Docker ofrece la posibilidad de usar diferente sistemas de ficheros para el almacenamiento de los contenedores y toda la información necesaria para correr Docker en una máquina. Estos son:

- aufs
- btrfs
- devicemapper
- vfs
- overlayfs
- zfs (recientemente zfs)

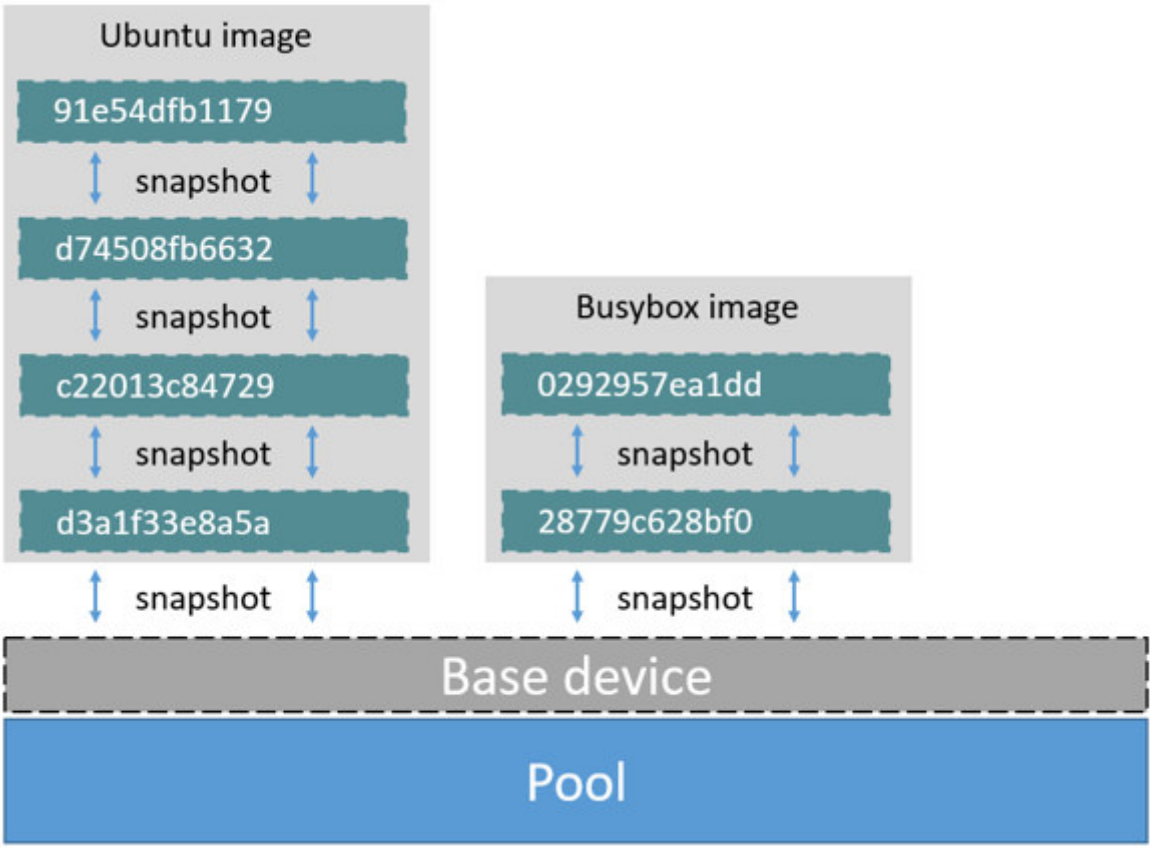
Para utilizar cualquiera de estos sistema de ficheros necesitamos pasar el siguiente argumento cuando arranquemos nuestro Docker daemon '–storage-driver='.

```
$ sudo docker daemon --storage-driver=STORAGE_DRIVER &
```

Aufs: Este sistema hace uso del sistema de ficheros Aufs union. Este sistema no está soportado en la mayoría de distros y por lo tanto no está recomendado su uso en producción.

Este almacena cada docker layer como un directorio normal y corriente, conteniendo ficheros y metadata de aufs. Este sistema de ficheros combina todos los layers en un unico mountpoint. Funciona como una pila donde cualquier cambio en este mountpoint va encima del primer layer.

