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, ensuree 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.**

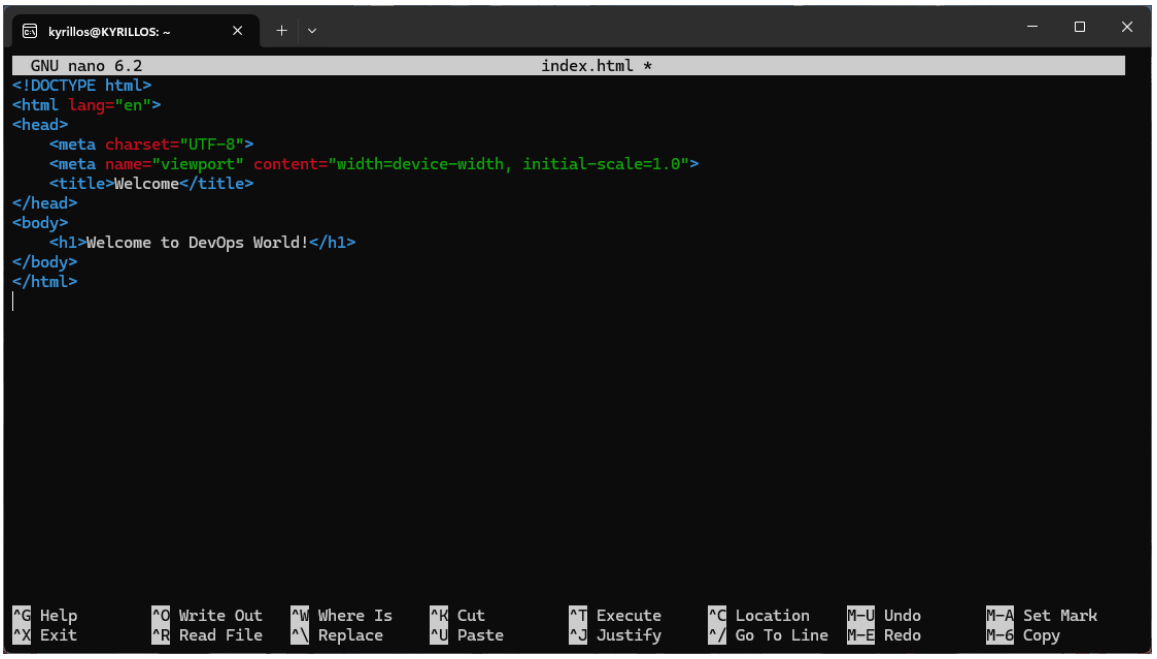
  Use a custom Docker network. Containers in the same network can communicate by name.

