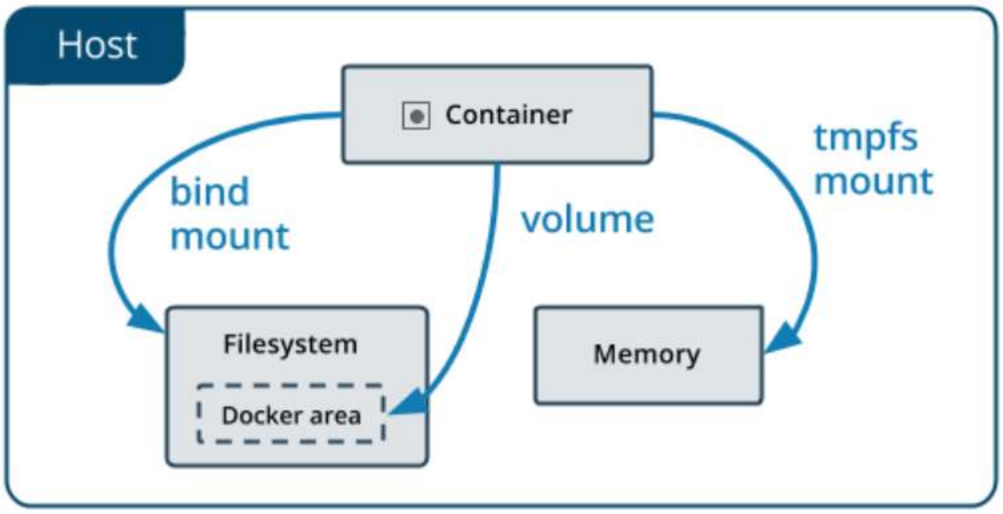
Managing container and image data with Docker Volumes and Storage drivers

Managing application data in running containers is one of the biggest challenges. We have to trade of between available volume types (and its associated storage drivers) with system performance.

Volume's contents exist outside the lifecycle of a given container. This is called data persistency.

Below snapshot depicts the mechanism for persistent data in containers.



- **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.

  Example: Start ngnix container with a volume

  ```
  $docker run -d --mount type=volume,source=myvol,target=/app ngnix:latest
  ```

- **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.

  Example: Start ngnix container with bind mount

  ```
  $docker run -d --mount type=bind,source=/techieglobus/myvol,target=/app nginx:latest
  ```

- **tmpfs mounts** are stored in the host system's memory only, and are never written to the host system's filesystem.

  Example: Use a tmpfs mount in ngnix container

  ```
  $docker run -d --mount type=tmpfs,destination=/app nginx:latest
  ```

Docker storage drivers help organise and manage containers data in background. It’s very tricky to choose what kind of storage data suits your requirements. While selecting the storage driver we have to trade-off between stability, workload and performance.

Storage drivers act differently with different OS and File Systems. Below table depicts same.

Storage Driver	**Type**	**Kernel Support**	**Quota**
AUFS	File System	No	No
BTRFS	Block	Yes	Yes
DeviceMapper	Block	Yes	Yes
Overlay	File System	Yes	No
Overlay2	File system	Yes	No
ZFS	Block	No	Yes

*Note: Overlay2, AUFS and ZFS are considered easy to work on.

Docker storage drivers come as pluggable modules. These modules can be installed, enabled, disabled and deleted at any point of time. We lose containers’ inside data when we change the storage driver types on running host.

Example: Select overlay2 as docker storage driver.

```
#service docker stop
```

```
#echo ‘{ “storage-driver”: “overlay2” }’ >> /etc/docker/daemon.json
```

Note: Even while creating volumes we can select what storage-driver they should use.

Example: Create a volume called myvol and select storage-driver as btrfs.

```
#docker volume create --name myvol --driver btrfs
```

(Before executing above command make sure btrfs storage driver plugin should be installed)