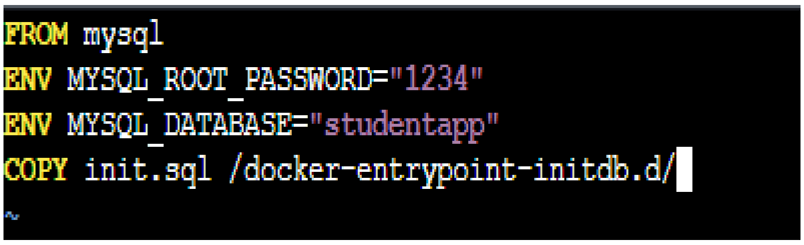
Hosting Tomcat Website in a Docker Container using Dockerfile

- First we need to create a container for mysql and then create a database.
- Create a directory mysql.
- Create a file called Dockerfile in the mysql directory.
- Add the following content in the Dockerfile of mysql.
  FROM mysql
  ENV MYSQL_ROOT_PASSWORD="1234"
  ENV MYSQL_DATABASE="studentapp"
  COPY init.sql /docker-entrypoint-initdb.d/



```
FROM mysql
ENV MYSQL_ROOT_PASSWORD="1234"
ENV MYSQL_DATABASE="studentapp"
COPY init.sql /docker-entrypoint-initdb.d/
~
```

Dockerfile Explaination :

FROM mysql: -> This line specifies the base image to use, which is the official MySQL Docker image. This image includes all the necessary software and configurations needed to run a MySQL server.

ENV -> These lines set environment variables within the Docker container:

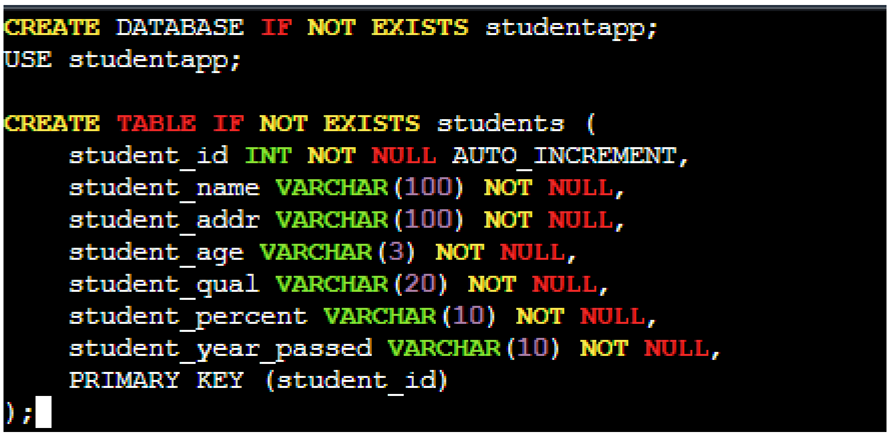
MYSQL_ROOT_PASSWORD is used to set the root password for the MySQL server to "1234".

MYSQL_DATABASE specifies the name of the default database to be created when the MySQL server starts.

COPY init.sql /docker-entrypoint-initdb.d/ -> This line copies an SQL script (init.sql) from the host environment to the docker-entrypoint-initdb.d/ directory in the Docker container.

- The init.sql file contains MySQL commands such as creating a database and tables, and inserting data.
- The docker-entrypoint-initdb.d/ directory is a special directory in the MySQL Docker image that automatically processes any scripts it contains when the container starts.
- This setup allows you to automate the initialization of the MySQL database with your specific schema and data when the container starts.

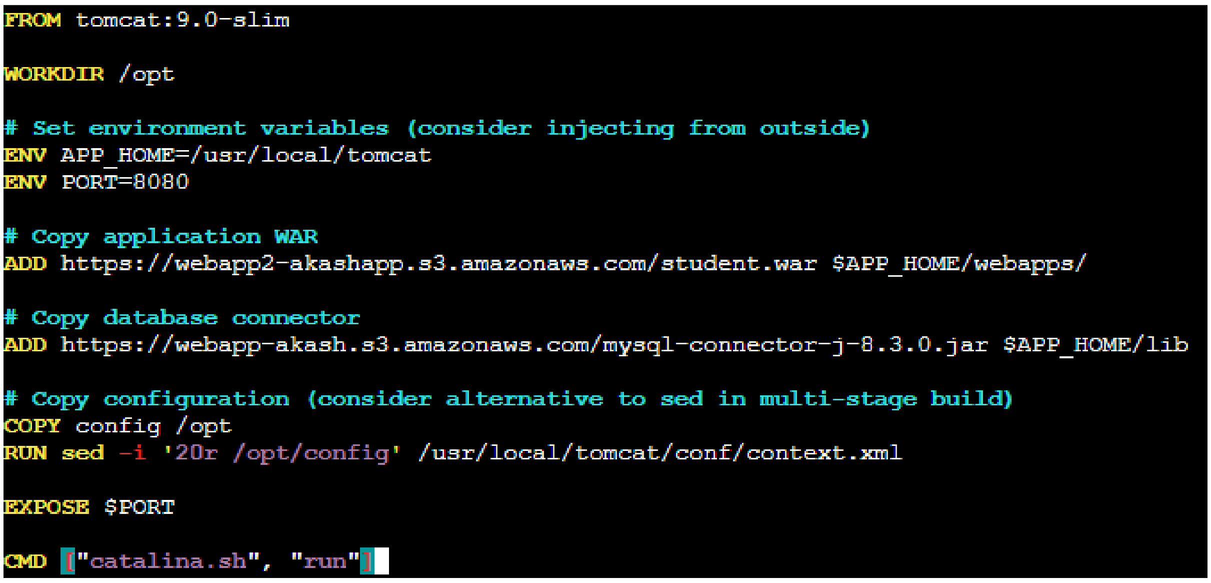
- Now we need to Create a init.sql file in the mysql directory.
- init.sql file



```
CREATE DATABASE IF NOT EXISTS studentapp;
USE studentapp;

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)
);
```

- Now come back to the /home/ec2-user.
- Create a Dockerfile here for our application.



```
FROM tomcat:9.0-slim

WORKDIR /opt

# Set environment variables (consider injecting from outside)
ENV APP_HOME=/usr/local/tomcat
ENV PORT=8080

# Copy application WAR
ADD https://webapp2-akashapp.s3.amazonaws.com/student.war $APP_HOME/webapps/

# Copy database connector
ADD https://webapp-akash.s3.amazonaws.com/mysql-connector-j-8.3.0.jar $APP_HOME/lib

# Copy configuration (consider alternative to sed in multi-stage build)
COPY config /opt
RUN sed -i '20r /opt/config' /usr/local/tomcat/conf/context.xml

EXPOSE $PORT

CMD ["catalina.sh", "run"]
```

Dockerfile Explaination:

- **Base Image: FROM tomcat:9.0-slim :**
- This line specifies the base image as Tomcat 9.0-slim, which is a lightweight version of the official Tomcat 9.0 image.

- **Working Directory: WORKDIR /opt:**
- This line sets the working directory to `/opt` within the Docker container. All subsequent commands that involve file paths will operate relative to this directory.

- **Environment Variables: ENV APP_HOME=/usr/local/tomcat and ENV PORT=8080:**
- APP_HOME is set to /usr/local/tomcat, specifying the Tomcat home directory where the server is installed.
- PORT is set to 8080, indicating the default port for the Tomcat server.

