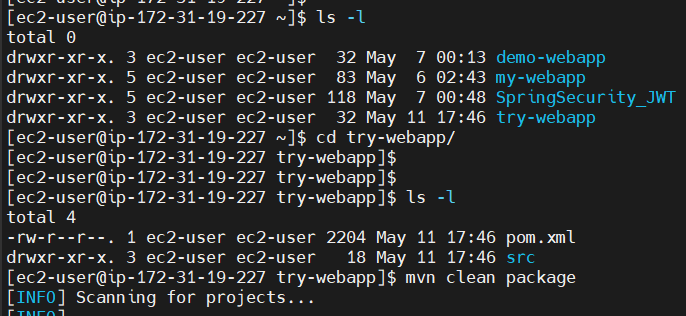
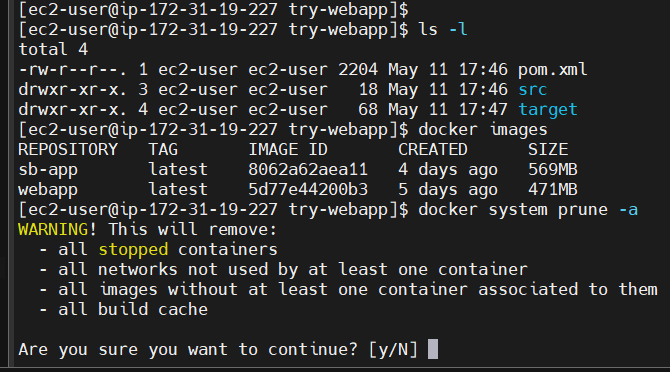
Docker 3:

mvn archetype:generate -DgroupId=com.example -DartifactId=demo-webapp -DarchetypeArtifactId=maven-archetype-webapp -DarchetypeVersion=1.4 -DinteractiveMode=false

[ec2-user@ip-172-31-19-227 ~]$ mvn archetype:generate -DgroupId=com.example -DartifactId=try-webapp -DarchetypeArtifactId=maven-archetype-webapp -DarchetypeVersion=1.4 -DinteractiveMode=false





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

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

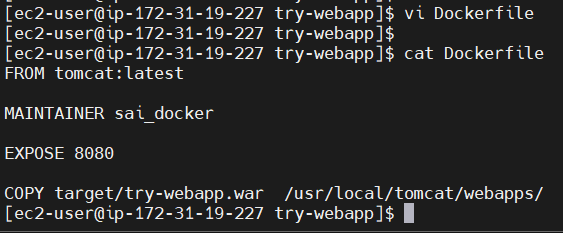
[ec2-user@ip-172-31-19-227 try-webapp]$ cat Dockerfile

FROM tomcat:latest

MAINTAINER sai\_docker

EXPOSE 8080

COPY target/try-webapp.war /usr/local/tomcat/webapps/

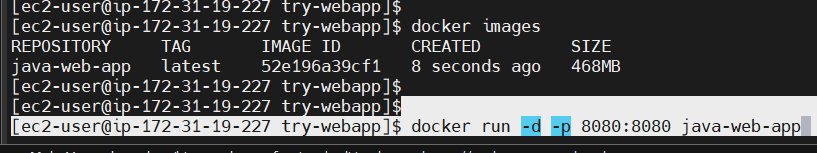


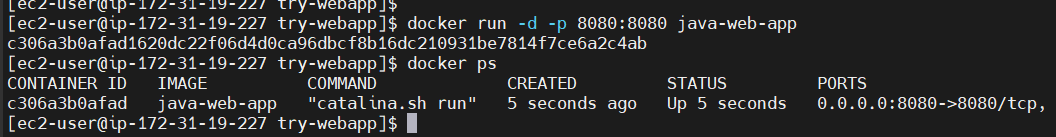
[ec2-user@ip-172-31-19-227 try-webapp]$ docker build -t java-web-app .

