Title- The objective of this report is to offer a clear understanding of Docker and its container technology. It will cover the key concepts, architecture, and benefits of Docker, showing how containers enable efficient software development and deployment. The report will also differentiate Docker containers from traditional virtual machines, explaining why Docker is widely adopted in modern software engineering.

Prerequisites:

- Basic understanding of virtualization and cloud computing concepts.
- Familiarity with software development and deployment processes.
- Knowledge of command-line interfaces (CLI) and basic Linux commands.
- Awareness of DevOps practices and the importance of automation in software engineering.

Theory:

Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization. Containers are lightweight, standalone software packages that include everything needed to run an application, ensuring consistent performance across different environments. Unlike virtual machines, containers share the host system's kernel, making them more efficient and faster. Docker simplifies the creation and management of these containers with tools like Docker Engine, Docker Images, and Docker Hub, making it a key technology in modern software development, especially within DevOps and microservices architectures.

Materials and Equipment:

- Computer with Windows, macOS, or Linux.
- Docker Desktop installed.
- Internet connection for Docker images and Docker Hub.
- Text editor or IDE for Dockerfiles and app management.
- Basic CLI tools for running Docker commands.

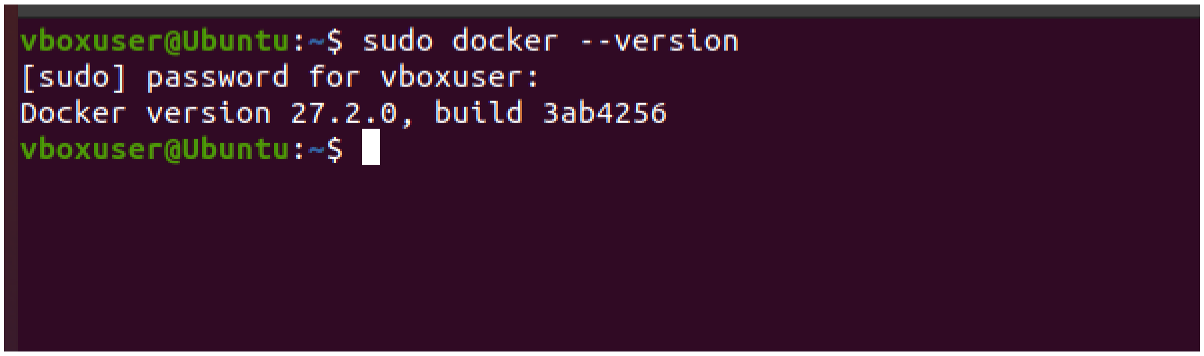
Procedure:

1. Install Docker Desktop:

- Download Docker Desktop from the official Docker website.
- Follow the installation instructions for your operating system.

2. Verify Docker Installation:

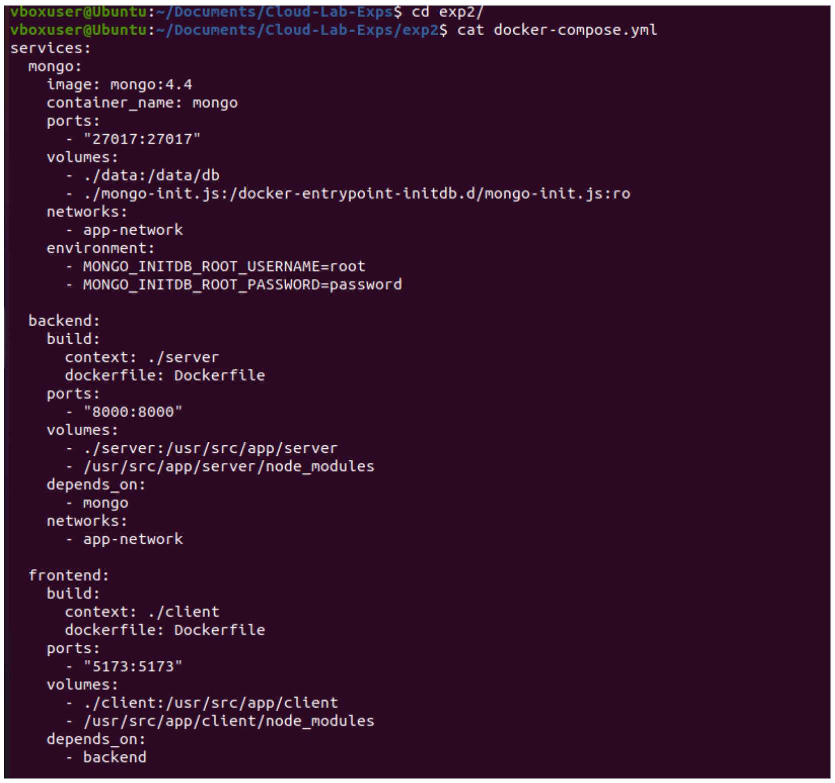
- Open a terminal or command prompt.
- Run the command `docker --version` to check if Docker is installed correctly.



```
vboxuser@Ubuntu:~$ sudo docker --version
[sudo] password for vboxuser:
Docker version 27.2.0, build 3ab4256
vboxuser@Ubuntu:~$
```

