

Task 2

Kyrillos Wahid Wahib

1. Theory Questions

a. Docker Fundamentals

- **What is a Docker container, and how is it different from a virtual machine (VM)?**

Docker Container: is a lightweight, standalone, and executable software package that includes everything needed to run a piece of software, including the code, runtime, libraries, and system tools.

Docker Container	Virtual Machine
Share the host OS kernel.	Each VM runs a full OS on top of a Hypervisor.
Lightweight, typically megabytes.	Heavyweight, usually gigabytes.
Fast startup time (seconds).	Slower startup time (minutes).
Shares resources with the host (efficient).	Resource-intensive
Ideal for microservices and cloud-native apps.	Better for running multiple different OSs.

- **What is the purpose of a Dockerfile? Explain the significance of directives like FROM , COPY , RUN , and CMD .**

Purpose of Dockerfile: Automate the process of creating a Docker image, ensure consistent builds by documenting the setup process and enable sharing of the setup configuration with others.

FROM	Sets the foundation for your image, determining the OS and environment.
COPY	Transfers application code, configuration files, or other assets into the container image.
RUN	Performs tasks needed to prepare the image before it is run as a container.
CMD	Determines the main process to execute in the container.

b. Image Management

- **Describe the layers of a Docker image. How does Docker optimize space and performance using these layers?**

Docker images are composed of layers, each representing an instruction in the Dockerfile.

- Base Layers: usually the base image specified in the `FROM` directive.
- Intermediate Layers: include changes made to the filesystem during the build process.
- Final Layers: includes all the cumulative changes and is what makes up the Docker image that containers use.

Optimizing space and performance using these layers:

- Identical layers are shared between images to save space
 - Usually layers are reused from the cache, speeding up rebuilds.
 - Layers are stored once and reused across images, minimizing disk usage.
- **What are the benefits of using Docker volumes? Give an example where data persistence is crucial in a Docker container.**

Benefits of using Docker Volumes:

- Volumes allow data to persist even if the container is stopped, restarted, or removed.
- Data stored in volumes is isolated to prevent accidental deletion.
- Volumes can be shared across multiple containers.
- Volumes can be easily backed up or restored for better recovery.

When running a database in a Docker container:

- Without a volume: The database files are stored in the container's filesystem. If the container is removed, all data is lost.
- With a volume: The database files are stored in a Docker volume, ensuring that data persists even if the container is recreated.

c. Networking in Docker

- **How does Docker handle networking? Explain the difference between bridge, host, and none network modes in Docker.**

Docker enables containers to communicate with each other and the outside world through its networking features. By default, Docker sets up a virtual network environment that isolates containers while allowing necessary communication.

-	Bridge	Host	None
Isolation	Containers are isolated from host network	Containers share the host network namespace	Containers have no network connectivity
IP Address	Yes (within Docker network)	No (uses host's IP)	No external IP (only loopback)
Port Mapping	Required to expose services to host	Not required, services are directly accessible	-
Communication	With other containers on the same network	With host and external networks directly	None (no network interfaces)
Use	General-purpose, multi-container apps	High-performance, monitoring tools	Secure, isolated tasks without network access

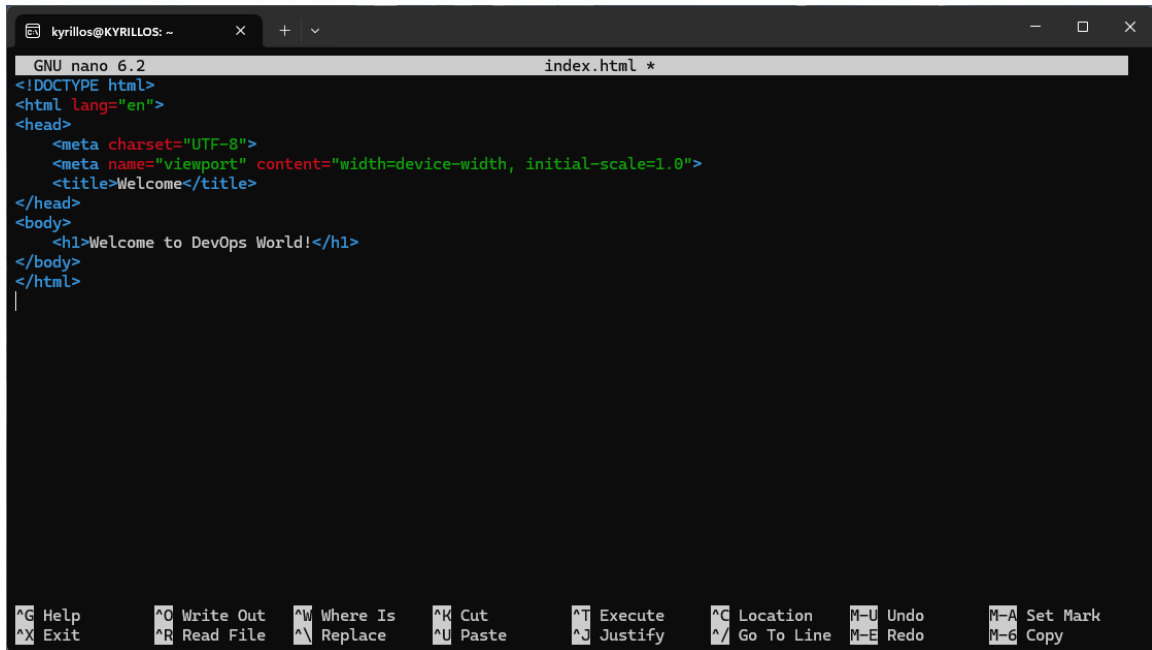
- **Describe how you would configure container-to-container communication within a Docker network.**

