**Light Container**

**by**

**Ishraq Ahmed Esha**

**19301261**

**A Lab Assignment 3 submitted to the CSE484 Cloud Computing**

**Course of Sec: 1**

**Brac University**

**November 17, 2022**

**Answer to the Question No 1**

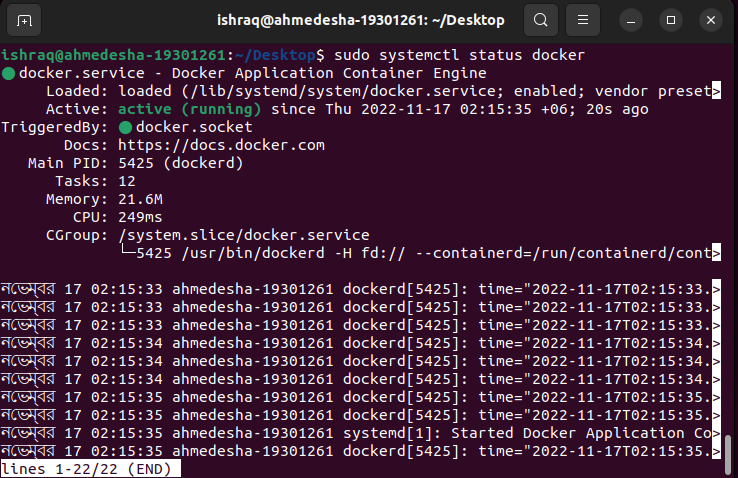
To get the latest version we will install Docker from the official Docker repository and will add a new package source, add the GPG key from Docker to ensure the downloads are valid, and then install the package. Now we have to follow a few steps underneath.

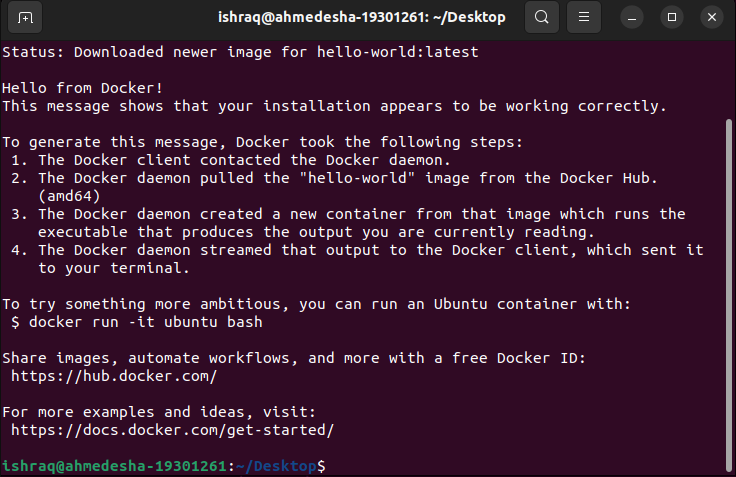
1. Update existing list of packages
   1. “***sudo apt update***”
2. Install few prerequisite packages which let “apt” use packages over HTTPS
   1. “***sudo apt install apt-transport-https ca-certificates curl gnupg-agent software-properties-common***”
3. Now add the GPG key for the official Docker repository to the system
   1. “***curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -***”
4. Add a key
   1. “***sudo apt-key fingerprint 0EBFCD88***”
5. Have to add the docker repository to APT source
   1. “***sudo add-apt-repository \***

***“Deb [arch=amd64] https://download.docker.com/linux/ubuntu \***

***$(lsb\_release -cs) \***

***stable”*** ”

1. Install from the docker repo instead of the default ubuntu repo
   1. “***apt-cache policy docker-ce***”
2. Again update the list of packages
   1. “***sudo apt-get update***”
3. Install the docker
   1. “***sudo apt-get install docker-ce docker-ce-cli containerd.io***”
4. Check the docker status
   1. “***sudo systemctl status docker***”
5. Now just check how to work with docker images
   1. “***sudo docker run hello-world***”



