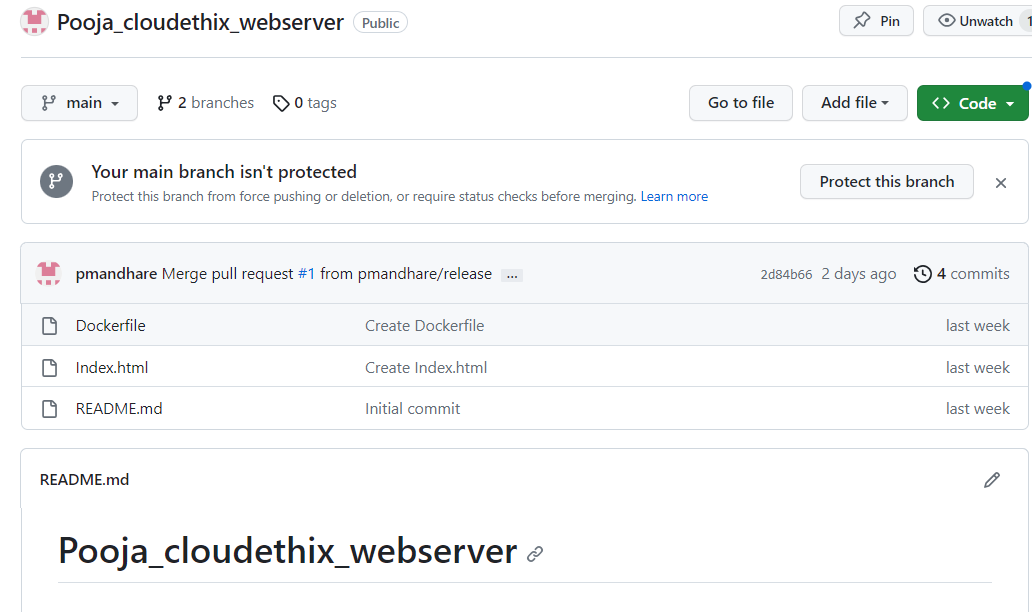
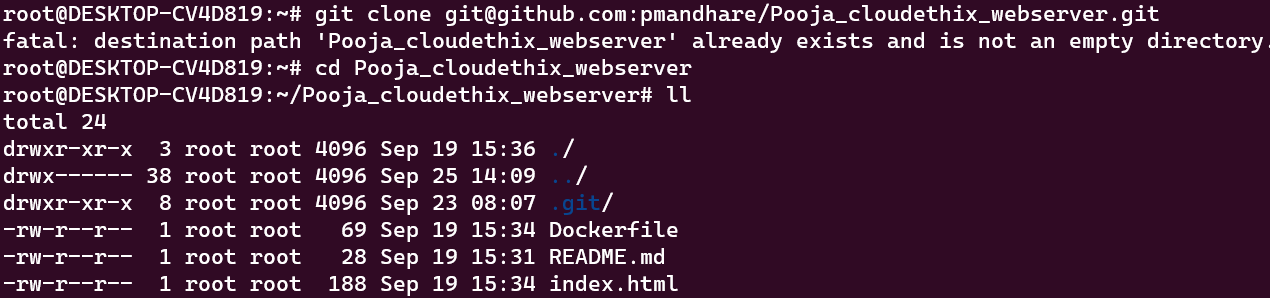
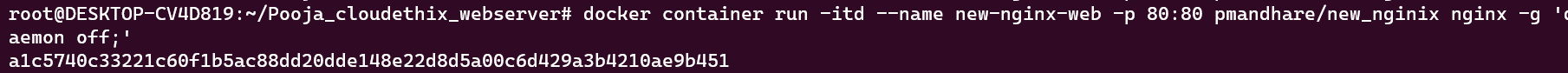
Docker Images and Containers Assignments

01: Basic Image Creation  
   - Create a Docker image for a simple web application using a Dockerfile. Ensure it runs on port 80.

