An-Najah National University

Faculty of Engineering and Information Technology

Network Administration Lab Report

Docker Network Administration

EXP#5

Dr. Bakr Abd Al-Haq

Team Members: Ali Khuffash, Wajd Abd Al-Hadee

Table of Contents

1. Abstract

This report details the execution of a network administration lab focused on Docker container management. The experiment covered fundamental concepts such as installing Docker, downloading and managing container images, executing commands within containers, and implementing networking. The results demonstrate the successful setup and management of Docker containers along with persistent storage and network communication.

2. Introduction

Docker is a containerization platform that enables applications to run in isolated environments called containers. This experiment aims to explore Linux containers, distinguish them from virtual machines (VMs), install Docker, pull and build container images, manage containers, and configure network settings.

3. Objectives

Upon completion of this lab, the following objectives were achieved:

- Understanding the difference between containers and VMs.
- Identifying various container solutions.
- Installing and configuring Docker on an Ubuntu server.
- Using pre-built container images from Docker Hub.
- Creating and managing custom container images.
- Configuring container networking.

4. Pre-requisites & Resources

Pre-reading

- Linux Containers (LXC) - Wikipedia
- Docker Documentation

- Snap Packages - Wikipedia
- Dockerfile Reference - Docker Docs Required Resources
- Ubuntu server VM with a bridged adapter
- Internet connection

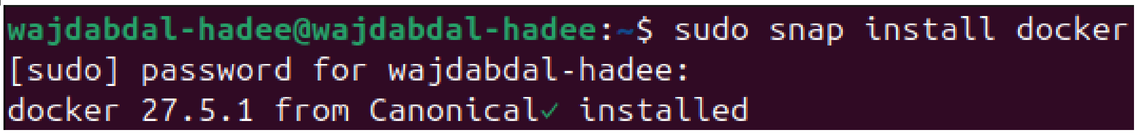
5.Methodology

5.1 Installing Docker

Docker is an open-source platform used to develop, ship, and run applications in isolated environments called containers. It allows developers to run applications with all necessary dependencies and environments reliably, whether on a local machine or on servers.

Docker was installed using Snap Package:

```
sudo snap install docker
```



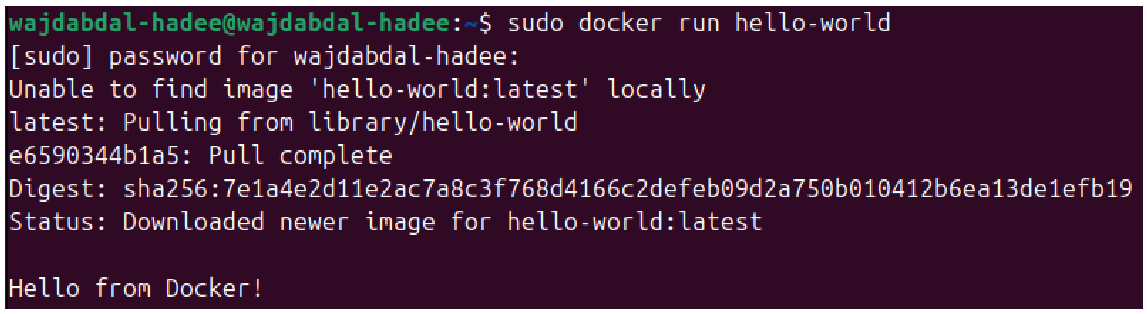
```
wajdabdal-hadee@wajdabdal-hadee:~$ sudo snap install docker
[sudo] password for wajdabdal-hadee:
docker 27.5.1 from Canonical✓ installed
```

Figure 5.1.1 Installing the docker.

To verify the installation:

```
sudo docker run hello-world
```

see figure 5.1.2



```
wajdabdal-hadee@wajdabdal-hadee:~$ sudo docker run hello-world
[sudo] password for wajdabdal-hadee:
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b010412b6ea13de1efb19
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
```

Figure 5.1.2 Verify the installation of dockers.

To enable Docker for a non-root user(use docker without root permissions):

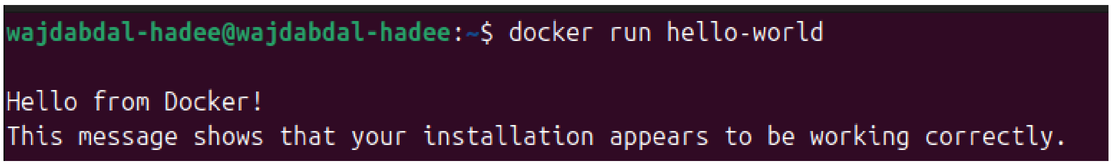
```
sudo groupadd docker

sudo usermod -aG docker $USER

newgrp docker
```

You should reboot your machine to apply the updates.

