Docker 4:

What’s alternative for docker-compose? it is kubernetes

[ec2-user@ip-172-31-19-227 ~]$ git clone https://github.com/Haider7214/springboot-mysql-docker-compose.git

[ec2-user@ip-172-31-19-227 ~]$ ls -l

total 0

drwxr-xr-x. 3 ec2-user ec2-user 32 May 7 00:13 demo-webapp

drwxr-xr-x. 5 ec2-user ec2-user 83 May 6 02:43 my-webapp

drwxr-xr-x. 4 ec2-user ec2-user 40 May 11 20:32 SpringBootApp

drwxr-xr-x. 4 ec2-user ec2-user 58 May 13 23:55 springboot-mysql-docker-compose

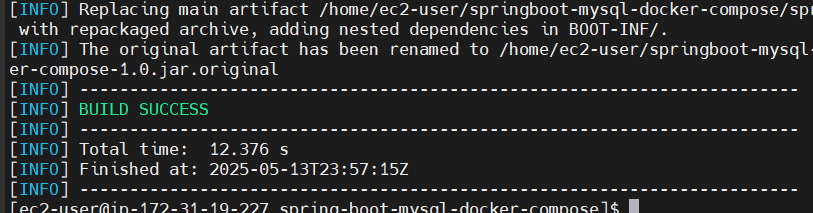
drwxr-xr-x. 5 ec2-user ec2-user 118 May 7 00:48 SpringSecurity\_JWT

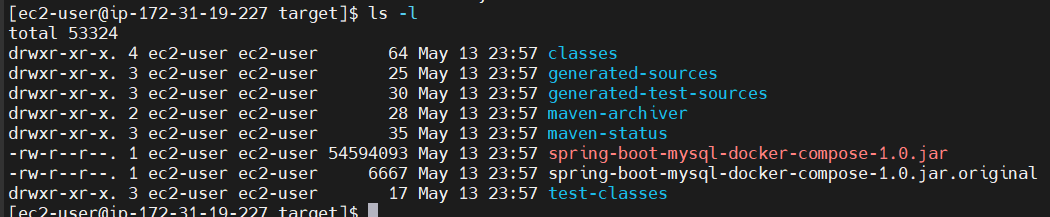
drwxr-xr-x. 4 ec2-user ec2-user 64 May 11 17:54 try-webapp

drwxr-xr-x. 4 ec2-user ec2-user 36 May 13 02:30 WebappCRM

[ec2-user@ip-172-31-19-227 spring-boot-mysql-docker-compose]$ mvn clean package

[ec2-user@ip-172-31-19-227 spring-boot-mysql-docker-compose]$ mvn clean package -DskipTests





[ec2-user@ip-172-31-19-227 target]$ vi Dockerfile

[ec2-user@ip-172-31-19-227 target]$ cat Dockerfile

FROM openjdk:17

EXPOSE 8080

COPY target/spring-boot-mysql-docker-compose-1.0.jar spring-boot-mysql-docker-compose-1.0.jar

ENTRYPOINT ["java", "-jar", "/spring-boot-mysql-docker-compose-1.0.jar"]

[ec2-user@ip-172-31-19-227 target]$ vi docker-compose.yml

[ec2-user@ip-172-31-19-227 target]$ cat docker-compose.yml

version: "3.8"

services:

musqldb:

image: mysql:8.0

ports:

- "3306:3306"

environments:

- MYSQL\_ROOT\_PASSWORD: root

- MYSQL\_DATABASE: sbm

networks:

- springboot-db-net

application:

build: .

depends\_on:

mysqldb:

ports:

- "8080:8080"

networks:

- springboot-db-net

volumes:

- /data/springboot-app

networks:

springboot-db-net:

[ec2-user@ip-172-31-19-227 springboot-mysql-docker-compose]$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

try-webapp latest fcd71f541cea 2 days ago 468MB

springbootapp latest f06e88a7ddef 2 days ago 497MB

[ec2-user@ip-172-31-19-227 springboot-mysql-docker-compose]$ docker system prune -a

[ec2-user@ip-172-31-19-227 spring-boot-mysql-docker-compose]$ ls -l

total 32

-rw-r--r--. 1 ec2-user ec2-user 503 May 14 00:17 docker-compose.yml

-rw-r--r--. 1 ec2-user ec2-user 10665 May 13 23:55 mvnw

-rw-r--r--. 1 ec2-user ec2-user 7061 May 13 23:55 mvnw.cmd

-rw-r--r--. 1 ec2-user ec2-user 2051 May 13 23:55 pom.xml

drwxr-xr-x. 4 ec2-user ec2-user 30 May 13 23:55 src

drwxr-xr-x. 8 ec2-user ec2-user 4096 May 14 00:08 target

[ec2-user@ip-172-31-19-227 spring-boot-mysql-docker-compose]$ docker-compose up -d

Updated docker-compose.yml

[ec2-user@ip-172-31-19-227 spring-boot-mysql-docker-compose]$ cat docker-compose.yml

version: "3.8"

services:

mysqldb:

image: mysql:8.0

ports:

- "3306:3306"

environment:

MYSQL\_ROOT\_PASSWORD: root

MYSQL\_DATABASE: sbm

healthcheck:

test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]

interval: 10s

timeout: 5s

retries: 5

networks:

- springboot-db-net

application:

build: .

depends\_on:

mysqldb:

condition: service\_healthy

ports:

- "8080:8080"

networks:

- springboot-db-net

volumes:

- /data/springboot-app

networks:

springboot-db-net:

[ec2-user@ip-172-31-19-227 spring-boot-mysql-docker-compose]$ docker-compose up -d

[ec2-user@ip-172-31-19-227 spring-boot-mysql-docker-compose]$ docker-compose ps

Name Command State Ports

---------------------------------------------------------------------------------------------------------------------------------------------

spring-boot-mysql-docker-compose\_mysqldb\_1 docker-entrypoint.sh mysqld Up (healthy) 0.0.0.0:3306->3306/tcp,:::3306->3306/tcp, 33060/tcp

[ec2-user@ip-172-31-19-227 spring-boot-mysql-docker-compose]$ docker-compose up -d

Starting spring-boot-mysql-docker-compose\_mysqldb\_1 ... done

Starting spring-boot-mysql-docker-compose\_application\_1 ... done

Docker compose file

Clone the project from Github repo

git clone <https://github.com/Haider7214/springboot-mysql-docker-compose.git>

cd springboot-mysql-docker-compose

Mvn clean package -DskipTest

ls -l target (.jar file will be available)

Create one Dockerfile

vi Dockerfile

FROM openjdk:17

EXPOSE 8080

COPY target/spring-boot-mysql-docker-compose-1.0.jar spring-boot-mysql-docker-compose-1.0.jar

ENTRYPOINT ["java", "-jar", "/spring-boot-mysql-docker-compose-1.0.jar"]

docker-compose up -d ---> Create docker containers using docker-compose

docker-compose ps ---> Check docker containers running or not

docker-compose stop ---> to stop the docker containers

docker-compose start --->

docker-compose down ---> delete docker containers using docker-compose

Stateful containers: Data will be there permanently

Stateless containers: it will not recollect what has happened. Data will be deleted after container deletion

Note: Docker containers by default are stateless

Example: In our spring-boot-mysql-docker-compose app, we used MySQL as docker container to store data and when we stop the containers and re-created these containers we lost the data

To store application data permanently in this case we may have to make docker container as stateful and hence we need to use Docker volumes

Docker Volumes:

Volumes are used to persist data, which is generated by docker container and to avoid data loss.

With the help of docker volumes we can make our containers stateful

[ec2-user@ip-172-31-19-227 ~]$ docker volume ls

DRIVER VOLUME NAME

local 4de4ef265b2a67d1ccb84e0ddf1fa0fc55454dff5808c032e911a39d2935167d

local 41dbdcdaa0404f328aaa6960b467a43c46ed86f3b895aa67bb8479cfcdf11304

local 86983f6c81527b0612934d6bd4f98784b142cc0016659fc27281864ae6148dbd

local b936e5199f3e27d2657ad8010ed846a3177135105ecb49ac6d9941883f477bdc

local bd19e40d4ca00138523a0f5c98bc2f25fb47eabb73b5d895a756137c9613ba5c

local c9fb37b4b4ab19d104cb29a2bec665245d8e1fdb1534952fd6698c717a84b263

[ec2-user@ip-172-31-19-227 ~]$

[ec2-user@ip-172-31-19-227 ~]$ docker volume create demo-volume

demo-volume

[ec2-user@ip-172-31-19-227 ~]$ docker volume ls

DRIVER VOLUME NAME

local 4de4ef265b2a67d1ccb84e0ddf1fa0fc55454dff5808c032e911a39d2935167d

local 41dbdcdaa0404f328aaa6960b467a43c46ed86f3b895aa67bb8479cfcdf11304

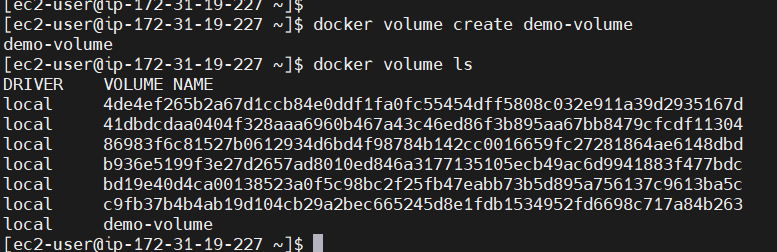
local 86983f6c81527b0612934d6bd4f98784b142cc0016659fc27281864ae6148dbd

local b936e5199f3e27d2657ad8010ed846a3177135105ecb49ac6d9941883f477bdc

local bd19e40d4ca00138523a0f5c98bc2f25fb47eabb73b5d895a756137c9613ba5c

local c9fb37b4b4ab19d104cb29a2bec665245d8e1fdb1534952fd6698c717a84b263

local demo-volume



[ec2-user@ip-172-31-19-227 ~]$ docker volume inspect demo-volume

[

{

"CreatedAt": "2025-05-14T02:57:21Z",

"Driver": "local",

"Labels": null,

"Mountpoint": "/var/lib/docker/volumes/demo-volume/\_data",

"Name": "demo-volume",

"Options": null,

"Scope": "local"

}

]

53:30