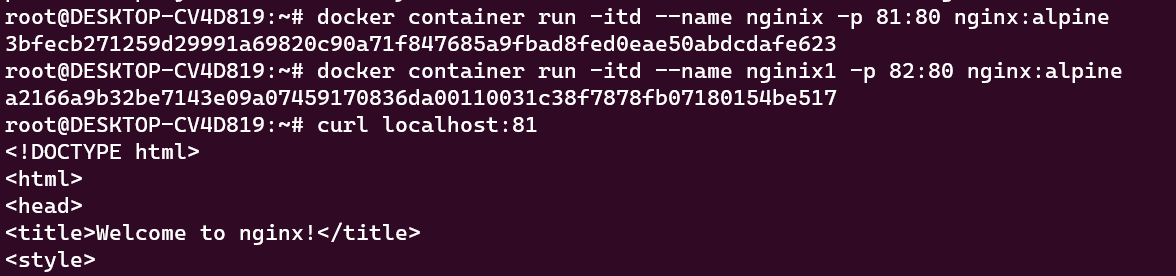


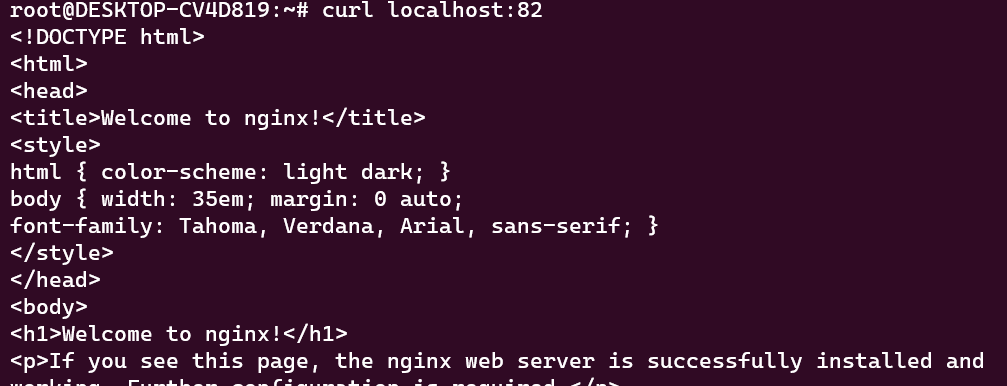


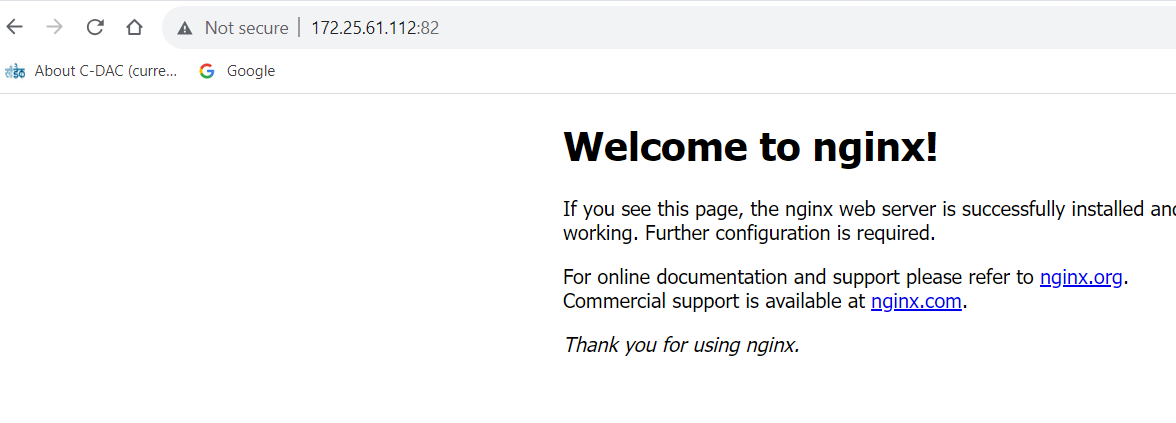


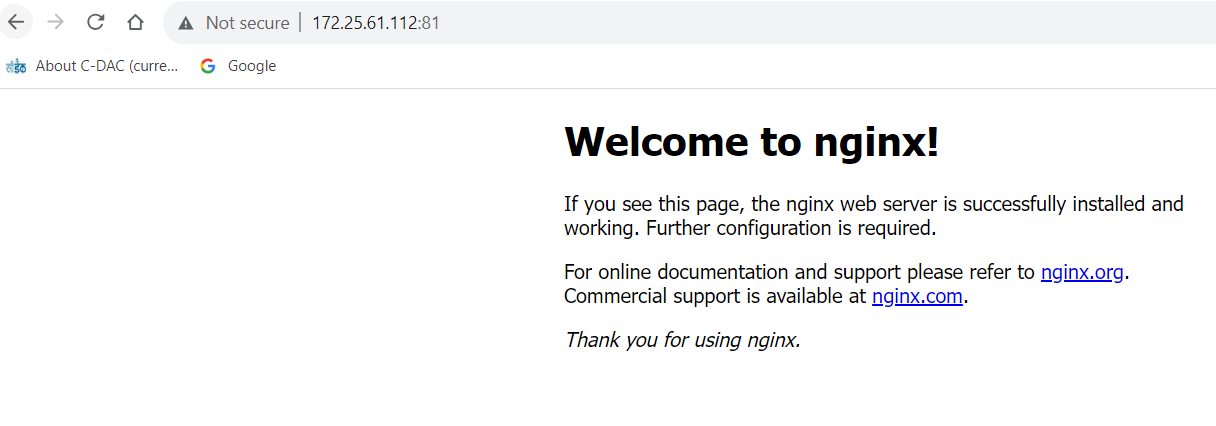


02: Running Containers  
   - Run two containers from the same image concurrently. Configure each container to listen on different ports. Test their accessibility.





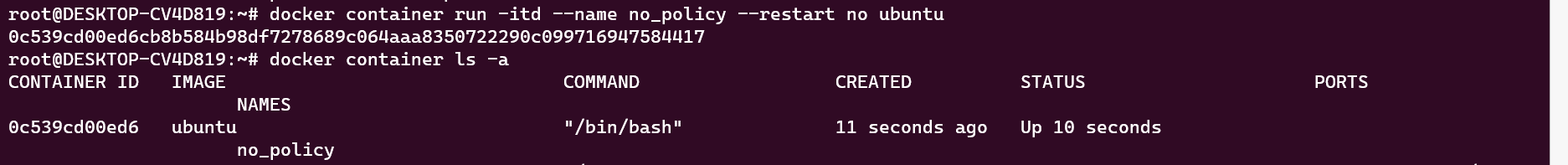


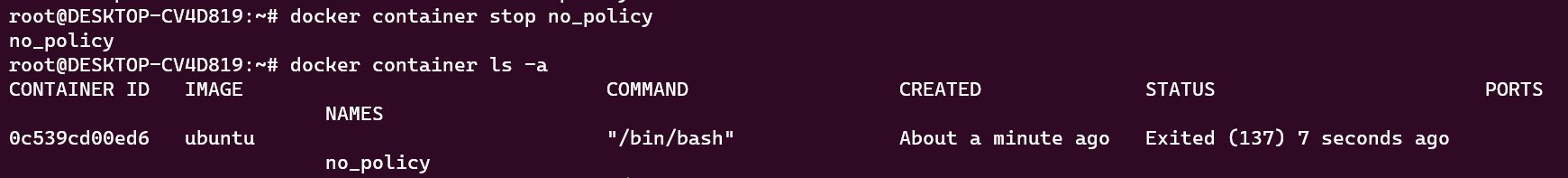


03: Docker Compose  
   - Use Docker Compose to define and run a multi-container application. Include at least two services and establish communication between them.

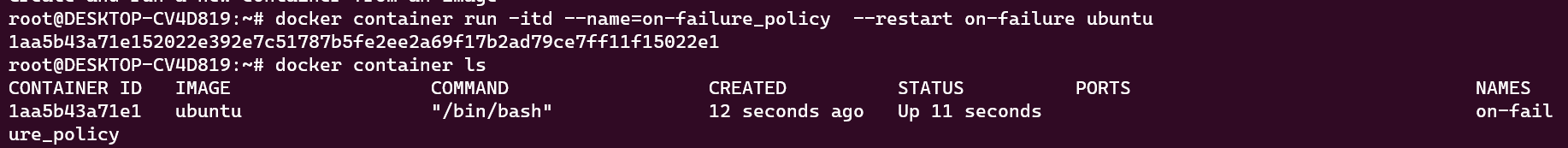
04: Container Management (Restart Policies)   
   - Run some docker containers and test all the restart policies.

