CONTINUOUS ASSESSMENT

LAB 3

CLOUD DATA CENTRES

ZAPPC4202

CREATED BY: SAEDATOUL ASTHEERAH BINTI MISLAN

ID NUMBER: C00290913

YEAR: 3

COURSE: INFORMATION TECHNOLOGY MANAGEMENT

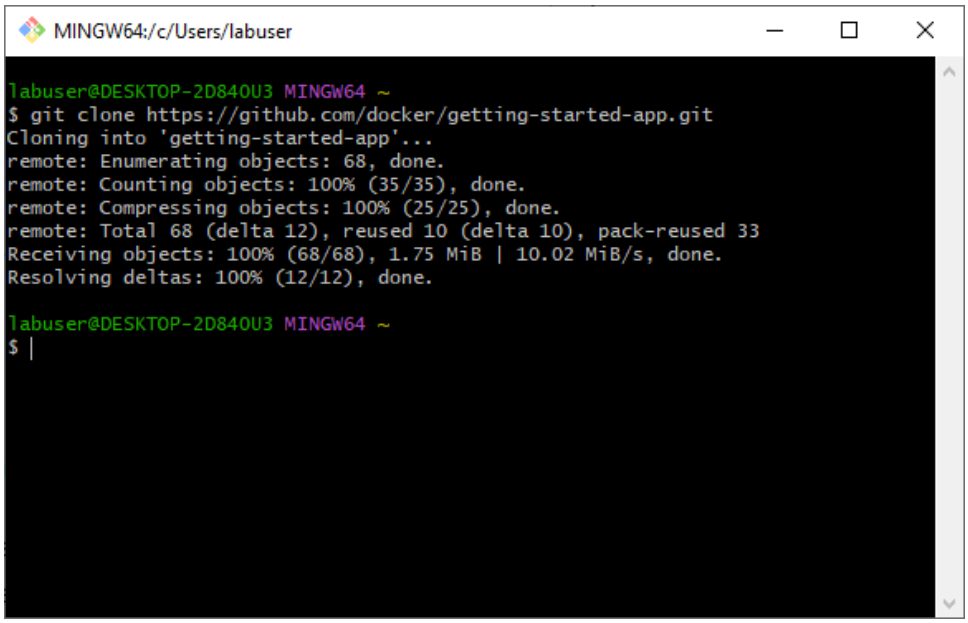
ISSUED YEAR: 2023

LECTURER: LEI SHI

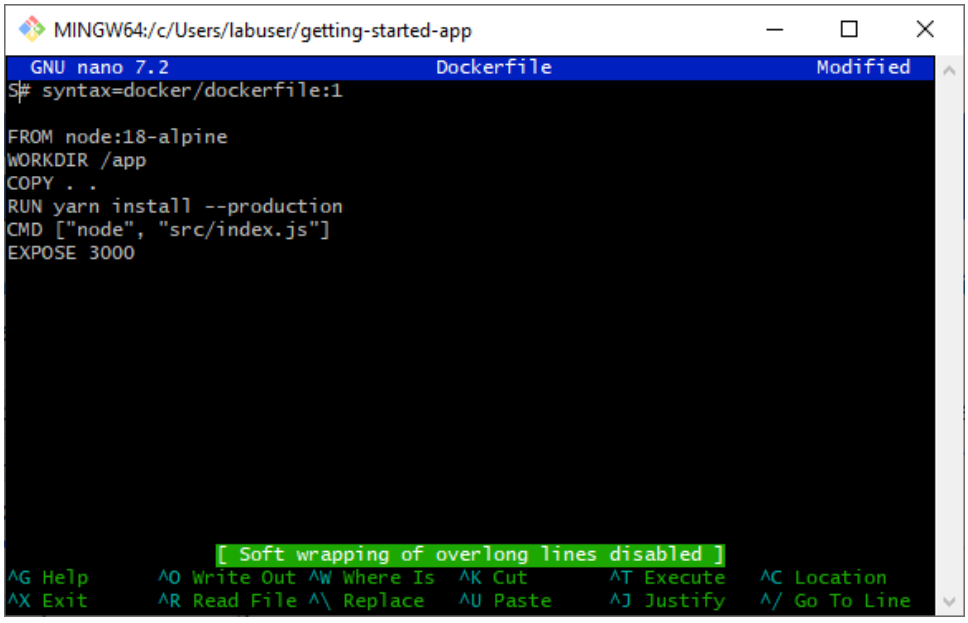
Contents

Part 2: Containerize an application

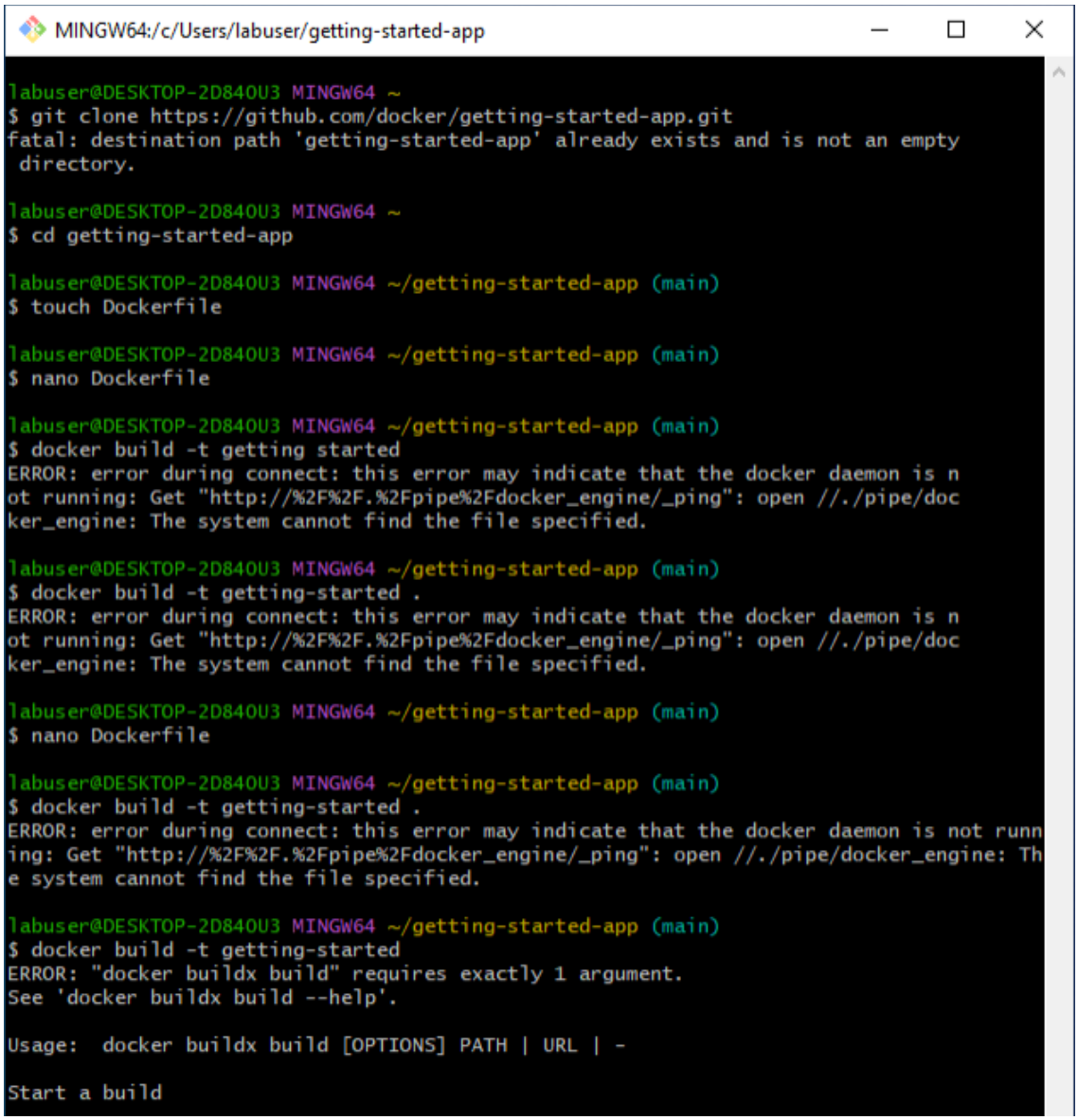
Here’s get the application source code onto my machine

