Exp 5 Docker

sudo snap install docker

to verify that docker installed successfully use the following command:

sudo docker run hello-world

set you current user account to use docker without root permissions:

sudo groupadd docker

sudo usermod -aG docker $USER
newgrp docker // يقوم هذا الأمر بتحديث إعدادات المستخدم الحالي بحيث تنعكس التعديلات التي قمنا بها دون الحاجة إلى إعادة تشغيل الجهاز.

if the last command fails reboot your machine and run the command again.

احتمال كبير ما يزبط وقتيها اعمل

sudo reboot

verify you have permission to use docker without root permissions:

docker run hello-world

Download images

Docker Hub هو المستودع الرسمي للصور الجاهزة والمعدة مسبقًا.

🔷 Docker Hub يمكنك الوصول إليه عبر الرابط:

يحتوي على آلاف الصور لأنظمة التشغيل والتطبيقات **Ubuntu, MySQL, Nginx, Python** مثل

To download that image, use the following command:

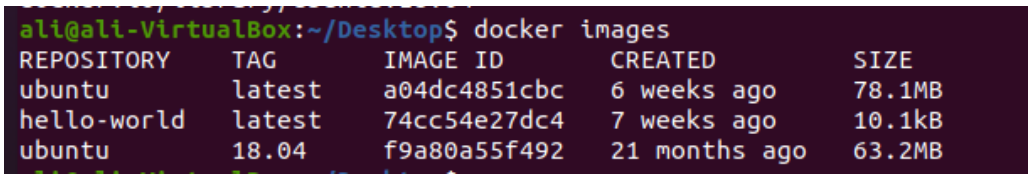
كيفية تحميل صورة باستخدام Docker ؟

`docker pull ubuntu` // هذا الأمر سيقوم بتحميل أحدث إصدار من صورة اوبنتو الى جهازك (`latest`)

`docker pull ubuntu:18.04` // **مثال على تحميل إصدار محدد (Ubuntu 18.04):**

To list the images, you have downloaded use the command:

`docker images`



- `REPOSITORY` → اسم الصورة (`ubuntu`)
- `TAG` → إصدار الصورة (`latest` أو `18.04`)
- `IMAGE ID` → معرف فريد للصورة
- `CREATED` → متى تم إنشاء هذه الصورة
- `SIZE` → حجم الصورة بالميجابايت

Container management:

To create a container from an image, we use the command:

```
docker run -dt --name ubuntu01 ubuntu
```

💡 شرح الأوامر:

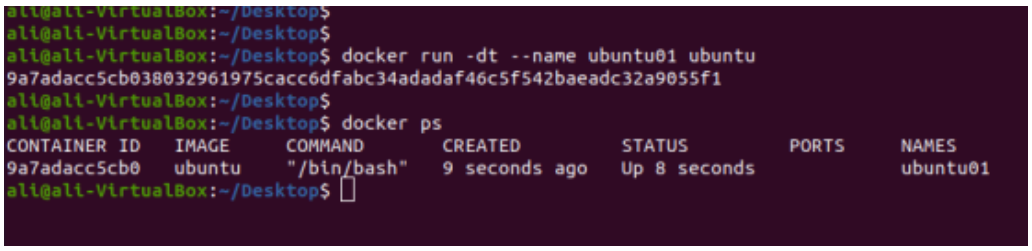
- `docker run` → ينشئ حاوية جديدة ويشغلها.
- `d` → تعني **detached mode**، أي تشغيل الحاوية في الخلفية (دون الدخول إليها مباشرة).
- `t` → تعني **TTY (terminal interface)**، مما يسمح بالحفاظ على تشغيل الحاوية حتى لو لم يكن هناك عملية تعمل بداخلها.
- `-name ubuntu01` → يعطي الحاوية اسمًا مخصصًا (`ubuntu01`).
- `ubuntu` → هي **الصورة** التي سيتم استخدامها لإنشاء الحاوية.

