Session 03: Create and run a multi-service application using Docker Compose (e.g., a web app with a database)

Pre-Lab

1. What is Docker Compose and why is it used?

Docker Compose is a tool used to **define and run multi-container Docker applications** using a single YAML file (docker-compose.yml).

Docker Compose is required to:

- Manage multiple containers together
- Start/stop services using one command
- Automatically create networks
- Simplify microservices deployment

2. What is a "service" in a docker-compose.yml file?

A **service** represents **one container** (e.g., web, database) defined inside docker-compose.yml.

3. What is the default file name used by Docker Compose?

docker-compose.yml

<u>In-Lab Tasks</u>

1. Use Docker Compose to set up and run three tier web application with two services (web and database) defined ports, dependencies, and volumes. Start the stack and verify both containers run and connect properly.

Step 1: Create Docker Compose File

1. Open **VS Code**
2. Install **YAML (Red Hat)** extension
3. Create a file named **docker-compose.yml**

Step 2: Define Services in docker-compose.yml

```
version: "3.8"

services:
  web:
    image: nginx
    container_name: web_container
    ports:
      - "8080:80"
    depends_on:
      - db

  db:
    image: mysql:8
    container_name: db_container
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: mydb
    volumes:
      - db_data:/var/lib/mysql

volumes:
  db_data:
```

Step 3: Explanation of Configuration

Web Service

- Uses **Nginx** web server
- Exposes container port **80** to host port **8080**
- Depends on database service to start first

Database Service

- Uses **MySQL 8**
- Root password and database name defined using environment variables
- Database data stored in a **named volume**

Volume

- db_data ensures **persistent storage** for MySQL data

Step 4: Start the Application Stack

Open terminal in the project directory and run:

```
docker-compose up -d
```

Step 5: Verify Containers Are Running

```
docker ps
```

Expected:

- web_container → running
- db_container → running

Step 6: Verify Web Application

Open browser and access:

http://localhost:8080

- Nginx default page confirms **web container is running**
- ✓ Database container runs in background and is reachable via Docker network

Step 7: Stop and Clean Up the Stack

```
docker-compose down
```

(To remove volumes also)

```
docker-compose down -v
```

Result

The three-tier web application was successfully deployed using Docker Compose. The web and database containers were started, verified, and managed using a single configuration file.

Post Lab Program

1. After setting up and running your Docker Compose stack with two services (web and database), explain the role of defining ports, volumes, and service dependencies in the docker-compose.yml file

1. Role of Ports

What ports do?

Ports define how a service inside a container is accessed from the host machine or external users.

Why ports are defined

- Containers run in an isolated environment.
- Without port mapping, services inside containers are not accessible from outside.

Example

```
ports:
  - "8080:80"
```

Explanation

- 80 → Port inside the container (web server)
- 8080 → Port on the host machine
- Allows users to access the web application using:
- http://localhost:8080

Importance

- Enables browser access to the web service
- Allows testing and debugging
- Supports external client communication

2. Role of Volumes

What volumes do?

Volumes provide **persistent storage** for container data.

Why volumes are needed

- Containers are temporary; data is lost when containers are removed.
- Databases require data persistence.

Example

```
volumes:
  - db_data:/var/lib/mysql
```

Explanation

- db_data → Named Docker volume
- /var/lib/mysql → Database data directory inside the container

Importance

- Preserves database data even after container restart or removal
- Enables data backup and migration
- Separates application lifecycle from data lifecycle

3. Role of Service Dependencies (depends_on)

What depends_on does?
Defines the startup order of services.

Why it is required

- The web service often depends on the database.
- Prevents the web container from starting before the database container.

Example

```
depends_on:
  - db
```

Explanation

- Ensures the database container starts before the web container
- Avoids initial connection failures

Important Note

- depends_on ensures **startup order**, not **service readiness**
- Additional health checks are needed for full readiness control

Summary Table

Component	Role
Ports	Expose container services to host/external users
Volumes	Provide persistent storage for container data
depends_on	Controls startup order between services

Conclusion

Defining **ports, volumes, and service dependencies** in the docker-compose.yml file is essential for enabling service access, ensuring data persistence, and maintaining correct startup order. Together, they ensure smooth communication and reliable operation of multi-container applications.