DOCKER STORAGE

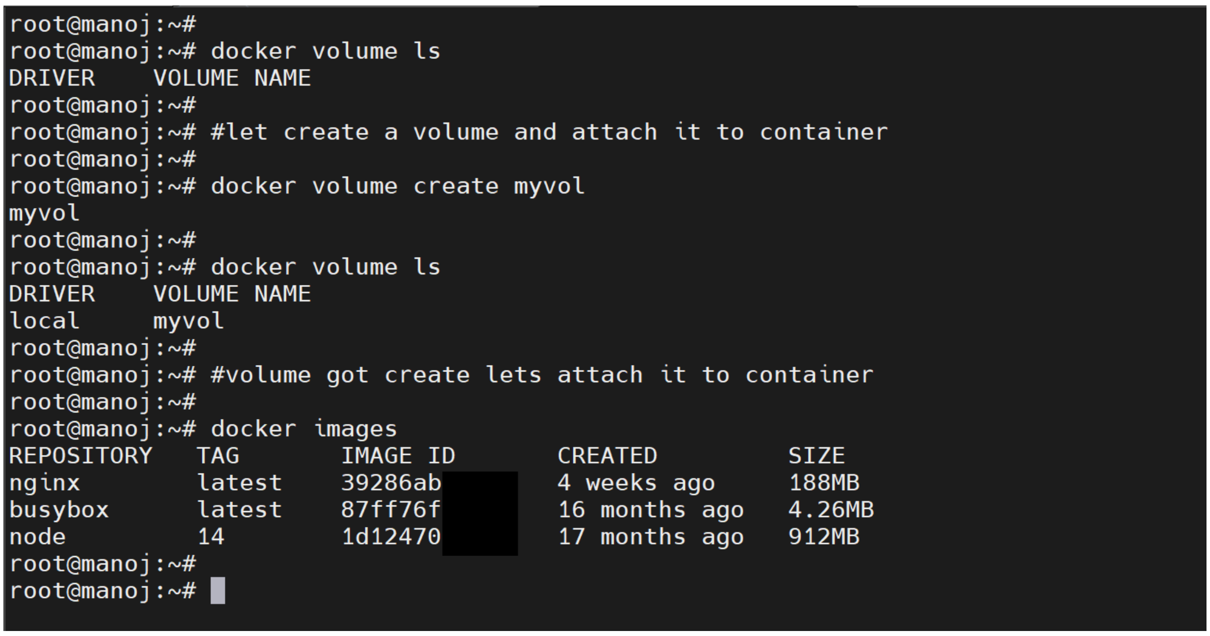
Docker storage refers to the management and persistence of data within containers. Containers are stateless and ephemeral, which means any data created within them can be lost when the container stops. Docker provides various storage options to ensure data persistence and manage application states effectively.

Two type of storage:

1) **Non-persistent**: data resides with the container , gets deleted when container deleted, By default all container has it.
   - ➔ Storage Drivers: Overlay2 (Default for Most Linux Systems)
2) **Persistent**: data doesn't resides within the container, doesn't get deleted when container deleted.
   - ➔ **Types of persistent storage**: Volume and Bind mount

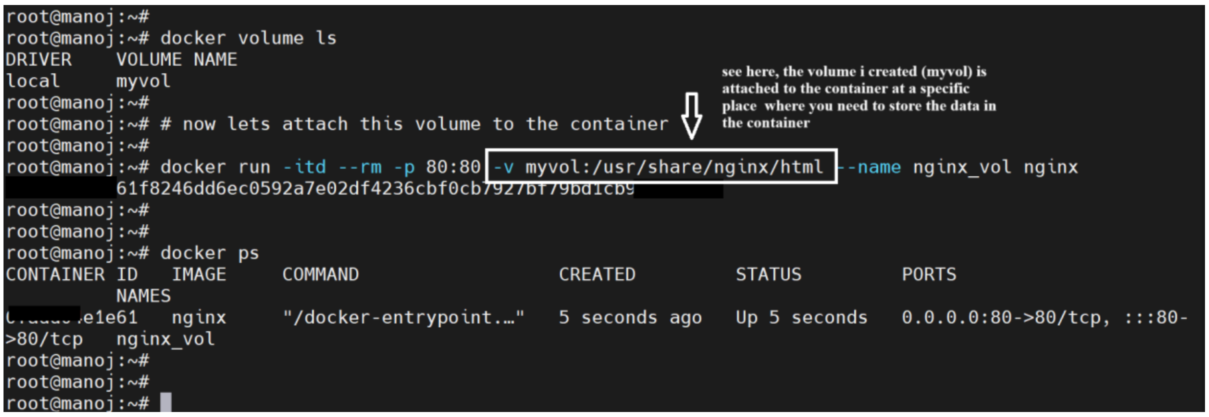
Volumes: Managed by docker and stored outside of the container filesystem. Volume are the best way to persist the data. Volumes can be shared between containers, backed up, restored, and mounted as read-only or read-write.

Creation of docker volume



```
root@manoj:~#
root@manoj:~# docker volume ls
DRIVER    VOLUME NAME
root@manoj:~#
root@manoj:~# #let create a volume and attach it to container
root@manoj:~#
root@manoj:~# docker volume create myvol
myvol
root@manoj:~#
root@manoj:~# docker volume ls
DRIVER    VOLUME NAME
local     myvol
root@manoj:~#
root@manoj:~# #volume got create lets attach it to container
root@manoj:~#
root@manoj:~# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
nginx        latest    39286ab[redacted]   4 weeks ago     188MB
busybox      latest    87ff76f[redacted]   16 months ago   4.26MB
node         14        1d12470[redacted]   17 months ago   912MB
root@manoj:~#
root@manoj:~# 
```

Mounting the created volume on to the container and providing the path where I want to store the volume in the container.