🔷 لماذا نستخدم `-d` و `-t`؟

- بدون `t`، ستتوقف الحاوية فورًا بعد تشغيلها لأنه لا يوجد أي تطبيق يعمل بداخلها.
- بدون `d`، سيتم إدخالك مباشرة إلى سطر أوامر الحاوية، لكنك قد تتعطل هناك إذا لم يكن محدد (Shell) لديك شيل

check if the container is running or not: **التحقق مما إذا كانت الحاوية تعمل**

docker ps

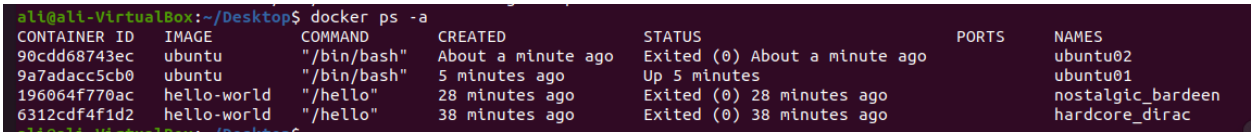


بدنا ننشأ حاوية باسم ثاني:

docker run -d --name ubuntu02 ubuntu

بعدين بدنا نتأكد اذا كان شغال عن طريق: docker ps

بس ما رح تظهر، ليش ؟
لانه مافي هناك عملية تعمل بداخل الحاوية، فإنها ستتوقف تلقائيًا. للتحقق من جميع الحاويات (بما فيها المتوقفة)
استخدم:
docker ps -a



```
ali@ali-VirtualBox:~/Desktop$ docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED              STATUS                          PORTS     NAMES
90cdd68743ec   ubuntu        "/bin/bash"   About a minute ago   Exited (0) About a minute ago             ubuntu02
9a7adacc5cb0   ubuntu        "/bin/bash"   5 minutes ago        Up 5 minutes                              ubuntu01
196064f770ac   hello-world   "/hello"      28 minutes ago       Exited (0) 28 minutes ago                 nostalgic_bardeen
6312cdf4f1d2   hello-world   "/hello"      38 minutes ago       Exited (0) 38 minutes ago                 hardcore_dirac
```

ستجد أن ubuntu02 مدرجة ولكن بحالة Exited، مما يعني أنها توقفت فورًا بعد التشغيل لأنها لم تكن تقوم بتشغيل أي عملية

execute a command inside the container:

إذا كنت تريد تنفيذ أوامر داخل الحاوية دون الدخول إليها، استخدم:

```
docker exec ubuntu01 pwd
docker exec ubuntu01 ls /home
docker exec ubuntu01 touch /home/test.txt
docker exec ubuntu01 ls /home
```

💡 شرح الأوامر:

- `docker exec ubuntu01 pwd` → يعرض المجلد الحالي داخل الحاوية.
- `docker exec ubuntu01 ls /home` → يعرض الملفات داخل `/home`.
- `docker exec ubuntu01 touch /home/test.txt` → ينشئ ملفًا جديدًا باسم `test.txt` داخل `/home`.

- `docker exec ubuntu01 ls /home` → للتحقق من أن الملف تم إنشاؤه.

to enter the shell of the container:

إذا كنت تريد الدخول إلى الحاوية بنفسك والعمل داخلها، استخدم أحد الأوامر التالية

docker exec -it ubuntu01 sh
or
docker attach ubuntu01

الفرق بين الطريقتين:

1. `docker exec -it ubuntu01 sh` → يسمح لك بتشغيل `sh` داخل الحاوية ويمنحك بيئة عمل جديدة.
2. `docker attach ubuntu01` → يربطك بالحاوية **كما لو كنت متصلاً بها منذ البداية**.

إيقاف وإعادة تشغيل الحاوية

لإيقاف الحاوية، استخدم:

docker stop ubuntu01

ثم تحقق مما إذا كانت قد توقفت باستخدام:

docker ps

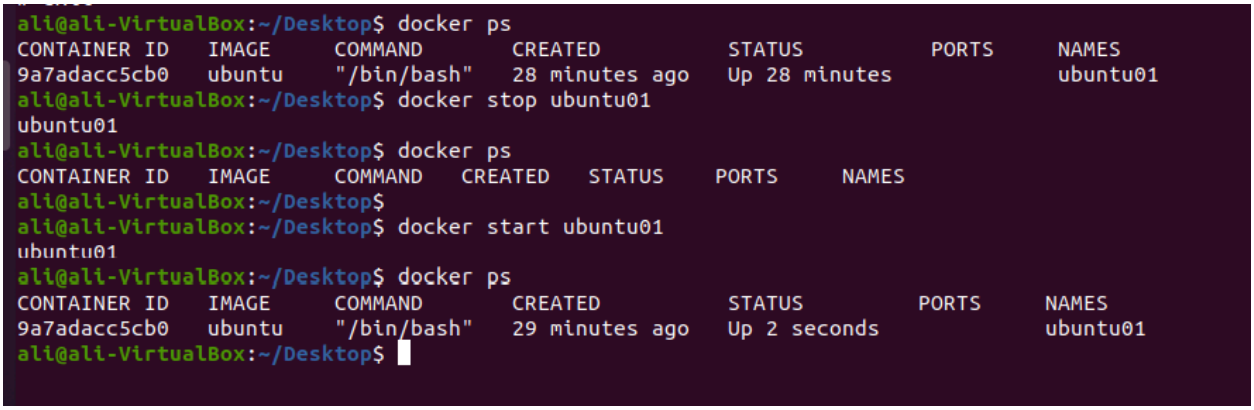
(لن تظهر لأنها توقفت، لكن يمكنك التحقق منها باستخدام `docker ps -a`).

لإعادة تشغيل الحاوية:

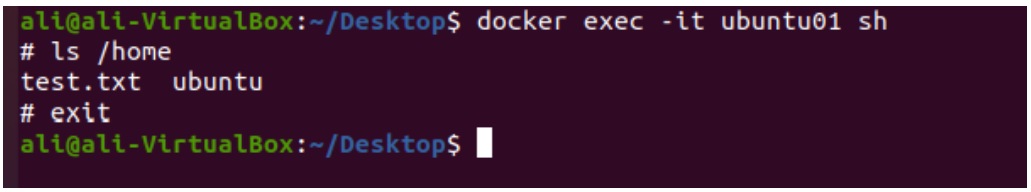
```
docker start ubuntu01
```

ثم تحقق من أنها تعمل مرة أخرى:

```
docker ps
```



After (docker start ubuntu01)
verify that the container is running. Connect to the container shell and check the contents of the home directory. Are your files still there?



Deleting a container:

to delete a container simply use the following set of commands:

First before you delete it, you must stop it.

`docker stop ubuntu01`

`docker rm ubuntu01`

verify that the container has been deleted. Using (`docker ps -a`)

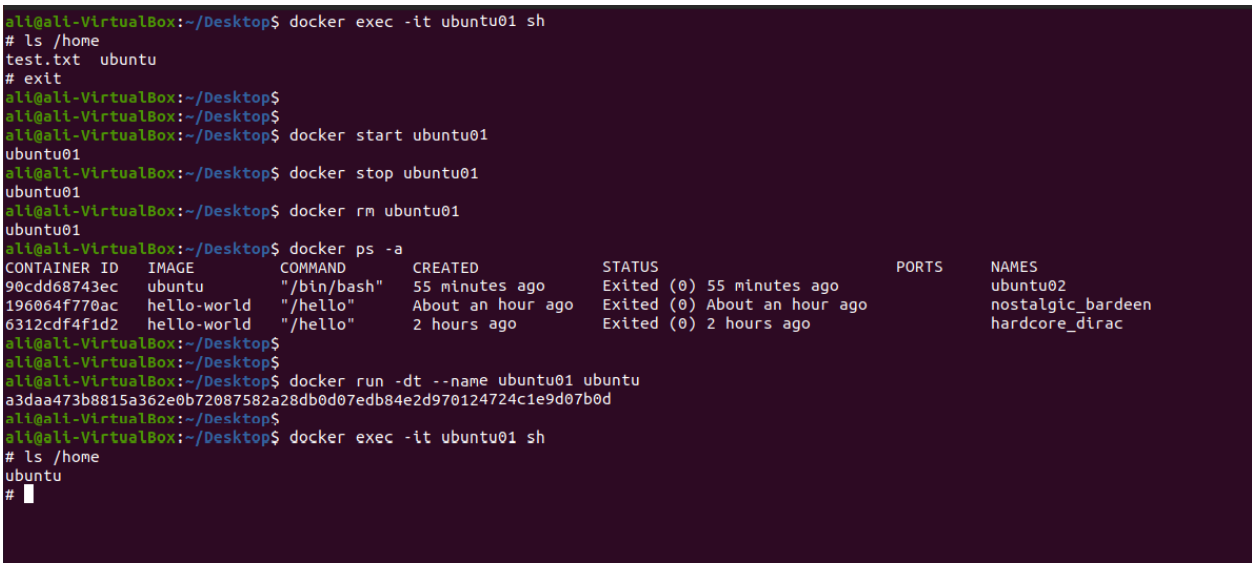
Now recreate the container again with the same name (Using
`docker run -dt --name ubuntu01 ubuntu`)

