

Lab2 Docker

Name: Kheng Chamnan

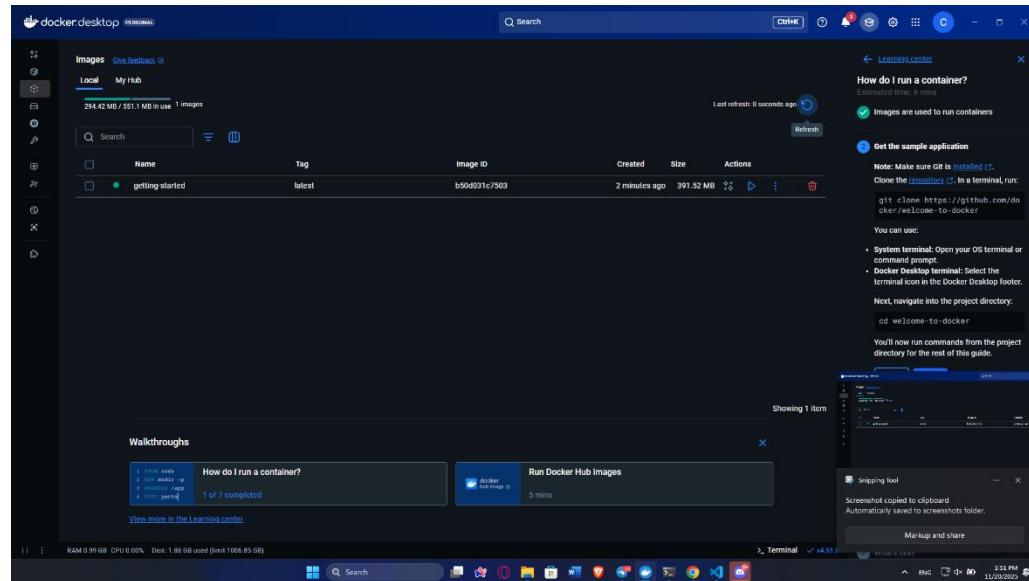
1. Container Applications

In this section I learn how to create a container with node:its-alpine base image how to set working directory inside the container within the existing repository ,copy my local file into WORKDIR

Using yarn command to install image specify index.js as the start application then host it on my localhost

Docker Build Screenshot

Docker Image In Docker Hub



Listing all Image

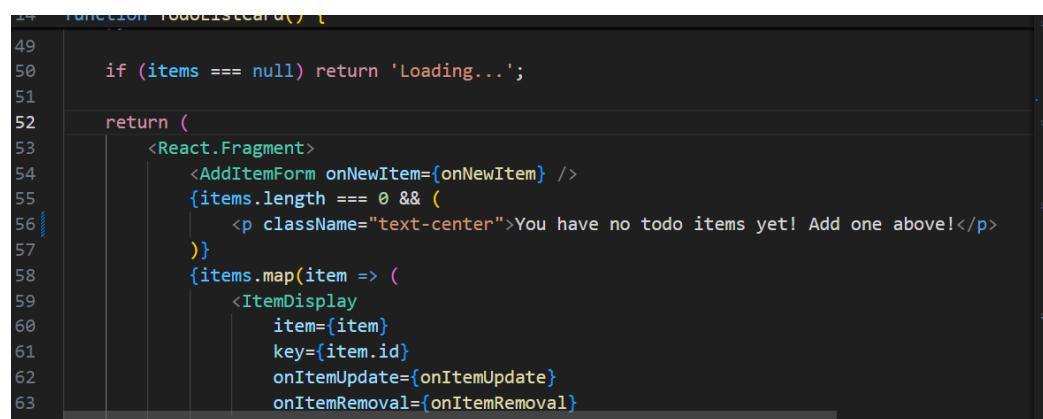
```
PS C:\Users\Lenovo\testing\getting-started-app> docker images -a
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
getting-started    latest   b50d031c7503   15 minutes ago  392MB
PS C:\Users\Lenovo\testing\getting-started-app>
```

2.Update Application

In This section I learn how to update applications in my source code and rebuild my update images

And learn that in order to build new container need to stop the old one by removing it since it host on the same port .