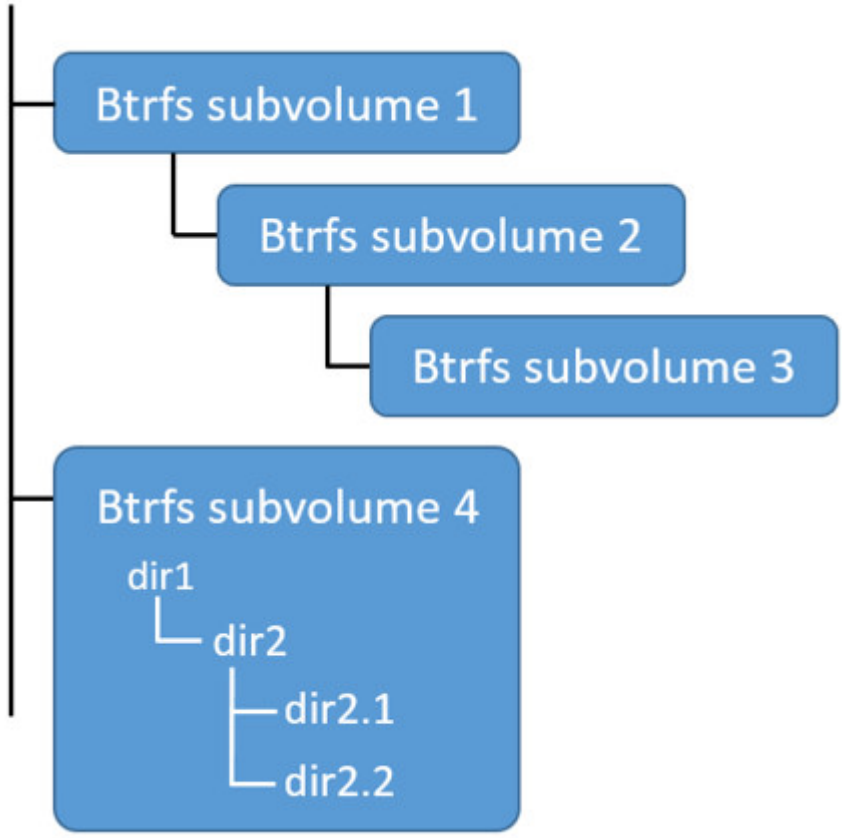
- `/var/lib/docker/aufs/mnt/diff` – donde la imagen layer y su contenido son almacenadores.
- `/var/lib/docker/aufs/layers/` – metadata sobre como los layers están apilados (organizados).
- `/var/lib/docker/aufs/mnt/<container-id>` – donde los contenedores son almacenados.



Típica imagen describiendo el uso de Aufs para contenedores en Docker

Btrfs: Este sistema usa snapshotting en el sistema de ficheros para crear los layers de las imagenes Docker. Este requiere, al igual que overlay, tener el directorio '/var/lib/docker' sobre un sistema de ficheros de tipo btrfs. Cada layer es almacenado como un subvolumen dentro del directorio '/var/lib/docker/btrfs/subvolumes' y en la forma de snapshot de un subvolumen padre. La siguiente imagen ilustra un poco mejor cómo funciona este sistema de ficheros.

- `/var/lib/docker/btrfs/subvolumes` – donde los volumes son visibles como un sistema ficheros
- `$ sudo btrfs subvolume list /var/lib/docker`



En comparación con otros este sistema de ficheros es muy rápido pero tiene bastantes problemas de estabilidad debido a su escasa madurez.

Devicemapper: Este sistema de ficheros usa el device-mapper (dm-thinp) para crear layers. Device-mapper es la parte del kernel de linux llamada LVM2 volumenes lógicos del sistema. Es un sistema basado en bloques copy-on-write que basicamente usa dos bloques, uno para datos y el otro para metadatos de los dispositivos. Devicemapper crea un pool que puede ser usado para crear otros bloques a partir de este pool. Estos bloques que usa el pool están inicialmente vacíos y las partes no usadas no son reservadas para el uso de este bloque. Permite copy-on-write snapshotting de un dispositivo para crear uno nuevo.

Docker crea un dispositivo base en el pool conteniendo un sistema de ficheros ext4 vacío. El resto de layers seran snapshots del layer base. El sistema de ficheros tiene un tamaño fijo (por defecto 10Gb) con lo cual todo los containers y imágenes tienen un tamaño máximo.