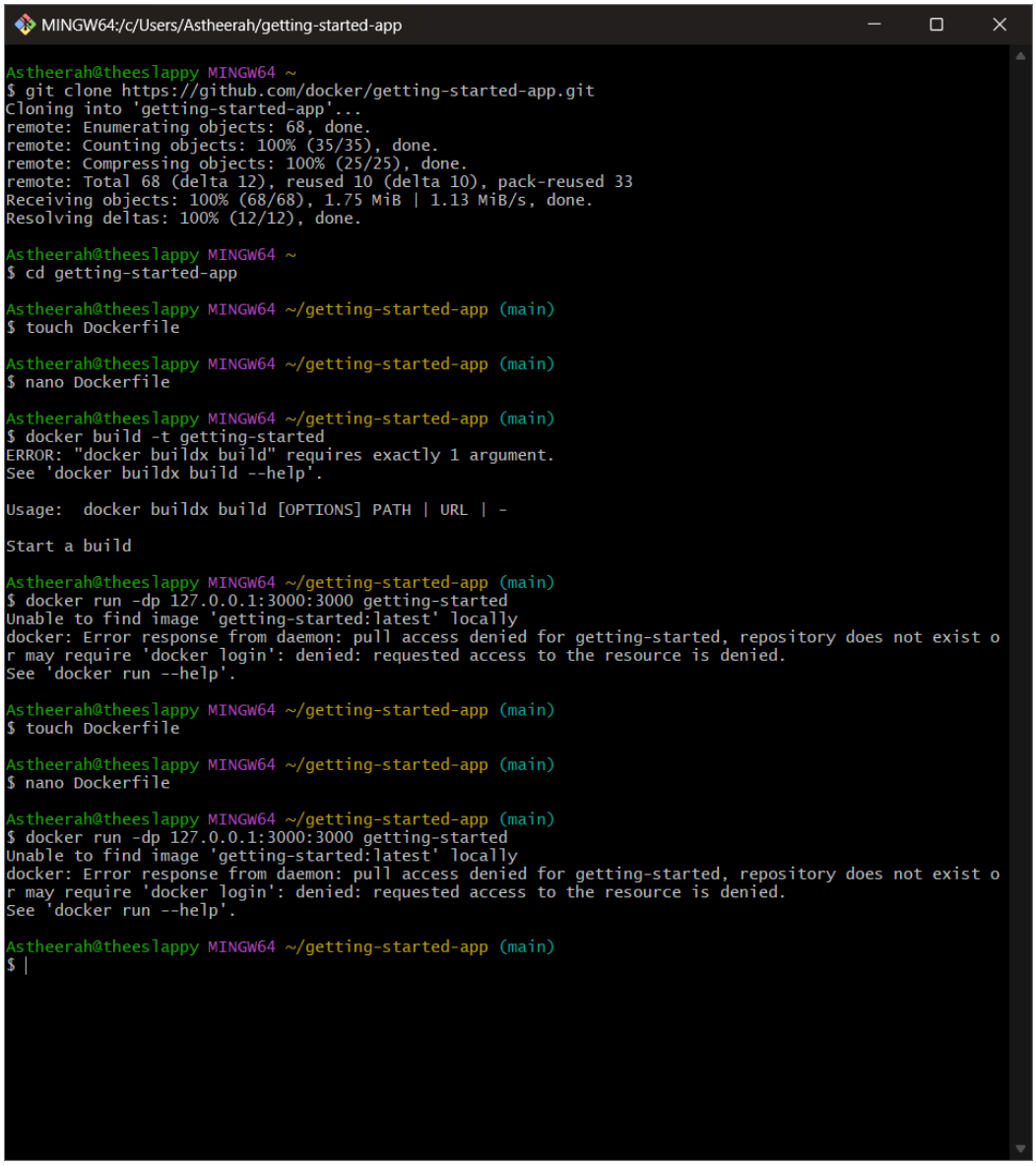


Apps of all kinds, both front-end and back-end apps, can be containerised.



Now that I have created the Dockerfile, I can build the Docker image by running this command in my project directory:



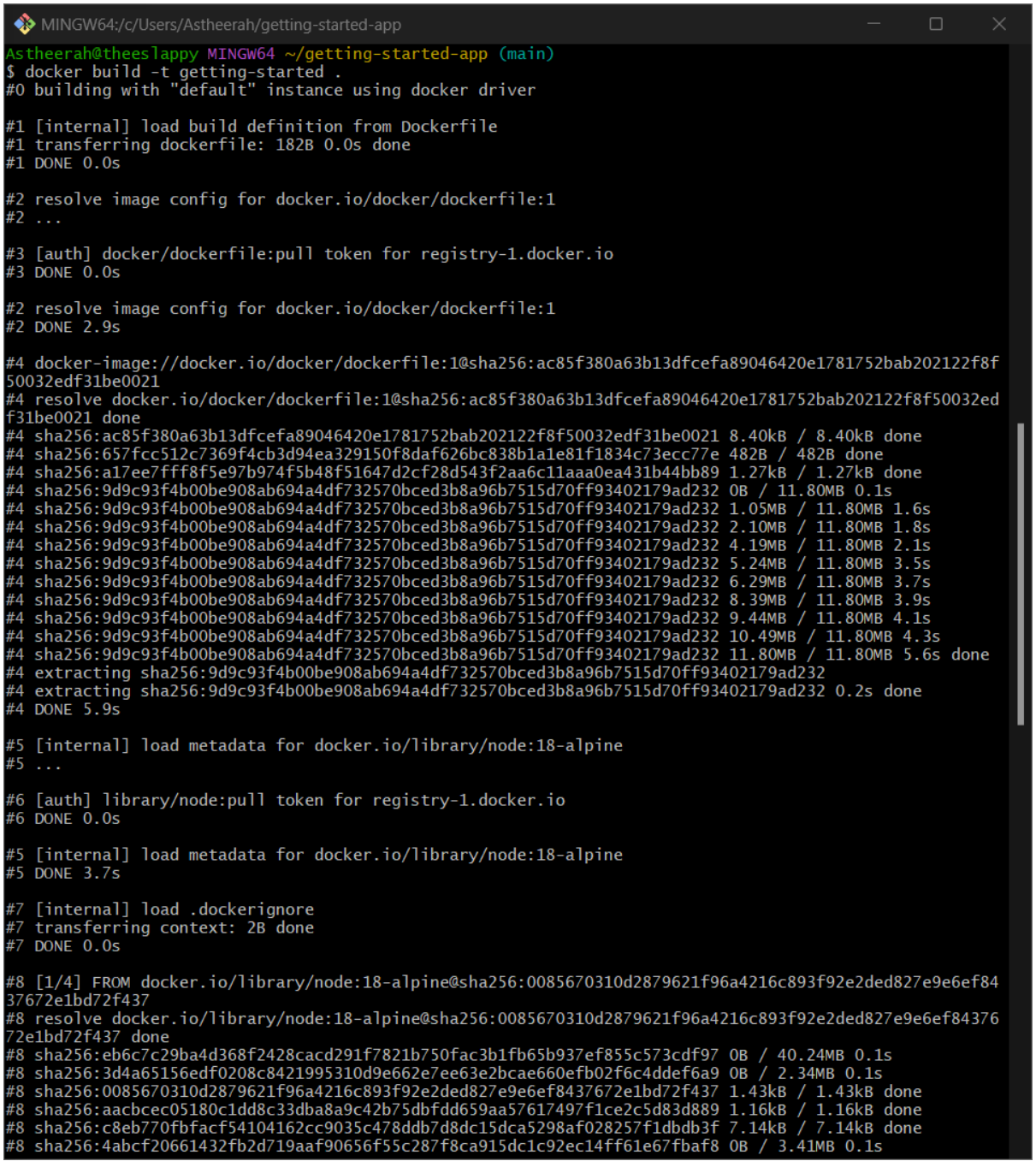


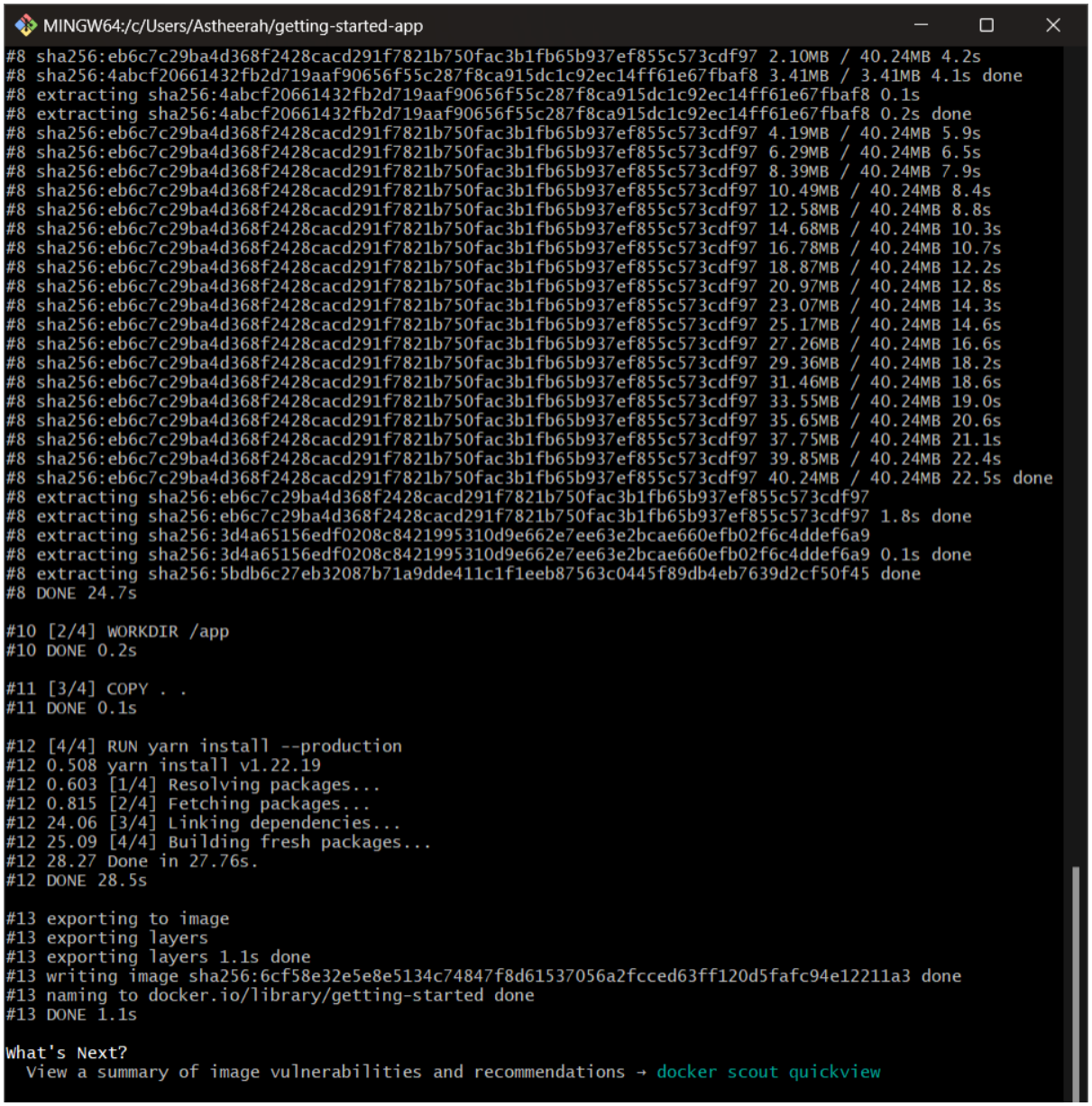
Using a text editor or code editor, add the following contents to the Dockerfile.