then attach to its shell and check the contents of the home directory Using:

```
docker exec -it ubuntu01 sh
ls /home
```

Are your files still there ? NO ❌

Explain: The previous files will not be there, because you created a completely new container, not restored the old one.



Persistent Storage Containers:

بشكل افتراضي، الحاويات متطايرة، مما يعني أن أي بيانات مخزنة داخل الحاوية ستُفقد بمجرد حذفها أو إعادة إنشائها.
متطايرة:Volatile

We use persistent storage by linking a folder inside the container to a folder on the host machine.

نستخدم التخزين الدائم عن طريق ربط مجلد داخل الحاوية بمجلد على الجهاز المضيف .

التخزين الدائم : (Persistent Storage) الجهاز المضيف : (Host Machine)

1️⃣ إنشاء مجلد على الجهاز المضيف:

أول خطوة هي إنشاء مجلد على الجهاز المضيف لحفظ البيانات التي سيتم ربطها بالحاوية.

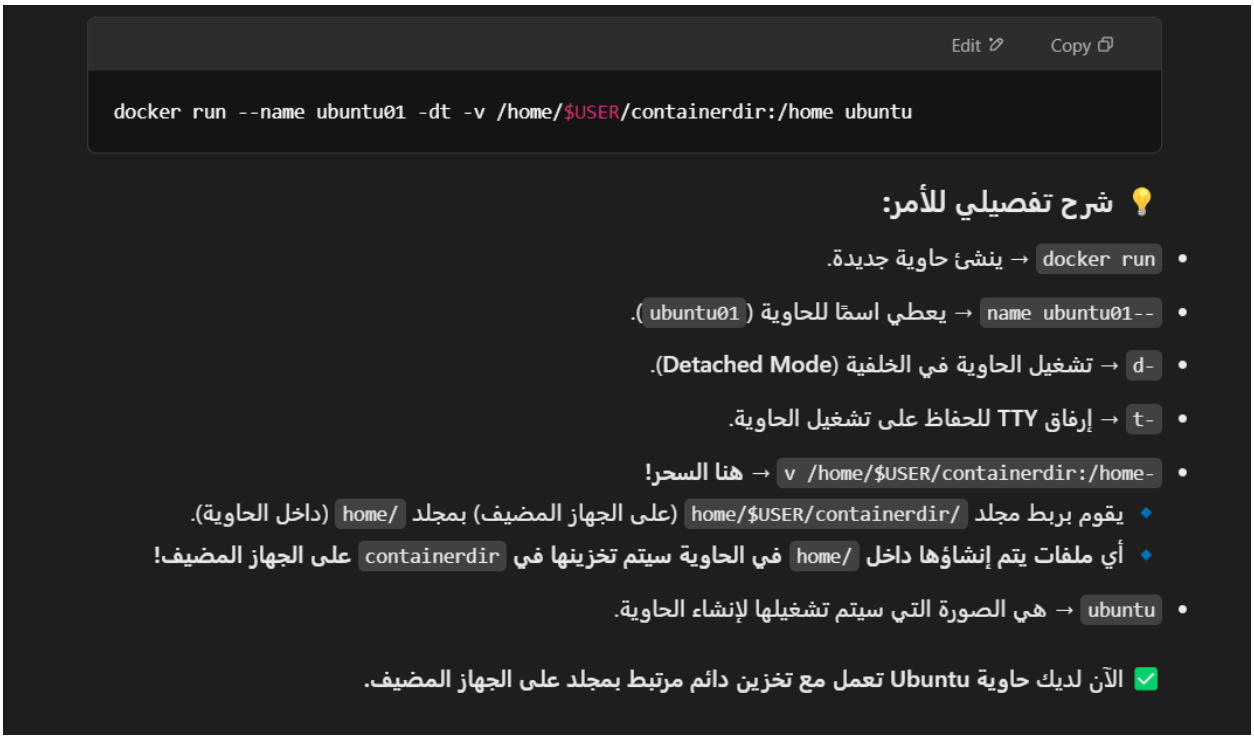
```
mkdir ~/containerdir

comment: ~/containerdir = /home/your-username(ali)/containerdir
```

