



LS Lab 1: Containerization and Application Layer Load Balancing

Name: Emeka Michael Nzeopara

► Details

To install the docker engine in my ubuntu machine we will run the following command. We need to uninstall the previous old version of the docker engine, we can achieve this using the following command

```
* sudo apt-get remove docker docker-engine docker.io containerd runc
```

We will set up the docker engine by first, update the apt package index and install the packages to allow apt to use a repository over https: This can be done with the following command.

```
* sudo apt-get update
* sudo apt-get install ca-certificates curl gnupg lsb-release
```

Next is to add docker's official gpg key using the following command

```
* sudo mkdir -p /etc/apt/keyrings
* curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/ap
```



Then we will set up the repository with the following command

```
* echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https:/
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```



We have successfully set-up the apt repository, we will update the package index using the command

```
* sudo apt-get update
```

Should incase there is an error; run the command

```
* sudo chmod a+r /etc/apt/keyrings/docker.gpg
* sudo apt-get update
```

To install the docker engine, containerd and docker Compose

```
* sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

After this has been completed, I obtained an error showing the docker service is not running properly

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: activating (auto-restart) (Result: exit-code) since Sat 2023-01-28 12:59:31 MSK; 16ms ago
 TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Process: 671733 ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock (code=exited, status=1/FAILURE)
   Main PID: 671733 (code=exited, status=1/FAILURE)
      CPU: 45ms

янв 28 12:59:31 st17-HP-ElliteDesk-800-G1-SFF systemd[1]: docker.service: Failed with result 'exit-code'.
янв 28 12:59:31 st17-HP-ElliteDesk-800-G1-SFF systemd[1]: Failed to start Docker Application Container Engine.
янв 28 12:59:33 st17-HP-ElliteDesk-800-G1-SFF systemd[1]: docker.service: Scheduled restart job, restart counter is at 1.
янв 28 12:59:33 st17-HP-ElliteDesk-800-G1-SFF systemd[1]: Stopped Docker Application Container Engine.
янв 28 12:59:33 st17-HP-ElliteDesk-800-G1-SFF systemd[1]: Starting Docker Application Container Engine...
янв 28 12:59:33 st17-HP-ElliteDesk-800-G1-SFF dockerd[671744]: time="2023-01-28T12:59:33.766037754+03:00" level=info msg="Starting up"
янв 28 12:59:33 st17-HP-ElliteDesk-800-G1-SFF dockerd[671744]: failed to load listeners: no sockets found via socket activation: make sure the service was started by systemd
янв 28 12:59:33 st17-HP-ElliteDesk-800-G1-SFF systemd[1]: docker.service: Main process exited, code=exited, status=1/FAILURE
янв 28 12:59:33 st17-HP-ElliteDesk-800-G1-SFF systemd[1]: docker.service: Failed with result 'exit-code'.
янв 28 12:59:33 st17-HP-ElliteDesk-800-G1-SFF systemd[1]: Failed to start Docker Application Container Engine.
dpkg: error processing package docker-ce (--configure):
 installed docker-ce package post-installation script subprocess returned error exit status 1
Processing triggers for man-db (2.10.2-1) ...
Errors were encountered while processing:
 docker-ce
E: Sub-process /usr/bin/dpkg returned an error code (1)
lneka@st17-HP-ElliteDesk-800-G1-SFF: $ systemctl status docker
docker.service - Docker Application Container Engine
```

Figure 1: Error after the Installation

The way i went about resolving this problem is by running an upgrade command on the apt, and restarting the docker

- * sudo apt upgrade
- * systemctl restart docker

```
Emeka@st17-HP-EliteDesk-800-G1-SFF:~$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2023-01-28 13:02:17 MSK; 7s ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Main PID: 672133 (dockerd)
      Tasks: 9
     Memory: 21.7M
        CPU: 230ms
    CGroup: /system.slice/docker.service
            └─672133 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jan 28 13:02:15 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:15.669437748+03:00" level=info msg="ccResolverWrapper: sending update to cc: [{unix:///run/containerd/containerd.sock}]"
Jan 28 13:02:15 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:15.669454863+03:00" level=info msg="clientConn switching balancer to \"pick_first\" module=grpc"
Jan 28 13:02:16 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:16.651317586+03:00" level=info msg="[graphdriver] using prior storage driver: overlay2"
Jan 28 13:02:16 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:16.802418432+03:00" level=info msg="Loading containers: start."
Jan 28 13:02:17 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:17.188580720+03:00" level=info msg="Default bridge (docker0) is assigned with an IP address 172.17.0.0/16. Daemon opti"
Jan 28 13:02:17 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:17.313697587+03:00" level=info msg="Loading containers: done."
Jan 28 13:02:17 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:17.533665418+03:00" level=info msg="Docker daemon" commit=6051f14 graphdriver(s)=overlay2 version=20.10.23
Jan 28 13:02:17 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:17.533723944+03:00" level=info msg="Daemon has completed initialization"
Jan 28 13:02:17 st17-HP-EliteDesk-800-G1-SFF dockerd[672133]: time="2023-01-28T13:02:17.678626511+03:00" level=info msg="API listen on /run/docker.sock"
```

Figure 2: After correcting the Error

Task 1: Get familiar with Docker Engine

1. Pull Nginx v1.23.3 image from dockerhub registry and confirm it is listed in local images.

This can be done by running the following command

- * docker pull nginx:1.23.3

```
Emeka@st17-HP-EliteDesk-800-G1-SFF:~$ docker pull nginx:1.23.3
1.23.3: Pulling from library/nginx
8740c948ffd4: Pull complete
d2c0556a17c5: Pull complete
c8b9881f2c6a: Pull complete
693c3ffa8f43: Pull complete
8316c5e80e6d: Pull complete
b2fe3577faa4: Pull complete
Digest: sha256:b8f2383a95879e1ae064940d9a200f67a6c79e710ed82ac42263397367e7cc4e
Status: Downloaded newer image for nginx:1.23.3
docker.io/library/nginx:1.23.3
Emeka@st17-HP-EliteDesk-800-G1-SFF:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                1.23.3             a99a39d070bf       2 weeks ago        142MB
gns3/openvswitch    latest             bd06ab09792b       6 months ago       17.6MB
hello-world         latest             feb5d9fea6a5       16 months ago      13.3kB
Emeka@st17-HP-EliteDesk-800-G1-SFF:~$
```

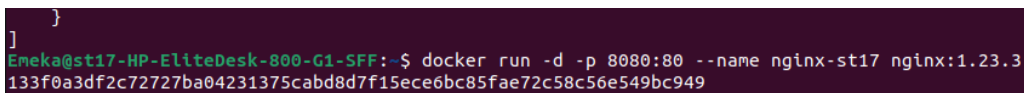
Figure 3: Pulling the Nginx Image

where the **docker pull** command will pull the specified nginx image and the **docker images** command will list the images in your local machine as shown in the picture above.

2. Run the pulled Nginx as a container with the below properties
 - a) Map the port to 8080.
 - b) Name the container as nginx-<stX>.
 - c) Run it as daemon
 - d) Access the page from your browser

To perform this task we will run the command as given thus, as given below