Build the image.

The command's '.' specifies the build context, which is the current directory; -t stands for tag and gives the image a name; flask-app is the name we will give the image.

Once I run the command, Docker will build my image, and the output will come out like this:




MINGW64:/c/Users/Astheerah/getting-started-app
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 2.10MB / 40.24MB 4.2s
#8 sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8 3.41MB / 3.41MB 4.1s done
#8 extracting sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8 0.1s
#8 extracting sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8 0.2s done
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 4.19MB / 40.24MB 5.9s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 6.29MB / 40.24MB 6.5s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 8.39MB / 40.24MB 7.9s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 10.49MB / 40.24MB 8.4s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 12.58MB / 40.24MB 8.8s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 14.68MB / 40.24MB 10.3s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 16.78MB / 40.24MB 10.7s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 18.87MB / 40.24MB 12.2s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 20.97MB / 40.24MB 12.8s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 23.07MB / 40.24MB 14.3s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 25.17MB / 40.24MB 14.6s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 27.26MB / 40.24MB 16.6s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 29.36MB / 40.24MB 18.2s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 31.46MB / 40.24MB 18.6s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 33.55MB / 40.24MB 19.0s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 35.65MB / 40.24MB 20.6s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 37.75MB / 40.24MB 21.1s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 39.85MB / 40.24MB 22.4s
#8 sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 40.24MB / 40.24MB 22.5s done
#8 extracting sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97
#8 extracting sha256:eb6c7c29ba4d368f2428cacd291f7821b750fac3b1fb65b937ef855c573cdf97 1.8s done
#8 extracting sha256:3d4a65156edf0208c8421995310d9e662e7ee63e2bcae660efb02f6c4ddef6a9
#8 extracting sha256:3d4a65156edf0208c8421995310d9e662e7ee63e2bcae660efb02f6c4ddef6a9 0.1s done
#8 extracting sha256:5bdb6c27eb32087b71a9dde411c1f1eeb87563c0445f89db4eb7639d2cf50f45 done
#8 DONE 24.7s
#10 [2/4] WORKDIR /app
#10 DONE 0.2s
#11 [3/4] COPY . .
#11 DONE 0.1s
#12 [4/4] RUN yarn install --production
#12 0.508 yarn install v1.22.19
#12 0.603 [1/4] Resolving packages...
#12 0.815 [2/4] Fetching packages...
#12 24.06 [3/4] Linking dependencies...
#12 25.09 [4/4] Building fresh packages...
#12 28.27 Done in 27.76s.
#12 DONE 28.5s
#13 exporting to image
#13 exporting layers
#13 exporting layers 1.1s done
#13 writing image sha256:6cf58e32e5e8e5134c74847f8d61537056a2fcced63ff120d5fafc94e12211a3 done
#13 naming to docker.io/library/getting-started done
#13 DONE 1.1s
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview

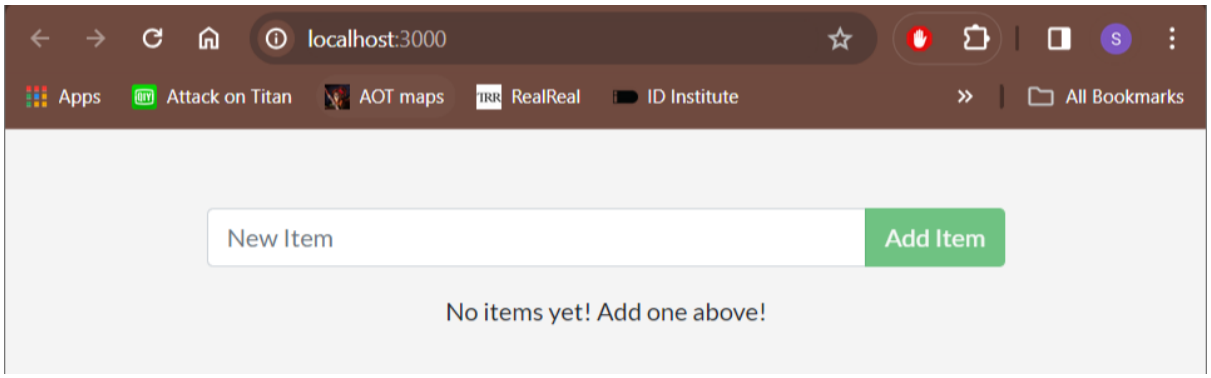
Running the Docker container

Now that the Docker image has been created, you can run the container with the following command in the project directory:

docker run -dp 127.0.0.1:3000:3000 getting-started

The command mentioned above instructs Docker to run the container in interactive mode, allowing you to communicate with it via its shell and assign a console to it that is text-based. However, since Flask isn't a command-line tool that needs an interactive terminal, using -it when launching the application isn't necessarily required. The port mapping of the container is specified by the -p flag. The port on the host system that you wish to map to the container port is the first 3000. The cargo port is located at 3000. The picture we just built and wish to execute is called flask-app.

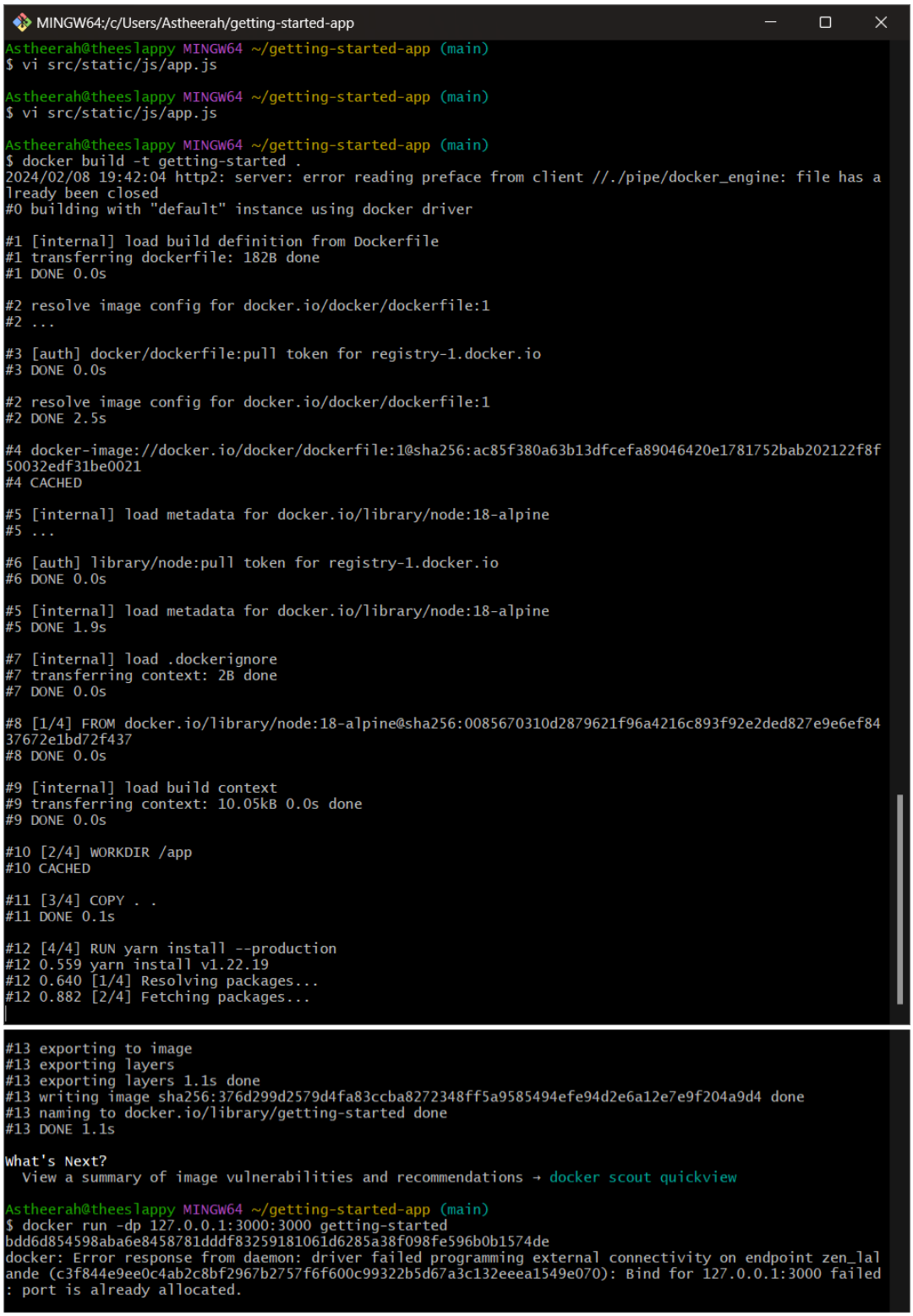
And when you browse to http://localhost:3000, you will see the very basic Flask application.



