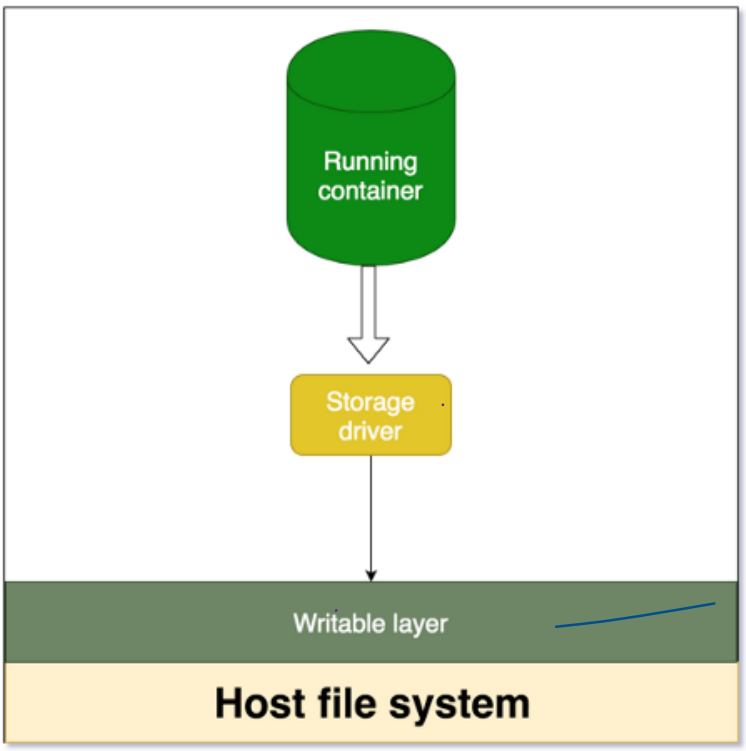
Docker volumes

09:02 AM

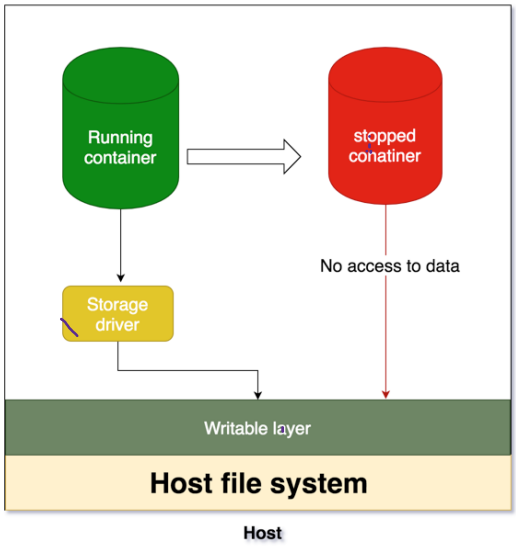
Before going deep into volumes, Let's understand how containers persist data in the host filesystem.



If we look at the above diagram, whenever running container wants to persist data, it actually put that data into the writable layer through storage driver. well, we have some problems with that!!!

What are the problems

- Data is no longer persisted and difficult to access if container stops as shown in the following diagram
- As we can see writable layer is tightly coupled with host filesystem and difficult to move the data.
- We have an extra layer of abstraction with a storage driver which reduces the performance.



Let's see in action

Let's pull the latest nginx image from the docker hub and run the container and load the home page which listens on port 80.

