**Question 1 :**

Step 1 : created images directory, inside it web directory is created

---> mkdir images

---> cd images

---> mkdir web

---> cd web

Step 2 : Inside web directory created index.html file

---> vi index.html

<h1>Hello Spandana</h1>

Step 3 : Outside web directory Dockerfile and nginx.conf file are created

---> vi Dockerfile

FROM ubuntu

USER root

RUN apt-get update

RUN apt-get install -y nginx nodejs

# Remove the default Nginx configuration file

RUN rm -v /etc/nginx/nginx.conf

# Copy a configuration file from the current directory

ADD nginx.conf /etc/nginx/

ADD web /usr/share/nginx/html/

ADD web /var/www/html/

# Append "daemon off;" to the beginning of the configuration

RUN echo "daemon off;" >> /etc/nginx/nginx.conf

# Expose ports

EXPOSE 80

# Set the default command to execute

# when creating a new container

CMD service nginx start

---> vi nginx.conf

worker\_processes 1;

events { worker\_connections 1024; }

http {

include mime.types;

sendfile on;

server {

root /usr/share/nginx/html/;

index index.html;

server\_name localhost;

listen 80;

}

}

Step 4 : Inside web directory created mimi.types file

---> vi mimi.types

types {

text/html html htm shtml;

}

Step 5 : Went to the directory where Dockerfile is located

Step 6 : Build the image

---> sudo docker build -t my\_nginx .

Step 7 : Run the container with external port 8080 and internal port 80

---> sudo docker run -it -p 8080:80 --name nginx\_container my\_nginx

Step 8 : Getting into container and installing nano

---> sudo docker exec -it 393e1ac7206d /bin/sh

# apt-get install nano

# exit

Step 9 :

---> curl <http://localhost:8080>

There is no change even after installing nano in container

Step 10 :

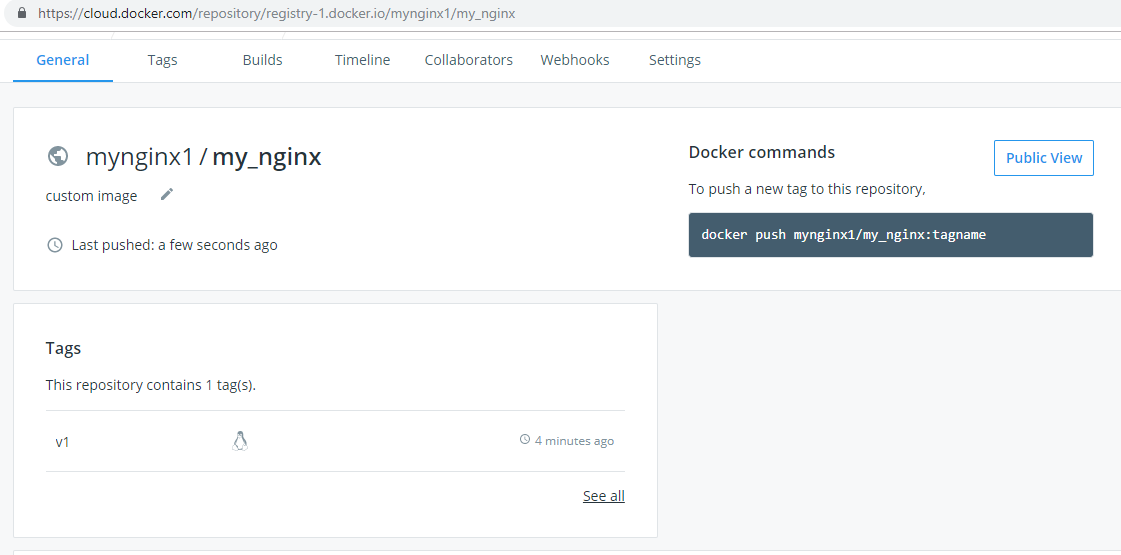
Push custom image into docker hub

create an account in docker hub, also a repo to be created

---> sudo docker login

---> sudo docker tag my\_nginx mynginx1/my\_nginx:v1

---> sudo docker push mynginx1/my\_nginx:v1



Push custom image into Google Cloud Registry

Authenticate to container registry

---> gcloud auth configure-docker

Pushing image

---> sudo docker tag my\_nginx gcr.io/pe-training/my\_nginx

---> sudo docker push gcr.io/pe-training/my\_nginx

Push custom image into Elastic Cloud Registry

Create repository in ECR

Get auth token

---> aws ecr get-authorization-token --region us-east-1 --output text \

--query authorizationData[].authorizationToken | base64 -d | cut -d: -f2

Logging in

---> sudo docker login -u AWS 488599217855.dkr.ecr.us-east-1.amazonaws.com/

Pushing image

---> sudo docker tag my\_nginx 488599217855.dkr.ecr.us-east-1.amazonaws.com/my\_nginx

---> sudo docker push 488599217855.dkr.ecr.us-east-1.amazonaws.com/my\_nginx

**Question 2 :**

Step 1 : Created app.py, Dockerfile, requirements.txt, docker-compose.yaml

app.py

import time

import redis

from flask import Flask

app = Flask(\_\_name\_\_)

cache = redis.Redis(host='redis', port=6379)

def get\_hit\_count():

retries = 5

while True:

try:

return cache.incr('hits')

except redis.exceptions.ConnectionError as exc:

if retries == 0:

raise exc

retries -= 1

time.sleep(0.5)

@app.route('/')

def hello():

count = get\_hit\_count()

return 'Hello World! I have been seen {} times.\n'.format(count)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0", debug=True)

Dockerfile

FROM python:3.7-alpine

ADD . /code

WORKDIR /code

RUN pip install -r requirements.txt

CMD ["python", "app.py"]

requirements.txt

flask

redis

docker-compose.yaml

version: '3'

services:

web:

build: .

ports:

- "5000:5000"

redis:

image: "redis:alpine"

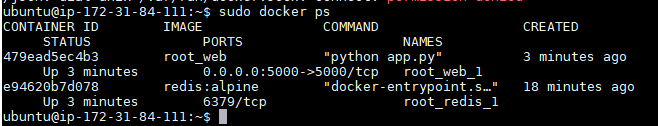
Step 2 :

---> docker-compose build

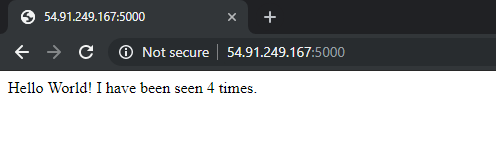
Step 3 :

---> docker-compose up -d

Step 4 : we could see the running containers



Step 5 : Access the ec2 instance on port 5000



**Question 3 :**

Step 1 : docker-composer.yaml is modified

modified docker-composer.yaml

version: '3'

services:

web:

build: .

ports:

- "5000:5000"

volumes:

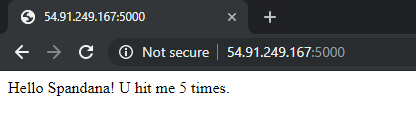
- .:/code

redis:

image: "redis:alpine"

Step 2 : Changed app.py file and this time docker-compose build is not run

---> docker-compose up -d



(without executing ‘docker-composer build’ command, changes in app.py got reflected on webpage )