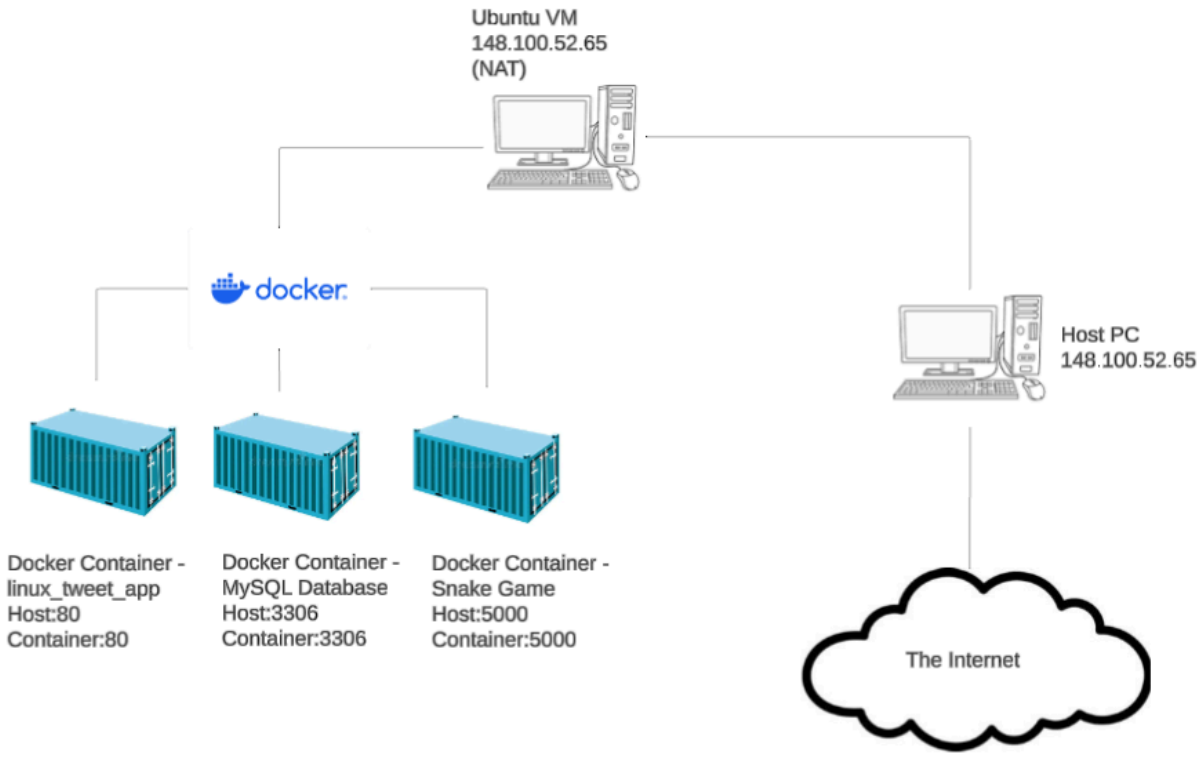
Eric Guzman
Professor Cannistra

4/9/2025
Network Virtualization

Lab 4

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

Topology:



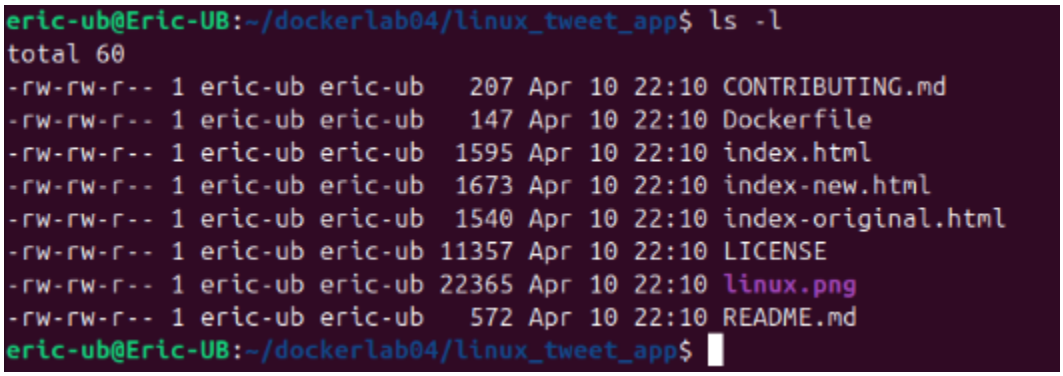
Syntax:

