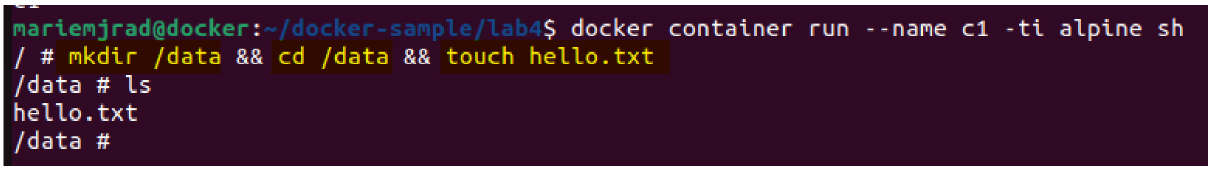
1.Understanding Data persistency without a volume

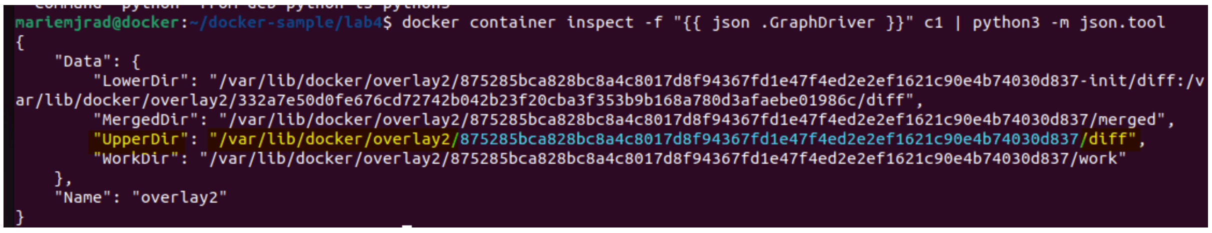
- We will first illustrate how data is not persisted outside of a container by default. Let's run an interactive shell within an alpine container named c1.



```
mariemjrad@docker:~/docker-sample/lab4$ docker container run --name c1 -ti alpine sh
/ # mkdir /data && cd /data && touch hello.txt
/data # ls
hello.txt
/data #
```

⇒Created a container and created a directory within it and a file named hello.txt
⇒Explanation : when we create a container a whole container layer is created with read-write access

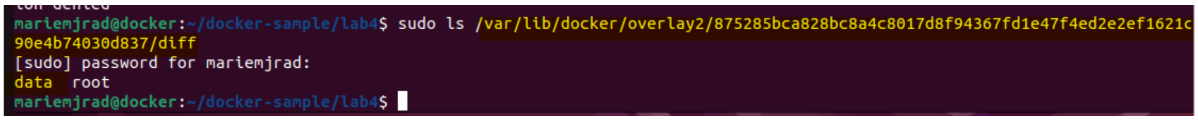
- after Existing the container we run the one of the following commands:
  - `docker container inspect c1`
  - or : `docker container inspect -f "{{ json .Mounts }}" c1 | python3 -m json.tool`



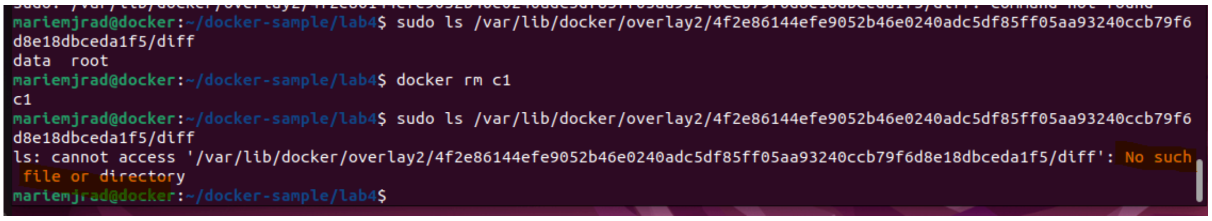
```
mariemjrad@docker:~/docker-sample/lab4$ docker container inspect -f "{{ json .GraphDriver }}" c1 | python3 -m json.tool
{
    "Data": {
        "LowerDir": "/var/lib/docker/overlay2/875285bca828bc8a4c8017d8f94367fd1e47f4ed2e2ef1621c90e4b74030d837-init/diff:/v
ar/lib/docker/overlay2/332a7e50d0fe676cd72742b042b23f20cba3f353b9b168a780d3afaebe01986c/diff",
        "MergedDir": "/var/lib/docker/overlay2/875285bca828bc8a4c8017d8f94367fd1e47f4ed2e2ef1621c90e4b74030d837/merged",
        "UpperDir": "/var/lib/docker/overlay2/875285bca828bc8a4c8017d8f94367fd1e47f4ed2e2ef1621c90e4b74030d837/diff",
        "WorkDir": "/var/lib/docker/overlay2/875285bca828bc8a4c8017d8f94367fd1e47f4ed2e2ef1621c90e4b74030d837/work"
    },
    "Name": "overlay2"
}
```