REPOSITORY TAG IMAGE ID CREATED SIZE

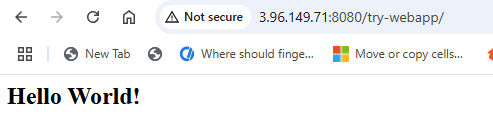
java-web-app latest 52e196a39cf1 8 seconds ago 468MB

[ec2-user@ip-172-31-19-227 try-webapp]$ docker run -d -p 8080:8080 java-web-app





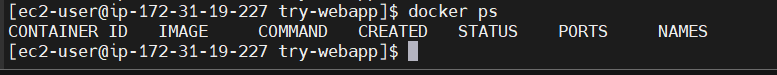
<http://3.96.149.71:8080/try-webapp/>



[ec2-user@ip-172-31-19-227 try-webapp]$ docker logs c306a3b0afad

[ec2-user@ip-172-31-19-227 try-webapp]$ docker stop c306a3b0afad

c306a3b0afad



Create a Maven Java web application or clone the existing Java web app from Github

mvn archetype:generate -DgroupId=com.example -DartifactId=try-webapp -DarchetypeArtifactId=maven-archetype-webapp -DarchetypeVersion=1.4 -DinteractiveMode=false

Go inside that web app

cd my-web-app

mvn clean package --> to create target folder and war file inside it

Create a Dockerfile

vi Dockerfile --->

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

FROM tomcat:latest

MAINTAINER sai\_docker

EXPOSE 8080

COPY target/try-webapp.war /usr/local/tomcat/webapps/

docker build -t java-web-app

docker images

docker run -d -p 8080:8080 java-web-app

==> Enable 8080 in security group of host machine and using public IP of the host machone try to access the web app from the browser

<public IP>:8080/my-web-app/ to see the result

Dockerizing Springboot application:

Springboot is one of the frameworks to develop Java-based applications

Tomcat will already be there inside Springboot application

Springboot internally uses one embedded-tomcat server

So when using Springboot we don’t need Tomcat server separately

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

git version 2.43.5

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

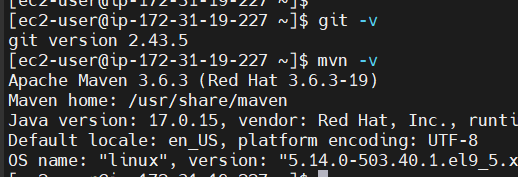
Apache Maven 3.6.3 (Red Hat 3.6.3-19)

Maven home: /usr/share/maven

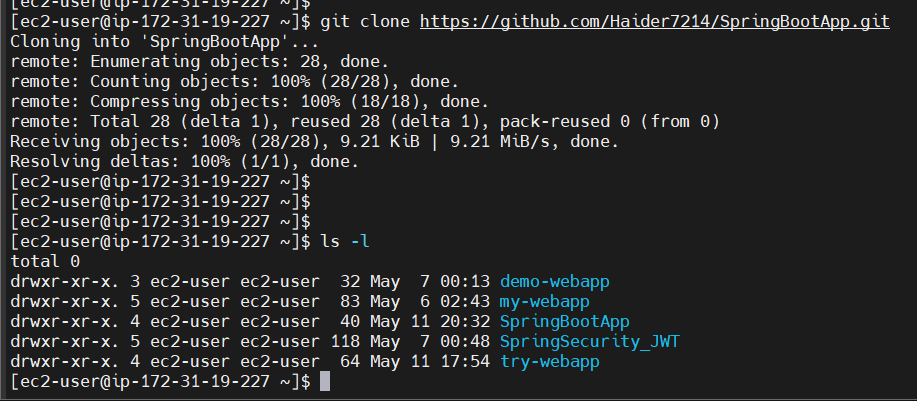
Java version: 17.0.15, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-17-openjdk-17.0.15.0.6-2.el9.x86\_64

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "5.14.0-503.40.1.el9\_5.x86\_64", arch: "amd64", family: "unix"



