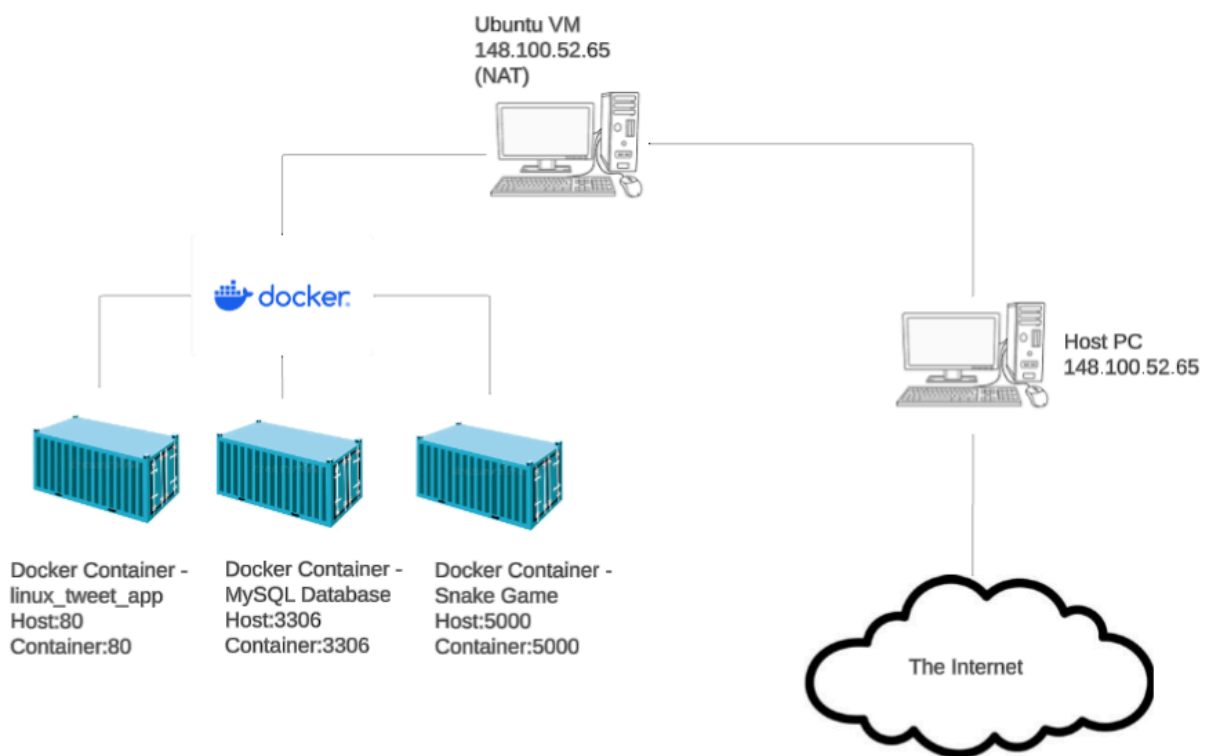


Lab 4

Description: In this lab we will download Docker and deploy a couple of applications on it to display containerization.

Topology:



Syntax:

Command	Description
Docker container ls --all	Shows all docker containers
Curl http://localhost	Sends a HTTP GET request, testing the web server

Docker compose up -d	Starts all services that are defined in docker-compose.yml
Docker run -it ubuntu /bin/bash	Starts a docker container interactively

Verification:

Task One:

```
eric-ub@Eric-UB:~/dockerlab04/linux_tweet_app$ ls -l
total 60
-rw-rw-r-- 1 eric-ub eric-ub  207 Apr 10 22:10 CONTRIBUTING.md
-rw-rw-r-- 1 eric-ub eric-ub  147 Apr 10 22:10 Dockerfile
-rw-rw-r-- 1 eric-ub eric-ub 1595 Apr 10 22:10 index.html
-rw-rw-r-- 1 eric-ub eric-ub 1673 Apr 10 22:10 index-new.html
-rw-rw-r-- 1 eric-ub eric-ub 1540 Apr 10 22:10 index-original.html
-rw-rw-r-- 1 eric-ub eric-ub 11357 Apr 10 22:10 LICENSE
-rw-rw-r-- 1 eric-ub eric-ub 22365 Apr 10 22:10 linux.png
-rw-rw-r-- 1 eric-ub eric-ub   572 Apr 10 22:10 README.md
eric-ub@Eric-UB:~/dockerlab04/linux_tweet_app$
```

This screenshot shows the directory listings for the linux_tweet_app that were cloned from the Docker Github repository.

Task Two:

```
eric-ub@Eric-UB:~/dockerlab04/linux_tweet_app$ sudo docker container run alpine hostname
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
f18232174bc9: Pull complete
Digest: sha256:a8560b36e8b8210634f77d9f7f9efd7ffa463e380b75e2e74aff4511df3ef88c
Status: Downloaded newer image for alpine:latest
4863386e9635
```

This screenshot shows the alpine Docker image being run and being pulled from docker hub. This container was run in non interactive mode since there was no terminal session opened, no user input needed and it ran the command then exited.

