# **🚀 Docker Setup and Notes**

>>>>>>> Applicable for Monolithic and Microservice both

### **🔧 Install Docker**

* Download Docker: [Get Docker](https://docs.docker.com/get-started/get-docker/)
* Install instructions for Ubuntu: [Official Docs](https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository)
* System requirements: [Linux Docker Desktop](https://docs.docker.com/desktop/setup/install/linux/#general-system-requirements)

### **🎥 Reference Tutorials**

* [Docker Full Playlist - YouTube](https://www.youtube.com/watch?v=NpdG3lmKJ5g&list=PLSVW22jAG8pDeU80nDzbUgr8qqzEMppi8&index=1)
* [Docker Concepts - YouTube](https://www.youtube.com/watch?v=YbSC1OsLp20&list=PLGRDMO4rOGcOlnu6QhogZDNFFwiwKh5X9)

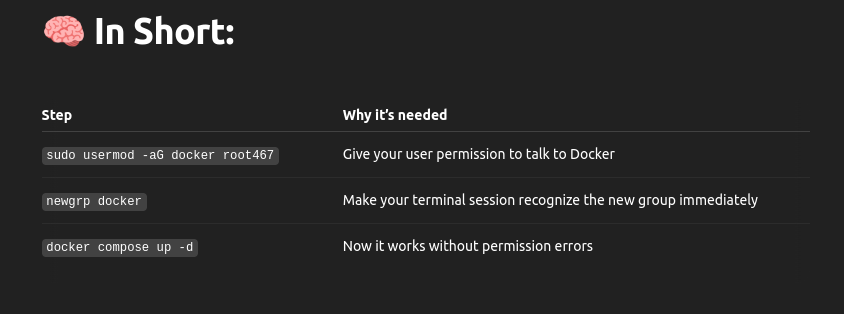
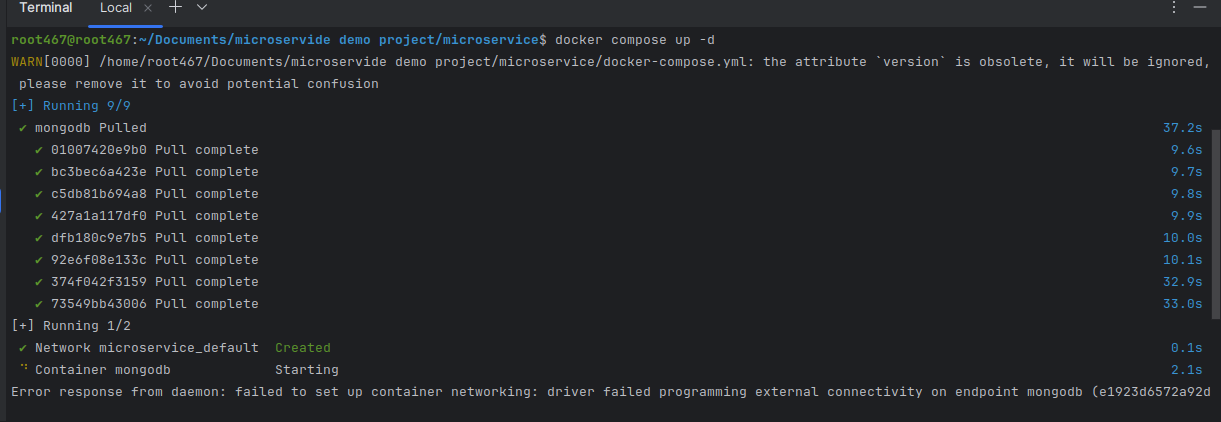
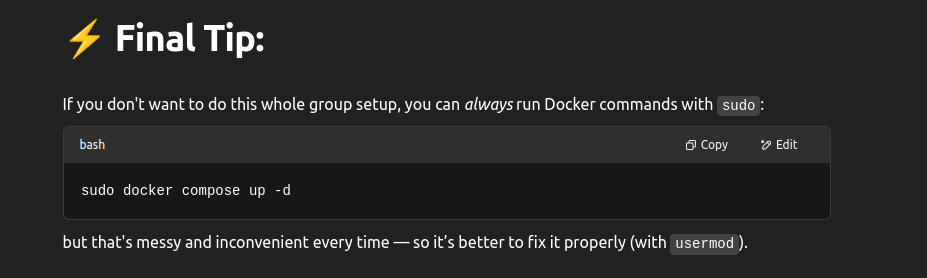
## **🧱 Docker Compose Setup**