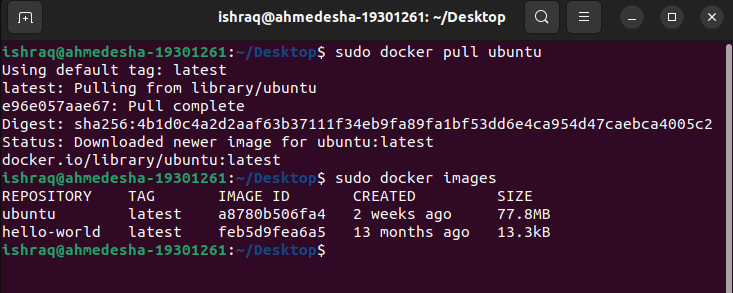
**Answer to the Question No 2**

Few Basic docker command are:

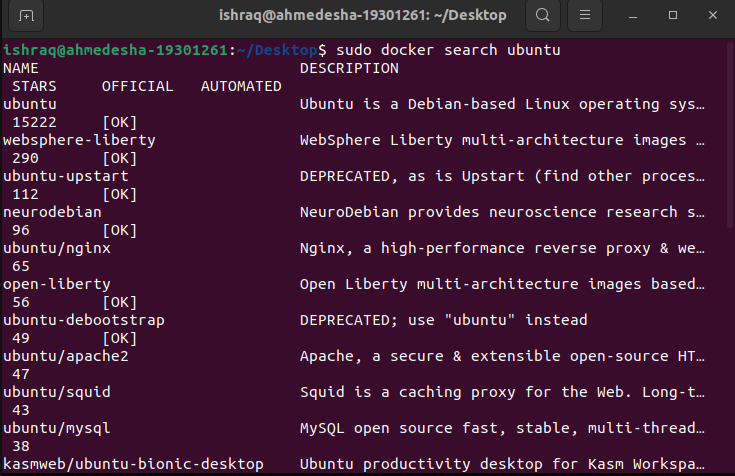
1. **pull**: pull command is used to pull an image or a repository from a registry or docker hub

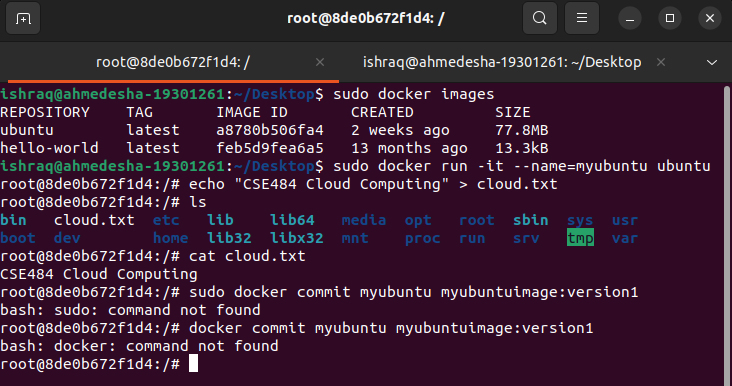
“***sudo docker pull ubuntu****”*

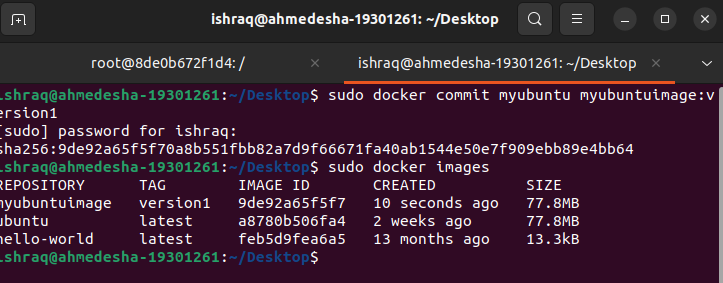
1. **image**: To see and manage the images this image command is used “***sudo docker images****”*