```
* docker run -d -p 8080:80 --name nginx-st17 nginx:1.23.3
```



```
}  
] Emeka@st17-HP-EliteDesk-800-G1-SFF:~$ docker run -d -p 8080:80 --name nginx-st17 nginx:1.23.3  
133f0a3df2c72727ba04231375cabd8d7f15ece6bc85fae72c58c56e549bc949
```

Figure 4: Nginx Command with the above properties

From the command given above they can be explained thus as:

'-d' flag is used to run the daemon in the background

'-p' is used to map port 8080 on the host to the port 80 on the container

'--name' this is used to name the image that has been created

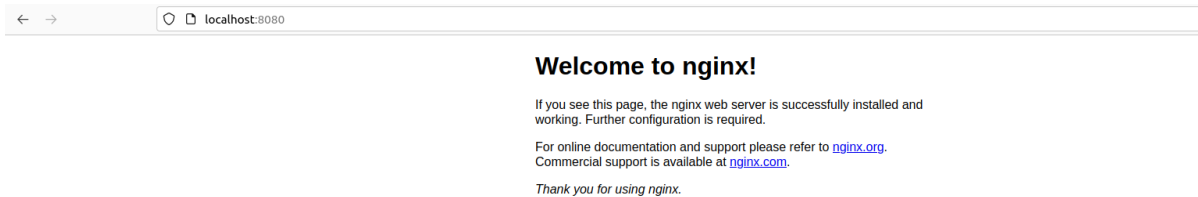


Figure 5: Accessing the Nginx page from my browser

3. Confirm port mapping.
 - a) List open ports in host machine
 - b) List open ports inside the running container

To run this command that will list these properties, we will run this command

```
* docker container inspect nginx-st17
```

```

meka@st17-HP-EliteDesk-800-G1-SFF:~$ docker container inspect nginx-st17
[
  {
    "Id": "133f0a3df2c72727ba04231375cabd8d7f15e6bc85fae72c58c56e549bc949",
    "Created": "2023-01-28T12:37:13.243331794Z",
    "Path": "/docker-entrypoint.sh",
    "Args": [
      "nginx",
      "-g",
      "daemon off;"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 680433,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-01-28T12:37:17.261318093Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:a99a39d070bdf1cb0fe65c45dea3a33764dc00a9546bf8dc46cb5a11b1b50e9",
    "ResolvConfPath": "/var/lib/docker/containers/133f0a3df2c72727ba04231375cabd8d7f15e6bc85fae72c58c56e549bc949/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/133f0a3df2c72727ba04231375cabd8d7f15e6bc85fae72c58c56e549bc949/hostname",
    "HostsPath": "/var/lib/docker/containers/133f0a3df2c72727ba04231375cabd8d7f15e6bc85fae72c58c56e549bc949/hosts",
    "LogPath": "/var/lib/docker/containers/133f0a3df2c72727ba04231375cabd8d7f15e6bc85fae72c58c56e549bc949/133f0a3df2c72727ba04231375cabd8d7f15e6bc85fae72c58c56e549bc949-json.log",
    "Name": "/nginx-st17",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "docker-default",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,
      "ContainerIDFile": "",
      "LogConfig": {
        "Type": "json-file",
        "Config": {}
      },
      "NetworkMode": "default",
      "PortBindings": {
        "80/tcp": [
          {
            "HostIp": "",
            "HostPort": "8080"
          }
        ]
      }
    }
  }
]

```

Figure 6: Inspect The Properties of the Nginx Container

The above command will output the properties of the container of Nginx, from the highlight part in the screenshot we can see that the parameters highlights both the container ports i.e. **80** to that of the host port i.e. **8080**

