lab 3

Student name: Sebastian Firsaev

Student Number: C00263348

Lecturer: Dr Lei Shi

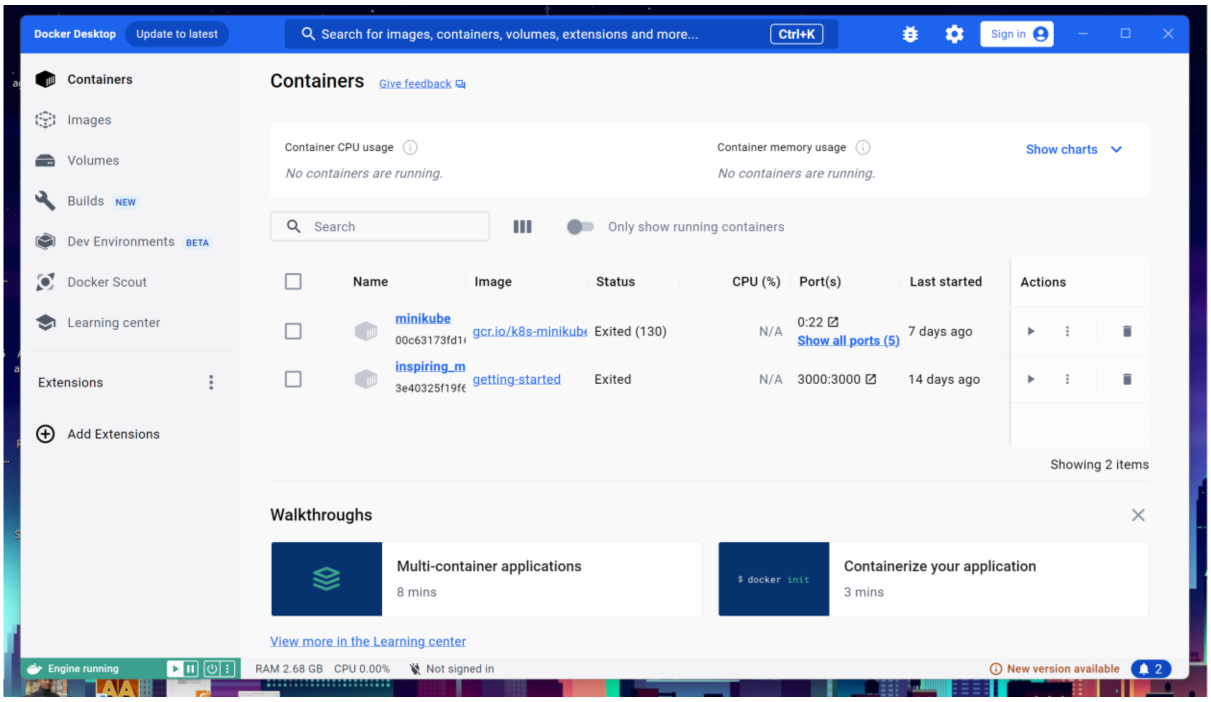
Table of Contents

Part 1: Overview of the get started guide

Introduction

The following section provides an overview of the "Get Started Guide" for Docker, detailing a series of practical steps aimed at facilitating the initial exploration and utilization of Docker technology. This guide systematically walks through fundamental procedures, including building and running images as containers, sharing images via Docker Hub, deploying Docker applications with multiple containers featuring a database, and executing applications through Docker Compose. To comprehend these procedures effectively, it's essential to grasp the concept of containers and images within the Docker ecosystem. This introductory segment elucidates the core functionalities and characteristics of containers and images, offering insights into their pivotal roles in modern software development and deployment practices.

Docker is installed:



Containers

Container CPU usage
No containers are running.

Container memory usage
No containers are running.

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
minikube 00c63173fd1	gcr.io/k8s-minikube	Exited (130)	N/A	0:22 Show all ports (5)	7 days ago	
inspiring_m 3e40325f19f6	getting-started	Exited	N/A	3000:3000	14 days ago	

Showing 2 items

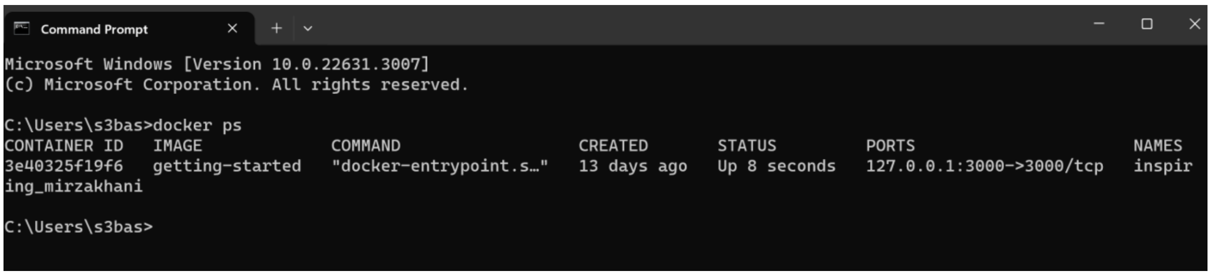
RAM 2.68 GB CPU 0.00%

Part 2: Containerize an application.

Introduction:

In this section of the report, I will outline the process of containerizing a simple todo list manager application developed in Node.js. This involves encapsulating the application within Docker containers, which enables portability and efficient deployment across different environments. Before proceeding, it's essential to ensure that all prerequisites are met, including the installation of Docker Desktop, a Git client, and an appropriate

Integrated Development Environment (IDE) or text editor. Once these requirements are fulfilled, the next steps involve obtaining the application source code by cloning the "getting-started-app" repository. Following this, I will create a Dockerfile in the application directory, which serves as a blueprint for building the container image. With the Dockerfile in place, I will utilize Docker commands to initiate the image construction process, incorporating dependencies and configurations specified in the Dockerfile. Subsequently, I will initiate a container running the application using the docker run command, ensuring proper port mapping for accessibility. Upon completion of these steps, the todo list manager application will be successfully containerized, ready for deployment and execution within the Docker environment.