- The highlighted directory UpperDir is the specified directory that contains the directory Data created in c1 container

  case 1



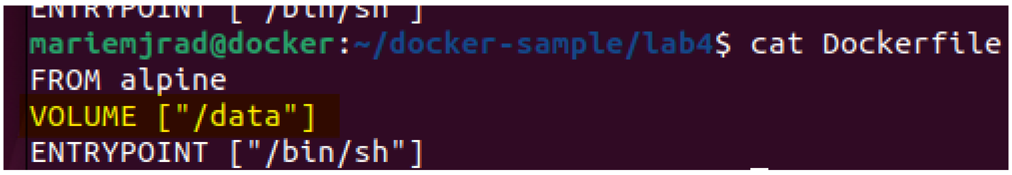
```
mariemjrad@docker:~/docker-sample/lab4$ sudo ls /var/lib/docker/overlay2/875285bca828bc8a4c8017d8f94367fd1e47f4ed2e2ef1621c
90e4b74030d837/diff
[sudo] password for mariemjrad:
data  root
mariemjrad@docker:~/docker-sample/lab4$
```



```
mariemjrad@docker:~/docker-sample/lab4$ sudo ls /var/lib/docker/overlay2/4f2e86144efe9052b46e0240adc5df85ff05aa93240ccb79f6
d8e18dbceda1f5/diff
data  root
mariemjrad@docker:~/docker-sample/lab4$ docker rm c1
c1
mariemjrad@docker:~/docker-sample/lab4$ sudo ls /var/lib/docker/overlay2/4f2e86144efe9052b46e0240adc5df85ff05aa93240ccb79f6
d8e18dbceda1f5/diff
ls: cannot access '/var/lib/docker/overlay2/4f2e86144efe9052b46e0240adc5df85ff05aa93240ccb79f6d8e18dbceda1f5/diff': No such
 file or directory
mariemjrad@docker:~/docker-sample/lab4$
```

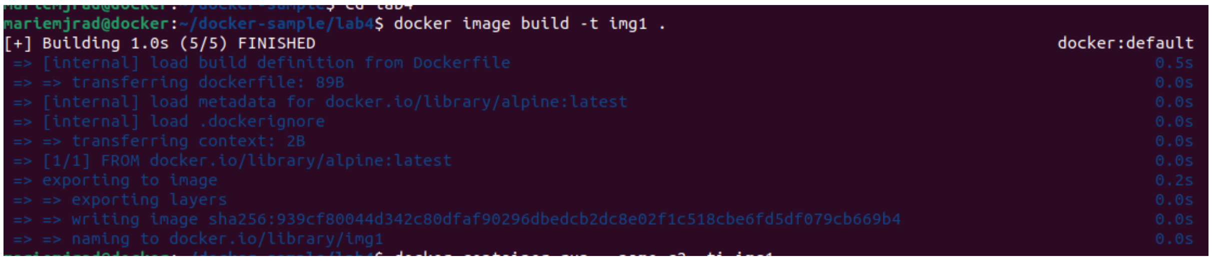
- when we delete the container the upperDir does not exist anymore so if the container is damaged or anything else nothing would be persisted

2.Docker volumes in a Dockerfile



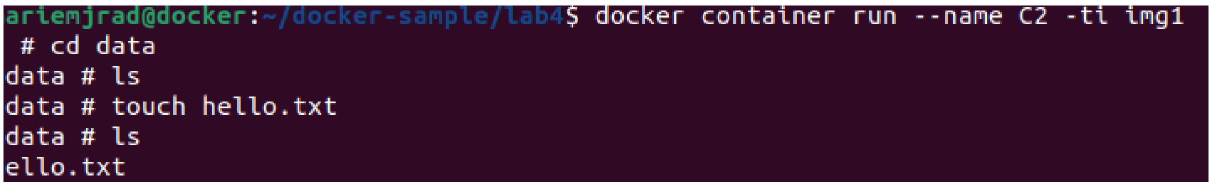
```
mariemjrad@docker:~/docker-sample/lab4$ cat Dockerfile
FROM alpine
VOLUME ["/data"]
ENTRYPOINT ["/bin/sh"]
```

- created a dockerfile where we specify **VOLUME:**
  - **The VOLUME ["/data"]** instruction marks /data as a volume, meaning any files stored there persist beyond the container's lifecycle
  - When a container is created from this image, Docker automatically creates an anonymous volume at /data, unless a specific volume is mounted at runtime which is called named volume or user-specified



