Docker Compose

Defination :- Docker compose is a tool that allows you to define, configure, and run one or more docker container using a Yaml configuration file called docker-compose.yml.

Why use Docker Compose

- Simplifies commands :- No need to type long docker run commands.
- Central configuration :- All setting in a single docker-compose.yml.
- Easier Networking :- Containers can communicate using service names.
- One Command Control :- Start, stop & manage containers with `docker compose up` & `docker compose down`

Working

1. Define service in docker-compose.yml (eg app, database).
2. Run `docker compose up` → Create containers, networks, and volumes.
3. Stop with `docker compose down` → removes containers & network not images.

Ex Mern Example

yaml (docker-compose.yml).

```
version: "3.8"  # Latest stable compose version

services:
  backend:
    build: ./backend   # build image from
                       #dockerfile in backend folder
    ports:
      - "5000:5000"    #map Host port 5000 to
                       #container port 5000
    networks:
      - mern-network   # connects to a custom
                       #network.
    depends_on:
      - mongodb        #wait for mongodb
#container to start before backend runs.
```

```
frontend:
    build: ./frontend  #build image from dockerfile
                       # in frontend
    ports:
      - "3000:3000"    #description
                       #same as previous
    networks:
      - mern-network
    depends_on:
      - backend

  mongodb:
      image : mongo:latest #use official mongodb
                           #image from dockers hub.
      volumes:
      -mongo-data:/data/db  # named volume for
                            #  mongodb persistent storage.
      networks:
        - mern_network.

volumes:
    mongo-data:   # named volume defination

networks:
    mern_network:
      driver: bridge # bridge network for isolated
                     # communication
```

Containes dependency

Use depends_on to control startup order.

```yaml
depends_on:
  service_name:
    condition : service_healthy
```

* condition : service_healthy waits for the healthcheck to pass before starting.

healthcheck Example

```
healthcheck:
  test : ["CMD", "mongo", "-eval", "db.admin.command('ping')"]
  interval : 10s
  timeout : 5s
  retries : 5
```

- test → command to check service health
- interval → time between checks
- timeout → wait time before considering failed
- retries → how attempts before making unhealthy.

Ex docker-compose.yml

```
# Networking also use servicename not containername
version : "3.8"
services:
  mongodb:
    image: mongo:latest
    container_name: mongodb
    ports:
      - "27017:27017"
    network:
      -mern-network
    healthcheck:
      test: ["CMD","mongo","--eval","db.admincommand('ping')"]
      interval: 10s
      timeout: 5s
      rubies: 5

  backend:
    build: ./backend
    container_name: mern-backend
    port: "3000-3000"
    depends_on:
      mongodb:
        condition: service_healthy
    environment:
      -MONGO_URI=mongodb://mongodb:27017/usersjs
    network:
      - mernnetwork
```

docker-compose cmd

	Commands	purpose
1.	docker compose up	Start all services (foreground)
2.	docker compose down	Stop & remove containers, networks but keep volumes.
3.	docker compose up -d	Start in background (detached)
4.	docker compose down -v	Stop & remove containers, networks & volumes.
5.	docker compose build	Rebuild images.
6.	docker compose logs	View logs of all services
7.	docker compose logs backend	View logs of backend only
8.	docker compose ps	Show running containers