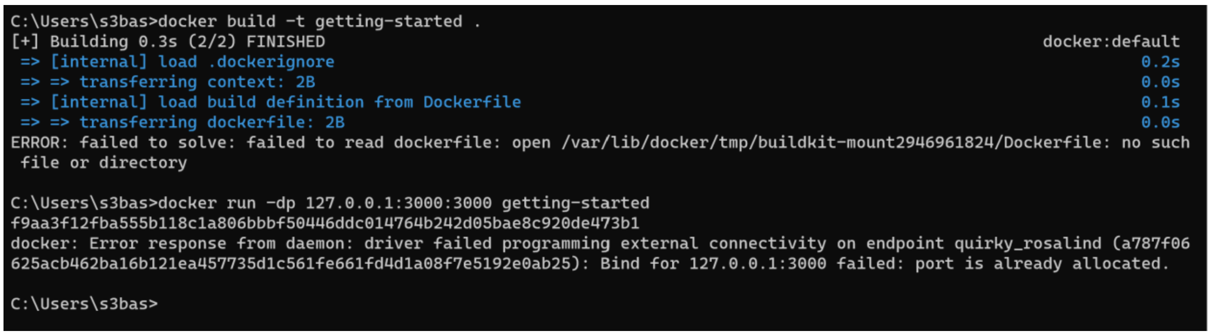


```
Microsoft Windows [Version 10.0.22631.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\s3bas>docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED       STATUS         PORTS                      NAMES
3e40325f19f6   getting-started   "docker-entrypoint.s…"   13 days ago   Up 8 seconds   127.0.0.1:3000->3000/tcp   inspir
ing_mirzakhani

C:\Users\s3bas>
```

Part 3: Update the application.

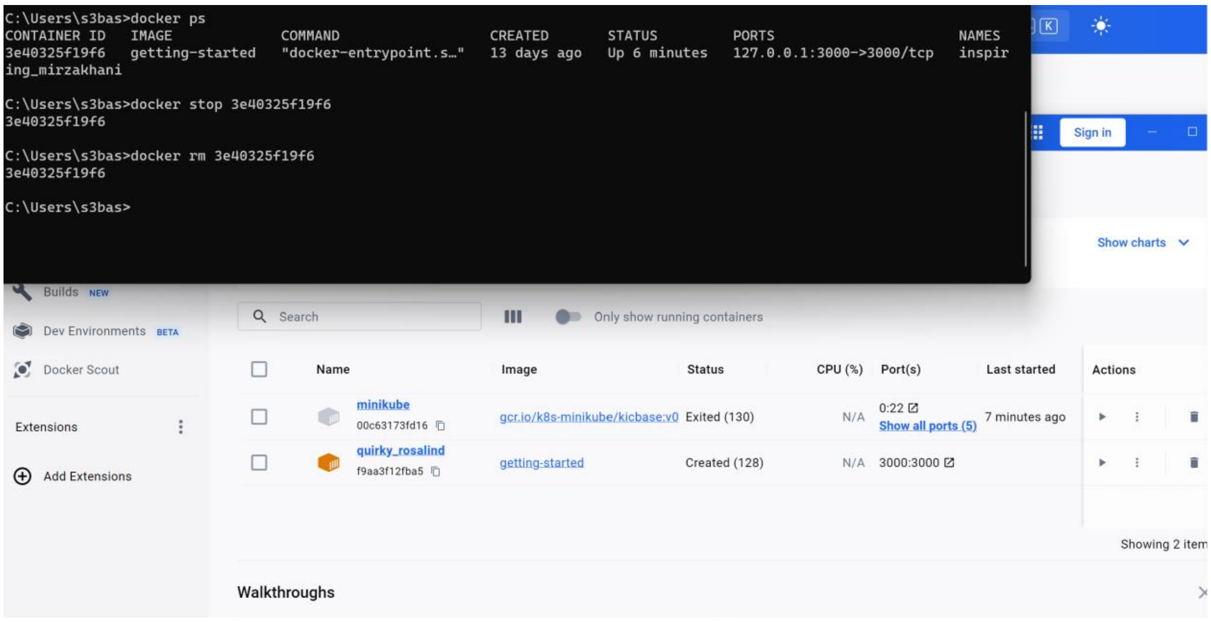
Introduction:

In this section, I will detail the process of updating the todo list application and its corresponding container image. The objective is to modify the application's user interface by changing the "empty text" message from "No items yet! Add one above!" to "You have no todo items yet! Add one above!". To implement this change, I will navigate to the src/static/js/app.js file and update the relevant line of code accordingly. Following the modification, I will rebuild the container image using the updated source code by executing the docker build command. Subsequently, I will initiate a new container using the docker run command, specifying the updated image. However, it's crucial to address any errors that may arise due to port conflicts, as attempting to start a new container while the previous one is running can lead to errors. To resolve this issue, I will stop and remove the old container using the docker stop and docker rm commands, respectively. Once the old container is removed, I can proceed to start the updated application container without encountering port allocation conflicts. Finally, I will verify the changes by refreshing the browser and confirming the updated help text on http://localhost:3000. This section concludes with a summary of the key learnings, including the process of updating and rebuilding a container, as well as managing containers through stopping and removal procedures.



```
C:\Users\s3bas>docker build -t getting-started .
[+] Building 0.3s (2/2) FINISHED                                                              docker:default
 => [internal] load .dockerignore                                                                       0.2s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load build definition from Dockerfile                                                    0.1s
 => => transferring dockerfile: 2B                                                                      0.0s
ERROR: failed to solve: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount2946961824/Dockerfile: no such
 file or directory

C:\Users\s3bas>docker run -dp 127.0.0.1:3000:3000 getting-started
f9aa3f12fba555b118c1a806bbbf50446ddc014764b242d05bae8c920de473b1
docker: Error response from daemon: driver failed programming external connectivity on endpoint quirky_rosalind (a787f06
625acb462ba16b121ea457735d1c561fe661fd4d1a08f7e5192e0ab25): Bind for 127.0.0.1:3000 failed: port is already allocated.

C:\Users\s3bas>
```