Verification:

```
docker run hello-world
```

see figure 5.1.2



```
wajdabdal-hadee@wajdabdal-hadee:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Figure 5.1.3 Check if docker now for non-root user or not.

5.2 Downloading Images

In Docker, images are templates or snapshots used to build and run containers. A Docker image is a static version of a runtime environment that includes everything needed for an application to run, such as:

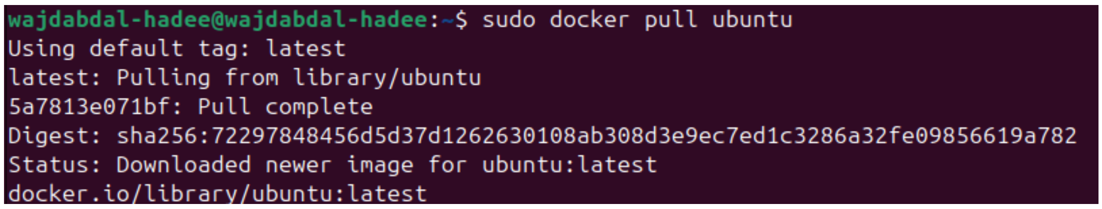
- The operating system (e.g., Ubuntu, Alpine, Debian, etc.).
- Software (e.g., Apache, Nginx, Node.js, Python, MySQL, and others).
- Configurations (such as server configuration files).

Images were retrieved from Docker Hub using:

```
docker pull ubuntu
```

In Docker, the docker pull command is used to download (or "pull") an image from an online repository, such as Docker

Hub, to your local machine, see figure 5.2.1



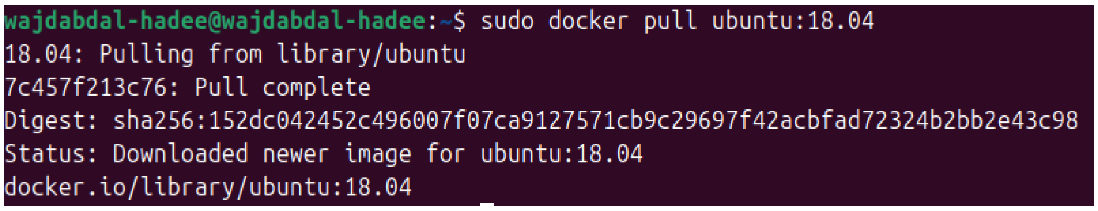
```
wajdabdal-hadee@wajdabdal-hadee:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
5a7813e071bf: Pull complete
Digest: sha256:72297848456d5d37d1262630108ab308d3e9ec7ed1c3286a32fe09856619a782
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

Figure 5.2.1 Retrieve the ubuntu.

To download a specific version:

```
docker pull ubuntu:18.04
```

see figure 5.2.2



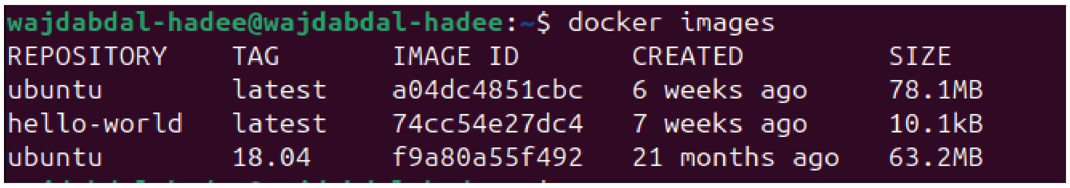
```
wajdabdal-hadee@wajdabdal-hadee:~$ sudo docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
7c457f213c76: Pull complete
Digest: sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04
```

Figure 5.2.2 Retrieve the ubuntu with specific version.

To list downloaded images:

```
docker images
```

see figure 5.2.3



```
wajdabdal-hadee@wajdabdal-hadee:~$ docker images
REPOSITORY     TAG       IMAGE ID       CREATED         SIZE
ubuntu         latest    a04dc4851cbc   6 weeks ago     78.1MB
hello-world    latest    74cc54e27dc4   7 weeks ago     10.1kB
ubuntu         18.04     f9a80a55f492   21 months ago   63.2MB
```

Figure 5.2.3 Display the images.

5.3 Container Management

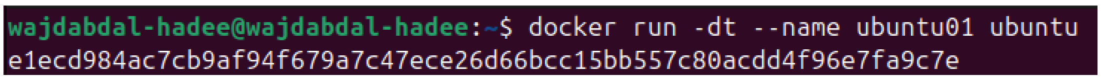
A container is an isolated and independent computing unit that includes everything an application needs to run in a specific runtime environment. Each container has all the necessary components and dependencies (such as libraries, tools, and other files) required for execution. In simple terms, containers provide a standardized way to run applications, independent of the host system."

- Containers ensure applications run in complete isolation from other systems.
- Unlike Virtual Machines (VMs), containers do not require a full operating system; instead, they share the host system's kernel, making them lighter and more efficient.
- Containers can run on any device or server with Docker or a similar platform, allowing easy application migration between environments (e.g., from a local machine to servers or the cloud)

Creating a container:

docker run -dt --name ubuntu01 ubuntu

see figure 5.3.1



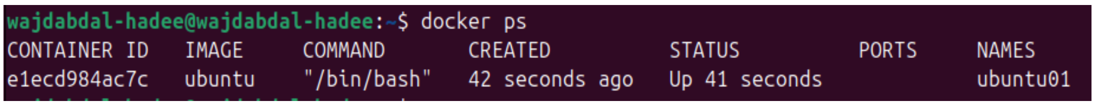
```
wajdabdal-hadee@wajdabdal-hadee:~$ docker run -dt --name ubuntu01 ubuntu
e1ecd984ac7cb9af94f679a7c47ece26d66bcc15bb557c80acdd4f96e7fa9c7e
```

Figure 5.3.1 Create a new container with ubuntu01 name.