- **Copying Application WAR File: ADD https://webapp2-akashapp.s3.amazonaws.com/student.war $APP_HOME/webapps/:**
- This line downloads the `student.war` file from a specified URL and copies it to the Tomcat webapps directory (`$APP_HOME/webapps/`).
- The WAR file contains the application that Tomcat will deploy and run.

- **Copying Database Connector: ADD https://webapp-akash.s3.amazonaws.com/mysql-connector-j-8.3.0.jar $APP_HOME/lib**:
- This line downloads the MySQL JDBC driver (mysql-connector-j-8.3.0.jar) from a specified URL and places it in the Tomcat lib directory (`$APP_HOME/lib`).
- This allows the application to connect to a MySQL database.

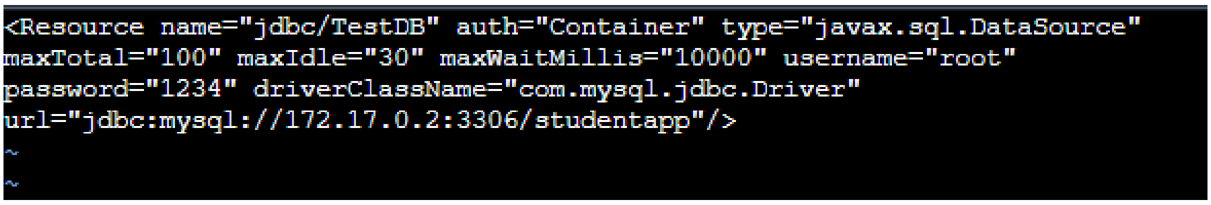
- **Copying Configuration: COPY config /opt:**
- This line copies a config file from the host to the /opt directory in the Docker container.
- This configuration file may contain custom settings or properties needed by the application.

- **Injecting Configuration: RUN sed -i '20r /opt/config' /usr/local/tomcat/conf/context.xml:**
- This line uses the `sed` command to modify the `context.xml` file located in the Tomcat configuration directory (`/usr/local/tomcat/conf/`).
- The -i flag allows sed to modify the file in place.
- The 20r /opt/config option tells sed to read the config file and insert its contents into context.xml at line 20.
- This allows you to inject configuration settings into the context.xml file.

- **Exposing Port: EXPOSE $PORT:**
- This line exposes the specified port (`$PORT`, which is set to `8080`) to allow access to the application running in the Tomcat container.

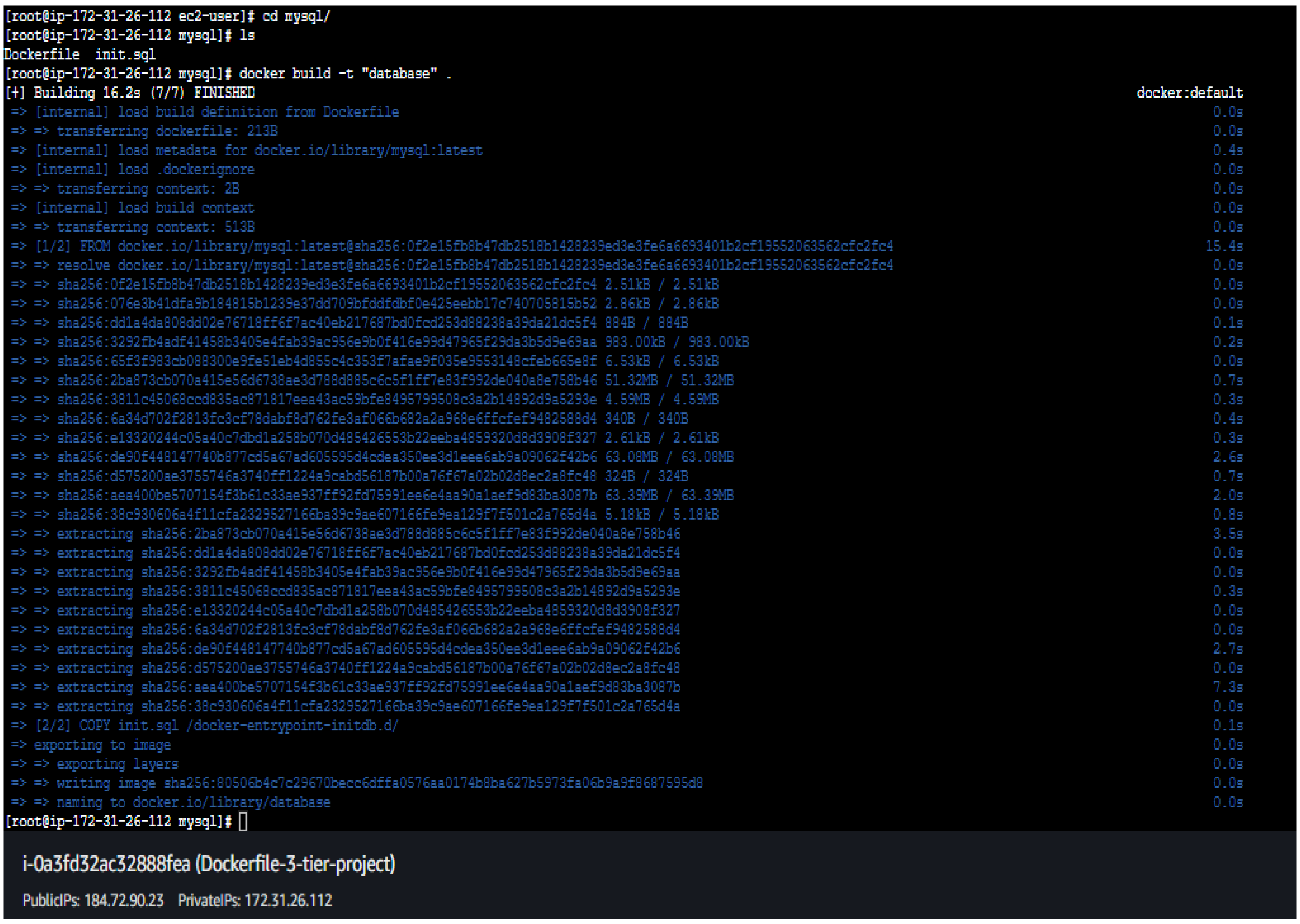
- **Command to Start Tomcat**: `CMD ["catalina.sh", "run"]:**
- This line specifies the command to start the Tomcat server when the container runs, using the `catalina.sh` script with the `run` argument.

