When we install docker, docker info command will not run and give the following error. So to fix it write the command:

ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/info": dial unix /var/run/docker.sock: connect: permission denied

**Command:** sudo chmod 666 /var/run/docker.sock

If docker-compose not found: Then run following command:

sudo curl -L https://github.com/docker/compose/releases/download/1.21.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

1. **With the help of Docker-compose deploy the ‘Wordpress’ and ‘Mysql’ container and access the front end of ‘Wordpress’**

Docker Compose file to be written for this experiment:

version: '3'

services:

# Database services for wordpress we use mysql

mysql\_db:

container\_name: mysql\_container

image: mysql:8.2

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: it

MYSQL\_DATABASE: wordpress\_db

MYSQL\_USER: Ankur

MYSQL\_PASSWORD: Ankur2003@

volumes:

- mysql:/var/lib/mysql

wordpress:

depends\_on:

- mysql\_db

image: wordpress:latest

restart: always

ports:

- "8080:80"

environment:

WORDPRESS\_DB\_HOST: mysql\_db:3306

WORDPRESS\_DB\_USER: Ankur

WORDPRESS\_DB\_PASSWORD: Ankur2003@

WORDPRESS\_DB\_NAME: wordpress\_db

volumes:

- "./:/var/www/html"

volumes:

mysql: {}

Command: sudo docker-compose up

To Open Website: localhost

1. **Create a simple Hello-world python flask application and create the docker image of that Flask application. Run application on port 5000.**

Make A directory Named: Docker\_Flask.

Make 3 files: app.py, Dockerfile, requirements.txt

app.py:

from flask import Flask

import os

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

@app.route("/")

def hello():

return "Flask inside Docker!!"

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

port = int(os.environ.get("PORT", 5000))

app.run(debug=True,host='0.0.0.0',port=port)

requirements.txt

flask

Dockerfile:

FROM python:3.6

COPY . /app

WORKDIR /app

RUN pip install -r requirements.txt

ENTRYPOINT ["python"]

CMD ["app.py"]

Then Run two commands:

docker build -t simple-flask-app:latest .

docker run -d -p 5000:5000 simple-flask-app

Link: localhost:5000

21.  **Pull the LAMP Stack container from docker hub and host a web application of your own. Push that image back to repository. Make use of database.**

<https://github.com/raptor-2001/php-dockerized-form>

28. **Write a python program to perform arithmetic operations and create Docker image accordingly.**

Create a Directory: Python-App.

A file app.py with content:

print("Hello World")

Change the program with arithmetic operations program.

Dockerfile:

FROM python:3.9

WORKDIR /app

COPY . /app

EXPOSE 80

ENV NAME World

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

Build the image:

docker build -t my-python-app .

docker run -p 4000:80 my-python-app

1. **Create the ‘nginx’ container from ‘nginx’ image. And create the load balancing so that if we go to the address of ‘nginx ‘ it can redirect it to the above created applications (Flask and Wordpress).**

In this experiment, we need to perform load balancing between two applications/image, {wordpress, flask application}.

**Directory Structure:**

Root Directory:

flask\_app

app.py

Dockerfile

requirements.txt

docker-compose.yaml

nginx.conf

**Command to make application work:** docker-compose up

localhost:5000 -> Flask Application

localhost:8080 -> Wordpress Application

localhost -> Any application depending upon balancer.

**Create Flask Application similar to that of experiment 24.**

**Content of Dockerfile:**

events {}

http {

upstream backend {

server flask:5000;

server wordpress:80;

}

server {

listen 80;

location / {

proxy\_pass http://backend;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

}

**Content of docker-compose.yaml:**

version: '3'

services:

nginx:

image: nginx:latest

ports:

- "80:80"

volumes:

- ./nginx.conf:/etc/nginx/nginx.conf

depends\_on:

- flask

- wordpress

flask:

build:

context: ./flask\_app

ports:

- "5000:5000"

mysql\_db:

container\_name: mysql\_container

image: mysql:8.2

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: it

MYSQL\_DATABASE: wordpress\_db

MYSQL\_USER: Ankur

MYSQL\_PASSWORD: Ankur2003@

volumes:

- mysql:/var/lib/mysql

wordpress:

depends\_on:

- mysql\_db

image: wordpress:latest

restart: always

ports:

- "8080:80"

environment:

WORDPRESS\_DB\_HOST: mysql\_db:3306

WORDPRESS\_DB\_USER: Ankur

WORDPRESS\_DB\_PASSWORD: Ankur2003@

WORDPRESS\_DB\_NAME: wordpress\_db

volumes:

- "./:/var/www/html"

volumes:

mysql: {}

**Command to run this experiment:**

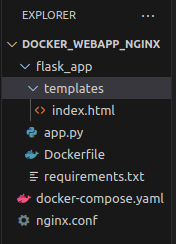
docker-compose up -d

Write localhost to access site.

**31. Create a web application with simple web page containing login details and create a docker image of the application.(Use Ngnix Web server)**

We need to build a flask application with login details and using nginx web server, we need to create image.

**Directory Structure:**

****

**Content of flask application similar to that of experiment 24. Some changes in app.py and templates folder is added.**

**app.py:**

from flask import Flask, render\_template

import os

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

@app.route('/')

def index():

return render\_template('index.html')

if \_\_name\_\_ == '\_\_main\_\_':

port = int(os.environ.get("PORT", 5000))

app.run(debug=True,host='0.0.0.0',port=port)

**Index.html:**

<!-- templates/index.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login Page</title>

</head>

<body>

<h2>Login</h2>

<form action="/login" method="post">

<label for="username">Username:</label>

<input type="text" id="username" name="username" required>

<br>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

<br>

<input type="submit" value="Login">

</form>

</body>

</html>

**nginx.conf file:**

events {}

http {

upstream backend {

server flask:5000;

}

server {

listen 80;

location / {

proxy\_pass http://backend;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

}

**docker-compose.yaml:**

version: '3'

services:

nginx:

image: nginx:latest

ports:

- "80:80"

volumes:

- ./nginx.conf:/etc/nginx/nginx.conf

depends\_on:

- flask

flask:

build:

context: ./flask\_app

ports:

- "5000:5000"