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



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

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

total 0

drwxr-xr-x. 4 ec2-user ec2-user 112 May 11 20:32 WebAppProject1

[ec2-user@ip-172-31-19-227 SpringBootApp]$ cd WebAppProject1/

[ec2-user@ip-172-31-19-227 WebAppProject1]$ ls -l

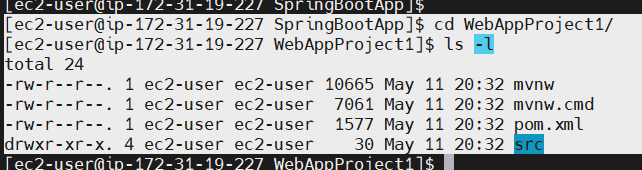
total 24

-rw-r--r--. 1 ec2-user ec2-user 10665 May 11 20:32 mvnw

-rw-r--r--. 1 ec2-user ec2-user 7061 May 11 20:32 mvnw.cmd

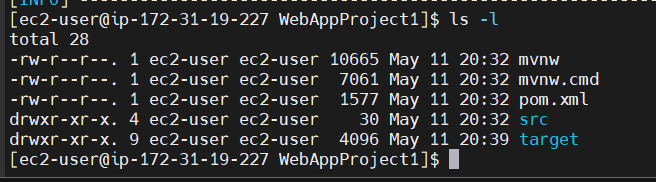
-rw-r--r--. 1 ec2-user ec2-user 1577 May 11 20:32 pom.xml

drwxr-xr-x. 4 ec2-user ec2-user 30 May 11 20:32 src



There is no target folder, how do we get target folder?

[ec2-user@ip-172-31-19-227 WebAppProject1]$ mvn clean package



Now we see the target folder

[ec2-user@ip-172-31-19-227 WebAppProject1]$ cat Dockerfile

FROM openjdk:17

MAINTAINER sai\_docker

COPY target/WebAppProject1-0.0.1.jar /usr/app/

WORKDIR /usr/app/

EXPOSE 8080

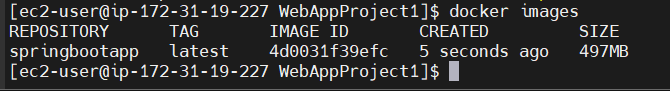
ENTRYPOINT ["java", "-jar", "WebAppProject1-0.0.1.jar"]

[ec2-user@ip-172-31-19-227 WebAppProject1]$ docker build -t springbootapp .

[ec2-user@ip-172-31-19-227 WebAppProject1]$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

springbootapp latest 4d0031f39efc 5 seconds ago 497MB

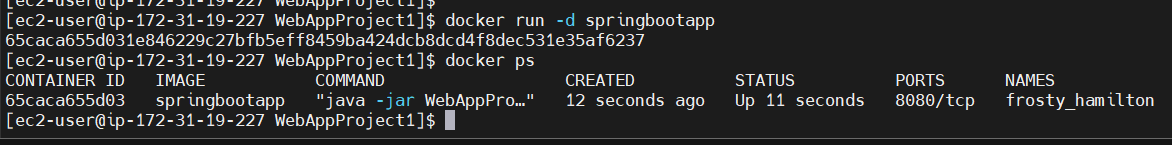


[ec2-user@ip-172-31-19-227 WebAppProject1]$ docker run -d springbootapp

[ec2-user@ip-172-31-19-227 WebAppProject1]$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

65caca655d03 springbootapp "java -jar WebAppPro…" 12 seconds ago Up 11 seconds 8080/tcp frosty\_hamilton



If we have to manage multiple containers, that’s where docker-compose comes into picture

[ec2-user@ip-172-31-19-227 WebAppProject1]$ docker stop 65caca655d03

65caca655d03

Dockerizing SpringBoot App

Clone existing SpringBoot app from Github

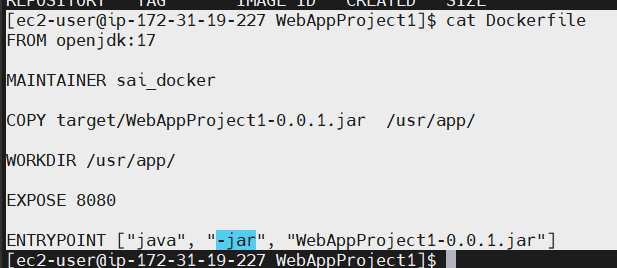
(Make sure git and maven are installed)

git clone <repo-link>

git clone https://github.com/Haider7214/SpringBootApp.git

cd SpringBootApp

mvn clean package (Create a target folder)



[ec2-user@ip-172-31-19-227 SpringBootApp]$ docker run -d -p 8080:8080 springbootapp .