Run the following docker ps command in a terminal to list your containers.

Part 3: Update the application

Update the source code



In the src/static/js/app.js file, update line 56 to use the new empty text. This is to change the "empty text" to "You have no todo items yet! Add one above!"

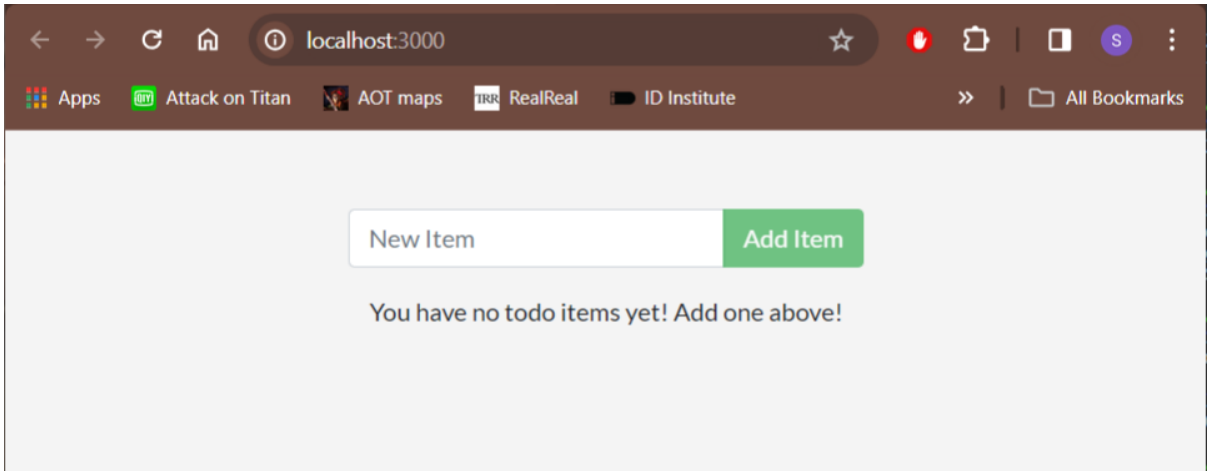
MINGW64:/c/Users/Astheerah/getting-started-app

```
    if (items === null) return 'Loading...';

    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}
                />
            ))}
        </React.Fragment>
    );
}

function AddItemForm({ onNewItem }) {
    const { Form, InputGroup, Button } = ReactBootstrap;

    const [newItem, setNewItem] = React.useState('');
    const [submitting, setSubmitting] = React.useState(false);

    const submitNewItem = e => {
        e.preventDefault();
        setSubmitting(true);
        fetch('/items', {
            method: 'POST',
            body: JSON.stringify({ name: newItem }),
            headers: { 'Content-Type': 'application/json' },
        })
            .then(r => r.json())
            .then(item => {
                onNewItem(item);
                setSubmitting(false);
                setNewItem('');
            });
    };

    return (
        <Form onSubmit={submitNewItem}>
            <InputGroup className="mb-3">
                <Form.Control
                    value={newItem}
                    onChange={e => setNewItem(e.target.value)}
                    type="text"
                    placeholder="New Item"
                    aria-describedby="basic-addon1"
                />
                <InputGroup.Append>
                    <Button
                        type="submit"
                        variant="success"
                        disabled={!newItem.length}
                        className={submitting ? 'disabled' : ''}
                    >
src/static/js/app.js[+] [dos] (19:39 08/02/2024)                    56,86 40%
-- INSERT --
```

Remove a container using the CLI and start the updated app container

Updating Docker images essentially involves deleting the previous image and launching fresh containers using the updated version of the image. Maintaining a proactive approach and consistently updating your Docker images guarantees that your apps flourish in a constantly evolving setting.

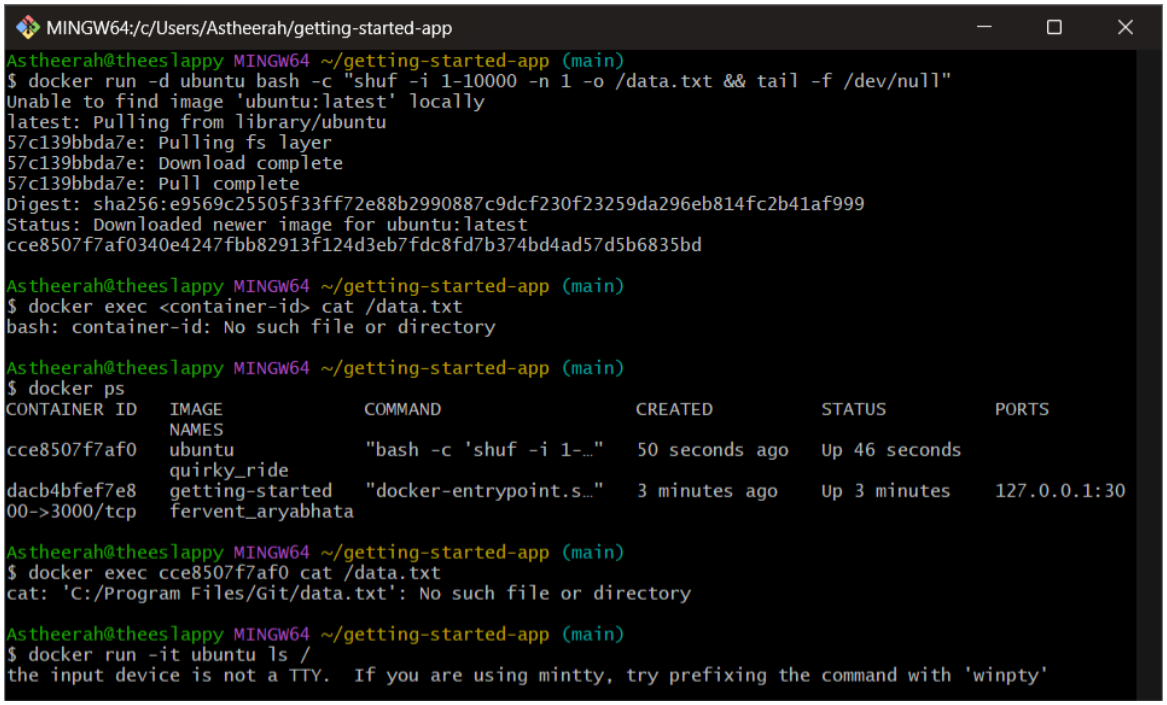
```
Astheerah@theeslappy MINGW64 ~/getting-started-app (main)
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
          NAMES
08aae997bd4b   6cf58e32e5e8   "docker-entrypoint.s…"   15 minutes ago   Up 15 minutes   127.0.0.1:3000-
>3000/tcp   quirky_morse

Astheerah@theeslappy MINGW64 ~/getting-started-app (main)
$ docker stop <the-container-id^C

Astheerah@theeslappy MINGW64 ~/getting-started-app (main)
$ docker stop 08aae997bd4b
08aae997bd4b

Astheerah@theeslappy MINGW64 ~/getting-started-app (main)
$ docker rm 08aae997bd4b
08aae997bd4b

Astheerah@theeslappy MINGW64 ~/getting-started-app (main)
$ docker run -dp 127.0.0.1:3000:3000 getting-started
dacb4bfef7e8b31ad6ca11e51b202c017b03fc5da3108b3d801108e7a803283c

Astheerah@theeslappy MINGW64 ~/getting-started-app (main)
$
```