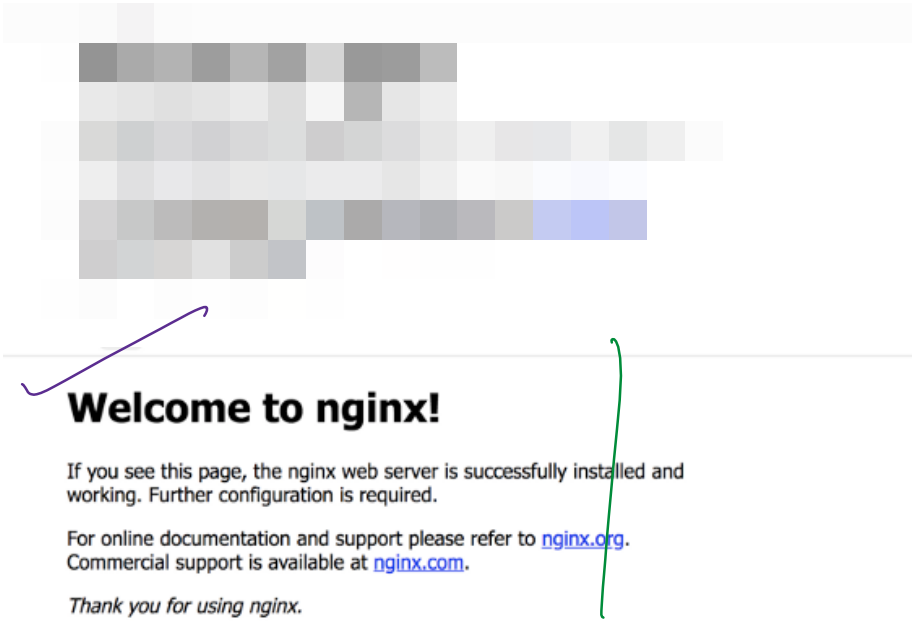
// pull the nginx image

docker pull nginx// run the container

docker run -it --name=webApp -d -p 80:80 nginx

```
Bhargavs-MacBook-Pro:Desktop bhargavbachina$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
6ae821421a7d: Pull complete
da4474e5966c: Pull complete
eb2aec2b9c9f: Pull complete
Digest: sha256:dd2d0ac3fff2f007d99e033b64854be0941e19a2ad51f174d9240dda28d9f534
Status: Downloaded newer image for nginx:latest
Bhargavs-MacBook-Pro:Desktop bhargavbachina$ docker run -it --name=webApp -d -p 80:80 nginx
0eaeb2162d0a8f6850a630185453771b5d4ccdeac1c0aa27c85258c01e26d75c
Bhargavs-MacBook-Pro:Desktop bhargavbachina$
```

docker container



localhost:80

Let's use the ***docker exec*** command to edit the welcome page and load it.