No policy



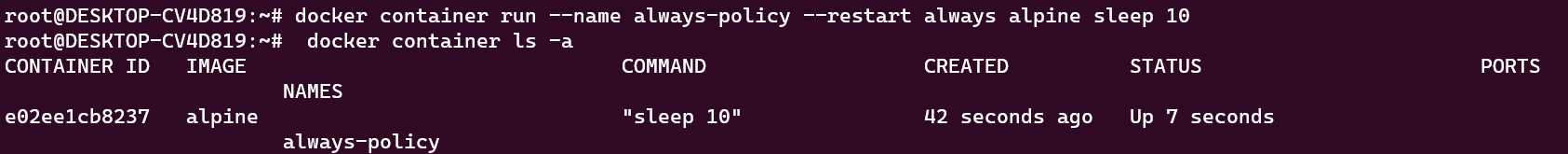


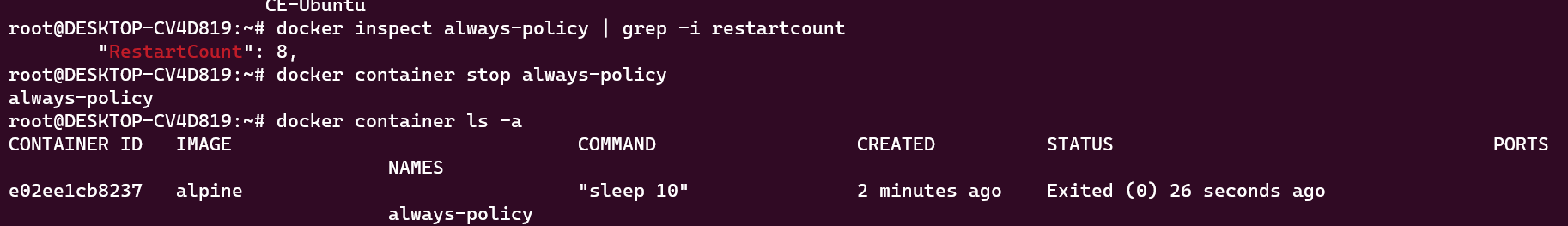
on-failure policy

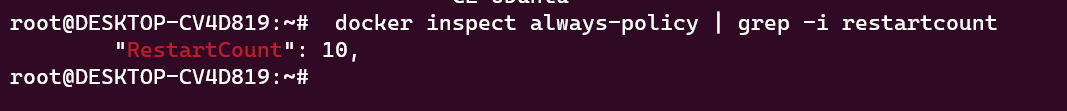




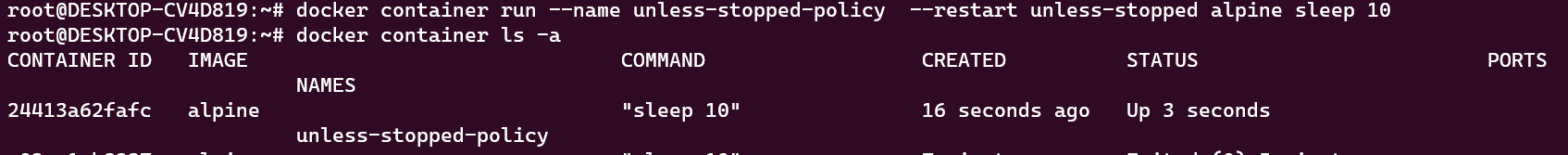
always restart policy

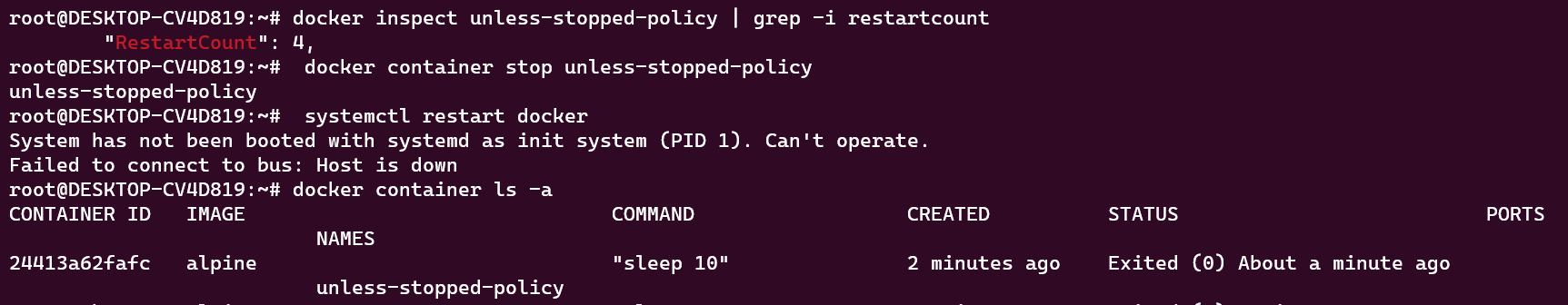




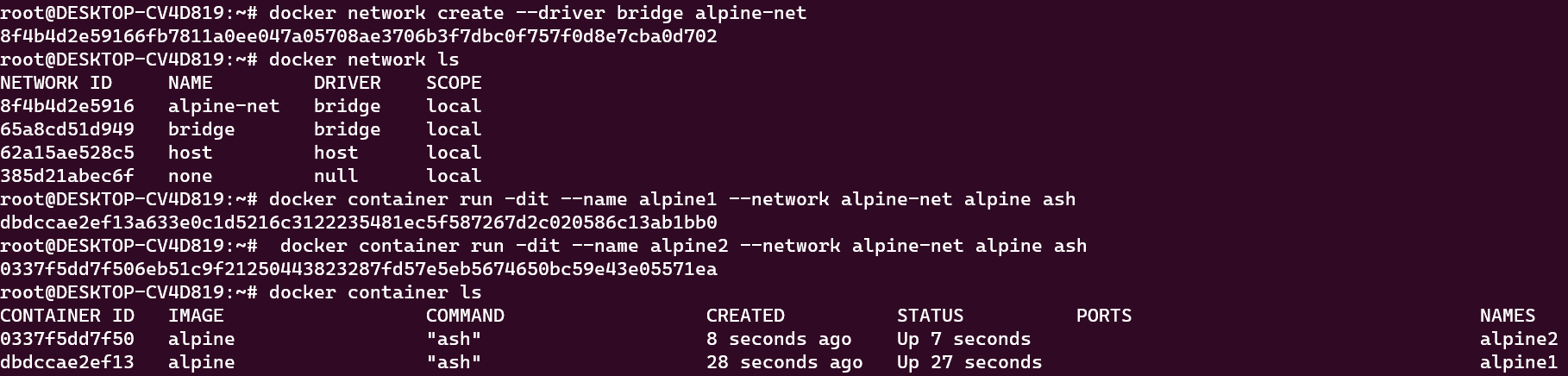