Change Source Code



```
function TodoListCard() {
  const [items, setItems] = useState([]);
  const [newItem, setNewItem] = useState("");
  const [error, setError] = useState(null);

  useEffect(() => {
    fetch("http://localhost:3001/items")
      .then((res) => res.json())
      .then((data) => setItems(data));
  }, []);

  const handleNewItemChange = (e) => {
    setNewItem(e.target.value);
  };

  const handleAddItem = (e) => {
    e.preventDefault();
    if (!newItem) return;
    const newItemObject = { id: Date.now(), title: newItem };
    setItems([newItemObject, ...items]);
    setNewItem("");
    setError(null);
  };

  const handleRemoveItem = (id) => {
    const updatedItems = items.filter((item) => item.id !== id);
    setItems(updatedItems);
  };

  const handleUpdateItem = (id, title) => {
    const updatedItems = items.map((item) => {
      if (item.id === id) {
        item.title = title;
      }
      return item;
    });
    setItems(updatedItems);
  };

  return (
    <React.Fragment>
      <AddItemForm onNewItem={onNewItem} />
      {items.length === 0 && (
        <p className="text-center">You have no todo items yet! Add one above!</p>
      )}
      {items.map(item => (
        <ItemDisplay
          item={item}
          key={item.id}
          onItemUpdate={onItemUpdate}
          onItemRemoval={onItemRemoval}
        </ItemDisplay>
      ))}
    </React.Fragment>
  );
}
```

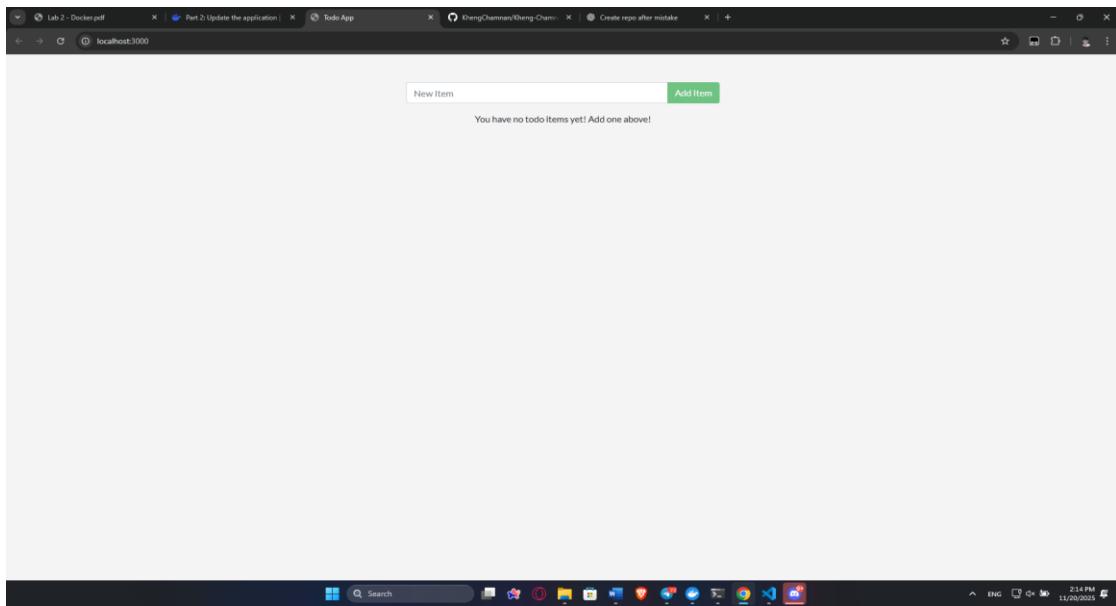
Error From start new docker

```
PS C:\Users\Lenovo\testing\getting-started-app> docker run -dp 127.0.0.1:3000 getting-started
826e27f3a47dd706ec35260c3a7ba6433693de49c2ceb552ecf88bab8597656e
docker: Error response from daemon: failed to set up container networking: driver failed programming external connectivity on endpoint reverent_kare (ed2d2d71340ea327eb28ea9b9108104ac08d9505e4616f
de18b3af4eabaf00c): Bind for 127.0.0.1:3000 failed: port is already allocated
Run 'docker run --help' for more information
PS C:\Users\Lenovo\testing\getting-started-app>
```

Remove old container and rebuild new container

```
Run 'docker --help' for more information
PS C:\Users\Lenovo\testing\getting-started-app> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
f8c30826f9f4        b50d031c7503   "docker-entrypoint.s..."   22 minutes ago   Up 22 minutes   127.0.0.1:3000->3000/tcp   compassionate_ramanujan
PS C:\Users\Lenovo\testing\getting-started-app> docker stop f8c30826f9f4
f8c30826f9f4
PS C:\Users\Lenovo\testing\getting-started-app> docker rm f8c30826f9f4
f8c30826f9f4
PS C:\Users\Lenovo\testing\getting-started-app> docker run -dp 127.0.0.1:3000:3000 getting-started
>>
76718df3ff6421d5bf9e8d7c2237d09ad9c300817207c967e256a00d0243ecdb
PS C:\Users\Lenovo\testing\getting-started-app>
```

New Updated Application



3. Share The application

In this section I learning how to push my local docker to docker repo and can be share anyone for example we can share it on dockerplay

Docker push erro with wrong DockerID

```
PS C:\Users\Lenovo\testing\getting-started-app> docker push docker/getting-started
@>>
Using default tag: latest
The push refers to repository [docker.io/docker/getting-started]
tag does not exist: docker/getting-started:latest
```