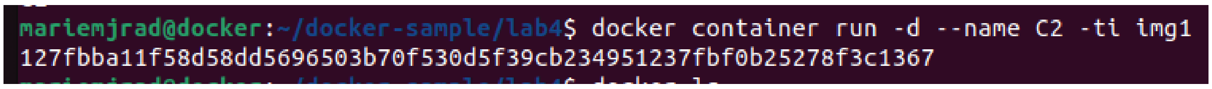
```
mariemjrad@docker:~/docker-sample/lab4$ docker image build -t img1 .
[+] Building 1.0s (5/5) FINISHED                                                               docker:default
 => [internal] load build definition from Dockerfile                                                      0.5s
 => => transferring dockerfile: 89B                                                                       0.0s
 => [internal] load metadata for docker.io/library/alpine:latest                                          0.0s
 => [internal] load .dockerignore                                                                         0.0s
 => => transferring context: 2B                                                                           0.0s
 => [1/1] FROM docker.io/library/alpine:latest                                                            0.0s
 => exporting to image                                                                                    0.2s
 => => exporting layers                                                                                   0.0s
 => => writing image sha256:939cf80044d342c80dfaf90296dbedcb2dc8e02f1c518cbe6fd5df079cb669b4              0.0s
 => => naming to docker.io/library/img1                                                                   0.0s
```

then run a container named C2 with the img1 recently created: **docker container run --name c2 -ti img1**



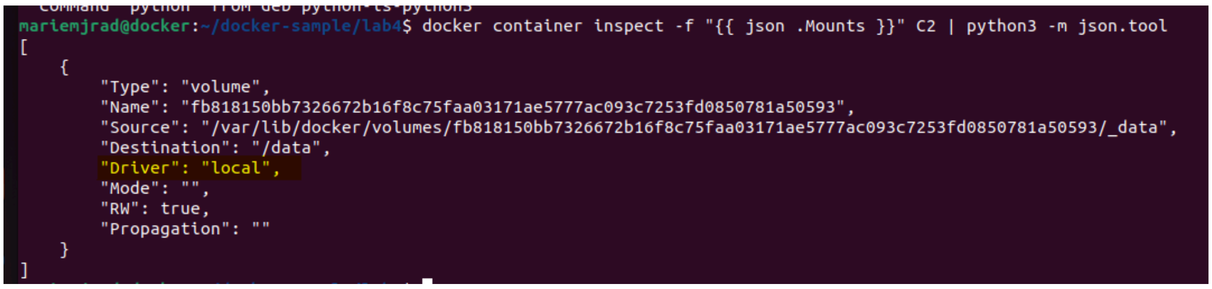
```
ariemjrad@docker:~/docker-sample/lab4$ docker container run --name C2 -ti img1
 # cd data
data # ls
data # touch hello.txt
data # ls
ello.txt
```

- once the container is running we could see that it is already built in with a directory /data, which is the mont directory defined in the dockerfile that means somewhere on the host the directory /data is stored , afterwards we created a file hello.txt



```
mariemjrad@docker:~/docker-sample/lab4$ docker container run -d --name C2 -ti img1
127fbba11f58d58dd5696503b70f530d5f39cb234951237fbf0b25278f3c1367
```

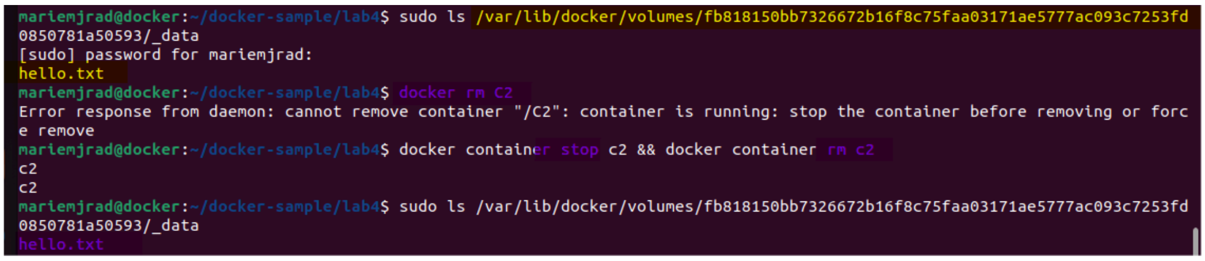
- here i tried to run it with the option -d in order to run it in the background



```
mariemjrad@docker:~/docker-sample/lab4$ docker container inspect -f "{{ json .Mounts }}" C2 | python3 -m json.tool
[
    {
        "Type": "volume",
        "Name": "fb818150bb7326672b16f8c75faa03171ae5777ac093c7253fd0850781a50593",
        "Source": "/var/lib/docker/volumes/fb818150bb7326672b16f8c75faa03171ae5777ac093c7253fd0850781a50593/_data",
        "Destination": "/data",
        "Driver": "local",
        "Mode": "",
        "RW": true,
        "Propagation": ""
    }
]
```

```
docker container inspect -f "{{ json .Mounts }}" c1 | python3 -m
json.tool
```

