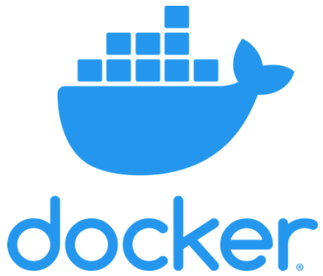
Docker Deep Dive

Chapter #10



Docker Volumes

Docker

VOLUMES

DOCKER DATA

Persistent vs Non-persistent data

DOCKER VOLUMES

Attaching docker volumes with containers

www.networknuts.net

DOCKER DATA

Two main categories of data.
Persistent & non persistent.

CONTAINER OWN STORAGE

Every docker container has its own non-persistent storage, which is created while creating container.

DOCKER VOLUMES

Used to store persistent data. They are not attached with the lifecycle of container.

Docker Volumes

Docker volumes: deep dive

There are two main categories of data. Persistent and non-persistent.

Persistent is the stuff you need to keep. Things like; customer records, financials, bookings, audit logs, and even some types of application log data. Non-persistent is the stuff you don’t need to keep.

Both are important, and Docker has options for both.

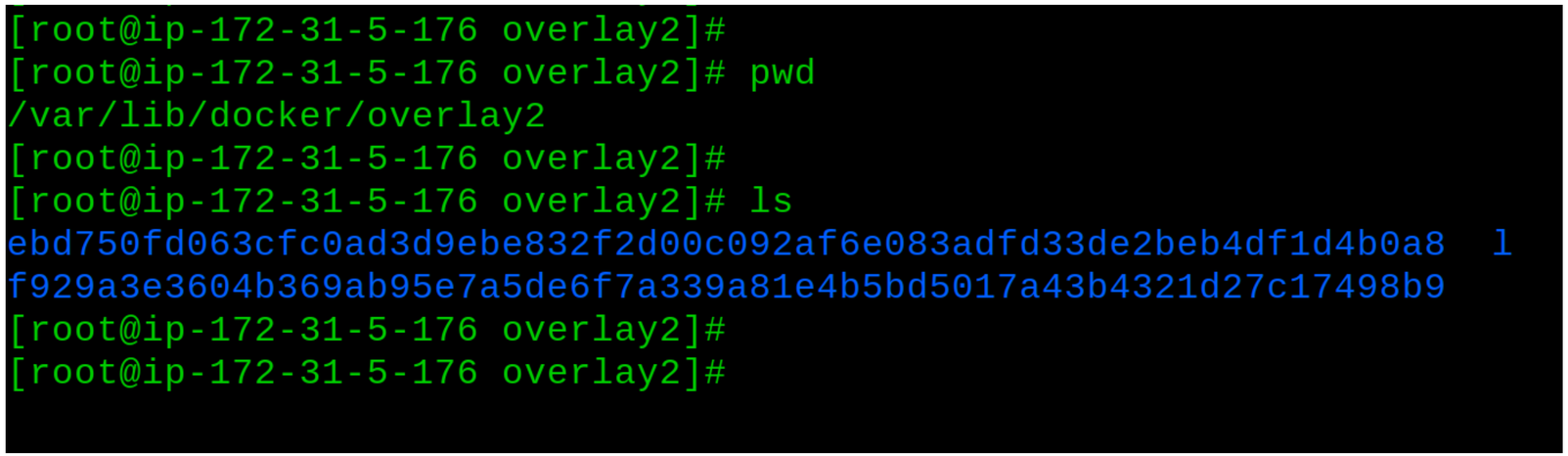
Every Docker container gets its own non-persistent storage. It’s automatically created, alongside the container, and it’s tied to the lifecycle of the container. That means deleting the container will delete this storage and any data on it.

If you want your container’s data to stick around (persist), you need to put it on a volume. Volumes are decoupled from containers, meaning you create and manage them separately, and they’re not tied to the lifecycle of any container. You can delete a container with a volume, and the volume will not be deleted.

Containers are great at stateless and non-persistent stuff.