El siguiente comando permite ver todos los dispositivos que devicemapper ha creado.

```
$ sudo lsblk
```

Lista de directorios destacables:

- `/var/lib/docker/devicemapper/devicemapper` – contiene data y metadata bloques devices creados en el pool usando la técnica de loopback.
- `/var/lib/docker/devicemapper/devicemapper/json` – contiene la información que permite mapear los layers con los ids en el pool.

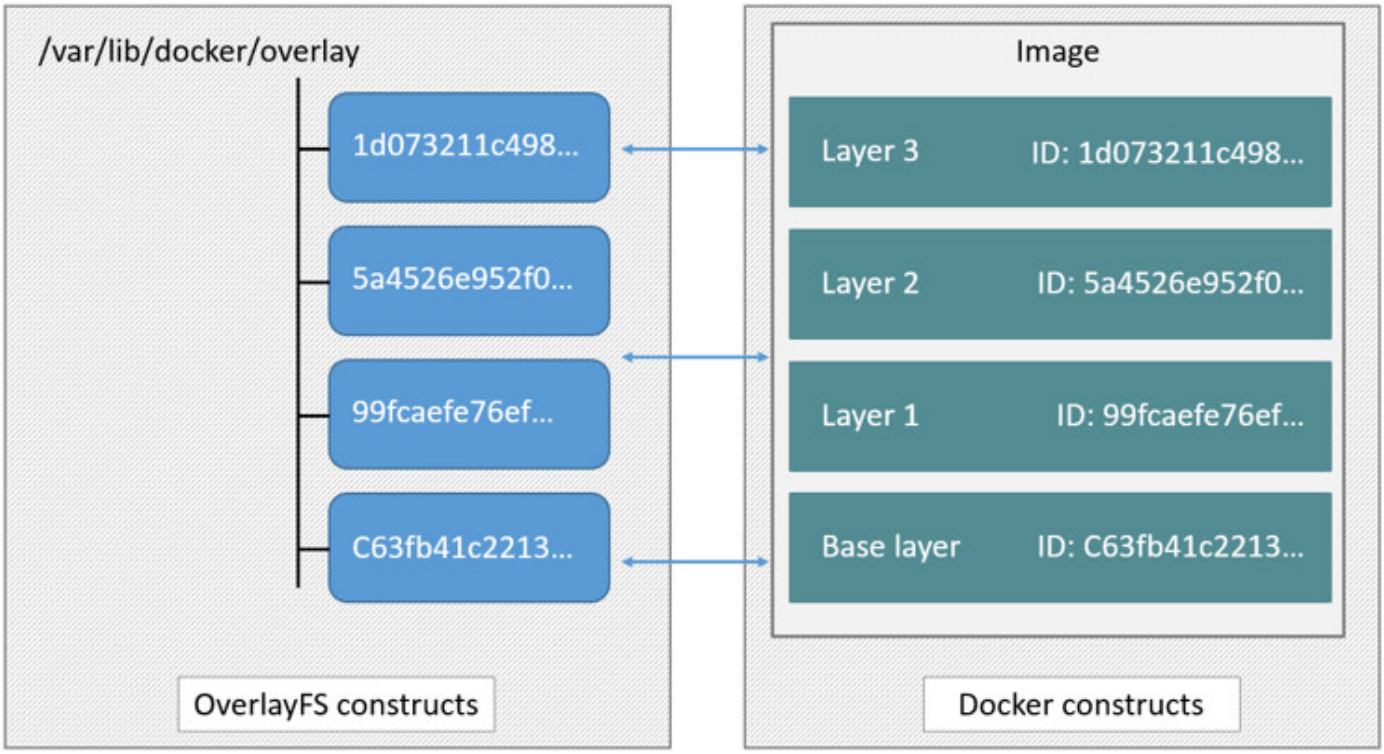
Vfs: Este sistema de ficheros usa un sencillo mecanismo de fallback que no soporta copy on write. Cada layer is un directorio a parte, cuando creamos a nuevo layer otro

layer es creado como un clon del primero en un nuevo directorio. La creación de layer es muy costosa a nivel de performance aunque es uno de los más robustos que funciona en cualquier sistema operativo. Este no comparte el uso de disco compartido entre layers lo cual es un problema a la hora de gestionar los layers de una imagen Docker.