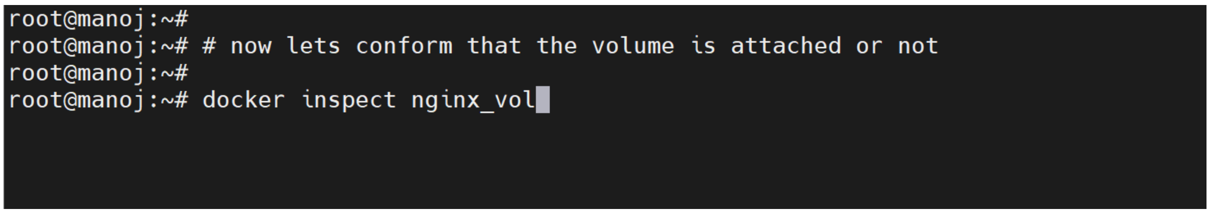


```
root@manoj:~#
root@manoj:~# docker volume ls
DRIVER    VOLUME NAME
local     myvol
root@manoj:~#
root@manoj:~# # now lets attach this volume to the container
root@manoj:~#
root@manoj:~# docker run -itd --rm -p 80:80 -v myvol:/usr/share/nginx/html --name nginx_vol nginx
[redacted]61f8246dd6ec0592a7e02df4236cbf0cb7927bf79bd1cb9[redacted]
root@manoj:~#
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
          NAMES
[redacted]e1e61   nginx     "/docker-entrypoint.…"   5 seconds ago   Up 5 seconds   0.0.0.0:80->80/tcp, :::80-
>80/tcp   nginx_vol
root@manoj:~#
root@manoj:~#
root@manoj:~# 
```

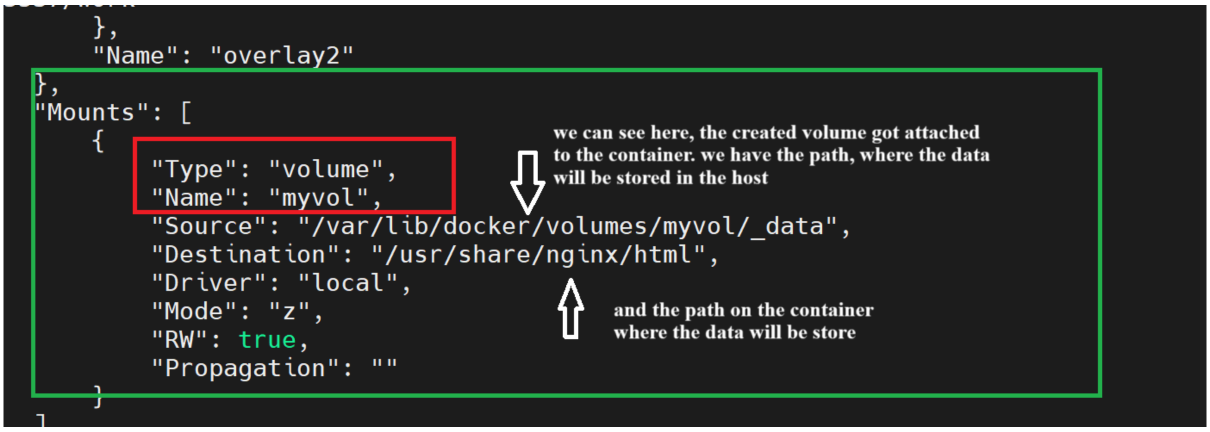
see here, the volume i created (myvol) is attached to the container at a specific place where you need to store the data in the container

DOCKER STORAGE

Using Docker inspect we can check volume type that mounted onto the container



```
root@manoj:~#
root@manoj:~# # now lets conform that the volume is attached or not
root@manoj:~#
root@manoj:~# docker inspect nginx_vol
```

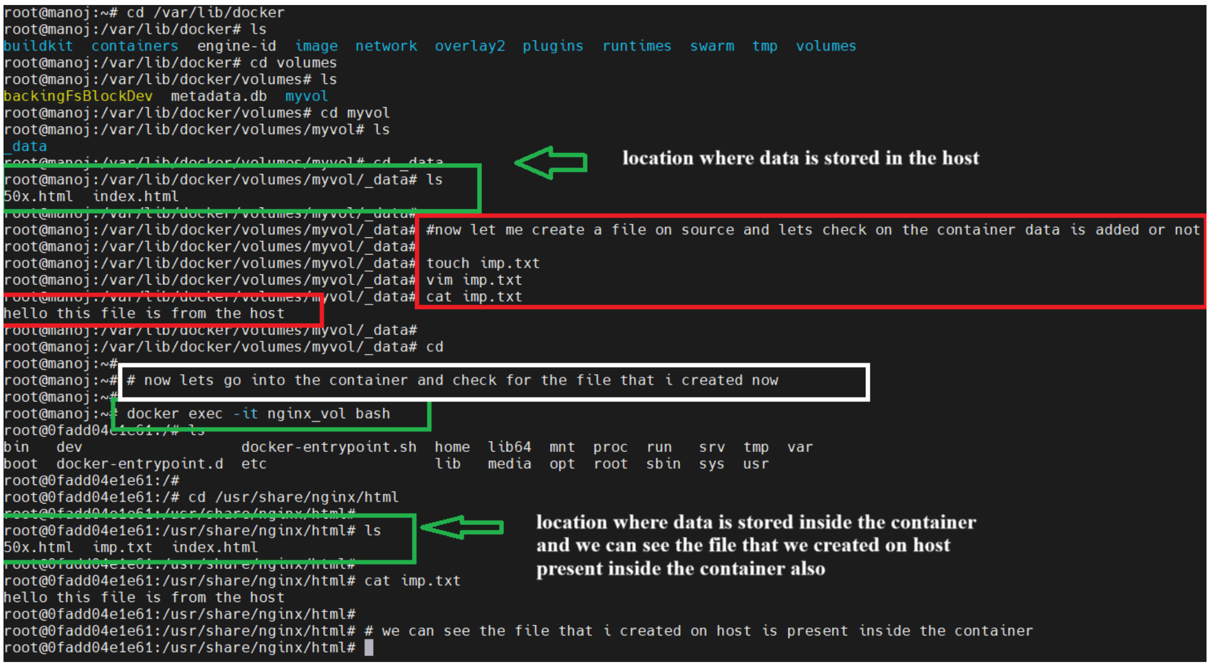


```
        },
        "Name": "overlay2"
    },
    "Mounts": [
        {
            "Type": "volume",
            "Name": "myvol",
            "Source": "/var/lib/docker/volumes/myvol/_data",
            "Destination": "/usr/share/nginx/html",
            "Driver": "local",
            "Mode": "z",
            "RW": true,
            "Propagation": ""
        }
    ]
```