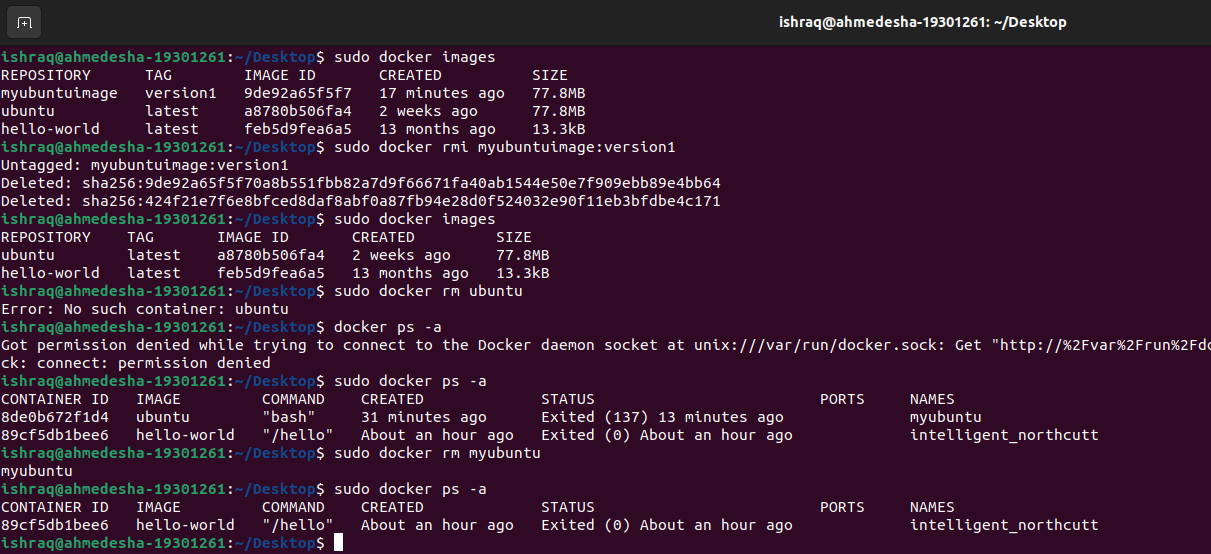


1. **search**: In the Docker Hub search images **“*sudo docker search ubuntu***”
2. **build**: When we have to build an image from a Dockerfile we have to use the build command “***sudo docker build https://github.com/docker/rootfs.git#container:docker***”



1. **run**: To run a command in a new container as we have run the hello-world image above “***sudo docker run hello-world***” . Now if we want to create an ubuntu container and make some changes inside it. We will create a new file inside the container. “***sudo docker run -it --name=myubuntu ubuntu***”
2. **commit**: To save changes to a new image we use a commit command with some message. Now keeping the container running we have to open a new terminal and execute the following command for commit “***sudo docker commit myubuntu myubuntuimage:version1***” . The new file which we created earlier that’s going to be committed with version1 message in new repository of myubuntuimage. 



1. **stop**: To stop a running container we have to use the following command “***sudo docker stop myubuntu***” by executing this command running container myubuntu will stop.
2. **rmi**: To remove one or more images we use the following command “***sudo docker rmi myubuntuimage myubuntu:version1***” Here we will remove the newly created image which was myubuntuimage and Tag was version1
3. **rm**: To remove one or more containers we use the following command “***sudo docker rm myubuntu***” . Earlier we stopped the myubuntu container. Now if we execute the command then the myubuntu container will be removed. 

**Answer to the Question No 3**

To create a docker image using dockerfile we have to follow the following steps:

1. First of all, we have to create a directory to store the dockerfile. Using the following command “***mkdir Dockerfiles***”
2. In the dockerfile directory create a file named dockerfile and edit the file with the commands that want to execute and save the file. Use the following command “***vim Dockerfiles/dockerfile***”
3. Here we will build an apache web server image as an example. In the dockerfile we have to write the following commands to execute the server and show output.

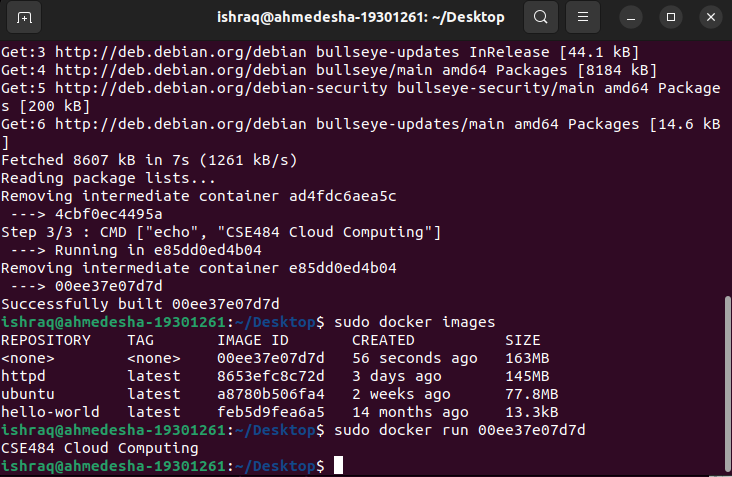
“***#Getting the image from docker hub***