- Now we need to create a file named config in /home/ec2-user.
- Add the configuration in config file



```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
password="1234" driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://172.17.0.2:3306/studentapp"/>
~
~
```

- Here I have given the IP address of the mysql container.
- When mysql container is created for the first time it has the same IP as given.
- And if there is already one mysql container present and again you create another mysql container the IP increases by 1.
- Now save the file and exit.
- Now our both Dockerfiles and configuration is ready.
- Change directory to mysql.
- Hit command '**docker build -t "database" .**'

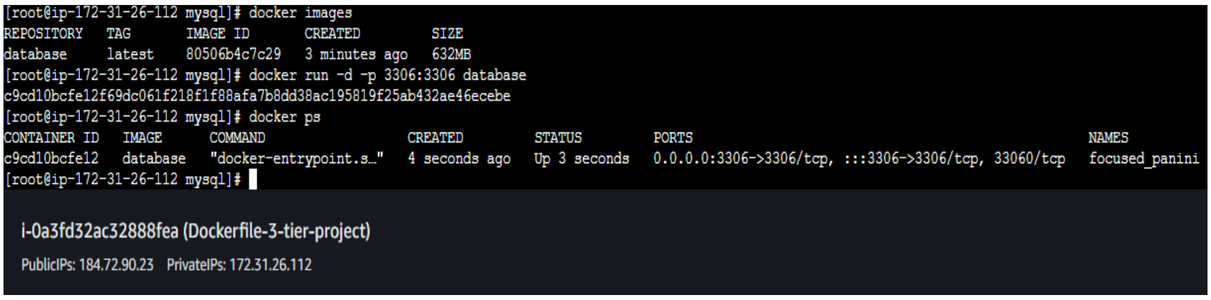


```
[root@ip-172-31-26-112 ec2-user]# cd mysql/
[root@ip-172-31-26-112 mysql]# ls
Dockerfile  init.sql
[root@ip-172-31-26-112 mysql]# docker build -t "database" .
[+] Building 16.2s (7/7) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 213B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/mysql:latest                                                               0.4s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [internal] load build context                                                                                             0.0s
 => => transferring context: 513B                                                                                             0.0s
 => [1/2] FROM docker.io/library/mysql:latest@sha256:0f2e15fb8b47db2518b1428239ed3e3fe6a6693401b2cf19552063562cfc2fc4        15.4s
 => => resolve docker.io/library/mysql:latest@sha256:0f2e15fb8b47db2518b1428239ed3e3fe6a6693401b2cf19552063562cfc2fc4         0.0s
 => => sha256:0f2e15fb8b47db2518b1428239ed3e3fe6a6693401b2cf19552063562cfc2fc4 2.51kB / 2.51kB                                0.0s
 => => sha256:076e3b41dfa9b184815b1239e37dd709bfddfdbf0e425eebb17c740705815b52 2.86kB / 2.86kB                                0.0s
 => => sha256:dd1a4da808dd02e76718ff6f7ac40eb217687bd0fcd253d88238a39da21dc5f4 884B / 884B                                    0.1s
 => => sha256:3292fb4adf41458b3405e4fab39ac956e9b0f416e99d47965f29da3b5d9e69aa 983.00kB / 983.00kB                            0.2s
 => => sha256:65f3f983cb088300e9fe51eb4d855c4c353f7afae9f035e9553148cfeb665e8f 6.53kB / 6.53kB                                0.0s
 => => sha256:2ba873cb070a415e56d6738ae3d788d885c6c5f1ff7e83f992de040a8e758b46 51.32MB / 51.32MB                              0.7s
 => => sha256:3811c45068ccd835ac871817eea43ac59bfe8495799508c3a2b14892d9a5293e 4.59MB / 4.59MB                                0.3s
 => => sha256:6a34d702f2813fc3cf78dabf8d762fe3af066b682a2a968e6ffcfef9482588d4 340B / 340B                                    0.4s
 => => sha256:e13320244c05a40c7dbd1a258b070d485426553b22eeba4859320d8d3908f327 2.61kB / 2.61kB                                0.3s
 => => sha256:de90f448147740b877cd5a67ad605595d4cdea350ee3d1eee6ab9a09062f42b6 63.08MB / 63.08MB                              2.6s
 => => sha256:d575200ae3755746a3740ff1224a9cabd56187b00a76f67a02b02d8ec2a8fc48 324B / 324B                                    0.7s
 => => sha256:aea400be5707154f3b61c33ae937ff92fd75991ee6e4aa90a1aef9d83ba3087b 63.39MB / 63.39MB                              2.0s
 => => sha256:38c930606a4f11cfa2329527166ba39c9ae607166fe9ea129f7f501c2a765d4a 5.18kB / 5.18kB                                0.8s
 => => extracting sha256:2ba873cb070a415e56d6738ae3d788d885c6c5f1ff7e83f992de040a8e758b46                                     3.5s
 => => extracting sha256:dd1a4da808dd02e76718ff6f7ac40eb217687bd0fcd253d88238a39da21dc5f4                                     0.0s
 => => extracting sha256:3292fb4adf41458b3405e4fab39ac956e9b0f416e99d47965f29da3b5d9e69aa                                     0.0s
 => => extracting sha256:3811c45068ccd835ac871817eea43ac59bfe8495799508c3a2b14892d9a5293e                                     0.3s
 => => extracting sha256:e13320244c05a40c7dbd1a258b070d485426553b22eeba4859320d8d3908f327                                     0.0s
 => => extracting sha256:6a34d702f2813fc3cf78dabf8d762fe3af066b682a2a968e6ffcfef9482588d4                                     0.0s
 => => extracting sha256:de90f448147740b877cd5a67ad605595d4cdea350ee3d1eee6ab9a09062f42b6                                     2.7s
 => => extracting sha256:d575200ae3755746a3740ff1224a9cabd56187b00a76f67a02b02d8ec2a8fc48                                     0.0s
 => => extracting sha256:aea400be5707154f3b61c33ae937ff92fd75991ee6e4aa90a1aef9d83ba3087b                                     7.3s
 => => extracting sha256:38c930606a4f11cfa2329527166ba39c9ae607166fe9ea129f7f501c2a765d4a                                     0.0s
 => [2/2] COPY init.sql /docker-entrypoint-initdb.d/                                                                          0.1s
 => exporting to image                                                                                                        0.0s
 => => exporting layers                                                                                                       0.0s
 => => writing image sha256:80506b4c7c29670becc6dffa0576aa0174b8ba627b5973fa06b9a9f8687595d8                                  0.0s
 => => naming to docker.io/library/database                                                                                   0.0s
[root@ip-172-31-26-112 mysql]# 
```

i-0a3fd32ac32888fea (Dockerfile-3-tier-project)

PublicIPs: 184.72.90.23   PrivateIPs: 172.31.26.112

- Now the image is created from Dockerfile.
- Hit command “docker images” to check the image.
- Hit command “docker run -d -p 3306:3306 database” to run the image in background to create a container.