Verifying running containers:

docker ps

see figure 5.3.2



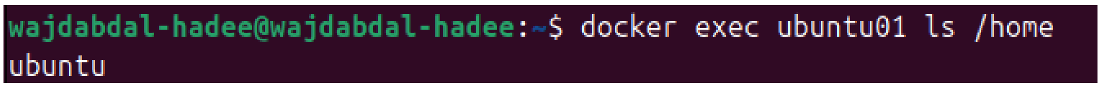
```
wajdabdal-hadee@wajdabdal-hadee:~$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
e1ecd984ac7c   ubuntu    "/bin/bash"   42 seconds ago   Up 41 seconds             ubuntu01
```

Figure 5.3.2 Display the running containers.

Executing commands inside the container:

docker exec ubuntu01 ls /home

see figure 5.3.3



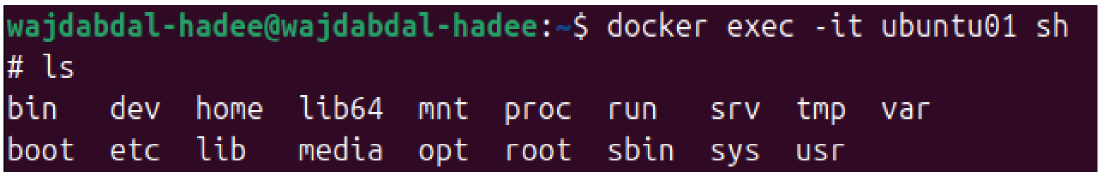
```
wajdabdal-hadee@wajdabdal-hadee:~$ docker exec ubuntu01 ls /home
ubuntu
```

Figure 5.3.3 Execute container commands in the terminal.

Entering the container's shell:

> docker exec -it ubuntu01 sh

see figure 5.3.4



```
wajdabdal-hadee@wajdabdal-hadee:~$ docker exec -it ubuntu01 sh
# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

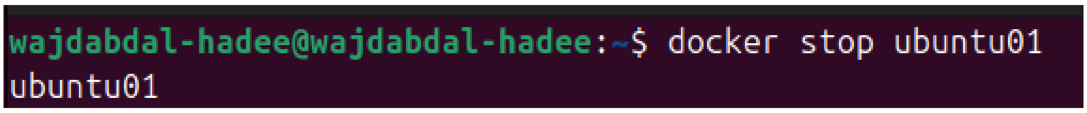
Figure 5.3.4 Convert into container terminal.

5.4 Stopping & Deleting Containers

Stopping a container:

> docker stop ubuntu01

see figure 5.4.1



```
wajdabdal-hadee@wajdabdal-hadee:~$ docker stop ubuntu01
ubuntu01
```

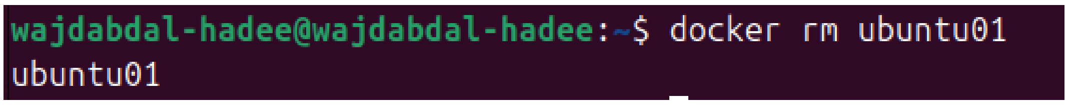
Figure 5.4.1 Stop the container.

Maybe it will take some seconds to stop the container

Deleting a container:

> docker rm ubuntu01

see figure 5.4.2



```
wajdabdal-hadee@wajdabdal-hadee:~$ docker rm ubuntu01
ubuntu01
```

Figure 5.4.2 Remove the container.

You should know, that you can't remove any container before stop it. So, stop the container and then remove it.

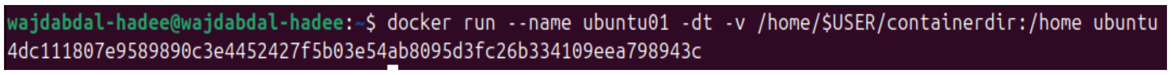
5.5 Persistent Storage

Persistent storage is a type of storage that retains data even after restarting or stopping the device, container, or application. Unlike ephemeral storage—the default storage mode for containers, which is lost when a container stops or restarts—persistent storage ensures data remains intact and accessible later.

Creating a persistent storage container:

```
docker run --name ubuntu01 -dt -v
/home/$USER/containerdir:/home ubuntu
```

see figure 5.5.1



```
wajdabdal-hadee@wajdabdal-hadee:~$ docker run --name ubuntu01 -dt -v /home/$USER/containerdir:/home ubuntu
4dc111807e9589890c3e4452427f5b03e54ab8095d3fc26b334109eea798943c
```

Figure 5.5.1 Creating a persistent storage container.

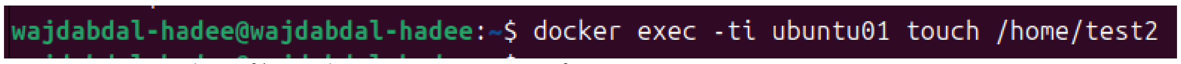
Verifying persistent storage:

```
docker exec -ti ubuntu01 sh

touch /home/testfile
```

or:

```
docker exec -ti ubuntu01 touch /home/test2
```

see figure 5.5.2



```
wajdabdal-hadee@wajdabdal-hadee:~$ docker exec -ti ubuntu01 touch /home/test2
```

Figure 5.5.2 Make a file in the container to verify.

5.6 Networking Configuration

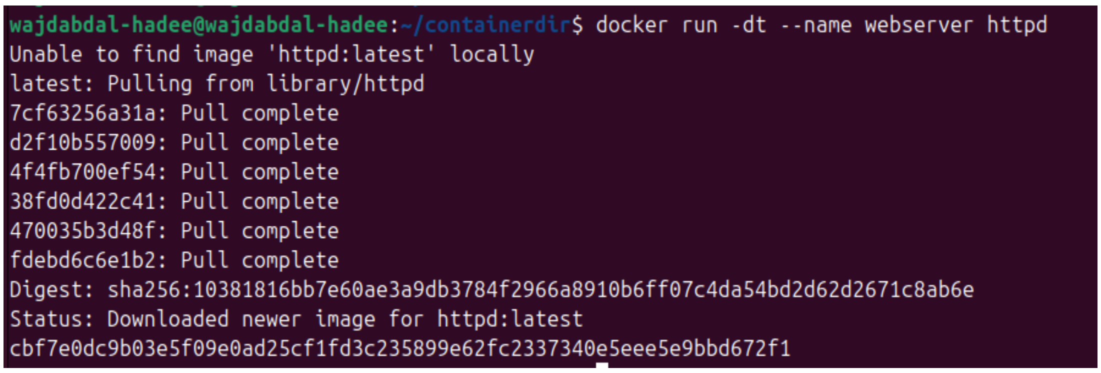
Networking configuration in Docker defines how containers connect to each other and to external networks, such as the internet or local networks. Docker offers multiple network setup options, allowing you to control container communication with each other and with the host system.

Setting up a web server container:

```
docker run -dt --name webserver httpd
```

Apache HTTP Server (or simply Apache) is an open-source web server software used to serve content on the internet. It is responsible for running web applications, serving HTML pages, and handling incoming requests from browsers such as Chrome or Firefox.

'httpd' stands for 'HyperText Transfer Protocol Daemon,' and it refers to the name of the Apache service in operating systems like Linux and Unix. See figure 5.6.1



```
wajdabdal-hadee@wajdabdal-hadee:~/containerdir$ docker run -dt --name webserver httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
7cf63256a31a: Pull complete
d2f10b557009: Pull complete
4f4fb700ef54: Pull complete
38fd0d422c41: Pull complete
470035b3d48f: Pull complete
fdebd6c6e1b2: Pull complete
Digest: sha256:10381816bb7e60ae3a9db3784f2966a8910b6ff07c4da54bd2d62d2671c8ab6e
Status: Downloaded newer image for httpd:latest
cbf7e0dc9b03e5f09e0ad25cf1fd3c235899e62fc2337340e5eee5e9bbd672f1
```

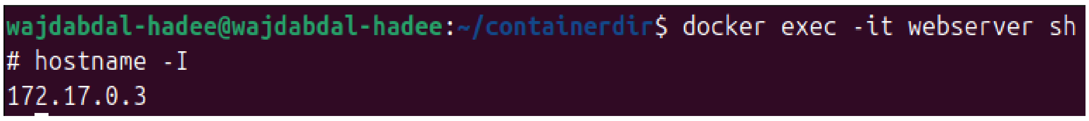
Figure 5.6.1 Setting a webserver container

Checking container IP:

docker exec -ti webserver sh

hostname -I

see figure 5.6.2



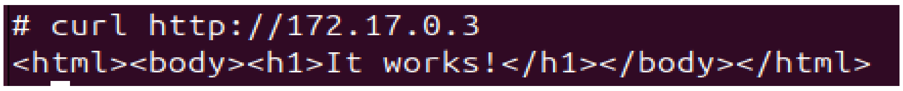
```
wajdabdal-hadee@wajdabdal-hadee:~/containerdir$ docker exec -it webserver sh
# hostname -I
172.17.0.3
```

Figure 5.6.2 Checking the container IP.

Accessing the web server from the host:

curl http://[container_IP]

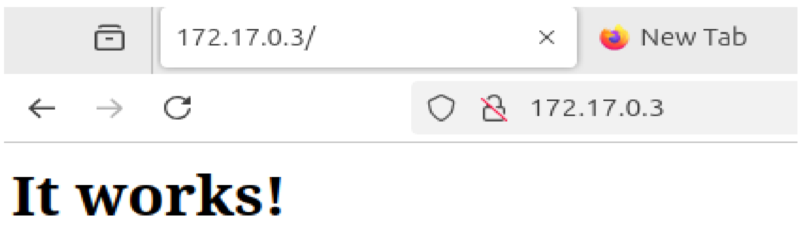
curl is a command-line tool used to send and receive data over the internet using various protocols such as HTTP, HTTPS, FTP, SMTP, and others. It is a powerful tool commonly used in various applications, from testing APIs to downloading files from the internet. See figure 5.6.3



```
# curl http://172.17.0.3
<html><body><h1>It works!</h1></body></html>
```

Figure 5.6.3 Verifying by curl command in the container terminal.

Or you can see the result by your browser, see figure 5.7.4



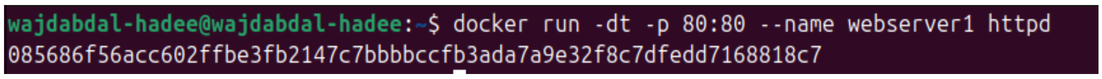
It works!

Figure 5.6.4 Verifying by IP in the fire-fox browser.

Exposing the server to external machines:

```
docker run -dt -p 80:80 --name webserver httpd
```

see figure 5.6.5



```
wajdabdal-hadee@wajdabdal-hadee:~$ docker run -dt -p 80:80 --name webserver1 httpd
085686f56acc602ffbe3fb2147c7bbbbccfb3ada7a9e32f8c7dfedd7168818c7
```

Figure 5.6.5 Exposing the server to external machines.

5.7 Building a Custom Docker Image

A Dockerfile is a text file containing a set of commands and instructions used to create a Docker image. It defines how to build a container image that includes everything the application needs to run in a Docker environment. In other words, the Dockerfile is used to create an image that contains the application, its dependencies, and the necessary configurations for its execution.

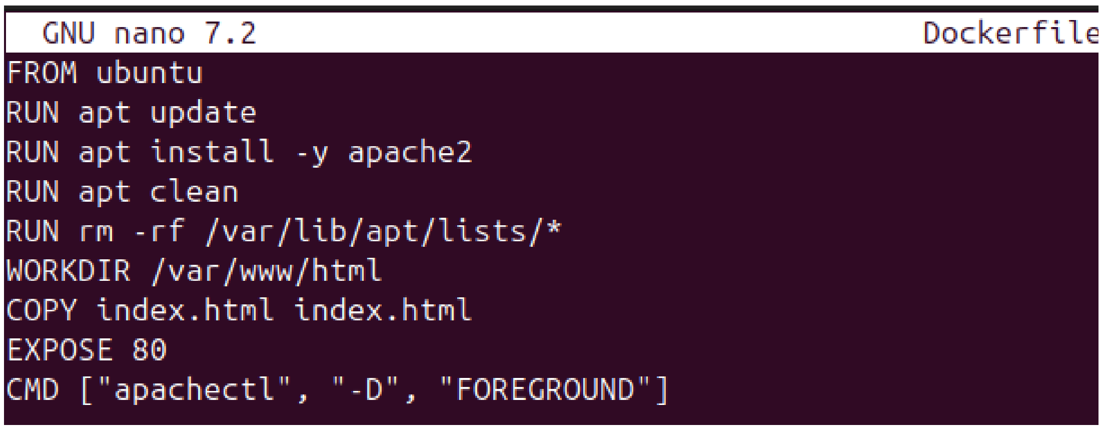
Creating a Dockerfile:

```
mkdir dockerexample
cd dockerexample
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 index.html

EXPOSE 80

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

See figure 5.7.1



```
  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 index.html
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

Figure 5.7.1 Dockerfile configuration.

Creating an index.html file:

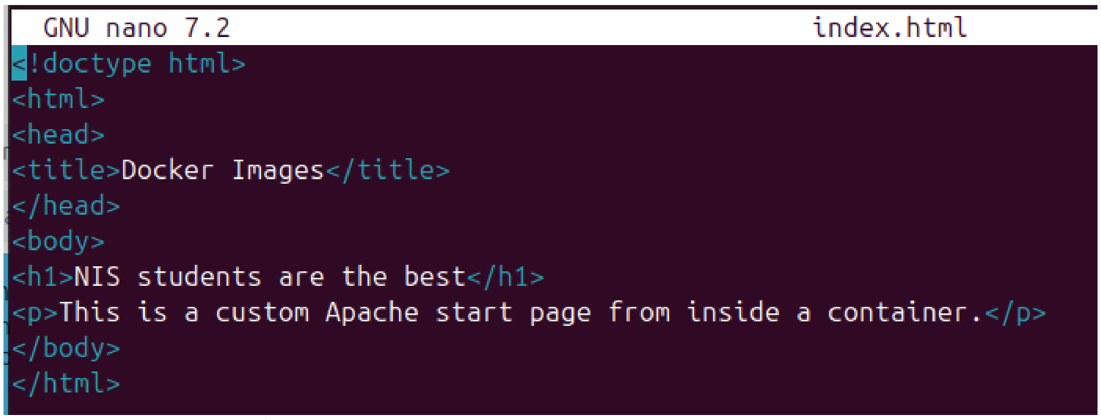
```
nano index.html
```

index.html content:

```
<!doctype html>

<html>

<head>

<title>Docker Images</title>

</head>

<body>

<h1>NIS students are the best</h1>

<p>This is a custom Apache start page from inside a
container.</p>
```

```
</body>

</html>
```

See figure 5.7.2



```
  GNU nano 7.2                                            index.html
<!doctype html>
<html>
<head>
<title>Docker Images</title>
</head>
<body>
<h1>NIS students are the best</h1>
<p>This is a custom Apache start page from inside a container.</p>
</body>
</html>
```

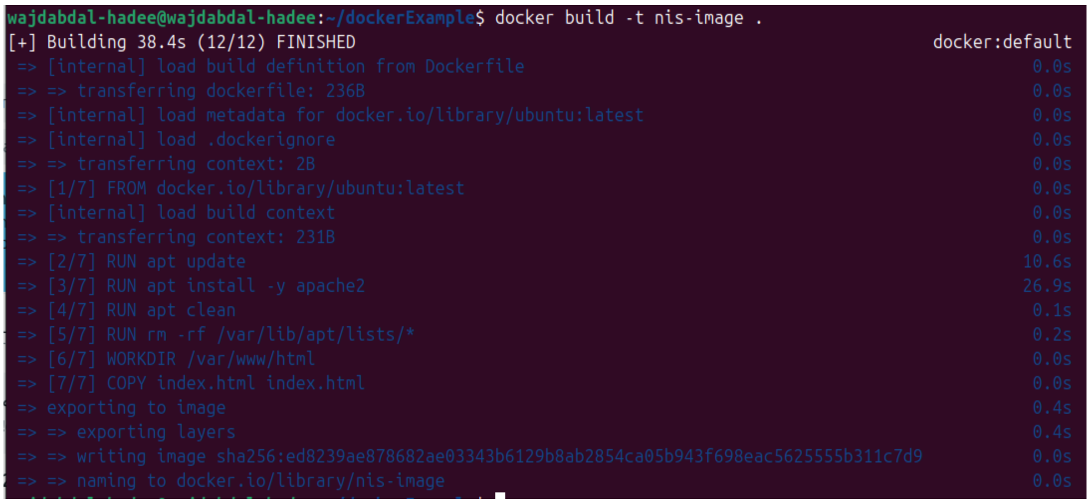
Figure 5.7.2 html configuration.

Building the image:

In the context of Docker, 'build' refers to the process of creating a Docker image from a Dockerfile (build file) using specific commands. This process transforms the Dockerfile into an image that contains everything needed for the application or runtime environment to run inside a Docker container. See figure 5.7.3

```
docker build -t nis-image .
```

. meaning the current directory.



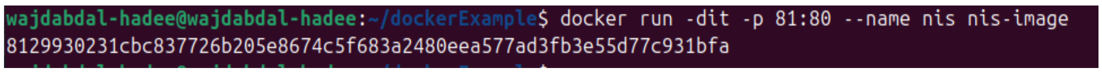
```
wajdabdal-hadee@wajdabdal-hadee:~/dockerExample$ docker build -t nis-image .
[+] Building 38.4s (12/12) FINISHED                                              docker:default
 => [internal] load build definition from Dockerfile                                        0.0s
 => => transferring dockerfile: 236B                                                        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: 231B                                                           0.0s
 => [2/7] RUN apt update                                                                   10.6s
 => [3/7] RUN apt install -y apache2                                                       26.9s
 => [4/7] RUN apt clean                                                                     0.1s
 => [5/7] RUN rm -rf /var/lib/apt/lists/*                                                   0.2s
 => [6/7] WORKDIR /var/www/html                                                             0.0s
 => [7/7] COPY index.html index.html                                                        0.0s
 => exporting to image                                                                      0.4s
 => => exporting layers                                                                     0.4s
 => => writing image sha256:ed8239ae878682ae03343b6129b8ab2854ca05b943f698eac5625555b311c7d9  0.0s
 => => naming to docker.io/library/nis-image                                                0.0s
```

Figure 5.7.3 Building the image.

Running a container with the custom image:

```
docker run -dit -p 80:80 --name nis nis-image
```

You should be aware that it is not possible to use -p more than once for the same port. see figure 5.7.4



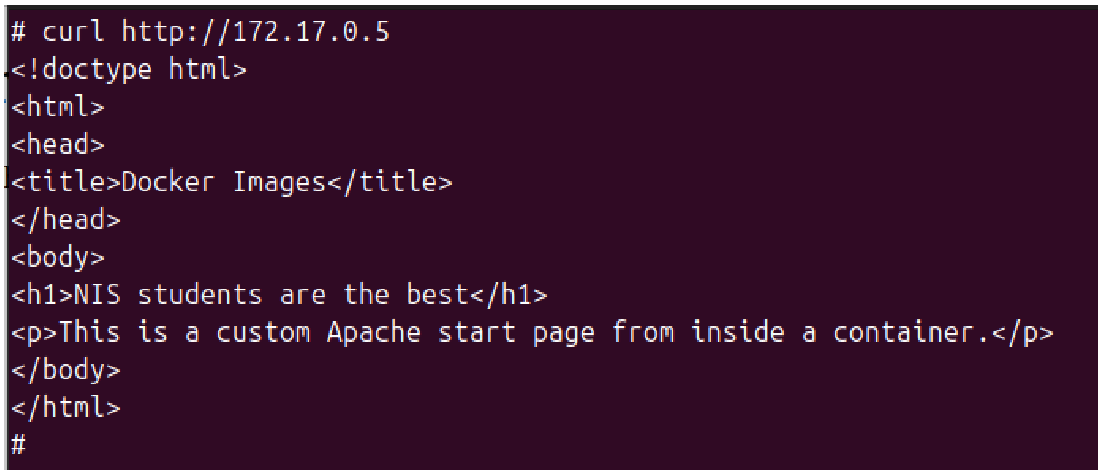
```
wajdabdal-hadee@wajdabdal-hadee:~/dockerExample$ docker run -dit -p 81:80 --name nis nis-image
8129930231cbc837726b205e8674c5f683a2480eea577ad3fb3e55d77c931bfa
```

Figure 5.7.4 Running the container with the custom image.

Verifying webpage access from the host:

curl http://[VM_IP]

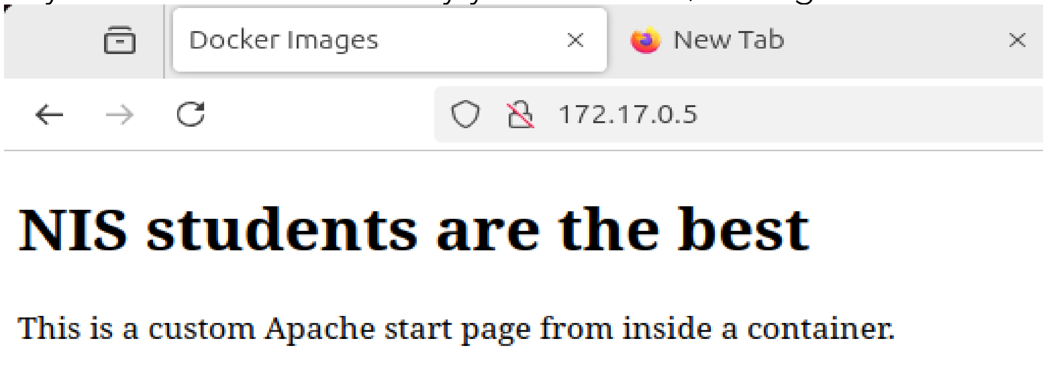
see figure 5.7.5



