**Docker:**

Docker is a tool for running an application in an isolated environment.

Advantages:

* Same environment
* Sand boxing the project
* And it works everywhere

They start up in seconds, use few resources and space.

Container is a running instance of an image (template of environment).

Images are defined in a docker file, building image using docker file which runs in container.

There are some pre-defined images available on the docker hub and we can use the concept of layering and build our application on top of an image.

If we made any changes in the file it won’t be reflected because when we build the image we made a copy of the same for retaining the persistence we can use the concept of Volumes.

Volumes:

* Persist and share data between containers
* Share folder between host and container, mounting a local director on pc as volume inside the container.

Creating a basic application hello world:

Creating a basic app in php:

Index.php:

<?php

echo “my first application”

now specifying the dependencies for our application in dokcerfile

Docker file (code to configure our environment):

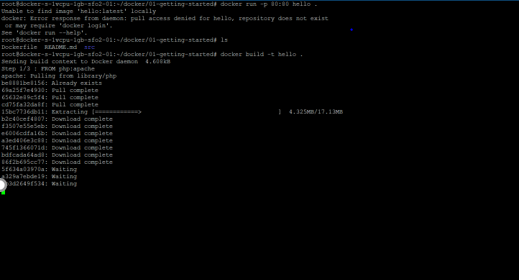
* Using an already build image in our docker file and we build on top of it.

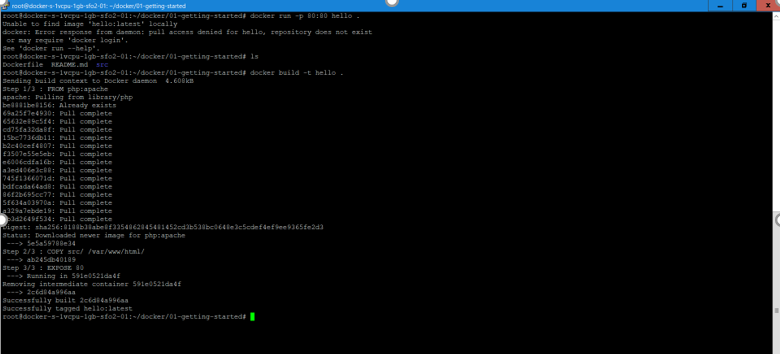
From php: 7.0-apache // defining the base image

