Docker Storage

Shreyas Dhareshwar
Micro Focus
13th October 2018

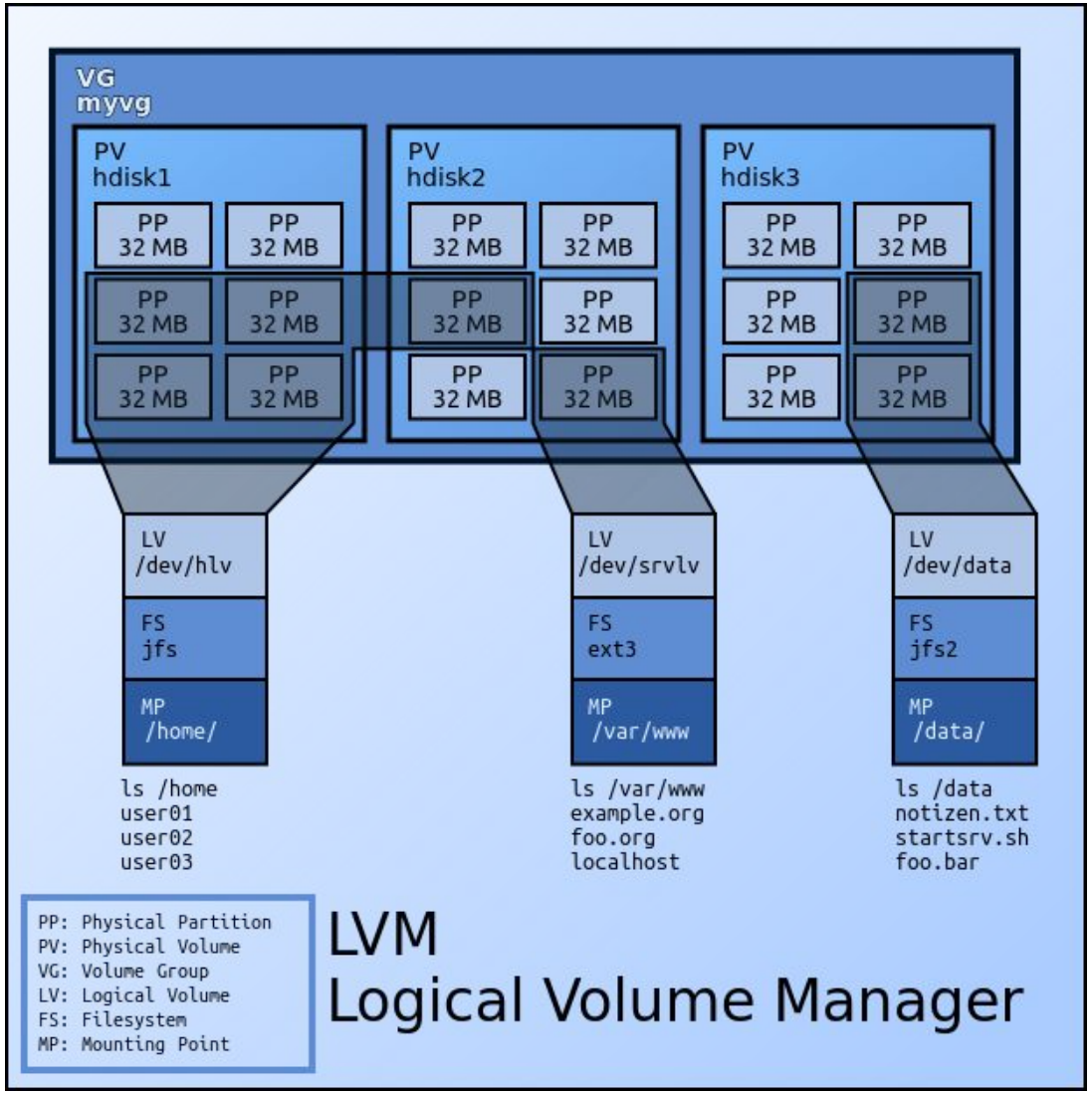
Agenda

- ***Brush-up on Docker basics***
- ***Linux Volumes***
- ***AUFS and Default Docker File System***
- ***Types of Docker mount***
  - ***Volumes***
  - ***Bind mounts***
  - ***Tempfs***
- ***Hands-on***

