Hosting Tomcat Website in a Docker Container using Dockerfile

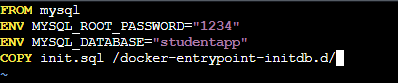
* First we need to create a container for mysql and then create a database.
* Create a directory mysql.
* Create a file called Dockerfile in the mysql directory.
* Add the following content in the Dockerfile of mysql.

FROM mysql

ENV MYSQL\_ROOT\_PASSWORD="1234"

ENV MYSQL\_DATABASE="studentapp"

COPY init.sql /docker-entrypoint-initdb.d/



**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.

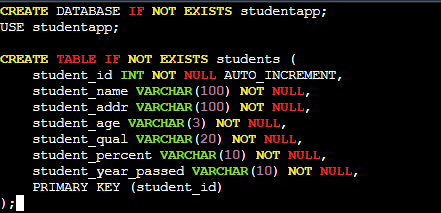
**ENV** -> These lines set environment variables within the Docker container:

**MYSQL\_ROOT\_PASSWORD** is used to set the root password for the MySQL server to "1234".

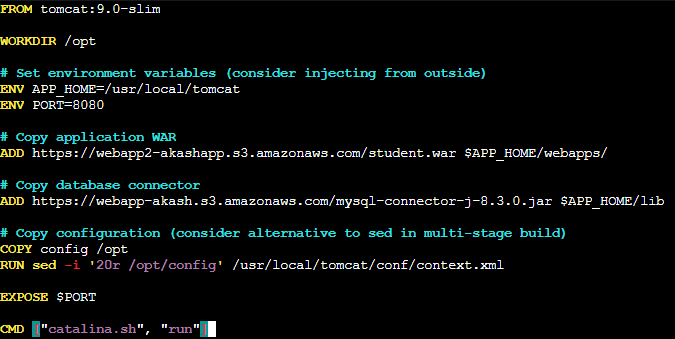
**MYSQL\_DATABASE** specifies the name of the default database to be created when the MySQL server starts.

**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

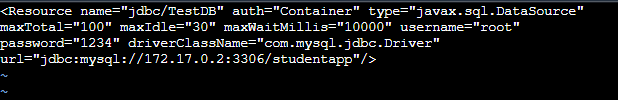


* Now come back to the /home/ec2-user.
* Create a Dockerfile here for our application.

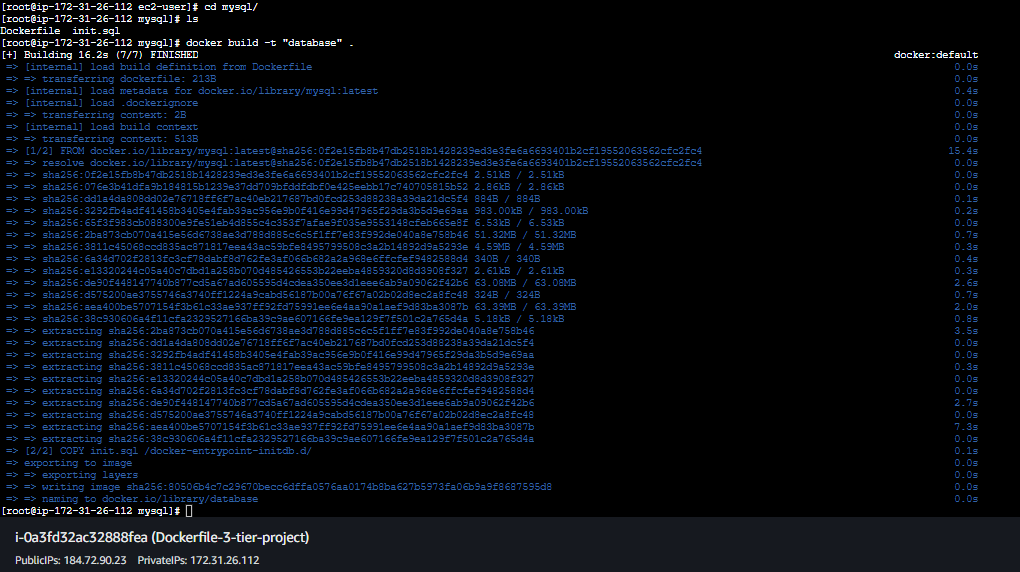


Dockerfile Explaination:

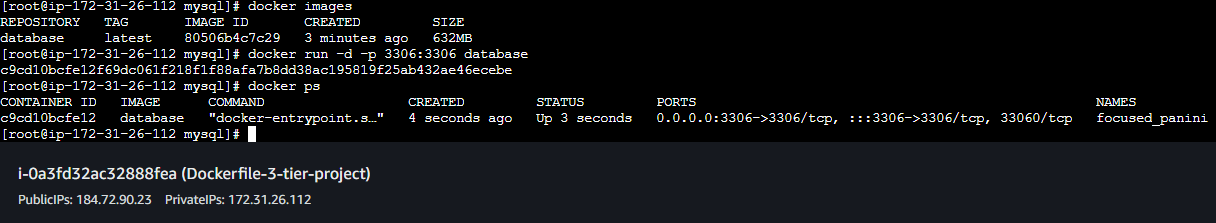
* **Base Image: FROM tomcat:9.0-slim :**
* This line specifies the base image as Tomcat 9.0-slim, which is a lightweight version of the official Tomcat 9.0 image.
* **Working Directory: WORKDIR /opt:**
* This line sets the working directory to `/opt` within the Docker container. All subsequent commands 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



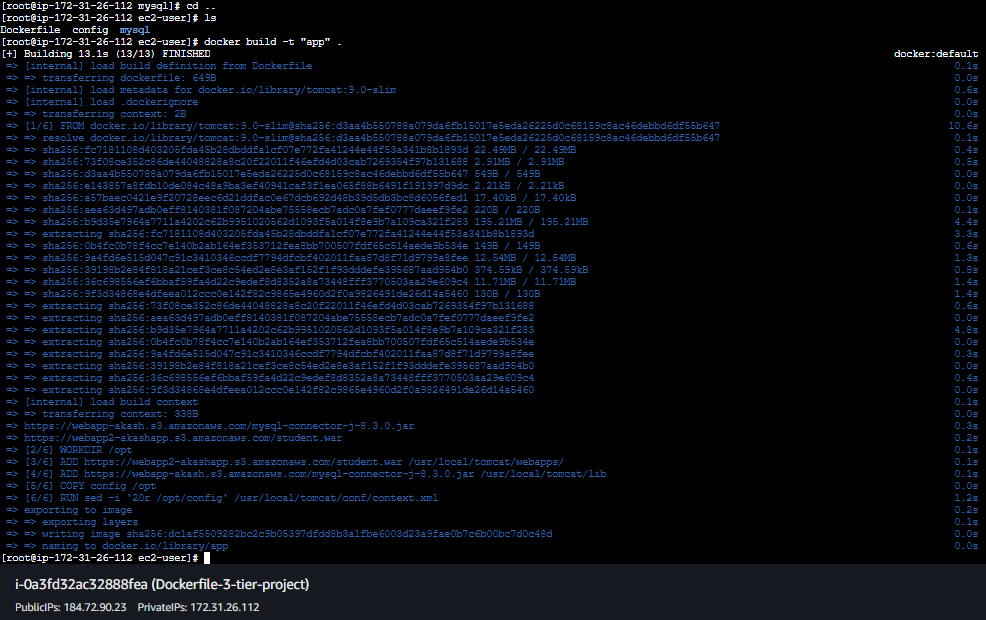
* Here I have given the IP address of the mysql container.
* When mysql container is created for the first time it has the same IP as given.
* And if there is already one mysql container present and again you create another mysql container the IP increases by 1.
* Now save the file and exit.
* Now our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



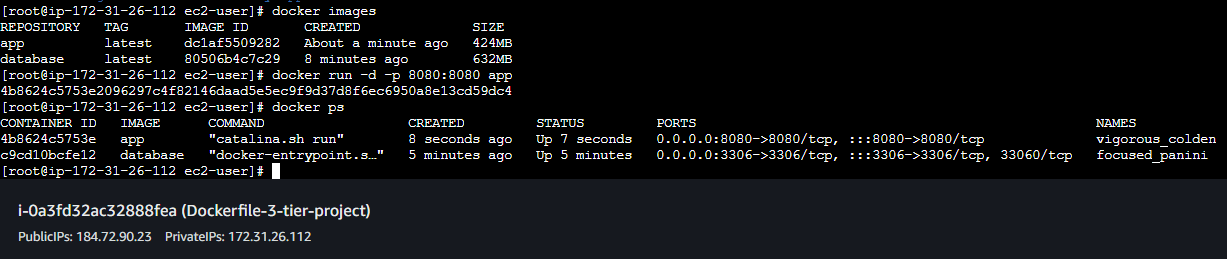
* Now the image is created from Dockerfile.
* Hit command “docker images” to check the image.
* Hit command “docker run -d -p 3306:3306 database” to run the image in background to create a container.