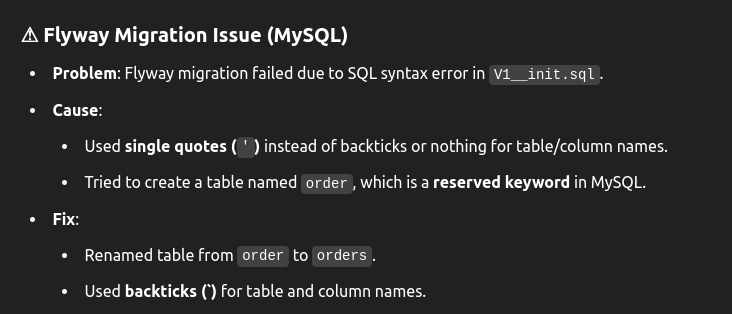
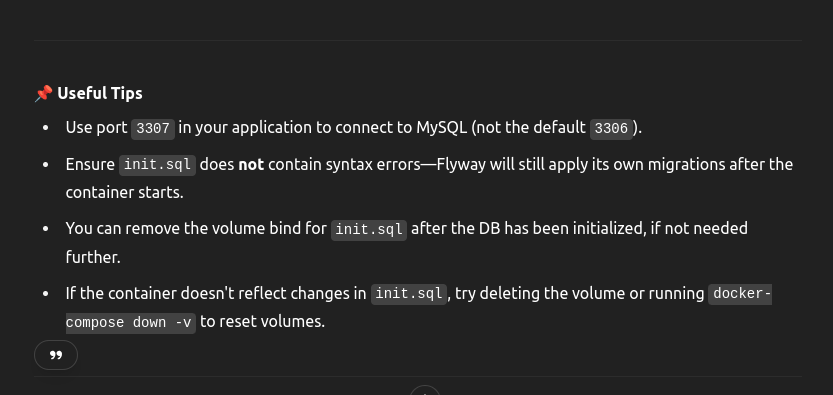
### **📄 docker-compose.yml Example**

version: '3.8'  
  
services:  
 mysql:  
 image: mysql:8.0  
 container\_name: mysql-container  
 environment:  
 MYSQL\_ROOT\_PASSWORD: root  
 MYSQL\_DATABASE: docker\_demo  
 MYSQL\_USER: testuser  
 MYSQL\_PASSWORD: testpass  
 ports:  
 - "3307:3306"  
 volumes:  
*# - mysql\_data:/var/lib/mysql*  
- ./mysql/data:/var/lib/mysql  
 - ./docker/mysql/init.sql:/docker-entrypoint-initdb.d/init.sql  
 networks:  
 - docker-app-network  
  
 springboot-app:  
 container\_name: springboot-container  
 build:  
 context: .  
 dockerfile: Dockerfile  
 depends\_on:  
 - mysql  
 ports:  
 - "9092:8080"  
 environment:  
 SPRING\_DATASOURCE\_URL: jdbc:mysql://mysql:3306/docker\_demo *# also use can 3307*  
SPRING\_DATASOURCE\_USERNAME: testuser  
 SPRING\_DATASOURCE\_PASSWORD: testpass  
 networks:  
 - docker-app-network  
  
volumes:  
 mysql\_dataa:  
  
networks:  
 docker-app-network:

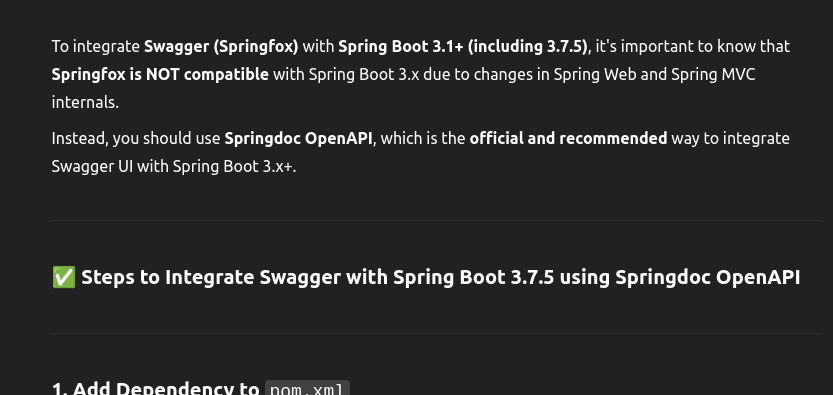
### **📌 Notes on Port Mapping**

* Format: "HOST\_PORT:CONTAINER\_PORT"
* MySQL default container port is 3306, so we change the **host port** (e.g. 3333) for each project to avoid conflicts.

run below command to check its going well or not :  
  
result of above command :  
  
  
OR this command :  


  
  
  
  
above in comps file inport 1st is host port and other is container port so when in local default port is3306 we need o change second one so we cam utilize docker host and we can change default port when we are working wit diff version of swl we will change second port and first docar host port will be same and oter will change so that we will not get same port n use error

## **📘 Swagger Config for Spring Boot**

  
Spring config class which you can can customise :

package com.example.demo.config;  
  
import io.swagger.v3.oas.models.OpenAPI;  
import io.swagger.v3.oas.models.info.Info;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
public class SwaggerConfig {  
  
 @Bean  
 public OpenAPI customOpenAPI() {  
 return new OpenAPI()  
 .info(new Info()  
 .title("Docker Demo Service API")  
 .version("1.0")  
 .description("API documentation for the Docker Demo Service"));  
 }  
}

**Maven Dependency:**

<dependency>  
 <groupId>org.springdoc</groupId>  
 <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  
 <version>2.5.0</version>  
</dependency>

Swagger URLs (Spring Boot):