Container stopped and removed

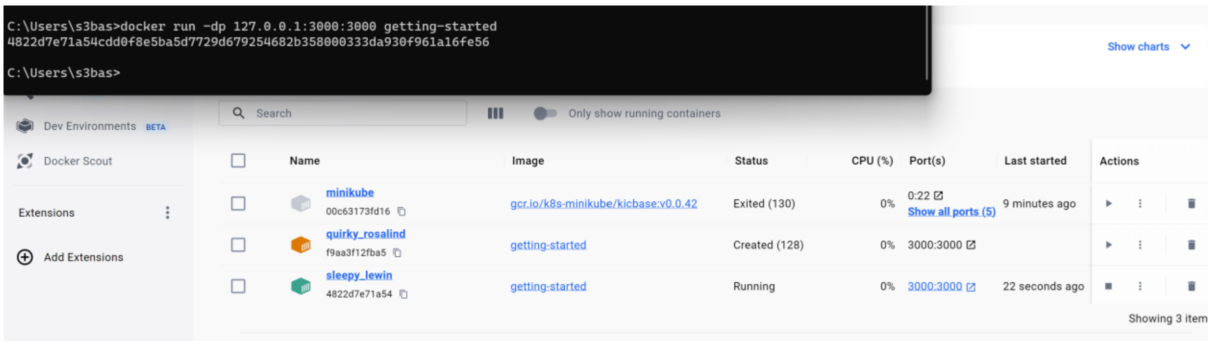


```
C:\Users\s3bas>docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED        STATUS         PORTS                      NAMES
3e40325f19f6   getting-started   "docker-entrypoint.s…"   13 days ago    Up 6 minutes   127.0.0.1:3000->3000/tcp   inspir
ing_mirzakhani

C:\Users\s3bas>docker stop 3e40325f19f6
3e40325f19f6

C:\Users\s3bas>docker rm 3e40325f19f6
3e40325f19f6

C:\Users\s3bas>
```

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
	minikube 00c63173fd16	gcr.io/k8s-minikube/kicbase:v0	Exited (130)	N/A	0:22 Show all ports (5)	7 minutes ago	
	quirky_rosalind f9aa3f12fba5	getting-started	Created (128)	N/A	3000:3000		

Showing 2 item

New container updated and started



```
C:\Users\s3bas>docker run -dp 127.0.0.1:3000:3000 getting-started
4822d7e71a54cdd0f8e5ba5d7729d679254682b358000333da930f961a16fe56

C:\Users\s3bas>
```

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
	minikube 00c63173fd16	gcr.io/k8s-minikube/kicbase:v0.0.42	Exited (130)	0%	0:22 Show all ports (5)	9 minutes ago	
	quirky_rosalind f9aa3f12fba5	getting-started	Created (128)	0%	3000:3000		
	sleepy_lewin 4822d7e71a54	getting-started	Running	0%	3000:3000	22 seconds ago	

Showing 3 item

Part 5: Persist the DB

Introduction:

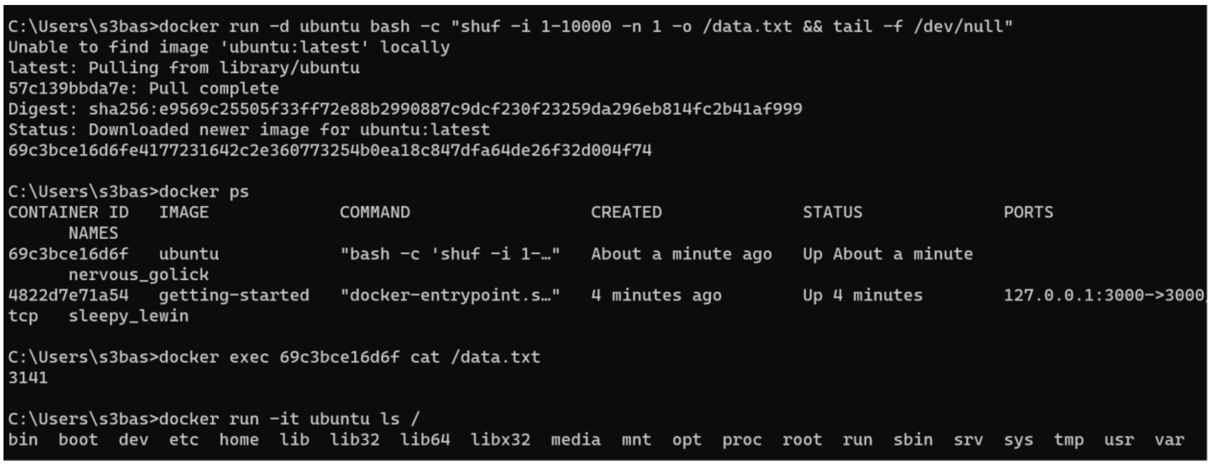
In this section, I will address the issue of the todo list being reset every time the container is launched. This occurs because each container uses its own isolated filesystem, derived from the image's layers, and changes made within a container are not reflected in other containers using the same image. To illustrate this concept, I will conduct an experiment by starting two Alpine containers, creating a file in one, and verifying its absence in the other. Subsequently, I will introduce the concept of volumes, which enable the persistence of data across container instances.

Volumes allow specific filesystem paths within a container to be connected back to the host machine, ensuring that changes made within the container are reflected on the host. I will demonstrate how to persist the todo list data by creating a volume and mounting it to the directory where the data is stored in the container. This volume, named "todo-db", will capture all files created at the specified path, ensuring the persistence of the todo list data across container restarts.

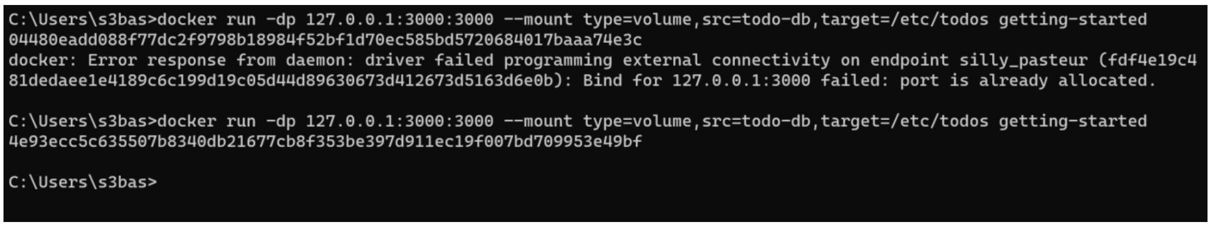
After creating the volume, I will start the todo app container, incorporating the volume mount. Upon launching the container, I will verify that the todo list data persists by adding items to the list, stopping and removing the container, and then starting a new container using the same steps. By examining the todo list, I will confirm that the items added previously are still present, demonstrating the successful persistence of data.