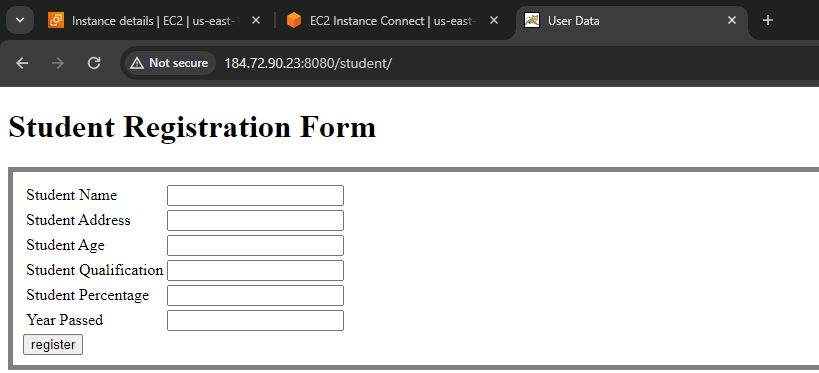
* Now go back to /home/ec2-user.
* Here also we have to run the Dockerfile for our application.
* Hit command ‘docker build -t “app” .’



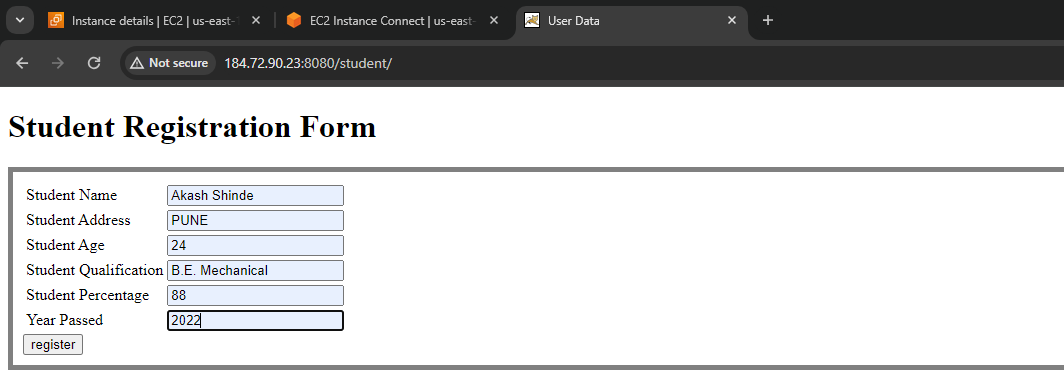
* Now the image is created.
* To check hit command “docker images”.
* Now we need to run the image so that the container will be created.
* Hit command “docker run -d -p 8080:8080 app”.
* To check if the container is running hit command “docker ps”.



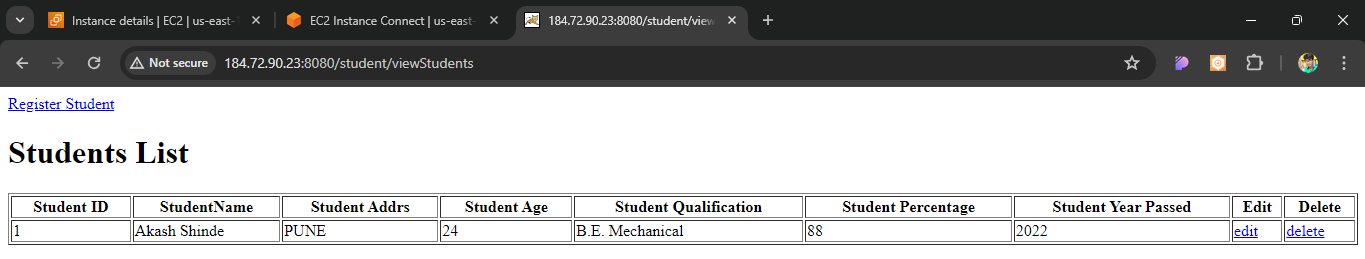
* Now hit the IP address of the instance with port 8080/student.



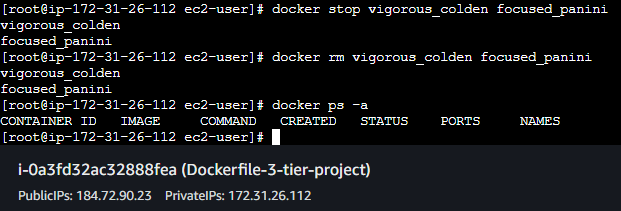
* Page is loading successfully.
* Now fill the form and click register to check if our data is going to database.



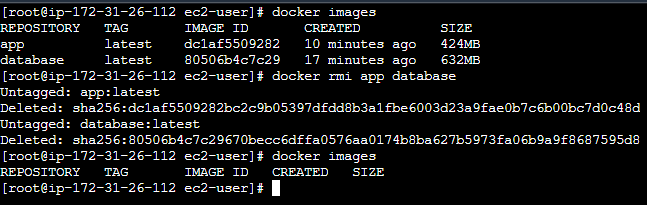
* Our data is going to the database.



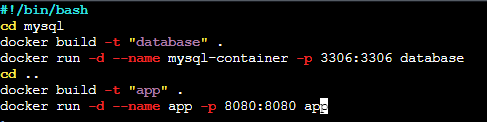
* Here we are building the Docker images from the Dockerfile separately by typing commands.
* We can do the same with script.
* We can write all the commands in a script and execute the script.
* To do the same follow the steps.
* First we need to stop the containers and remove them.



