**Setup & Scaling Steps:**

1. **Define all services in a single docker-compose.yml:**
   * Include services, networks, volumes, environment variables.
   * Example for microservice with DB and config:

services:

service-registry:

image: image-service-registry

networks:

- micro-net

ports:

- "8761:8761"

config-server:

image: image-config-server

depends\_on:

- service-registry

networks:

- micro-net

ports:

- "8888:8888"

orders:

image: image-orders

depends\_on:

- service-registry

- config-server

networks:

- micro-net

environment:

- SPRING\_DATASOURCE\_URL=jdbc:postgresql://orders-db:5432/ordersdb

ports:

- "8081:8080"

orders-db:

image: postgres

networks:

- micro-net

environment:

POSTGRES\_DB: ordersdb

POSTGRES\_USER: ordersuser

POSTGRES\_PASSWORD: orderspass

volumes:

- orders-db-data:/var/lib/postgresql/data

volumes:

orders-db-data:

networks:

micro-net:

driver: bridge

1. **Start and scale containers with Docker Compose commands:**
   * Start all services:

docker-compose up -d

* + Scale a specific service:

docker-compose up -d --scale orders=3

1. **Docker Compose automatically manages:**
   * Container networking on defined networks.
   * Service dependencies via depends\_on.
   * Shared volumes for data persistence.
2. **Port mapping considerations:**
   * When scaling multiple replicas exposing ports, **we do NOT map all replicas to the same host port**.
   * Either expose only one service externally (e.g., API Gateway) or use internal networking + a load balancer.