**COMSATS UNIVERSITY ISLAMABAD, ATTOCK CAMPUS**



**Mid Task**

**Submitted To:** Dr. Farhan Aadil

**Submitted By:** Muhammad Shehzad Khan

**Reg #** FA19-BCS-038

**Class:** BCS 8(A)

**Subject: TICS**

**Dated:**  10may 2023

**Docker-Compose**

**Screenshorts of task:**

* Define a service named web in your docker-compose.yml file. This service should use the httpd:2.4 Docker image and should mount the current directory as a volume at /usr/local/apache2/htdocs.

services:

  web:

    image: httpd:2.4

    ports:

      - "80:80"

    volumes:

      - .:/usr/local/apache2/htdocs

**Description:**

The "image" field specifies the Docker image to use, which in this case is httpd:2.4. The "ports" field maps the container's port 80 to the host's port 80, allowing the web server to be accessed through a browser. Finally, the "volumes" field specifies that the current directory should be mounted as a volume at /usr/local/apache2/htdocs in the container.

* Define a service named db in your docker-compose.yml file. This service should use the mysql:5.7 Docker image and should set the environment variables MYSQL\_ROOT\_PASSWORD and MYSQL\_DATABASE.

db:

    image: mysql:5.7

    environment:

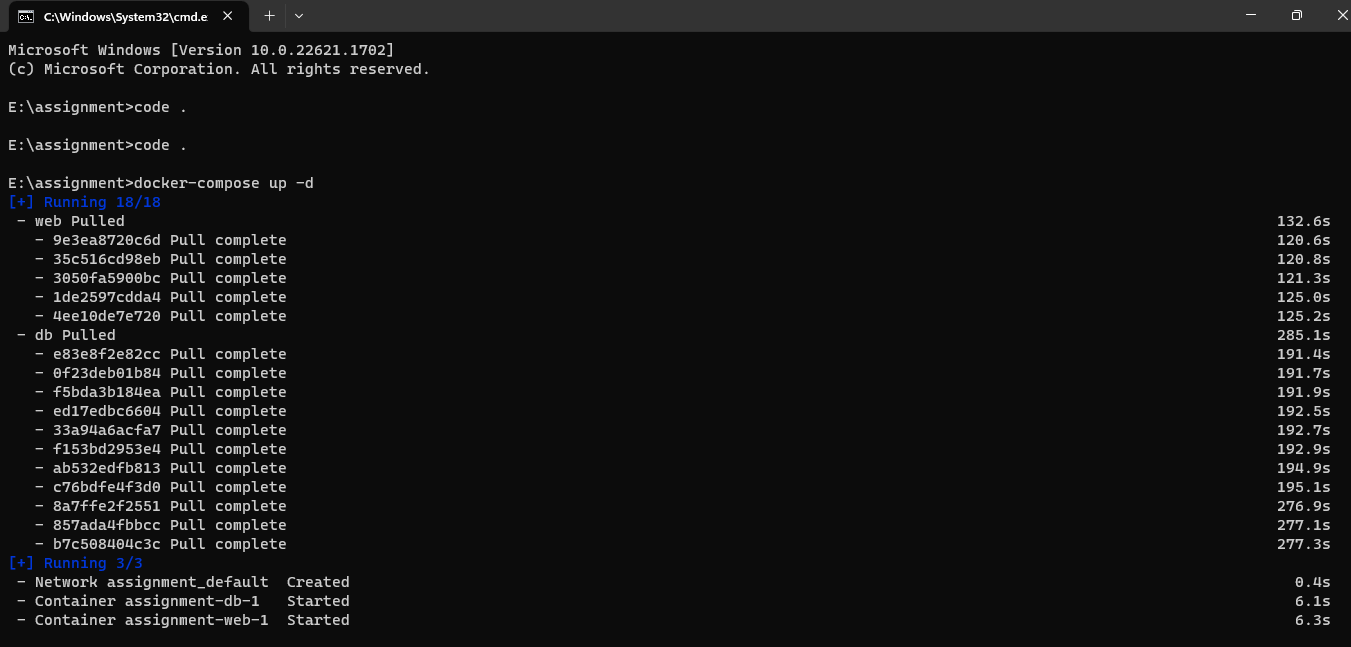
      MYSQL\_ROOT\_PASSWORD: mysecretpassword

      MYSQL\_DATABASE: mydatabase

**Description:**

The "image" field specifies the Docker image to use, which in this case is mysql:5.7. The "environment" field specifies a list of environment variables to set in the container. In this case, we're setting the MYSQL\_ROOT\_PASSWORD to "mysecretpassword" and the MYSQL\_DATABASE to "mydatabase". These environment variables will be available to the MySQL instance running in the container.

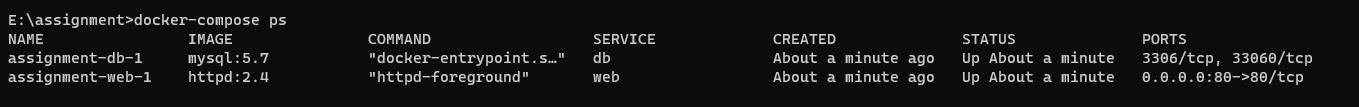
* Build and start the application using Docker Compose by running the command docker-compose up -d in your terminal. The -d flag runs the containers in the background.