```

meka@st17-HP-EliteDesk-800-G1-SFF:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED     STATUS      PORTS                               NAMES
133f0a3df2c7   nginx:1.23.3   "/docker-entrypoint..."  56 minutes ago    Up 56 minutes    0.0.0.0:8080->80/tcp, :::8080->80/tcp   nginx-st17
35ab1b8dad6e   hello-world   "/hello"                  4 hours ago      Exited (0) 4 hours ago                               pedantic_mclean

```

Figure 7: Alternative way to find out by running the ps command

- * 'netstat -lntp' or 'ss -lntp' (this will show the opened port in the host)
- * 'docker exec -it nginx-st17 netstat -lntp' or 'docker exec -it nginx-st17 ss -lntp'

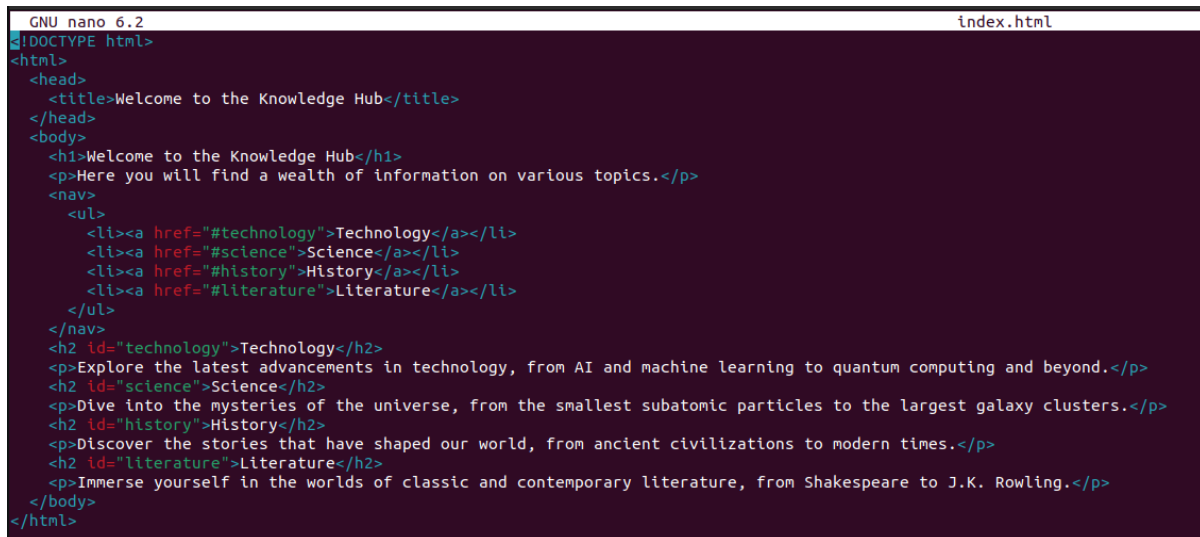


The command above is sort of a repetition of a shell command, the first one is the command to run when you are in a shell to obtain the open ports available whereas the second row shows that we will enter the shell of the docker before the command to obtain the ports.

4. Create Dockerfile similar with below properties (let's call it container A).
 - a. Image tag should be Nginx v1.23.3.
 - b. Create a custom index.html file and copy it to your docker image to replace Nginx default web page.
 - c. Build the image from the Dockerfile, tag it during build as nginx:<stX>, check/validate local images, and run your custom made docker image.
 - d. Access via browser and validate your custom page is hosted

First I created a index.html file using the following command in a particular directory

- * `sudo mkdir docker && cd docker`
- * `sudo nano index.html`

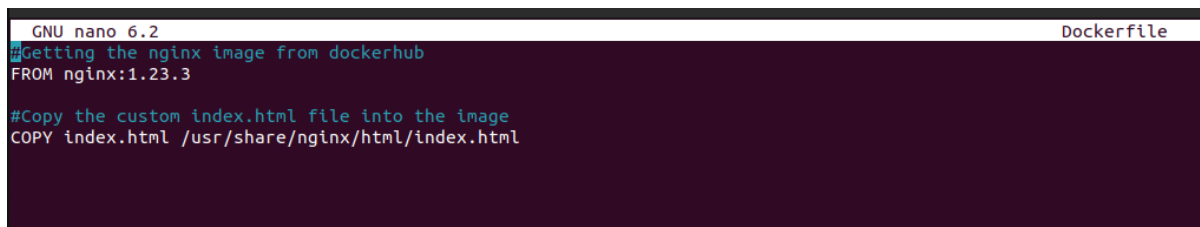


```
GNU nano 6.2 index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Welcome to the Knowledge Hub</title>
  </head>
  <body>
    <h1>Welcome to the Knowledge Hub</h1>
    <p>Here you will find a wealth of information on various topics.</p>
    <nav>
      <ul>
        <li><a href="#technology">Technology</a></li>
        <li><a href="#science">Science</a></li>
        <li><a href="#history">History</a></li>
        <li><a href="#literature">Literature</a></li>
      </ul>
    </nav>
    <h2 id="technology">Technology</h2>
    <p>Explore the latest advancements in technology, from AI and machine learning to quantum computing and beyond.</p>
    <h2 id="science">Science</h2>
    <p>Dive into the mysteries of the universe, from the smallest subatomic particles to the largest galaxy clusters.</p>
    <h2 id="history">History</h2>
    <p>Discover the stories that have shaped our world, from ancient civilizations to modern times.</p>
    <h2 id="literature">Literature</h2>
    <p>Immerse yourself in the worlds of classic and contemporary literature, from Shakespeare to J.K. Rowling.</p>
  </body>
</html>
```

Figure 8: index.html file created

To create the Dockerfile using the above properties I created a Dockerfile as given thus

- * `sudo nano Dockerfile`



```
GNU nano 6.2 Dockerfile
#Getting the nginx image from dockerhub
FROM nginx:1.23.3

#Copy the custom index.html file into the image
COPY index.html /usr/share/nginx/html/index.html
```

Figure 9: Dockerfile with the properties

From the Dockerfile created above the **FROM** command is used to pull the image nginx file from the dockerhub repository and in this case it is the *Nginxv1.23.3*

The **COPY** command is used to copy the custom *index.html* to that of the Nginx docker image

The next thing we would do is to build the image from the Dockerfile using the following command, running it from the directory that contains the Dockerfile.

We will also validate the command with the following command

- * `docker build -t nginx:stX .`
- * `docker images`

```

Emeka@st17-HP-EliteDesk-800-G1-SFF:~/docker$ docker build -t nginx:st17 .
Sending build context to Docker daemon 4.096kB
Step 1/2 : FROM nginx:1.23.3
--> a99a39d070bf
Step 2/2 : COPY index.html /usr/share/nginx/html/index.html
--> daf7f2207c67
Successfully built daf7f2207c67
Successfully tagged nginx:st17
Emeka@st17-HP-EliteDesk-800-G1-SFF:~/docker$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                st17               daf7f2207c67       2 minutes ago      142MB
nginx                1.23.3            a99a39d070bf       2 weeks ago        142MB
gns3/openvswitch    latest            bd06ab09792b       6 months ago       17.6MB
hello-world         latest            feb5d9fea6a5       16 months ago      13.3kB
Emeka@st17-HP-EliteDesk-800-G1-SFF:~/docker$

```

Figure 10: Building the image from the Dockerfile for container A

To run the custom image that has been created, we will run the following command