COPY // copying the content where apache will look in its own file system

EXPOSE 80 //container will listen to port 80

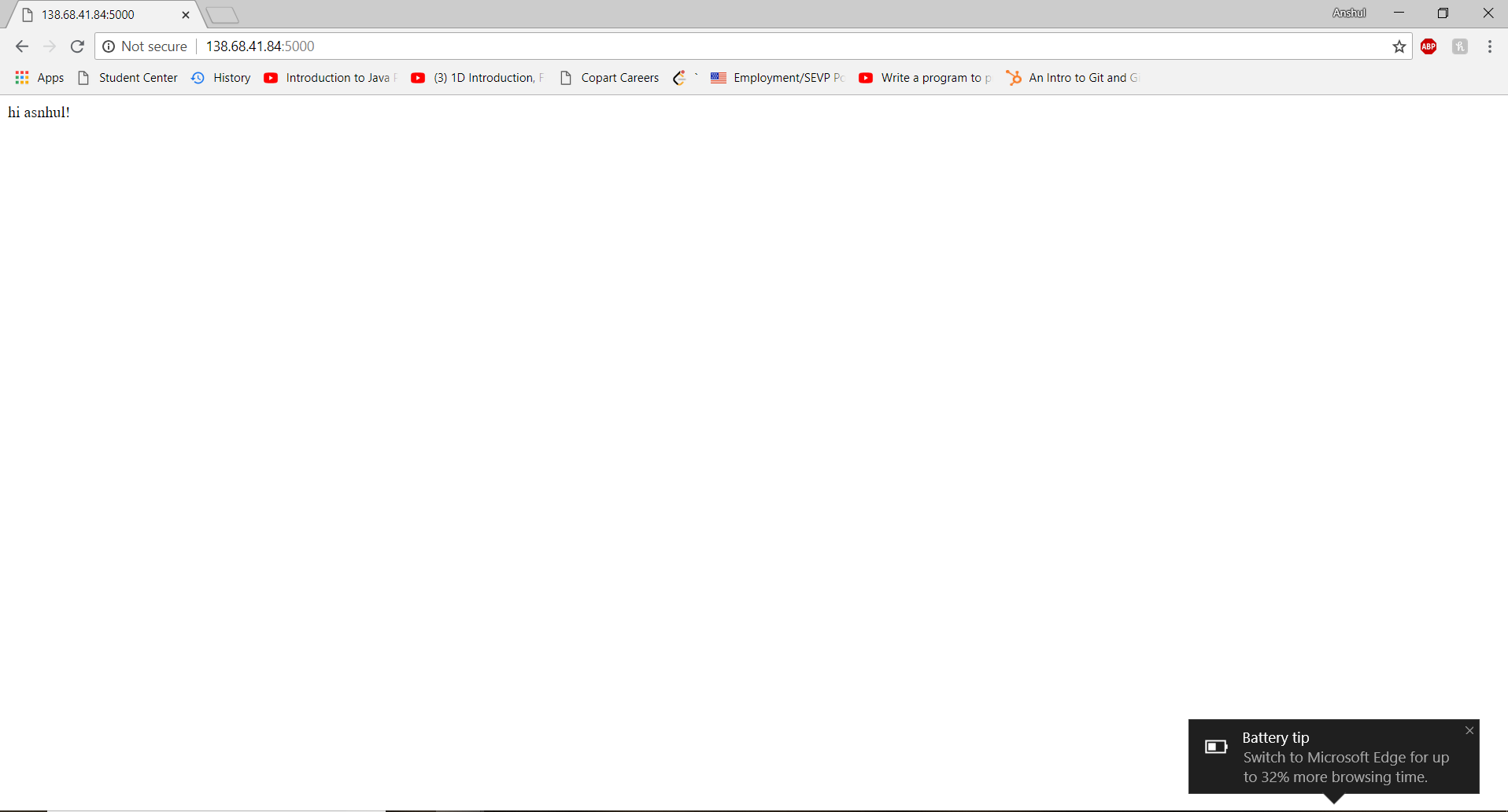
Building the application, downloading the dependencies specified in the dockerfile:





Getting the output on the local host (Digital ocean):

Running the docker file docker run -p 80:5000 app

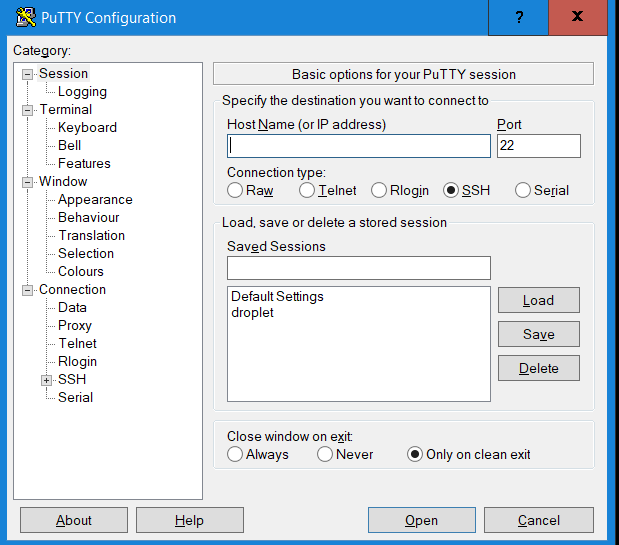


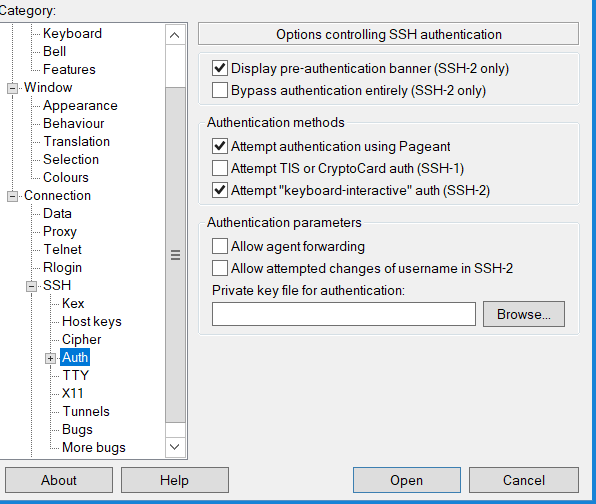
**stop all containers:**  
docker kill $(docker ps -q)