* Swagger UI: <http://localhost:8081/swagger-ui/index.html>
* API Docs: <http://localhost:8081/v3/api-docs>

**Install portainer to see docker gui in linux** : **Install Portainer (Docker GUI)**

sudo docker volume create portainer\_data

sudo docker run -d -p 9000:9000 --name portainer \

--restart=always \

-v /var/run/docker.sock:/var/run/docker.sock \

-v portainer\_data:/data \

portainer/portainer-ce

Open Portainer UI: [http://localhost:9000](http://localhost:9000/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **📌 Best Practices**

* One image per container
* One DB per container; change host port if needed
* Run multiple containers with same internal port using different host ports
* Use Docker Compose for managing multiple services
* Use Portainer for GUI-based management
* One container multiple image not possible : it will give error
* Mysql in docker always run on 3306[default port] if you want to change need to change in mysql.cnf and after only above mentioned step container port will change
* Diff project and diff container for every image then port will diff
* Keep apps on host.[app or whatever you want run ]
* Just connect to Docker DBs using mapped **host ports[in Ui ment]**.
* So we need to change host port to where we want to run and container port will be default port always
* Ex: **Host ↔ Container Port Mapping fo**r **Dockerize You**r **DBs in spring app**  
  docket-compos ports:  
  "HOST\_PORT:CONTAINER\_PORT"  
    
   "3333:3306" # MySQL: Host port 3333 → Container port 3306[default]

"27018:27017" # MongoDB: Host 27018 → Container 27017 [default]

"9043:9042" # Cassandra: Host 9043 → Container 9042 [default]

application.properties

**MySQL**  
spring.datasource.url=jdbc:mysql://localhost:3333/db\_name  
 **MongoDB**  
spring.data.mongodb.uri=mongodb://localhost:27018/db\_name  
 **Cassandra**  
spring.data.cassandra.contact-points=localhost   
spring.data.cassandra.port=9043

## **Dockerize Your Spring Boot App:>**

## **⚙️ Dockerfile for Spring Boot App**

# 1. Use a base image with Java installed  
FROM openjdk:17-jdk-slim  
  
# 2. Set working directory  
WORKDIR /app  
  
# 3. Copy built JAR into container  
COPY target/\*.jar app.jar  
  
# 4. Expose Spring Boot port  
EXPOSE 8080  
  
# 5. Run the app  
ENTRYPOINT ["java", "-jar", "app.jar"]

Notes for docker file:

# 2. Set working directory inside the container  
WORKDIR /app

* All following commands will run inside the /app directory **in the container**.
* It's like cd /app.

# 3. Copy the built JAR file from your host into the container  
COPY target/\*.jar app.jar

* This assumes your Spring Boot app is **already built** (i.e., you've run mvn package).
* It copies the .jar file from the target/ directory on your local machine to /app/app.jar inside the container.

# 4. Expose the port Spring Boot runs on (inside container)  
EXPOSE 8080

* This tells Docker that your app listens on port 8080.
* Note: This does NOT publish the port to the host. That’s done in docker-compose.

# 5. Run the Spring Boot app when the container starts  
ENTRYPOINT ["java", "-jar", "app.jar"]

* This defines the **command to run** when the container starts.
* In this case, it's running the Spring Boot JAR using java -jar.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

* **🧪 Docker Commands**

### **🔹 General**

* docker --version – Show Docker version
* docker info – System-wide info

### **🔹 Images**

* docker build -t <name>:<tag> . – Build image
* docker pull <image> – Download from Docker Hub
* docker images – List images
* docker rmi <image> – Remove image

### **🔹 Containers**

* docker run <image> – Run image
* docker run -it <image> /bin/bash – Interactive shell
* docker run -d <image> – Detached mode
* docker ps – Running containers
* docker ps -a – All containers
* docker stop/start/restart <container> – Manage lifecycle
* docker rm <container> – Remove container

### **🔹 Volumes**

* docker volume ls – List volumes
* docker volume create <volume> – Create volume
* docker volume inspect <volume> – Volume details
* docker volume rm <volume> – Delete volume
* docker volume prune – Remove unused volumes

### **🔹 Logs & Inspect**

* docker logs -f <container> – Follow logs
* docker inspect <container> – Inspect details

### **🔹 Networks**

* docker network ls – List networks

### **🔹 Compose**

* docker-compose up -d – Start in background
* docker-compose down – Stop and clean up
* docker-compose logs -f – Follow service logs
* docker-compose ps – View service states

### **🔹 Cleanup**

* docker system prune – Remove all unused objects
* docker image prune – Remove unused images

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_---

ref link :  
<https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository>

<https://docs.docker.com/desktop/setup/install/linux/#general-system-requirements>

<http://localhost:9000/#!/3/docker/containers/75e1795799bb2136123148b0eccd995f53932b7e0a1ee8925682e446e6062dbc/logs>

<https://www.youtube.com/watch?v=vdAkGFCq-M4&list=PLSVW22jAG8pDeU80nDzbUgr8qqzEMppi8&index=2>

<http://localhost:8081/v3/api-docs>

<http://localhost:8081/swagger-ui/index.html#/Order%20Controller/placeOrder>