To delve further into the volume and understand its storage location on the disk, I will utilize the docker volume inspect command to retrieve information about the "todo-db" volume. This command will provide details such as the creation time, driver, mountpoint, and other relevant information.

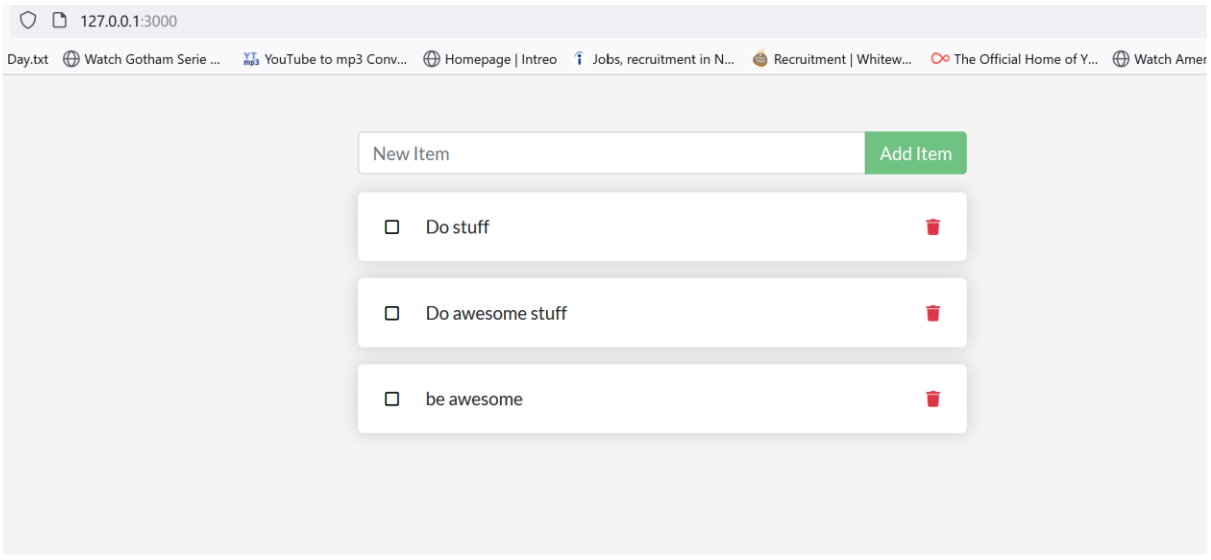
In summary, this section provides comprehensive guidance on persisting container data using volumes, ensuring the continuity of the todo list across container instances.



```
C:\Users\s3bas>docker run -d ubuntu bash -c "shuf -i 1-10000 -n 1 -o /data.txt && tail -f /dev/null"
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
57c139bbda7e: Pull complete
Digest: sha256:e9569c25505f33ff72e88b2990887c9dcf230f23259da296eb814fc2b41af999
Status: Downloaded newer image for ubuntu:latest
69c3bce16d6fe4177231642c2e360773254b0ea18c847dfa64de26f32d004f74

C:\Users\s3bas>docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED              STATUS              PORTS
      NAMES
69c3bce16d6f   ubuntu            "bash -c 'shuf -i 1-…"   About a minute ago   Up About a minute
      nervous_golick
4822d7e71a54   getting-started   "docker-entrypoint.s…"   4 minutes ago        Up 4 minutes        127.0.0.1:3000->3000
tcp   sleepy_lewin

C:\Users\s3bas>docker exec 69c3bce16d6f cat /data.txt
3141

C:\Users\s3bas>docker run -it ubuntu ls /
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```



```
C:\Users\s3bas>docker run -dp 127.0.0.1:3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
04480eadd088f77dc2f9798b18984f52bf1d70ec585bd5720684017baaa74e3c
docker: Error response from daemon: driver failed programming external connectivity on endpoint silly_pasteur (fdf4e19c4
81dedaee1e4189c6c199d19c05d44d89630673d412673d5163d6e0b): Bind for 127.0.0.1:3000 failed: port is already allocated.

C:\Users\s3bas>docker run -dp 127.0.0.1:3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
4e93ecc5c635507b8340db21677cb8f353be397d911ec19f007bd709953e49bf