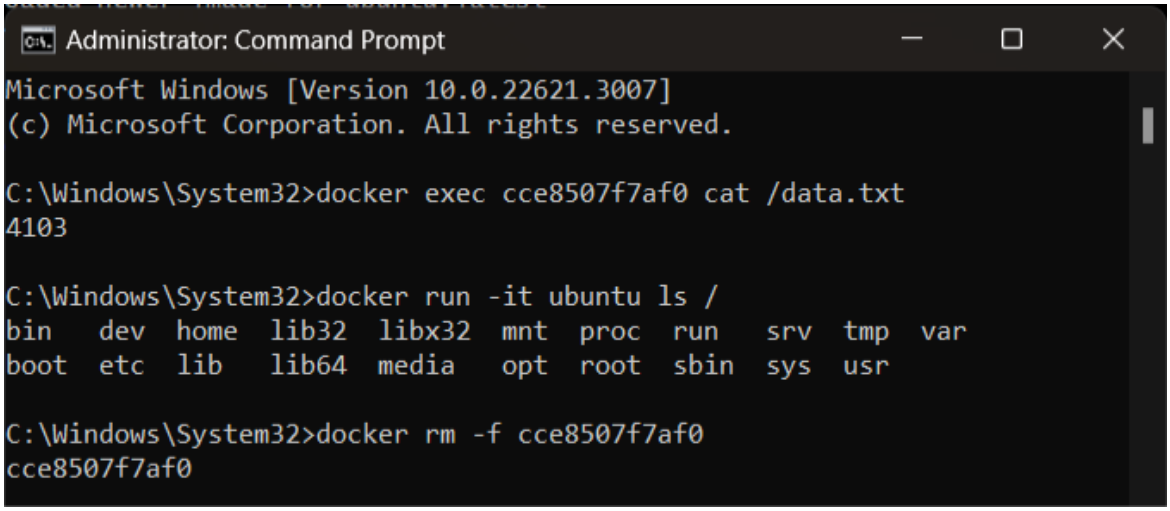


Part 5: Persist the DB

A container's filesystem is made up of the different layers from the image when it is operating. Also, each container has a dedicated "scratch space" for adding, deleting, and updating files. Even if two containers use the same image, none of the changes will be visible to the other.

The container's filesystem





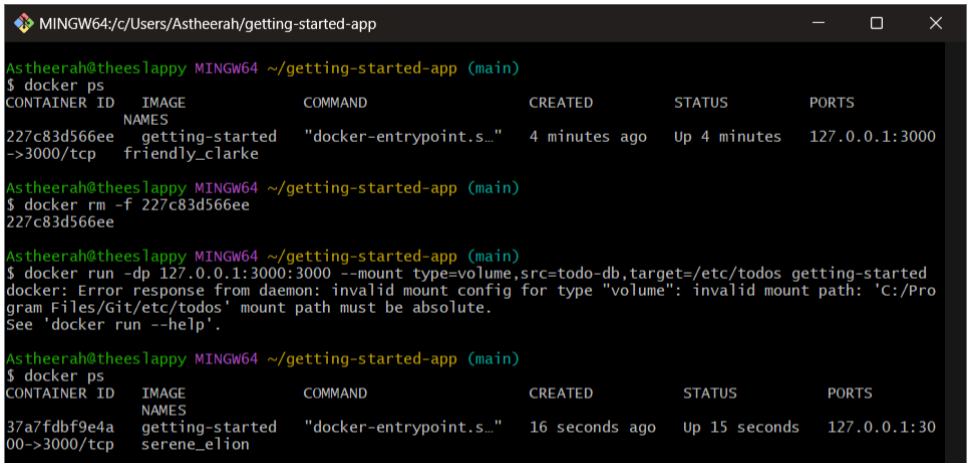
Container volumes

Docker isolates all changes made to a container, even if it has the ability to create, update, and delete files. These changes are lost when the container is removed. You may adjust all of this using volumes.

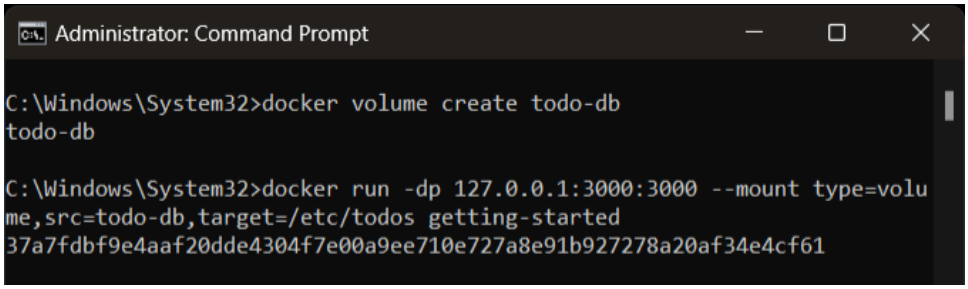
Persist the todo data

Create a volume and start the container

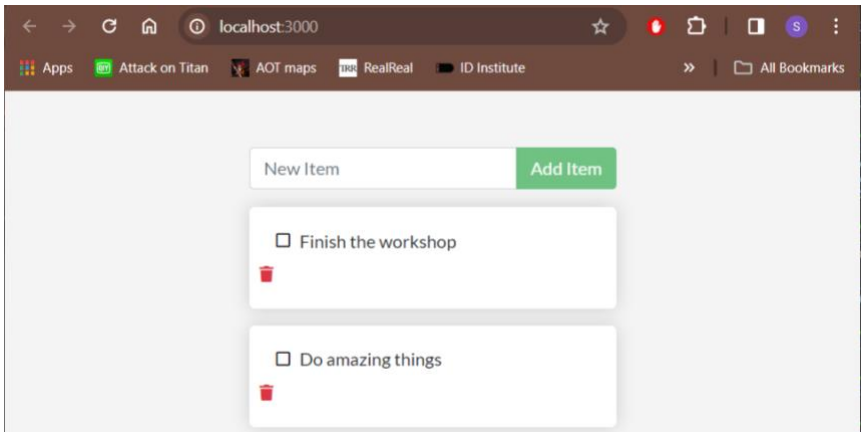
Since the todo app container is still operating without using the persistent volume, stop and delete it again with docker rm -f <id>.

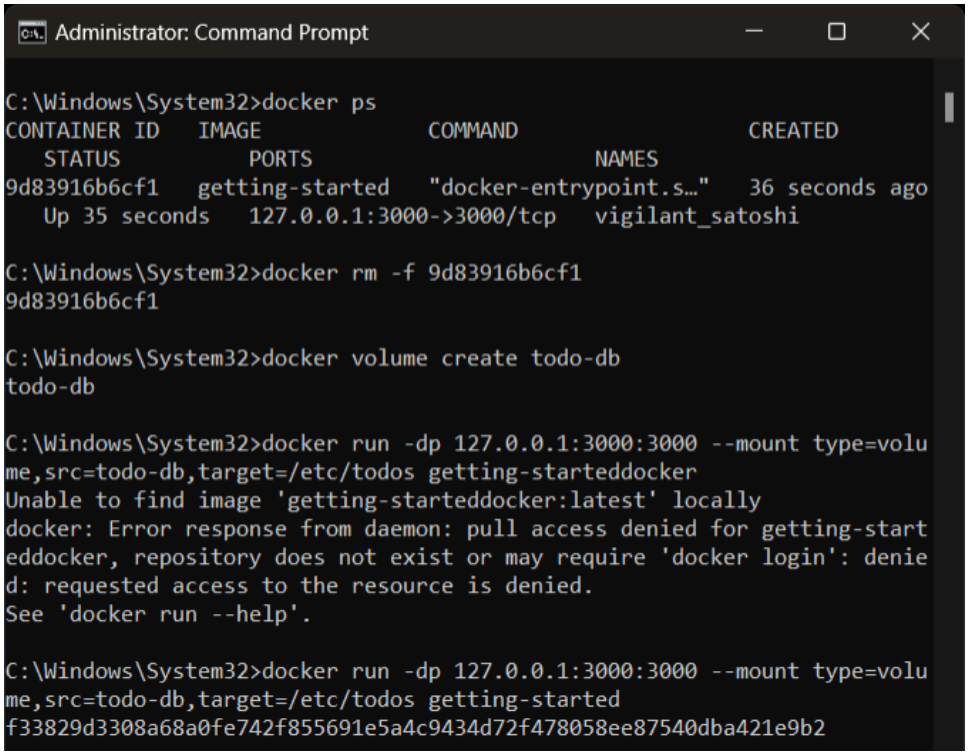


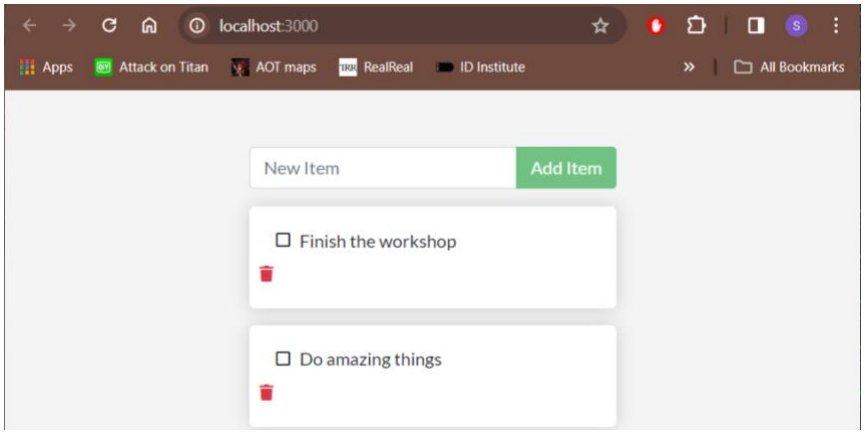
Create a volume by using the docker volume create command.