Later we stop the container and delete if not in use

Summary:

Application Architecture

Tech Stack

Application Environments

Challenges in app deployment process

Need for Docker

Containerization and Docker

Docker Architecture

Dockerfile, Docker image, Docker registry (Docker hub), Docker containers

Dockerfile --> Keyword

Docker port mapping and Detached mode

Dockerizing Java web application (.war)

Dockerizing Java Springboot app (.jar) file

Docker Network:

Say one container wants to communicate to another container. Say SpringBoot container wants to communicate to DB (MySQL). that’s where docker network comes into picture. C1 is container 1 and C2 is container 2

C1

C2

Docker server

--> Network is all about communication and Docker network is used to provide isolated network for containers

--> If we run two containers under same network then these two containers can communicate with each other

--> By default, 3 default networks are available in Docker -> bridge, host, none

None means no network is available

Whenever we have standalone containers, then we go with the bridge network

Out of three, Bridge network is the default network for docker container --> it is used to run standalone containers and it will assign one IP for container

Host network is also used to run standalone containers and it will not assign an IP for our containers

None refers to no network will be available for containers

Docker Networking - Other Network types

Overlay network (used for Orchestration purpose) and Macvlan network (assign a real physical IP address from your LAN to container). if you want to assign physical IP address to your container then use Macvlan network

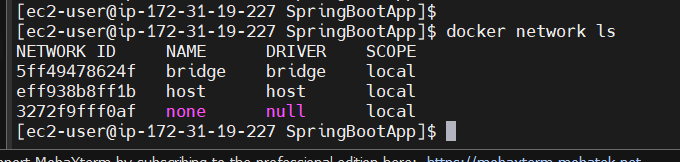
[ec2-user@ip-172-31-19-227 SpringBootApp]$ docker network ls

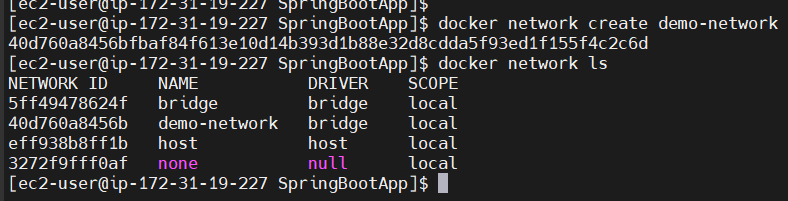
NETWORK ID NAME DRIVER SCOPE

5ff49478624f bridge bridge local

eff938b8ff1b host host local

3272f9fff0af none null local





Default configuration is there

[ec2-user@ip-172-31-19-227 SpringBootApp]$ docker network inspect demo-network

[

{

"Name": "demo-network",

"Id": "40d760a8456bfbaf84f613e10d14b393d1b88e32d8cdda5f93ed1f155f4c2c6d",

"Created": "2025-05-11T21:53:17.844037279Z",

"Scope": "local",

"Driver": "bridge",

"EnableIPv4": true,

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": {},

"Config": [

{

"Subnet": "172.18.0.0/16",

"Gateway": "172.18.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

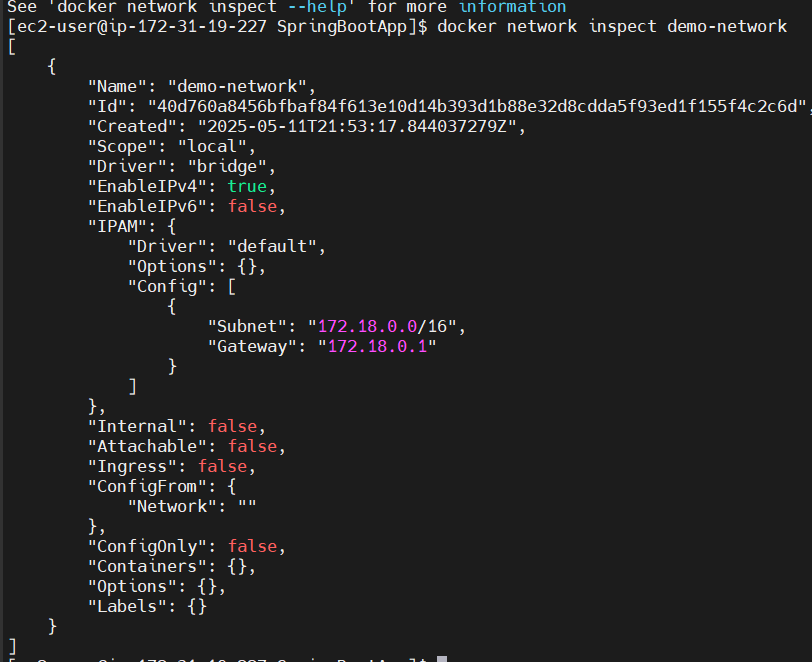
"Containers": {},

"Options": {},

"Labels": {}

}

]