```
* docker run -p 8081:80 --name nginx-custom -d nginx:st17
```

```

Emeka@st17-HP-EliteDesk-800-G1-SFF:~/docker$ docker run -p 8081:80 --name nginx-custom -d nginx:st17
1ef35ff7a1d87ba410183492c7ae187a1e97134e567651929a2f93cccec2c91f
Emeka@st17-HP-EliteDesk-800-G1-SFF:~/docker$

```

Figure 11: Running the custom image

To access the custom image from the browser

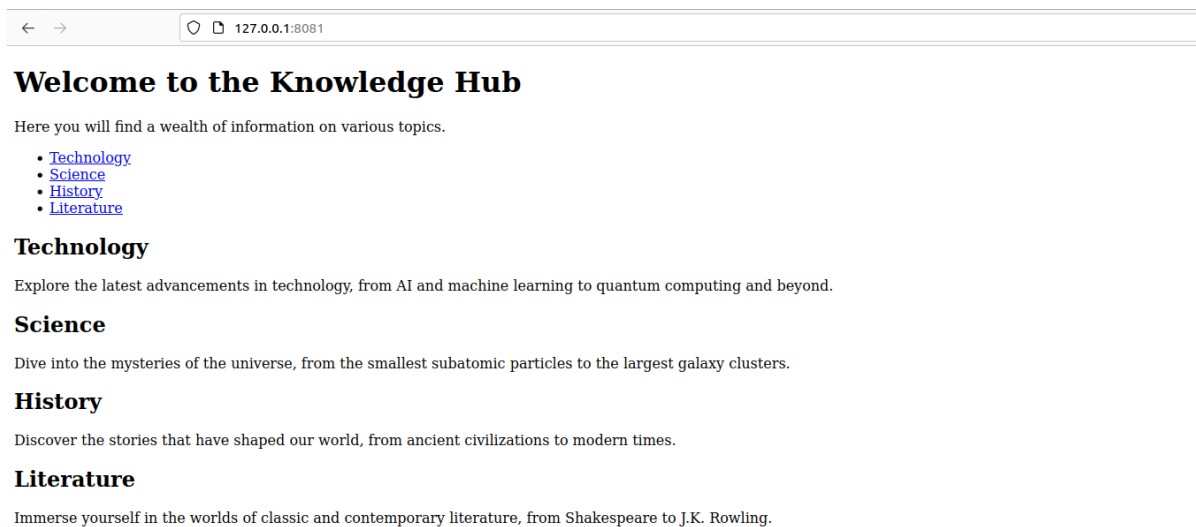
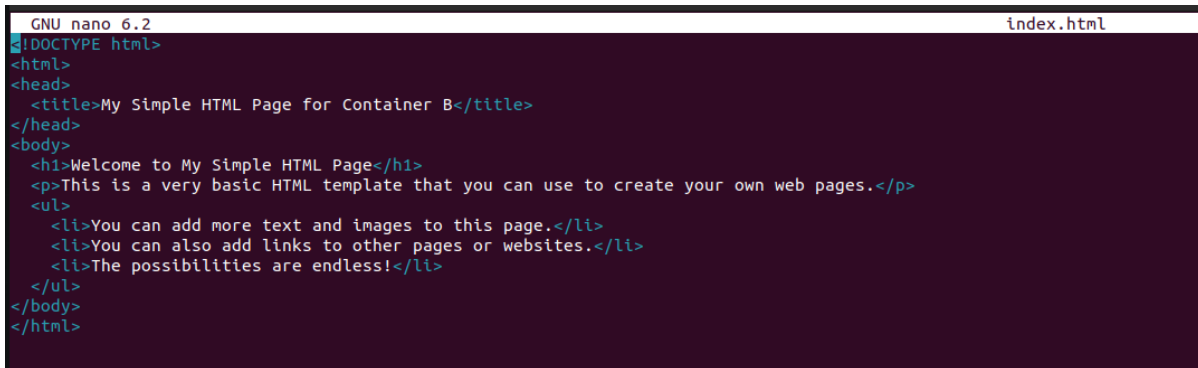


Figure 12: Accessing the custom image from Browser

Task 2: Work With Multi-Container Environment

1. Create another Dockerfile similar to step 1.4 (Let's call it container B), and an index.html with different content.

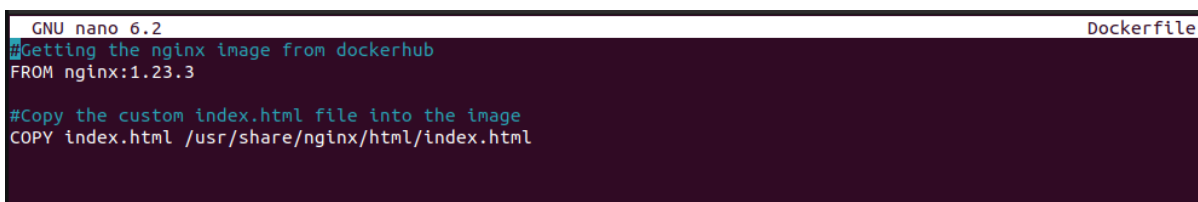
Following the same process in the previous task, the html command found is given as

A screenshot of a terminal window showing the GNU nano 6.2 text editor. The file being edited is index.html. The content of the file is a basic HTML template with a title, a welcome message, and a list of instructions for adding content to the page.

```
GNU nano 6.2 index.html
<!DOCTYPE html>
<html>
<head>
  <title>My Simple HTML Page for Container B</title>
</head>
<body>
  <h1>Welcome to My Simple HTML Page</h1>
  <p>This is a very basic HTML template that you can use to create your own web pages.</p>
  <ul>
    <li>You can add more text and images to this page.</li>
    <li>You can also add links to other pages or websites.</li>
    <li>The possibilities are endless!</li>
  </ul>
</body>
</html>
```

Figure 12: Index of html page for container B

The Dockerfile for container B is similar to that of container A, it is shown below as

A screenshot of a terminal window showing the GNU nano 6.2 text editor. The file being edited is Dockerfile. The content of the file includes instructions to get the nginx image from Docker Hub and to copy a custom index.html file into the image.

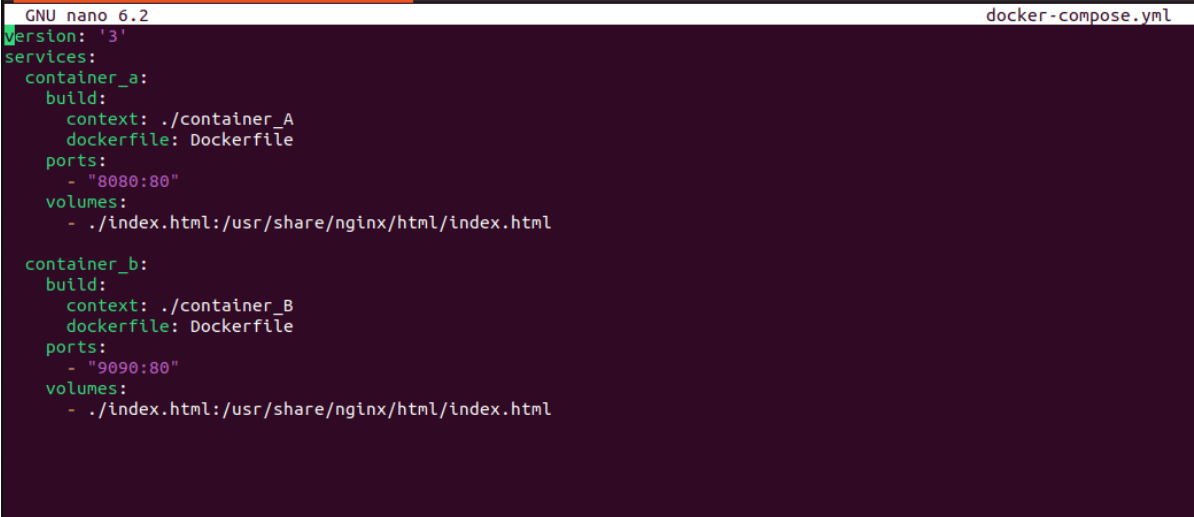
```
GNU nano 6.2 Dockerfile
#Getting the nginx image from dockerhub
FROM nginx:1.23.3