C:\Users\s3bas>
```

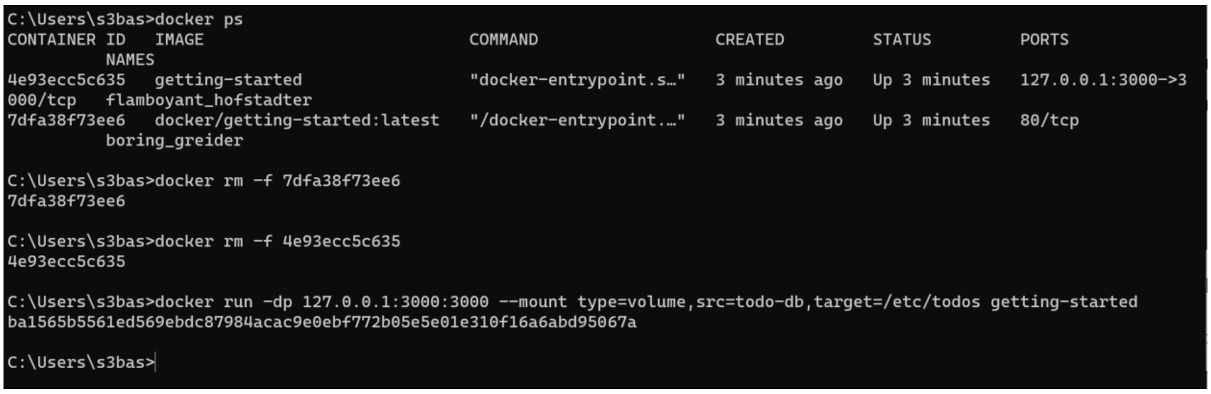


127.0.0.1:3000

Day.txt | Watch Gotham Serie ... | YouTube to mp3 Conv... | Homepage | Intreo | Jobs, recruitment in N... | Recruitment | Whitew... | The Official Home of Y... | Watch Amer

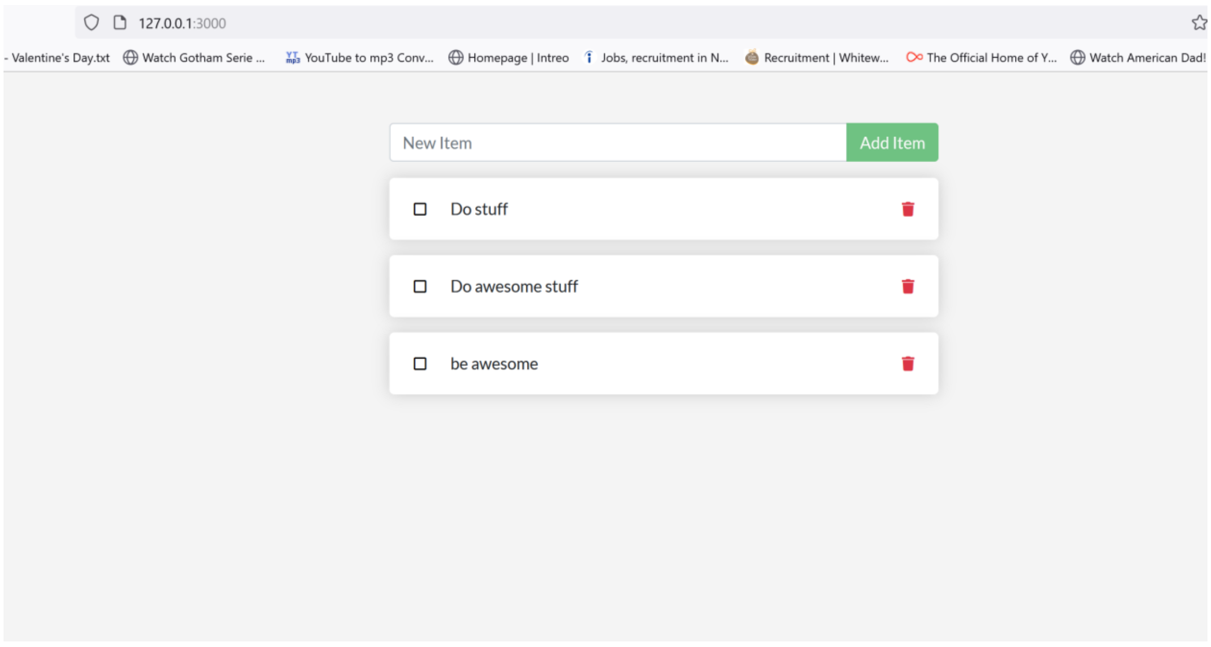
- [ ] Do stuff
- [ ] Do awesome stuff
- [ ] be awesome

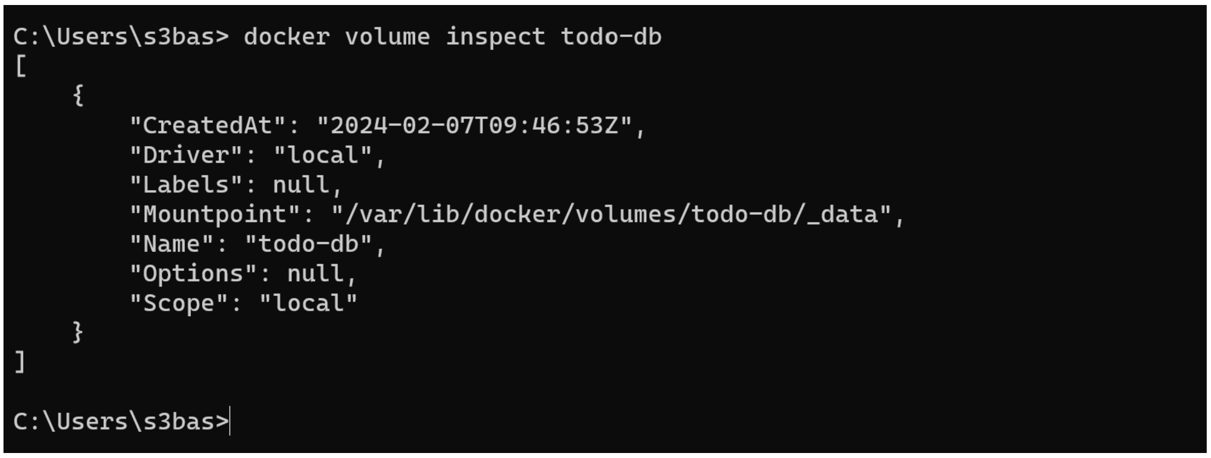
container removed, new container started:



```
C:\Users\s3bas>docker ps
CONTAINER ID   IMAGE                           COMMAND                  CREATED         STATUS         PORTS
          NAMES
4e93ecc5c635   getting-started                 "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   127.0.0.1:3000->3
000/tcp   flamboyant_hofstadter
7dfa38f73ee6   docker/getting-started:latest   "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   80/tcp
          boring_greider

C:\Users\s3bas>docker rm -f 7dfa38f73ee6
7dfa38f73ee6

C:\Users\s3bas>docker rm -f 4e93ecc5c635
4e93ecc5c635

C:\Users\s3bas>docker run -dp 127.0.0.1:3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
ba1565b5561ed569ebdc87984acac9e0ebf772b05e5e01e310f16a6abd95067a