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

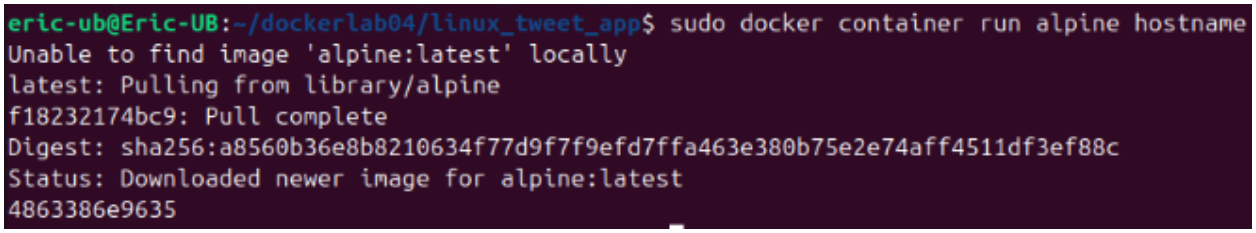
Verification:

Task One:



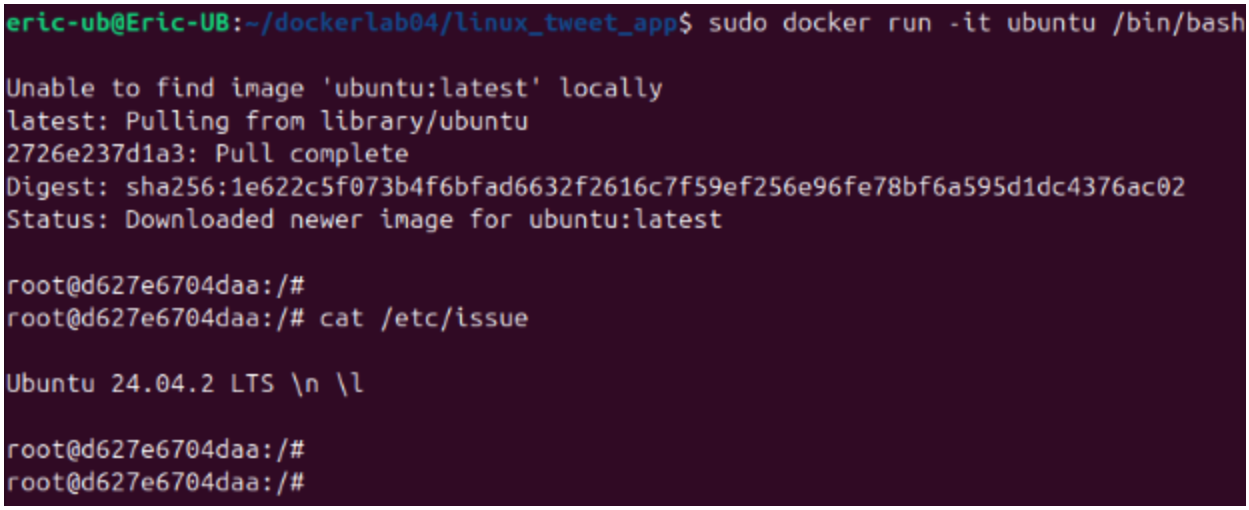
This screenshot shows the directory listings for the linux_tweet_app that were cloned from the Docker Github repository.

Task Two:



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:



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:

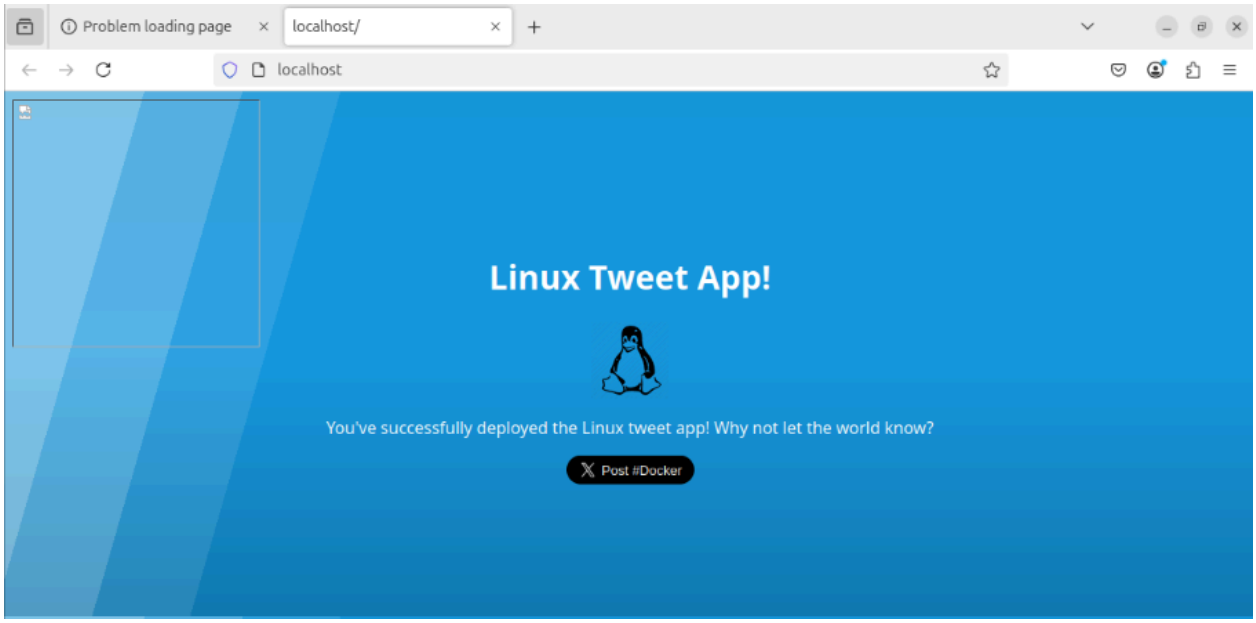
This screenshot shows the MySQL server container launching in a detached mode as a background process using the command sudo docker container run --detach.