Brush-up on Docker basics

- docker run hello-world, docker pull and docker images
- docker run ubuntu bash
- docker run -idt ubuntu bash , install vi and docker commit
- docker exec –it «container_id»
- docker run --name web -d -p 3000:80 nginx
- docker run --name web1 -d -p 3100:80 --mount type=volume,source=nginx-vol,destination=/usr/share/nginx/html nginx
- docker run --name web2 -d -p 3200:80 -v /root/Public:/usr/share/nginx/html nginx
- docker rm -f $(docker ps -qa)

Linux Volumes

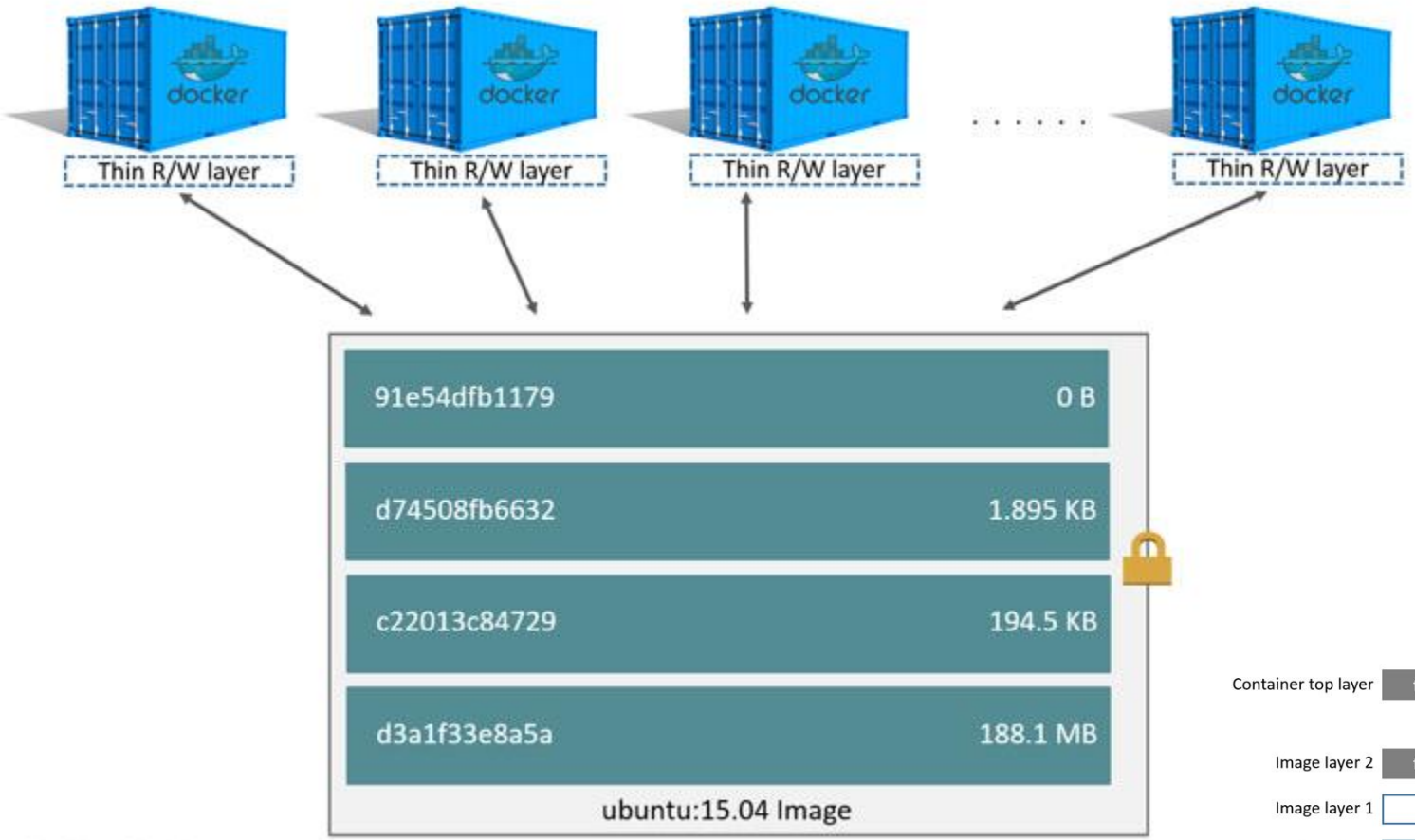


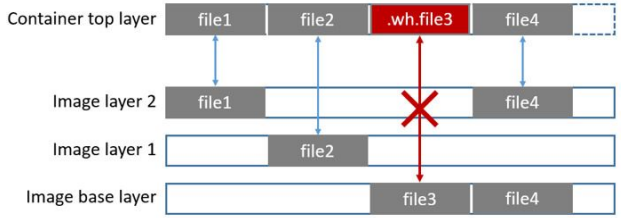
Storage Drivers (AUFS, overlay, etc)