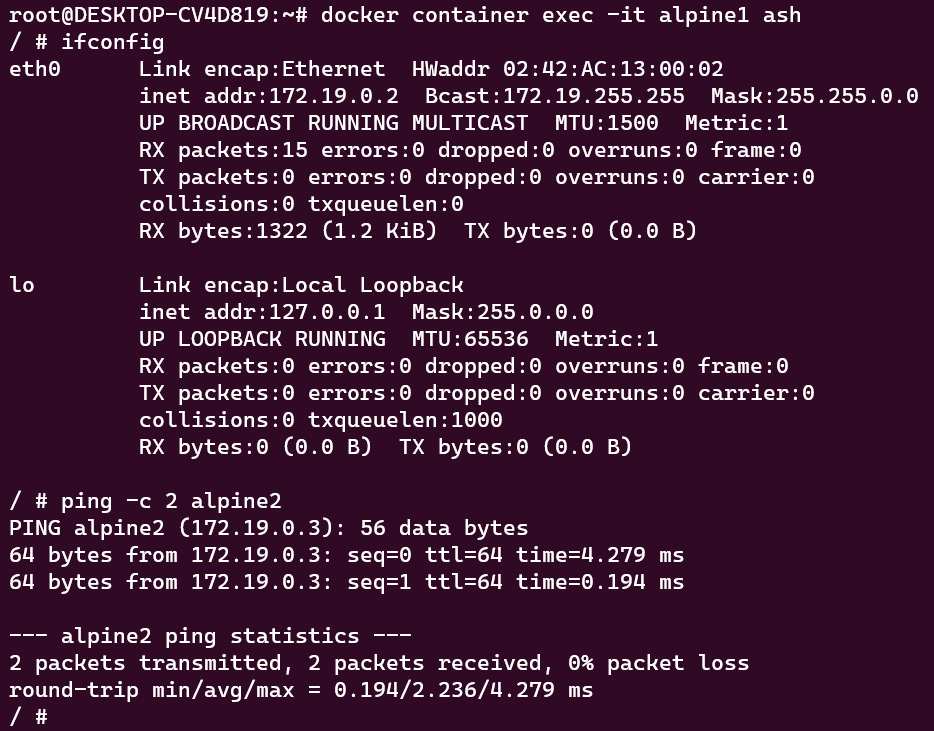
unless-stopped policy



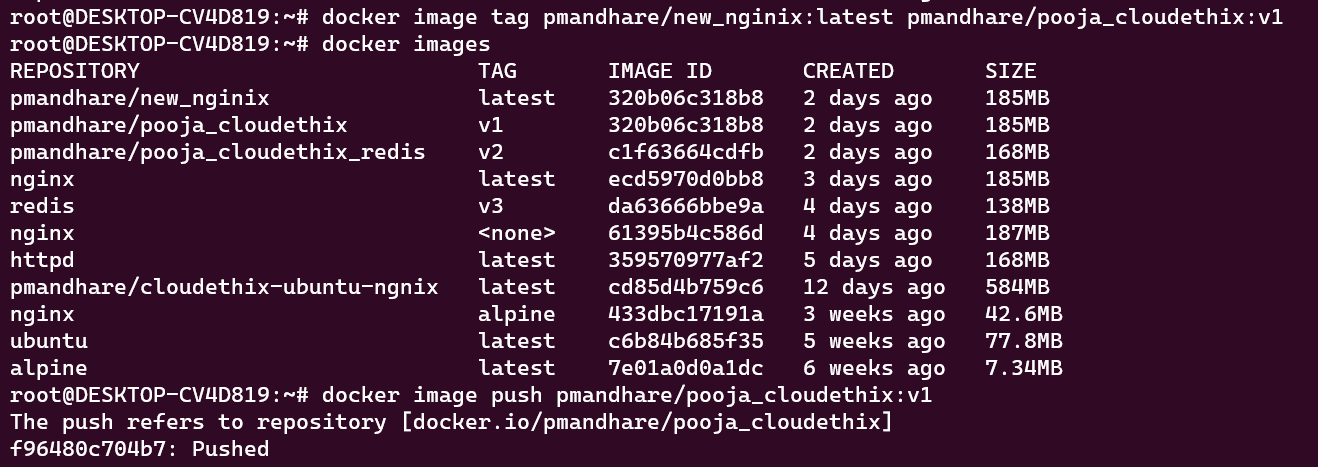


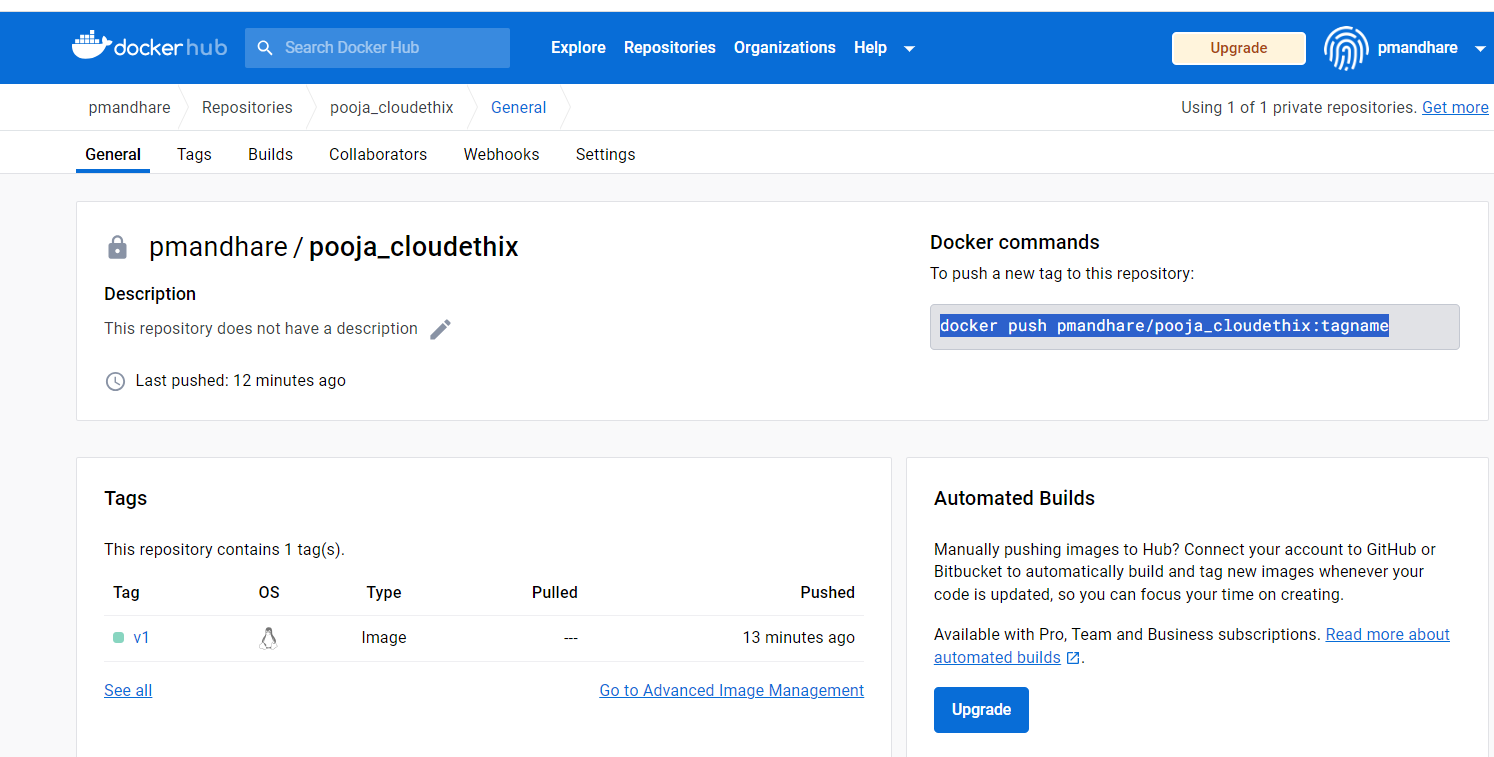
05: Docker Networks  
   - Create a custom Docker network and launch two containers within it. Verify that they can