Pratik Chandrakant Borge

Three Tier Studentapp using Docker Compose

Step 1:- Create an EC2 instance
Ports:22, 80, 8080, 3306
Step 2:- Connect to EC2 instance
Step 3:- Install Docker latest version
Step 4:- Create database mysql
docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=1234 mysql
docker exec -it cont.id mysql -u root -p1234

```
mysql> create database studentapp;
Query OK, 1 row affected (0.02 sec)

mysql> use studentapp;
Database changed
mysql> CREATE TABLE if not exists students(student_id INT NOT NULL AUTO_INCREMENT,
    -> student_name VARCHAR(100) NOT NULL,
    ->     student_addr VARCHAR(100) NOT NULL,
    -> student_age VARCHAR(3) NOT NULL,
    -> student_qual VARCHAR(20) NOT NULL,
    -> student_percent VARCHAR(10) NOT NULL,
    -> student_year_passed VARCHAR(10) NOT NULL,
    -> PRIMARY KEY (student_id)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> desc students;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| student_id         | int          | NO   | PRI | NULL    | auto_increment |
| student_name       | varchar(100) | NO   |     | NULL    |                |
| student_addr       | varchar(100) | NO   |     | NULL    |                |
| student_age        | varchar(3)   | NO   |     | NULL    |                |
| student_qual       | varchar(20)  | NO   |     | NULL    |                |
| student_percent    | varchar(10)  | NO   |     | NULL    |                |
| student_year_passed | varchar(10) | NO   |     | NULL    |                |
+--------------------+--------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)

mysql> exit
Bye
```

Step 5:- Get IP of database
docker inspect cont.id | grep “IP”
Step 6:- Create dockerfiles of frontend and backend

Dockerfile of frontend:-

```
FROM centos:7
LABEL app="studentappFE"

USER root
RUN yum install httpd -y
COPY index.html /var/www/html/index.html
EXPOSE 80
CMD httpd -DFOREGROUND
```

Make sure to add index.html where the dockerfile of frontend is present

Edit index.html file
sample index.html=
**<h1 style="text-align: center;">Welcome to Student Application on AWS.</h1> **
**   <h2style="text-align:center;"><ahref="http://18.136.199.200:8080/student">Enter to Student Application</h2>   **

Replace it with your IP of backend server

Dockerfile of backend:-

```
FROM centos:7
LABEL app="studentapp"
USER root
WORKDIR /opt/
ADD https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.99/bin/apache-tomcat-8.5.99.tar.gz  .
RUN tar -xzvf apache-tomcat-8.5.99.tar.gz
WORKDIR /opt/apache-tomcat-8.5.99
ADD https://s3-us-west-2.amazonaws.com/studentapi-cit/student.war  webapps/student.war
ADD https://s3-us-west-2.amazonaws.com/studentapi-cit/mysql-connector.jar
lib/mysql-connector.jar
COPY context.xml conf/context.xml
RUN yum install java -y
EXPOSE 8080
CMD [ "bin/catalina.sh", "run" ]
```

Make sure you have context.xml file where you have dockerfile of backend because we have used COPY command in this dockerfile.
And paste the Resource in context.xml

```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
                maxTotal="100" maxIdle="30" maxWaitMillis="10000"
                username="USERNAME" password="PASSWORD"
                driverClassName="com.mysql.jdbc.Driver"
                url="jdbc:mysql://DB-ENDPOINT:3306/DATABASE"/>
```

Edit this resource according to the database created
Username, password, url needs to be edited.

Step 7:- Create docker compose file(YAML) at location where the directories of frontend and backend are present.

Check docker compose version present in instance to use it in compose file
docker-compose.yml

```
version : "2.24.7"

services:
  frontendimg :
    build : /root/Doc_Three_Tier/Frontend
    container_name: frontend
    ports:
    - "80:80"
    network_mode: bridge
  backendimg :
   build : /root/Doc_Three_Tier/Backend
   container_name: backend
   ports:
   - "8080:8080"
   network_mode: bridge
   depends_on:
   - "frontendimg"
```

Always give path of dockerfile in build section properly.
Give suitable container name.
Give ports for mapping.
Use network_mode to give the default network bridge so frontend,backend and database will be present in one network.
Use depends_on so that frontend and backend can form a connection.

Into your EC2 instance go to the path where your compose file is located.

Use command:-

docker compose up -d(to run compose in detatched

mode) It then builds images and then containers from those images.