C:\Users\s3bas>
```

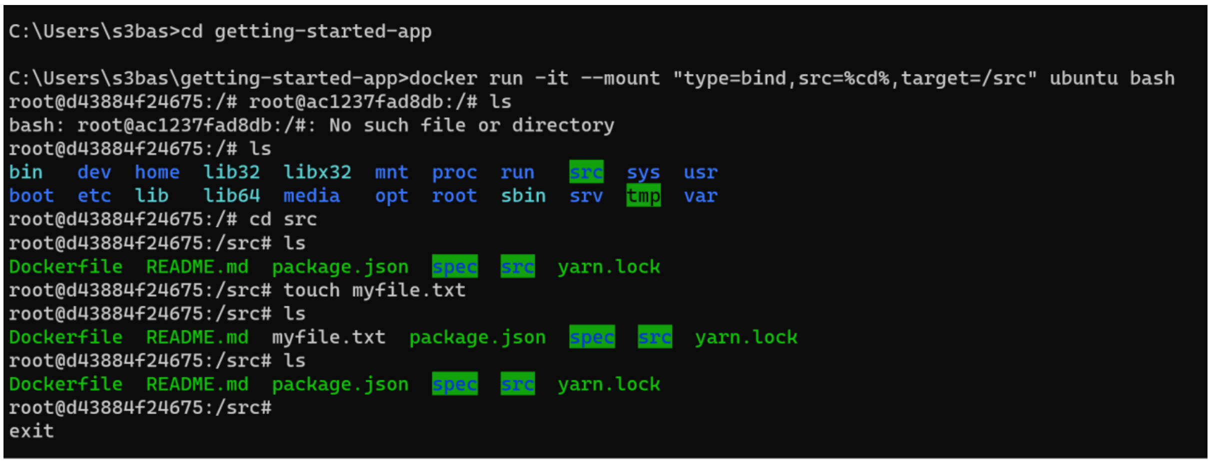
Data persistence verified:



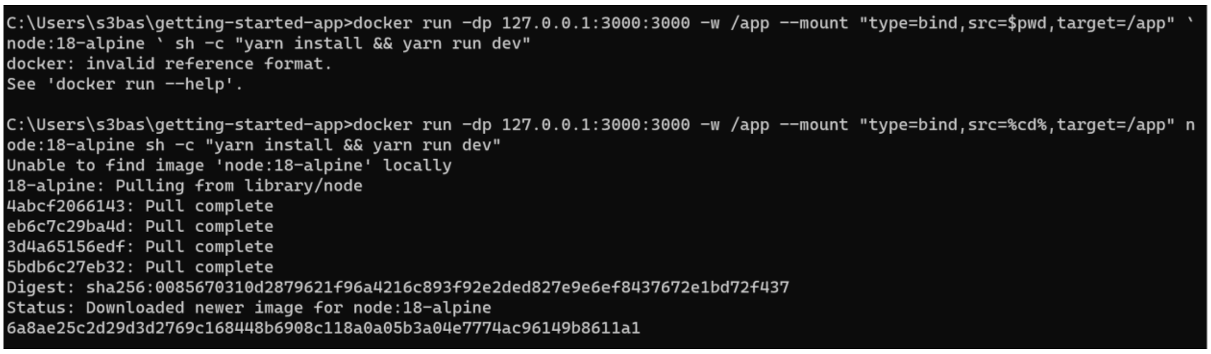


```
C:\Users\s3bas> docker volume inspect todo-db
[
    {
        "CreatedAt": "2024-02-07T09:46:53Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/todo-db/_data",
        "Name": "todo-db",
        "Options": null,
        "Scope": "local"
    }
]

C:\Users\s3bas>
```

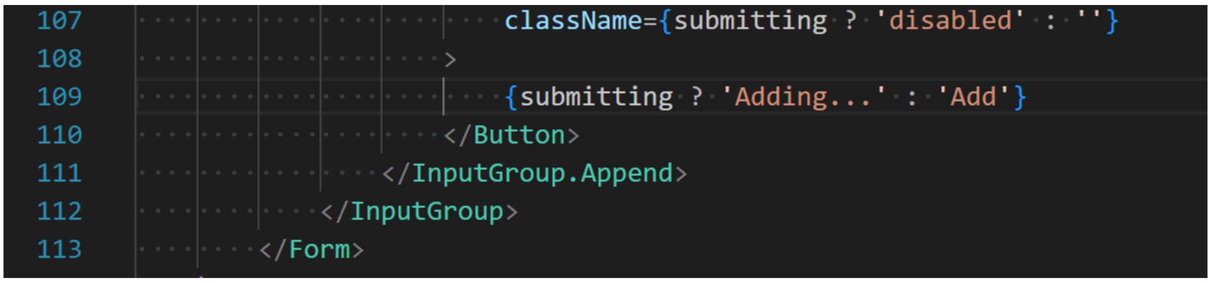
Part 6: Using Blind Mounts



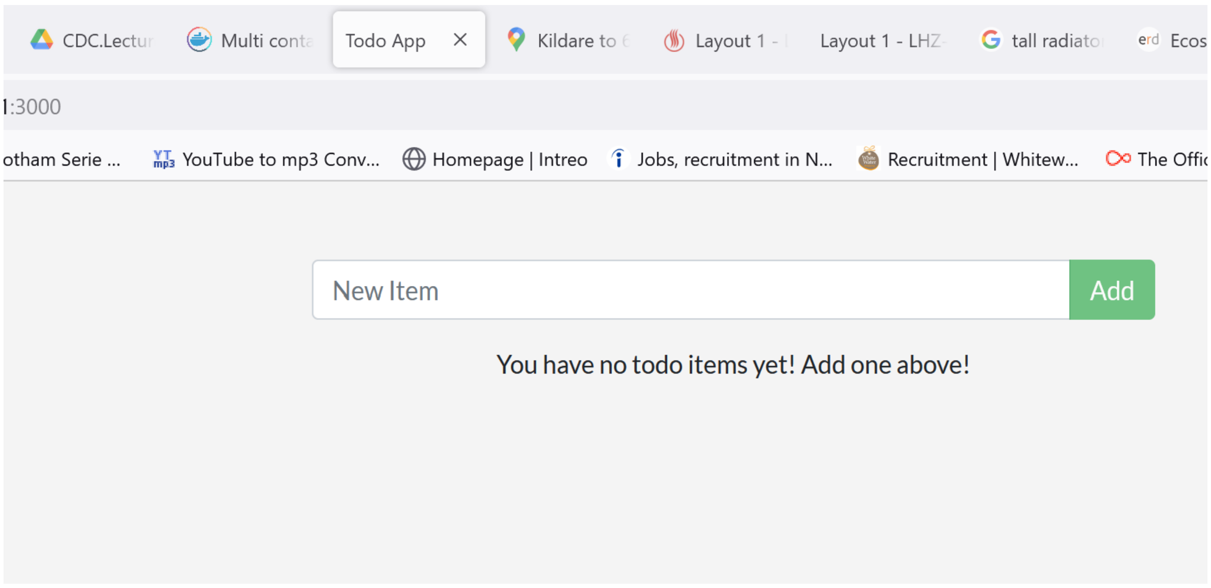
```
C:\Users\s3bas>cd getting-started-app

C:\Users\s3bas\getting-started-app>docker run -it --mount "type=bind,src=%cd%,target=/src" ubuntu bash
root@d43884f24675:/# root@ac1237fad8db:/# ls
bash: root@ac1237fad8db:/#: No such file or directory
root@d43884f24675:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   src  sys  usr
boot  etc  lib   lib64  media   opt  root  sbin  srv  tmp  var
root@d43884f24675:/# cd src
root@d43884f24675:/src# ls
Dockerfile  README.md  package.json  spec  src  yarn.lock
root@d43884f24675:/src# touch myfile.txt
root@d43884f24675:/src# ls
Dockerfile  README.md  myfile.txt  package.json  spec  src  yarn.lock
root@d43884f24675:/src# ls
Dockerfile  README.md  package.json  spec  src  yarn.lock
root@d43884f24675:/src#
exit
```



```
C:\Users\s3bas\getting-started-app>docker run -dp 127.0.0.1:3000:3000 -w /app --mount "type=bind,src=$pwd,target=/app" `
node:18-alpine ` sh -c "yarn install && yarn run dev"
docker: invalid reference format.
See 'docker run --help'.