Every container automatically gets a bunch of local storage. By default, this is where all of the container’s files and filesystem go. Its also known as; local storage, graphdriver storage, and snapshotter storage. It’s an integral part of the container, and is tied to the container’s lifecycle. It gets created when the container gets created, and it gets deleted when the container gets deleted.

On Linux systems, it exists somewhere under /var/lib/docker/<storage-driver>/ as part of the container.



By default, all storage within a container uses this local storage. So every directory in a container uses this storage by default.

Create a container, go inside it and create a "hello.txt" file inside /tmp/

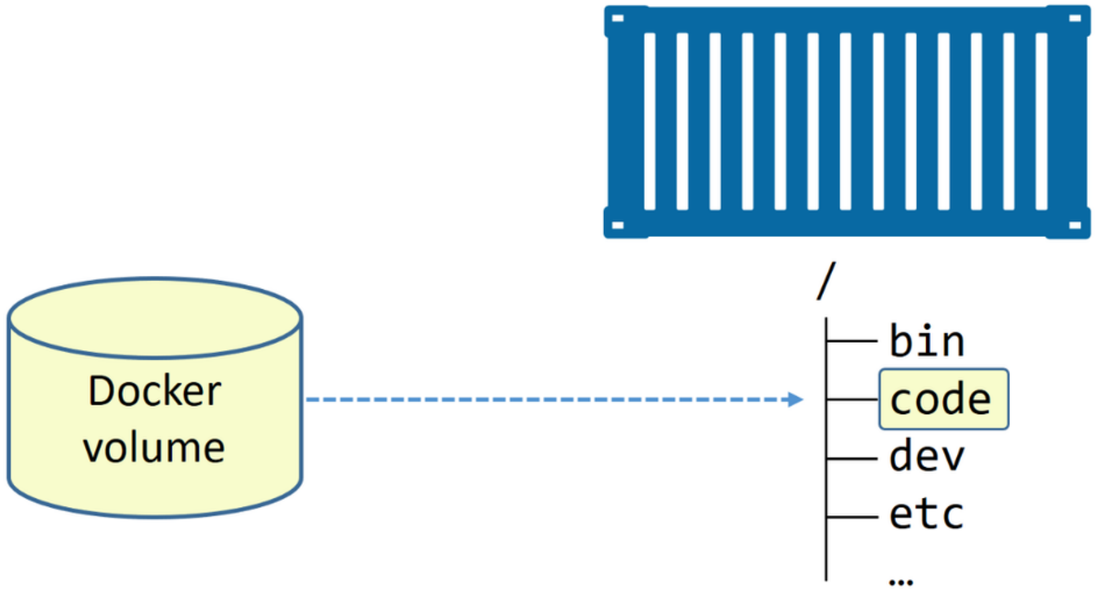
```
drwx------ 5 root root 4096 Jun 10 17:31 4cc44aeb81269d4846d97c9faa2d05c3d3d0cd83a0ba15294fc05f1603fcff8d
drwx------ 4 root root 4096 Jun 10 17:31 4cc44aeb81269d4846d97c9faa2d05c3d3d0cd83a0ba15294fc05f1603fcff8d-init
drwx------ 3 root root 4096 Jun 10 16:39 ebd750fd063cfc0ad3d9ebe832f2d00c092af6e083adfd33de2beb4df1d4b0a8
drwx------ 3 root root 4096 Jun 10 16:34 f929a3e3604b369ab95e7a5de6f7a339a81e4b5bd5017a43b4321d27c17498b9
drwx------ 2 root root 4096 Jun 10 17:31 l
[root@ip-172-31-5-176 overlay2]# cd 4cc44aeb81269d4846d97c9faa2d05c3d3d0cd83a0ba15294fc05f1603fcff8d
[root@ip-172-31-5-176 4cc44aeb81269d4846d97c9faa2d05c3d3d0cd83a0ba15294fc05f1603fcff8d]# ll
total 20
drwxr-xr-x 3 root root 4096 Jun 10 17:31 diff
-rw-r--r-- 1 root root   26 Jun 10 17:31 link
-rw-r--r-- 1 root root   57 Jun 10 17:31 lower
drwxr-xr-x 1 root root 4096 Jun 10 17:31 merged
drwx------ 3 root root 4096 Jun 10 17:31 work
[root@ip-172-31-5-176 4cc44aeb81269d4846d97c9faa2d05c3d3d0cd83a0ba15294fc05f1603fcff8d]# cd diff/
[root@ip-172-31-5-176 diff]# ll
total 4
drwxrwxrwt 2 root root 4096 Jun 10 17:33 tmp
[root@ip-172-31-5-176 diff]# cd tmp/
[root@ip-172-31-5-176 tmp]# ll
total 4
-rw-r--r-- 1 root root 32 Jun 10 17:33 hello.txt
[root@ip-172-31-5-176 tmp]# cat hello.txt
this is a file inside container
[root@ip-172-31-5-176 tmp]#
```