***FROM httpd***

***RUN apt-get update***

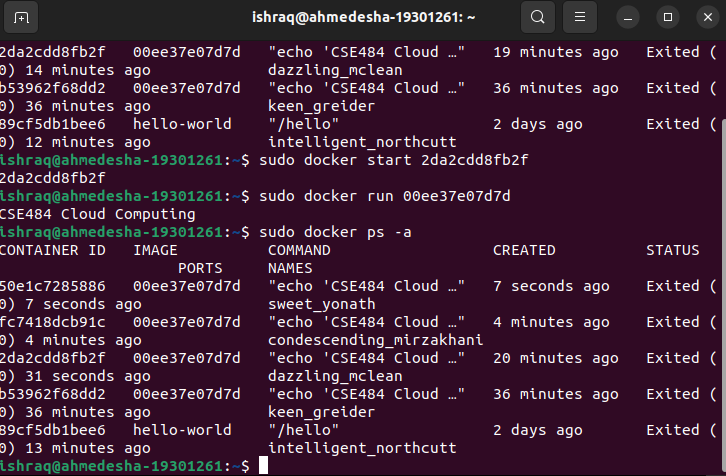
***CMd [“echo”, “CSE484 Cloud Computing”]***”

1. Now to build the dockerfile run the following command from the terminal “***sudo docker build dockerfiles/***” . After completing the command “*successfully built*” message will be shown.
2. Now check the images with the following command “***sudo docker images***” . Here we can see the new image ID after successful build of our image from the dockerfile.
3. We have to run the image to see our output with the following command: “***sudo docker run 00ee37e07d7d***” . After that we can see our output “*CSE484 Cloud Computing*”. So we have successfully created a docker image from the dockerfile and run it.



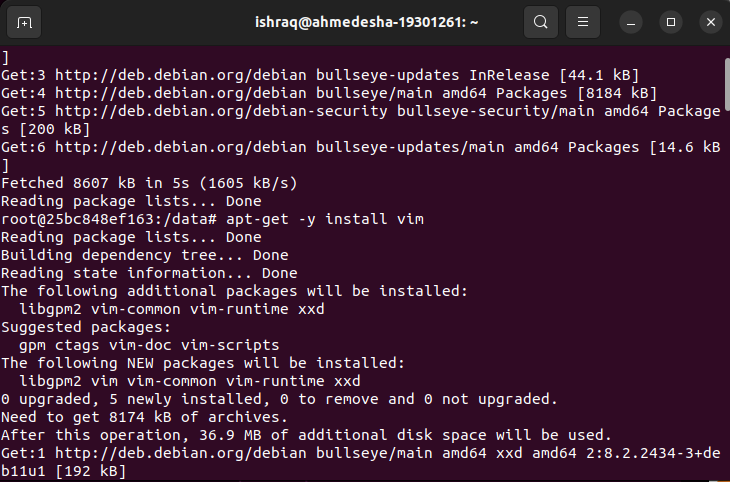
**Answer to the Question No 4**

To run a single container we have to start the container with the container ID, with following command “***sudo docker run 2da2ccdd8fb2f***” then to show the output of the docker image we have to run the docker image file with the docker image id, with following command “***sudo docker run 00ee37e07d7d***” . After that we can see the output as “*CSE484 Cloud Computing*”. Then to show the status of all containers we will execute the following command “***sudo docker ps -a***” . After executing this command we can see the container ID, image ID, command, created time, status, ports and names. Underneath the image which is shown clearly.