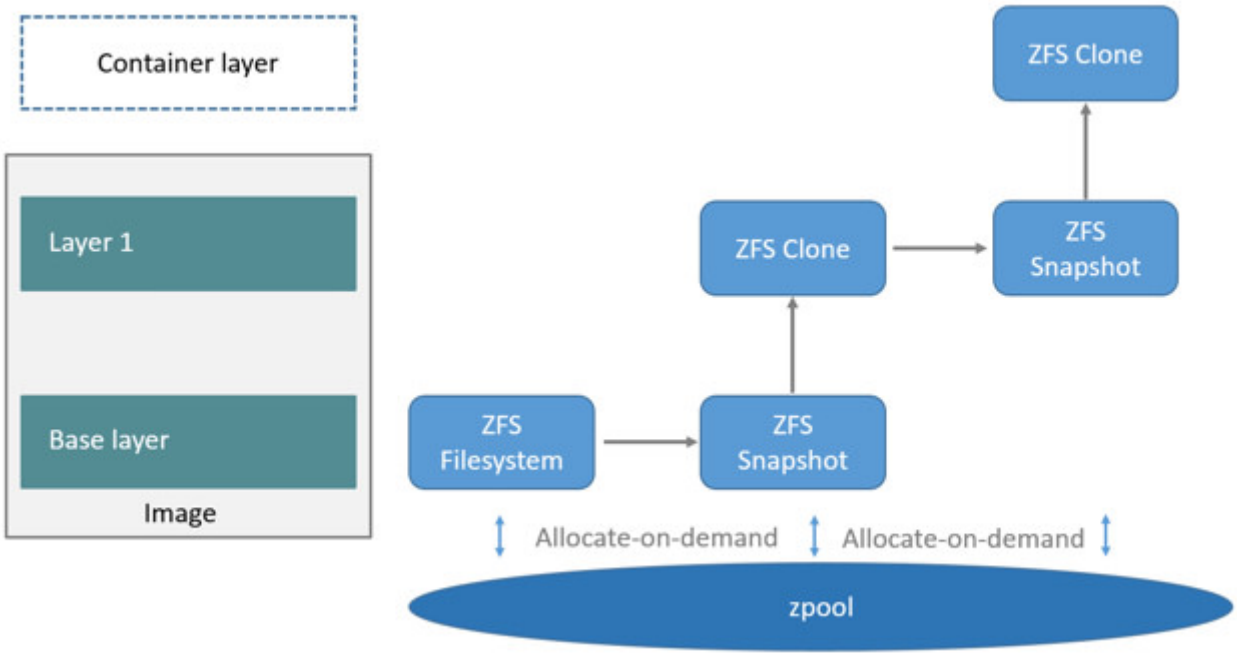
OverlayFS: Es visto como el sucesor de aufs aunque todavía es algo inmaduro para ser adoptado en producción. OverlayFS maneja un concepto basado en dos directorios en el cual uno se apoya sobre el otro mostrando una visión unificada de los layers de una imagen Docker. Para conseguir este efecto OverlayFS usa una tecnología llamada union mount. Los nombres de los directorios son `upperdir` , `lowerdir` mientras que la versión unificada es expuesta como un directorio llamada `merged` .

```
$ ls -l /var/lib/docker/overlay/
```

En OverlayFS cada image layer representa un directorio en la `/var/lib/docker/overlay` . Para crear un contenedor, overlay combina el directorio que representa el último layer con un nuevo directorio para el contenedor. Así el layer de la imagen es el 'lowerdir' en modo lectura, y el nuevo directorio para el contenedor es el 'upperdir' en modo lectura/escritura.



Zfs: Este eficiente sistema de ficheros combina snapshotting y clonado de layers. Las snapshots tiene acceso de lectura y los clones tienen acceso de lectura/escritura. Clones solo pueden ser creados a partir del snapshot de un layer. En Docker una imagen base forma un sistema de ficheros ZFS, donde sus hijos son clones de un snapshot del layer situado más abajo. Consequentemente, un contenedor es un clon basado en una snapshot del layer situado en la cola de la imagen a partir de la cual fue creado.


Container layer
Layer 1
Base layer
Image
ZFS Clone
ZFS Clone
ZFS Snapshot
ZFS Filesystem
ZFS Snapshot
Allocate-on-demand
Allocate-on-demand
zpool