#04 practical class

Docker Volumes and Networks

COMPUTING SYSTEMS AND INFRASTRUCTURES

(SISTEMAS E INFRAESTRUTURAS DE COMPUTAÇÃO)

Overview

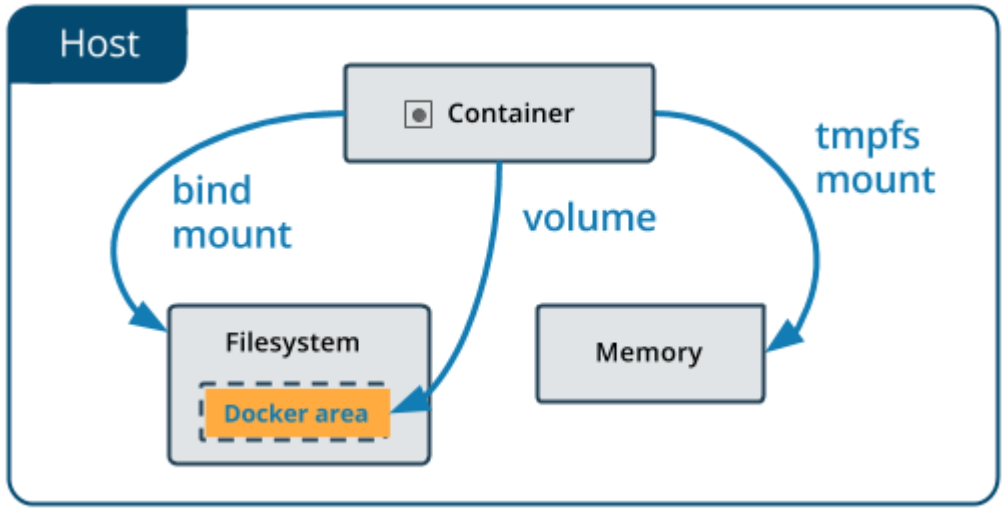
- Saving local images
- Persistent storage
- Network
- Exercise

Saving local images

- Saving a local image to a .tar file
  - `#>docker images`
  - `#>docker pull httpd:latest`
  - `#>docker save -o apache-server-portable.tar httpd:latest`
- Loading a .tar image
  - `#>docker rmi httpd:latest`
  - `#>docker load -i apache-server-portable.tar`
  - `#>docker images`

Containers storage

- **Bind mounts** may be stored anywhere on the host system
- **Volumes** are stored in a part of the host filesystem which is managed by Docker
- **tmpfs mounts** are stored in the host system's memory



From https://docs.docker.com

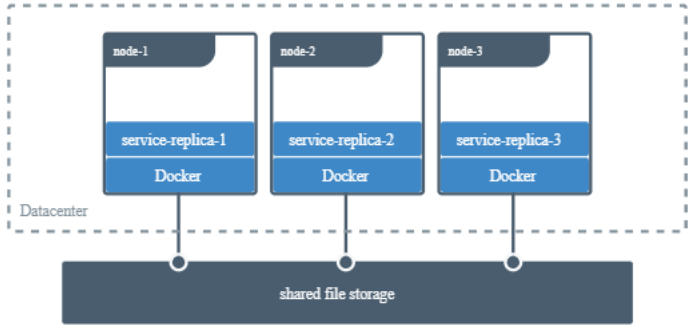
Bind mounts

- The file or directory is referenced by its absolute path on the host machine
- Very performant, but they rely on the host machine's filesystem
- If you bind-mount into a non-empty directory on the container, the directory’s existing contents are obscured by the bind mount
- Useful for
  - Configuration files
  - Development code (sync host/container)

```
docker run -dit --name apache-server \
  -p 8080:80 \
  -v /root/pl3/apache-data:/usr/local/apache2/htdocs \
  httpd:latest
```