⇒ The output of the command above shows that the volume defined in /data is stored in **/var/lib/docker/volumes/fb818…593/_data** on the host



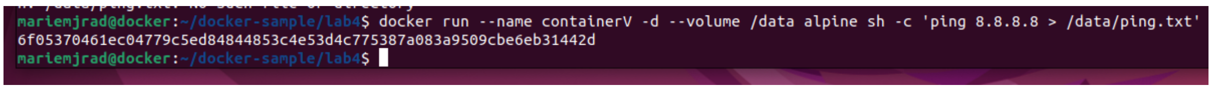
```
mariemjrad@docker:~/docker-sample/lab4$ sudo ls /var/lib/docker/volumes/fb818150bb7326672b16f8c75faa03171ae5777ac093c7253fd
0850781a50593/_data
[sudo] password for mariemjrad:
hello.txt
mariemjrad@docker:~/docker-sample/lab4$ docker rm C2
Error response from daemon: cannot remove container "/C2": container is running: stop the container before removing or forc
e remove
mariemjrad@docker:~/docker-sample/lab4$ docker container stop c2 && docker container rm c2
c2
c2
mariemjrad@docker:~/docker-sample/lab4$ sudo ls /var/lib/docker/volumes/fb818150bb7326672b16f8c75faa03171ae5777ac093c7253fd
0850781a50593/_data
hello.txt
```

- Even after deleting the container the hello file remained unlike the previous case 1

3.Defining a volume at runtime

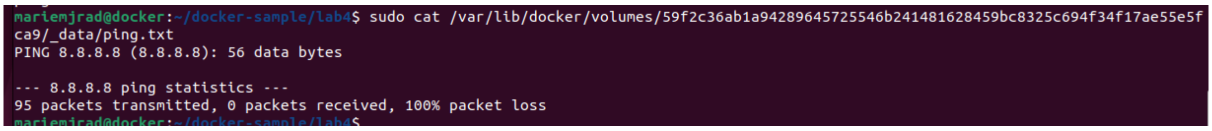
In this part we will explore the named volumes because earlier we have seen the anonymous volumes where we don't specify the volume name on the host

```
docker container run --name containerV -d -v /data alpine sh -c 'ping
8.8.8.8 > /data/ping.txt'
```



```
mariemjrad@docker:~/docker-sample/lab4$ docker run --name containerV -d --volume /data alpine sh -c 'ping 8.8.8.8 > /data/ping.txt'
6f05370461ec04779c5ed84844853c4e53d4c775387a083a9509cbe6eb31442d
mariemjrad@docker:~/docker-sample/lab4$
```

- this one has saved the ping output to file ping.txt under /data directory



```
mariemjrad@docker:~/docker-sample/lab4$ sudo cat /var/lib/docker/volumes/59f2c36ab1a94289645725546b241481628459bc8325c694f34f17ae55e5f
ca9/_data/ping.txt
PING 8.8.8.8 (8.8.8.8): 56 data bytes

--- 8.8.8.8 ping statistics ---
95 packets transmitted, 0 packets received, 100% packet loss
mariemjrad@docker:~/docker-sample/lab4$
```

- We inspected the volume and it contains the ping tracing (unfortunately the ping is not working here 😢)

⇒So to resume we used in the third section the anonymous volume by providing the volume argument and specifying the mount directory in this case it is :

3.1.Anonymous Volume

```
docker run -it --name container_anonymous -v /data alpine
```

- Anonymous volumes are automatically created by Docker when using -v /data without specifying a volume name.
- They persist as long as the container exists but are deleted when the container is removed with docker rm -v.
- Unlike named volumes, they are harder to manage because they are assigned random names.
- They are useful for temporary storage inside containers.