we can see here, the created volume got attached to the container. we have the path, where the data will be stored in the host

and the path on the container where the data will be store

Location where data will be stored In the Host: **/var/lib/docker/volumes/ and** I created a file in host location and we can see that same data is present in the container location also



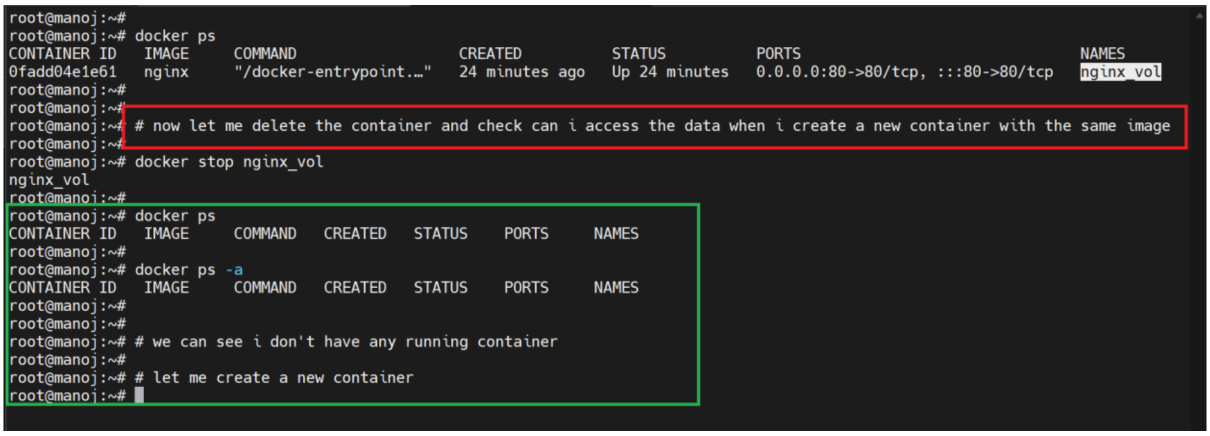
```
root@manoj:~# cd /var/lib/docker
root@manoj:/var/lib/docker# ls
buildkit  containers  engine-id  image  network  overlay2  plugins  runtimes  swarm  tmp  volumes
root@manoj:/var/lib/docker# cd volumes
root@manoj:/var/lib/docker/volumes# ls
backingFsBlockDev  metadata.db  myvol
root@manoj:/var/lib/docker/volumes# cd myvol
root@manoj:/var/lib/docker/volumes/myvol# ls
_data
root@manoj:/var/lib/docker/volumes/myvol# cd _data
root@manoj:/var/lib/docker/volumes/myvol/_data# ls
50x.html  index.html
root@manoj:/var/lib/docker/volumes/myvol/_data#
root@manoj:/var/lib/docker/volumes/myvol/_data# #now let me create a file on source and lets check on the container data is added or not
root@manoj:/var/lib/docker/volumes/myvol/_data#
root@manoj:/var/lib/docker/volumes/myvol/_data# touch imp.txt
root@manoj:/var/lib/docker/volumes/myvol/_data# vim imp.txt
root@manoj:/var/lib/docker/volumes/myvol/_data# cat imp.txt
hello this file is from the host
root@manoj:/var/lib/docker/volumes/myvol/_data#
root@manoj:/var/lib/docker/volumes/myvol/_data# cd
root@manoj:~#
root@manoj:~# # now lets go into the container and check for the file that i created now
root@manoj:~#
root@manoj:~# docker exec -it nginx_vol bash
root@0fadd04e1e61:/# ls
bin   dev                   docker-entrypoint.sh  home  lib64  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   media  opt  root  sbin  sys  usr
root@0fadd04e1e61:/#
root@0fadd04e1e61:/# cd /usr/share/nginx/html
root@0fadd04e1e61:/usr/share/nginx/html#
root@0fadd04e1e61:/usr/share/nginx/html# ls
50x.html  imp.txt  index.html
root@0fadd04e1e61:/usr/share/nginx/html#
root@0fadd04e1e61:/usr/share/nginx/html# cat imp.txt
hello this file is from the host
root@0fadd04e1e61:/usr/share/nginx/html#
root@0fadd04e1e61:/usr/share/nginx/html# # we can see the file that i created on host is present inside the container
root@0fadd04e1e61:/usr/share/nginx/html# 
```

location where data is stored in the host

location where data is stored inside the container and we can see the file that we created on host present inside the container also