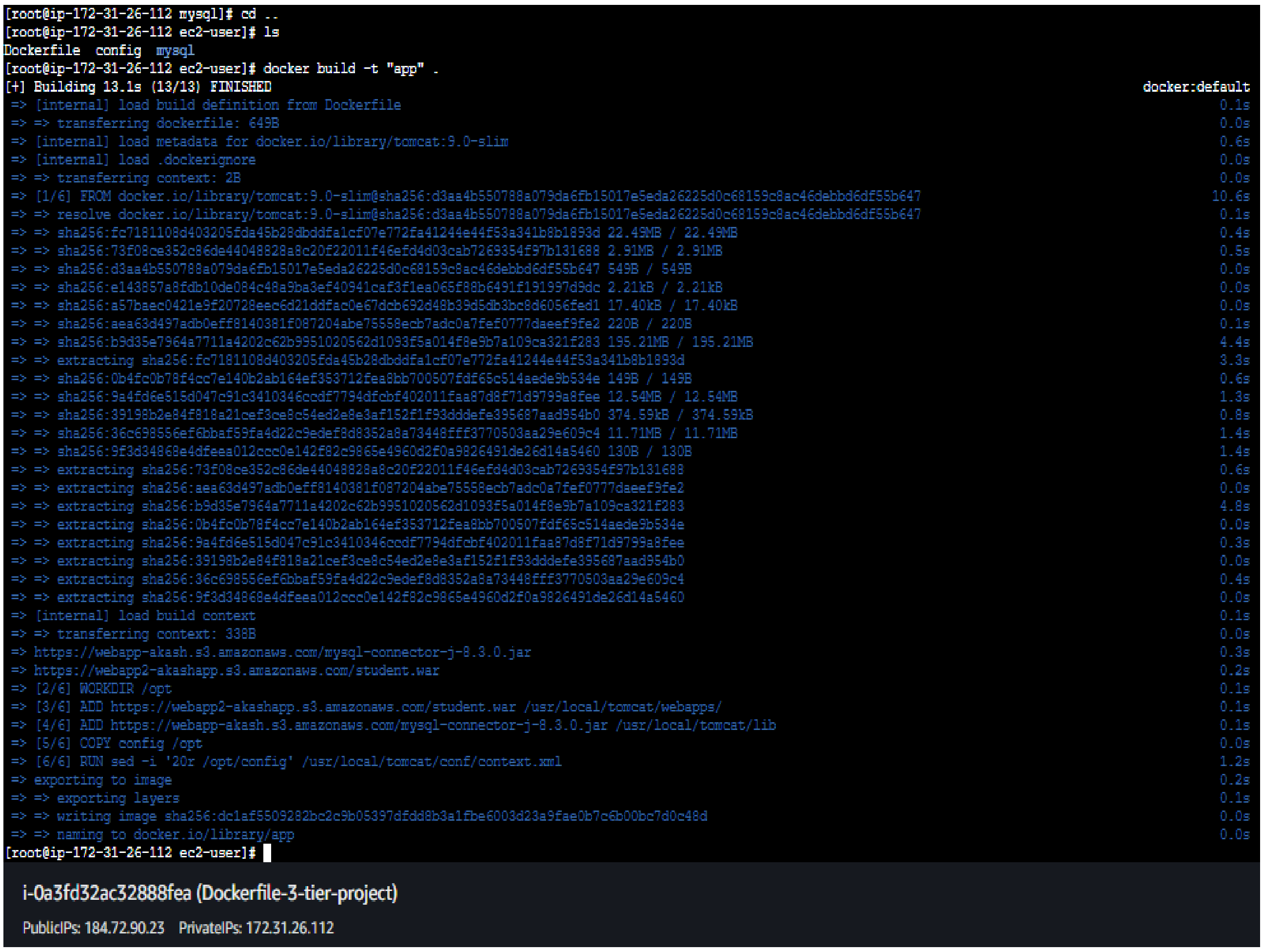


```
[root@ip-172-31-26-112 mysql]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
database     latest    80506b4c7c29   3 minutes ago   632MB
[root@ip-172-31-26-112 mysql]# docker run -d -p 3306:3306 database
c9cd10bcfe12f69dc061f218f1f88afa7b8dd38ac195819f25ab432ae46ecebe
[root@ip-172-31-26-112 mysql]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED         STATUS         PORTS                                                  NAMES
c9cd10bcfe12   database   "docker-entrypoint.s…"   4 seconds ago   Up 3 seconds   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   focused_panini
[root@ip-172-31-26-112 mysql]# 
```

i-0a3fd32ac32888fea (Dockerfile-3-tier-project)

PublicIPs: 184.72.90.23   PrivateIPs: 172.31.26.112

- Now go back to /home/ec2-user.
- Here also we have to run the Dockerfile for our application.
- Hit command ‘docker build -t “app” .’



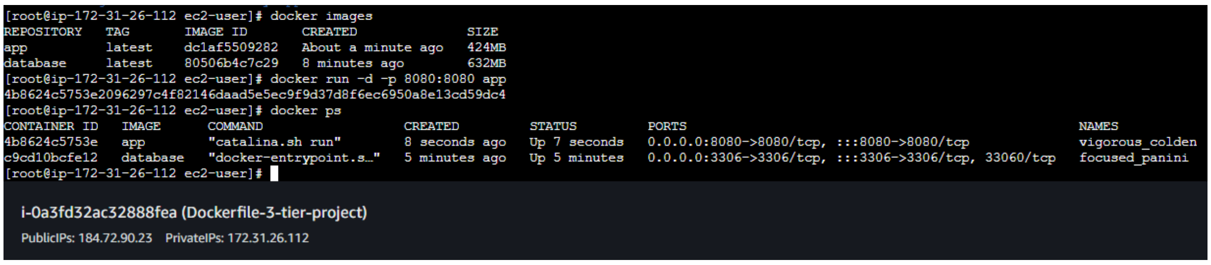
```
[root@ip-172-31-26-112 mysql]# cd ..
[root@ip-172-31-26-112 ec2-user]# ls
Dockerfile  config  mysql
[root@ip-172-31-26-112 ec2-user]# docker build -t "app" .
[+] Building 13.1s (13/13) FINISHED                                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                                           0.1s
 => => transferring dockerfile: 649B                                                                                           0.0s
 => [internal] load metadata for docker.io/library/tomcat:9.0-slim                                                             0.6s
 => [internal] load .dockerignore                                                                                              0.0s
 => => transferring context: 2B                                                                                                0.0s
 => [1/6] FROM docker.io/library/tomcat:9.0-slim@sha256:d3aa4b550788a079da6fb15017e5eda26225d0c68159c8ac46debbd6df55b647       10.6s
 => => resolve docker.io/library/tomcat:9.0-slim@sha256:d3aa4b550788a079da6fb15017e5eda26225d0c68159c8ac46debbd6df55b647        0.1s
 => => sha256:fc7181108d403205fda45b28dbddfa1cf07e772fa41244e44f53a341b8b1893d 22.49MB / 22.49MB                               0.4s
 => => sha256:73f08ce352c86de44048828a8c20f22011f46efd4d03cab7269354f97b131688 2.91MB / 2.91MB                                 0.5s
 => => sha256:d3aa4b550788a079da6fb15017e5eda26225d0c68159c8ac46debbd6df55b647 549B / 549B                                     0.0s
 => => sha256:e143857a8fdb10de084c48a9ba3ef40941caf3f1ea065f88b6491f191997d9dc 2.21kB / 2.21kB                                 0.0s
 => => sha256:a57baec0421e9f20728eec6d21ddfac0e67dcb692d48b39d5db3bc8d6056fed1 17.40kB / 17.40kB                               0.0s
 => => sha256:aea63d497adb0eff8140381f087204abe75558ecb7adc0a7fef0777daeef9fe2 220B / 220B                                     0.1s
 => => sha256:b9d35e7964a7711a4202c62b9951020562d1093f5a014f8e9b7a109ca321f283 195.21MB / 195.21MB                             4.4s
 => => extracting sha256:fc7181108d403205fda45b28dbddfa1cf07e772fa41244e44f53a341b8b1893d                                      3.3s
 => => sha256:0b4fc0b78f4cc7e140b2ab164ef353712fea8bb700507fdf65c514aede9b534e 149B / 149B                                     0.6s
 => => sha256:9a4fd6e515d047c91c3410346ccdf7794dfcbf402011faa87d8f71d9799a8fee 12.54MB / 12.54MB                               1.3s
 => => sha256:39198b2e84f818a21cef3ce8c54ed2e8e3af152f1f93dddefe395687aad954b0 374.59kB / 374.59kB                             0.8s
 => => sha256:36c698556ef6bbaf59fa4d22c9edef8d8352a8a73448fff3770503aa29e609c4 11.71MB / 11.71MB                               1.4s
 => => sha256:9f3d34868e4dfeea012ccc0e142f82c9865e4960d2f0a9826491de26d14a5460 130B / 130B                                     1.4s
 => => extracting sha256:73f08ce352c86de44048828a8c20f22011f46efd4d03cab7269354f97b131688                                      0.6s
 => => extracting sha256:aea63d497adb0eff8140381f087204abe75558ecb7adc0a7fef0777daeef9fe2                                      0.0s
 => => extracting sha256:b9d35e7964a7711a4202c62b9951020562d1093f5a014f8e9b7a109ca321f283                                      4.8s
 => => extracting sha256:0b4fc0b78f4cc7e140b2ab164ef353712fea8bb700507fdf65c514aede9b534e                                      0.0s
 => => extracting sha256:9a4fd6e515d047c91c3410346ccdf7794dfcbf402011faa87d8f71d9799a8fee                                      0.3s
 => => extracting sha256:39198b2e84f818a21cef3ce8c54ed2e8e3af152f1f93dddefe395687aad954b0                                      0.0s
 => => extracting sha256:36c698556ef6bbaf59fa4d22c9edef8d8352a8a73448fff3770503aa29e609c4                                      0.4s
 => => extracting sha256:9f3d34868e4dfeea012ccc0e142f82c9865e4960d2f0a9826491de26d14a5460                                      0.0s
 => [internal] load build context                                                                                              0.1s
 => => transferring context: 338B                                                                                              0.0s
 => https://webapp-akash.s3.amazonaws.com/mysql-connector-j-8.3.0.jar                                                          0.3s
 => https://webapp2-akashapp.s3.amazonaws.com/student.war                                                                      0.2s
 => [2/6] WORKDIR /opt                                                                                                         0.1s
 => [3/6] ADD https://webapp2-akashapp.s3.amazonaws.com/student.war /usr/local/tomcat/webapps/                                 0.1s
 => [4/6] ADD https://webapp-akash.s3.amazonaws.com/mysql-connector-j-8.3.0.jar /usr/local/tomcat/lib                          0.1s
 => [5/6] COPY config /opt                                                                                                     0.0s
 => [6/6] RUN sed -i '20r /opt/config' /usr/local/tomcat/conf/context.xml                                                      1.2s
 => exporting to image                                                                                                         0.2s
 => => exporting layers                                                                                                        0.1s
 => => writing image sha256:dc1af5509282bc2c9b05397dfdd8b3a1fbe6003d23a9fae0b7c6b00bc7d0c48d                                   0.0s
 => => naming to docker.io/library/app                                                                                         0.0s
[root@ip-172-31-26-112 ec2-user]# 
```