Persistent data

Recommended way to persist data in containers is with volumes.

You create a volume, then you create a container, & you mount the volume into it. The volume gets mounted to a directory in the container's filesystem, and anything written to that directory is written to the volume. If you then delete the container, the volume and its data will still exist.

Docker volume mounted into a container at /code . Any data written to the /code directory will be stored on the volume and will exist after the container is deleted.



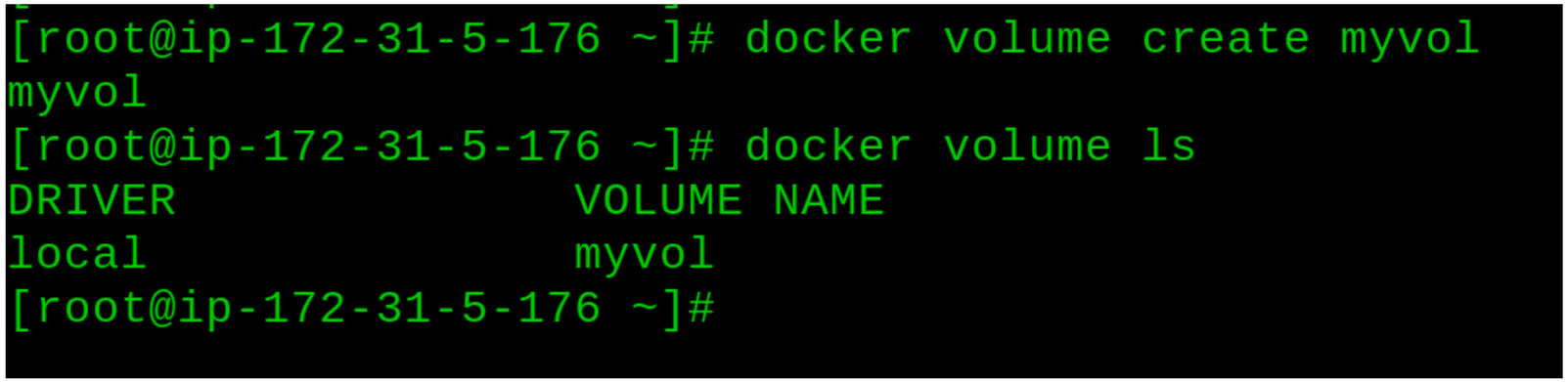
/code directory is a Docker volume. All other directories use the containers ephemeral local storage. The arrow from the volume to the /code directory is a dashed line to represent the decoupled relationship between volumes and containers.

Creating & managing docker volumes

Lets create a new volume called myvol.

 # docker volume create myvol

By default, Docker creates new volumes with the built-in local driver. As the name suggests, local volumes are only available to containers on the node they're created on. Use the -d flag to specify a different driver.



We can get more information about volume with docker volume inspect command.

```
[root@ip-172-31-5-176 ~]# docker volume inspect myvol
[
    {
        "CreatedAt": "2020-06-10T17:46:09Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/myvol/_data",
        "Name": "myvol",
        "Options": {},
        "Scope": "local"
    }
]
[root@ip-172-31-5-176 ~]#
```

The driver and scope are both local . This means the volume was created with the default local driver, and is only available to containers on this Docker host.