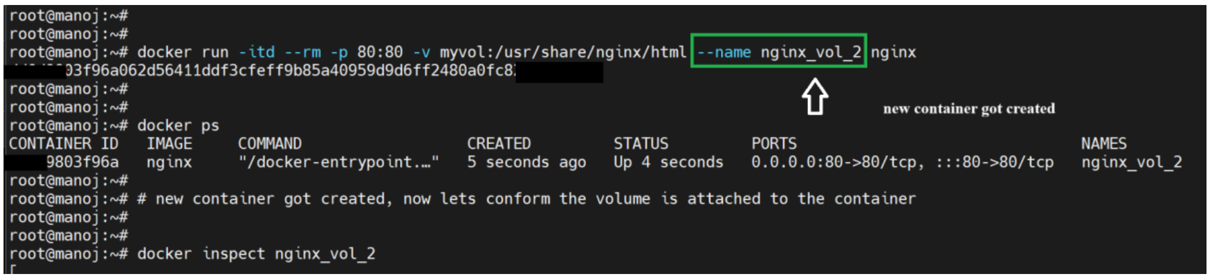
DOCKER STORAGE

I will delete the docker container and see can we still get our data



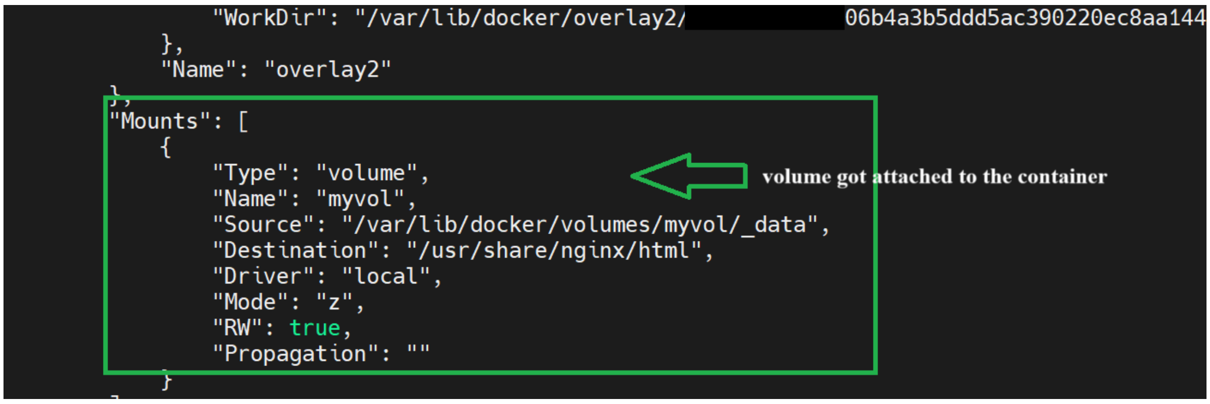
```
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                 NAMES
0fadd04e1e61   nginx     "/docker-entrypoint.…"   24 minutes ago   Up 24 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp     nginx_vol
root@manoj:~#
root@manoj:~#
root@manoj:~# # now let me delete the container and check can i access the data when i create a new container with the same image
root@manoj:~#
root@manoj:~# docker stop nginx_vol
nginx_vol
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@manoj:~#
root@manoj:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@manoj:~#
root@manoj:~#
root@manoj:~# # we can see i don't have any running container
root@manoj:~#
root@manoj:~# # let me create a new container
root@manoj:~# 
```

Creating the new container and mounting the volume that we created earlier.



```
root@manoj:~#
root@manoj:~#
root@manoj:~# docker run -itd --rm -p 80:80 -v myvol:/usr/share/nginx/html --name nginx_vol_2 nginx
[redacted]03f96a062d56411ddf3cfeff9b85a40959d9d6ff2480a0fc8[redacted]
root@manoj:~#
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                 NAMES
[redacted]9803f96a   nginx     "/docker-entrypoint.…"   5 seconds ago   Up 4 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp     nginx_vol_2
root@manoj:~#
root@manoj:~# # new container got created, now lets conform the volume is attached to the container
root@manoj:~#
root@manoj:~#
root@manoj:~# docker inspect nginx_vol_2
```

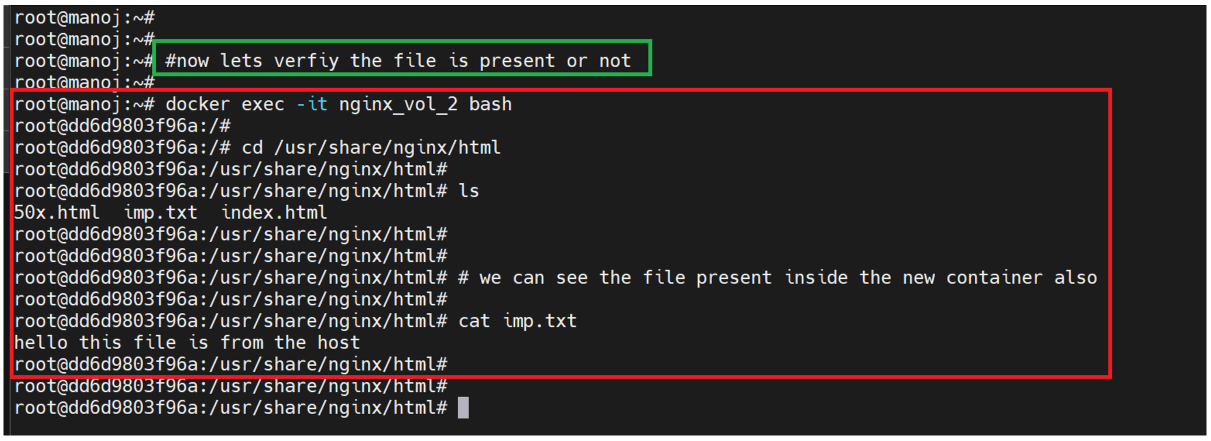
We can verify using docker inspect that volume got attached to the container.



```
                "WorkDir": "/var/lib/docker/overlay2/[redacted]06b4a3b5ddd5ac390220ec8aa144
            },
            "Name": "overlay2"
        },
        "Mounts": [
            {
                "Type": "volume",
                "Name": "myvol",
                "Source": "/var/lib/docker/volumes/myvol/_data",
                "Destination": "/usr/share/nginx/html",
                "Driver": "local",
                "Mode": "z",
                "RW": true,
                "Propagation": ""
            }
```