3 . Check the contents of the docker-compose.yml file

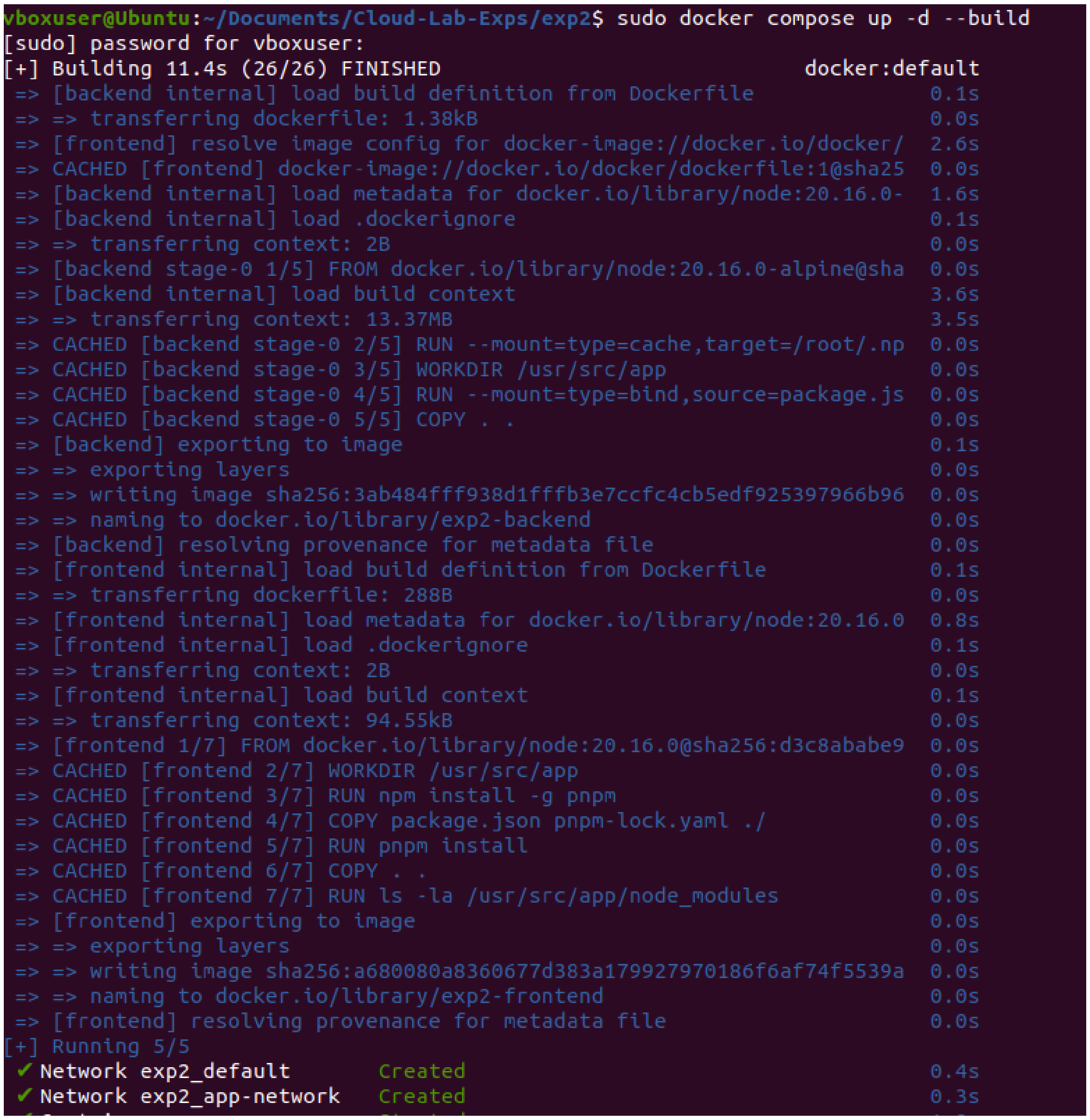
→ Navigate to the exp2 folder in directory and check the contents of the docker-compose.yml file