2️⃣ إنشاء حاوية وربط المجلد معها:

الآن سنقوم بإنشاء حاوية من صورة Ubuntu وربط مجلد containerdir داخل /home داخل الحاوية.

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



3️⃣ الدخول إلى الحاوية:

الآن سندخل إلى الحاوية ونتحقق من الربط بين المجلدات.

docker exec -ti ubuntu01 sh

4️⃣ اختبار التخزين الدائم

touch /home/testfile

Persistent Storage Test
🔷 Create a file inside /home inside the container

touch /home/testfile
What does this command do?

touch /home/testfile → Creates an empty file named testfile inside /home in the containr.
Since /home in the container is linked to the containerdir folder on the host machine, the file should appear in both locations.

5️⃣ الخروج من الحاوية والتحقق من وجود الملف على الجهاز المضيف:

```
ls ~/containerdir
```

Networking:

عندما تقوم بتشغيل تطبيقات داخل **الحاويات** ، فإنها تعمل بشكل **معزول**.

- الحاوية لا تكون متاحة إلا من داخل الجهاز المضيف، **بشكل افتراضي** .

إذا أردت أن يتمكن مستخدمون آخرون من الوصول إلى التطبيق داخل الحاوية، عليك تهيئة الشبكة بشكل صحيح.

If you're running a service inside the container (like a website), by default it cannot be accessed from outside. Therefore, we need to bind the ports.

Basic Installation and Updates inside the Container (
docker exec -it ubuntu01 sh):

```
apt update
apt install curl
apt install -y apache2
```

Running an Apache server inside the container you need to bind it:

```
docker run -dt --name webserver httpd
```
(container = webserver)

Docker checks if the httpd image exists locally.

If it doesn't exist, it will be downloaded from Docker Hub automatically.
A new container is created based on this image.
Apache starts inside the container, but it will only be accessible from within the host machine
. لكنه لن يكون متاحًا إلا من داخل الجهاز المضيف.

```
wajdabdal-hadee@wajdabdal-hadee:~$ 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: sha256:10381816bb7e60ae3a9db3784f2966a8910b6ff07c4da54bd2d62d2671c
8ab6e
Status: Downloaded newer image for httpd:latest
8de4bcc534453839147bada001c08a0e580eaaabcbff5b296e4fa0457317d54c
```

Finding the container's IP:

docker exec -ti webserver sh
hostname -i

```
wajdabdal-hadee@wajdabdal-hadee:~$ docker exec -ti webserver sh
# hostname -i
172.17.0.3
```

The IP Address of the Container

Accessing the website from the host :

Inside the container curl http://container_IP to verify that it's working well.