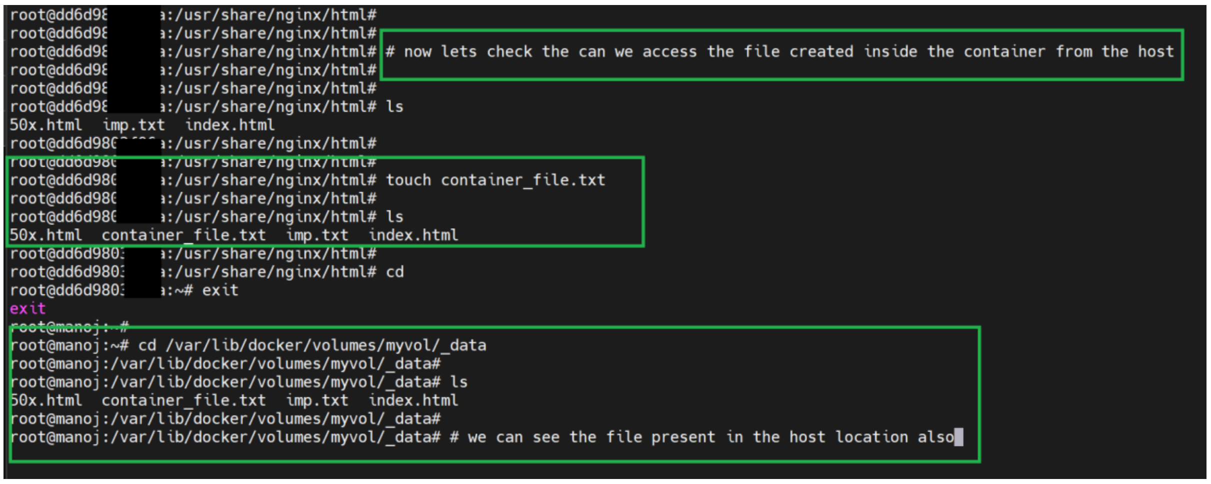
Now lets verify the files are still present or not. we can see that files are still present in the host, even after deleting the container.

DOCKER STORAGE



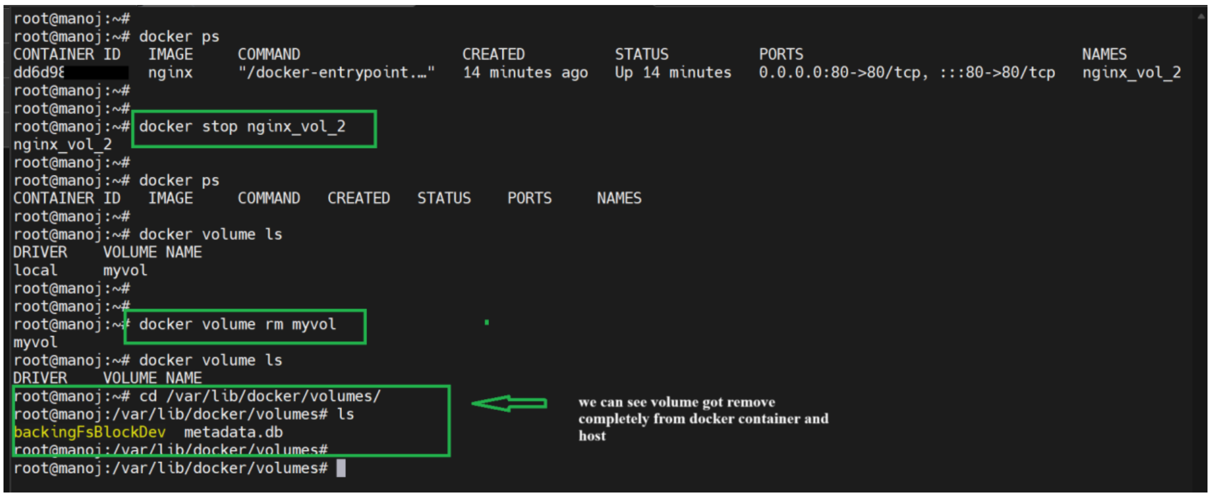
```
root@manoj:~#
root@manoj:~#
root@manoj:~# #now lets verfiy the file is present or not
root@manoj:~#
root@manoj:~# docker exec -it nginx_vol_2 bash
root@dd6d9803f96a:/#
root@dd6d9803f96a:/# cd /usr/share/nginx/html
root@dd6d9803f96a:/usr/share/nginx/html#
root@dd6d9803f96a:/usr/share/nginx/html# ls
50x.html  imp.txt  index.html
root@dd6d9803f96a:/usr/share/nginx/html#
root@dd6d9803f96a:/usr/share/nginx/html#
root@dd6d9803f96a:/usr/share/nginx/html# # we can see the file present inside the new container also
root@dd6d9803f96a:/usr/share/nginx/html#
root@dd6d9803f96a:/usr/share/nginx/html# cat imp.txt
hello this file is from the host
root@dd6d9803f96a:/usr/share/nginx/html#
root@dd6d9803f96a:/usr/share/nginx/html#
root@dd6d9803f96a:/usr/share/nginx/html#
```

Docker container also holds the files inside the container.



```
root@dd6d98[redacted]a:/usr/share/nginx/html#
root@dd6d98[redacted]a:/usr/share/nginx/html#
root@dd6d98[redacted]a:/usr/share/nginx/html# # now lets check the can we access the file created inside the container from the host
root@dd6d98[redacted]a:/usr/share/nginx/html#
root@dd6d98[redacted]a:/usr/share/nginx/html#
root@dd6d98[redacted]a:/usr/share/nginx/html# ls
50x.html  imp.txt  index.html
root@dd6d980[redacted]a:/usr/share/nginx/html#
root@dd6d980[redacted]a:/usr/share/nginx/html#
root@dd6d980[redacted]a:/usr/share/nginx/html# touch container_file.txt
root@dd6d980[redacted]a:/usr/share/nginx/html#
root@dd6d980[redacted]a:/usr/share/nginx/html# ls
50x.html  container_file.txt  imp.txt  index.html
root@dd6d980[redacted]a:/usr/share/nginx/html#
root@dd6d980[redacted]a:/usr/share/nginx/html# cd
root@dd6d980[redacted]a:~# exit
exit
root@manoj:~#
root@manoj:~# cd /var/lib/docker/volumes/myvol/_data
root@manoj:/var/lib/docker/volumes/myvol/_data#
root@manoj:/var/lib/docker/volumes/myvol/_data# ls
50x.html  container_file.txt  imp.txt  index.html
root@manoj:/var/lib/docker/volumes/myvol/_data#
root@manoj:/var/lib/docker/volumes/myvol/_data# # we can see the file present in the host location also
```