1. Create a container: docker run --name web -d -p 3000:80 nginx
2. Connect to container and explore file system (/usr/share/nginx/html)
3. Check physical file system on host (/var/lib/docker/)
4. Create file on container
5. Find the file on the host
6. Check volume diff of the container
7. Kill and remove the container

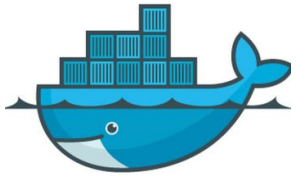
What happens to the container's file system?

AUFS


Thin R/W layer
Thin R/W layer
Thin R/W layer
Thin R/W layer
91e54dfb1179 0 B
d74508fb6632 1.895 KB
c22013c84729 194.5 KB
d3a1f33e8a5a 188.1 MB
ubuntu:15.04 Image

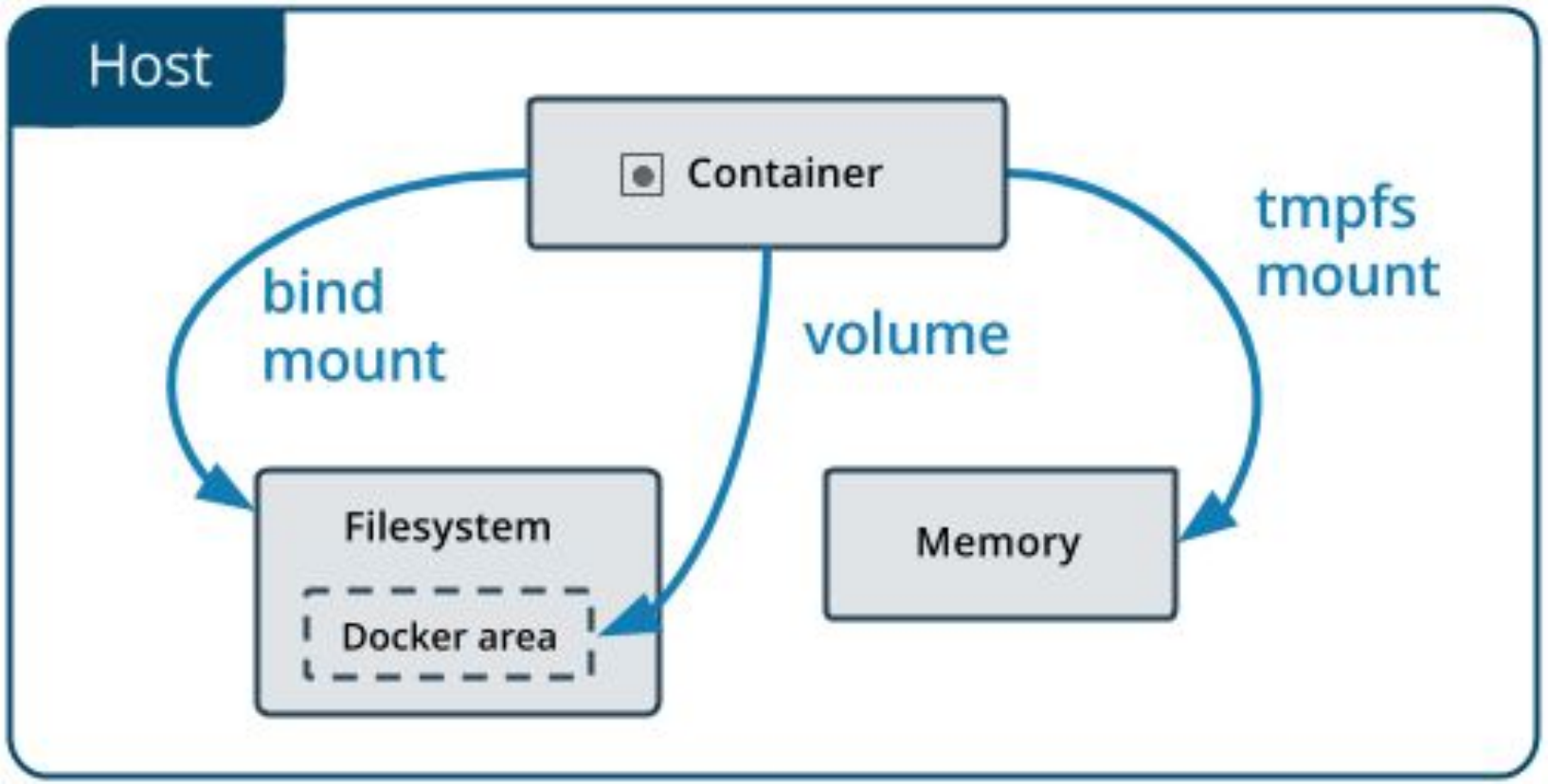

Container top layer
file1
file2
.wh.file3