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

---

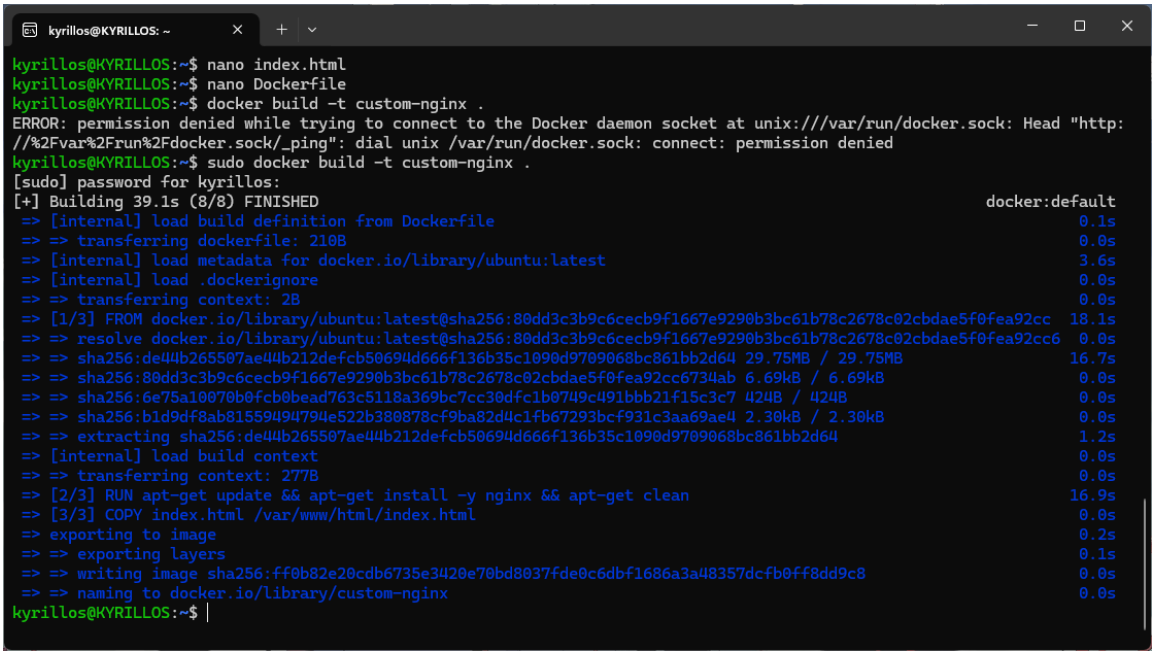
2. Practical Task

a. **Dockerfile Creation**

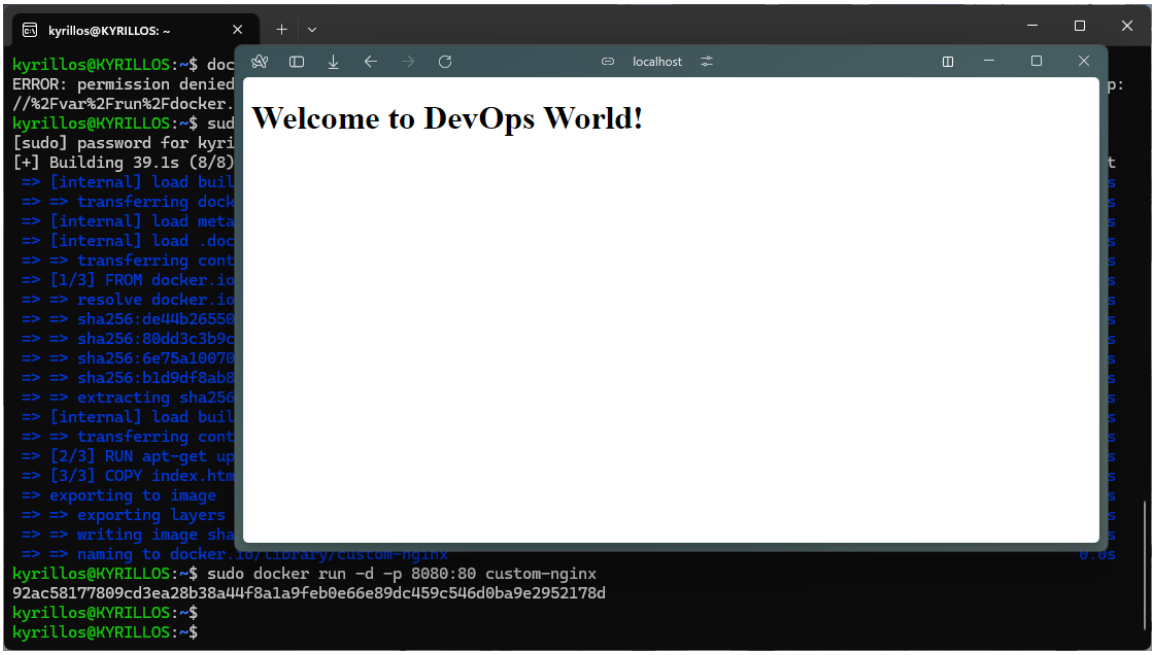


index.html

Dockerfile