Remove the docker volume



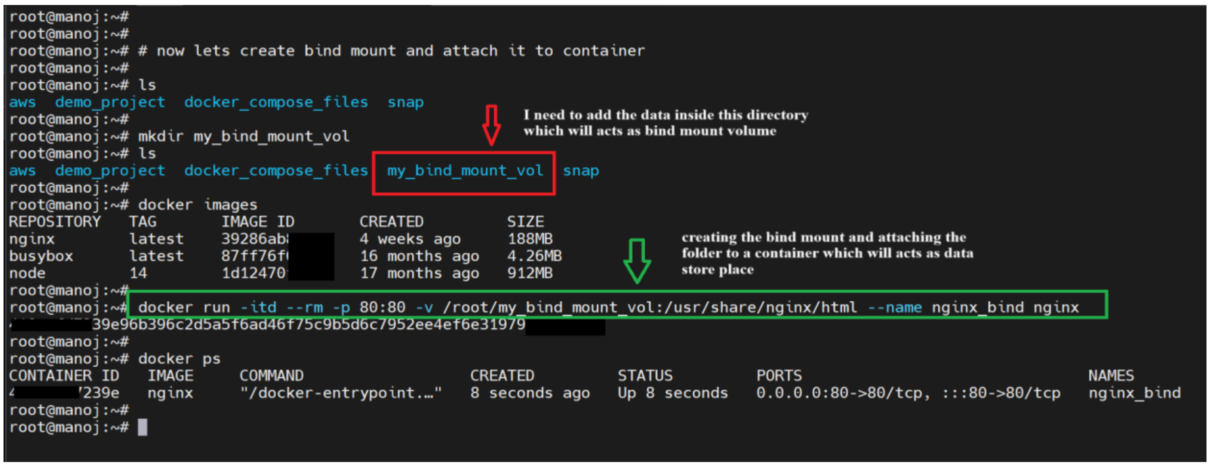
```
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                 NAMES
dd6d98[redacted]   nginx     "/docker-entrypoint.…"   14 minutes ago   Up 14 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp     nginx_vol_2
root@manoj:~#
root@manoj:~#
root@manoj:~# docker stop nginx_vol_2
nginx_vol_2
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@manoj:~#
root@manoj:~# docker volume ls
DRIVER    VOLUME NAME
local     myvol
root@manoj:~#
root@manoj:~#
root@manoj:~# docker volume rm myvol
myvol
root@manoj:~# docker volume ls
DRIVER    VOLUME NAME
root@manoj:~# cd /var/lib/docker/volumes/
root@manoj:/var/lib/docker/volumes# ls
backingFsBlockDev  metadata.db
root@manoj:/var/lib/docker/volumes#
root@manoj:/var/lib/docker/volumes#
```

we can see volume got remove completely from docker container and host

DOCKER STORAGE

Bind Mount: Bind mounts allow containers to access specific directories on the host's filesystem, outside the Docker-managed area. Ideal for development environments where you need real-time access to host files.

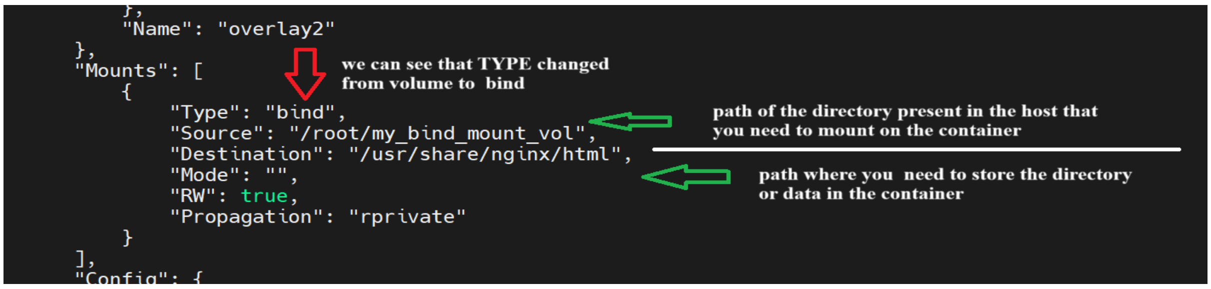
Creating the director to mount on the container.


I need to add the data inside this directory which will acts as bind mount volume
creating the bind mount and attaching the folder to a container which will acts as data store place

```
root@manoj:~#
root@manoj:~#
root@manoj:~# # now lets create bind mount and attach it to container
root@manoj:~#
root@manoj:~# ls
aws  demo_project  docker_compose_files  snap
root@manoj:~#
root@manoj:~# mkdir my_bind_mount_vol
root@manoj:~# ls
aws  demo_project  docker_compose_files  my_bind_mount_vol  snap
root@manoj:~#
root@manoj:~# docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
nginx        latest    39286ab[redacted] 4 weeks ago      188MB
busybox      latest    87ff76f           16 months ago    4.26MB
node         14        1d12470           17 months ago    912MB
root@manoj:~#
root@manoj:~# docker run -itd --rm -p 80:80 -v /root/my_bind_mount_vol:/usr/share/nginx/html --name nginx_bind nginx
[redacted]39e96b396c2d5a5f6ad46f75c9b5d6c7952ee4ef6e31979[redacted]
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                               NAMES
[redacted]239e   nginx     "/docker-entrypoint.…"   8 seconds ago   Up 8 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp   nginx_bind
root@manoj:~#
root@manoj:~# █
```