curl http://container_IP ⟶ curl http://172.17.0.3

this should display the html code of the start page of the server, This address is a host only network address that can be accessed only by the host.

```
# 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

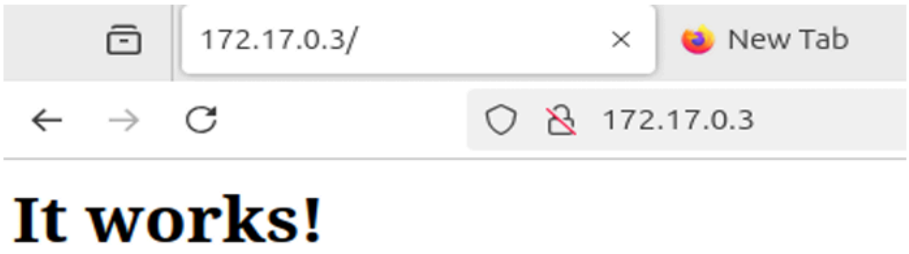


Figure 5.6.4 Verifying by IP in the fire-fox browser.

curl - -version

```
# curl --version
curl 8.5.0 (x86_64-pc-linux-gnu) libcurl/8.5.0 OpenSSL/3.0.13 zlib/1.3 brotli/1.1.0 zstd/1.5.5 libidn2/2.3.7 libpsl/0.21.2 (+libid
n2/2.3.7) libssh/0.10.6/openssl/zlib nghttp2/1.59.0 librtmp/2.3 OpenLDAP/2.6.7
Release-Date: 2023-12-06, security patched: 8.5.0-2ubuntu10.6
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smt
ps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM PSL SPNEGO SSL threadsafe
TLS-SRP UnixSockets zstd
```

Verifying that the Curl has been Installed

```
# service apache2 status
 * apache2 is running
# curl http://172.17.0.2
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
```

The Apache homepage

Making the website accessible to everyone via port mapping:

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

ربط المنفذ 80 في الجهاز المضيف بالمنفذ 80 داخل الحاوية، مما يسمح بالوصول إلى الموقع من المتصفح.

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.

ملاحظات اضافية ممكن تستنتج منها أشياء

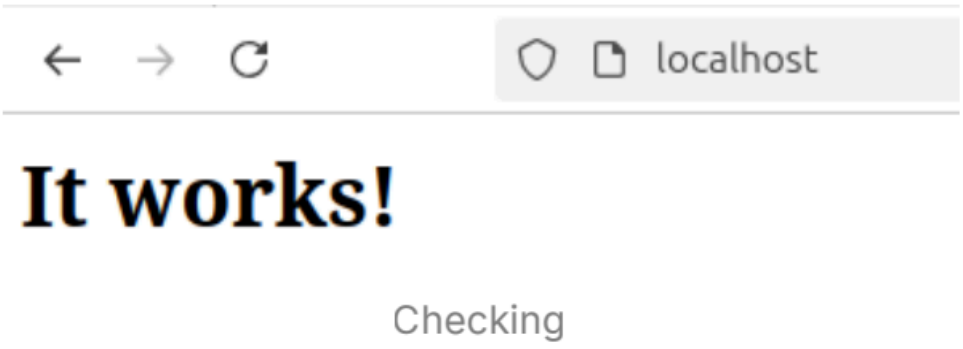
1. Docker يتحقق مما إذا كانت صورة `httpd` موجودة محليًا.
   - إذا لم تكن موجودة، سيتم تنزيلها من Docker Hub تلقائيًا.
2. ينشئ Docker حاوية جديدة بناءً على هذه الصورة.
3. يبدأ تشغيل خادم Apache داخل الحاوية.
4. يتم ربط منفذ 80 في الحاوية بمنفذ 80 في الجهاز المضيف، مما يسمح بالوصول إلى الموقع من أي جهاز على الشبكة.

where httpd is an image on docker hub.
display the running containers and notice the difference in the output.
now open the VM IP address from another machine [your PS host OS]. This should display a web page
that says “It works!”.

```
wajdabdal-hadee@wajdabdal-hadee:~$ docker run -dt -p 80:80 --name webserver httpd
docker: Error response from daemon: Conflict. The container name "/webserver" is already
 in use by container "8de4bcc534453839147bada001c08a0e580eaaabcbff5b296e4fa0457317d54c".
 You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
wajdabdal-hadee@wajdabdal-hadee:~$ docker rename webserver webserver_old
wajdabdal-hadee@wajdabdal-hadee:~$ docker run -dt -p 80:80 --name webserver httpd
4e29d3a91e829042cff359a0d5c2c6f55005cafc9641ec330d226e8d59860aaa
wajdabdal-hadee@wajdabdal-hadee:~$
```

Verifying Apache is running on your system by binds the ports:
If you're using the same device: Open your web browser and type one of the following addresses in the address bar:

http://localhost



or by using: curl http://localhost in container

If everything is working correctly, you should see the HTML text that includes the "It works" page of Apache.

Building your own images:

If you want to create a custom image containing a static website, you can use a Dockerfile.
Write the following commands:

```
mkdir dockerexample
cd dockerexample
```

Creating The Dockerfile:

> 💡 A Dockerfile is a text file that contains instructions for building a Docker image. It defines the necessary steps to create the application's runtime environment, such as:
>
> - Installing libraries
> - Copying application files
> - Setting environment variables
> - Running commands
>
> The image is built using the `docker build` command.
>
> Dockerfiles help create reproducible and easily deployable images.

```
nano Dockerfile
```

and inside it:

```
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"]

```
  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.

السطر	الوظيفة	التفسير
`FROM ubuntu`	تحديد نظام التشغيل الأساسي	يحدد أن الصورة ستعتمد على **Ubuntu** كنظام تشغيل داخل الحاوية.
`RUN apt update`	تحديث الحزم	يقوم بتحديث قائمة الحزم المتاحة لضمان تحميل الإصدارات الأحدث من البرامج.
`RUN apt install -y apache2`	تثبيت Apache	يثبت **Apache HTTP Server**، وهو خادم ويب شهير.
`RUN apt clean`	تنظيف الملفات غير الضرورية	يحذف الملفات المؤقتة بعد التثبيت لتقليل حجم الصورة.
`RUN rm -rf */*/var/lib/apt/lists`	إزالة ملفات القائمة	يحذف ملفات قائمة الحزم التي لم تعد ضرورية بعد التثبيت.
`WORKDIR /var/www/html`	تحديد مجلد العمل	يحدد **المجلد الافتراضي** داخل الحاوية إلى `var/www/html/`، حيث يتم تخزين صفحات الويب.
`COPY index.html index.html`	نسخ ملف `index.html`	ينسخ **ملف** `index.html` **من الجهاز المضيف إلى الحاوية** داخل `/var/www/html/`.
`EXPOSE 80`	فتح منفذ **80**	يعلن أن **الحاوية ستستمع للطلبات على المنفذ 80**، وهو المنفذ الافتراضي لـ Apache.
`CMD ["apachectl", "-D", "FOREGROUND"]`	تشغيل Apache عند بدء الحاوية	يشغل **Apache في وضع التشغيل الأمامي** (`FOREGROUND`) حتى تظل الحاوية قيد التشغيل.

Creating The Website: (An index.html file)

nano index.html

<!doctype html>
<html>
<head>
<title>Docker Images</title>

</head>
<body>

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

This is a custom apache start page from inside a container made with a custom image.

</body>
</html>

Creating The ***Website***:

`nano index.html`

and inside it:

```
<!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 made with a custom image.</p>
</body>
</html>
```

using the code:

docker build -t mysrvrimg .

Running a container with the custom image:

docker run -dit -p 80:81 --name mysrvr01 mysrvrimg (we change the port number)

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:

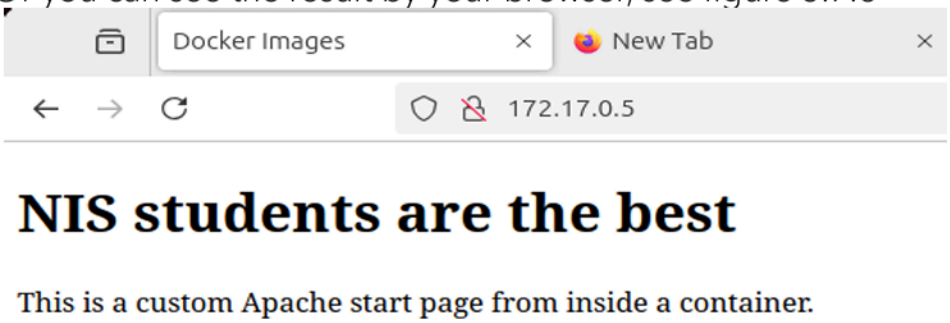
curl http://[VM_IP]

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



5.8 TO DO:

Create 2 containers with the following specs:

1. Container 1 should be a web server. The website must be persistent with the index.html page

containing the names of your group members and must be accessible from other machines.

1. Container 2 based on httpd with all default settings. This also should be accessible from all other machines.