Login and set a right docker tag base on my created Repo

```
up chamnann1: no such host
● PS C:\Users\Lenovo\testing\getting-started-app> docker login
Authenticating with existing credentials... [Username: chamnann1]

💡 Info → To Login with a different account, run 'docker Logout' followed by 'docker Login'

Login Succeeded
● PS C:\Users\Lenovo\testing\getting-started-app> docker tag getting-started chamnann1/getting-started
○ PS C:\Users\Lenovo\testing\getting-started-app> █
```

Push image to repo

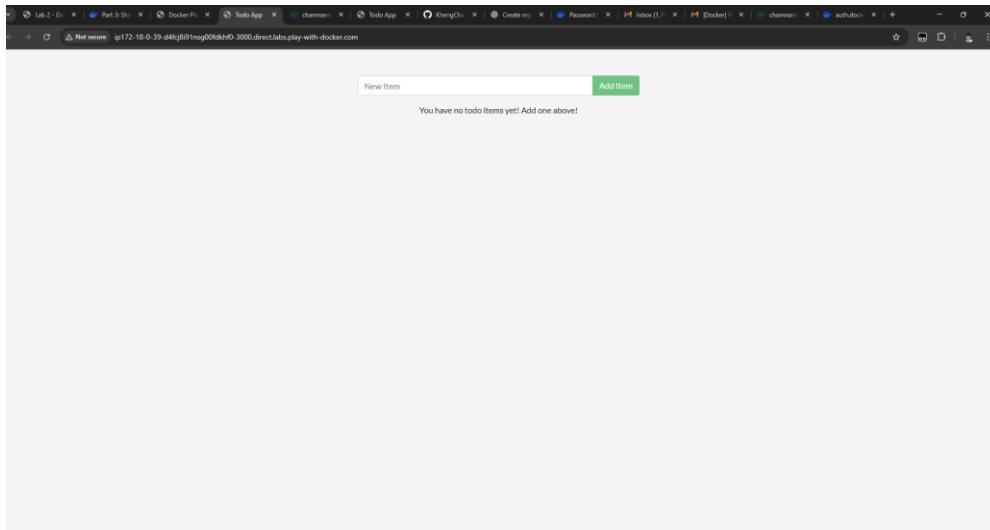
```
💡 Info → To Login with a different account, run 'docker Logout' followed by 'docker Login'

Login Succeeded
PS C:\Users\Lenovo\testing\getting-started-app>
● PS C:\Users\Lenovo\testing\getting-started-app> docker push chamnann1/getting-started:latest
The push refers to repository [docker.io/chamnann1/getting-started]
2d35ebdb57d9: Pushed
e731d1c17be0: Pushed
b5c085f6a80a: Pushed
0ad6b75bfecf: Pushed
28b3fff5fc34: Pushed
1cf8ffe47b87: Pushed
c381d9bdb5ee: Pushed
572392f439b9: Pushed
latest: digest: sha256:3b91ba959384a952851f1444dbbf088281874a7c2c382b7870e85ffcb5a8b54e size: 856
○ PS C:\Users\Lenovo\testing\getting-started-app> █
```

Push image to dockerplay

```
#####
#           WARNING!!!
# This is a sandbox environment. Using personal credentials
# is HIGHLY! discouraged. Any consequences of doing so are
# completely the user's responsibilities.
#
# The PWD team.
#####
(node1) (local) root@192.168.0.29 ~
$ docker run -dp 0.0.0.0:3000 chamnann1/getting-started
docker: invalid hostPort: 0.0.0.0.
See 'docker run --help'.
(node1) (local) root@192.168.0.29 ~
$ docker run -dp 0.0.0.0:3000:000 chamnann1/getting-started
Unable to find image 'chamnann1/getting-started:latest' locally
latest: Pulling from chamnann1/getting-started
2d35ebdb57d9: Full complete
0ad6b75bfecf: Full complete
572392f439b9: Full complete
e731d1c17be0: Full complete
b5c085f6a80a: Full complete
28b3fff5fc34: Full complete
c381d9bdb5ee: Full complete
Digest: sha256:3b91ba959384a952851f1444dbbf088281874a7c2c382b7870e85ffcb5a8b54e
Status: Downloaded newer image for chamnann1/getting-started:latest
95fa1cd5f45371996d702d463a205d42a43497449ff3cf13b4645546f4f60b5
(node1) (local) root@192.168.0.29 ~
$ █
```

Application Run on dockerplay



4. Persist the DB

In this section I learn that data are not persist if we run new container using same image unless we mount it to out volume

