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

Table of Contents

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.

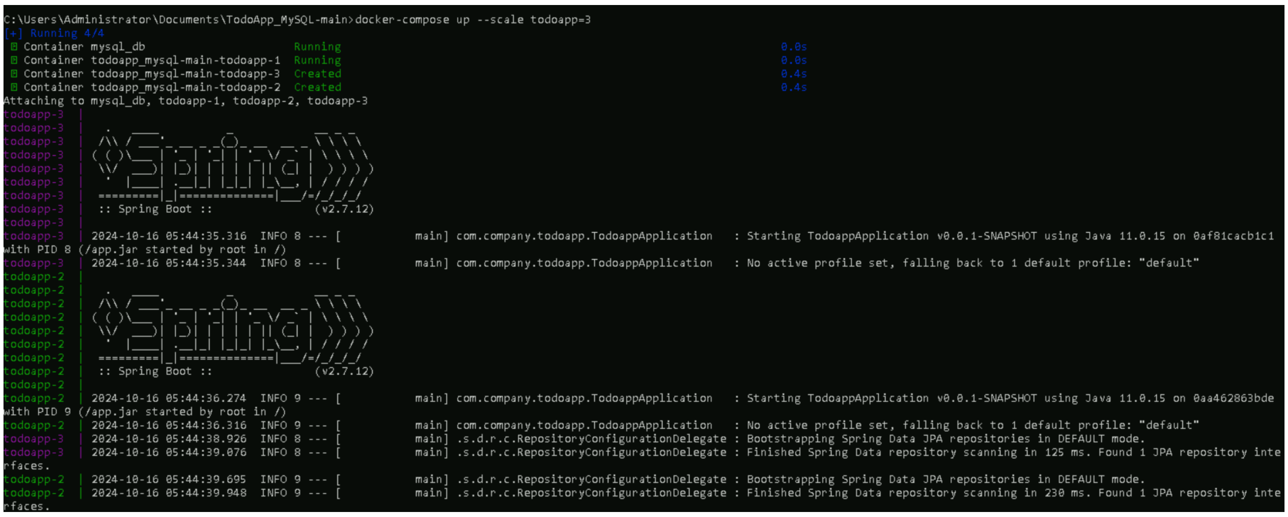
1. **Scaling the TodoApp Service**:

   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.



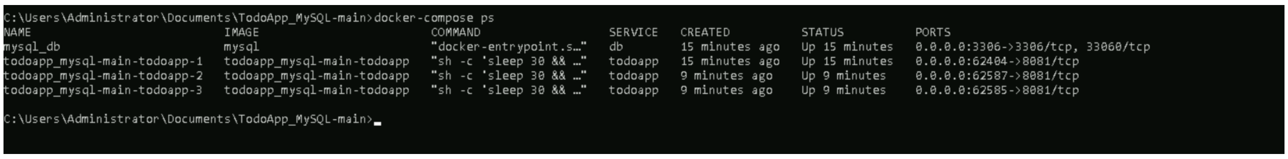
```
C:\Users\Administrator\Documents\TodoApp_MySQL-main>docker-compose up --scale todoapp=3
[+] Running 4/4
 ✔ Container mysql_db                      Running                                                   0.0s
 ✔ Container todoapp_mysql-main-todoapp-1  Running                                                   0.0s
 ✔ Container todoapp_mysql-main-todoapp-3  Created                                                   0.4s
 ✔ Container todoapp_mysql-main-todoapp-2  Created                                                   0.4s
Attaching to mysql_db, todoapp-1, todoapp-2, todoapp-3
todoapp-3  |
todoapp-3  |   .   ____          _            __ _ _
todoapp-3  |  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
todoapp-3  | ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
todoapp-3  |  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
todoapp-3  |   '  |____| .__|_| |_|_| |_\__, | / / / /
todoapp-3  |  =========|_|==============|___/=/_/_/_/
todoapp-3  |  :: Spring Boot ::               (v2.7.12)
todoapp-3  |
todoapp-3  | 2024-10-16 05:44:35.316  INFO 8 --- [           main] com.company.todoapp.TodoappApplication   : Starting TodoappApplication v0.0.1-SNAPSHOT using Java 11.0.15 on 0af81cacb1c1
with PID 8 (/app.jar started by root in /)
todoapp-3  | 2024-10-16 05:44:35.344  INFO 8 --- [           main] com.company.todoapp.TodoappApplication   : No active profile set, falling back to 1 default profile: "default"
todoapp-2  |
todoapp-2  |   .   ____          _            __ _ _
todoapp-2  |  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
todoapp-2  | ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
todoapp-2  |  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
todoapp-2  |   '  |____| .__|_| |_|_| |_\__, | / / / /
todoapp-2  |  =========|_|==============|___/=/_/_/_/
todoapp-2  |  :: Spring Boot ::               (v2.7.12)
todoapp-2  |
todoapp-2  | 2024-10-16 05:44:36.274  INFO 9 --- [           main] com.company.todoapp.TodoappApplication   : Starting TodoappApplication v0.0.1-SNAPSHOT using Java 11.0.15 on 0aa462863bde
with PID 9 (/app.jar started by root in /)
todoapp-2  | 2024-10-16 05:44:36.316  INFO 9 --- [           main] com.company.todoapp.TodoappApplication   : No active profile set, falling back to 1 default profile: "default"
todoapp-3  | 2024-10-16 05:44:38.926  INFO 8 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
todoapp-3  | 2024-10-16 05:44:39.076  INFO 8 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 125 ms. Found 1 JPA repository inte
rfaces.
todoapp-2  | 2024-10-16 05:44:39.695  INFO 9 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
todoapp-2  | 2024-10-16 05:44:39.948  INFO 9 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 230 ms. Found 1 JPA repository inte
rfaces.
```

