5.8 TO DO

Make a directory:

```
wajdabdal-hadee@wajdabdal-hadee:~$ mkdir grp-members
wajdabdal-hadee@wajdabdal-hadee:~$ cd grp-members/
wajdabdal-hadee@wajdabdal-hadee:~/grp-members$
```

Figure 5.8.1 Make a directory for the example.

Creating a Dockerfile:

mkdir grp-members

cd grp-members

nano Dockerfile

Dockerfile content:

FROM ubuntu

RUN apt update

RUN apt install -y apache2

RUN apt clean

RUN rm -rf /var/lib/apt/lists/*

WORKDIR /var/www/html

COPY ./index.html /usr/local/apache2/htdocs/index.html

EXPOSE 80

CMD ["apachectl", "-D", "FOREGROUND"]

See figure 5.8.2

```
  GNU nano 7.2                                  Dockerfile
FROM ubuntu
RUN apt update
RUN apt install -y apache2
RUN apt clean
RUN rm -rf /var/lib/apt/lists/*
WORKDIR /var/www/html
COPY ./index.html /usr/local/apache2/htdocs/index.html
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

Figure 5.8.2 Dockerfile configuration.

Creating an index.html file:

```
nano index.html
```

index.html content:

```
<!doctype html>
<center>
<html>
<head>
<title>Group Members</title>
</head>
<body>
<h1>Ali Khuffash</h1>
<h1>Wajd Abd Al-Hadee</h1>
<p>Admin Lab</p>
</body>
</html>
```

See figure 5.8.3

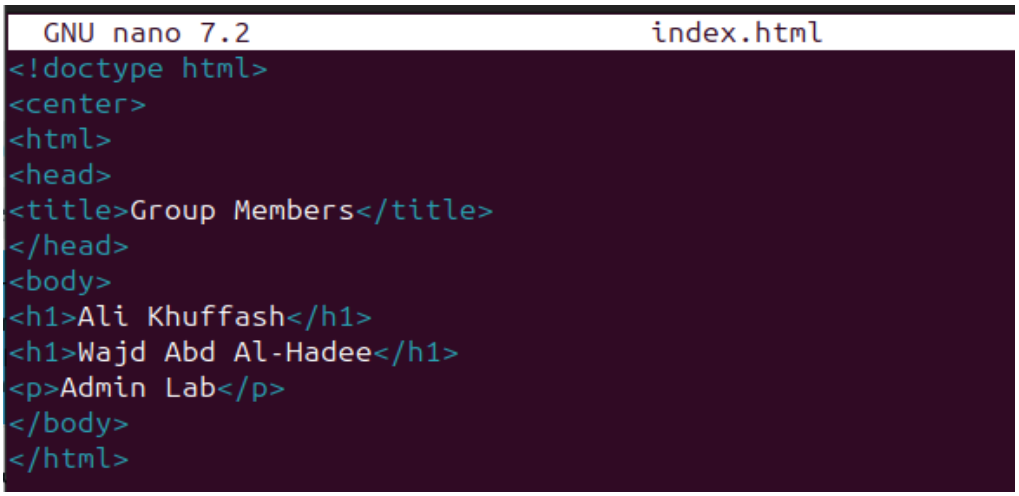

Figure 5.8.3 html configuration.

Building the image:

```
wajdabdal-hadee@wajdabdal-hadee:~/grp-members$ docker build -t grp-image .
[+] Building 0.1s (12/12) FINISHED                                  docker:default
 => [internal] load build definition from Dockerfile                         0.0s
 => => transferring dockerfile: 264B                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest             0.0s
 => [internal] load .dockerignore                                            0.0s
 => => transferring context: 2B                                              0.0s
 => [1/7] FROM docker.io/library/ubuntu:latest                               0.0s
 => [internal] load build context                                            0.0s
 => => transferring context: 204B                                            0.0s
 => CACHED [2/7] RUN apt update                                              0.0s
 => CACHED [3/7] RUN apt install -y apache2                                  0.0s
 => CACHED [4/7] RUN apt clean                                               0.0s
 => CACHED [5/7] RUN rm -rf /var/lib/apt/lists/*                             0.0s
 => CACHED [6/7] WORKDIR /var/www/html                                       0.0s
 => [7/7] COPY ./index.html /usr/local/apache2/htdocs/index.html             0.0s
 => exporting to image                                                       0.0s
 => => exporting layers                                                      0.0s
 => => writing image sha256:baa354208633510e2025d72536866dfb60e9dd13c0f1fc4c5f323091e82  0.0s
 => => naming to docker.io/library/grp-image                                 0.0s
```

Figure 5.8.4 Building the image.

Running a container with the custom image:

But in this step, another modification will be added, which is that if the administrator makes any changes to the HTML file, it will be displayed directly on the browser without the need for new configurations, see figure 5.8.5

```
wajdabdal-hadee@wajdabdal-hadee:~/grp-members$ docker run -dt -p 8082:80 --name members -v $(pwd)/:/usr/local/
apache2/htdocs/ httpd
b2914d5838342966604d4789b06417022fff467f1644f1cde0cba7148a44d0fe
```

Figure 5.8.5 Running the container with persistent updates.

Verifying webpage access from the host:

By http://172.17.0.5:8082 on your browser, see figure 5.8.6

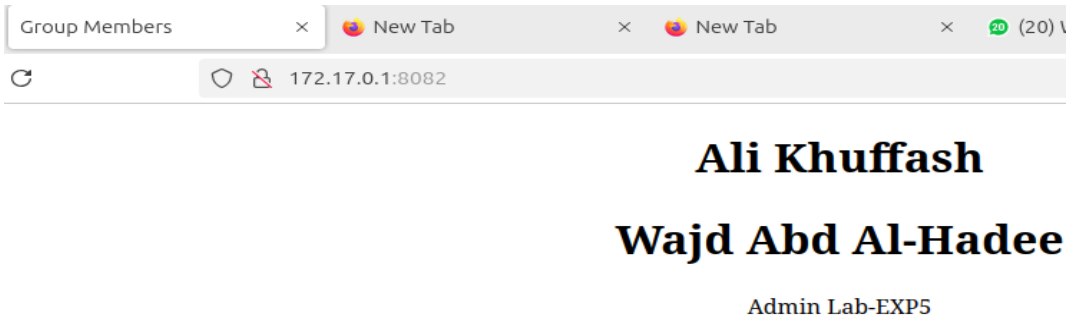

Figure 5.8. 6 Verifying by IP in the fire-fox browser.

Now try to edit anything in the index.html file. Save the new updates in the html file. See figure 5.8.7 and figure 5.8.8 for more information.

```
wajdabdal-hadee@wajdabdal-hadee:~/grp-members$ docker restart members
members
```

Figure 5.8.7 Restart the docker for the new updates.

Press CTRL + shift + R if the page was not changed!

Figure 5.8.7 The page after refresh it with new changes.

6 Results and Observations

- Docker was successfully installed and configured.
- Containers were created and managed efficiently.
- Persistent storage was successfully implemented.
- Network settings were configured to allow external access.
- A custom container image with a pre-configured web server was built and deployed.

7 Conclusion

This lab provided hands-on experience with Docker, demonstrating the ease of deploying and

managing containers. The experiment emphasized key containerization concepts, persistent storage

techniques, and networking configurations.