Volumes

- Volumes can be managed using Docker CLI commands or the Docker API
- Volumes can be more safely shared among multiple containers
- Volume drivers support the storage of volumes on remote hosts or cloud providers and to encrypt the contents of volumes
- Volumes can be pre-populated by a container
- Several backup and restore options

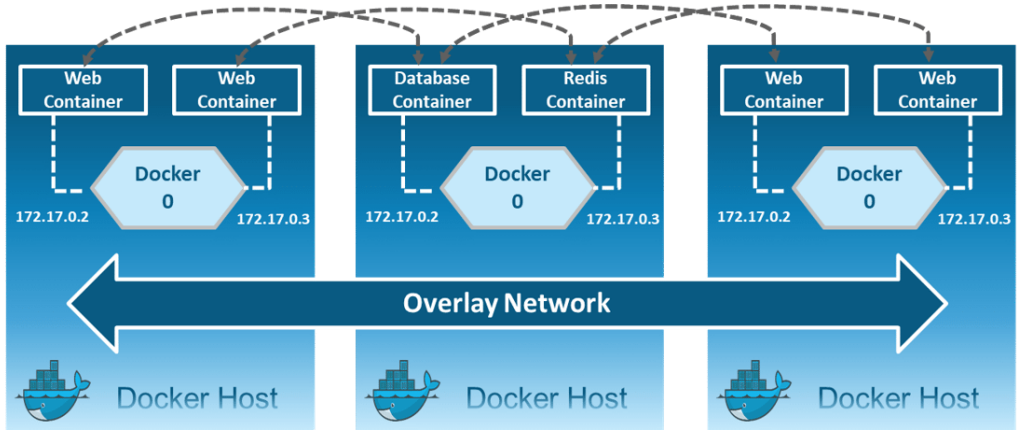



```
docker volume create apache-data

docker volume ls

docker volume inspect apache-data

docker run -dit --name apache-server \
  -p 8080:80 \
  -v apache-data:/usr/local/apache2/htdocs \
  httpd:latest
```

From https://docs.docker.com

Network

- **Bridge**: the default network driver
- **Host**: for standalone containers. Remove network isolation between the container and the Docker host
- **Overlay**: connect multiple Docker daemons together
- **Ipvlan:** total control over both IPv4 and IPv6 addressing with control of layer 2 VLAN tagging
- **Macvlan:** assign a MAC address to a container
- **None:** disable all networking
- **Network plugins:** third-party network plugins



