Docker Volume (Manage Data in Docker)

Problem Statement: By default, all files created inside a container are stored on a writable container layer. This means that:

- **The data doesn't persist** when that **container no longer exists**, and it can be difficult to get the data out of the container if another process needs it.

- A container's writable layer is tightly coupled to the host machine where the container is running. You **can't** easily move the data somewhere else.

- Writing into a container's writable layer requires a **storage driver** (Storage drivers allow you to create data in the writable layer of your container. The files won't be persisted after the container is deleted) to manage the filesystem. The storage driver provides a union filesystem (This allows a file system to appear as writable), using the Linux kernel. This reduces performance because it writes directly to the host filesystem.

Solution:

Docker has two options for containers to store files in the host machine, so that the files are persisted even after the container stops: **volumes**, and **bind mounts**.

1. **Volumes** are stored in a part of the host filesystem which is *managed by Docker* (/var/lib/docker/volumes/ on Linux). Non-Docker processes should not modify this part of the filesystem. Volumes are the best way to persist data in Docker.

2. **Bind mounts** may be stored *anywhere* on the host system. They may even be important system files or directories. Non-Docker processes on the Docker host or a Docker container can modify them at any time.

Docker Volume:

- Created and managed by Docker. You can create a volume explicitly using the **docker volume create** command, or Docker can create a volume during container or service creation.

- When you create a volume, it is stored within a directory on the Docker host. When you mount the volume into a container, this directory is what is mounted into the container. This is similar to the way that bind mounts work, except that volumes are managed by Docker and are isolated from the core functionality of the host machine.

- A given volume can be mounted into multiple containers simultaneously.

- When no running container is using a volume, the volume is still available to Docker and is not removed automatically.
  You can remove unused volumes using

docker volume prune

Docker Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.

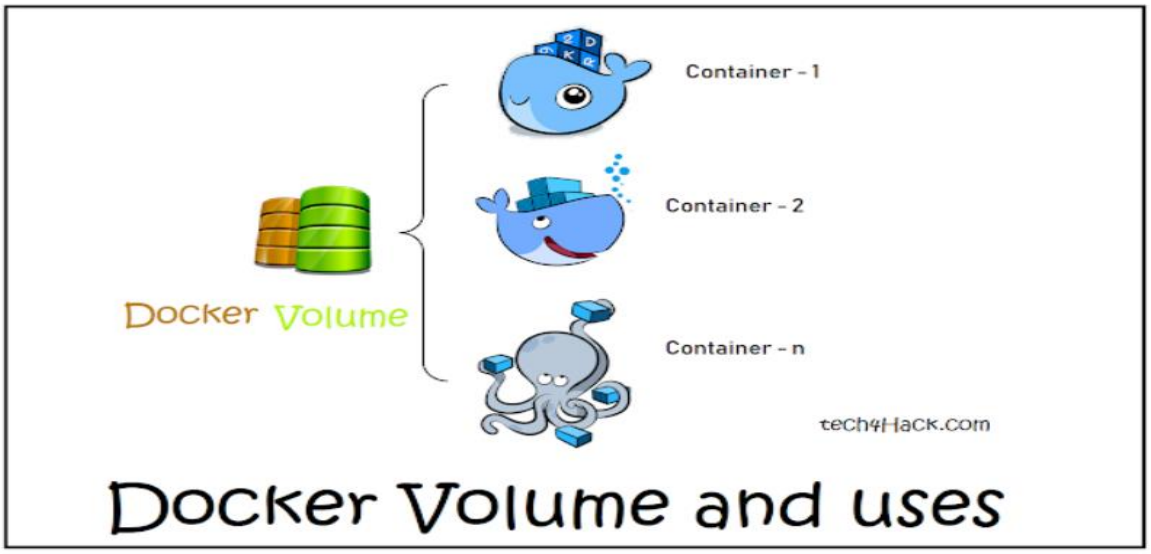
docker volume ls

In Docker when we Create Container, Container need some place where container store data, when we do not provide any specific location, it gets store in the container and when we delete the container it deletes that data.

When we work in Enterprise level, Data should not be deleted or required to create more container with old data and share data between containers.