#Copy the custom index.html file into the image
COPY index.html /usr/share/nginx/html/index.html
```

Figure 13: Dockerfile for container B

2. Write a docker-compose file with below properties
 - a) Multi-build: Builds both Dockerfiles and run both images.
 - b) Port mapping: Container A should listen to port 8080 and container B should listen to port 9090. (They host two different web pages)
 - c) Volumes: Mount (bind) a directory from the host file system to Nginx containers to replace the default Nginx web page with the two index.html files created in Steps 1.4.b and 2.1 .

The docker compose file for the properties given above is thus:



```
GNU nano 6.2 docker-compose.yml
version: '3'
services:
  container_a:
    build:
      context: ./container_A
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    volumes:
      - ./index.html:/usr/share/nginx/html/index.html
  container_b:
    build:
      context: ./container_B
      dockerfile: Dockerfile
    ports:
      - "9090:80"
    volumes:
      - ./index.html:/usr/share/nginx/html/index.html
```

Figure 14: docker-compose file

The above file states the docker-compose file with the properties stated in this task.

The first 'container' is used to describe the services; the 'build' indicates the building section of the docker, the 'context' indicates the directory to find the Dockerfile used to build that docker, and the 'ports and volume' indicates the two ports to access each of the service from the web browser where the latter indicates the directory from the host file to replace the Nginx containers default page.

3. Run the docker compose file and validate you have access to both Nginx web pages in your browser via their respective ports.

Firstly, I noticed that there is no docker-compose package to run the command in my ubuntu machine. So I need to install the docker-compose using the command:

```
* sudo apt install docker-compose
```

```

Emekagst17-HP-EliteDesk-800-G1-SFF:~/docker/container_a$ sudo apt install docker-compose
[sudo] password for Emeka:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-docker python3-dockerpty python3-doccopt python3-dotenv python3-jjsonschema python3-pyrsistent python3-texttable python3-websocket
Suggested packages:
  python3-jjsonschema-doc
Recommended packages:
  docker.io
The following NEW packages will be installed:
  docker-compose python3-docker python3-dockerpty python3-doccopt python3-dotenv python3-jjsonschema python3-pyrsistent python3-texttable python3-websocket
0 upgraded, 9 newly installed, 0 to remove and 11 not upgraded.
Need to get 388 kB of archives.
After this operation, 2 065 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ru.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-websocket all 1.2.3-1 [34,7 kB]
Get:2 http://ru.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-docker all 5.0.3-1 [89,3 kB]
Get:3 http://ru.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-dockerpty all 0.4.1-2 [11,1 kB]
Get:4 http://ru.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-doccopt all 0.6.2-4 [26,9 kB]
Get:5 http://ru.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-dotenv all 0.19.2-1 [20,5 kB]
Get:6 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 python3-pyrsistent amd64 0.18.1-1build1 [55,5 kB]
Get:7 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 python3-jjsonschema all 3.2.0-0ubuntu2 [43,1 kB]
Get:8 http://ru.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-texttable all 1.6.4-1 [11,4 kB]
Get:9 http://ru.archive.ubuntu.com/ubuntu jammy/universe amd64 docker-compose all 1.29.2-1 [95,8 kB]
Fetched 388 kB in 0s (910 kB/s)
Selecting previously unselected package python3-websocket.
(Reading database ... 257323 files and directories currently installed.)
Preparing to unpack .../0-python3-websocket_1.2.3-1_all.deb ...
Unpacking python3-websocket (1.2.3-1) ...
Selecting previously unselected package python3-docker.
Preparing to unpack .../1-python3-docker_5.0.3-1_all.deb ...
Unpacking python3-docker (5.0.3-1) ...
Selecting previously unselected package python3-dockerpty.

```

Figure 15: Install the docker-compose package

Since we have docker-compose install in the machine now, we need to build the containers using docker-compose and the command goes thus:

* docker-compose up