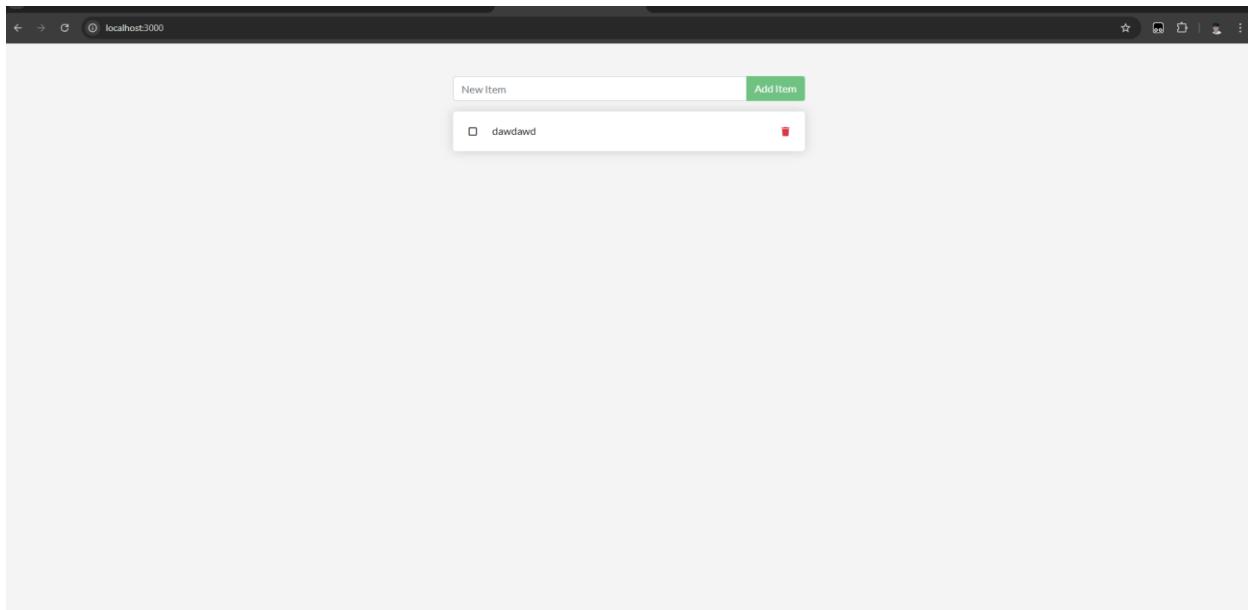
Data is not persist

```
① >> C:\Users\Lenovo\testing\getting-started-app>
    stat: can't stat 'greeting.txt': No such file or directory
② PS C:\Users\Lenovo\testing\getting-started-app> docker run --rm alpine stat greeting.txt
    >>
    stat: can't stat 'greeting.txt': No such file or directory
③ PS C:\Users\Lenovo\testing\getting-started-app> docker run --rm alpine touch greeting.txt
```

Create volume and mount to todos on my host machine

```
todo-db
● PS C:\Users\Lenovo\testing\getting-started-app> docker rm -f 66efcd40e5cd
66efcd40e5cd
● PS C:\Users\Lenovo\testing\getting-started-app> docker run -dp 127.0.0.1:3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
02222b5752f0e7590f683754a8ff8460530edd95f3d2c688947d4322b33d4101
○ PS C:\Users\Lenovo\testing\getting-started-app>
```

Data still exist when we run our container mount to todo volume



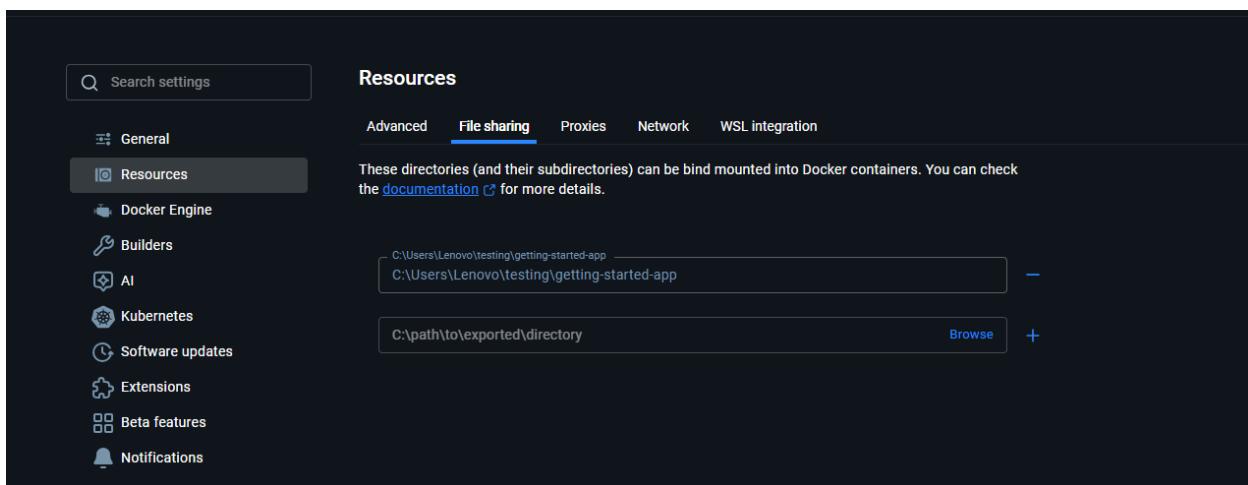
Proof of mount to local machine

```
[  
● PS C:\Users\Lenovo\testing\getting-started-app> docker volume ls  
DRIVER      VOLUME NAME  
local        todo-db  
○ PS C:\Users\Lenovo\testing\getting-started-app> ]
```