Uses of Docker Volume

1. To keep data around when a container is removed
2. To share data between the hosts file system and the Docker container
3. To share data with other Docker containers



Docker Volume commands

To create a volume

```
docker volume  create <vol_name>
```

To check storage driver for docker.

- **overlay2** is the **preferred storage driver**, for all currently supported Linux distributions, and requires no extra configuration.

  /**var/lib/docker/overlay2**

  Why: Overlay2 is more efficient in terms of utilization.

```
[root@ip-172-31-46-1 shm]# docker info
Containers: 1
 Running: 0
 Paused: 0
 Stopped: 1
Images: 6
Server Version: 18.06.1-ce
Storage Driver: overlay2
 Backing Filesystem: xfs
```

- **aufs** is the preferred storage driver for Docker 18.06 and older, when running on Ubuntu 14.04 on kernel 3.13 which has no support for overlay2.

- **devicemapper** devicemapper was the recommended storage driver for CentOS and RHEL, as their kernel version did not support overlay2. However, current versions of CentOS and RHEL now have support for overlay2, which is now the recommended driver.

- The btrfs and zfs storage drivers are used if they are the backing filesystem (the filesystem of the host on which Docker is installed). These filesystems allow for advanced options, such as creating “snapshots”, but require more maintenance and setup. Each of these relies on the backing filesystem being configured correctly.

- The vfs storage driver is intended for testing purposes, and for situations where no copy-on-write filesystem can be used. Performance of this storage driver is poor, and is not generally recommended for production use.

If we don’t provide any driver , automatically it will attach Local driver

Local driver: When you write something on docker writeable layer

To check docker images on host vm.

/var/lib/docker/image/overlay2/imaged

To check docker containers filesystem on host vm.

/var/lib/docker/containers

Docker image’s actual layer

/var/lib/docker/overlay2

Display detailed information on one or more volumes

docker volume inspect <vol_name>

List volumes

```
docker volume ls
```

Remove all unused local volumes

```
docker volume prune
```

Remove one or more volumes

```
docker volume rm
```

Lab1: Persistent Volume

```
docker volume  create  myVOl
```

Create a 1st container as ubuntu1

```
docker run -itd --name ubunut1 -v myVOl:/home ubuntu /bin/bash
docker exec -it ubunut1 /bin/bash
cd /home
touch abc.txt
```

Create a 2nd container as ubuntu2

```
docker run -itd --name ubunut1 -v myVOl:/home ubuntu /bin/bash
```

login to 2nd container and verify the text file

Delete both the containers and created 3rd container data will be there

```
docker run -itd --name ubunut3 -v myVOl:/home ubuntu /bin/bash
```

Lab2:

docker volume create myVOl1

To check volume.

/var/lib/docker/volumes

Case-1: Attach this volume with container while creating the container

docker run -d --name=myJenkins1 -v myVol1:/var/jenkins_home -p 8181:8080 -p 50001:50000 jenkins

docker exec -it <container Id> cat /var/jenkins_home/secrets/initialAdminPassword

Explanation of above command

--name=myJenkins1	Giving name to my container 1
-v myVol1:/var/jenkins_home	myVol1 - volume of my base machine that is attaching with /var/jenkins_home (Volume of container)
-p 8181:8080	8181 - port of my base machine which is attaching with port 8080 of container on Jenkins is running
p 50001:50000	this port used for API configuration

Case -2: Attach this same volume with container no. 2 while creating the container

docker run -d --name=myJenkins2 -v myVol1:/var/jenkins_home -p 8282:8080 -p 50002:50000 jenkins

Now verify , how many volumes are attached to our container:

docker inspect -f '{{ .Mounts }}' <containerid>

Case -4: we can use any specific Container and a storage volume container and can share that volume with any number of Docker container

These container called as "Data Only Containers"

- *Create a data only container. As we don't have to do any operations inside this container, we used "/bin/true". It will stop the container,*
- ***/bin/true" command just returns a success status code***

```
docker run -it --name=data-only -v /DATA centos:latest /bin/true
```

- *We already stopped this container so it will show in*

```
docker ps –a
```

- *Now create one more container and mount the volume from base container using "--volumes-from" option.*

docker run -it --volumes-from data-only centos /bin/bash