From https://www.edureka.co/blog/docker-networking

Expose ports

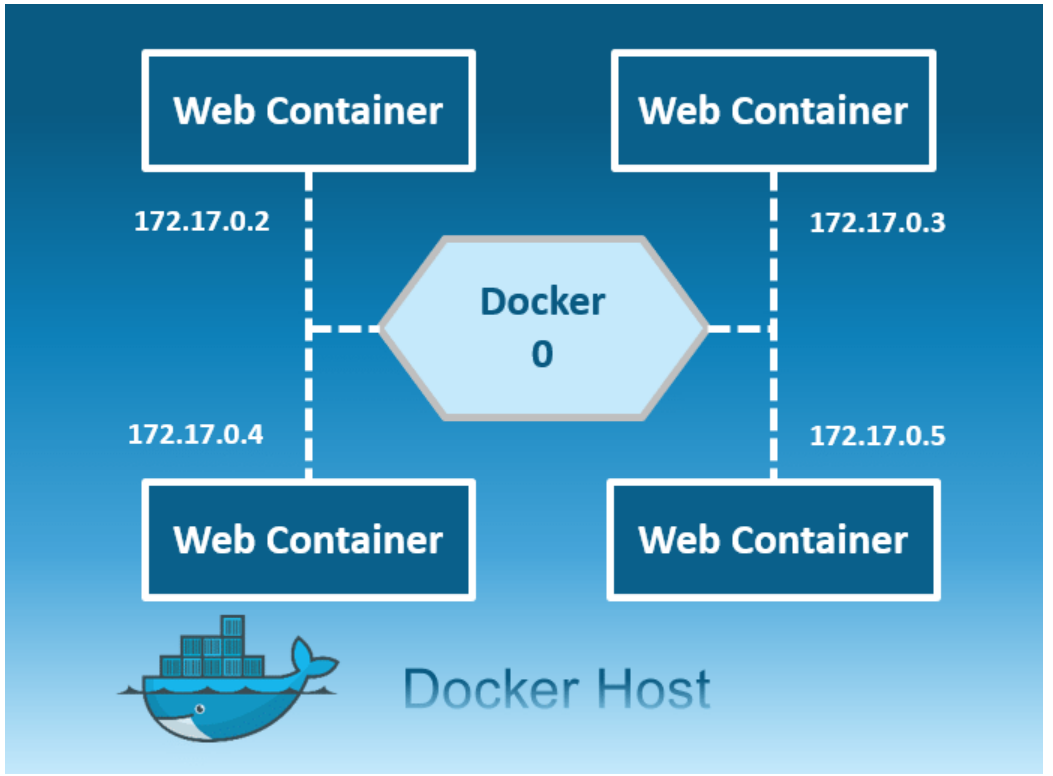
- Expose
  - Informs Docker that the container listens on the specified network ports at runtime
  - Does not make the ports of the container accessible to the host
  - Available only for inter-container communication
- Publish (-p)
  - Bind a container port with the host machine

```
docker image inspect httpd

docker run -dit --name apache-server2 \
  -p 8080:80 \
  httpd

docker port apache-server2
```

```
sic:~/storage# docker image inspect httpd
[
    {
        "Id": "sha256:f2789344c57324805883b174676365eb807fdb4eccfb9878fbb19054fd0c7b7e",
        "RepoTags": [
            "httpd:latest"
        ],
        "RepoDigests": [
            "httpd@sha256:71e882df50adc606c57e46e5deb3c933288e2c7775472a639326d9e4e40a47c2"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2022-09-13T13:16:23.927916429Z",
        "Container": "fe513212ea2e7c9b003a687d664197a88577493163acc3eedb86d28d9e998c4a",
        "ContainerConfig": {
            "Hostname": "fe513212ea2e",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "ExposedPorts": {
                "80/tcp": {}
            },
            "Tty": false,
```

Bridge network

- Default network driver
- Forwards traffic between network segments
- Allows containers connected to the same bridge network to communicate
- Provide isolation from containers that are not connected to the same bridge network



From https://www.edureka.co/blog/docker-networking

User-defined bridges

- User-defined bridges provide automatic DNS resolution between containers
- User-defined bridges provide better isolation
- Containers can be attached and detached from user-defined networks on the fly

```
docker network create my-net

docker network ls

docker network connect my-net apache-server2
```

```
sic:~/storage# docker network ls
NETWORK ID     NAME                          DRIVER    SCOPE
cc42688f5de9   bridge                        bridge    local
c9a3143b9d86   compose-env-test_default      bridge    local
3cac493e243f   host                          host      local
fc64db0f8d2b   my-net                        bridge    local
71e0391ef46f   none                          null      local
sic:~/storage#
```

Example

MariaDB & PHPMyAdmin

```
docker network create mariadb-network

docker run -dit --name mariadb \
    --network mariadb-network \
    -e MARIADB_ROOT_PASSWORD=my-secret-pw \
    -e MARIADB_DATABASE=testdb \
    -v /home/mariadb:/var/lib/mysql \
    mariadb:latest

docker run -dit --name phpmyadmin \
    --network mariadb-network \
    -e PMA_HOST=mariadb \
    -p 8080:80 \
    phpmyadmin
```

Exercise

MariaDB & PHPMyAdmin

- Create a docker compose to build a MariaDB and a PHPMySQL service
- Use a bridge network
  - mariadb-network
- Use a volume to store MariaDB data
  - mariadb-data