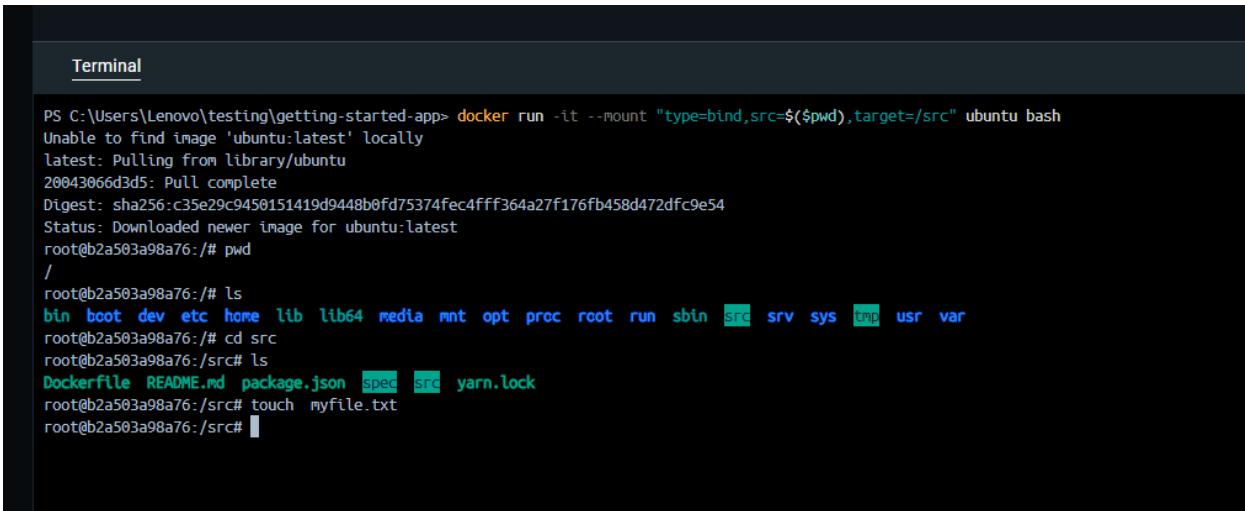
5. Use Bind mount

In this section

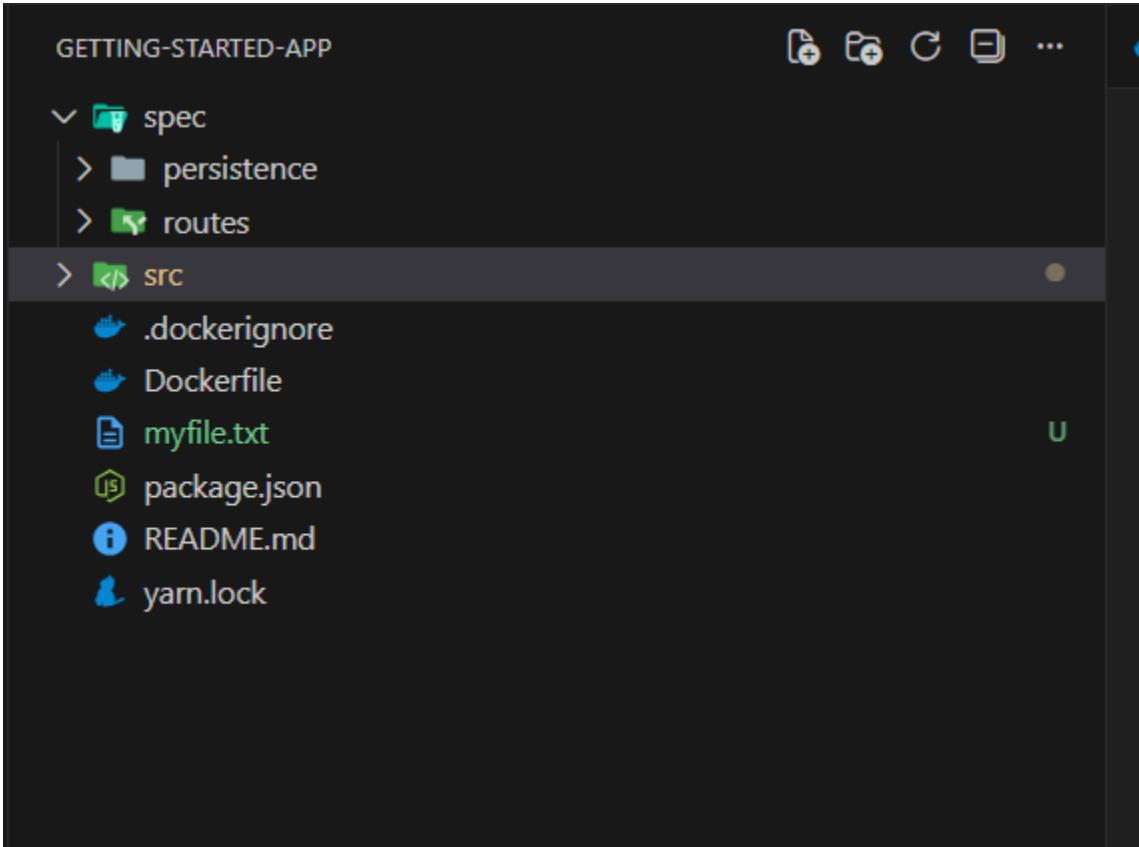
Define directory



Mount to my local machine and verify



```
PS C:\Users\Lenovo\testing\getting-started-app> docker run -it --mount "type=bind,src=$pwd,target=/src" ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
20043066d3d5: Pull complete
Digest: sha256:c35e29c9450151419d9448b0fd75374fec4fff364a27f176fb458d472dfc9e54
Status: Downloaded newer image for ubuntu:latest
root@b2a503a98a76:/# pwd
/
root@b2a503a98a76:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin sys tmp usr var
root@b2a503a98a76:/# cd src
root@b2a503a98a76:/src# ls
Dockerfile README.md package.json spec src yarn.lock
root@b2a503a98a76:/src# touch myfile.txt
root@b2a503a98a76:/src#
```



Verify delete in host and check in the container session

```
root@b2a503a98a76:/src# ls
Dockerfile README.md package.json spec src yarn.lock
root@b2a503a98a76:/src# touch myfile.txt
root@b2a503a98a76:/src# ls
Dockerfile README.md myfile.txt package.json spec src yarn.lock
root@b2a503a98a76:/src# ls
Dockerfile README.md package.json spec src yarn.lock
root@b2a503a98a76:/src#
```

6. Multiple container

In this section I learn how to run other container such as mysql and know how to let each container talk to each other by staying in the same network by mount the volume to the mysql our data now are store in mysql

Mysql container

```
>> -e MYSQL_DATABASE=todos ...
PS C:\Users\Lenovo\testing\getting-started-app> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
fbab3ec3bb7 mysql:8.0 "docker-entrypoint.s..." 7 seconds ago Up 4 seconds 3306/tcp, 33060/tcp romantic_mcclintock
PS C:\Users\Lenovo\testing\getting-started-app> docker exec -it fbab3ec3bb7 mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.44 MySQL Community Server - GPL

Copyright (c) 2000, 2025, 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.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| todos |
+-----+
5 rows in set (0.01 sec)

mysql>
```