```
docker run -dit --name webserver2 -p 9090:80 httpd
```

كيف تتأكد إنها شغالة؟

1. شغلي الحاوية بالأمر اللي فوق.
2. افتحي متصفح على أي جهاز بالشبكة (جهازك أو غيره).
3. اكتبي في العنوان:

```
http://[IP-VM]:9090
```

🧱 الحاوية الثانية – Container 2

📌 المطلوب:

> Container 2 based on `httpd` with all default settings. This also should be accessible from all other machines".

يعني:

✅ المطلوب منها:

1. تستخدم صورة اسمها `httpd` ← هذه صورة جاهزة من Docker Hub تحتوي على خادم Apache.
2. ما بدنا نضيف أو نغيّر فيها إشي (no custom files or changes).
3. لازم تكون متاحة لأي جهاز بالشبكة (يعني تفتحيها من المتصفح على أي جهاز ثاني).

👇 طيب شو يعني صورة `httpd`؟

- `httpd` هي صورة رسمية من Docker Hub.
- تحتوي على **خادم Apache Web Server جاهز**.
- لما تشغليها، هي تلقائيًا تعرض صفحة ترحيب بـ Apache فيها "It works!".

🛠️ كيف تشغليها؟

Edit Copy

```
docker run -dit --name webserver2 -p 9090:80 httpd
```

شرح الأمر:

الجزء	شو بعمل
`docker run`	تشغيل حاوية جديدة
`dit-`	تشغيل بالخلفية وبشكل تفاعلي
`name webserver2--`	اسم الحاوية: webserver2
`p 9090:80-`	ربط منفذ 80 من داخل الحاوية بمنفذ 9090 من جهازك
`httpd`	استخدام صورة `httpd` من Docker Hub

شرح 9090:80 بالتفصيل

✅ شو يعني 9090:80 ؟

الجزء	المعنى البسيط
`80`	المنفذ داخل الحاوية (container) – Apache بيشتغل عليه تلقائيًا
`9090`	المنفذ على جهازك أو الـ VM (host) اللي الناس من برّا رح يتصلوا فيه

🪢 التشبيه:

تخيلي الحاوية (container) زي بيت داخلي، وعنده باب اسمه 80.

بس هذا البيت جوّا عمارة (الـ VM أو الجهاز الحقيقي)، والناس برا ما بيعرفوا أبواب البيت... فإنتي فتحتي باب خارجي (رقمه 9090) وقلتي:

> **"أي حدا يدخل من الباب 9090، دخّلوه على الباب 80 جوّا البيت."**

🧱 كيف بيشتغل على أرض الواقع؟

1. Apache داخل الحاوية بيشتغل على البورت 80.
2. بس لو حدا حاول يدخل مباشرة على 80 من برّا، ما رح يقدر لأنه جوّا الحاوية.
3. لذلك مربوط 9090 ← 80 باستخدام:

```
-p 9090:80
```

4. هيك أي جهاز يفتح:

```
http://[IP الجهاز]:9090
```

رح يوصل للـ Apache داخل الحاوية على البورت 80.

🎯 لو استخدمنا `-p 80:80` بدل `9090:80`؟

يعني:

- **المنفذ 80** في الحاوية مربوط على **المنفذ 80** في الجهاز.
- فالدخول بيكون من:

```
http://[IP]:80
```

بس إذا عندك أكتر من سيرفر، أو النظام عنده خدمة ثانية على 80، راح يصير تعارض، لذلك بنستخدم 9090 أو أي رقم ثاني.

📌 خلاصة:

عنصر	مين هو	شو بعمل
`80`	داخل الحاوية	المنفذ اللي شغال عليه Apache
`9090`	خارج الحاوية (host)	اللي المستخدمين يتواصلوا عليه