Once the container starts up, open the app and add a few items to your todo list.



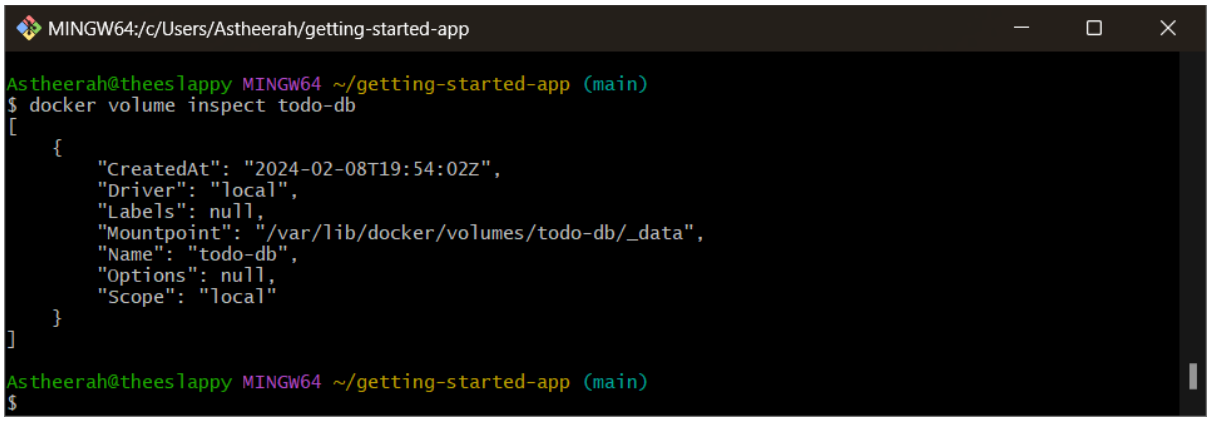




Dive into the volume

Data that is accessed by Docker containers is stored persistently in a Docker volume. When the container is removed, this data is still there because it is saved outside of the containers.

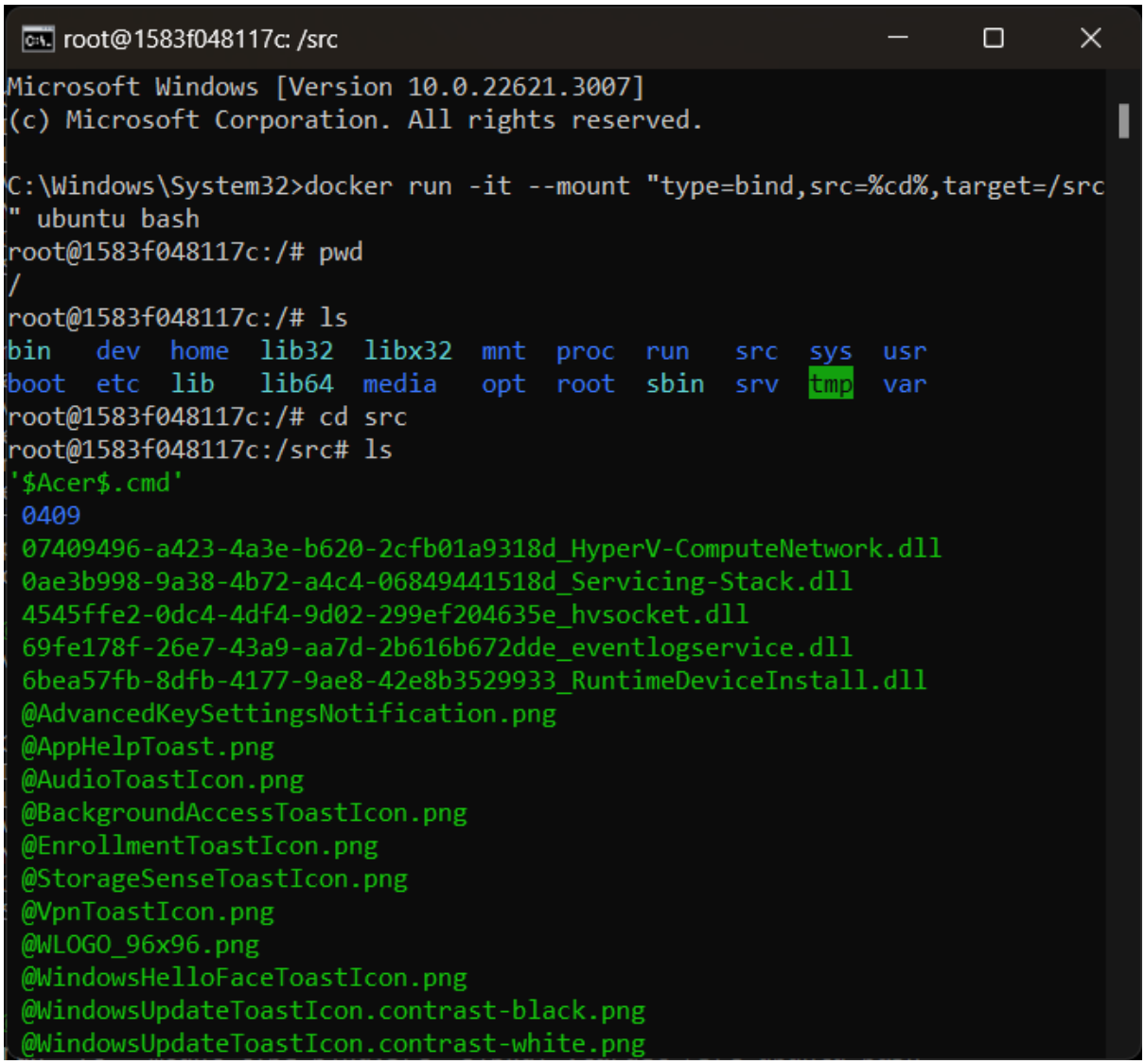
This enables you to keep data created and utilised by your containers around, even in the event that the container is removed.