Connect app to mysql

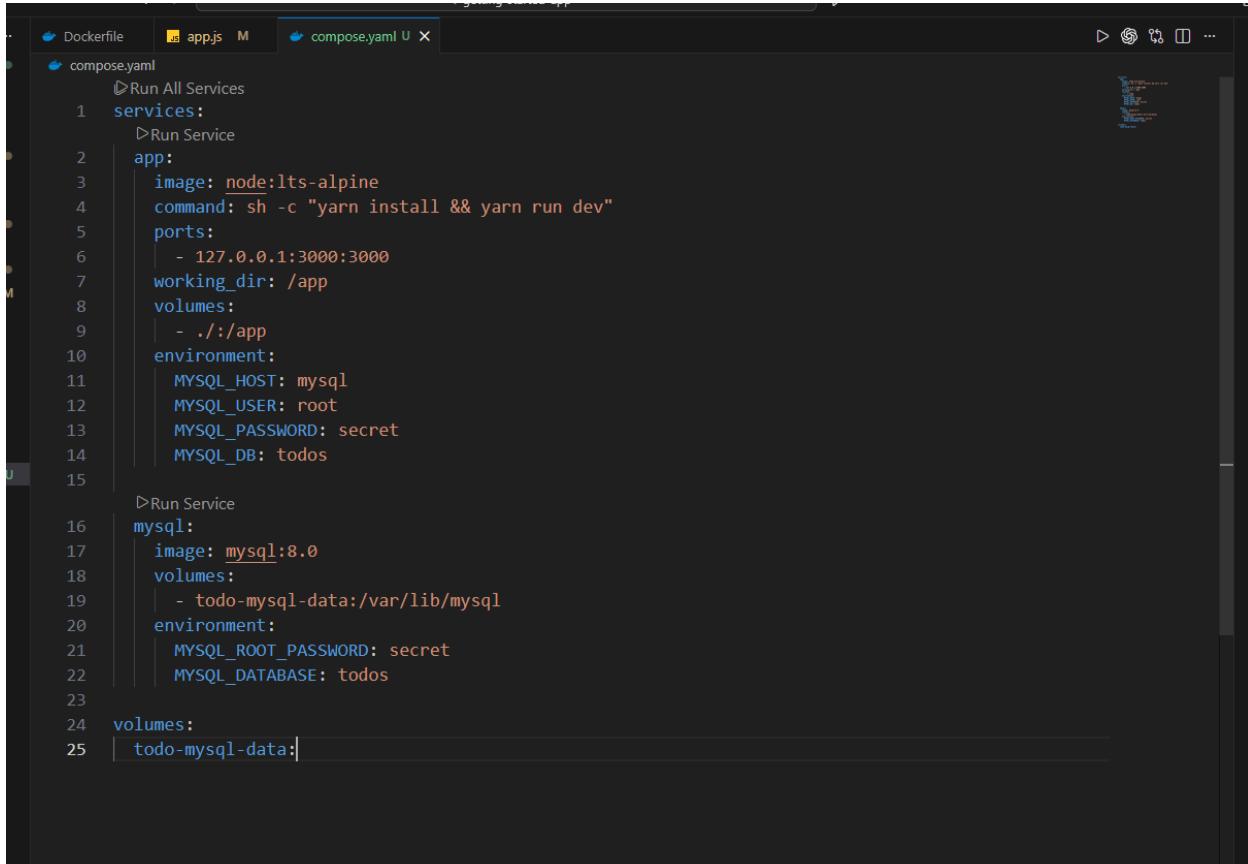
```
PS C:\Users\Lenovo\Testing\getting-started-app> docker run -it --network todo-app nicolaka/netshoot
4638114456c > exit
PS C:\Users\Lenovo\Testing\getting-started-app> docker run -dp 127.0.0.1:3000:3000
>> -u /app -v $(pwd):/app -
>> -e MYSQL_DATABASE=todos -
>> -e MYSQL_USER=root -
>> -e MYSQL_PASSWORD=secret -
>> -e MYSQL_DB=todos -
>> --node:alpine
>> which curl & curl install & curl run dev*
Unable to find image "node:alpine" locally
node:alpine: Pulling from library/node
Digest: sha256:28676558f9d8b5b059a0ffff28741f11a84d985c732e60e19e0e75c7239c43
Status: Downloaded newer image for node:alpine
3c8975268e75aeff38f8ac2c2d8ca35c37e3d018739f7ec96813b394a18d7ed10
PS C:\Users\Lenovo\Testing\getting-started-app> docker log -f 3c8975268e75aeff38f8ac2c2d8ca35c37e3d018739f7ec96813b394a18d7ed10
unknown shorthand flag: 'f' in '-f'

Usage: docker [OPTIONS] COMMAND [ARG...]

Run 'docker --help' for more information
PS C:\Users\Lenovo\Testing\getting-started-app> docker logs -f 3c8975268e75aeff38f8ac2c2d8ca35c37e3d018739f7ec96813b394a18d7ed10
yarn install v1.22.22
[1/4] Resolving packages...
(node:8) [DEP0109] DeprecationWarning: 'url.parse()' behavior is not standardized and prone to errors that have security implications. Use the WHATWG URL API instead.
  CVEs are not issued for 'url.parse()' vulnerabilities.
  (Use node --trace-deprecation ... to show where the warning was created)
[2/4] Linking dependencies...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
Done in 118.21s.
yarn run v1.22.22
$ nodemon -l ./src/index.js
[nodemon] 0.0.0.0:3000
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
Waiting for mysql:3306.
Connected to mysql db at host mysql
Listening on port 3000
|
```

7.Docker Compose

in this section I learn how to define all container using compose yaml which is easy instead of typing and run each container and benefit from compose yaml is that when we run stack of application it create a network for us automatically too.



```
Dockerfile app.js compose.yaml

compose.yaml
  services:
    app:
      image: node:lts-alpine
      command: sh -c "yarn install && yarn run dev"
      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
    mysql:
      image: mysql:8.0
      volumes:
        - todo-mysql-data:/var/lib/mysql
      environment:
        MYSQL_ROOT_PASSWORD: secret
        MYSQL_DATABASE: todos
  volumes:
    todo-mysql-data:
```



8. Image Building best practice

In this section, I learned about caching layers in Docker image building, which decreases the build time because unchanged layers (like dependencies) don't need to be rebuilt every time I make changes to my code. By copying package.json and yarn.lock first and installing dependencies before copying the rest of the code, I can reuse the cached layers for faster builds