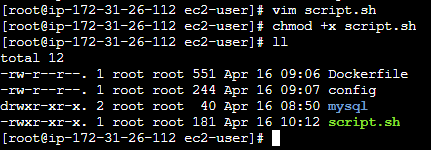
* We also need to remove the images.



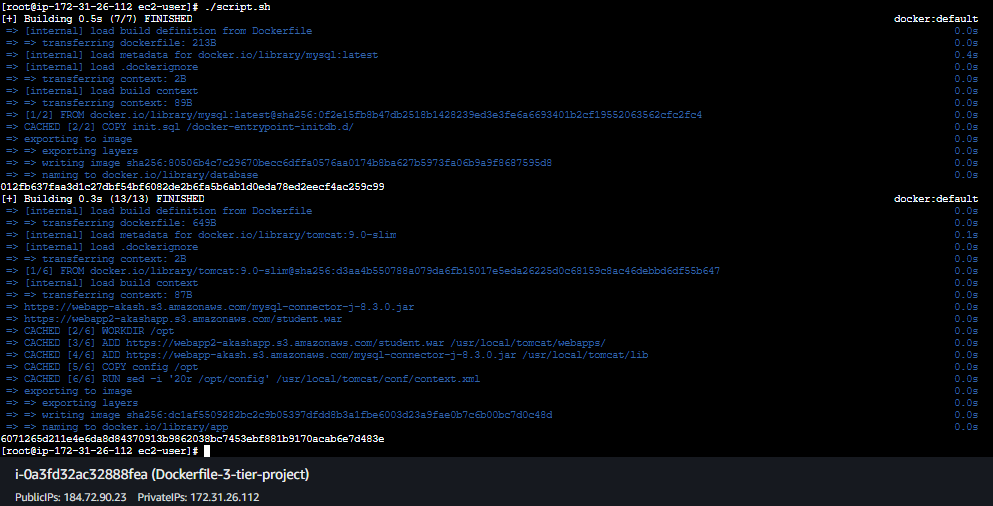
* Now create a file named script.sh in /home/ec2-user.
* Add the commands in the script.sh file.



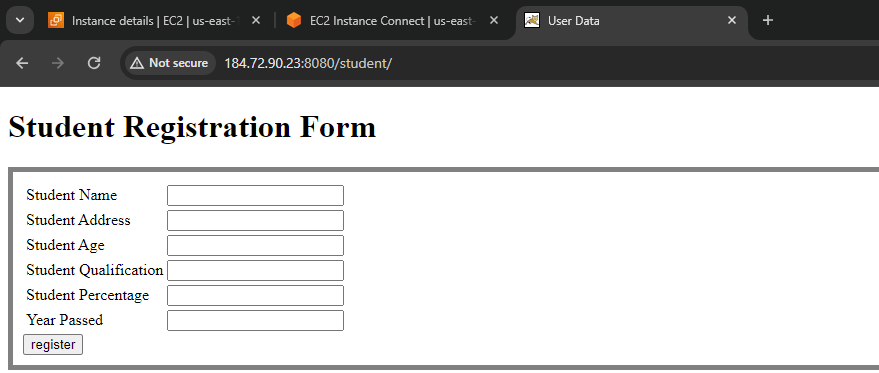
* Now we need to give the execute permissions to the script.sh file.
* Hit command “chmod +x script.sh” to give execute permissions.



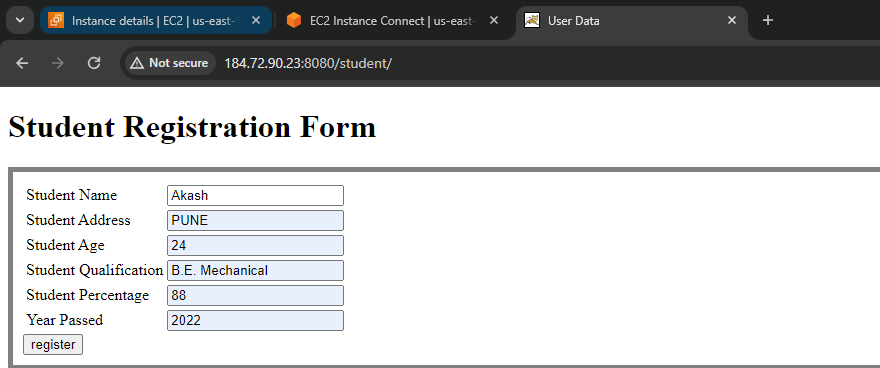
* Now run the script.
* Hit command “./script.sh” to run the script.



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



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



* Data is being saved successfully in the database.