3.2.Named Volumes

- create a docker volume

```
docker volume create mon_volume
```

- Lancer un conteneur avec ce volume:

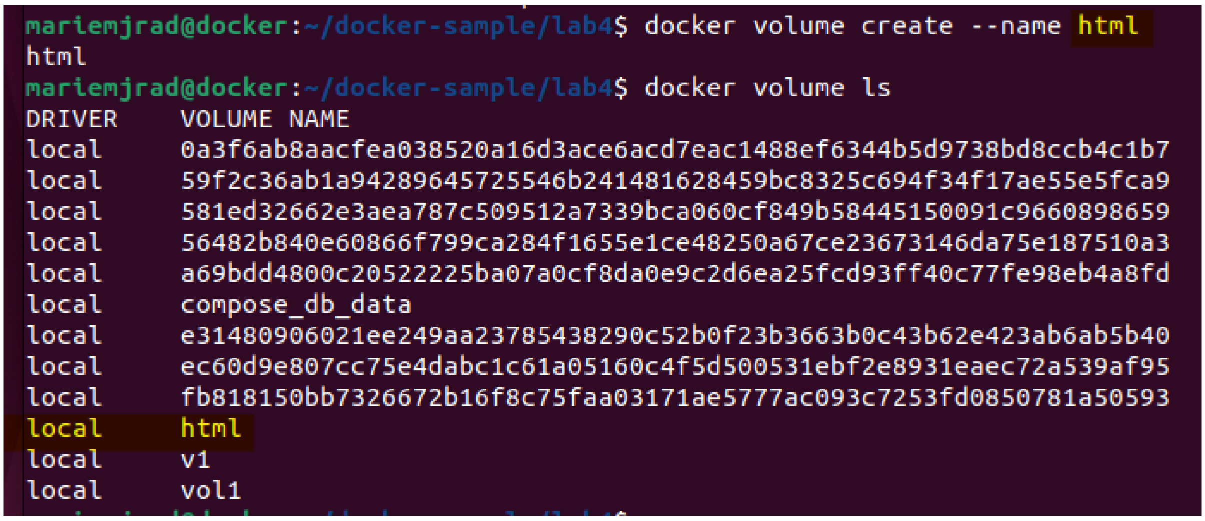
```
docker run -it --name container_named -v mon_volume:/data alpine
```

⇒A **named volume** in Docker is a type of persistent storage that is explicitly created with a specific name and can be easily reused across multiple containers. Named volumes are managed by Docker and are useful when you need to store data that persists across container restarts and can be shared between containers.

⚠️ **Do not confuse:**

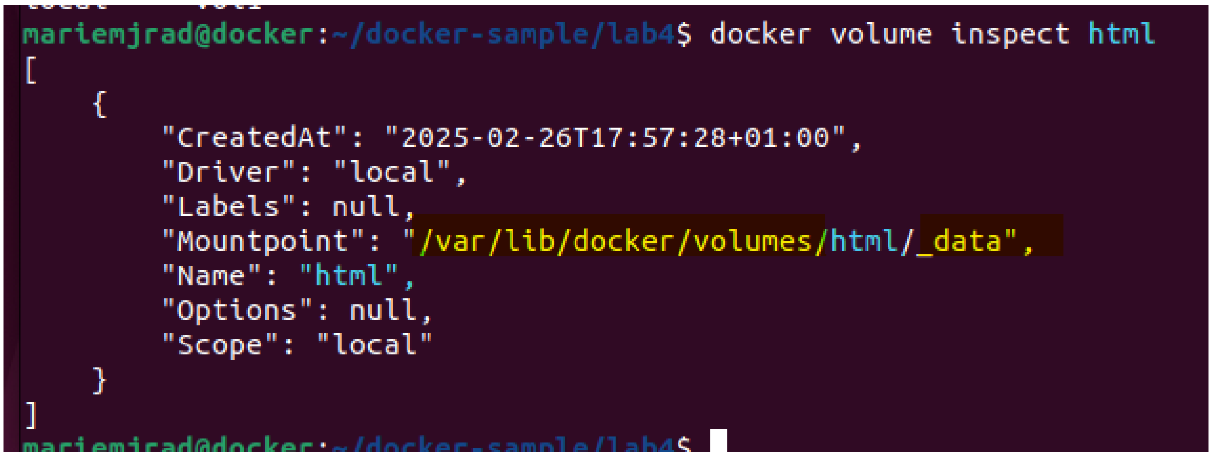
- docker volume create : creates a named volume independently.
- docker run -v : mounts this volume into a specific container, and if the volume does not exist, it is created at the time of execution.