communicate with each other using container names.

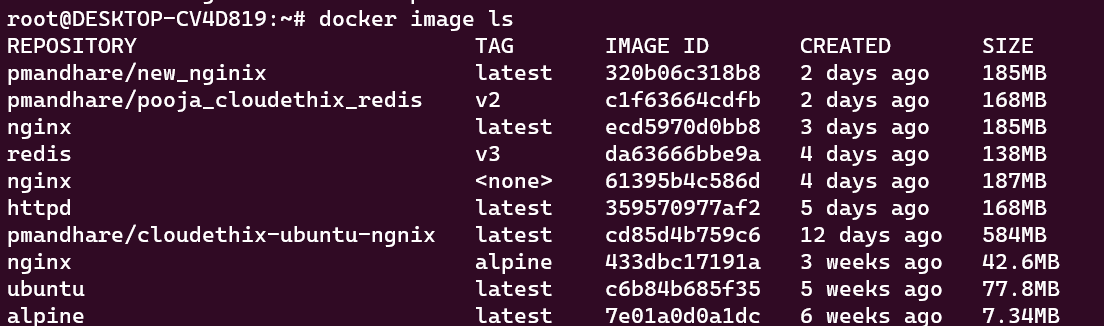


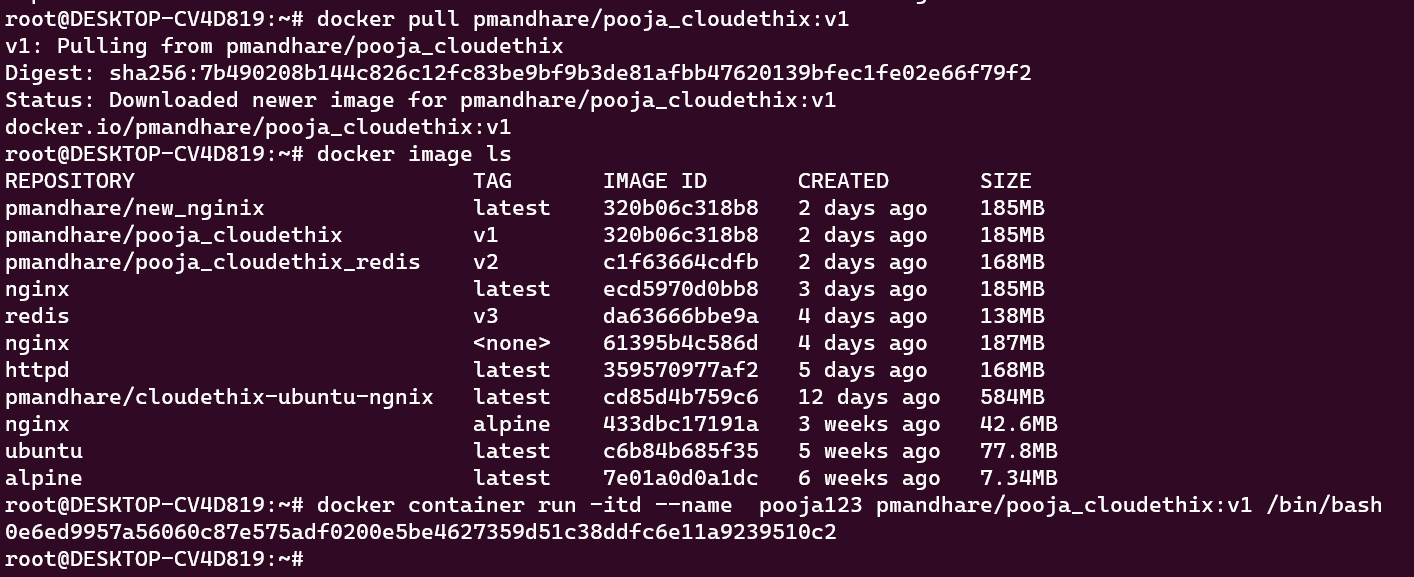


07: Docker Registry  
   - Set up a private Docker registry and push a custom Docker image to it. Then, pull the image from the registry and run containers.