Use a custom Docker network. Containers in the same network can communicate by name.

1. `docker network create my_network`
 2. `docker run -d --name container1 --network my_network image1`
`docker run -d --name container2 --network my_network image2`
 3. `docker exec -it container1 ping container2`
-


2. Practical Task

a. Dockerfile Creation



```
kyrillos@KYRILLOS: ~  
GNU nano 6.2 index.html *  
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Welcome</title>  
</head>  
<body>  
  <h1>Welcome to DevOps World!</h1>  
</body>  
</html>  
|  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo      M-G Copy
```

index.html




```
kyrillos@KYRILLOS: ~  
GNU nano 6.2 Dockerfile *  
FROM ubuntu:latest  
RUN apt-get update && apt-get install -y nginx && apt-get clean  
COPY index.html /var/www/html/index.html  
EXPOSE 8080  
CMD ["nginx", "-g", "daemon off;"]  
|  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo      M-G Copy
```

Dockerfile

```
kyrillos@KYRILLOS: ~$ nano index.html
kyrillos@KYRILLOS: ~$ nano Dockerfile
kyrillos@KYRILLOS: ~$ docker build -t custom-nginx .
ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http:
//%2Fvar%2Frun%2Fdocker.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied
kyrillos@KYRILLOS: ~$ sudo docker build -t custom-nginx .
[sudo] password for kyrillos:
[+] Building 39.1s (8/8) FINISHED
=> [internal] load build definition from Dockerfile                                docker:default 0.1s
=> => transferring dockerfile: 210B                                              0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest                  3.6s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [1/3] FROM docker.io/library/ubuntu:latest@sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc 18.1s
=> => resolve docker.io/library/ubuntu:latest@sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6 0.0s
=> => sha256:de44b265507ae44b212defcb50694d666f136b35c1090d9709068bc861bb2d64 29.75MB / 29.75MB 16.7s
=> => sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab 6.69kB / 6.69kB 0.0s
=> => sha256:6e75a10070b0fcb0bead763c5118a369bc7cc30dfc1b0749c491bbb21f15c3c7 424B / 424B 0.0s
=> => sha256:b1d9df8ab81559494794e522b380878cf9ba82d4c1fb67293bcf931c3aa69ae4 2.30kB / 2.30kB 0.0s
=> => extracting sha256:de44b265507ae44b212defcb50694d666f136b35c1090d9709068bc861bb2d64 1.2s
=> [internal] load build context                                                  0.0s
=> => transferring context: 277B                                                  0.0s
=> [2/3] RUN apt-get update && apt-get install -y nginx && apt-get clean        16.9s
=> [3/3] COPY index.html /var/www/html/index.html                               0.0s
=> exporting to image                                                            0.2s
=> => exporting layers                                                            0.1s
=> => writing image sha256:ff0b82e20cdb6735e3420e70bd8037fde0c6dbf1686a3a48357dcfb0ff8dd9c8 0.0s
=> => naming to docker.io/library/custom-nginx                                   0.0s
kyrillos@KYRILLOS: ~$ |
```

docker build

```
kyrillos@KYRILLOS: ~$ doc
ERROR: permission denied
//%2Fvar%2Frun%2Fdocker.
kyrillos@KYRILLOS: ~$ sud
[sudo] password for kyri
[+] Building 39.1s (8/8)
=> [internal] load build
=> => transferring dock
=> [internal] load meta
=> [internal] load .doc
=> => transferring cont
=> [1/3] FROM docker.io
=> => resolve docker.io
=> => sha256:de44b26550
=> => sha256:80dd3c3b9c
=> => sha256:6e75a10070
=> => sha256:b1d9df8ab8
=> => extracting sha256
=> [internal] load build
=> => transferring cont
=> [2/3] RUN apt-get up
=> [3/3] COPY index.htm
=> exporting to image
=> => exporting layers
=> => writing image sha
=> => naming to docker.io/library/custom-nginx
kyrillos@KYRILLOS: ~$ sudo docker run -d -p 8080:80 custom-nginx
92ac58177809cd3ea28b38a44f8a1a9feb0e66e89dc459c546d0ba9e2952178d
kyrillos@KYRILLOS: ~$
kyrillos@KYRILLOS: ~$
```