[ec2-user@ip-172-31-19-227 try-webapp]$ docker network ls

NETWORK ID NAME DRIVER SCOPE

5ff49478624f bridge bridge local

40d760a8456b demo-network bridge local

eff938b8ff1b host host local

efe0612c9047 new-demo-network bridge local

3272f9fff0af none null local

[ec2-user@ip-172-31-19-227 try-webapp]$ docker images

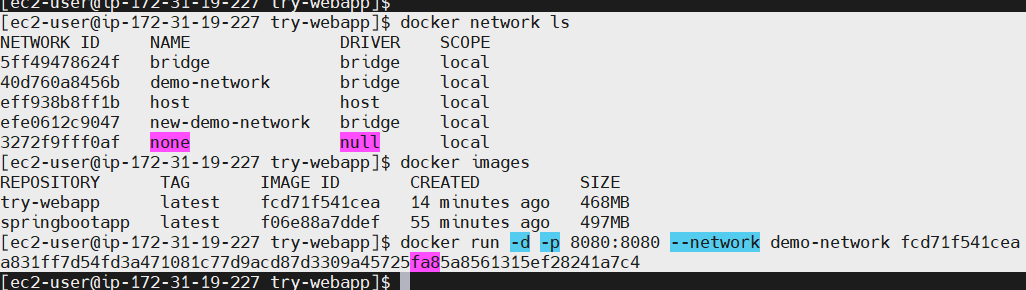
REPOSITORY TAG IMAGE ID CREATED SIZE

try-webapp latest fcd71f541cea 14 minutes ago 468MB

springbootapp latest f06e88a7ddef 55 minutes ago 497MB

[ec2-user@ip-172-31-19-227 try-webapp]$ docker run -d -p 8080:8080 --network demo-network fcd71f541cea

a831ff7d54fd3a471081c77d9acd87d3309a45725fa85a8561315ef28241a7c4



[ec2-user@ip-172-31-19-227 try-webapp]$ docker network inspect demo-network

[

{

"Name": "demo-network",

"Id": "40d760a8456bfbaf84f613e10d14b393d1b88e32d8cdda5f93ed1f155f4c2c6d",

"Created": "2025-05-11T21:53:17.844037279Z",

"Scope": "local",

"Driver": "bridge",

"EnableIPv4": true,

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": {},

"Config": [

{

"Subnet": "172.18.0.0/16",

"Gateway": "172.18.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {

"a831ff7d54fd3a471081c77d9acd87d3309a45725fa85a8561315ef28241a7c4": {

"Name": "nervous\_brown",

"EndpointID": "a19d785714632d9682f6846dc1c47f18242a78aace0bb4ffb5a916e55cc18714",

"MacAddress": "86:4c:f8:dc:d7:47",

"IPv4Address": "172.18.0.2/16",

"IPv6Address": ""

}

},

"Options": {},

"Labels": {}

}

]

In this network we can see a container running

"Containers": {

"a831ff7d54fd3a471081c77d9acd87d3309a45725fa85a8561315ef28241a7c4": {

"Name": "nervous\_brown",

"EndpointID": "a19d785714632d9682f6846dc1c47f18242a78aace0bb4ffb5a916e55cc18714",