4.Usage of the Volume API



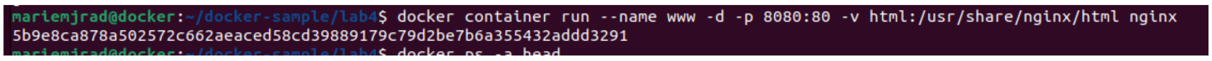
```
mariemjrad@docker:~/docker-sample/lab4$ docker volume create --name html
html
mariemjrad@docker:~/docker-sample/lab4$ docker volume ls
DRIVER    VOLUME NAME
local     0a3f6ab8aacfea038520a16d3ace6acd7eac1488ef6344b5d9738bd8ccb4c1b7
local     59f2c36ab1a94289645725546b241481628459bc8325c694f34f17ae55e5fca9
local     581ed32662e3aea787c509512a7339bca060cf849b58445150091c9660898659
local     56482b840e60866f799ca284f1655e1ce48250a67ce23673146da75e187510a3
local     a69bdd4800c20522225ba07a0cf8da0e9c2d6ea25fcd93ff40c77fe98eb4a8fd
local     compose_db_data
local     e31480906021ee249aa23785438290c52b0f23b3663b0c43b62e423ab6ab5b40
local     ec60d9e807cc75e4dabc1c61a05160c4f5d500531ebf2e8931eaec72a539af95
local     fb818150bb7326672b16f8c75faa03171ae5777ac093c7253fd0850781a50593
local     html
local     v1
local     vol1
```

- created a volume named html.



```
mariemjrad@docker:~/docker-sample/lab4$ docker volume inspect html
[
    {
        "CreatedAt": "2025-02-26T17:57:28+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/html/_data",
        "Name": "html",
        "Options": null,
        "Scope": "local"
    }
]
mariemjrad@docker:~/docker-sample/lab4$
```

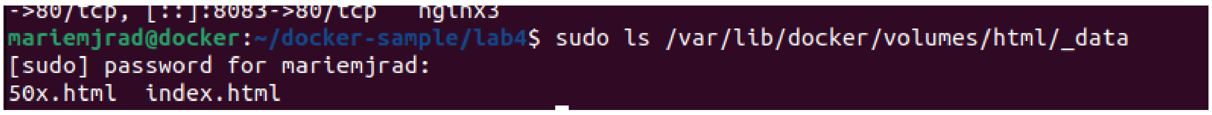
- The MountPoint defined here is the path on the Docker host where the volume can be accessed. We can note that this path uses the name of the volume instead of the autogenerated ID we saw in the example above.



```
mariemjrad@docker:~/docker-sample/lab4$ docker container run --name www -d -p 8080:80 -v html:/usr/share/nginx/html nginx
5b9e8ca878a502572c662aeaced58cd39889179c79d2be7b6a355432addd3291
mariemjrad@docker:~/docker-sample/lab4$ docker ps -a head
```

```
docker container run --name www -d -p 8080:80 -v
html:/usr/share/nginx/html nginx
```

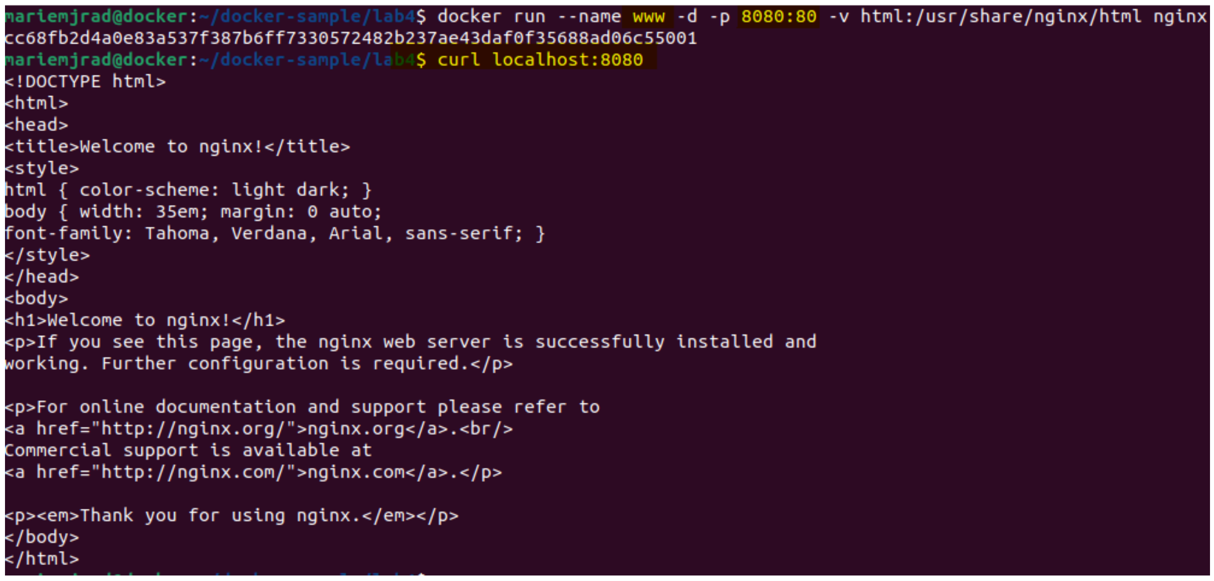
- Use a Nginx image and mount the html volume onto `/usr/share/nginx/html` folder within the container.
- Note: `/usr/share/nginx/html` is the default folder served by nginx. It contains 2 files: index.html and 50x.html
- The -p option to map the nginx default port (80) to a port on the host (8080)



```
->80/tcp, [::]:8083->80/tcp   nginx3
mariemjrad@docker:~/docker-sample/lab4$ sudo ls /var/lib/docker/volumes/html/_data
[sudo] password for mariemjrad:
50x.html  index.html
```

- The content of the `/usr/share/nginx/html` folder of the www container has been copied into the `/var/lib/docker/volumes/html/_data` folder on the host.