C:\Users\s3bas\getting-started-app>docker run -dp 127.0.0.1:3000:3000 -w /app --mount "type=bind,src=%cd%,target=/app" n
ode:18-alpine sh -c "yarn install && yarn run dev"
Unable to find image 'node:18-alpine' locally
18-alpine: Pulling from library/node
4abcf2066143: Pull complete
eb6c7c29ba4d: Pull complete
3d4a65156edf: Pull complete
5bdb6c27eb32: Pull complete
Digest: sha256:0085670310d2879621f96a4216c893f92e2ded827e9e6ef8437672e1bd72f437
Status: Downloaded newer image for node:18-alpine
6a8ae25c2d29d3d2769c168448b6908c118a0a05b3a04e7774ac96149b8611a1
```

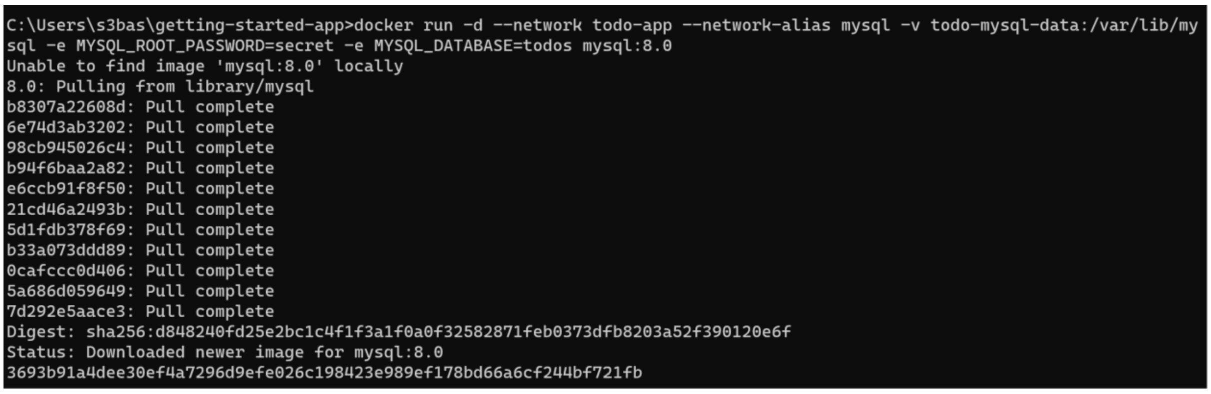


