Docker and Kubernetes

Session 4

Docker storage

Agenda

- Assignment Review & Guides
- Storage overview
- Docker volumes
- Bind mounts
- Store data within containers

Manage data in Docker

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
- A container's writable layer is tightly coupled to the host machine where the container is running
- Writing into a container's writable layer requires a storage driver to manage the filesystem. This extra abstraction reduces performance as compared to using ***data volumes***

Manage data in Docker

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
- A container's writable layer is tightly coupled to the host machine where the container is running
- Writing into a container's writable layer requires a storage driver to manage the filesystem. This extra abstraction reduces performance as compared to using ***data volumes***

Choose the right type of mount

- **Volumes** are stored in a part of the host filesystem which is managed by Docker (/var/lib/docker/volumes/ on Linux).
- **Bind mounts** may be stored anywhere on the host system. They may even be important system files or directories.
- **tmpfs** mounts are stored in the host system's memory only, and are never written to the host system's filesystem.

Docker Volumes

- **Volumes** mount a directory on the host into the container at a specific location
- Can be used to share (and persist) data between containers
- Directory persists after the container is deleted (Unless you explicitly delete it)
- Can be created in a Dockerfile or via CLI

Use volumes

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While bind mounts are dependent on the directory structure and OS of the host machine, volumes are completely managed by Docker. Volumes have several advantages over bind mounts

Create and manage volumes

Create a volume:

```
$ docker volume create my-vol
```

List volumes:

```
$ docker volume

Local               my-vol
```

Inspect a volume:

```
$ docker volume inspect my-vol
```

Remove a volume:

```
$ docker volume rm my-vol
```

Start a container with a volume

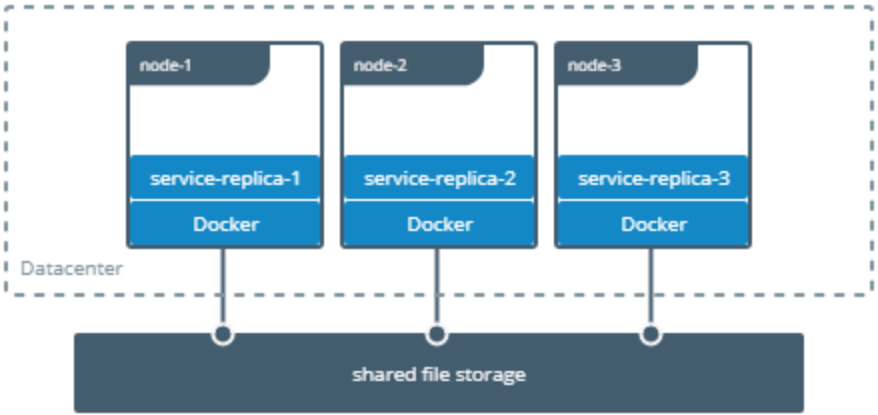
If you start a container with a volume that does not yet exist, Docker creates the volume for you

```
$ docker run -d \
  --name devtest \
  -v myvol2:/app \
  nginx:latest
```

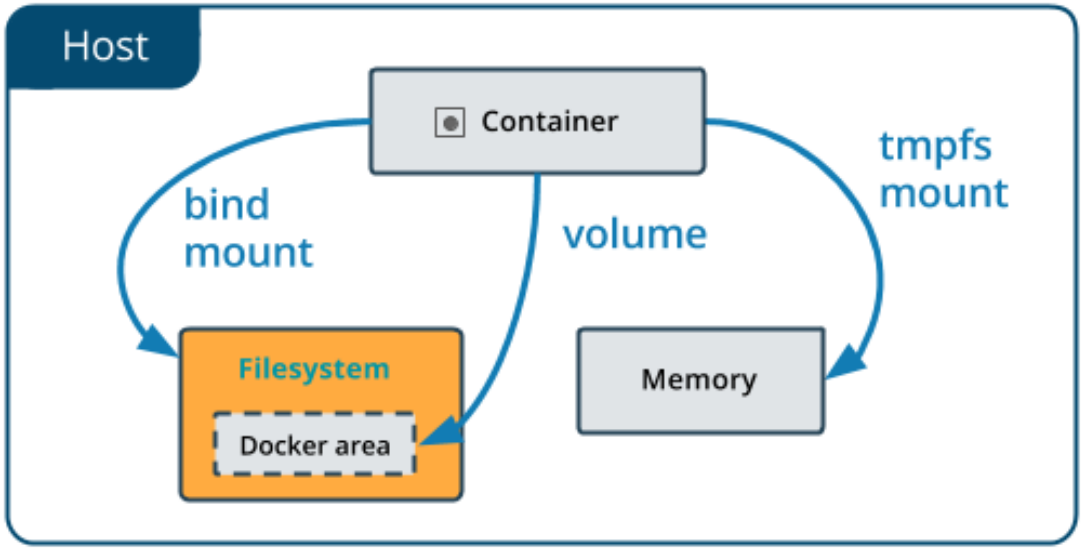
Use docker inspect devtest to verify that the volume was created and mounted correctly

Share data among machines

When building fault-tolerant applications, you might need to configure multiple replicas of the same service to have access to the same files.