Task Three:

```
eric-ub@Eric-UB:~/dockerlab04/linux_tweet_app$ sudo docker run -it ubuntu /bin/bash

Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
2726e237d1a3: Pull complete
Digest: sha256:1e622c5f073b4f6bfad6632f2616c7f59ef256e96fe78bf6a595d1dc4376ac02
Status: Downloaded newer image for ubuntu:latest

root@d627e6704daa:/#
root@d627e6704daa:/# cat /etc/issue

Ubuntu 24.04.2 LTS \n \l

root@d627e6704daa:/#
root@d627e6704daa:/#
```

This screenshot shows an interactive docker session. I used the command `cat /etc/issue` to show the Linux distribution and version (24.04.2 LTS). This proves that it is running a different OS instance than my host VM.

Task Four:

```
eric-ub@Eric-UB:~/dockerlab04/linux_tweet_app$ sudo docker container run \
> --detach \
> --name msis603db \
> -e MYSQL_ROOT_PASSWORD=msis603pw \
> mysql:latest
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
cea172a6e83b: Pull complete
daac2c594bdd: Pull complete
cb8acbf2440c: Pull complete
fae51f7de1fb: Pull complete
b2ead3e96e6b: Pull complete
769c3ac51f88: Pull complete
79f239a40e62: Pull complete
c11056354384: Pull complete
49978e7ccddf: Pull complete
548990e33276: Pull complete
Digest: sha256:0596fa224cdf3b3355ce3ddbdf7ce77be27ec9e51841dfc5d2e1c8b81eea69d2
Status: Downloaded newer image for mysql:latest
cb1634e5551a90a8b7a5ab82d799b0c2a1c0a40401e2870c5e73a5d86e8dc483
eric-ub@Eric-UB:~/dockerlab04/linux_tweet_app$
```

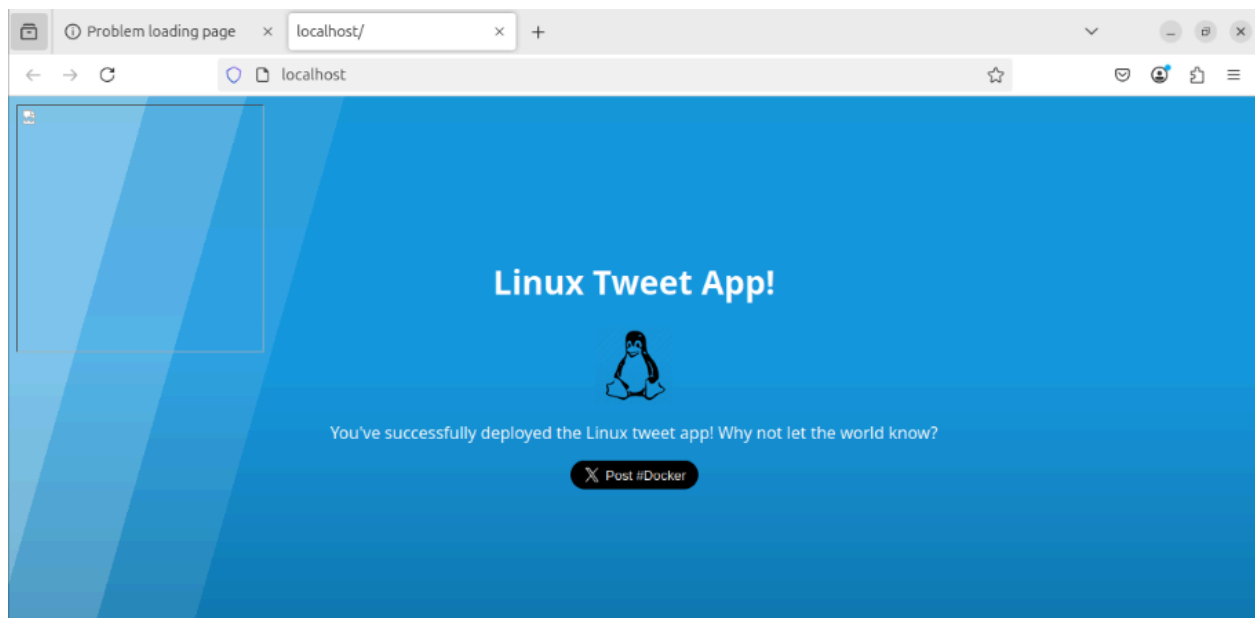
This screenshot shows the MySQL server container launching in a detached mode as a background process using the command `sudo docker container run --detach`.