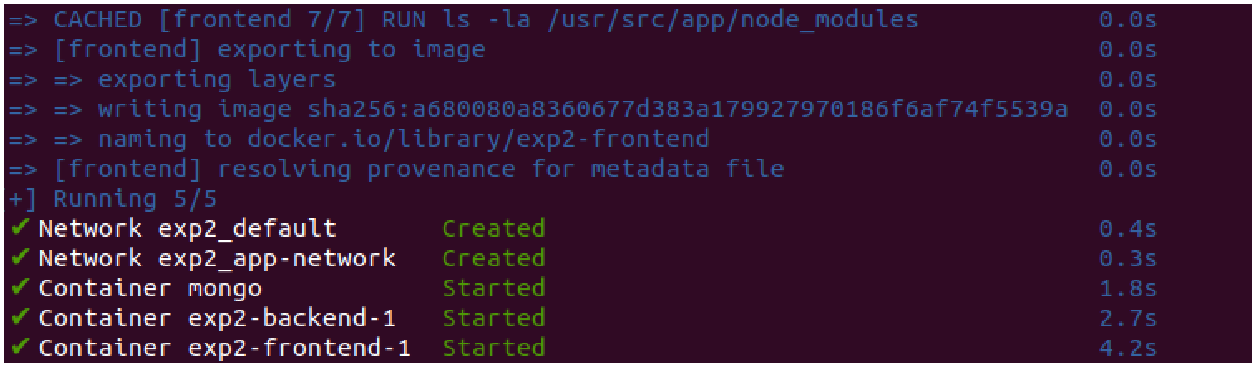
```
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps$ cd exp2/
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ cat docker-compose.yml
services:
  mongo:
    image: mongo:4.4
    container_name: mongo
    ports:
      - "27017:27017"
    volumes:
      - ./data:/data/db
      - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
    networks:
      - app-network
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=password

  backend:
    build:
      context: ./server
      dockerfile: Dockerfile
    ports:
      - "8000:8000"
    volumes:
      - ./server:/usr/src/app/server
      - /usr/src/app/server/node_modules
    depends_on:
      - mongo
    networks:
      - app-network

  frontend:
    build:
      context: ./client
      dockerfile: Dockerfile
    ports:
      - "5173:5173"
    volumes:
      - ./client:/usr/src/app/client
      - /usr/src/app/client/node_modules
    depends_on:
      - backend
```