```

Emeka@st17-HP-EliteDesk-800-G1-SFF:~/Docker$ docker-compose up
Creating network "docker_default" with the default driver
Creating docker_container_b_1 ... done
Creating docker_container_a_1 ... done
Attaching to docker_container_a_1, docker_container_b_1
container_a_1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
container_a_1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
container_a_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
container_b_1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
container_b_1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
container_a_1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
container_a_1 | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
container_b_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
container_a_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
container_a_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
container_b_1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
container_a_1 | /docker-entrypoint.sh: Configuration complete; ready for start up
container_b_1 | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: using the "epoll" event method
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: nginx/1.23.3
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: OS: Linux 5.15.0-58-generic
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: start worker processes
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: start worker process 29
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: start worker process 30
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: start worker process 31
container_b_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
container_a_1 | 2023/01/28 16:16:10 [notice] 1#1: start worker process 32
container_b_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
container_b_1 | /docker-entrypoint.sh: Configuration complete; ready for start up
container_b_1 | 2023/01/28 16:16:10 [notice] 1#1: using the "epoll" event method
container_b_1 | 2023/01/28 16:16:10 [notice] 1#1: nginx/1.23.3
container_b_1 | 2023/01/28 16:16:10 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)

```

Figure 16: Running Docker Compose

The web page for both service can be accessed from the web browsers as shown below

For the Container_A:

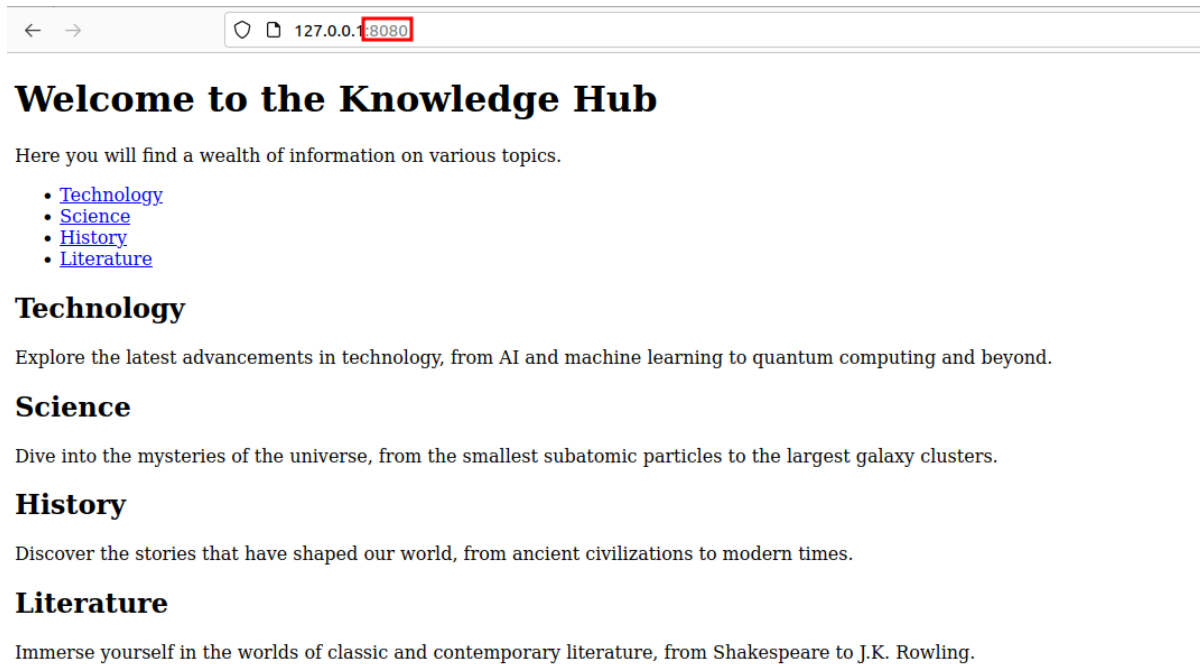


Figure 17: Webpage of Container A

For the Container B:

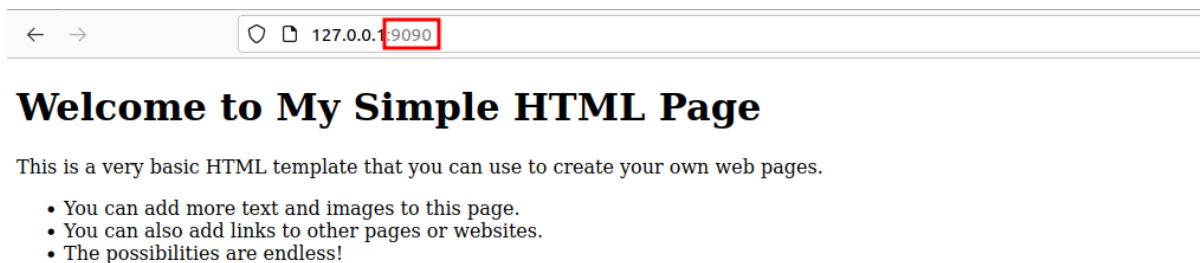


Figure 18: Webpage of Container B

4. Configure L7 Loadbalancer

- a) Install Nginx in the host machine, and configure it in front of two containers in a manner that it should distribute the load in RR approach

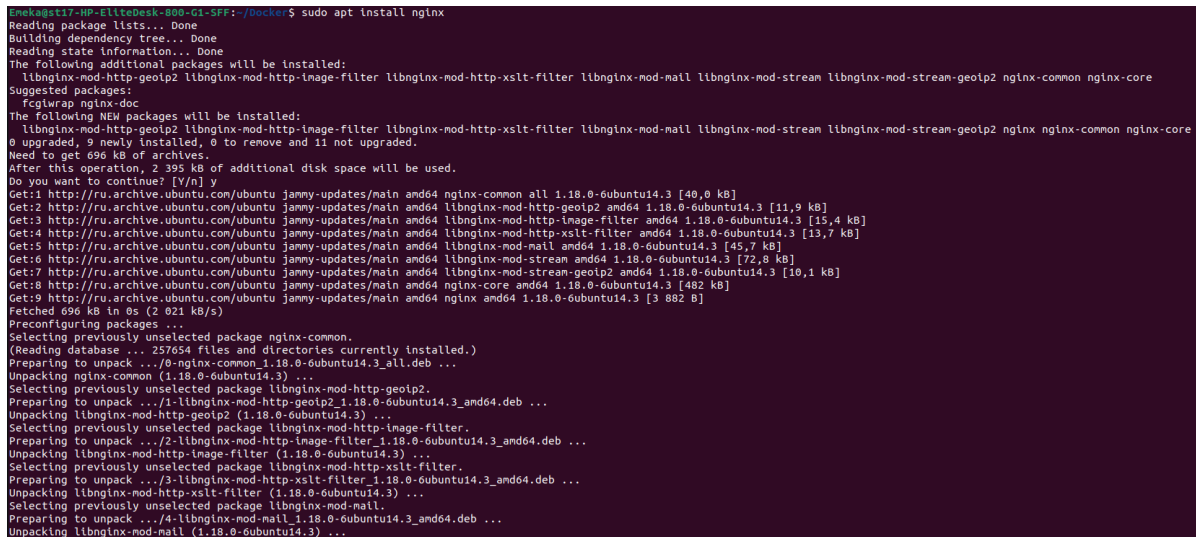
An L7 loadbalancer means a layer 7 load balancer i.e application layer of the OSI model. What this means is that the loadbalancer directs traffic based on the contents of the application layer protocol, such as HTTP, HTTPS or FTP. This is quite impressive because it can lead to better traffic management and routing decision.

Configuring the server in front means that we should configure Nginx server as a proxy server, and place the two running containers at the back of it.

The RR approach means that the Load balancing technique should be round robin technique i.e the request are evenly distributed between the servers.

To install the Nginx Server, we will run the command:

```
* sudo apt install nginx
```



```

mnekagst17-MP-EliteDesk-800-G1-SFF:~/docker$ sudo apt install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip2 nginx-common nginx-core
Suggested packages:
  fcgiwrap nginx-doc
The following NEW packages will be installed:
  libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip2 nginx nginx-common nginx-core
0 upgraded, 9 newly installed, 0 to remove and 11 not upgraded.
Need to get 696 kB of archives.
After this operation, 2 395 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 nginx-common all 1.18.0-6ubuntu14.3 [40.0 kB]
Get:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnginx-mod-http-geoip2 amd64 1.18.0-6ubuntu14.3 [11.9 kB]
Get:3 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnginx-mod-http-image-filter amd64 1.18.0-6ubuntu14.3 [15.4 kB]
Get:4 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnginx-mod-http-xslt-filter amd64 1.18.0-6ubuntu14.3 [13.7 kB]
Get:5 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnginx-mod-mail amd64 1.18.0-6ubuntu14.3 [45.7 kB]
Get:6 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnginx-mod-stream amd64 1.18.0-6ubuntu14.3 [72.8 kB]
Get:7 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnginx-mod-stream-geoip2 amd64 1.18.0-6ubuntu14.3 [10.1 kB]
Get:8 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 nginx-core amd64 1.18.0-6ubuntu14.3 [482 kB]
Get:9 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 nginx amd64 1.18.0-6ubuntu14.3 [3 882 B]
Fetched 696 kB in 0s (2 021 kB/s)
Preconfiguring packages ...
Selecting previously unselected package nginx-common.
(Reading database ... 257654 files and directories currently installed.)
Preparing to unpack .../0-nginx-common_1.18.0-6ubuntu14.3_all.deb ...
Unpacking nginx-common (1.18.0-6ubuntu14.3) ...
Selecting previously unselected package libnginx-mod-http-geoip2.
Preparing to unpack .../1-libnginx-mod-http-geoip2_1.18.0-6ubuntu14.3_and64.deb ...
Unpacking libnginx-mod-http-geoip2 (1.18.0-6ubuntu14.3) ...
Selecting previously unselected package libnginx-mod-http-image-filter.
Preparing to unpack .../2-libnginx-mod-http-image-filter_1.18.0-6ubuntu14.3_and64.deb ...
Unpacking libnginx-mod-http-image-filter (1.18.0-6ubuntu14.3) ...
Selecting previously unselected package libnginx-mod-http-xslt-filter.
Preparing to unpack .../3-libnginx-mod-http-xslt-filter_1.18.0-6ubuntu14.3_and64.deb ...
Unpacking libnginx-mod-http-xslt-filter (1.18.0-6ubuntu14.3) ...
Selecting previously unselected package libnginx-mod-mail.
Preparing to unpack .../4-libnginx-mod-mail_1.18.0-6ubuntu14.3_and64.deb ...
Unpacking libnginx-mod-mail (1.18.0-6ubuntu14.3) ...

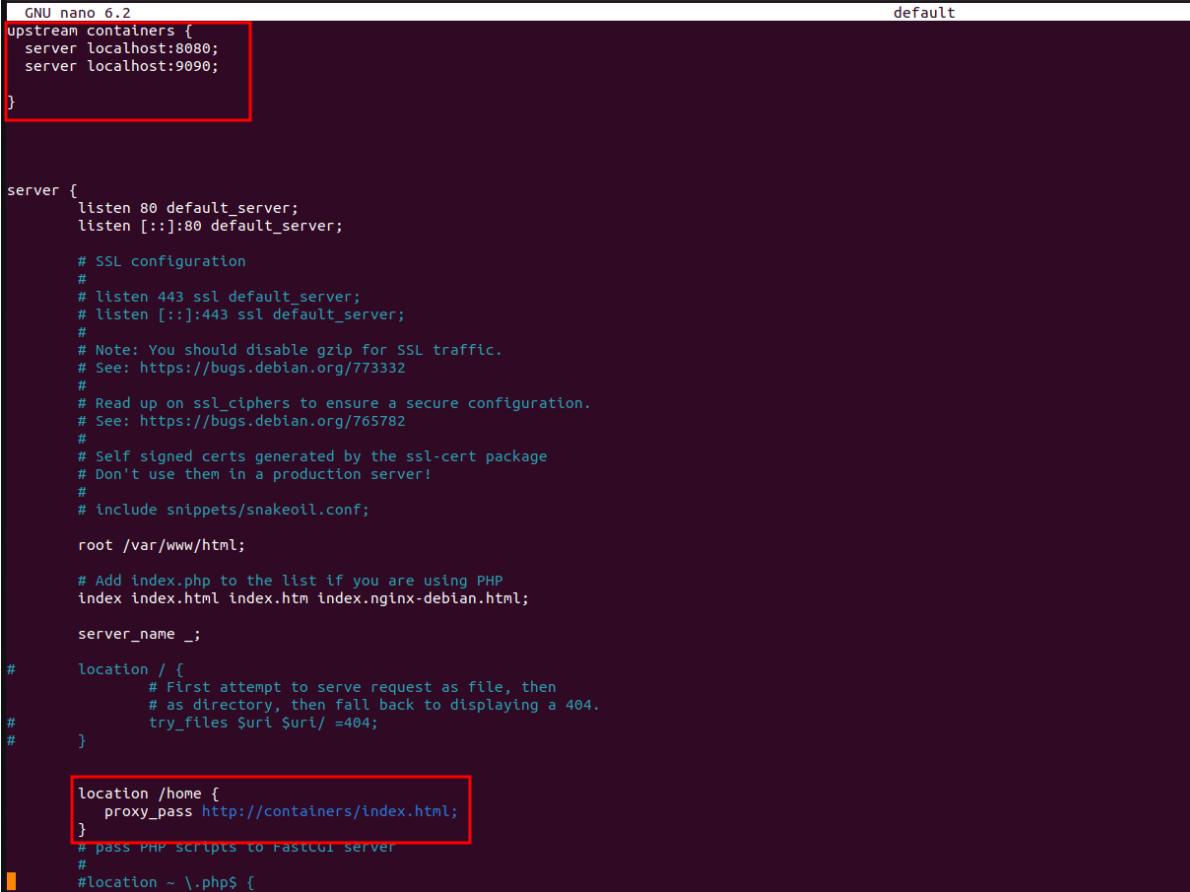
```

Figure 19: Installed Nginx Webserver

To configure the Nginx loadbalancer, I will access the configuration file of Nginx webserver through the command

```
* sudo nano /etc/nginx/sites-available/default
```

And the configuration for the server goes thus in the screenshot



```
GNU nano 6.2                                     default
upstream containers {
server localhost:8080;
server localhost:9090;
}

server {
listen 80 default_server;
listen [::]:80 default_server;