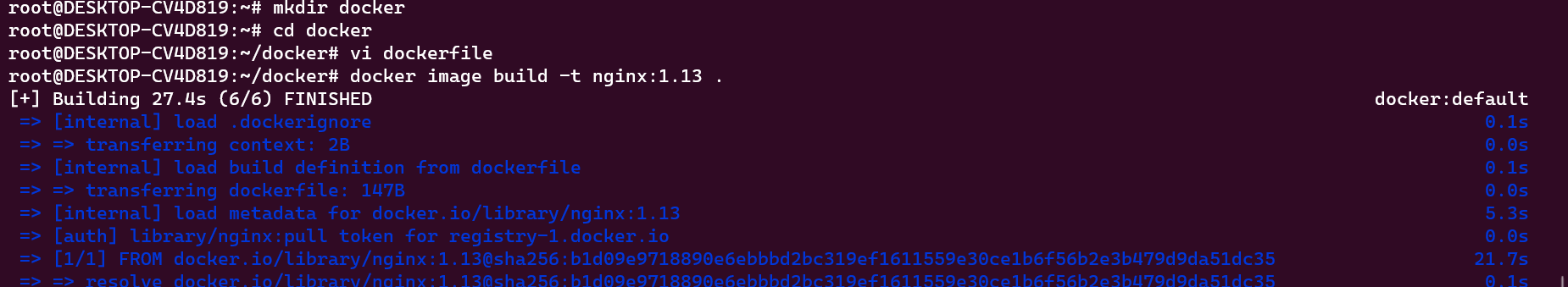


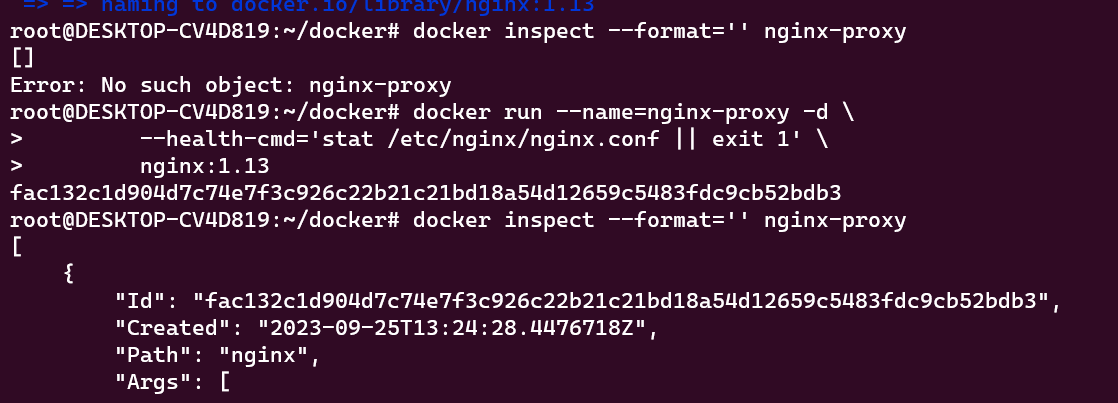


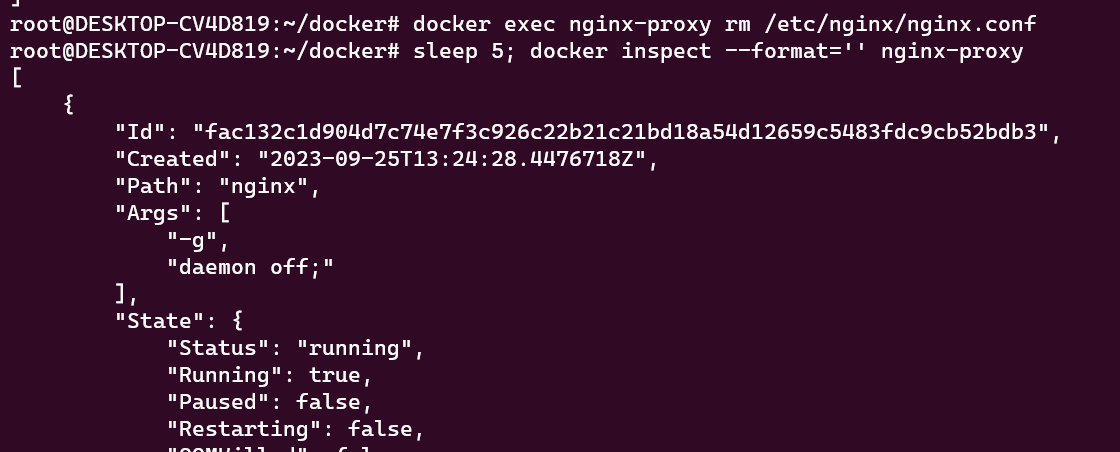


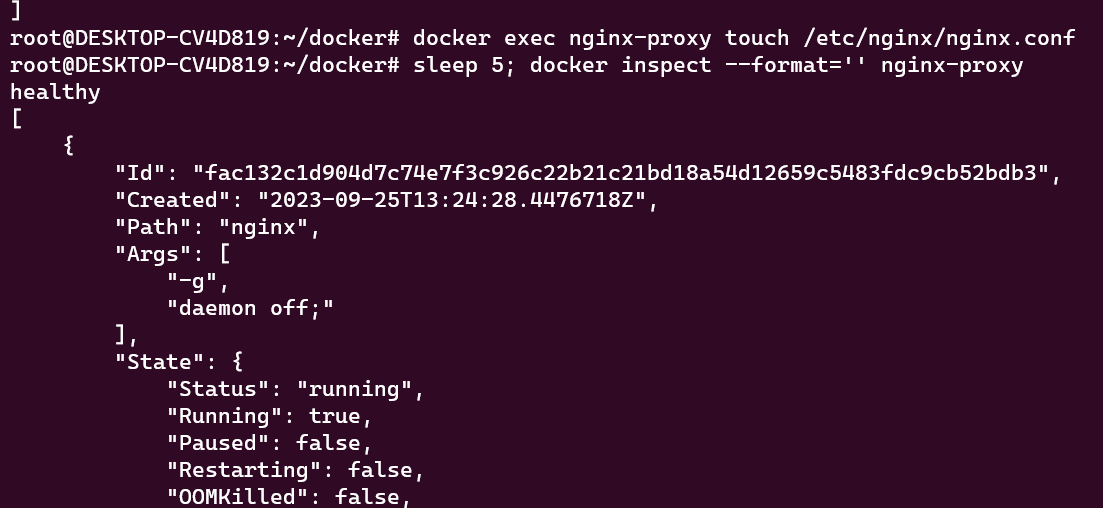


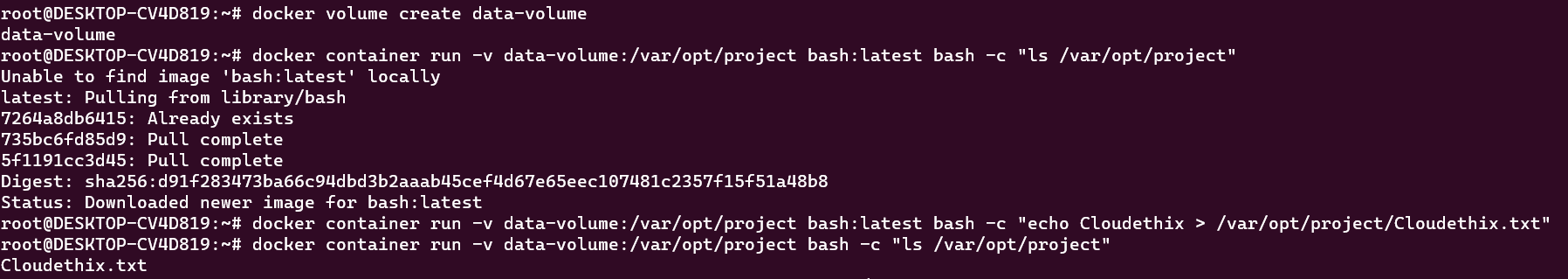
08: Docker Health Checks  
   - Create a Dockerfile for a service that includes health checks. Implement custom health checks and verify the container's health status.