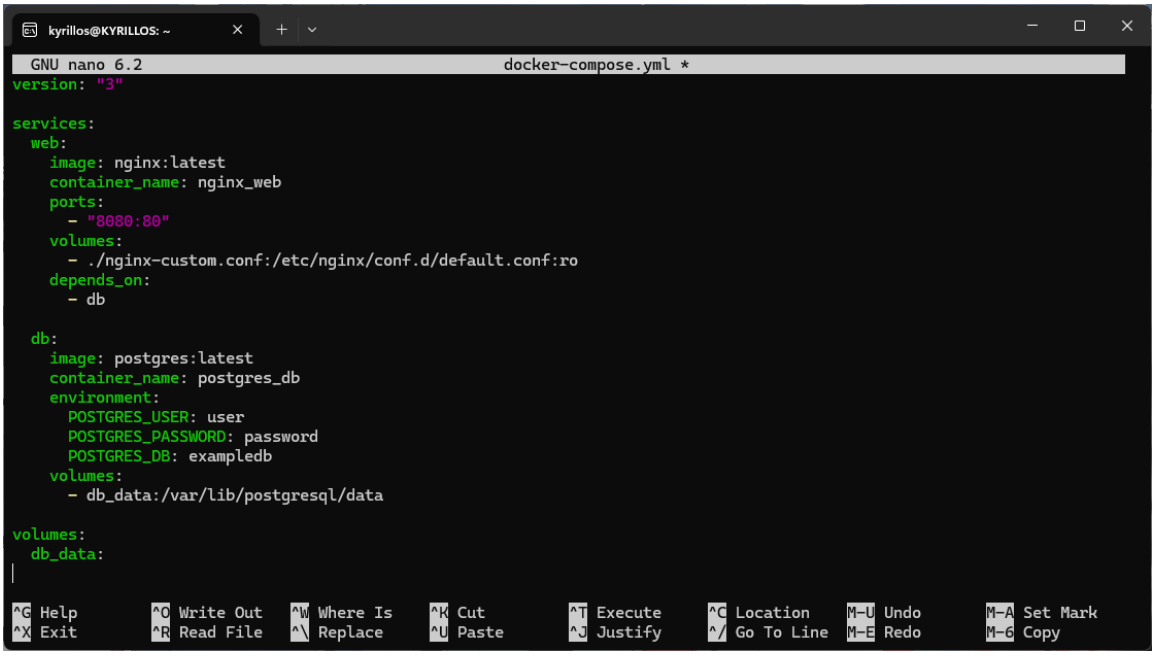


docker build

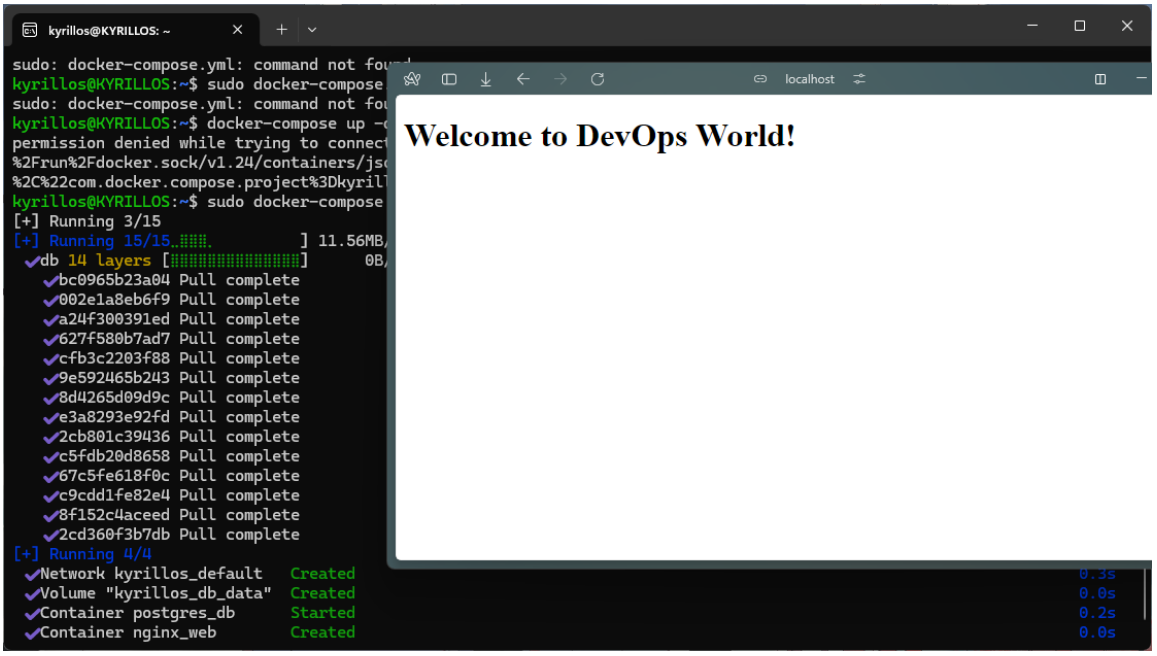


docker run + browser

b. **Multi-Container Setup**



docker-compose.yml



docker-compose

c. Resource Limiting