i-0a3fd32ac32888fea (Dockerfile-3-tier-project)

PublicIPs: 184.72.90.23   PrivateIPs: 172.31.26.112

- Now the image is created.
- To check hit command “docker images”.
- Now we need to run the image so that the container will be created.
- Hit command “docker run -d -p 8080:8080 app”.

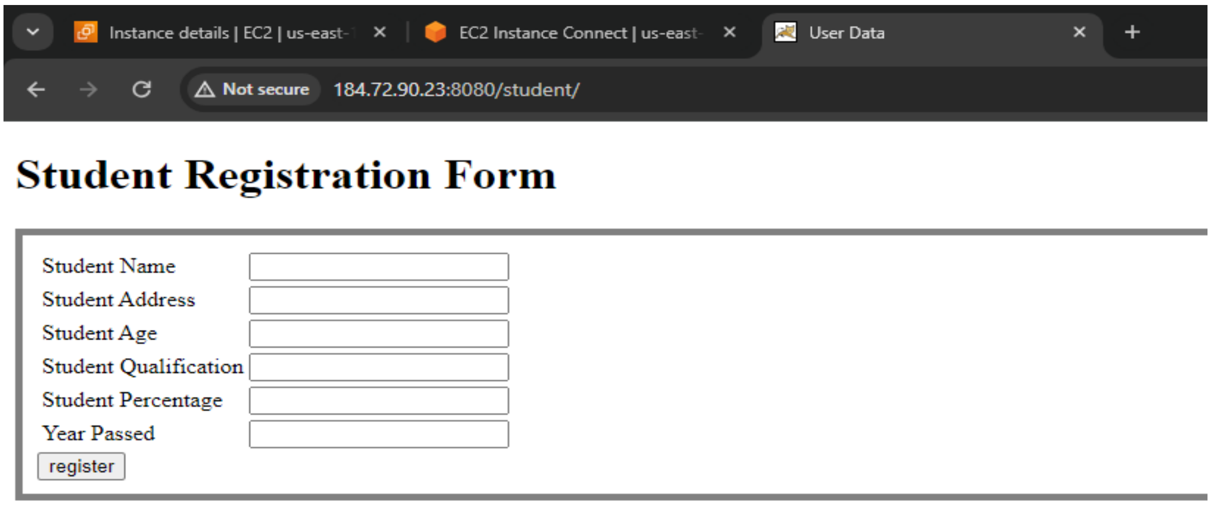
- To check if the container is running hit command “docker ps”.



```
[root@ip-172-31-26-112 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
app          latest    dc1af5509282   About a minute ago   424MB
database     latest    80506b4c7c29   8 minutes ago        632MB
[root@ip-172-31-26-112 ec2-user]# docker run -d -p 8080:8080 app
4b8624c5753e2096297c4f82146daad5e5ec9f9d37d8f6ec6950a8e13cd59dc4
[root@ip-172-31-26-112 ec2-user]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS         PORTS                                                  NAMES
4b8624c5753e   app        "catalina.sh run"        8 seconds ago    Up 7 seconds   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp              vigorous_colden
c9cd10bcfe12   database   "docker-entrypoint.s…"   5 minutes ago    Up 5 minutes   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   focused_panini
[root@ip-172-31-26-112 ec2-user]#
```

i-0a3fd32ac32888fea (Dockerfile-3-tier-project)
PublicIPs: 184.72.90.23   PrivateIPs: 172.31.26.112

- Now hit the IP address of the instance with port 8080/student.



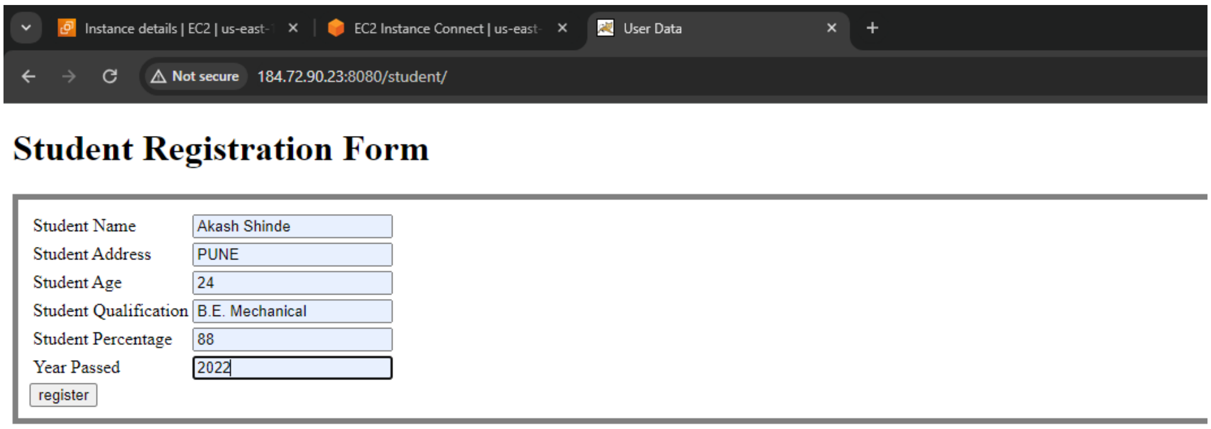
Instance details | EC2 | us-east-
EC2 Instance Connect | us-east-
User Data
Not secure
184.72.90.23:8080/student/

Student Registration Form

Student Name	
Student Address	
Student Age	
Student Qualification	
Student Percentage	
Year Passed	

register

- Page is loading successfully.
- Now fill the form and click register to check if our data is going to database.