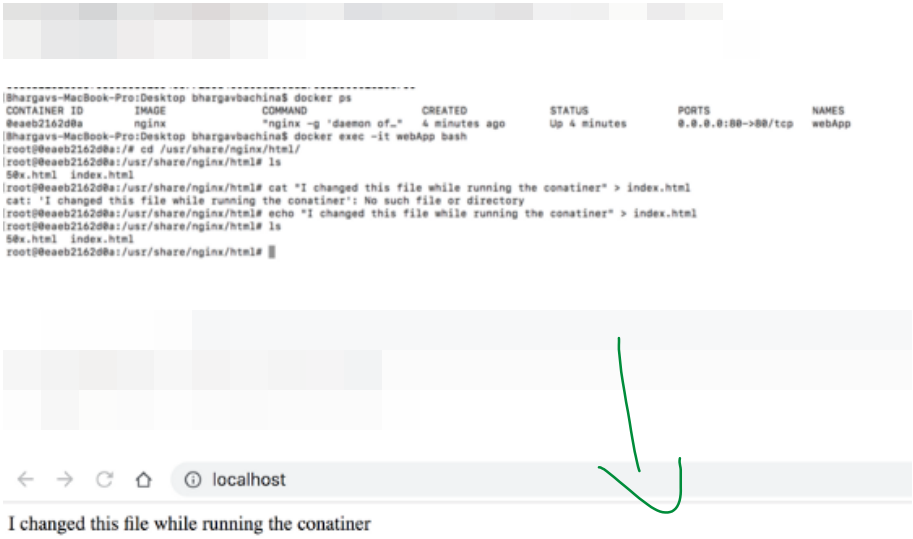
// list the running containers

docker ps// exec command

docker exec -it webApp bash// cd to welcome page and edit it

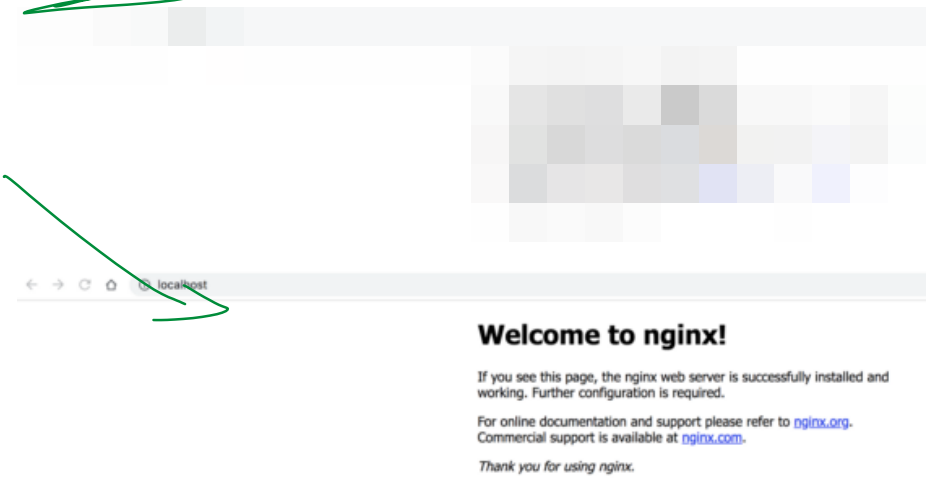
cd /usr/share/nginx/html

echo "I changed this file while running the conatiner" > index.html



localhost:80

Let's stop the container and start it again. we can still see the changes that we made. what if we stop this container and start another one and load the page. There is no way that we could access the file that we have changed in another container.