In order to browse the nginx server we will use host's port 8080 because it is mapped with container's port 80 that it is hosting nginx

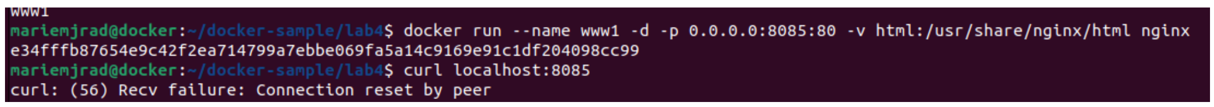


```
mariemjrad@docker:~/docker-sample/lab4$ docker run --name www -d -p 8080:80 -v html:/usr/share/nginx/html nginx
cc68fb2d4a0e83a537f387b6ff7330572482b237ae43daf0f35688ad06c55001
mariemjrad@docker:~/docker-sample/lab4$ curl localhost:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Note: it did not work earlier i had this error so i tried to run with a different port and a new container



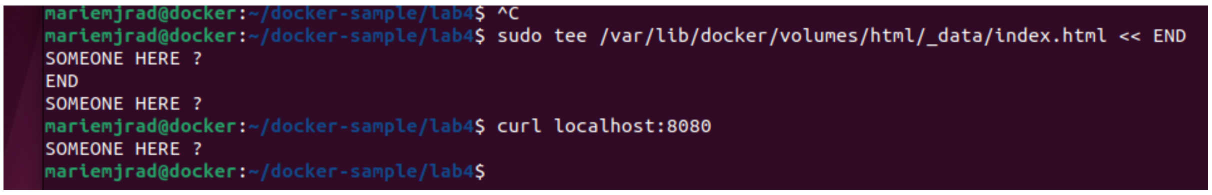
```
www1
mariemjrad@docker:~/docker-sample/lab4$ docker run --name www1 -d -p 0.0.0.0:8085:80 -v html:/usr/share/nginx/html nginx
e34fffb87654e9c42f2ea714799a7ebbe069fa5a14c9169e91c1df204098cc99
mariemjrad@docker:~/docker-sample/lab4$ curl localhost:8085
curl: (56) Recv failure: Connection reset by peer
```

But still not working

```
docker network inspect bridge
```

⇒ lists the containers using bridge network and www container
⇒ inside the container the nginx server works
So the problem was over the host , i checked to see if the port is blocked and it was not so i restarted docker and delete all the unused networks

```
sudo systemctl restart docker
docker network prune -f
docker stop www && docker rm www
docker run --name www -d -p 8085:80 -v html:/usr/share/nginx/html nginx
```

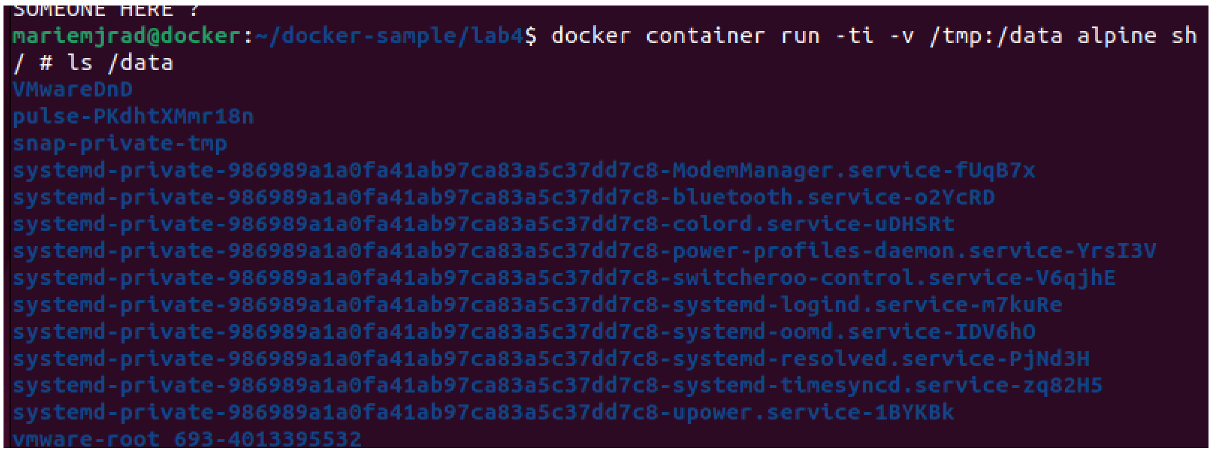