file4
Image layer 2
file1
file4
Image layer 1
file2
Image base layer
file3
file4

Default Docker file system

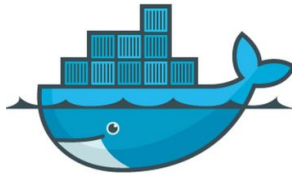


1. “Writable container layer”
2. No static path from outside the container
3. No container, no data
4. Tightly coupled with host machine
5. Union FS - slower performance

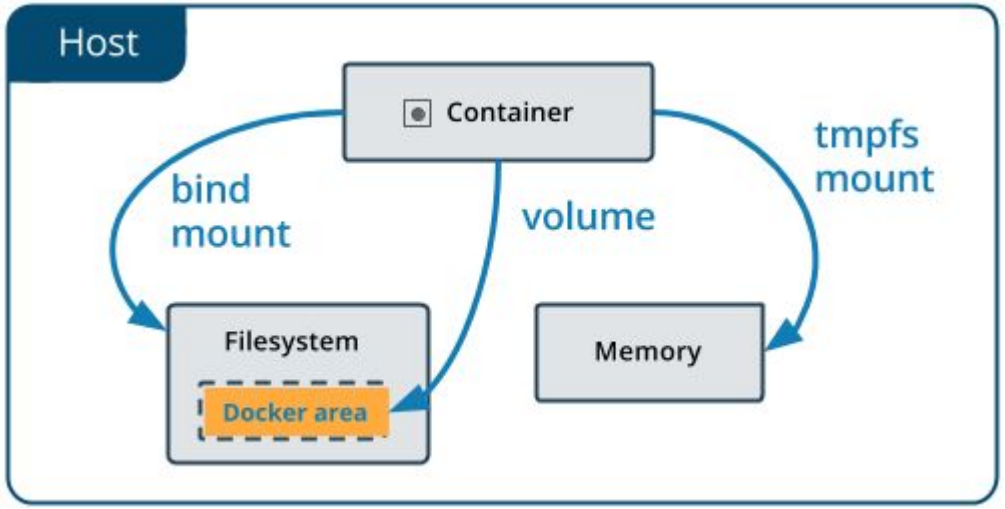
Types of Docker mount



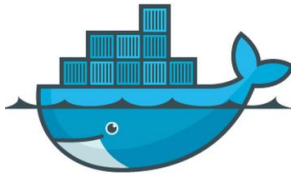
Volumes



- Created and managed by Docker
  - Explicitly (`docker volume create`)
  - Implicitly (during container creation)
- Can be Named or Anonymous
- Supports volume drivers
- Cleanup possible (`docker volume prune`)

