

Scale services in a Docker Compose application using docker-compose scale

Table of Contents

- [Description](#)
- [Problem Statement](#)
- [Prerequisites](#)
 - [Software Requirement](#)
 - [Hardware Requirement](#)
- [Implementation Steps](#)
 - [Step-1 :: Create Standard Folder Structure](#)
 - [Step-2 :: Write docker-compose.yml](#)
 - [Step-3 :: Deploy the Multi-Container Application](#)
 - [Step-4 :: Scale Services in the Docker Compose Application](#)
- [References](#)

Description

This section explains how to scale services within a **Docker Compose** application. Scaling a service means running multiple instances of that service, which is useful for load balancing or handling increased traffic. In this case, we will demonstrate how to scale the **Java TodoApp** service while leaving the **MySQL** service as a single instance.

Problem Statement

Scaling is required when there is a need to handle a higher volume of requests. For example, the **Java TodoApp** may need to handle more users simultaneously, so additional instances of the app should be spun up. With Docker Compose, you can easily scale a service by specifying the number of container instances you want to run.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-07) is required before proceeding with Lab Guide-08.

Software Requirement

- **Docker Desktop**: Installed on your Windows machine.
- **Docker Compose**: Included in Docker Desktop.
- **TodoAPP_MYSQL**: To download the source folder [click here](#)

Hardware Requirement

- Minimum of 4 GB RAM
- At least 2 cores in the processor
- Additional CPU and RAM may be required based on the number of scaled instances

Implementation Steps

Step-1 :: Create Standard Folder Structure

Ensure your folder structure and the **docker-compose.yml** file are correctly set up.

Step-2 :: Write docker-compose.yml

The **docker-compose.yml** file defines the services and their configurations. Here's the multi-container configuration for the **Java TodoApp** and **MySQL**:

```
services:
  # MySQL Database Service
  db:
    image: mysql
    container_name: mysql_db
    environment:
      MYSQL_ROOT_PASSWORD: P@ssw0rd
      MYSQL_DATABASE: tododb
      MYSQL_PASSWORD: P@ssw0rd
    ports:
      - "3306:3306"
    networks:
      - todoapp_network
    volumes:
      - db_data:/var/lib/mysql

  # Java TodoApp Service
  todoapp:
    build:
      context: ./todoapp
    container_name: todoapp
    ports:
      - "8081"
    depends_on:
      - db
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://db:3306/tododb
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: P@ssw0rd
    networks:
      - todoapp_network

networks:
  todoapp_network:
    driver: bridge
```

```
volumes:  
  db_data:
```

Note: To scale the todoapp service, you need to remove the static port mapping (8081:8081) for the containers. Instead, let Docker assign random dynamic ports on the host for each container while keeping the internal application port the same.

Step-3 :: Deploy the Multi-Container Application

To deploy the application with Docker Compose, run:

```
docker-compose up --build
```

This will start both the **TodoApp** and **MySQL** containers.

Step-4 :: Scale Services in the Docker Compose Application

To handle increased load on your **Java TodoApp**, you can scale the **todoapp** service to run multiple containers.

1. Scaling the TodoApp Service:

To scale the **TodoApp** service to 3 instances, run:

```
docker-compose up --scale todoapp=3
```

This command will:

- Spin up 3 instances of the **todoapp** service.
- Automatically assign different ports to each new instance (if necessary).
- Ensure all instances are connected to the same network and can interact with the **MySQL** database.



2. Verifying Scaled Containers:

You can check the running instances using the following command:

```
docker-compose ps
```

The output should show multiple **todoapp** containers running:



3. **Scaling Down:**

To reduce the number of running instances (e.g., scaling down to 1 instance):

```
docker-compose up --scale todoapp=1
```

This will stop and remove the extra instances while keeping one instance running.



References

- Docker Compose official documentation: <https://docs.docker.com/compose/>
- Scaling services with Docker Compose: [Docker Compose Scaling](#)