```
mariemjrad@docker:~/docker-sample/lab4$ ^C
mariemjrad@docker:~/docker-sample/lab4$ sudo tee /var/lib/docker/volumes/html/_data/index.html << END
SOMEONE HERE ?
END
SOMEONE HERE ?
mariemjrad@docker:~/docker-sample/lab4$ curl localhost:8080
SOMEONE HERE ?
mariemjrad@docker:~/docker-sample/lab4$
```

- From our host, i modified the server web page and it worked

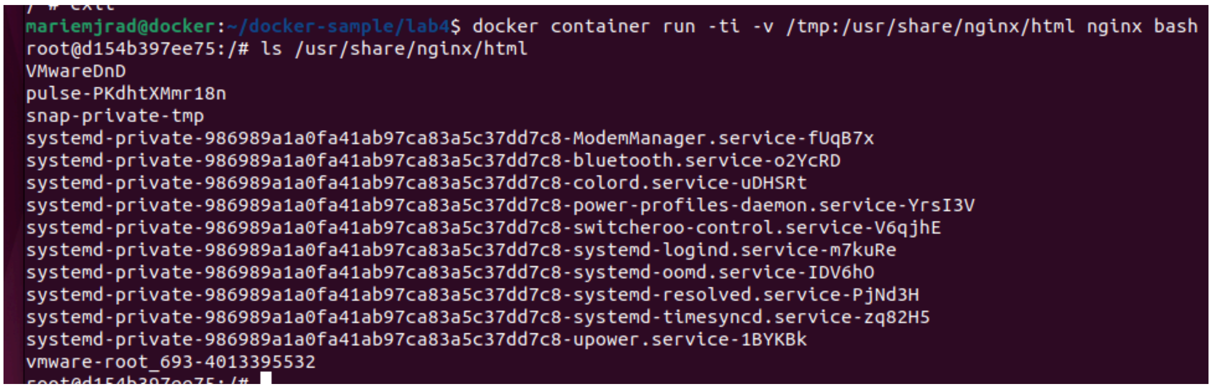
5.Mount host’s folder into a container

- Bind mounts allow your container to access **host files and folders** in real-time.
- If the container path **does not exist**, Docker creates it.
- If the container path **already exists**, the original content is **hidden** by the bind mount.



```
SOMEONE HERE ?
mariemjrad@docker:~/docker-sample/lab4$ docker container run -ti -v /tmp:/data alpine sh
/ # ls /data
VMwareDnD
pulse-PKdhtXMmr18n
snap-private-tmp
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-ModemManager.service-fUqB7x
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-bluetooth.service-o2YcRD
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-colord.service-uDHSRt
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-power-profiles-daemon.service-YrsI3V
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-switcheroo-control.service-V6qjhE
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-systemd-logind.service-m7kuRe
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-systemd-oomd.service-IDV6hO
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-systemd-resolved.service-PjNd3H
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-systemd-timesyncd.service-zq82H5
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-upower.service-1BYKBk
vmware-root_693-4013395532
```

- The /data folder has been created inside the container and it contains the content of the /tmp folder of the host



```
/ # exit
mariemjrad@docker:~/docker-sample/lab4$ docker container run -ti -v /tmp:/usr/share/nginx/html nginx bash
root@d154b397ee75:/# ls /usr/share/nginx/html
VMwareDnD
pulse-PKdhtXMmr18n
snap-private-tmp
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-ModemManager.service-fUqB7x
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-bluetooth.service-o2YcRD
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-colord.service-uDHSRt
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-power-profiles-daemon.service-YrsI3V
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-switcheroo-control.service-V6qjhE
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-systemd-logind.service-m7kuRe
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-systemd-oomd.service-IDV6hO
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-systemd-resolved.service-PjNd3H
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-systemd-timesyncd.service-zq82H5
systemd-private-986989a1a0fa41ab97ca83a5c37dd7c8-upower.service-1BYKBk
vmware-root_693-4013395532
root@d154b397ee75:/#
```

- we specified an already existing directory which contains the nginx server files which is supposed to index.html and 50x.html so that trick should be considered when dealing with bind mounts because in this case it has overridden the existing content

```
docker run -d --name container2 --volumes-from container1 nginx
```

⇒ Create a volume to a container from an another container