4. Start the docker container

→ using the command - sudo docker compose up -d --build



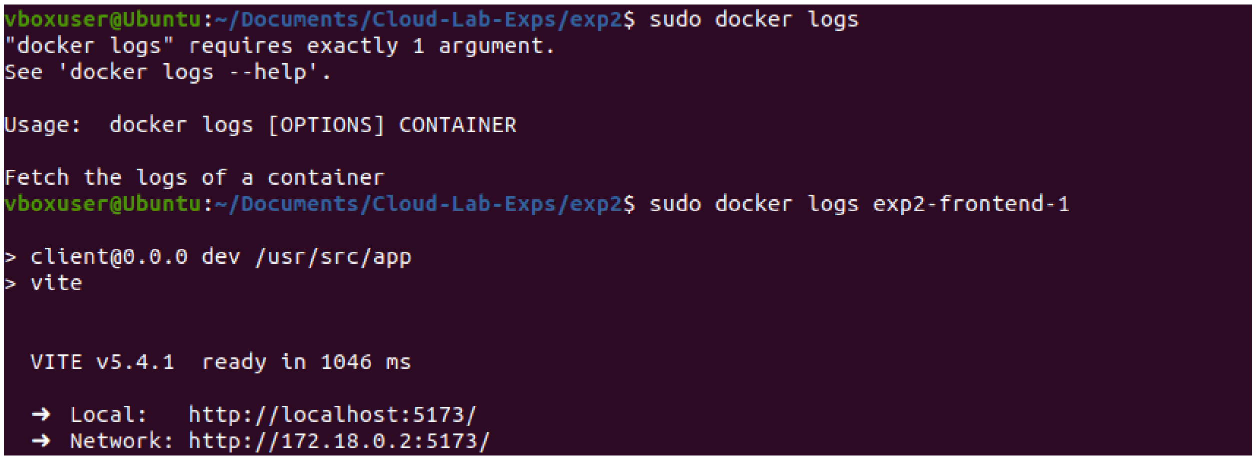
```
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker compose up -d --build
[sudo] password for vboxuser:
[+] Building 11.4s (26/26) FINISHED                           docker:default
 => [backend internal] load build definition from Dockerfile            0.1s
 => => transferring dockerfile: 1.38kB                                  0.0s
 => [frontend] resolve image config for docker-image://docker.io/docker/ 2.6s
 => CACHED [frontend] docker-image://docker.io/docker/dockerfile:1@sha25 0.0s
 => [backend internal] load metadata for docker.io/library/node:20.16.0- 1.6s
 => [backend internal] load .dockerignore                               0.1s
 => => transferring context: 2B                                         0.0s
 => [backend stage-0 1/5] FROM docker.io/library/node:20.16.0-alpine@sha 0.0s
 => [backend internal] load build context                               3.6s
 => => transferring context: 13.37MB                                    3.5s
 => CACHED [backend stage-0 2/5] RUN --mount=type=cache,target=/root/.np 0.0s
 => CACHED [backend stage-0 3/5] WORKDIR /usr/src/app                   0.0s
 => CACHED [backend stage-0 4/5] RUN --mount=type=bind,source=package.js 0.0s
 => CACHED [backend stage-0 5/5] COPY . .                               0.0s
 => [backend] exporting to image                                        0.1s
 => => exporting layers                                                 0.0s
 => => writing image sha256:3ab484fff938d1fffb3e7ccfc4cb5edf925397966b96 0.0s
 => => naming to docker.io/library/exp2-backend                         0.0s
 => [backend] resolving provenance for metadata file                    0.0s
 => [frontend internal] load build definition from Dockerfile           0.1s
 => => transferring dockerfile: 288B                                    0.0s
 => [frontend internal] load metadata for docker.io/library/node:20.16.0 0.8s
 => [frontend internal] load .dockerignore                              0.1s
 => => transferring context: 2B                                         0.0s
 => [frontend internal] load build context                              0.1s
 => => transferring context: 94.55kB                                    0.0s
 => [frontend 1/7] FROM docker.io/library/node:20.16.0@sha256:d3c8ababe9 0.0s
 => CACHED [frontend 2/7] WORKDIR /usr/src/app                          0.0s
 => CACHED [frontend 3/7] RUN npm install -g pnpm                       0.0s
 => CACHED [frontend 4/7] COPY package.json pnpm-lock.yaml ./           0.0s
 => CACHED [frontend 5/7] RUN pnpm install                              0.0s
 => CACHED [frontend 6/7] COPY . .                                      0.0s
 => CACHED [frontend 7/7] RUN ls -la /usr/src/app/node_modules          0.0s
 => [frontend] exporting to image                                       0.0s
 => => exporting layers                                                 0.0s
 => => writing image sha256:a680080a8360677d383a179927970186f6af74f5539a 0.0s
 => => naming to docker.io/library/exp2-frontend                        0.0s
 => [frontend] resolving provenance for metadata file                   0.0s
[+] Running 5/5
 ✔ Network exp2_default       Created                                   0.4s
 ✔ Network exp2_app-network   Created                                   0.3s
```



```
=> CACHED [frontend 7/7] RUN ls -la /usr/src/app/node_modules          0.0s
=> [frontend] exporting to image                                       0.0s
=> => exporting layers                                                 0.0s
=> => writing image sha256:a680080a8360677d383a179927970186f6af74f5539a 0.0s
=> => naming to docker.io/library/exp2-frontend                        0.0s
=> [frontend] resolving provenance for metadata file                   0.0s
+] Running 5/5
✔ Network exp2_default       Created                                   0.4s
✔ Network exp2_app-network   Created                                   0.3s
✔ Container mongo            Started                                   1.8s
✔ Container exp2-backend-1   Started                                   2.7s
✔ Container exp2-frontend-1  Started                                   4.2s
```