Build image with cache

```
PS C:\Users\Lenovo\testing\getting-started-app> docker build -t getting-started .
[+] Building 27.4s (13/13) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 21B
  => resolve image config for docker-image://docker.io/docker/dockerfile:1
  => [auth] docker/dockerfile:pull token for registry-1.docker.io
  => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:b6af4d42430b15f2d2a4c5a02b919e98a525b785b1aaff16747d2f623364e39b6
  => => resolve docker.io/docker/dockerfile:1@sha256:b6af4d42430b15f2d2a4c5a02b919e98a525b785b1aaff16747d2f623364e39b6
  => [internal] load metadata for docker.io/library/node:lts-alpine
  => [auth] library/node:pull token for registry-1.docker.io
  => [internal] load .dockerignore
  => => transferring context: 66B
  => [1/4] FROM docker.io/library/node:lts-alpine@sha256:2867d550cf9d8bb50059a0fff528741f11a84d985c732e60e19e8e75c7239c43
  => => resolve docker.io/library/node:lts-alpine@sha256:2867d550cf9d8bb50059a0fff528741f11a84d985c732e60e19e8e75c7239c43
  => [internal] load build context
  => => transferring context: 64B
  => CACHED [2/4] WORKDIR /app
  => [3/4] COPY package.json yarn.lock ./ 
  => [4/4] RUN yarn install --production
  => exporting to image
  => => exporting layers
  => => exporting manifest sha256:2beaf5fdef9ac752bff56d12bbfc94dd5bfa0e23656e775f79333d84b12fa626
  => => exporting config sha256:6354c322b1306ed2d15217b688cbae9955aa96474e5fa0515fc8bf2a92ccf3e
  => => exporting attestation manifest sha256:64248f12ce1649cdf8a3143c16ab30981d481d54eac1a14f9d4186d8807346c
  => => exporting manifest list sha256:2a993c7cdc3477c9ca5b374308a755ec2973003348818f470b1d356124e14440
  => => naming to docker.io/library/getting-started:latest
  => => unpacking to docker.io/library/getting-started:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/wx2u902zsystc98zfcnpr0iq5
PS C:\Users\Lenovo\testing\getting-started-app>
```

With no change dependencies

```
PS C:\Users\Lenovo\testing\getting-started-app> docker build -t getting-started .
[+] Building 2.5s (11/11) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 21B
  => resolve image config for docker-image://docker.io/docker/dockerfile:1
  => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:b6af4d42430b15f2d2a4c5a02b919e98a525b785b1aaff16747d2f623364e39b6
  => => resolve docker.io/docker/dockerfile:1@sha256:b6af4d42430b15f2d2a4c5a02b919e98a525b785b1aaff16747d2f623364e39b6
  => [internal] load metadata for docker.io/library/node:lts-alpine
  => [internal] load .dockerignore
  => => transferring context: 66B
  => [1/4] FROM docker.io/library/node:lts-alpine@sha256:2867d550cf9d8bb50059a0fff528741f11a84d985c732e60e19e8e75c7239c43
  => => resolve docker.io/library/node:lts-alpine@sha256:2867d550cf9d8bb50059a0fff528741f11a84d985c732e60e19e8e75c7239c43
  => [internal] load build context
  => => transferring context: 64B
  => CACHED [2/4] WORKDIR /app
  => CACHED [3/4] COPY package.json yarn.lock ./ 
  => CACHED [4/4] RUN yarn install --production
  => exporting to image
  => => exporting layers
  => => exporting manifest sha256:2beaf5fdef9ac752bff56d12bbfc94dd5bfa0e23656e775f79333d84b12fa626
  => => exporting config sha256:6354c322b1306ed2d15217b688cbae9955aa96474e5fa0515fc8bf2a92ccf3e
  => => exporting attestation manifest sha256:c0f69d0566eadf3a7fa47c62cb5848ed693a2f434ee0d65fc56fb63642756d
  => => exporting manifest list sha256:0f0cd23411f16d6914bbb8ca9af38e65c63056a7c658eb8899dc94385674dc07
  => => naming to docker.io/library/getting-started:latest
  => => unpacking to docker.io/library/getting-started:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/crreoxfp194jz74hf1jissbq
PS C:\Users\Lenovo\testing\getting-started-app>
```

Check to see if data are add to the database

```
mysql> select *from todo_items
-> ;
ERROR 1046 (3D000): No database selected
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| todos |
+-----+
5 rows in set (0.00 sec)

mysql> use todos;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from todo_items;
+----+-----+-----+
| id | name | completed |
+----+-----+-----+
| 272efa7f-68e3-4633-87c4-b739ab3ded43 | hi | 0 |
| df2fb5c8-7dd8-4810-b1e4-89f1e60ebfff | hello | 0 |
+----+-----+-----+
2 rows in set (0.01 sec)
```

App running on compose