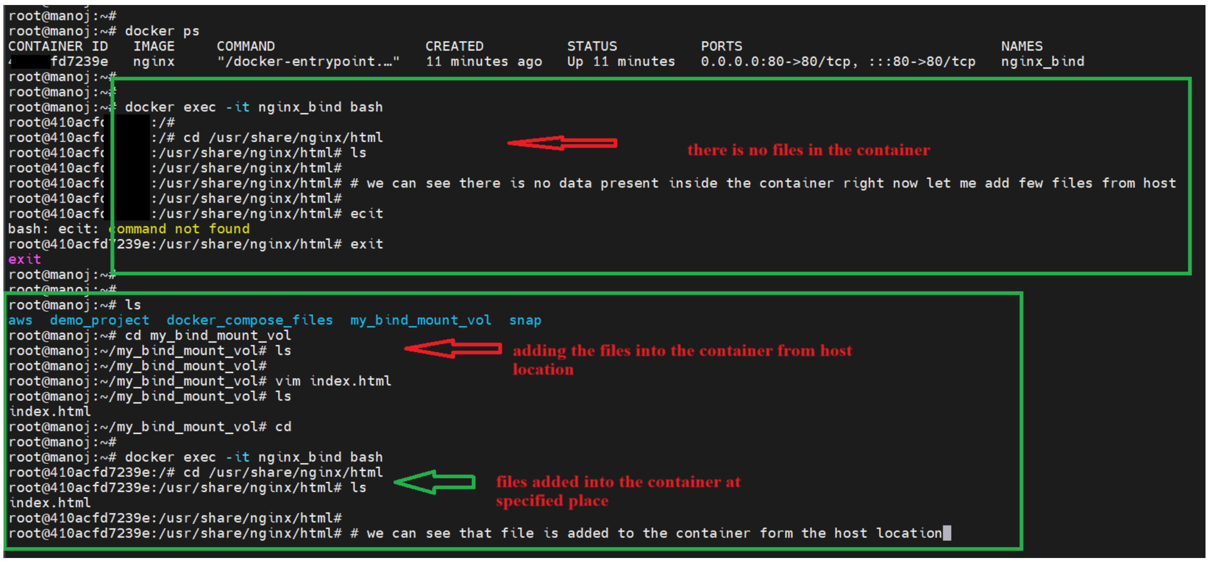
Now we can see here type of mount changes to bind, this show that we mounted bind volume onto the docker container.

Even we can see the source location present on the host and Destination location present on the container here.


we can see that TYPE changed from volume to bind
path of the directory present in the host that you need to mount on the container
path where you need to store the directory or data in the container

```
            },
            "Name": "overlay2"
        },
        "Mounts": [
            {
                "Type": "bind",
                "Source": "/root/my_bind_mount_vol",
                "Destination": "/usr/share/nginx/html",
                "Mode": "",
                "RW": true,
                "Propagation": "rprivate"
            }
        ],
        "Config": {
```

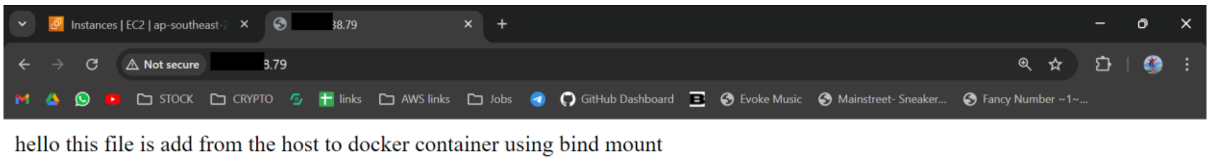
Now let me add file in the host location and check can we see same file in docker bind mount location also.


there is no files in the container
adding the files into the container from host location
files added into the container at specified place

```
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                               NAMES
[redacted]fd7239e   nginx     "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp   nginx_bind
root@manoj:~#
root@manoj:~#
root@manoj:~# docker exec -it nginx_bind bash
root@410acf[redacted]:/#
root@410acf:/# cd /usr/share/nginx/html
root@410acf:/usr/share/nginx/html# ls
root@410acf:/usr/share/nginx/html#
root@410acf:/usr/share/nginx/html# # we can see there is no data present inside the container right now let me add few files from host
root@410acf:/usr/share/nginx/html#
root@410acf:/usr/share/nginx/html# ecit
bash: ecit: command not found
root@410acfd7239e:/usr/share/nginx/html# exit
exit
root@manoj:~#
root@manoj:~#
root@manoj:~# ls
aws  demo_project  docker_compose_files  my_bind_mount_vol  snap
root@manoj:~# cd my_bind_mount_vol
root@manoj:~/my_bind_mount_vol# ls
root@manoj:~/my_bind_mount_vol#
root@manoj:~/my_bind_mount_vol# vim index.html
root@manoj:~/my_bind_mount_vol# ls
index.html
root@manoj:~/my_bind_mount_vol# cd
root@manoj:~#
root@manoj:~# docker exec -it nginx_bind bash
root@410acfd7239e:/# cd /usr/share/nginx/html
root@410acfd7239e:/usr/share/nginx/html# ls
index.html
root@410acfd7239e:/usr/share/nginx/html#
root@410acfd7239e:/usr/share/nginx/html# # we can see that file is added to the container form the host location█
```