docker run + browser

b. Multi-Container Setup

```
kyrillos@KYRILLOS: ~  
GNU nano 6.2 docker-compose.yml *  
version: "3"  
  
services:  
  web:  
    image: nginx:latest  
    container_name: nginx_web  
    ports:  
      - "8080:80"  
    volumes:  
      - ./nginx-custom.conf:/etc/nginx/conf.d/default.conf:ro  
    depends_on:  
      - db  
  
  db:  
    image: postgres:latest  
    container_name: postgres_db  
    environment:  
      POSTGRES_USER: user  
      POSTGRES_PASSWORD: password  
      POSTGRES_DB: exampledb  
    volumes:  
      - db_data:/var/lib/postgresql/data  
  
volumes:  
  db_data:
```

docker-compose.yml

```
kyrillos@KYRILLOS: ~  
sudo: docker-compose.yml: command not found  
kyrillos@KYRILLOS:~$ sudo docker-compose  
sudo: docker-compose.yml: command not found  
kyrillos@KYRILLOS:~$ docker-compose up -d  
permission denied while trying to connect  
%2Frun%2Fdocker.sock/v1.24/containers/jse  
%2C%22com.docker.compose.project%3Dkyril  
kyrillos@KYRILLOS:~$ sudo docker-compose  
[+] Running 3/15  
[+] Running 15/15 [#####] 11.56MB/  
✓db 14 layers [#####] 0B/  
✓bc0965b23a04 Pull complete  
✓002e1a8eb6f9 Pull complete  
✓a24f300391ed Pull complete  
✓627f580b7ad7 Pull complete  
✓cfb3c2203f88 Pull complete  
✓9e592465b243 Pull complete  
✓8d4265d09d9c Pull complete  
✓e3a8293e92fd Pull complete  
✓2cb801c39436 Pull complete  
✓c5fdb20d8658 Pull complete  
✓67c5fe618f0c Pull complete  
✓c9cdd1fe82e4 Pull complete  
✓8f152c4aceed Pull complete  
✓2cd360f3b7db Pull complete  
[+] Running 4/4  
✓Network kyrillos_default Created 0.3s  
✓Volume "kyrillos_db_data" Created 0.0s  
✓Container postgres_db Started 0.2s  
✓Container nginx_web Created 0.0s
```

docker-compose

c. Resource Limiting

```
kyrillos@KYRILLOS: ~  
[+] Running 15/15.### ] 11.56MB/36.3MB Pulling 18.4s  
✓db 14 layers [#####] 0B/0B Pulled 103.4s  
✓bc0965b23a04 Pull complete 29.7s  
✓002e1a8eb6f9 Pull complete 1.3s  
✓a24f300391ed Pull complete 9.2s  
✓627f580b7ad7 Pull complete 4.4s  
✓cfb3c2203f88 Pull complete 16.8s  
✓9e592465b243 Pull complete 11.0s  
✓8d4265d09d9c Pull complete 17.5s  
✓e3a8293e92fd Pull complete 17.9s  
✓2cb801c39436 Pull complete 91.2s  
✓c5fdb20d8658 Pull complete 19.0s  
✓67c5fe618f0c Pull complete 21.2s  
✓c9cdd1fe82e4 Pull complete 22.8s  
✓8f152c4aceed Pull complete 27.0s  
✓2cd360f3b7db Pull complete 28.1s  
[+] Running 4/4  
✓Network kyrillos_default Created 0.3s  
✓Volume "kyrillos_db_data" Created 0.0s  
✓Container postgres_db Started 0.2s  
✓Container nginx_web Created 0.0s  
Error response from daemon: driver failed programming external connectivity on endpoint nginx_web (75c8099688e74075c4db7411ff90d120f8550ebc43c815cd6a3a119dfff5114e7): Bind for 0.0.0.0:8080 failed: port is already allocated  
kyrillos@KYRILLOS:~$ sudo docker run -d \  
--name limited_web \  
--memory="512m" \  
--cpus="1" \  
-p 8080:80 \  
custom-nginx  
e5d89b32fa88e3ad4ca507453a97953d871daaf5595d4d0f30968fd737a27b33
```