```
# curl http://172.17.0.5
<!doctype html>
<html>
<head>
<title>Docker Images</title>
</head>
<body>
<h1>NIS students are the best</h1>
<p>This is a custom Apache start page from inside a container.</p>
</body>
</html>
#
```

Figure 5.7.5 Verifying by curl command in the container terminal.

Or you can see the result by your browser, see figure 5.7.6



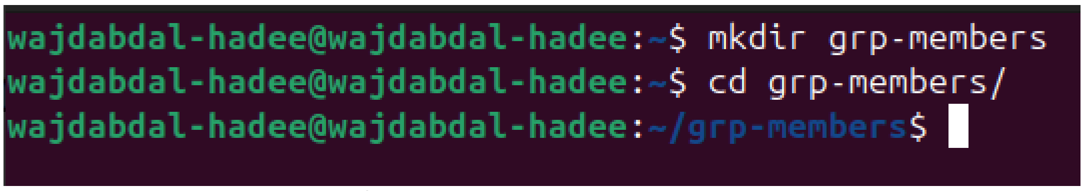
NIS students are the best

This is a custom Apache start page from inside a container.

Figure 5.7.6 Verifying by IP in the fire-fox browser.

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.

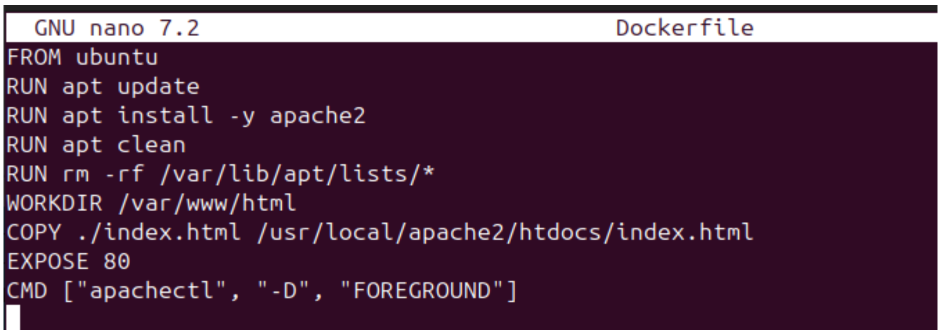
Creating a Dockerfile:

```
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.

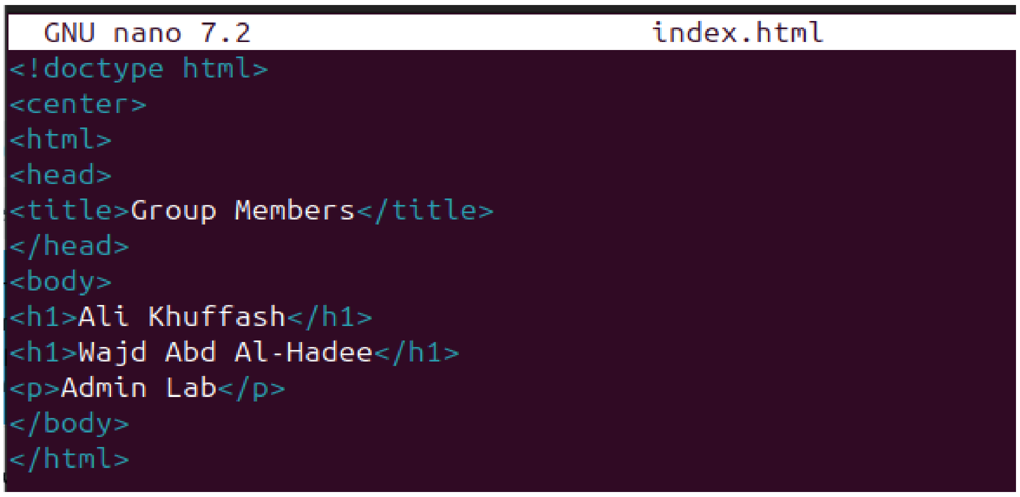
Creating an index.html file:

```
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>
```