**remove all containers**  
docker rm $(docker ps -a -q)

**remove all docker images**  
docker rmi $(docker images -q)

Installing docekr on an instance of linux in ditigal ocean and connecting to it using Putty on windows and using public key encryption to make the connection secure:





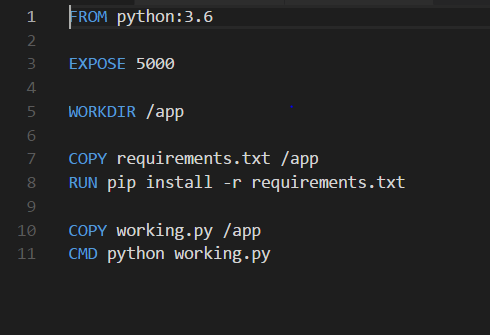
Now deploying web application on docker:

* Application running in one container
* Database in another container

a) app.py - it has the flask application which connects the application with the database and has the rest api.

b) init.sql -has all the queries to initialize the database.

c) Creating a Docker image:



it contains instructions describing our image and allows it to build automatically, which can run in container.

creating a docker file in our application directory.

1.python (python run time image)

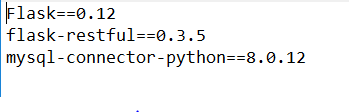
2.exposing the port in container

3.setting the working directory

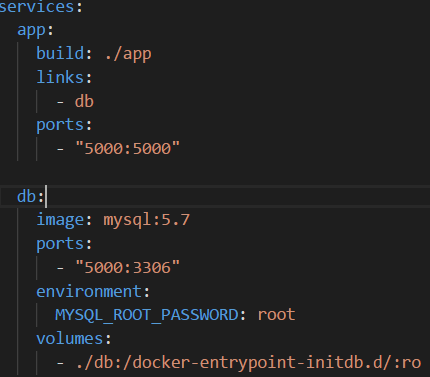
4.installing the specified requirements

5.running the application

d) creating requirement file:



f) Creating the Docker- Compose file:(yml)



The separate parts of your application (the different images) should run in separate

pods.

using 2 services, one for the api and one for the database, using docker-compose for 2 containers into one application.

one container has the application and the other has the database.

build: specifies the directory where the Dockerfile has instructions for building the service

links: links services to other containers.

port: used for mapping the port (host and Container)