```
                        className={submitting ? 'disabled' : ''}
                    >
                        {submitting ? 'Adding...' : 'Add'}
                    </Button>
                </InputGroup.Append>
            </InputGroup>
        </Form>
```

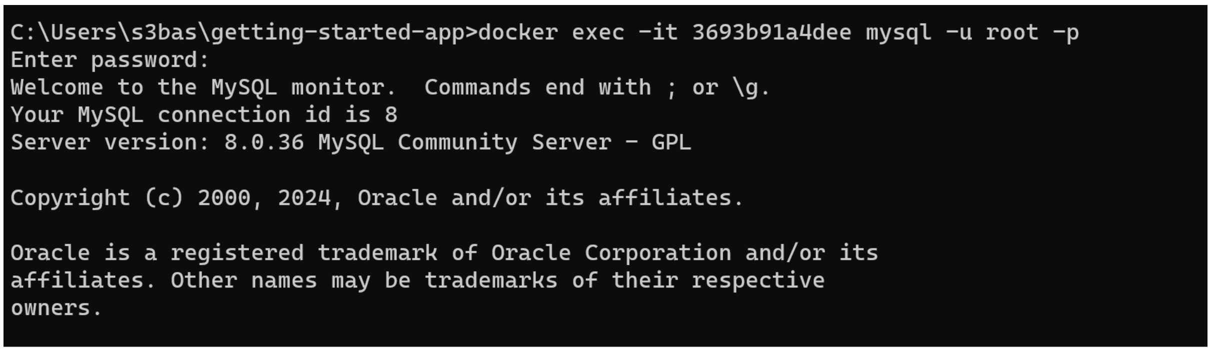


You have no todo items yet! Add one above!

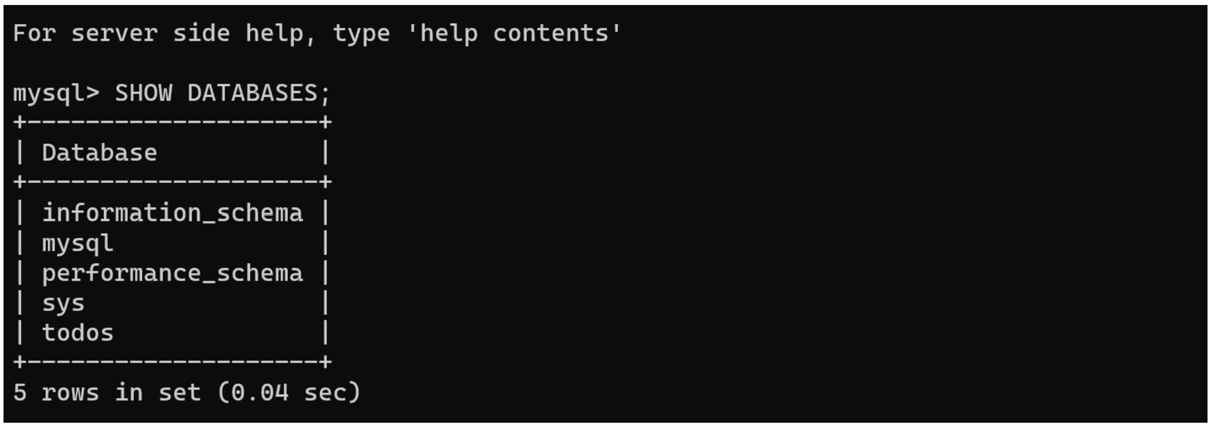
Part 7: Multi container apps



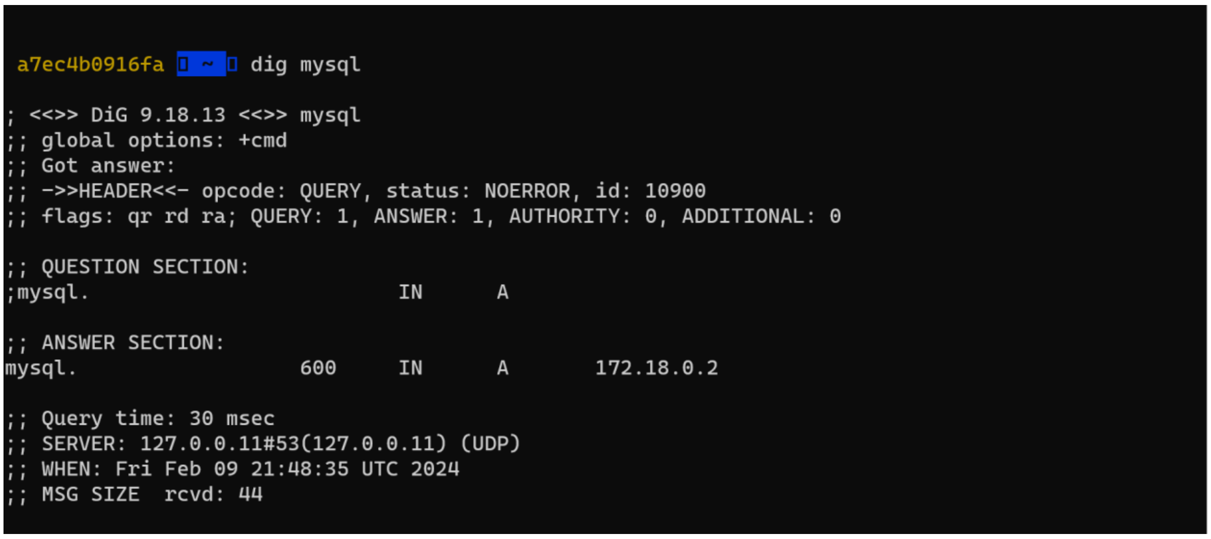
```
C:\Users\s3bas\getting-started-app>docker run -d --network todo-app --network-alias mysql -v todo-mysql-data:/var/lib/my
sql -e MYSQL_ROOT_PASSWORD=secret -e MYSQL_DATABASE=todos mysql:8.0
Unable to find image 'mysql:8.0' locally
8.0: Pulling from library/mysql
b8307a22608d: Pull complete
6e74d3ab3202: Pull complete
98cb945026c4: Pull complete
b94f6baa2a82: Pull complete
e6ccb91f8f50: Pull complete
21cd46a2493b: Pull complete
5d1fdb378f69: Pull complete
b33a073ddd89: Pull complete
0cafccc0d406: Pull complete
5a686d059649: Pull complete
7d292e5aace3: Pull complete
Digest: sha256:d848240fd25e2bc1c4f1f3a1f0a0f32582871feb0373dfb8203a52f390120e6f
Status: Downloaded newer image for mysql:8.0
3693b91a4dee30ef4a7296d9efe026c198423e989ef178bd66a6cf244bf721fb
```



```
C:\Users\s3bas\getting-started-app>docker exec -it 3693b91a4dee mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

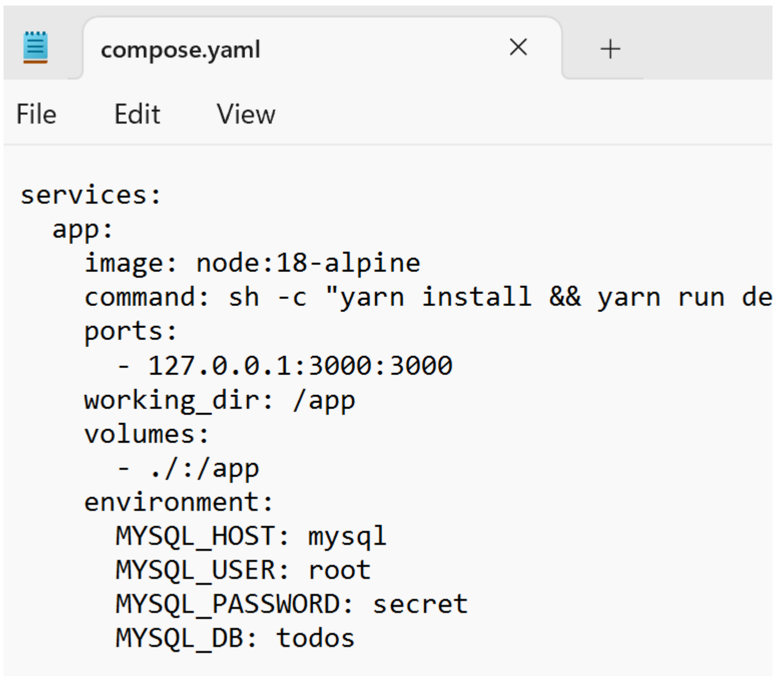


```
For server side help, type 'help contents'

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| todos              |
+--------------------+
5 rows in set (0.04 sec)
```



```
 a7ec4b0916fa  ~  dig mysql

; <<>> DiG 9.18.13 <<>> mysql
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10900
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;mysql.                         IN      A

;; ANSWER SECTION:
mysql.                  600     IN      A       172.18.0.2

;; Query time: 30 msec
;; SERVER: 127.0.0.11#53(127.0.0.11) (UDP)
;; WHEN: Fri Feb 09 21:48:35 UTC 2024
;; MSG SIZE  rcvd: 44
```

Part 8: Using Docker Compose



```
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run de
    ports:
      - 127.0.0.1:3000:3000
    working_dir: /app
    volumes:
      - ./:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos
```