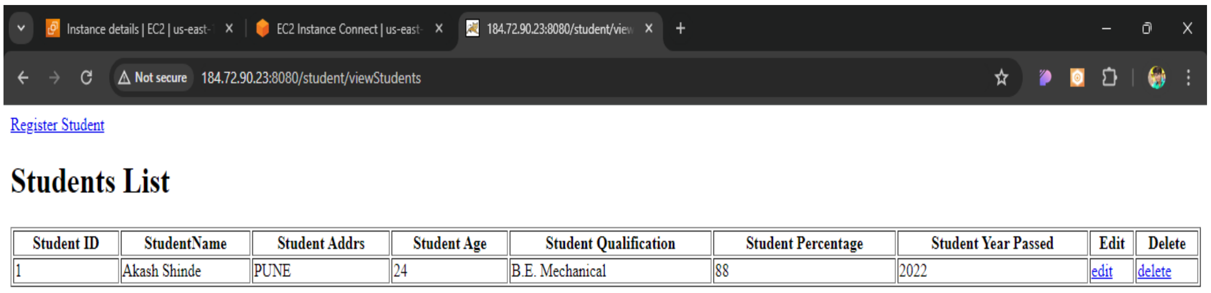
Instance details | EC2 | us-east-
EC2 Instance Connect | us-east-
User Data
Not secure
184.72.90.23:8080/student/

Student Registration Form

Student Name	Akash Shinde
Student Address	PUNE
Student Age	24
Student Qualification	B.E. Mechanical
Student Percentage	88
Year Passed	2022

register

- Our data is going to the database.



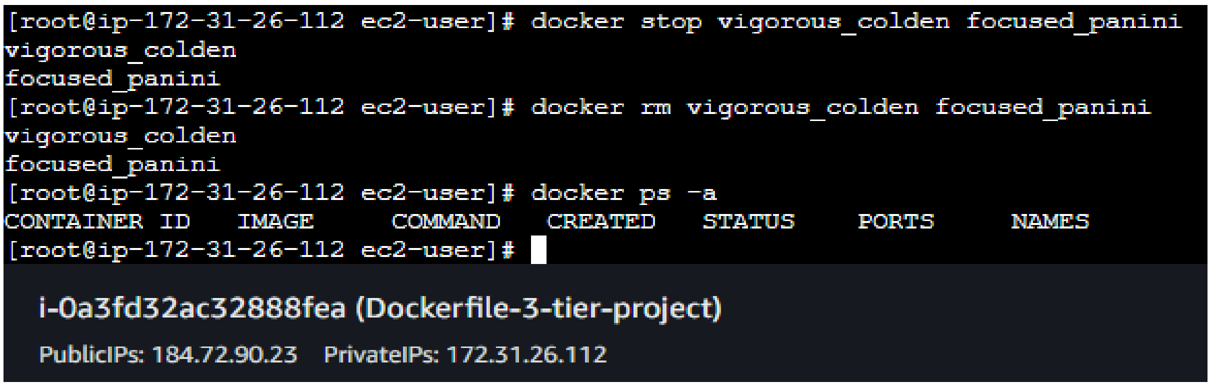
Instance details | EC2 | us-east-
EC2 Instance Connect | us-east-
184.72.90.23:8080/student/view
Not secure
184.72.90.23:8080/student/viewStudents

Register Student

Students List

Student ID	StudentName	Student Addrs	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
1	Akash Shinde	PUNE	24	B.E. Mechanical	88	2022	edit	delete

- Here we are building the Docker images from the Dockerfile separately by typing commands.
- We can do the same with script.
- We can write all the commands in a script and execute the script.
- To do the same follow the steps.
- First we need to stop the containers and remove them.

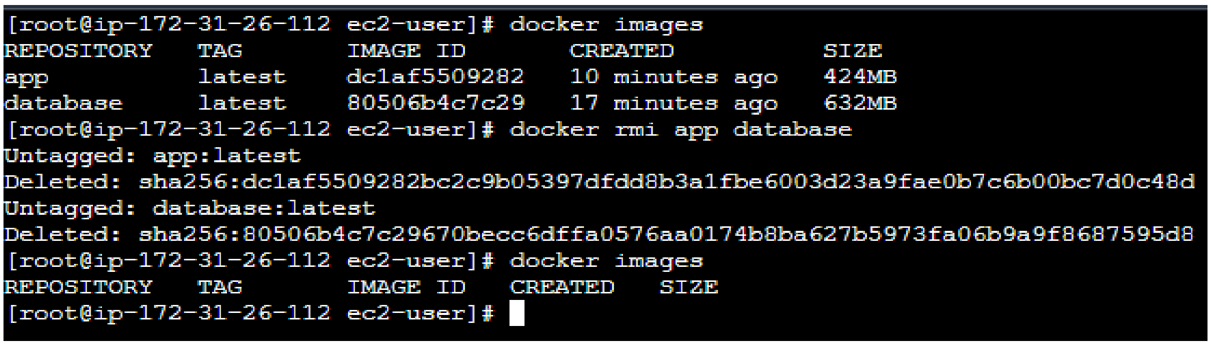


```
[root@ip-172-31-26-112 ec2-user]# docker stop vigorous_colden focused_panini
vigorous_colden
focused_panini
[root@ip-172-31-26-112 ec2-user]# docker rm vigorous_colden focused_panini
vigorous_colden
focused_panini
[root@ip-172-31-26-112 ec2-user]# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-26-112 ec2-user]#
```

i-0a3fd32ac32888fea (Dockerfile-3-tier-project)

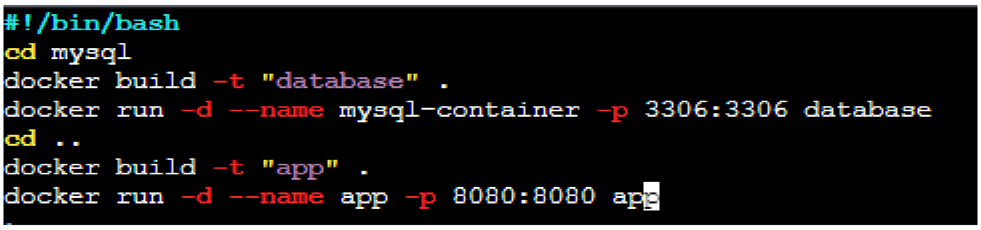
PublicIPs: 184.72.90.23   PrivateIPs: 172.31.26.112

- We also need to remove the images.