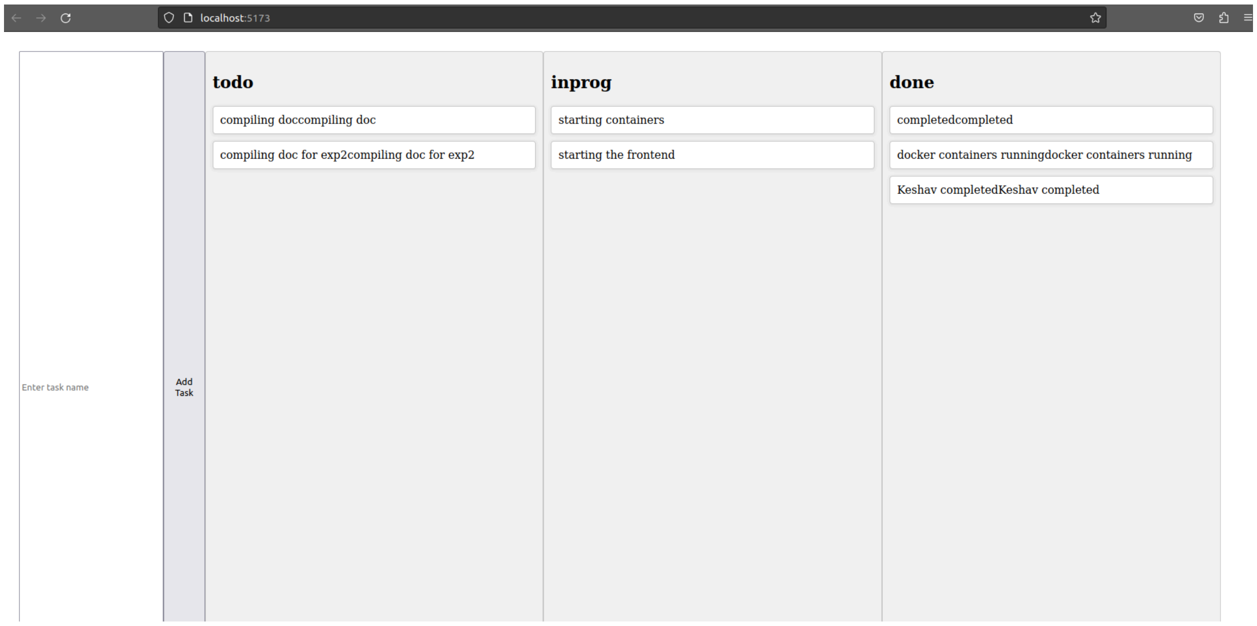
5. Check the logs of the running containers

→ Using the command sudo docker logs exp2-frontend-1 where exp2-frontend-1 is the <container-name>



```
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker logs
"docker logs" requires exactly 1 argument.
See 'docker logs --help'.

Usage:  docker logs [OPTIONS] CONTAINER

Fetch the logs of a container
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker logs exp2-frontend-1

> client@0.0.0 dev /usr/src/app
> vite


  VITE v5.4.1  ready in 1046 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: http://172.18.0.2:5173/
```

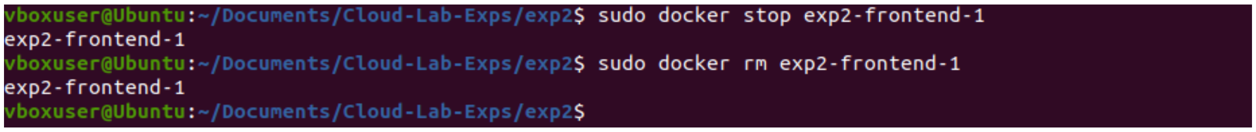
6. Ctrl+ click on the link to see the app running in the browser using the containers



todo	inprog	done
compiling doccompiling doc	starting containers	completedcompleted
compiling doc for exp2compiling doc for exp2	starting the frontend	docker containers runningdocker containers running
		Keshav completedKeshav completed