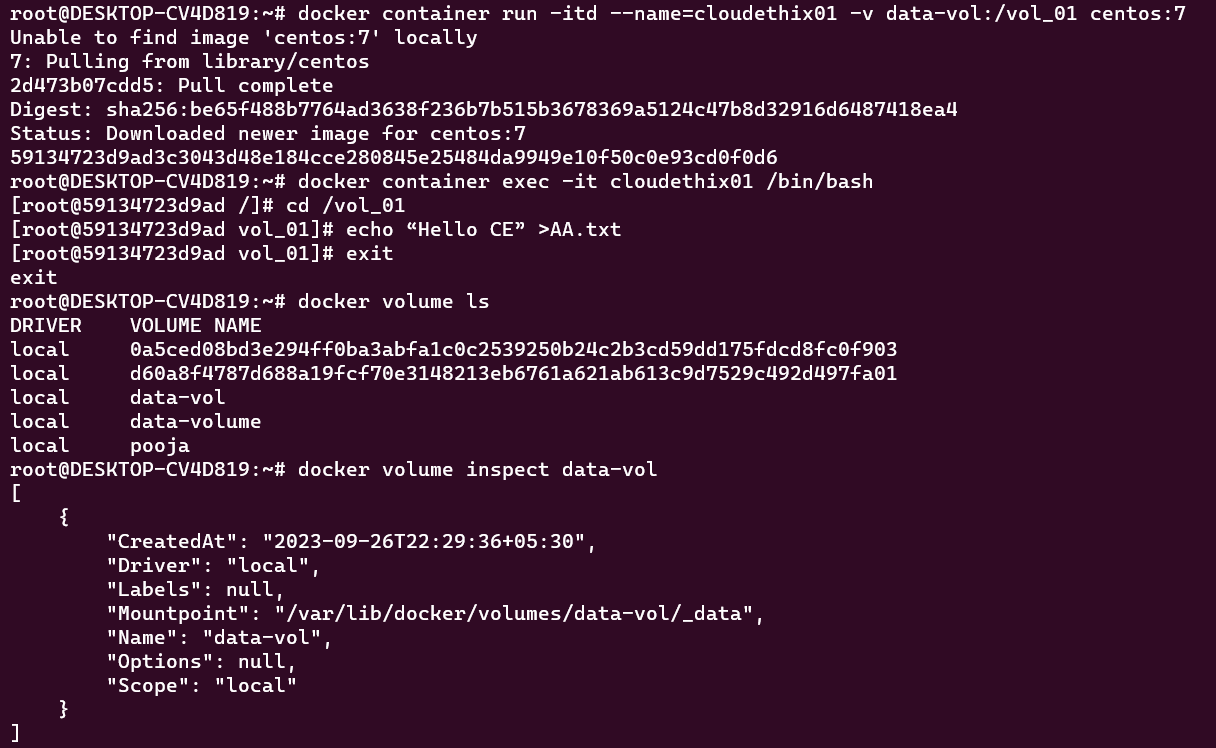


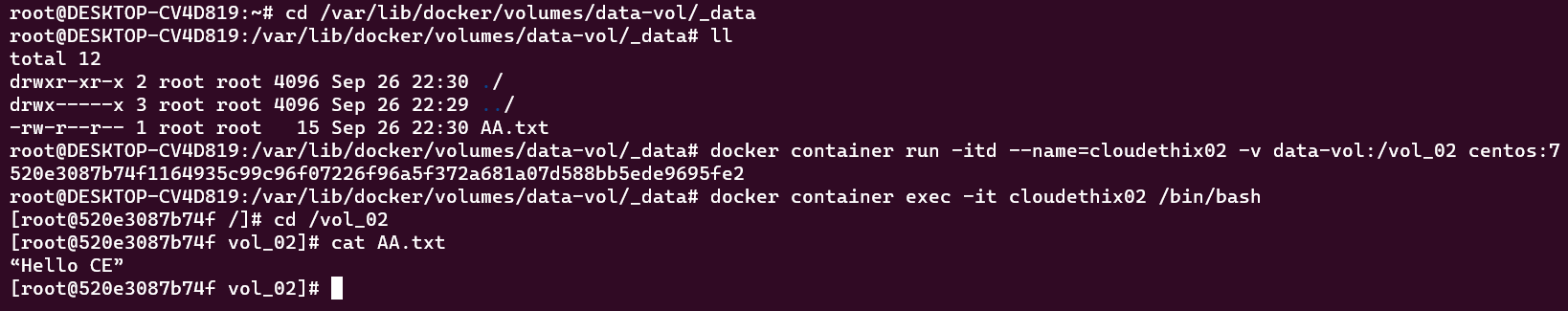




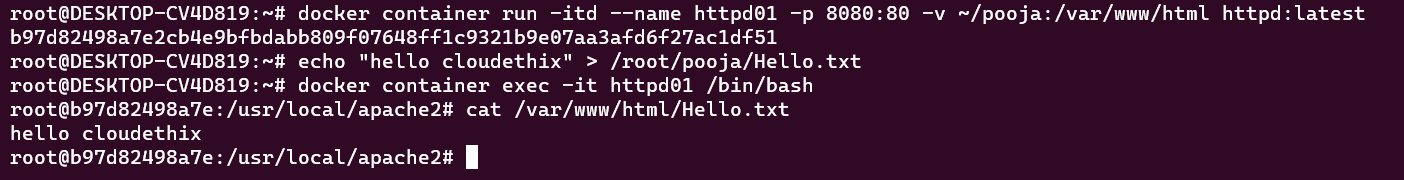
  
  
09: Docker Volumes  
   - Create a Docker volume and use it to persist data generated by a running container. Demonstrate data persistence across container restarts.