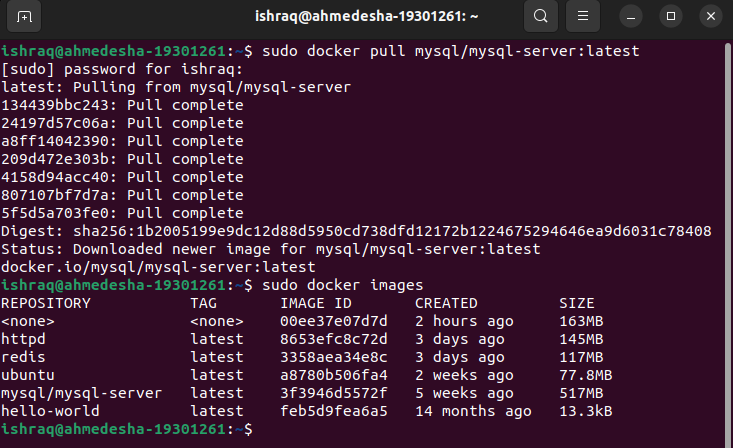
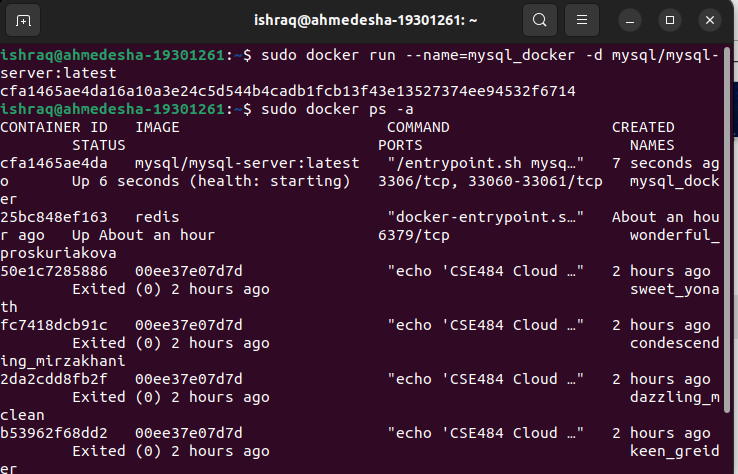
**Answer to the Question No 5**

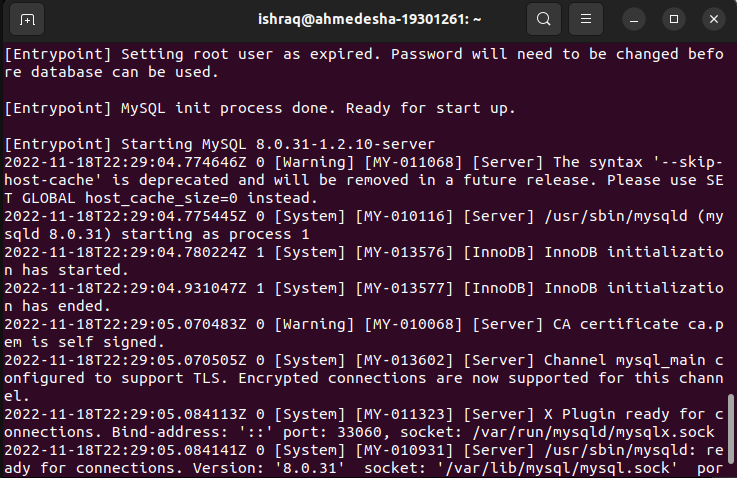
I will run the docker container in interactive mode. The advantage of docker interactive mode is that it allows us to execute commands at the time of running the container. Here I will use the Redis container. I can first start a Redis docker container in the background using the following command “***sudo docker run -d redis***” . Now to check the ID of the running containers we will use “***sudo docker ps -a***” . Lastly, using the ID of the container, I can use the following command to issue a different command to the running container in interactive mode “ ***sudo docker exe -it redis-cli***” . Here I am issuing the command “redis-cli” on the container & “-it” for interactive mode. Or I can use bash to execute commands. This will open a redis-cli command prompt where I can execute commands on the Redis server. After that I execute the following command to install the vim in the running container. First of all, to generate a cache in the image following command “***apt-get update***” then install the vim package with the following command “***apt-get -y install vim***” . By following these steps I can run a container in an interactive mode and install the packages in the running container.



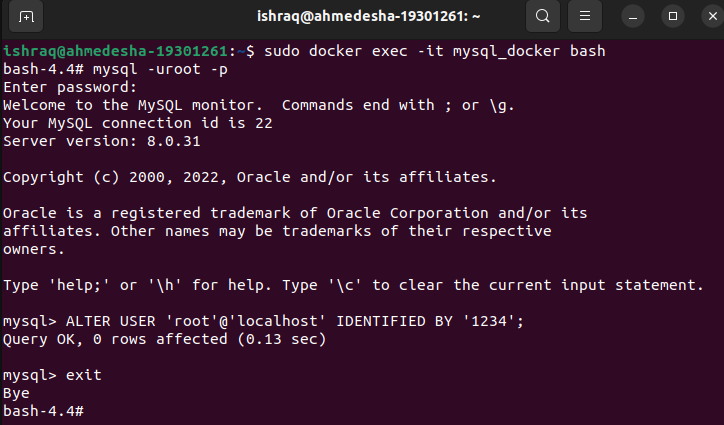
**Answer to the Question No 6**

To run a database container in the background, showing logs, and some sql queries in interactive mode we have to follow below steps: Here I am using MySQL database.

