

Container gets deleted but volume doesn't get deleted.
Container to container, host to container,


Lec-28 - Docker Volume & How to Share it

- Volume is simply a directory inside our Container.
- Firstly, we have to declare this directory as a Volume and then share Volume.
- Even if we stop Container, still we can access Volume.
- Volume will be created in one Container.
- You can declare a directory as a Volume only while creating Container.
- You can't create Volume from existing Container.
- You can share one Volume across any number of Containers.

→ Volume will not be included when you update an image.


You can map Volume in two ways →

- ① Container ↔ Container
- ② Host ↔ Container



Benefits of Volume

- Decoupling Container from storage
- Share Volume among different Containers
- Attach Volume to Containers
- On deleting Container Volume does not delete.



Creating Volume from Dockerfile

① Create a Dockerfile and Write *•wq*

```
FROM ubuntu
VOLUME ["/myvolume"]
```

Then Create image from this dockerfile

→ `docker build -t myimage .`

Now Create a Container from this image & Run

```
docker run -it --name container1 myimage /bin/bash
```

Now do `ls`, you Can see `myvolume`

Now, Share Volume with another Container

Container ↔ Container

→ `docker run -it --name Container2 (new) --privileged=true --volumes-from container1 ubuntu /bin/bash`

Now after Creating Container2, `myvolume` is visible. Whatever you do in One Volume, Can see from other Volume.

→ `touch /myvolume/samplefile`

→ `docker start Container1`

→ `docker attach Container1`

→ `ls /myvolume`

you Can see Samplefile here

`exit`

--privileged true to give permission to container2 for volume .

```
Dockerfile
1 FROM ubuntu
2 VOLUME ["/myvolume"]

root@host01:~/workspace$ docker build -t myimage .
[+] Building 185.7s (5/5) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 69B
=> [internal] load metadata for docker.io/library/ubuntu:lat
=> [1/1] FROM docker.io/library/ubuntu@sha256:67211c14fa74f0
=> => resolve docker.io/library/ubuntu@sha256:67211c14fa74f0
=> => sha256:08d22c0ceb150ddeb2237c5fa3129c0183f3cc6f5eeb2e7
=> => sha256:2ab09b027e7f3a0c2e8bb1944ac46de38cebab7145f0bd6
=> => sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3
=> => sha256:7a57c69fe1e9d5b97c5fe649849e79f2cfc3bf11d10bbd5
=> => extracting sha256:2ab09b027e7f3a0c2e8bb1944ac46de38ceb
=> exporting to image
=> => exporting layers
=> => writing image sha256:8419955127158a7c4cd62a0d83b36a530
=> => naming to docker.io/library/myimage

root@host01:~/workspace$
root@host01:~/workspace$ docker run -it --name sapna myimage:latest /bin/bash
root@6cc6bc29b549:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt myvolume opt proc root run sbin srv sys usr var
root@6cc6bc29b549:/#
```

```

root@host01:~/workspace$ docker run -it --name sapna1 --privileged=true --volumes-from sapna myimage:latest /bin/bash
root@c6358f07a146:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt myvolume opt proc root run sbin srv sys tmp usr
root@c6358f07a146:/# cd myvolume/
root@c6358f07a146:/myvolume# ls
v1 v2 v3
root@c6358f07a146:/myvolume# exit
exit
root@host01:~/workspace$

```

Now, try to Create Volume by
using Command

→ `docker run -it --name container3`
`-v /volume2 ubuntu /bin/bash`

Do `ls` → `cd /volume2`

Now Create One file Cont3file and exit

Now Create One more Container, and
Share Volume2

→ `docker run -it --name Container4`
`(--privileged=true --volumes-from Container3`
`ubuntu /bin/bash`

Now you are inside Container, do
`ls`, you can see Volume2

Now Create One file inside this
Volume and then check in
Container 3, you can see that
file.

```

root@host01:~/workspace$ docker run -it --name sapna2 -v /volume2 ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
2ab09b027e7f: Already exists
Digest: sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a6d21
Status: Downloaded newer image for ubuntu:latest
root@e4cb388b4965:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var volume2
root@e4cb388b4965:/# cd v
bash: cd: v: No such file or directory
root@e4cb388b4965:/# cd volume2/
root@e4cb388b4965:/volume2# ls
root@e4cb388b4965:/volume2# touch v1 v2 v3
root@e4cb388b4965:/volume2# ls
v1 v2 v3
root@e4cb388b4965:/volume2# exit
exit
root@host01:~/workspace$ docker run -it --name sapna3 --privileged=true --volumes-from sapna2 ubuntu /bin/bash
root@bc16c35363a2:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr
root@bc16c35363a2:/# cd volume2/
root@bc16c35363a2:/volume2# ls
v1 v2 v3
root@bc16c35363a2:/volume2#

```

```

root@bc16c35363a2:/# cd volume2/
root@bc16c35363a2:/volume2# ls
v1 v2 v3
root@bc16c35363a2:/volume2# touch v4
root@bc16c35363a2:/volume2# exity
bash: exity: command not found
root@bc16c35363a2:/volume2# exit
exit
root@host01:~/workspace$ docker start sapna2
sapna2
root@host01:~/workspace$ docker attach sapna2
root@e4cb388b4965:/# ls
bin boot dev etc home lib lib32 lib64 libx32 med
root@e4cb388b4965:/# cd volume2/
root@e4cb388b4965:/volume2# ls
v1 v2 v3 v4
root@e4cb388b4965:/volume2#

```


Files we created in containers are now shared between containers .

Volumes (Host - Container)

- Verify files in /home/ec2-user
- docker run -it --name hostCont -v /home/ec2-user:/rajput --privileged=true
host ubuntu /bin/bash → container
- cd /rajput
 Do ls, now you can see all files of host machine
- touch rajputfile (in Container)
 exit
- Now check in EC2 machine, you can see this file

Some other Commands

- docker volume ls
- docker volume create <volumename>
- docker volume rm <volumename>
- docker volume prune
 { It removed all unused docker }
 Volume
- docker volume inspect <volumename>
- docker Container inspect <ContainerName>



```
Dockerfile
root@host01:~/workspace$ pwd
/root/workspace
root@host01:~/workspace$ docker run -it --name hostcont -v /root/workspace:/sapna --privileged=true ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
2ab09b027e7f: Pull complete
Digest: sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a6d21
Status: Downloaded newer image for ubuntu:latest
root@e8ae8bacb560:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sapna sbin srv sys usr va
root@e8ae8bacb560:/# cd sapna
root@e8ae8bacb560:/sapna# ls
Dockerfile
root@e8ae8bacb560:/sapna# exit
exit
root@host01:~/workspace$ ls
Dockerfile
root@host01:~/workspace$
```

```
root@host01:~/workspace$ docker start hostcont
hostcont
root@host01:~/workspace$ docker attach hostcont
root@e8ae8bacb560:/# ls
bin boot dev etc home lib lib32 lib64 li
root@e8ae8bacb560:/# cd sapna
root@e8ae8bacb560:/sapna# ls
Dockerfile
root@e8ae8bacb560:/sapna# touch new
root@e8ae8bacb560:/sapna# ls
Dockerfile new
root@e8ae8bacb560:/sapna# exit
exit
root@host01:~/workspace$ ls
Dockerfile new
root@host01:~/workspace$
```