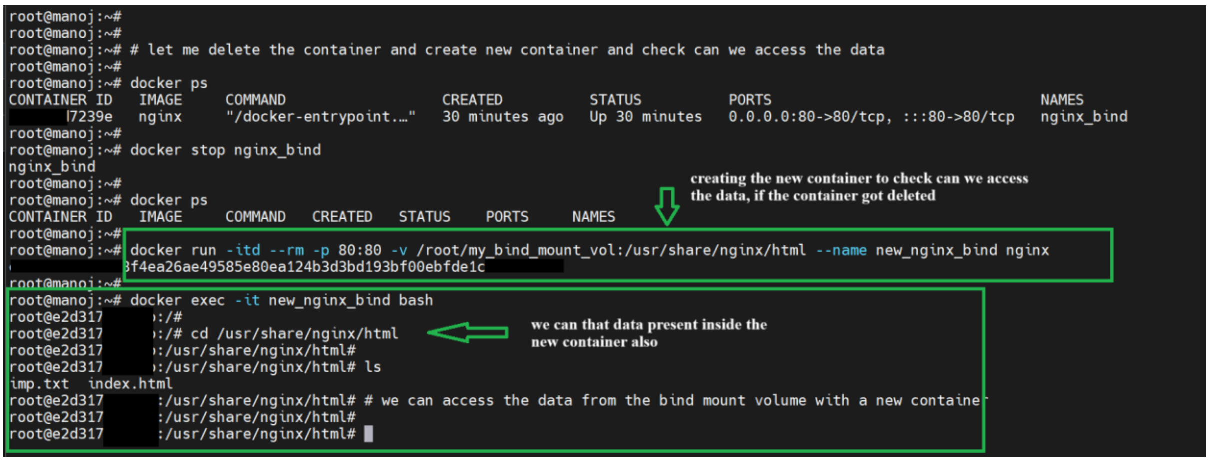
DOCKER STORAGE

We can see that, I an access the data from the bind mount location



hello this file is add from the host to docker container using bind mount

Deleted the current container and built new container to check still can we access the data present in bind mount directory.

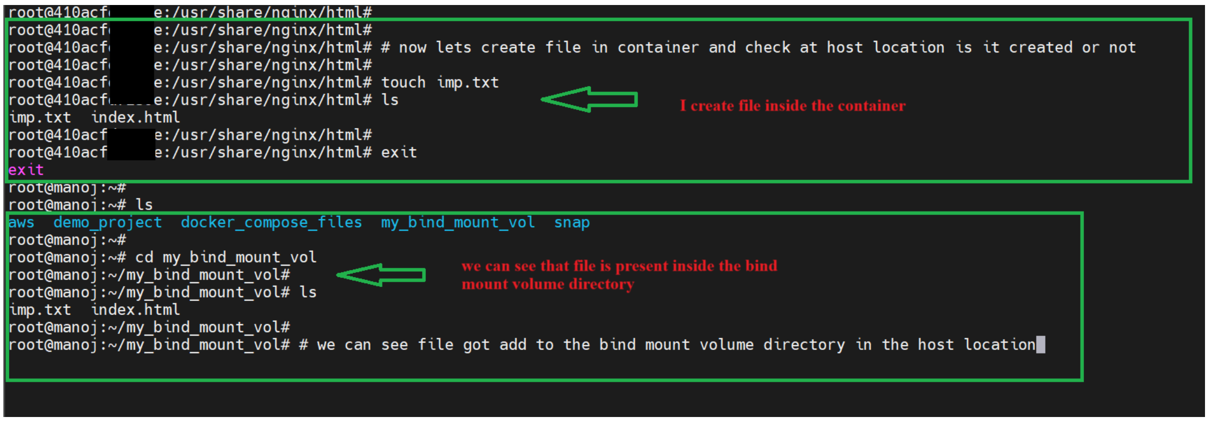


```
root@manoj:~#
root@manoj:~#
root@manoj:~# # let me delete the container and create new container and check can we access the data
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                               NAMES
[redacted]l7239e   nginx     "/docker-entrypoint.…"   30 minutes ago   Up 30 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp   nginx_bind
root@manoj:~#
root@manoj:~# docker stop nginx_bind
nginx_bind
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@manoj:~#
root@manoj:~# docker run -itd --rm -p 80:80 -v /root/my_bind_mount_vol:/usr/share/nginx/html --name new_nginx_bind nginx
[redacted]3f4ea26ae49585e80ea124b3d3bd193bf00ebfde1c[redacted]
root@manoj:~#
root@manoj:~# docker exec -it new_nginx_bind bash
root@e2d317[redacted]:/#
root@e2d317[redacted]:/# cd /usr/share/nginx/html
root@e2d317[redacted]:/usr/share/nginx/html#
root@e2d317[redacted]:/usr/share/nginx/html# ls
imp.txt  index.html
root@e2d317[redacted]:/usr/share/nginx/html# # we can access the data from the bind mount volume with a new containe
root@e2d317[redacted]:/usr/share/nginx/html#
root@e2d317[redacted]:/usr/share/nginx/html# █
```

creating the new container to check can we access the data, if the container got deleted

we can that data present inside the new container also

And we can see still I can access the data, if a bind mount directory on the host is deleted, the container will lose access to the data in that directory. The container will still run, but attempts to read or write to the bind-mounted path will fail or result in an empty directory.



```
root@410acf[redacted]e:/usr/share/nginx/html#
root@410acf[redacted]e:/usr/share/nginx/html#
root@410acf[redacted]e:/usr/share/nginx/html# # now lets create file in container and check at host location is it created or not
root@410acf[redacted]e:/usr/share/nginx/html#
root@410acf[redacted]e:/usr/share/nginx/html# touch imp.txt
root@410acf[redacted]e:/usr/share/nginx/html# ls
imp.txt  index.html
root@410acf[redacted]e:/usr/share/nginx/html#
root@410acf[redacted]e:/usr/share/nginx/html# exit
exit
root@manoj:~#
root@manoj:~# ls
aws  demo_project  docker_compose_files  my_bind_mount_vol  snap
root@manoj:~#
root@manoj:~# cd my_bind_mount_vol
root@manoj:~/my_bind_mount_vol#
root@manoj:~/my_bind_mount_vol# ls
imp.txt  index.html
root@manoj:~/my_bind_mount_vol#
root@manoj:~/my_bind_mount_vol# # we can see file got add to the bind mount volume directory in the host location█
```

I create file inside the container

we can see that file is present inside the bind mount volume directory