1. Pull the MySQL Docker Image: “***sudo docker pull mysql-mysql-server:latest***”
2. To verify the image I can use the “***sudo docker images***” command and will see the Image ID.
3. Deploy the MySQL Container: “***sudo docker run --name=mysql\_docker -d mysql-server:latest***” . Here I have provided a container name as mysql\_docker and “-d” for running in the background.
4. I can check the MySQL container running status with the following command “***sudo docker ps -a***”
5. Connect to the MySQL Docker Container: Before connecting with the MySQL server container with the host, I need to make sure the MySQL client package is installed with the following command “***apt-get install mysql-client***”
6. Then have to open the logs file for MySQL container to find the generated root password with the following command “***sudo docker logs mysql\_docker***” . From here I have to copy the generated password for the next step.



1. Now will go to the bash shell of the MySQL container with the following command “***sudo docker exec -it mysql\_docker bash***”
2. Now I will write a sql query to change the root password from the container bash. Executing the following sql query I can change the root password for the database “***ALTER USER ‘root’@’localhost’ IDENTIFIED BY ‘1234’;***”



1. Few sql queries to create a database and create a table then insert some data and show that.

“***CREATE DATABASE Test;***”

“***CREATE TABLE StudentInfo(***

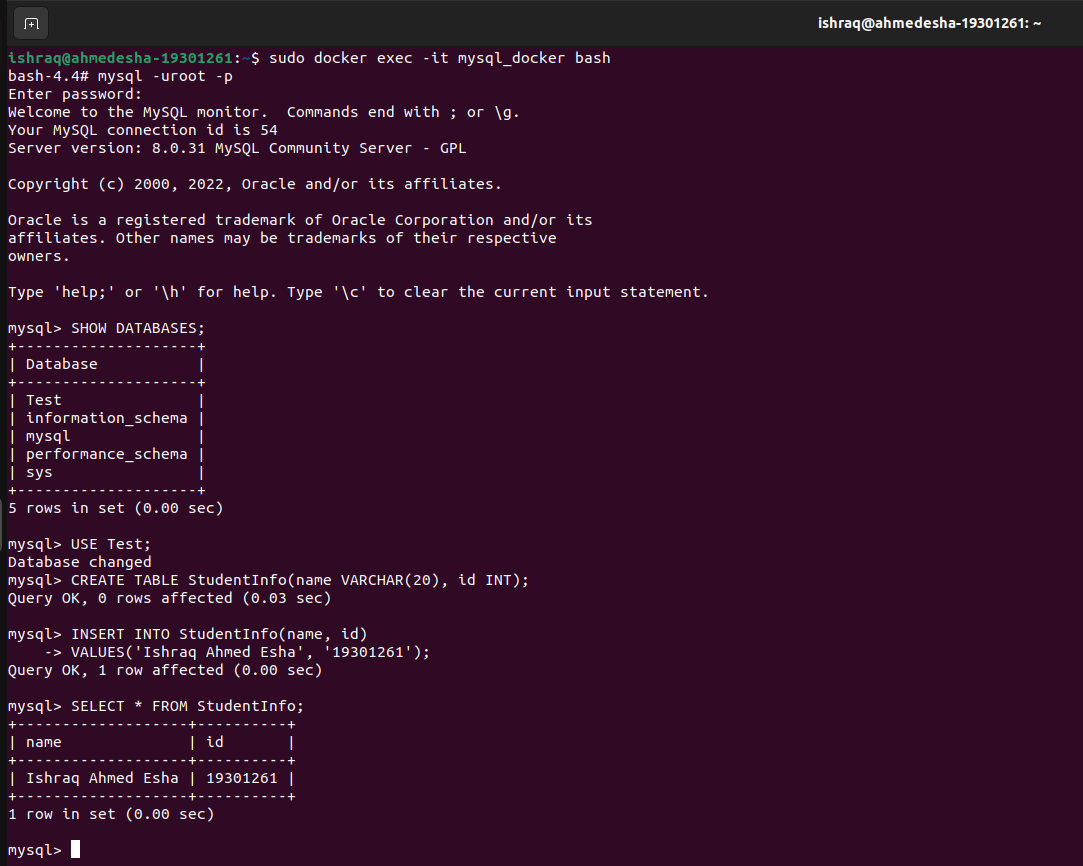