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
```

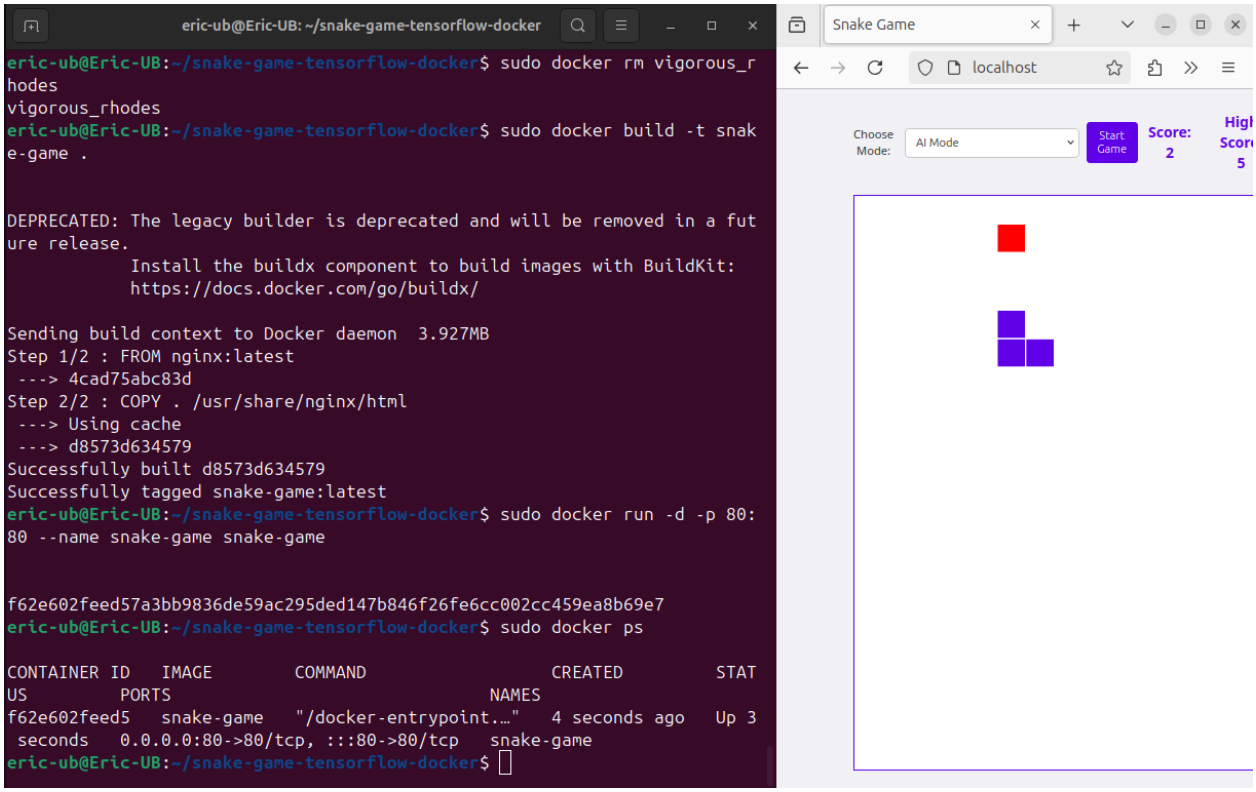
This screenshot shows my host VM trying to telnet into the Docker container. There is a successful connection message and the MySQL version number is shown proving that the database docker container is accessible on TCP port 3306.

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.