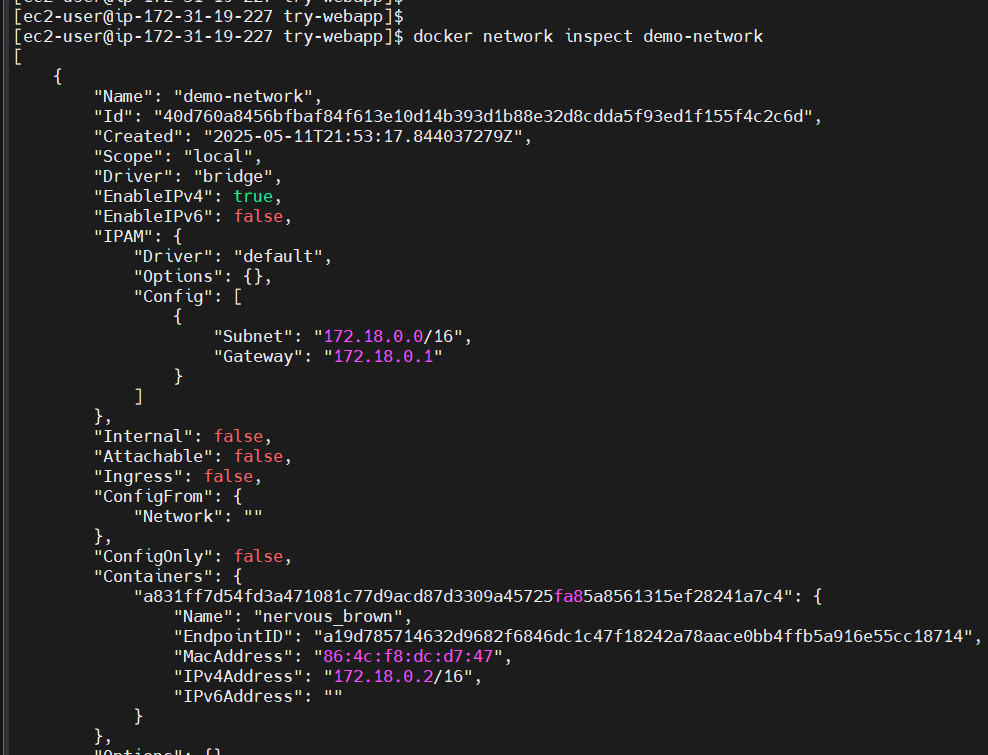
"MacAddress": "86:4c:f8:dc:d7:47",

"IPv4Address": "172.18.0.2/16",

"IPv6Address": ""

}

},



[ec2-user@ip-172-31-19-227 try-webapp]$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

a831ff7d54fd fcd71f541cea "catalina.sh run" 2 minutes ago Up 2 minutes 0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp nervous\_brown

[ec2-user@ip-172-31-19-227 try-webapp]$ docker stop a831ff7d54fd

a831ff7d54fd

[ec2-user@ip-172-31-19-227 try-webapp]$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

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

[ec2-user@ip-172-31-19-227 try-webapp]$ docker network ls

NETWORK ID NAME DRIVER SCOPE

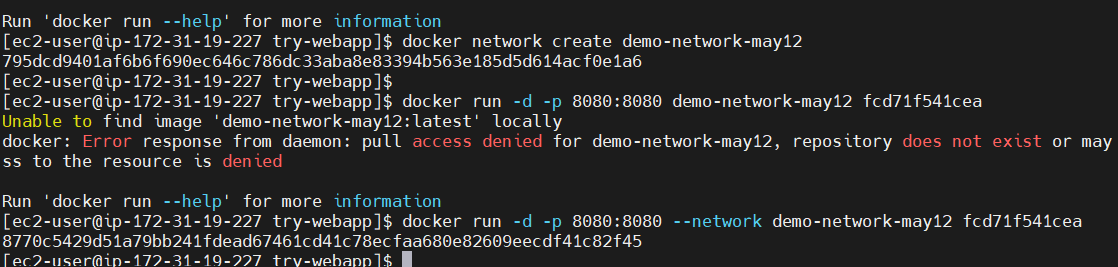
8347e80f315b bridge bridge local

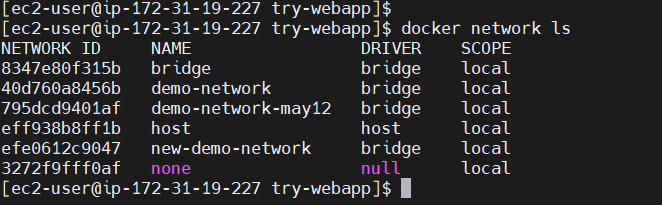
40d760a8456b demo-network bridge local

eff938b8ff1b host host local

efe0612c9047 new-demo-network bridge local

3272f9fff0af none null local





Frontend container

Backend container

Database container

Microservices --> Multiple containers will be there

If you want to manage multiple containers then we need Docker-compose. When we have multiple containers, concept of docker-compose comes into picture

Docker-compose:

Docker-compose is a tool for running multiple container Docker applications.