Docker bind mount



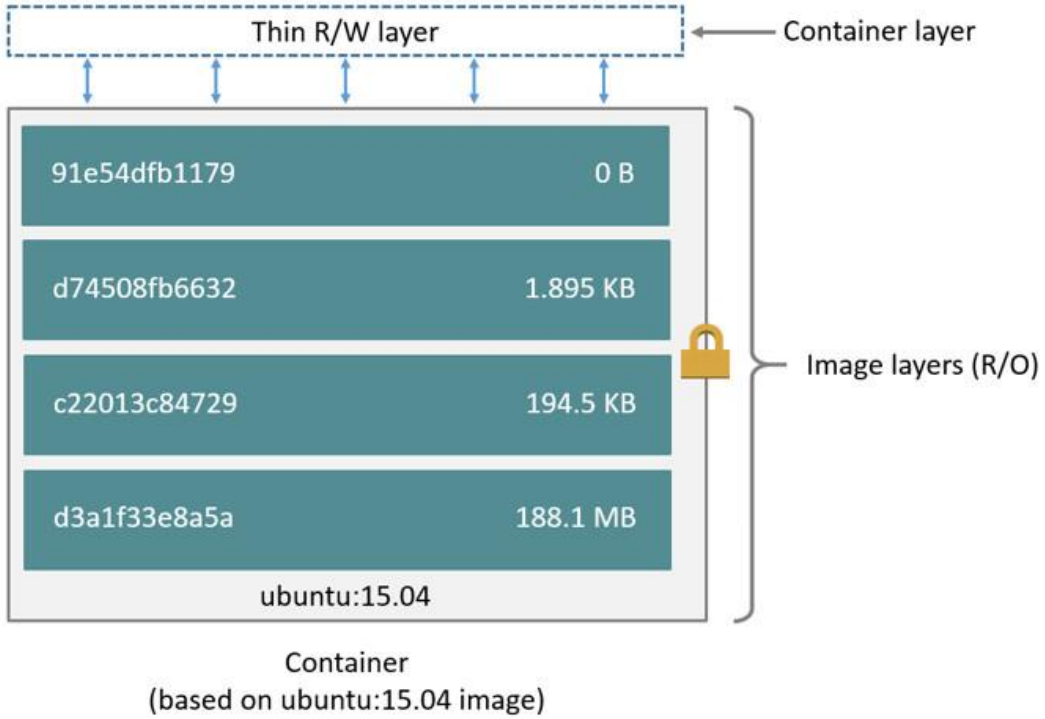
Use tmpfs mounts

Volumes and **bind mounts** let you share files between the host machine and container so that you can persist data even after the container is stopped.

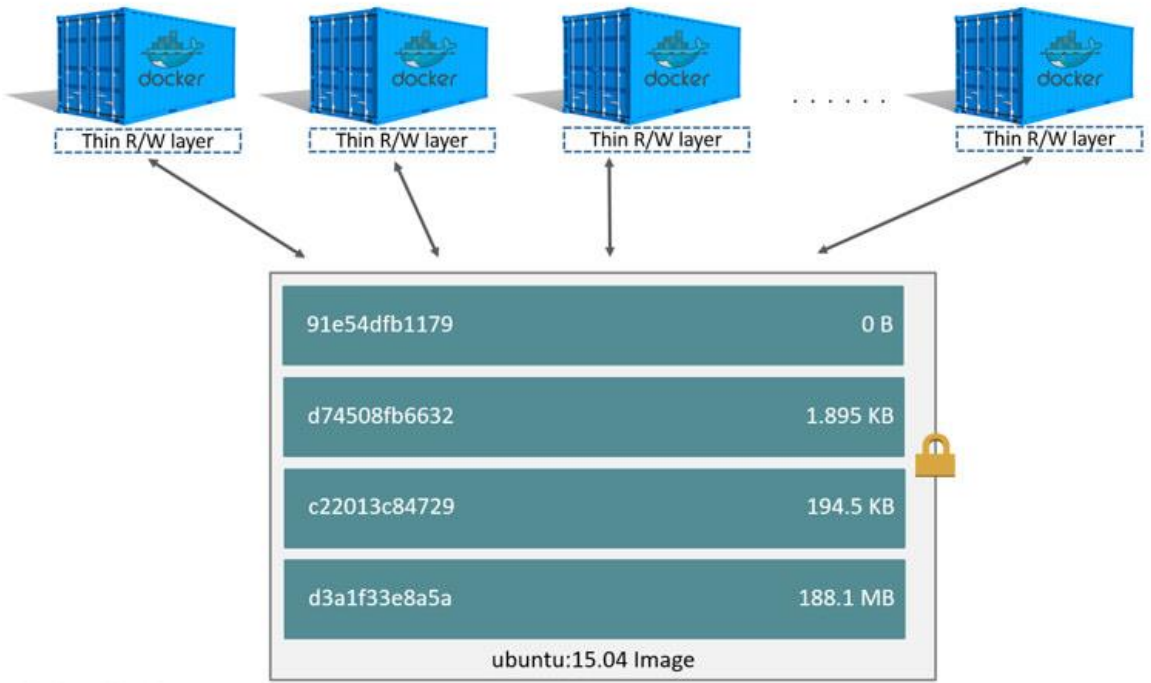
If you're running Docker on Linux, you have a third option: tmpfs mounts. When you create a container with a tmpfs mount, the container can create files outside the container's writable layer.

Storage data within container

A ***storage driver*** handles the details about the way these layers interact with each other. Different storage drivers are available, which have advantages and disadvantages in different situations.



Container and layers



Why Use Volumes

- Mount local source code into a running container

  docker container run -v $(pwd):/usr/src/app/ myapp

- Improve performance

  As directory structures get complicated traversing the tree can slow system performance

- Data persistence

PRACTICE DOCKER STORAGE

Q&A