Task Five:

```
eric-ub@Eric-UB:~/dockerlab04/linux_tweet_app$ telnet 127.0.0.1 3306
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
I
9.2.0
```

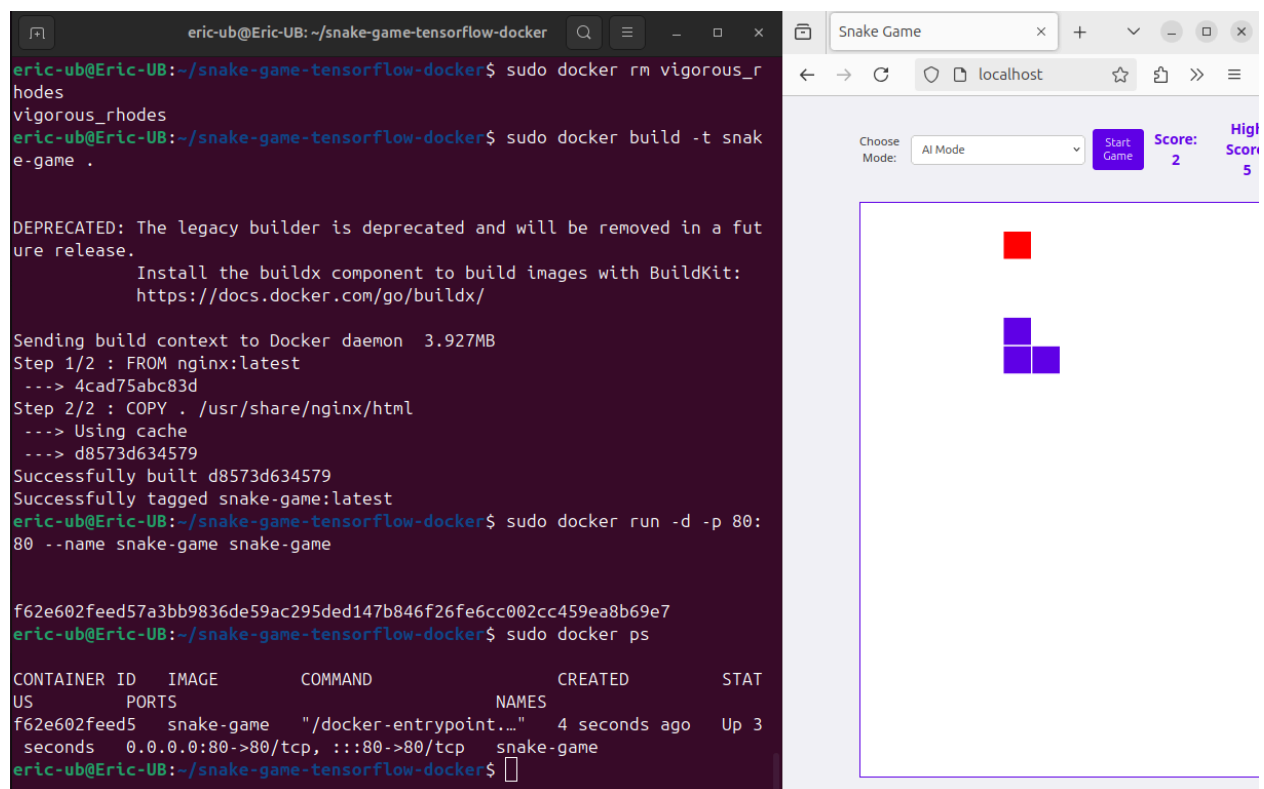
This screenshot shows my host VM trying to telnet into the Docker container. There is a successful connection message and the MySQL version number is shown proving that the database docker container is accessible on TCP port 3306.

Task Six:



This screenshot shows the Linux Tweet App being served through a docker container and being accessed from a browser on localhost. This proves that the containerized web server is successfully running on TCP port 80.

Task Seven:



This screenshot shows that I was able to run the AI snake game in my web browser on localhost from a docker container. I also showed the terminal output showing that the container is serving the content on port 80.

Puzzler Task:

```
GNU nano 7.2 my-seccomp.json
{
  "defaultAction": "SCMP_ACT_ALLOW",
  "syscalls": [
    {
      "names": ["mkdir"],
      "action": "SCMP_ACT_ERRNO"
    }
  ]
}
```

```
root@3d4c1779cf05:/#
root@3d4c1779cf05:/# mkdir test
mkdir: cannot create directory 'test': Operation not permitted
root@3d4c1779cf05:/#
root@3d4c1779cf05:/# exit
```

The first screenshot shows my seccomp json profile that is being used to block mkdir syscalls. The second screenshot shows that it works and the container denies any directory creation because of the seccomp profile.

Conclusion:

I really enjoyed learning docker in this lab. I had heard a lot about it but never really tried it out myself. It took a minute to get the hang of things at first but once I got everything working, it was fun spinning up each container and accessing them in the browser.