***name VARCHAR(20),***

***id INT);***”

“***INSERT INTO StudentInfo (name, id)***

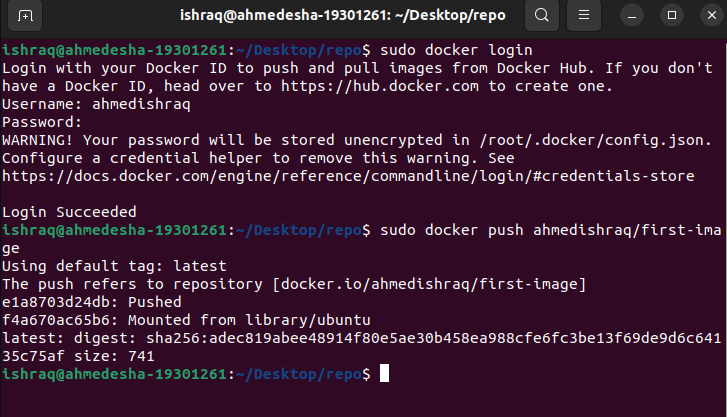
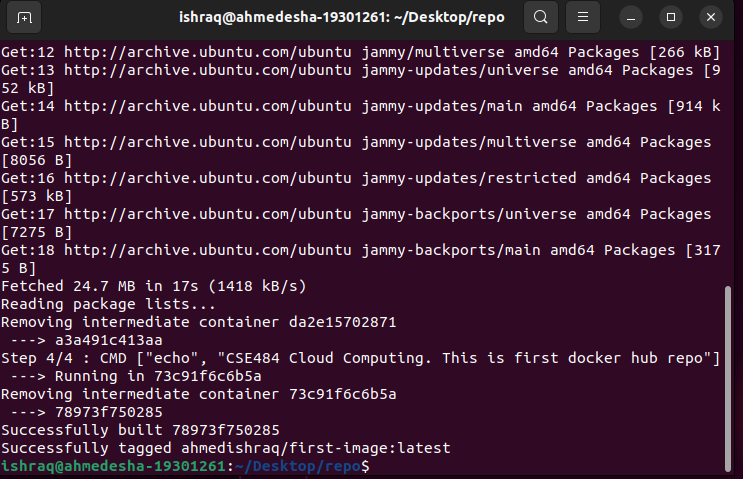
***VALUES***

***(‘Ishraq Ahmed Esha’, ‘19301261’);***”



**Answer to the Question No 7**

I will create a new dockerfile named dockerfile then will edit that with desired commands and save that. After that I will build the docker image with my docker hub username and repository name with the following command “***sudo docker build -t ahmedishraq/first-image .***”. Then I have to push the image to docker hub, to do so I need to login my docker hub account with the following command “***sudo docker login***” . Now push the docker image with the following command “***sudo docker push ahmedishraq/first-image***”



**Answer to the Question No 8**

To make my own private registry have to follow the steps:

1. First of all, have to create own registry, to do so first of all install the docker-compose using following command “***sudo apt install docker-compose***”
2. Then edit the docker-compose.yml file with vim with the following commands

“***version: ‘3’***

***services:***

***registry:***

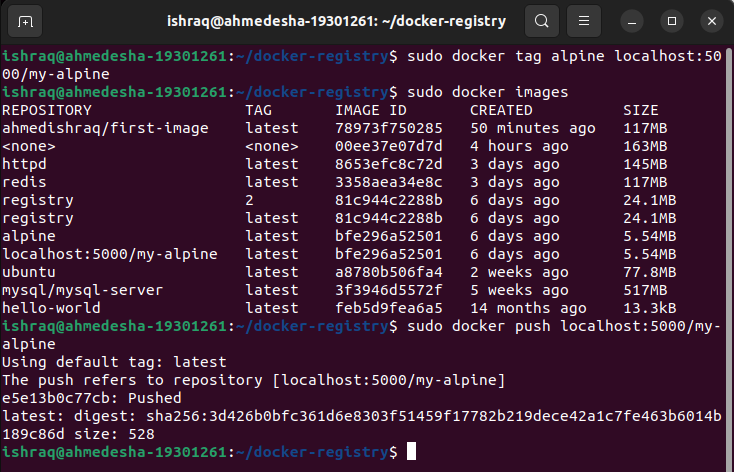
***image: registry:2***

***ports:***

* ***“5000:5000***”

The configuration uses the official registry image and forwards the port 5000 of the container to the host machine. This allows it to send requests to port 5000 on the server that runs the registry.