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:



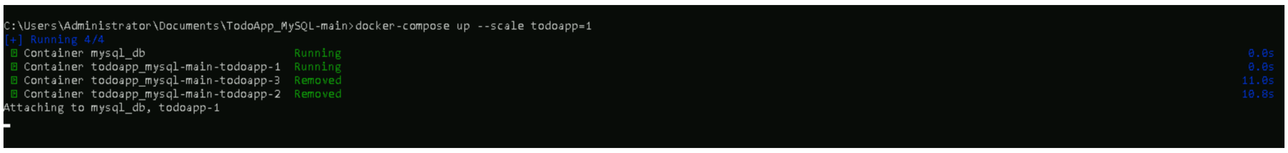
```
C:\Users\Administrator\Documents\TodoApp_MySQL-main>docker-compose ps
NAME                           IMAGE                        COMMAND                  SERVICE    CREATED          STATUS          PORTS
mysql_db                       mysql                        "docker-entrypoint.s…"   db         15 minutes ago   Up 15 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp
todoapp_mysql-main-todoapp-1   todoapp_mysql-main-todoapp   "sh -c 'sleep 30 && …"   todoapp    15 minutes ago   Up 15 minutes   0.0.0.0:62404->8081/tcp
todoapp_mysql-main-todoapp-2   todoapp_mysql-main-todoapp   "sh -c 'sleep 30 && …"   todoapp    9 minutes ago    Up 9 minutes    0.0.0.0:62587->8081/tcp
todoapp_mysql-main-todoapp-3   todoapp_mysql-main-todoapp   "sh -c 'sleep 30 && …"   todoapp    9 minutes ago    Up 9 minutes    0.0.0.0:62585->8081/tcp

C:\Users\Administrator\Documents\TodoApp_MySQL-main>_
```

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.



```
C:\Users\Administrator\Documents\TodoApp_MySQL-main>docker-compose up --scale todoapp=1
[+] Running 4/4
 ✔ Container mysql_db                      Running                                                   0.0s
 ✔ Container todoapp_mysql-main-todoapp-1  Running                                                   0.0s
 ✔ Container todoapp_mysql-main-todoapp-3  Removed                                                  11.0s
 ✔ Container todoapp_mysql-main-todoapp-2  Removed                                                  10.8s
Attaching to mysql_db, todoapp-1
_
```

References

- Docker Compose official documentation: https://docs.docker.com/compose/
- Scaling services with Docker Compose: Docker Compose Scaling