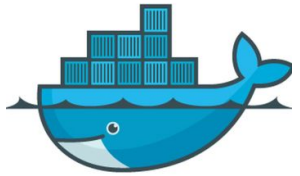


Volumes – Use cases

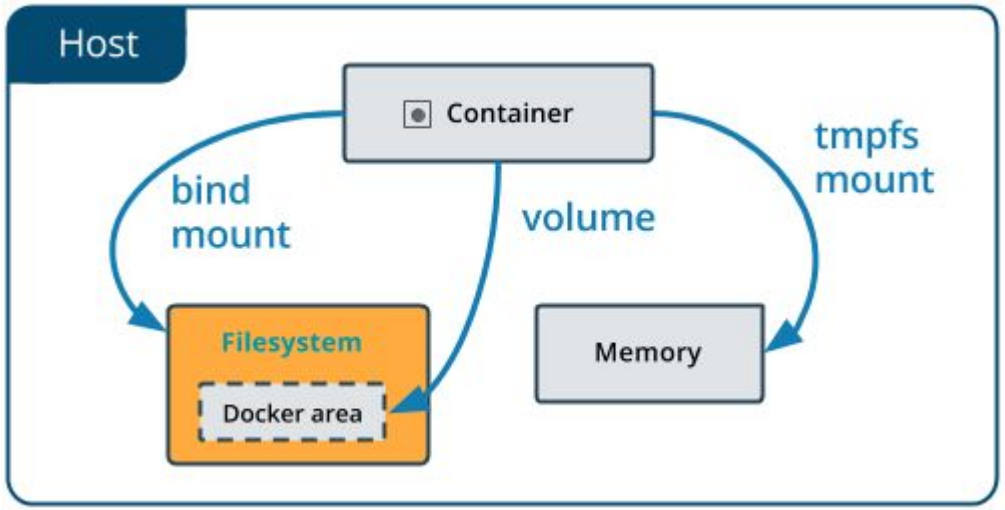


- Persistency across container lifecycle
- Sharing data across containers (say across 2 nginx's)
- Decoupled host-container file system architecture
- Storing data outside the host (central storage or cloud)
- Backup and restore

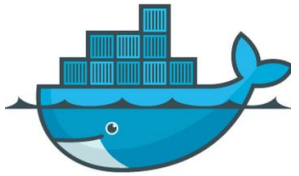
Bind mounts



- Created by docker if needed
- Maintained by host
- Can have security implications
- Performant
- Needs specific dir structure on host

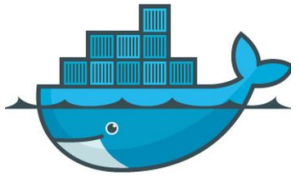


Bind mounts – Use cases

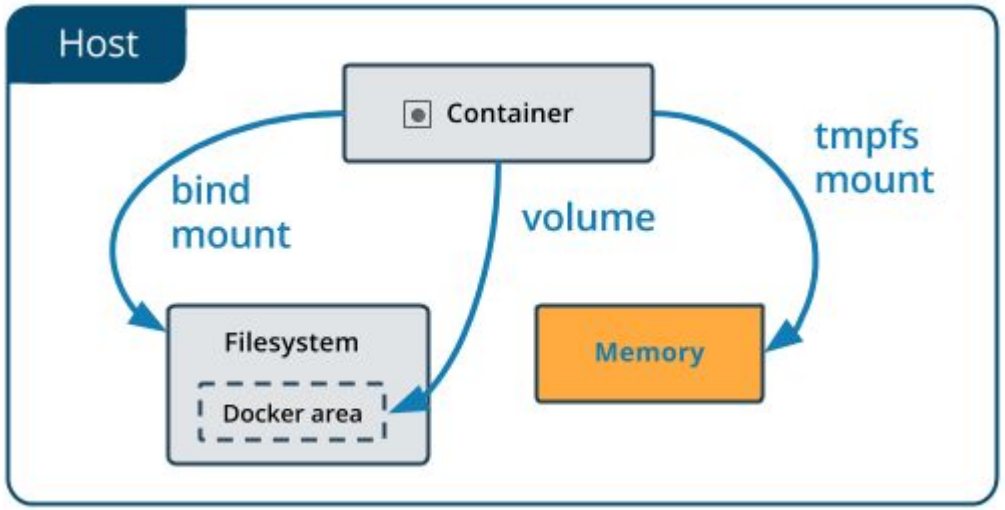


- Sharing config from host
- DevOps build lifecycle – *target* folder into container
- Persistency across container lifecycle
- Sharing data across containers (say across 2 nginx's)

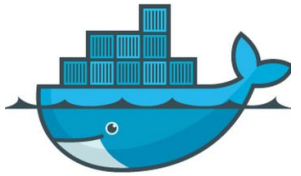
Tempfs



- Only in memory!
- Great for storing sensitive data
- Extremely fast
- No cleanup needed

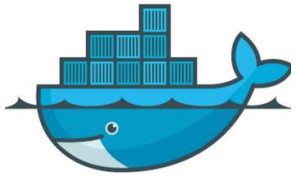


Tempfs – Use cases



- I/O sensitive projects
- Standalone containers with need to store runtime info

Reference material



- https://docs.docker.com/storage/

Thank You.