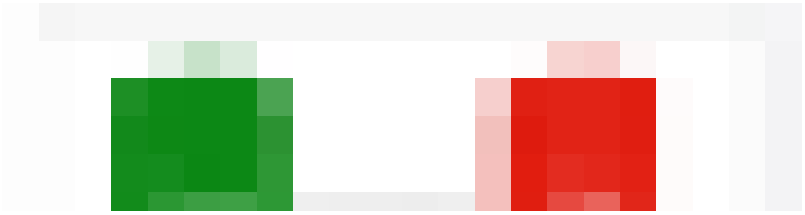
localhost:80

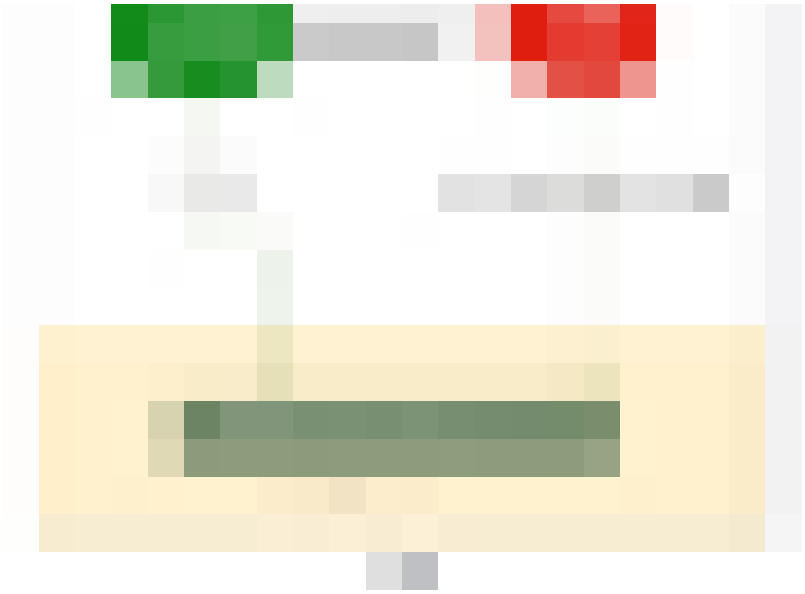
docker run command

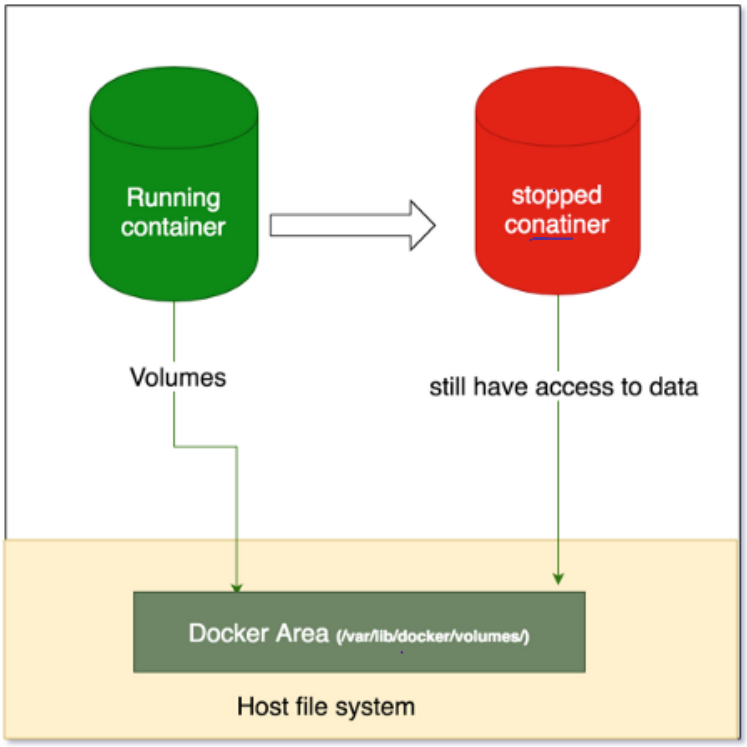
How Volumes can solve above issues

Volumes are saved in the host filesystem ***(/var/lib/docker/volumes/)*** which is owned and maintained by docker.

Any other nondocker process can't access it. But, As depicted in the below other docker processes/containers can still access the data even container is stopped since it is isolated from the container file system.







How to create a Volume

We can create a volume with the below command or while container/service creation

it is created in the directory of the docker host and When you mount the volume into a container, this directory is what is mounted into the container

. we should notice the difference between creation and mounting.

docker volume create <volumeName>

How to remove a Volume

docker volume prune

Let's put Theory into practice

Let's run these commands and see how it works!!. we can see the location of volumes in the docker area of the host file system with the inspect command.