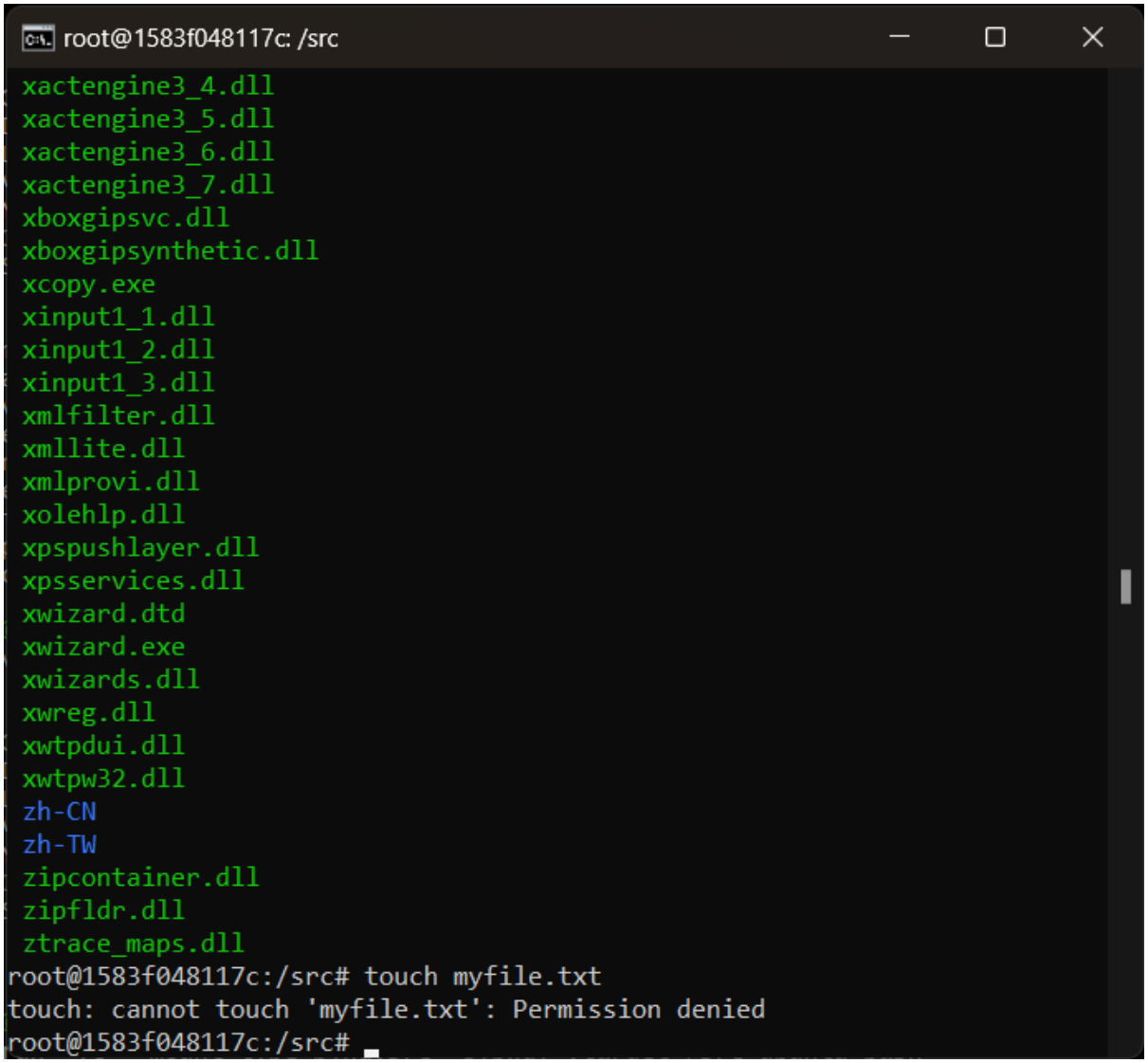


Part 6: Use bind mounts

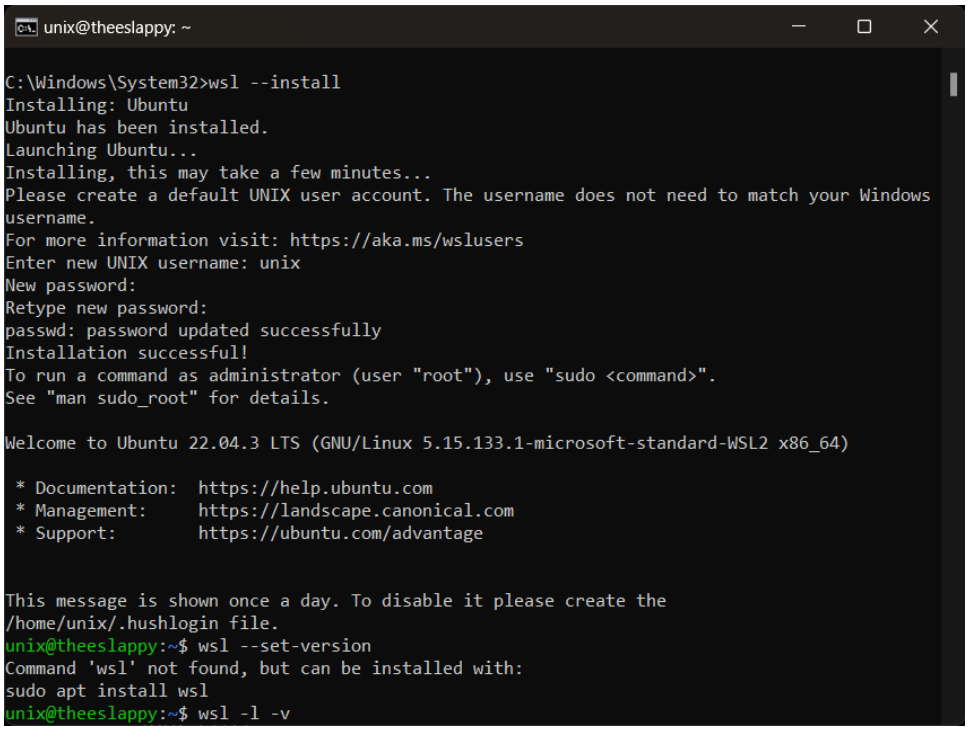
Trying out bind mounts

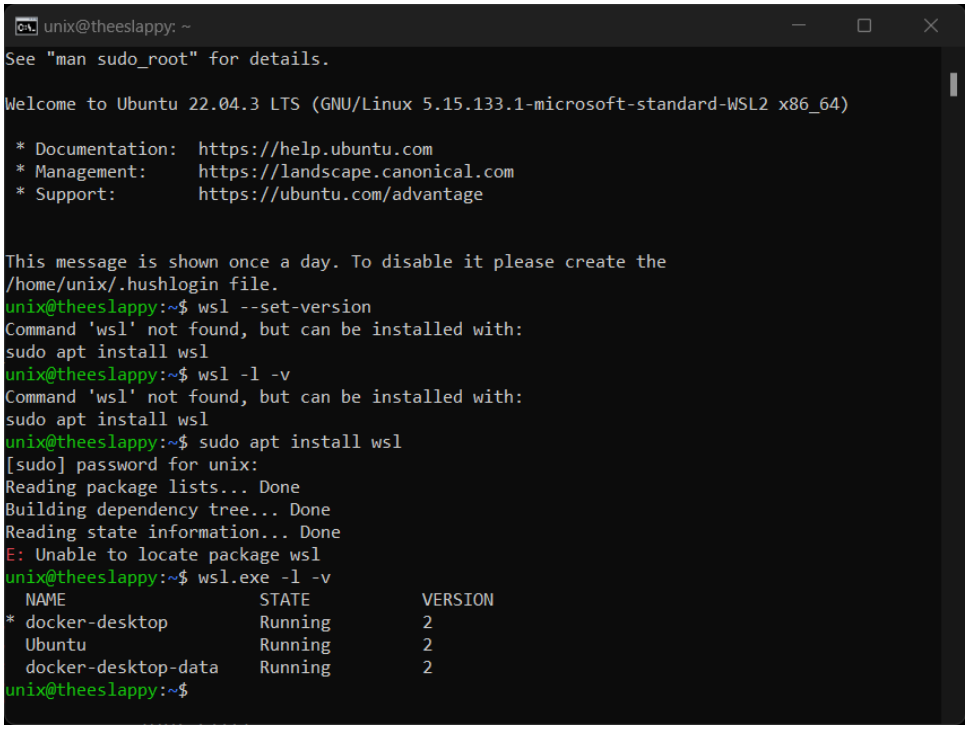
I had an issue in this part where ‘touch’ command doesn’t have permissions.



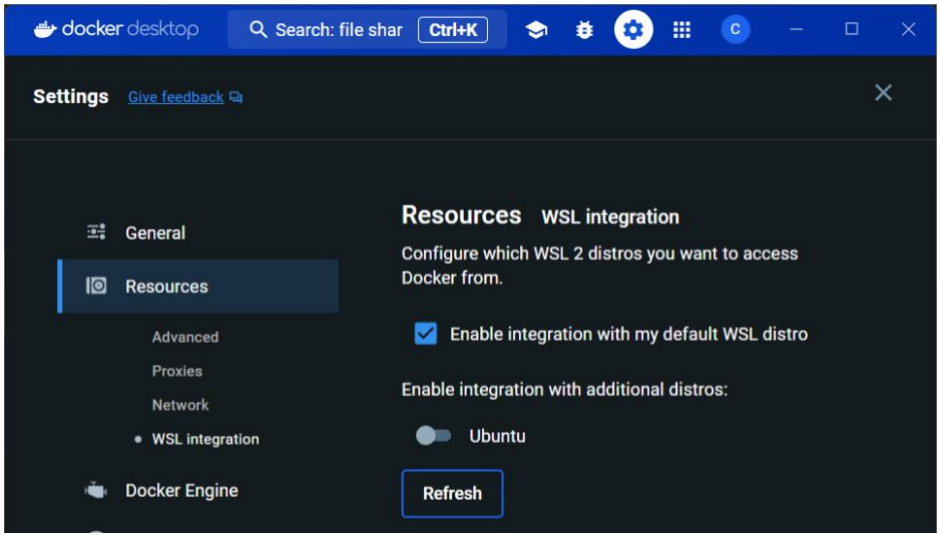

root@1583f048117c: /src
xactengine3_4.dll
xactengine3_5.dll
xactengine3_6.dll
xactengine3_7.dll
xboxgipsvc.dll
xboxgipsynthetic.dll
xcopy.exe
xinput1_1.dll
xinput1_2.dll
xinput1_3.dll
xmlfilter.dll
xmllite.dll
xmlprovi.dll
xolehlp.dll
xpspushlayer.dll
xpsservices.dll
xwizard.dtd
xwizard.exe
xwizards.dll
xwreg.dll
xwtpdui.dll
xwtpw32.dll
zh-CN
zh-TW
zipcontainer.dll
zipfldr.dll
ztrace_maps.dll
root@1583f048117c:/src# touch myfile.txt
touch: cannot touch 'myfile.txt': Permission denied
root@1583f048117c:/src#

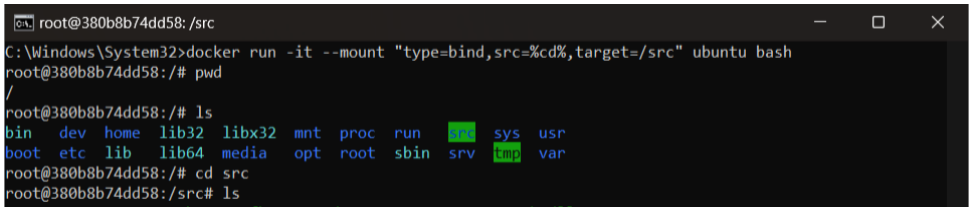
I figured out that I didn’t have WSL in my Docker Desktop, so I had to install it manually.