There are two ways to delete a Docker volume:

- docker volume prune
- docker volume rm

docker volume prune will delete all volumes that are not mounted into a container or service replica, so use with caution!

docker volume rm lets you specify exactly which volumes you want to delete.

Neither command will delete a volume that is in use by a container or service replica.

Now create a new standalone container and mount a volume called myvol.

```
[root@ip-172-31-5-176 ~]#
[root@ip-172-31-5-176 ~]# docker container run -dit --name=mybox \
> --mount source=myvol,target=/vol centos
1350c1a28b09bb8b31442cc2c0dcd50f2613341af601abb72b8741c28d3f9b8d
[root@ip-172-31-5-176 ~]#
[root@ip-172-31-5-176 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
1350c1a28b09        centos              "/bin/bash"         6 seconds ago       Up 5 seconds
[root@ip-172-31-5-176 ~]#
[root@ip-172-31-5-176 ~]#
```

Let’s exec onto the container and write some data to it.

```
[root@ip-172-31-5-176 ~]#
[root@ip-172-31-5-176 ~]# docker exec -it mybox /bin/bash
[root@1350c1a28b09 /]# df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay         7.9G  1.7G  6.1G  22% /
tmpfs            64M     0   64M   0% /dev
tmpfs           997M     0  997M   0% /sys/fs/cgroup
shm              64M     0   64M   0% /dev/shm
/dev/xvda1      7.9G  1.7G  6.1G  22% /vol
tmpfs           997M     0  997M   0% /proc/acpi
tmpfs           997M     0  997M   0% /proc/scsi
tmpfs           997M     0  997M   0% /sys/firmware
[root@1350c1a28b09 /]#
```

Let’s exec onto the container and write some data to it.

```
[root@1350c1a28b09 vol]# pwd
/vol
[root@1350c1a28b09 vol]#
[root@1350c1a28b09 vol]#
[root@1350c1a28b09 vol]# cat > hello.txt
this is a file inside volume attached
to a container
[root@1350c1a28b09 vol]#
[root@1350c1a28b09 vol]# ll
bash: ll: command not found
[root@1350c1a28b09 vol]# ls
hello.txt
[root@1350c1a28b09 vol]#
[root@1350c1a28b09 vol]#
```

Type exit to return to the shell of your Docker host, and then delete the container with the following command.

```
[root@1350c1a28b09 vol]#
[root@1350c1a28b09 vol]# exit
exit
[root@ip-172-31-5-176 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
1350c1a28b09        centos              "/bin/bash"         7 minutes ago
[root@ip-172-31-5-176 ~]# docker container stop mybox
mybox
[root@ip-172-31-5-176 ~]# docker container rm mybox
mybox
[root@ip-172-31-5-176 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
[root@ip-172-31-5-176 ~]#
[root@ip-172-31-5-176 ~]#
```

Because the volume still exists, you can look at its mount point on the host to check if the data we wrote is still there.

```
[root@ip-172-31-5-176 ~]#
[root@ip-172-31-5-176 ~]# docker volume inspect myvol
[
    {
        "CreatedAt": "2020-06-10T18:04:52Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/myvol/_data",
        "Name": "myvol",
        "Options": {},
        "Scope": "local"
    }
]
[root@ip-172-31-5-176 ~]# cd /var/lib/docker/volumes/myvol/_data/
[root@ip-172-31-5-176 _data]# ls
hello.txt
[root@ip-172-31-5-176 _data]# cat hello.txt
this is a file inside volume attached
to a container
[root@ip-172-31-5-176 _data]#
```

It’s even possible to mount the myvol volume into a new service or container.

WHAT WE LEARNED

- Two main types of data: persistent and non-persistent data.
- However, if your containers create data that you need to keep, you should store the data in a Docker volume.
- Docker volumes are managed independently of containers with their own docker volume sub-command. This means that deleting a container will not delete the volumes it was using.
- Volumes are the recommended way to work with persistent data in a Docker environment.