7. Stop the Docker Container and Remove the container

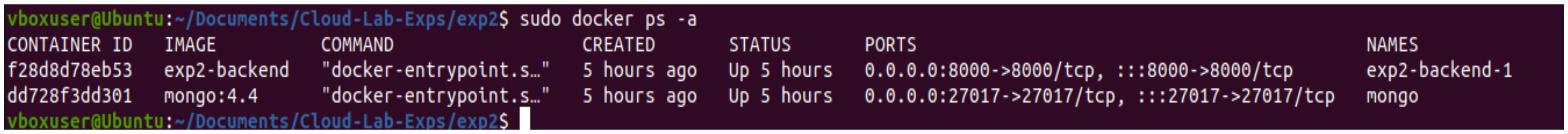
- Stop the container with `docker stop <container-name>`
- Remove the stopped container using `docker rm <container-name>`.



```
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker stop exp2-frontend-1
exp2-frontend-1
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker rm exp2-frontend-1
exp2-frontend-1
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$
```

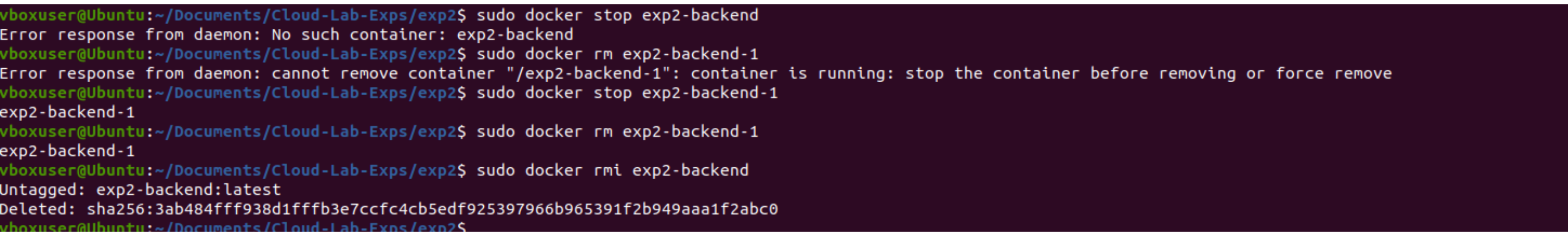
8.. List Running Containers:

- Use the command `docker ps -a` to see all active containers on your system.



```
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS       PORTS                                           NAMES
f28d8d78eb53   exp2-backend   "docker-entrypoint.s…"   5 hours ago   Up 5 hours   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp       exp2-backend-1
dd728f3dd301   mongo:4.4      "docker-entrypoint.s…"   5 hours ago   Up 5 hours   0.0.0.0:27017->27017/tcp, :::27017->27017/tcp   mongo
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ 
```

9. Clean up space



```
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker stop exp2-backend
Error response from daemon: No such container: exp2-backend
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker rm exp2-backend-1
Error response from daemon: cannot remove container "/exp2-backend-1": container is running: stop the container before removing or force remove
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker stop exp2-backend-1
exp2-backend-1
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker rm exp2-backend-1
exp2-backend-1
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$ sudo docker rmi exp2-backend
Untagged: exp2-backend:latest
Deleted: sha256:3ab484fff938d1fffb3e7ccfc4cb5edf925397966b965391f2b949aaa1f2abc0
vboxuser@Ubuntu:~/Documents/Cloud-Lab-Exps/exp2$
```

Expected Output:

The expected output includes a successful Docker Desktop installation without errors and confirmation of installation via the `docker --version` command. You should see the downloaded Docker image listed under Docker images with `docker images`, and the running container should be visible with `docker ps`, showing details like container ID and status. You should be able to access and interact with the container's terminal. After stopping the container with `docker stop`, it should be removable with `docker rm`, and unused images should be cleaned up, resulting in reduced disk usage.

Observations:

Docker Desktop installs and runs without issues. Images and containers appear as expected, with quick starts and efficient performance. Terminal access works, and containers stop and remove correctly, with unused images cleaned up effectively.

Result:

Understood the concept of using Docker-Container.