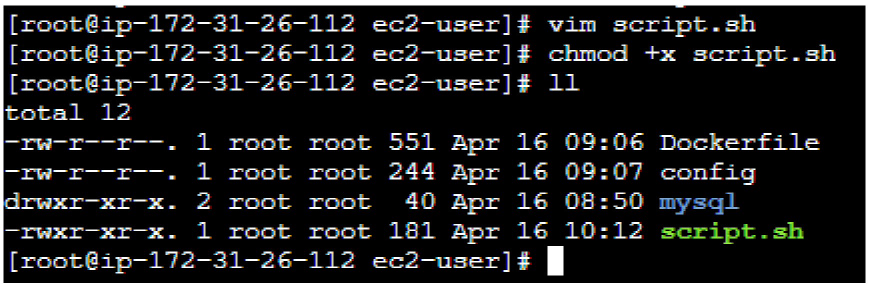
```
[root@ip-172-31-26-112 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
app           latest    dc1af5509282   10 minutes ago   424MB
database      latest    80506b4c7c29   17 minutes ago   632MB
[root@ip-172-31-26-112 ec2-user]# docker rmi app database
Untagged: app:latest
Deleted: sha256:dc1af5509282bc2c9b05397dfdd8b3a1fbe6003d23a9fae0b7c6b00bc7d0c48d
Untagged: database:latest
Deleted: sha256:80506b4c7c29670becc6dffa0576aa0174b8ba627b5973fa06b9a9f8687595d8
[root@ip-172-31-26-112 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[root@ip-172-31-26-112 ec2-user]#
```

- Now create a file named script.sh in /home/ec2-user.
- Add the commands in the script.sh file.



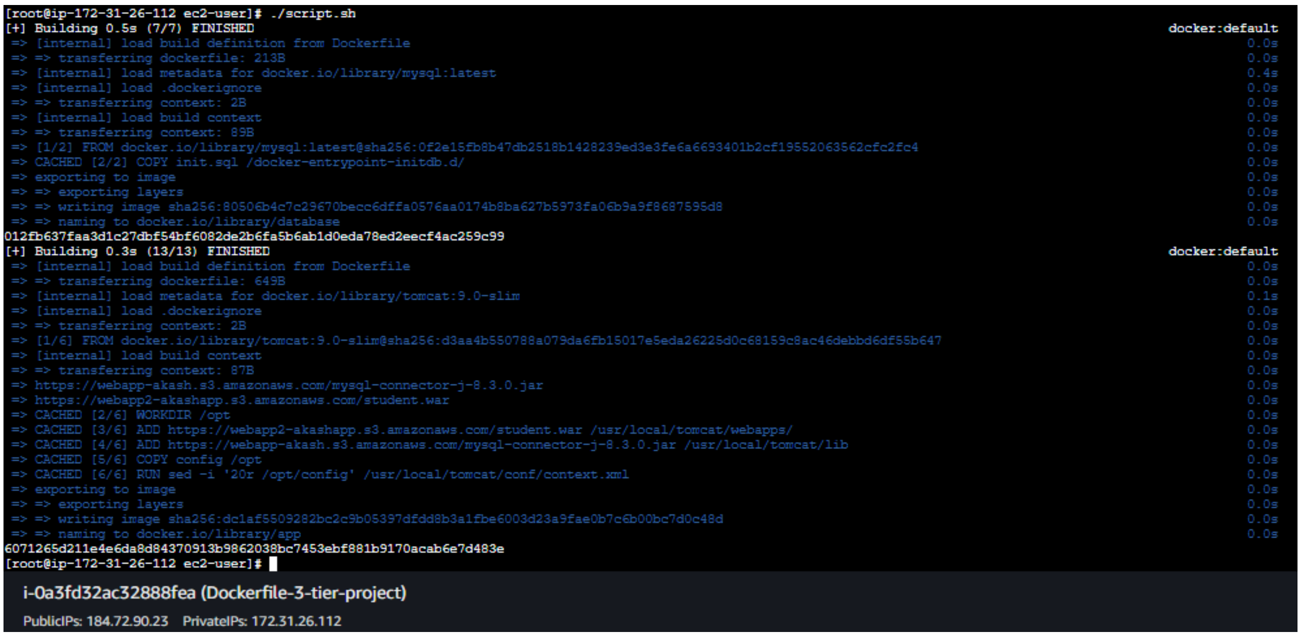
```
#!/bin/bash
cd mysql
docker build -t "database" .
docker run -d --name mysql-container -p 3306:3306 database
cd ..
docker build -t "app" .
docker run -d --name app -p 8080:8080 app
```

- Now we need to give the execute permissions to the script.sh file.
- Hit command “chmod +x script.sh” to give execute permissions.



```
[root@ip-172-31-26-112 ec2-user]# vim script.sh
[root@ip-172-31-26-112 ec2-user]# chmod +x script.sh
[root@ip-172-31-26-112 ec2-user]# ll
total 12
-rw-r--r--. 1 root root 551 Apr 16 09:06 Dockerfile
-rw-r--r--. 1 root root 244 Apr 16 09:07 config
drwxr-xr-x. 2 root root  40 Apr 16 08:50 mysql
-rwxr-xr-x. 1 root root 181 Apr 16 10:12 script.sh
[root@ip-172-31-26-112 ec2-user]#
```

- Now run the script.
- Hit command "./script.sh" to run the script.