Usually in a microservice application there will be multiple APIs and for every API we may have to create separate containers in that case. In that case, managing many containers would be difficult (create / start / stop containers), to overcome this problem we have Docker-compose. In Docker-compose using single command, we can create / start / stop multiple containers

It uses yaml file to configure the application services

docker-compose.yml is used to specify container information

Default and recommended name is docker-compose.yml (we can change it if required)

There will be 4 sections within docker-compose.yml file

Sections: Version (represents compose yml version), Services (represents container information image-name, port-mapping, …) , Network (represents docker network to run our containers), Volumes (represents container storage location)

[ec2-user@ip-172-31-19-227 try-webapp]$ docker -v

Docker version 28.1.1, build 4eba377

[ec2-user@ip-172-31-19-227 try-webapp]$ docker-compose --version

-bash: docker-compose: command not found

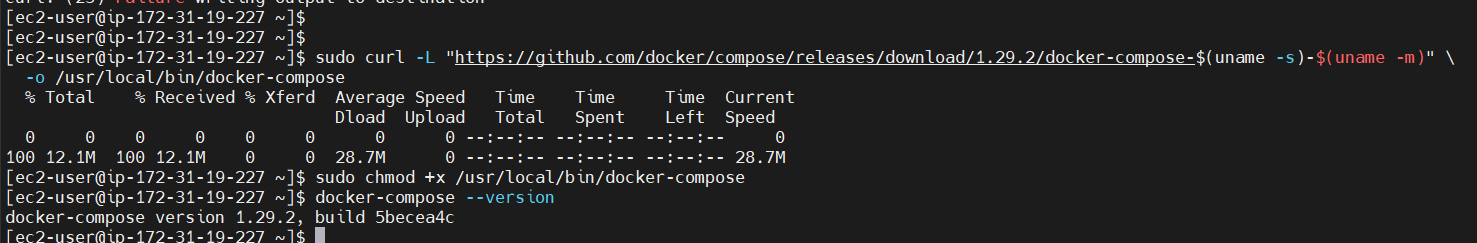
<https://docs.docker.com/compose/install/linux/>

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" \

-o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version



Docker compose file

docker-compose.yml

version: “3”

services:

application:

image: springboot-app

ports:

- “8080:8080”

networks:

- compose-network

depends\_on:

- mysqldb

volumes:

- /data/springboot-app

mysqldb:

image: mysql:8.3

networks:

- compose-network

ports:

- “3306:3306”

environment:

- MYSQL\_ROOT\_PASSWORD=root

- MYSQL\_DATABSE=sbms

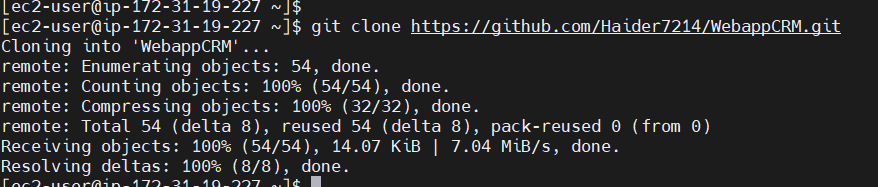
volumes:

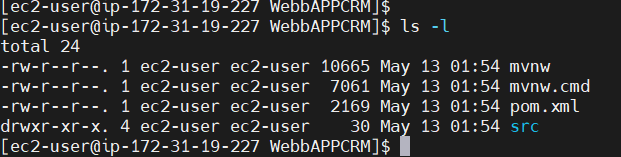
- /data/mysql

networks:

compose-network:

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





No dockerfile, docker-compose file nor target folder

[ec2-user@ip-172-31-19-227 WebbAPPCRM]$ mvn clean package

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

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

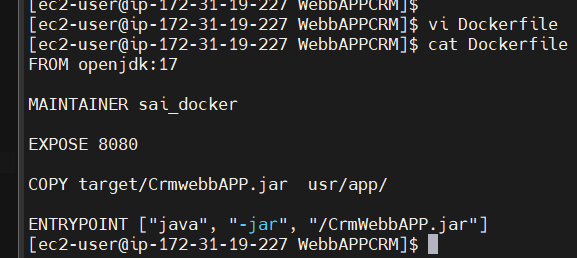
FROM openjdk:17

MAINTAINER sai\_docker

EXPOSE 8080

COPY target/CrmwebbAPP.jar usr/app/

ENTRYPOINT ["java", "-jar", "/CrmWebbAPP.jar"]