# SSL configuration
#
# listen 443 ssl default_server;
# listen [::]:443 ssl default_server;
#
# Note: You should disable gzip for SSL traffic.
# See: https://bugs.debian.org/773332
#
# Read up on ssl_ciphers to ensure a secure configuration.
# See: https://bugs.debian.org/765782
#
# Self signed certs generated by the ssl-cert package
# Don't use them in a production server!
#
# include snippets/snakeoil.conf;

root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

# location / {
#     # First attempt to serve request as file, then
#     # as directory, then fall back to displaying a 404.
#     try_files $uri $uri/ =404;
# }

location /home {
    proxy_pass http://containers/index.html;
}
# pass PHP scripts to FastCGI server
#
#location ~ \.php$ {
```

Figure 20: Loadbalancer Configurations

After this configuration has been update, we will restart the nginx webserver to activate the configuration using the command as given thus

```
* systemctl restart nginx
```

To verify that the Load balancer is working appropriately, we will check nginx server running on my host computer while specifying the path as shown in the snapshot



Welcome to the Knowledge Hub for Docker A

Here you will find a wealth of information on various topics.

- [Technology](#)
- [Science](#)
- [History](#)
- [Literature](#)

Technology

Explore the latest advancements in technology, from AI and machine learning to quantum computing and beyond.

Science

Dive into the mysteries of the universe, from the smallest subatomic particles to the largest galaxy clusters.

History

Discover the stories that have shaped our world, from ancient civilizations to modern times.

Literature

Immerse yourself in the worlds of classic and contemporary literature, from Shakespeare to J.K. Rowling.

Figure 21: Proxy and Load Balancing Nginx

To verify that the application is distributed among the containers, I will check the logs of the Nginx server, as given in the snapshot below

```
docker-container_a-1 | 172.20.0.1 - - [29/Jun/2023:23:02:20 +0000] "GET /index.html HTTP/1.1" 200 1103 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_a-1 | 172.20.0.1 - - [29/Jun/2023:23:03:52 +0000] "GET /index.html HTTP/1.0" 200 1103 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_b-1 | 172.20.0.1 - - [29/Jun/2023:23:03:59 +0000] "GET /index.html HTTP/1.0" 200 430 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_a-1 | 172.20.0.1 - - [29/Jun/2023:23:05:49 +0000] "GET /index.html HTTP/1.0" 200 430 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_b-1 | 172.20.0.1 - - [29/Jun/2023:23:06:02 +0000] "GET /index.html HTTP/1.0" 200 1103 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_b-1 | 172.20.0.1 - - [29/Jun/2023:23:06:44 +0000] "GET /index.html HTTP/1.0" 200 430 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_a-1 | 172.20.0.1 - - [29/Jun/2023:23:07:58 +0000] "GET /index.html HTTP/1.0" 200 1103 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_b-1 | 172.20.0.1 - - [29/Jun/2023:23:08:03 +0000] "GET /index.html HTTP/1.0" 200 430 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_b-1 | 172.20.0.1 - - [29/Jun/2023:23:08:12 +0000] "GET /index.html HTTP/1.0" 200 430 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_a-1 | 172.20.0.1 - - [29/Jun/2023:23:08:23 +0000] "GET /index.html HTTP/1.0" 200 1103 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
docker-container_b-1 | 172.20.0.1 - - [29/Jun/2023:23:09:15 +0000] "GET /home HTTP/1.1" 404 153 "-" Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0 "-"
```

Figure 22: Log Output of the Host Nginx Server

We can see from the logs that anytime the server is accessed, we could hit the two containers periodically.

5. Automate everything in a Bash script (Optional)

- Automate all of the process in a bash script
- Push your code to Version Control Systems (VCS)

To write a basic automation script that will run this process automatically, we can check out this code below

```
#!/bin/bash
```

```
#This bash script is used to automate the flow of creating the
#docker-compose file and tearing it down
```

```
#Change directory to where the compose file is located
cd ~/Docker
```

```
# Build the containers
docker-compose build
```

```
# Start the containers in detached mode
docker-compose up -d
```

```
#check the webpages of the containers
curl http://127.0.0.1:8080
curl http://127.0.0.1:9090
```

```
# Stop the containers
docker-compose stop
```

```
# Remove the containers
docker-compose down
```

The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a file named 'Automation.sh' under a folder 'LS_1'. The main editor area displays the content of 'Automation.sh', which is a bash script for automating Docker container management. The script includes comments and commands for building, starting, checking, stopping, and removing containers. The bottom pane shows the Terminal output, which includes the command to commit the script to a Git repository and the subsequent push command. The output shows that the commit was successful, creating a new file 'Automation.sh' and updating the 'LS Lab 1 22-23.pdf' file.

```
Automation.sh
1  #!/bin/bash
2
3  #This bash script is used to automate the flow of creating the
4  #docker-compose file and tearing it down
5
6  #Change directory to where the compose file is located
7  cd ~/Docker
8
9  # Build the containers
10 docker-compose build
11
12 # Start the containers in detached mode
13 docker-compose up -d
14
15 #check the webpages of the containers
16 curl http://127.0.0.1:8080
17 curl http://127.0.0.1:9090
18
19 # Stop the containers
```

```
micha@DESKTOP-TE3G23U MINGW64 ~/Documents/LS_1 (main)
$ git commit -m "Add the Automation script to the Repository"
[main 10705da] Add the Automation script to the Repository
2 files changed, 23 insertions(+)
create mode 100644 Automation.sh
create mode 100644 LS Lab 1 22-23.pdf

micha@DESKTOP-TE3G23U MINGW64 ~/Documents/LS_1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
```

Figure 23: Pushing Automation Script to Git

To verify my git repository






	Los-merengue Add the Automation script to ...	10705da 5 minutes ago	 3 commits
	Automation.sh	Add the Automation script to the Repository	5 minutes ago
	LS Lab 1 22-23.pdf	Add the Automation script to the Repository	5 minutes ago
	README.md	First commit	12 minutes ago

Figure 24: GitHub Repository

Reference

1. Docker Installation in Ubuntu (<https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository>)
2. Learning Docker (<https://docker-curriculum.com/#prerequisites>)
3. Nginx Documentation (<https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/>)
4. Cloud Infra Services (<https://cloudinfrastructureservices.co.uk/install-nginx-with-docker-compose/>)
5. Nginx Load Balancer (<https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-setup-an-Nginx-load-balancer-example>)
6. Docker-compose documentation (<https://docs.docker.com/compose/features-uses/>)