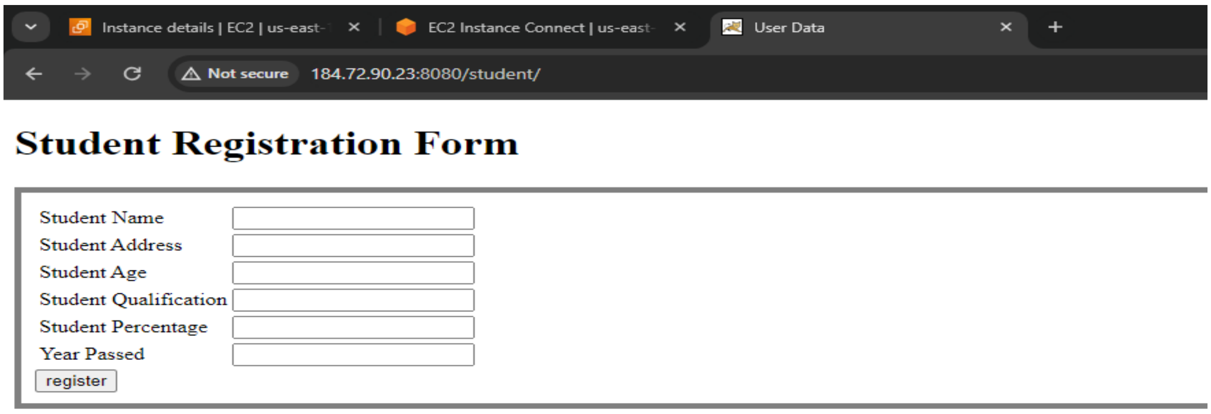


```
[root@ip-172-31-26-112 ec2-user]# ./script.sh
[+] Building 0.5s (7/7) FINISHED                                                                                                                     docker:default
 => [internal] load build definition from Dockerfile                                                                                                          0.0s
 => => transferring dockerfile: 213B                                                                                                                          0.0s
 => [internal] load metadata for docker.io/library/mysql:latest                                                                                               0.4s
 => [internal] load .dockerignore                                                                                                                             0.0s
 => => transferring context: 2B                                                                                                                               0.0s
 => [internal] load build context                                                                                                                             0.0s
 => => transferring context: 89B                                                                                                                              0.0s
 => [1/2] FROM docker.io/library/mysql:latest@sha256:0f2e15fb8b47db2518b1428239ed3e3fe6a6693401b2cf19552063562cfc2fc4                                        0.0s
 => CACHED [2/2] COPY init.sql /docker-entrypoint-initdb.d/                                                                                                   0.0s
 => exporting to image                                                                                                                                        0.0s
 => => exporting layers                                                                                                                                       0.0s
 => => writing image sha256:80506b4c7c29670becc6dffa0576aa0174b8ba627b5973fa06b9a9f8687595d8                                                                  0.0s
 => => naming to docker.io/library/database                                                                                                                   0.0s
012fb637faa3d1c27dbf54bf6082de2b6fa5b6ab1d0eda78ed2eecf4ac259c99
[+] Building 0.3s (13/13) FINISHED                                                                                                                   docker:default
 => [internal] load build definition from Dockerfile                                                                                                          0.0s
 => => transferring dockerfile: 649B                                                                                                                          0.0s
 => [internal] load metadata for docker.io/library/tomcat:9.0-slim                                                                                            0.1s
 => [internal] load .dockerignore                                                                                                                             0.0s
 => => transferring context: 2B                                                                                                                               0.0s
 => [1/6] FROM docker.io/library/tomcat:9.0-slim@sha256:d3aa4b550788a079da6fb15017e5eda26225d0c68159c8ac46debbd6df55b647                                      0.0s
 => [internal] load build context                                                                                                                             0.0s
 => => transferring context: 87B                                                                                                                              0.0s
 => https://webapp-akash.s3.amazonaws.com/mysql-connector-j-8.3.0.jar                                                                                         0.0s
 => https://webapp2-akashapp.s3.amazonaws.com/student.war                                                                                                     0.0s
 => CACHED [2/6] WORKDIR /opt                                                                                                                                 0.0s
 => CACHED [3/6] ADD https://webapp2-akashapp.s3.amazonaws.com/student.war /usr/local/tomcat/webapps/                                                         0.0s
 => CACHED [4/6] ADD https://webapp-akash.s3.amazonaws.com/mysql-connector-j-8.3.0.jar /usr/local/tomcat/lib                                                  0.0s
 => CACHED [5/6] COPY config /opt                                                                                                                             0.0s
 => CACHED [6/6] RUN sed -i '20r /opt/config' /usr/local/tomcat/conf/context.xml                                                                              0.0s
 => exporting to image                                                                                                                                        0.0s
 => => exporting layers                                                                                                                                       0.0s
 => => writing image sha256:dc1af5509282bc2c9b05397dfdd8b3a1fbe6003d23a9fae0b7c6b00bc7d0c48d                                                                  0.0s
 => => naming to docker.io/library/app                                                                                                                        0.0s
6071265d211e4e6da8d84370913b9862038bc7453ebf881b9170acab6e7d483e
[root@ip-172-31-26-112 ec2-user]#
```

i-0a3fd32ac32888fea (Dockerfile-3-tier-project)

PublicIPs: 184.72.90.23   PrivateIPs: 172.31.26.112

- Now hit the IP address of the instance with port 8080/student.



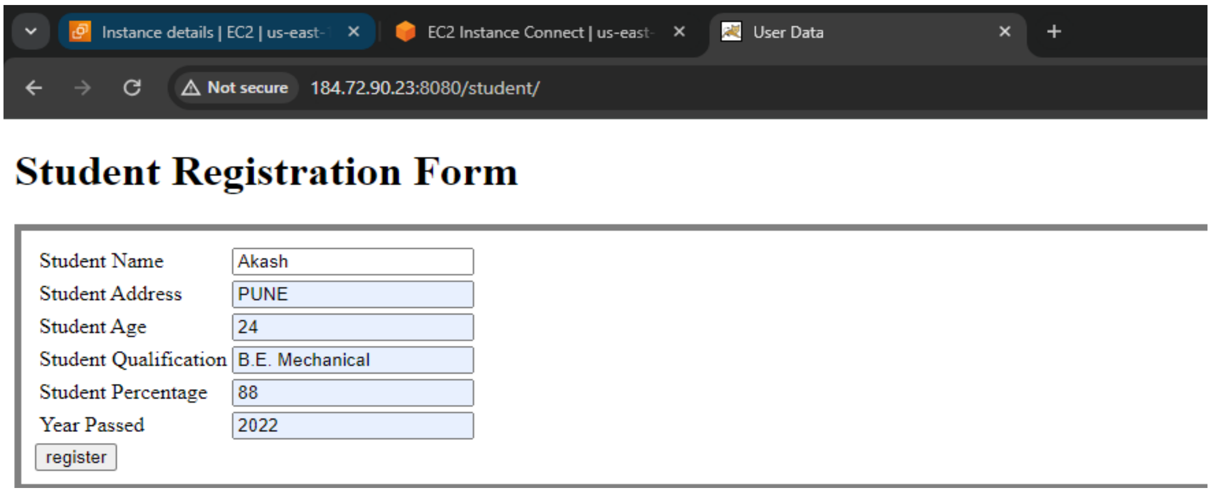
Instance details | EC2 | us-east-
EC2 Instance Connect | us-east-
User Data
Not secure
184.72.90.23:8080/student/

Student Registration Form

Student Name	
Student Address	
Student Age	
Student Qualification	
Student Percentage	
Year Passed	

register

- Page is visible now fill the form and click register to check if the data is being saved in the database.



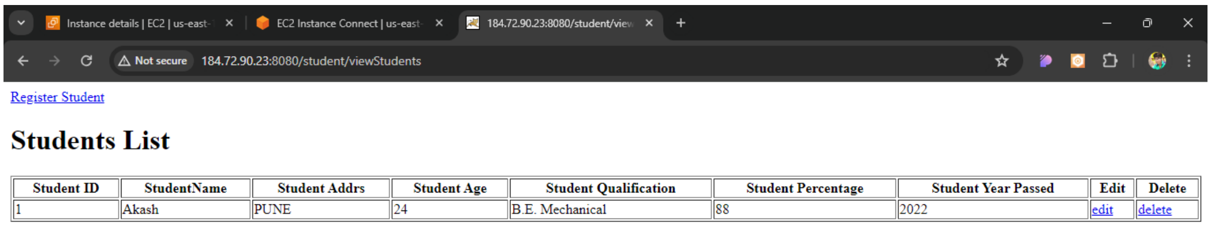
Instance details | EC2 | us-east-
EC2 Instance Connect | us-east-
User Data
Not secure
184.72.90.23:8080/student/

Student Registration Form

Student Name	Akash
Student Address	PUNE
Student Age	24
Student Qualification	B.E. Mechanical
Student Percentage	88
Year Passed	2022

register

- Data is being saved successfully in the database.


Instance details | EC2 | us-east-
EC2 Instance Connect | us-east-
184.72.90.23:8080/student/view
Not secure
184.72.90.23:8080/student/viewStudents

Register Student

Students List

Student ID	StudentName	Student Addrs	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
1	Akash	PUNE	24	B.E. Mechanical	88	2022	edit	delete