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

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

version: "3"

services:

application:

image: springboot-app

ports:

- "8080:8080"

networks:

- compose-network

depends\_on:

- mysqldb

volumes:

- /data/springboot-app

mysqldb:

image: mysql:8.3

networks:

- compose-network

ports:

- "3306:3306"

environment:

- MYSQL\_ROOT\_PASSWORD=root

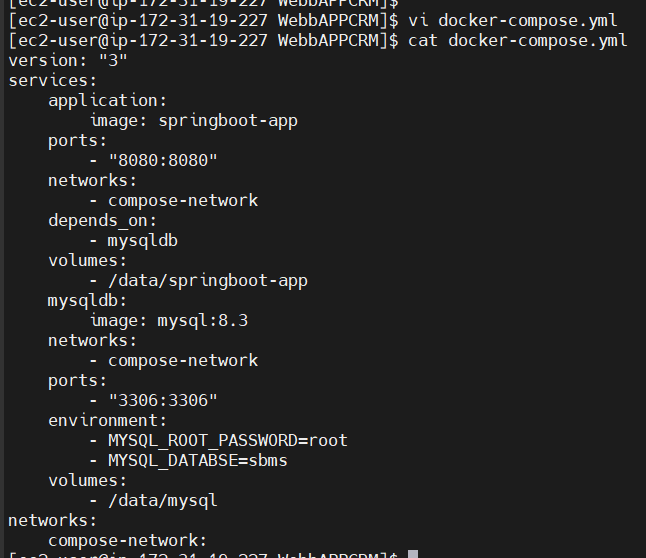
- MYSQL\_DATABSE=sbms

volumes:

- /data/mysql

networks:

compose-network:



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

total 32

-rw-r--r--. 1 ec2-user ec2-user 482 May 13 02:06 docker-compose.yml

-rw-r--r--. 1 ec2-user ec2-user 138 May 13 01:59 Dockerfile

-rw-r--r--. 1 ec2-user ec2-user 10665 May 13 01:54 mvnw

-rw-r--r--. 1 ec2-user ec2-user 7061 May 13 01:54 mvnw.cmd

-rw-r--r--. 1 ec2-user ec2-user 2169 May 13 01:54 pom.xml

drwxr-xr-x. 4 ec2-user ec2-user 30 May 13 01:54 src

drwxr-xr-x. 8 ec2-user ec2-user 140 May 13 01:56 target

[ec2-user@ip-172-31-19-227 WebbAPPCRM]$ docker network create compose-network

[ec2-user@ip-172-31-19-227 WebbAPPCRM]$ mvn clean package

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

FROM openjdk:17

MAINTAINER sai\_docker

EXPOSE 8080

COPY target/CrmWebbAPP.jar usr/app/

ENTRYPOINT ["java", "-jar", "/CrmWebbAPP.jar"]

Updated docker-compose

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

version: "3"

services:

application:

image: springboot-app

ports:

- "8080:8080"

networks:

- compose-network

depends\_on:

- mysqldb

volumes:

- /data/springboot-app

mysqldb:

image: mysql:8.3

networks:

- compose-network

ports:

- "3306:3306"

environment:

- MYSQL\_DATASOURCE\_URL: jdbc:mysql://mysqldb:3306/sbms

- MYSQL\_ROOT\_PASSWORD: root

- MYSQL\_DATABSE: sbms

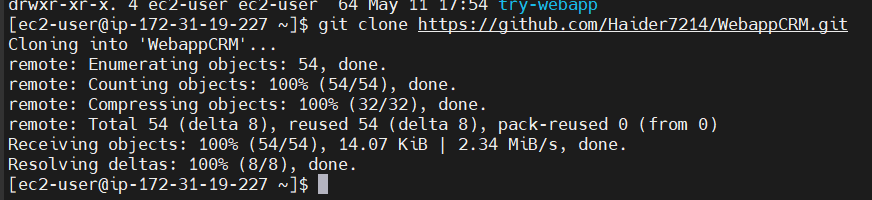
volumes:

- /data/mysql

networks:

compose-network:

Re-cloning due to some issue



Some problem in application will fix later

docker-compose up -d

docker-compose ps

docker-compose stop

docker-compose start