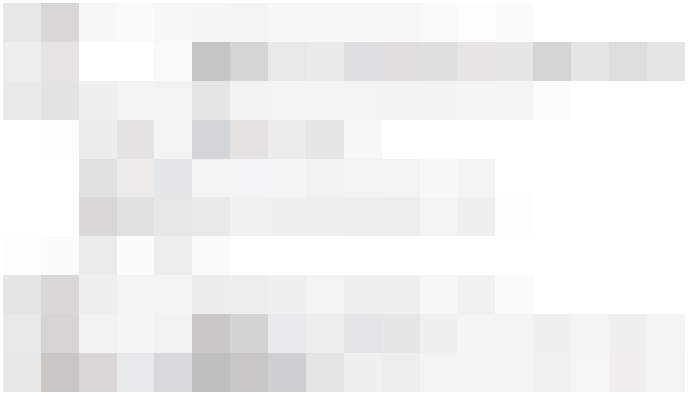
// create a volume

docker volume create new_vol// list volumes

docker volume ls// inspect volumes

docker volume inspect new_vol// removing volumes

docker volume rm new_vol

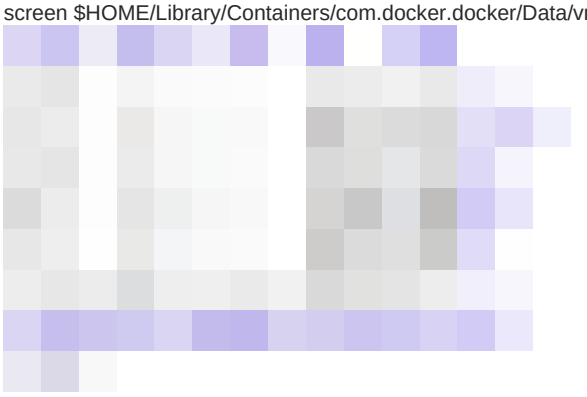


```
Bhargavs-MacBook-Pro:~ bhargavbachina$ docker volume create new_vol
new_vol
Bhargavs-MacBook-Pro:~ bhargavbachina$ docker volume ls
DRIVER              VOLUME NAME
local               6008e9183ddcd4cfb7df33a5569cfa89ca7876af3f3a74b571ebf7051448a22d
local               880b8068a0a67870dcccca7b4dacb5824c7606becbc441e8f1997e4f7b6890ca
local               new_vol
Bhargavs-MacBook-Pro:~ bhargavbachina$ docker volume inspect new_vol
[
    {
        "CreatedAt": "2019-02-20T21:45:14Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/new_vol/_data",
        "Name": "new_vol",
        "Options": {},
        "Scope": "local"
    }
]
Bhargavs-MacBook-Pro:~ bhargavbachina$ docker volume rm new_vol
new_vol
Bhargavs-MacBook-Pro:~ bhargavbachina$ docker volume ls
DRIVER              VOLUME NAME
local               6008e9183ddcd4cfb7df33a5569cfa89ca7876af3f3a74b571ebf7051448a22d
local               880b8068a0a67870dcccca7b4dacb5824c7606becbc441e8f1997e4f7b6890ca
Bhargavs-MacBook-Pro:~ bhargavbachina$
```

docker volumes

login into docker VM and check the filesystem and volumes location. We can see docker volumes location in the below image.

screen $HOME/Library/Containers/com.docker.docker/Data/vms/0/tty



```
builder    containers  overlay2    swarm       volumes
linuxkit-025000000001:/var/lib/docker# ls -ll
total 60
drwx------    2 root     root          4096 Sep  3 02:07 builder
drwx------    4 root     root          4096 Sep  3 02:07 buildkit
drwx------    3 root     root          4096 Sep  3 02:07 containerd
drwx------    3 root     root          4096 Feb 18 19:33 containers
drwx------    3 root     root          4096 Sep  3 02:07 image
drwxr-x---    3 root     root          4096 Sep  3 02:07 network
drwx------   11 root     root         12288 Feb 21 01:16 overlay2
drwx------    4 root     root          4096 Sep  3 02:07 plugins
drwx------    2 root     root          4096 Feb 21 01:16 runtimes
drwx------    2 root     root          4096 Sep  3 02:07 swarm
drwx------    2 root     root          4096 Feb 21 01:16 tmp
drwx------    2 root     root          4096 Sep  3 02:07 trust
drwx------    5 root     root          4096 Feb 21 01:11 volumes
linuxkit-025000000001:/var/lib/docker# cd volumes/
```

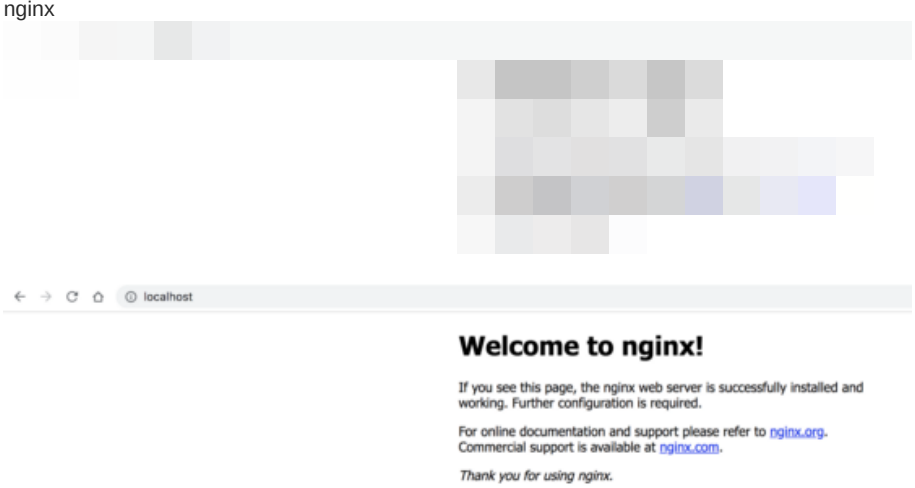
```
linuxkit-025000000001:/var/lib/docker/volumes# ls
6008e9183ddcd4cfb7df33a5569cfa89ca7876af3f3a74b571ebf7051440a22d
880b0060a0a67870dcccca7b4dacb5824c7606becbc441e8f1997e4f7b6090ca
metadata.db
new_vol
linuxkit-025000000001:/var/lib/docker/volumes#
```

Let's see the same example with volumes

Let's run the nginx container with the below command. we are starting nginx container with the welcome page mounted to volume ***new_vol*** that we created above and exposing the port 80.