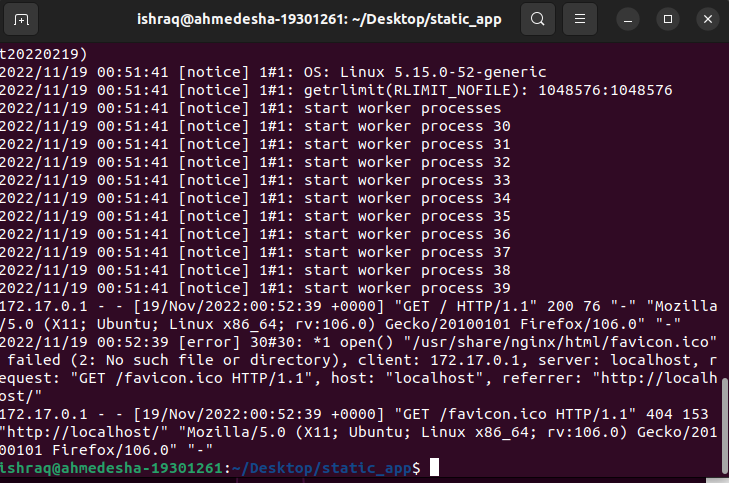
1. Now run the container with the following command “***sudo docker-compose up -d***”
2. Now I am ready to push an image to the registry, here I am going to use the alpine Linux image because it is small and downloads fast. Pull the image with the following command “***sudo docker pull alpine***” . Then add the tag “***sudo docker tag alpine localhost:5000/my-alpine***” . By using “***sudo docker images***” we can see the newly added images.
3. Now I can push the image to my private registry with the following command “***sudo docker push localhost:5000/my-alpine***” . This only works if my host registry on my local machine. If I want to host it on a server, I will need a secure SLL connection.

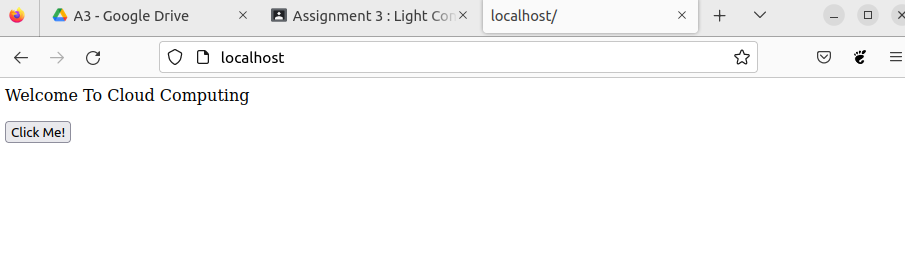


**Answer to the Question No 9**

I will create a small simple webapp that will run in the background of my host machine and I will browse that website from my local host machine. To do this follow the below steps:

1. Create a directory in home “**mkdir static\_app**” there I will create a index.html file for the website instruction and edit that file with vim “***vim index.html***” “***<p>Welcome To Cloud Computing <button type=”button”>Click Me!</button>***”
2. Now I have to create the dockerfile for the image with the following command “***vim dockerfile***” edit this file with following instructions “***FROM nginx:alpine COPY . /usr/share/nginx/html***” save the dockerfile.
3. Now it’s time to build the image with the following command “***sudo docker build -t static-app:v1 .***”
4. Image have build successfully now time to run the container and specify the port for the localhost with the following command “***sudo docker run -d --name=app-container -p 80:80 static-app:v1***”
5. Now if I type the localhost and port 80 to my host machine then I will see the website.





**Answer to the Question No 10**

If I push the webapp image into a public registry like docker hub, then anyone can pull it from the hub and if they run the image into their local host machine they also can have the local access of the webapp.