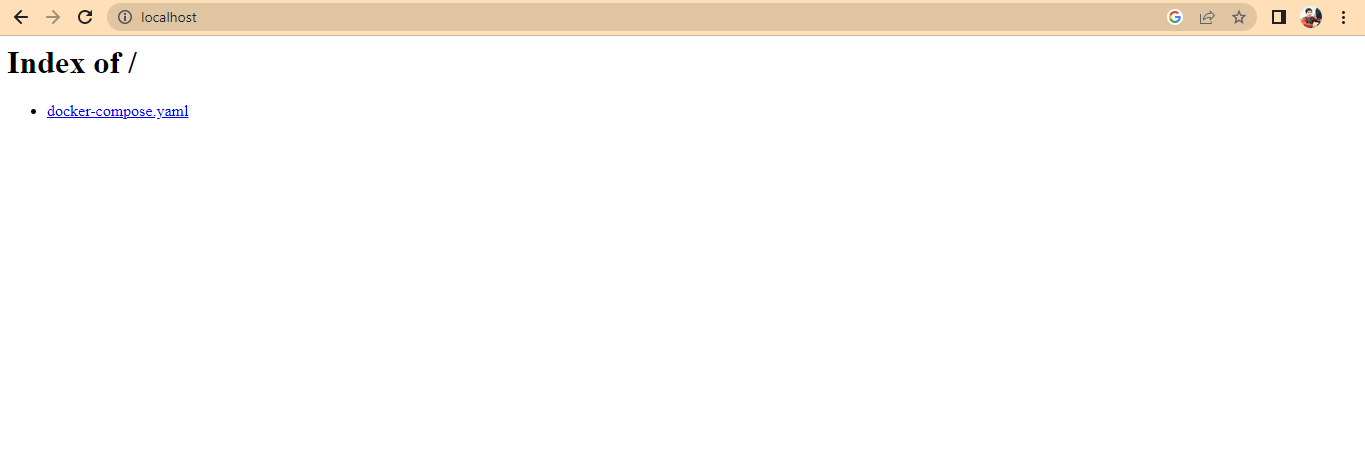
**Description:**

This will start the containers in the background using the configuration specified in the docker-compose.yml file. The "-d" flag runs the containers in detached mode, meaning they will continue to run in the background even after you close the terminal.

* Verify that the containers are running by running the command docker-compose ps in your terminal. You should see two containers listed: one for the web service and one for the db service.



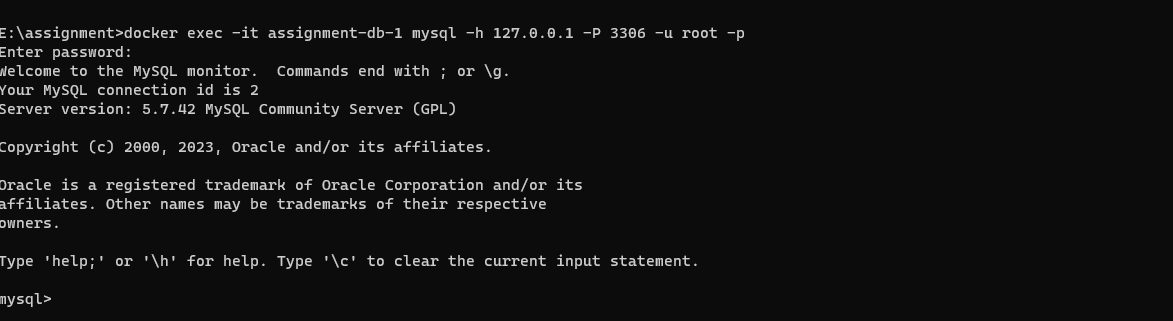
* Open a web browser and navigate to http://localhost. You should see the Apache default page, indicating that the web service is working correctly.



**Description:**

When you visit [**http://localhost**](http://localhost/)**:80** in your browser, it should display the default Apache web page, indicating that the web service is working correctly. This means that the httpd:2.4 Docker image has been successfully downloaded and is running in a container, serving web pages from the mounted volume at /usr/local/apache2/htdocs.

* Connect to the MySQL database by running the command mysql -h 127.0.0.1 -P 3306 -u root p in your terminal. You should be prompted for the root password you set in your docker-compose.yml file. Once you enter the password, you should be connected to the MySQL shell, indicating that the db service is working correctly.



**Description:**

The docker exec command runs the mysql command inside the specified container, using the same command-line arguments that you would use to connect to the MySQL server running on the host machine.

The -it flags in the docker exec command allow you to interact with the MySQL shell by attaching your terminal to the container's standard input and output.

* Stop the containers by running the command docker-compose down in your terminal.



**Description:**

Shutdown the containers.

* remove the Docker images for the web and db services by running the command docker rmi httpd:2.4 mysql:5.7.



**Building and running the multi-container application in Docker Compose:**

Building and running a multi-container application in Docker Compose can be a powerful way to develop and deploy complex applications with multiple dependencies. By defining each service as a separate container with its own configuration, you can easily manage and scale your application, while keeping each component isolated and easily configurable.

Building and running a multi-container application in Docker Compose involves defining the different services that make up your application in a docker-compose.yml file. Each service is defined as a separate container with its own configuration, such as the image to use, environment variables, and volumes.

Once you have defined your services in the docker-compose.yml file, you can use the docker-compose command-line tool to build and start your application. The docker-compose up command builds the images for each service, creates the containers, and starts the application. You can use the -d flag to run the containers in the background.

Once your application is running, you can interact with it using your web browser or by connecting to the various services from the command line. For example, you can connect to the MySQL database using the mysql command-line tool and run SQL queries, or you can run other commands inside the containers using the docker exec command.