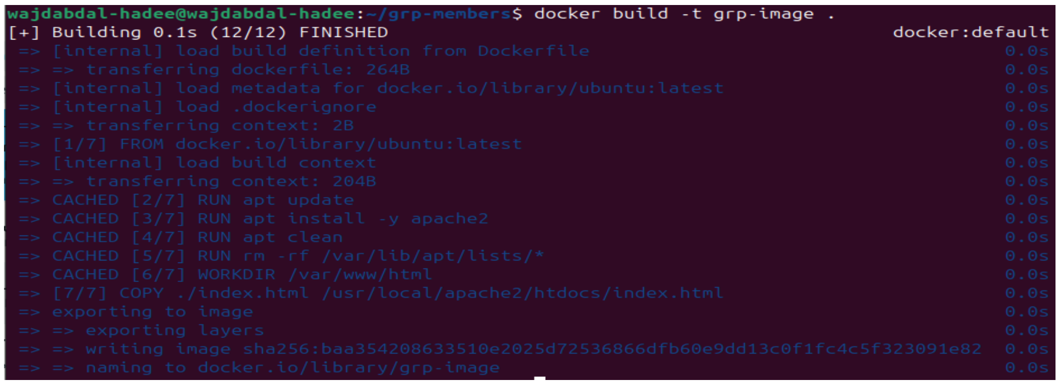
See figure 5.8.3



```
  GNU nano 7.2                                 index.html
<!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>
```

Figure 5.8.3 html configuration.

Building the image:

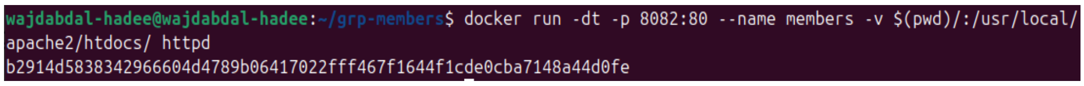


```
wajdabdal-hadee@wajdabdal-hadee:~/grp-members$ docker build -t grp-image .
[+] Building 0.1s (12/12) FINISHED                                    docker:default
 => [internal] load build definition from Dockerfile                           0.0s
 => => transferring dockerfile: 264B                                           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

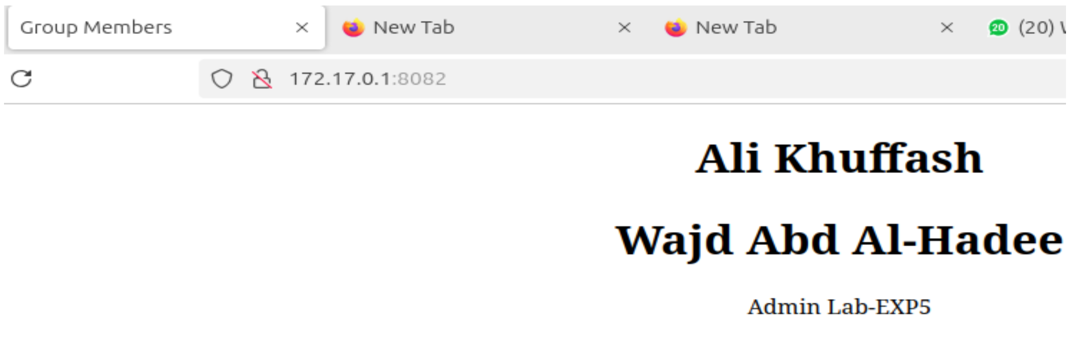


```
wajdabdal-hadee@wajdabdal-hadee:~/grp-members$ docker run -dt -p 8082:80 --name members -v $(pwd)/:/usr/local/
apache2/htdocs/ httpd
b2914d5838342966604d4789b06417022fff467f1644f1cde0cba7148a44d0fe
```

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



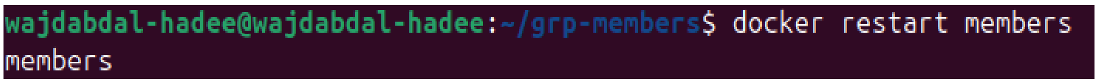
Ali Khuffash

Wajd Abd Al-Hadee

Admin Lab-EXP5

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

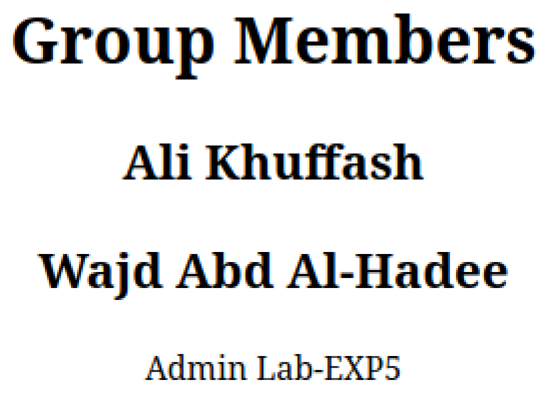
Now try to edit anything in the index.html file. Save the new updates in the html file. See figure 5.8.7 and figure 5.8.8 for more information.



```
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!



Group Members

Ali Khuffash

Wajd Abd Al-Hadee

Admin Lab-EXP5

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.

8 References

- Linux Containers (LXC) - Wikipedia
- Docker Documentation - https://docs.docker.com/
- Docker Hub - https://hub.docker.com/
- Snap Packages - Wikipedia

9 Appendices

- List of executed Docker commands
- Screenshots (if applicable)