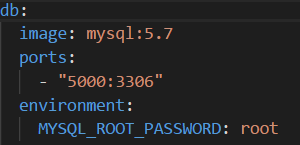


image: using existing image from the repository

volumes: initializing the database with our schema

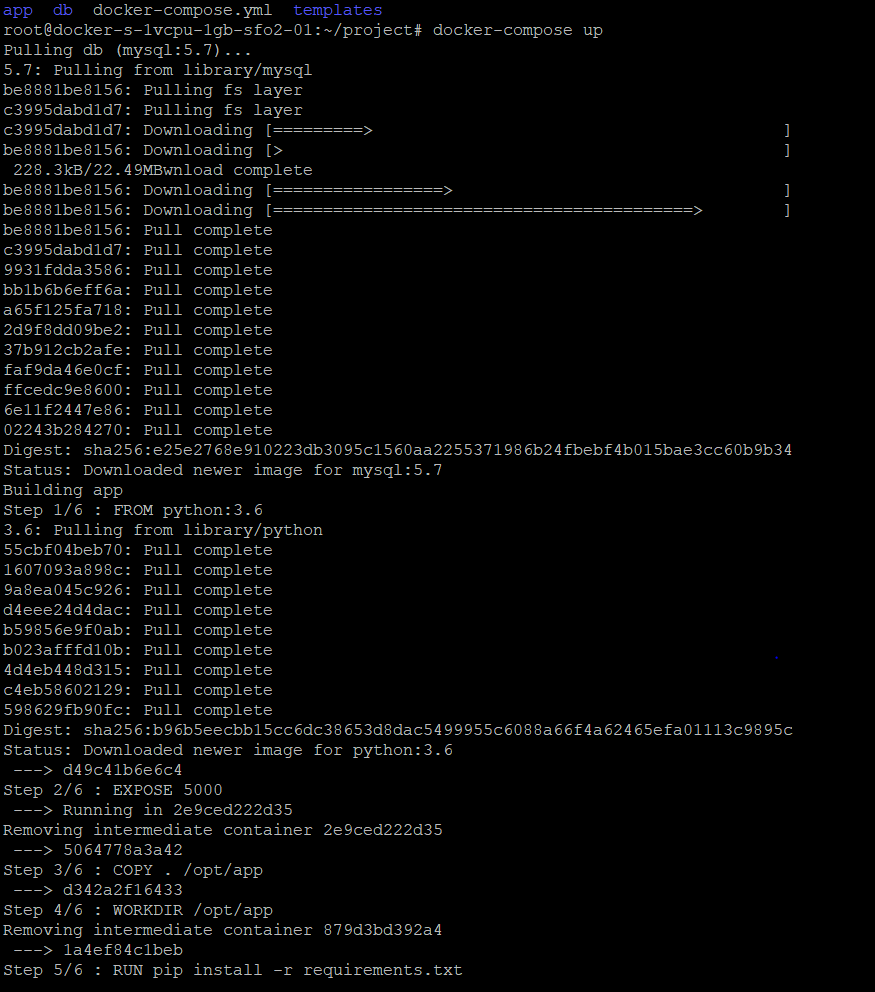
Now connecting the application to the database:

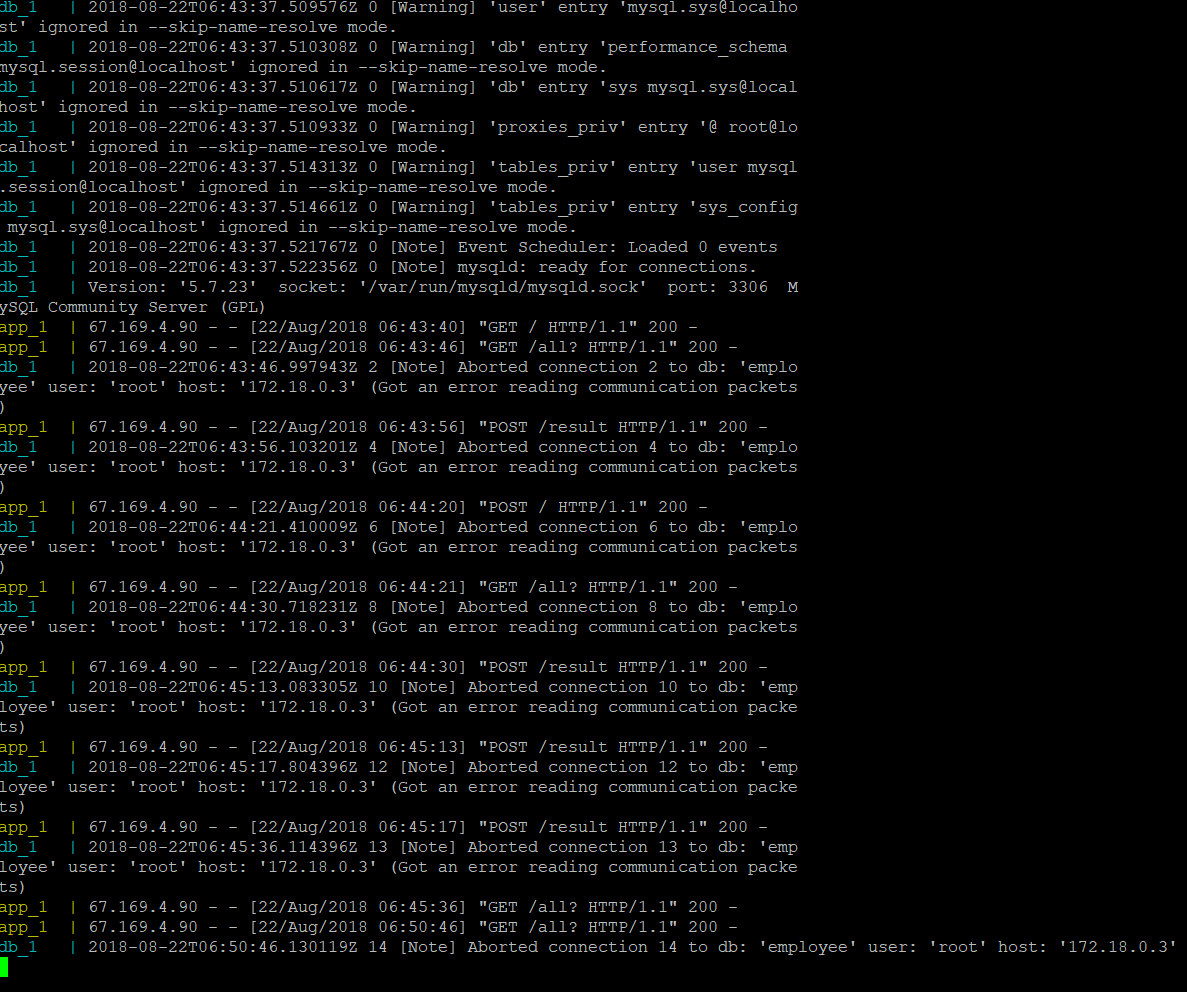


running the application

Building the project through docker-compose up command:

Installing the dependencies

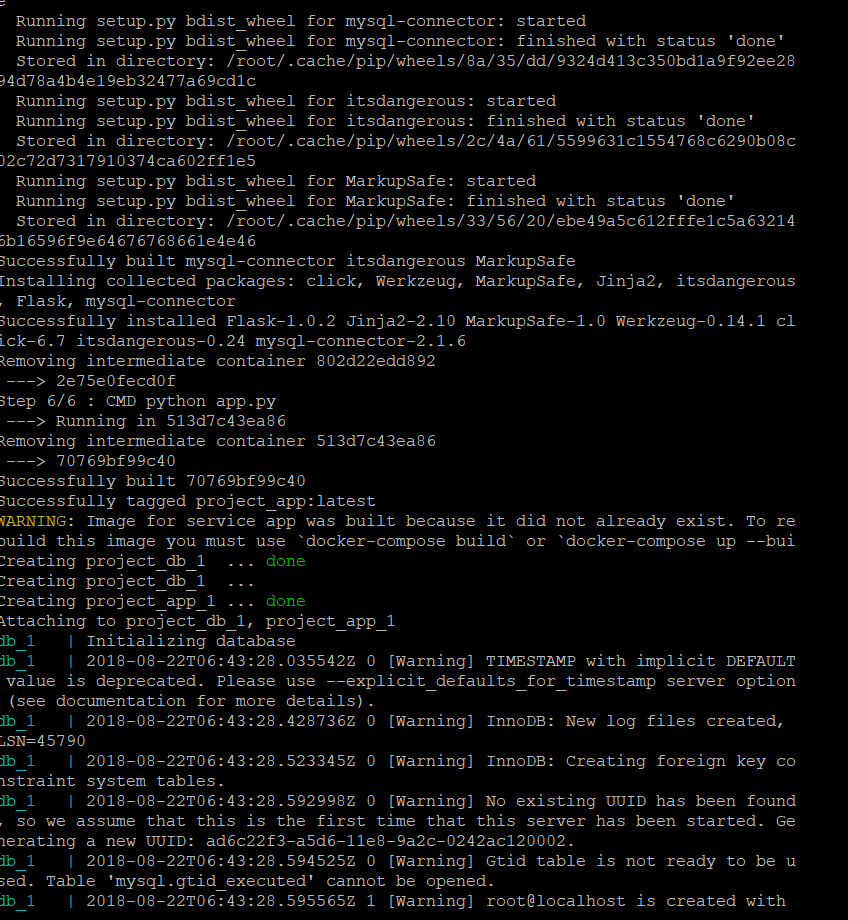




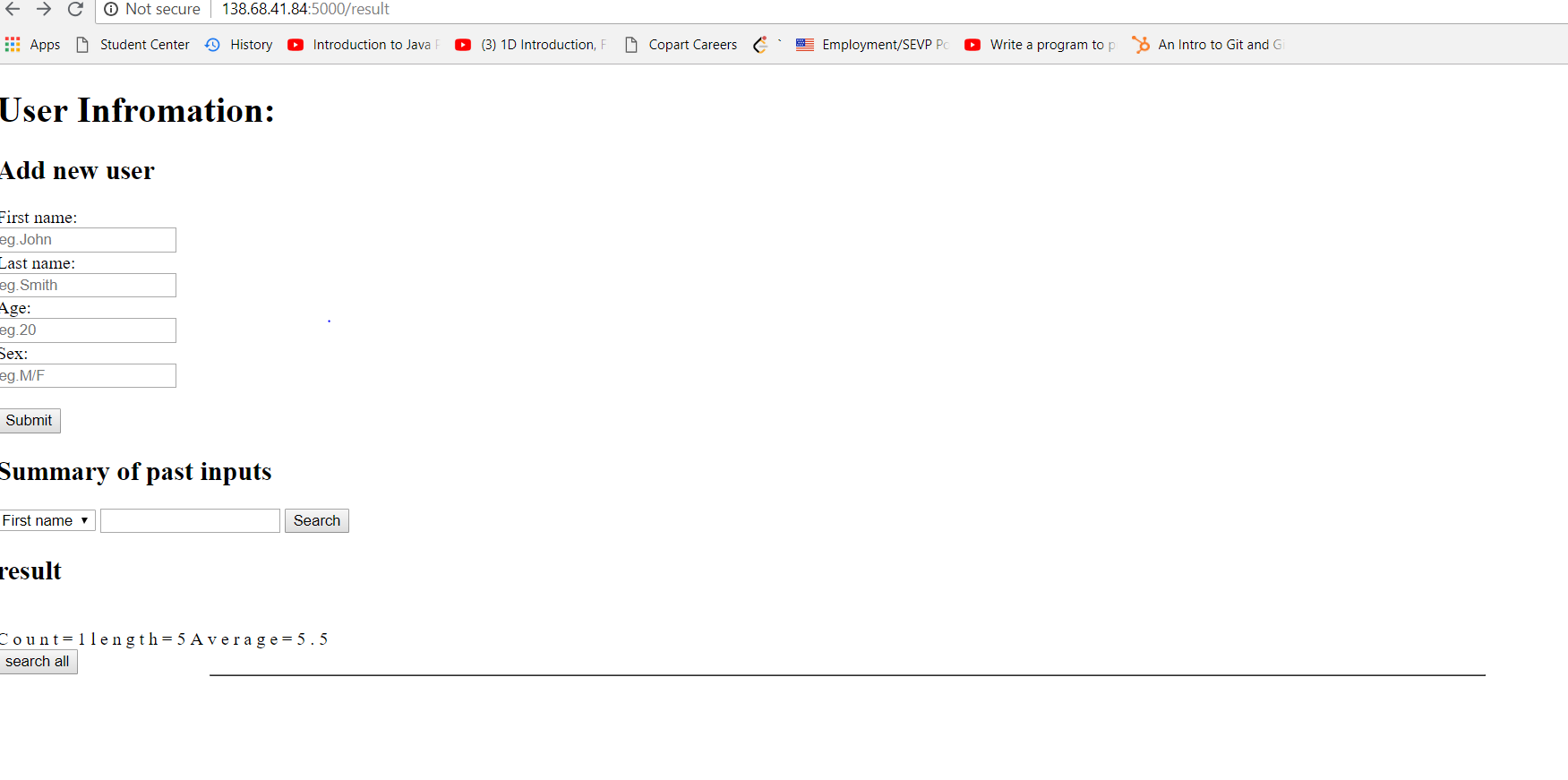
Creating the container :