 10: Bind Mounts  
   - Configure a Docker container to use a bind mount to access files from the host system. Modify files on the host and observe the changes within the container.

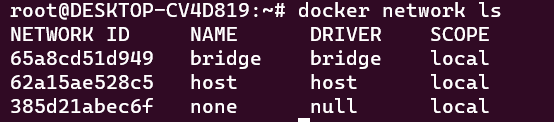


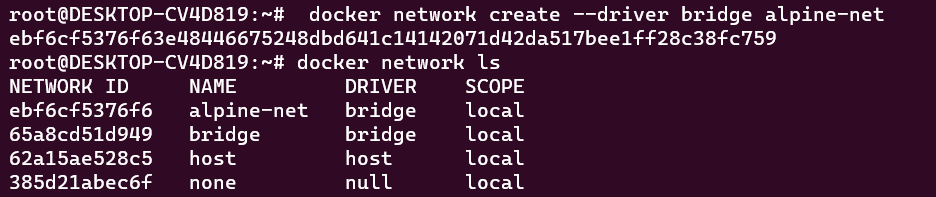
11: Docker Networking Modes  
- Experiment with different Docker networking modes (bridge, host, overlay) and explain their use cases and differences.

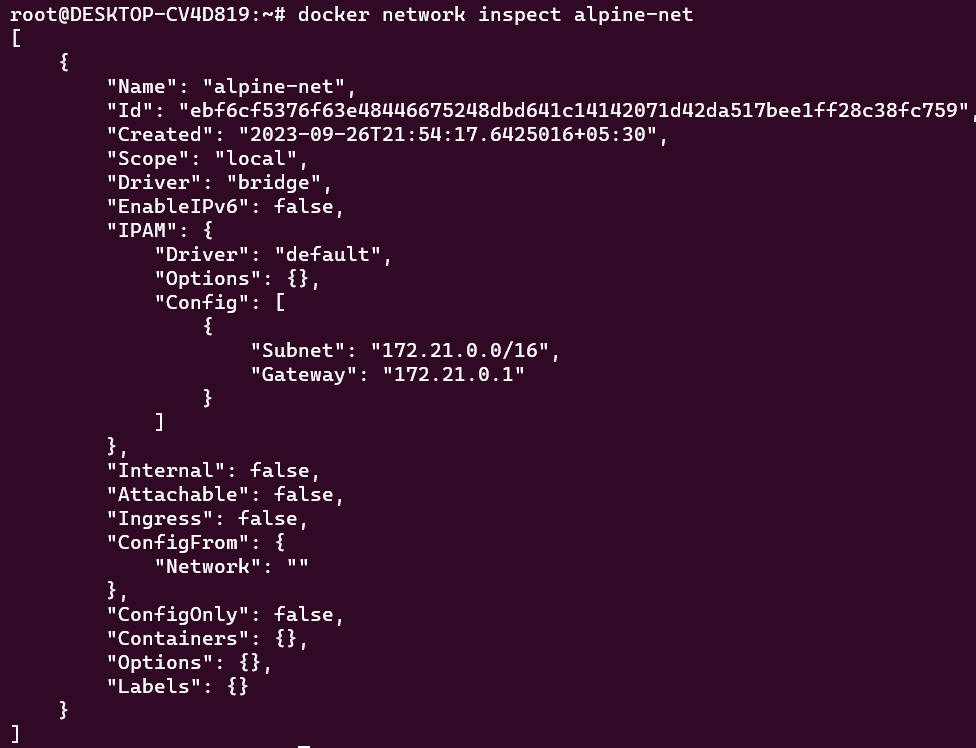
**Docker networking modes:**

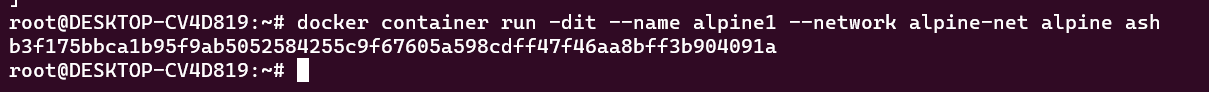
**Bridge Network:**

* List Docker’s networks

****

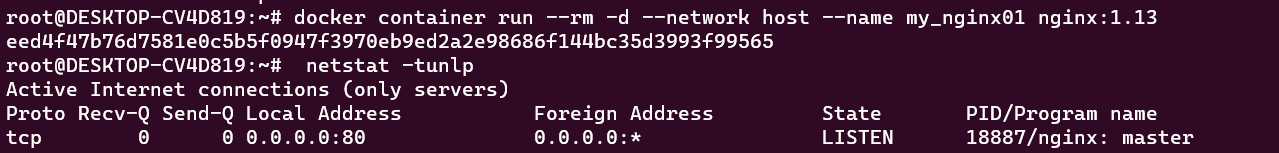
****

* Inspect the alpine-net network. This shows you its IP address and the fact that no containers are connected to it
* Here you will also noticed that it is created with range 172.18.0.0/16 which is separate range from default bridge network i.e. 172.17.0.0/16
* 
* You can only connect to one network during the docker run command, so you need to use docker network connect afterward to connect alpine4 to the bridge network as well.

****

**Host Network:**

Host networks are best when the network stack should not be isolated from the Docker host, but you want other aspects of the container to be isolated.



**None Network:**

With this network type containers are not attached to any network & it will not have access to external network as well as with other containers as well.

# docker container run --rm -d --network none --name my\_nginx nginx

****

12: Container Networking  
   - Create a custom Docker network and deploy a multi-container application where each container serves a unique purpose and communicates effectively.