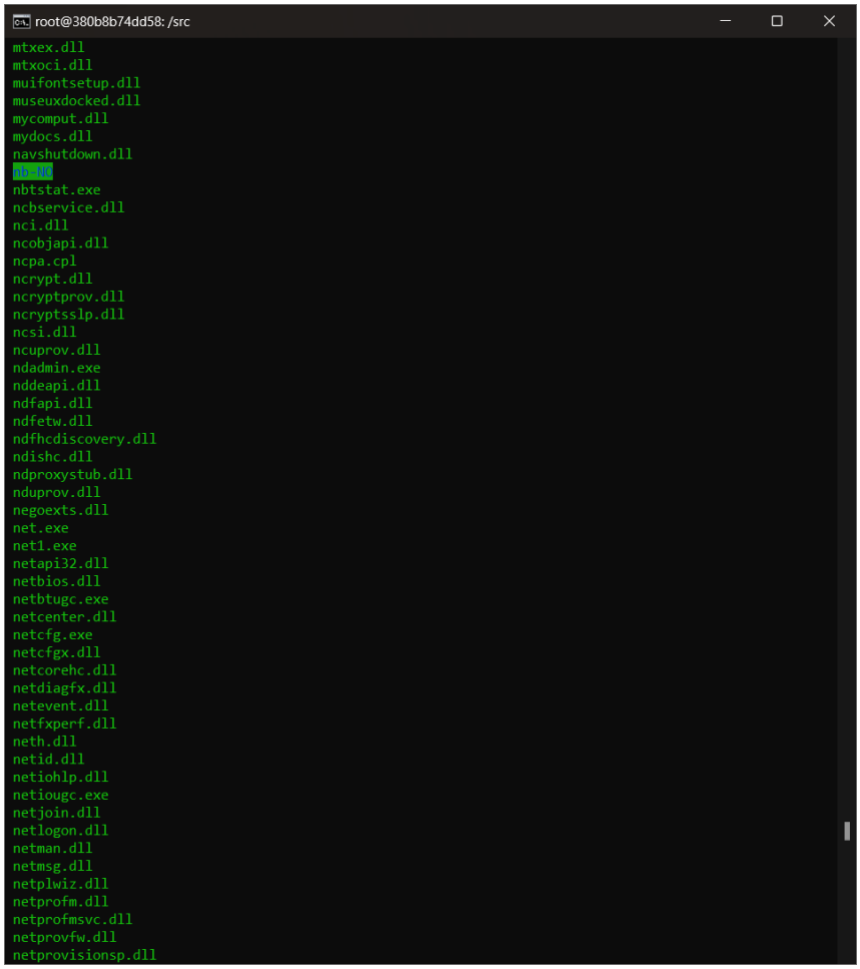




Here's the outcome:

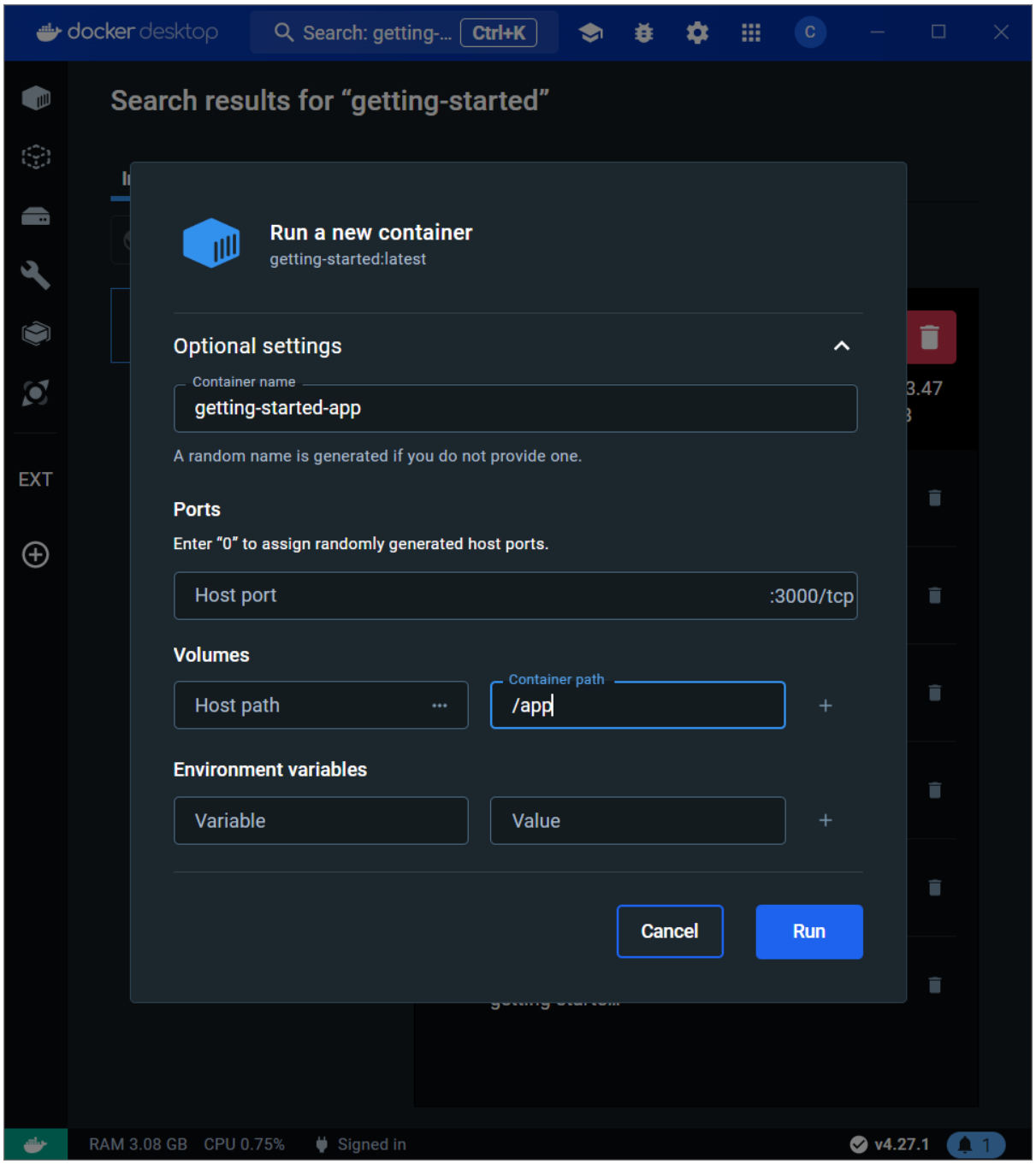





root@380b8b74dd58: /src
mtxex.dll
mtxoci.dll
muifontsetup.dll
museuxdocked.dll
mycomput.dll
mydocs.dll
navshutdown.dll
nb-NO
nbtstat.exe
ncbservice.dll
nci.dll
ncobjapi.dll
ncpa.cpl
ncrypt.dll
ncryptprov.dll
ncryptsslp.dll
ncsi.dll
ncuprov.dll
ndadmin.exe
nddeapi.dll
ndfapi.dll
ndfetw.dll
ndfhcdiscovery.dll
ndishc.dll
ndproxystub.dll
nduprov.dll
negoexts.dll
net.exe
net1.exe
netapi32.dll
netbios.dll
netbtugc.exe
netcenter.dll
netcfg.exe
netcfgx.dll
netcorehc.dll
netdiagfx.dll
netevent.dll
netfxperf.dll
neth.dll
netid.dll
netiohlp.dll
netiougc.exe
netjoin.dll
netlogon.dll
netman.dll
netmsg.dll
netplwiz.dll
netprofm.dll
netprofmsvc.dll
netprovfw.dll
netprovisionsp.dll

Part 7: Multi container apps

I didn't get to do this part and onwards.



Administrator: Command Prompt

```
C:\Windows\System32>docker run -dp 127.0.0.1:3000:3000
"docker run" requires at least 1 argument.
See 'docker run --help'.

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

Create and run a new container from an image

C:\Windows\System32>docker run -d ^
More?     --network todo-app --network-alias mysql ^
More?     -v todo-mysql-data:/var/lib/mysql ^
More?     -e MYSQL_ROOT_PASSWORD=secret ^
More?     -e MYSQL_DATABASE=todos ^
More?     mysql:8.0
69851f2b855b3caf22962015f45af95fa04f3277ed08f8503db7ff9532d8ada7

C:\Windows\System32>docker exec -it 69851f2b855b3caf22962015f45af95fa04f3277ed08f8503db7ff9532d8ada7 mys
ql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

C:\Windows\System32>docker exec -it 69851f2b855b3caf22962015f45af95fa04f3277ed08f8503db7ff9532d8ada7 mys
ql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

C:\Windows\System32>docker exec -it 69851f2b855b3caf22962015f45af95fa04f3277ed08f8503db7ff9532d8ada7 mys
ql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

C:\Windows\System32>docker exec -it 69851f2b855b3caf22962015f45af95fa04f3277ed08f8503db7ff9532d8ada7 mys
ql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

C:\Windows\System32>ROOT
'ROOT' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\System32>root
'root' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\System32>docker exec -it 69851f2b855b3caf22962015f45af95fa04f3277ed08f8503db7ff9532d8ada7 mys
ql -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)

C:\Windows\System32>sudo docker exec -it 69851f2b855b3caf22962015f45af95fa04f3277ed08f8503db7ff9532d8ada
7 mysql -u root
'sudo' is not recognized as an internal or external command,
```