Db1 :for the database

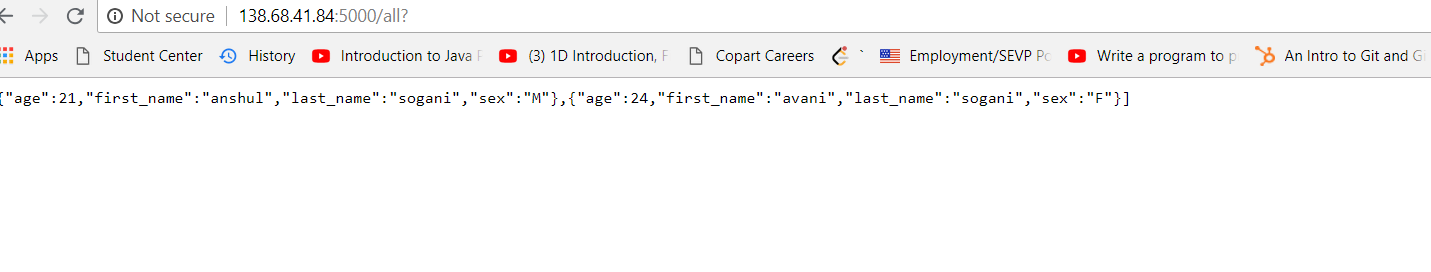
App1:for the application



Running the Web application and MYSQL database in containers and getting the output on local host of the linux machine deployed through digital ocean droplet:



Calling a REST api and it returns the list of all the inputs as a json file:



Kubernetes:

Installing the virtual box on the machine :

apt-get install virtualbox

Installing the mini kube on the machine using snap package manager:

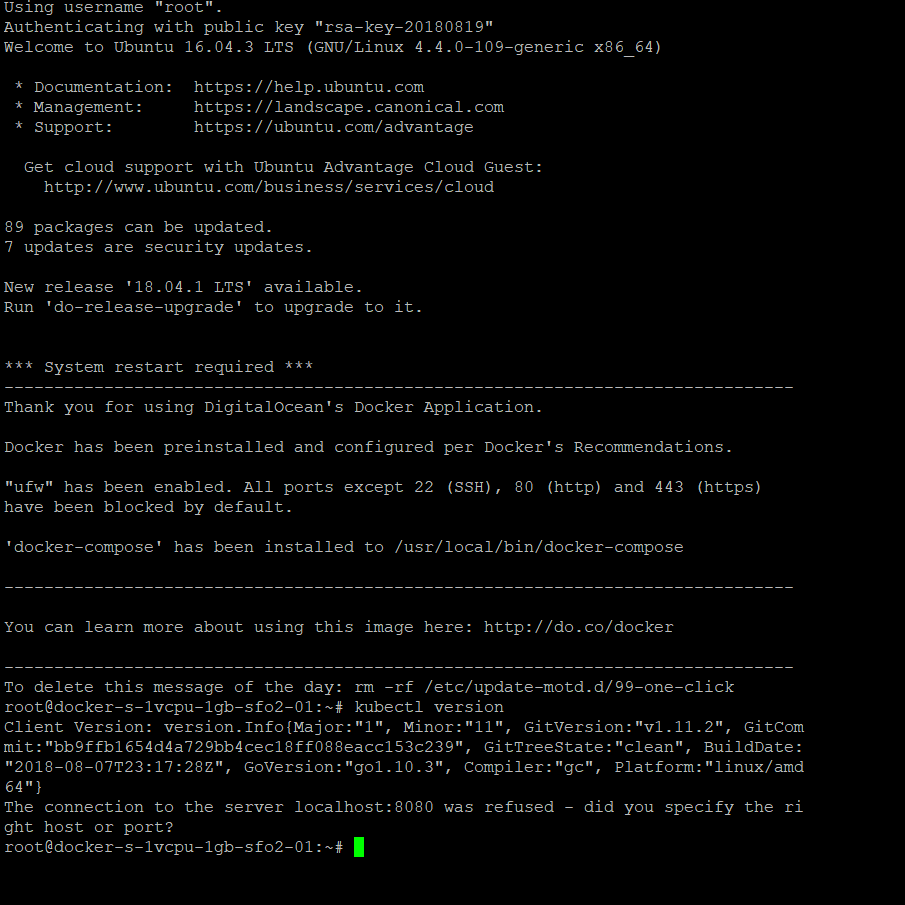
Installing the kubectl:

**snap install kubectl –classic**

installing minikube on linux operating system:

curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.28.2/minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /usr/local/bin/

Testing Kubectl:



Starting the minikube cluster:

Testing minikube running a basic application:

1. minikube start
2. kubectl run hello-minikube --image=gcr.io/google\_containers/echoserver:1.4 --port=8080
3. kubectl expose deployment hello-minikube  --type=NodePort
4. kubectl get pod
5. curl $(minikube service hello-minikube --url)

