DOCKER COMPOSE

What is Docker Compose?

- **Docker Compose** is a tool for defining and running multi-container Docker applications.
- With Compose, you can define all your app's services, networks, and volumes in a **single YAML file** (docker-compose.yml).
- It allows you to easily manage the lifecycle of multi-container applications, including starting, stopping, and rebuilding containers.

Why Use Docker Compose?

- **Simplifies Multi-Container Apps**: In full-stack applications, typically multiple containers (e.g., frontend, backend, database) are involved. Docker Compose lets you manage all these services with a single configuration file.
- **Environment Consistency**: Compose ensures that all services are consistently configured across different environments (development, staging, production).
- **Easier Management**: It allows for defining and managing services, volumes, and networks centrally, making deployment and scaling easier.

Basic Structure of a docker-compose.yml File

A basic docker-compose.yml file defines:

1. **Services**: The containers that make up your application (e.g., frontend, backend, database).
2. **Networks**: Defines how containers communicate with each other.
3. **Volumes**: Persists data between container restarts or different services.

Example of docker-compose.yml:

```yaml
networks:
  default:
    name: Nodemongo_APP
services:
  app: #node
    container_name: docker-node
    build: .
    ports:
      - $NODE_LOCAL_PORT:$NODE_DOCKER_PORT
    environment:
      - MONGO_URI=NODE_dbURL
    depends_on:
      - mongo
    networks:
      - default

  mongo:
    container_name: mongo
    image: mongo
    ports:
      - $DB_LOCAL_PORT:$DB_DOCKER_PORT
    volumes:
      - mongo_data:/data/db
    networks:
      - default

volumes:
  mongo_data:
```

Use hyphen/dash **-** to denote array items.

explore: https://www.bezkoder.com/docker-compose-nodejs-mongodb/

Explanation:

- **services**: Lists the containers to be run. In this example:
    - **backend**: A Node.js/Express server that connects to MongoDB.
    - **mongo**: A MongoDB service.
- **volumes**: Specifies persistent storage for MongoDB.

- **Networks**: In a Docker Compose setup, containers need a way to communicate to each other — like: Your **Node backend** communicating to **MongoDB.**

  Docker networks allow this using **internal DNS**. So instead of using IPs, your backend can connect to Mongo like this:

  mongoose.connect("mongodb://mongo:27017/mydb")

Here, "mongo" is the **service name** from the docker-compose.yml, not an IP/localhost.

Service Configuration in docker-compose.yml

Build Section

- **context**: Defines the directory that Docker uses for the build process. Typically, this will point to the project directory (e.g., ./frontend for a React app or ./server for the backend).
- **args**: Used to pass build arguments (e.g., REACT_APP_API_URL) to Docker at build time.

Ports Section

- **ports**: Exposes container ports to the host machine. For example:
  - "3000:80" means that port 80 inside the container will be mapped to port 3000 on the host machine.

Volumes Section

- **volumes**: Mounts directories from the host machine into the container to persist data or allow file sharing.
  - **Bind Mounts**: Directly maps a file or directory on the host to the container (./frontend:/app).
  - **Named Volumes**: Creates a persistent storage volume managed by Docker (e.g., mongo-data:/data/db).

Networks Section

- **networks**: Defines how containers communicate with each other.
  - You can create custom networks and specify which services should be connected to them.

depends_on: Ensures that one service starts only after another service is ready (e.g., the backend depends on MongoDB).

Running Docker Compose

Starting the Application

- Use docker-compose up to start all the services defined in the docker-compose.yml file.
    - **With Detached Mode**: Run in the background using -d:

      docker-compose up -- build -d

Command	Rebuild Image?	When to Use
docker-compose up -d	No	No changes to Dockerfile or image build config
docker-compose up --build -d	Yes	Dockerfile or dependencies changed

what Happens:

- It will pull the necessary images (if not available locally).
- Build the images (if needed).
- Start the containers based on the configuration.

Stopping the Application

- Use docker-compose down to stop the containers and remove networks and volumes.

  docker-compose down

Restarting Services

- Restart all services with docker-compose restart:

  docker-compose restart