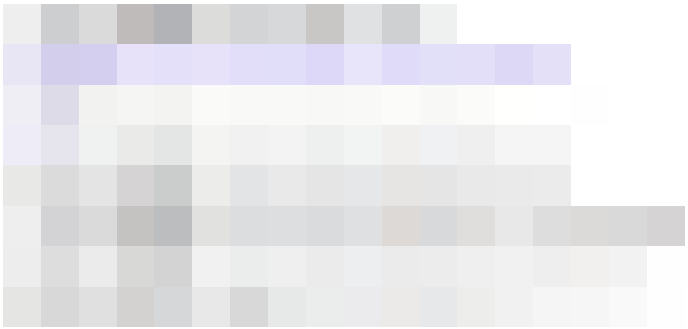
Once we run this command and ssh into docker volumes, we can see that volume prepopulated with default welcome page from nginx location ***/usr/share/nginx/html***

```
docker run -d --name=webApp1 --mount source=new_vol,destination=/usr/share/nginx/html -p 80:80
nginx
```



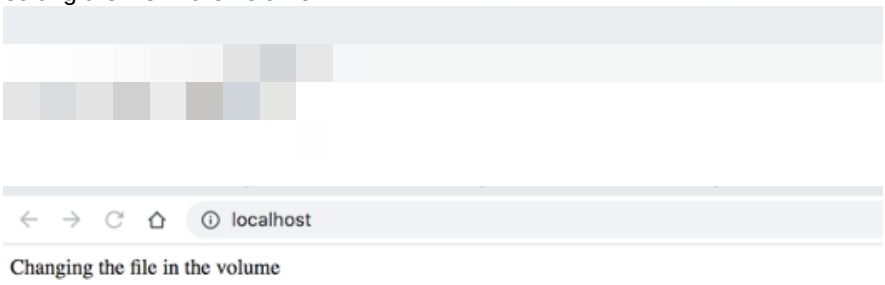
welcome page localhost:80

Let's go and change the index.html from the ***new_vol*** location by ssh into the docker.



```
linuxkit-025000000001:/var/lib/docker/volumes# ls
6008e9183ddcd4cfb7df33a5569cfa89ca7876af3f3a74b571ebf7051440a22d
880b0060a0a67870dcccca7b4dacb5824c7606becbc441e8f1997e4f7b6090ca
metadata.db
new_vol
linuxkit-025000000001:/var/lib/docker/volumes# cd new_vol/
linuxkit-025000000001:/var/lib/docker/volumes/new_vol# ls
_data
linuxkit-025000000001:/var/lib/docker/volumes/new_vol# cd _data/
linuxkit-025000000001:/var/lib/docker/volumes/new_vol/_data# ls
50x.html    index.html
linuxkit-025000000001:/var/lib/docker/volumes/new_vol/_data# echo "Changing the
file in the volume" > index.html
linuxkit-025000000001:/var/lib/docker/volumes/new_vol/_data# ls
50x.html    index.html
linuxkit-025000000001:/var/lib/docker/volumes/new_vol/_data# cat index.html
Changing the file in the volume
linuxkit-025000000001:/var/lib/docker/volumes/new_vol/_data#
```

editing the file in the volume



Let’s stop this container and start another one with the same command

```
docker stop webApp1docker run -d --name=webApp2 --mount
source=new_vol,destination=/usr/share/nginx/html -p 80:80 nginx
```

we can load the page again ***localhost:80*** and still see the html file that we edited in the volume.

So, with the help of volumes, we can easily access the data even we stop the container and it’s very easy to access data and import the data to anywhere.

Don’t forget to remove volumes

if you stop the container and remove it, you should remove volume ***new_vol*** manually. stopping or removing the containers doesn’t delete the volumes.

Conclusion

Volumes can be more safely shared among multiple containers. We can prepopulate the volume with the run command and we can even backup, restore and remove volumes.