Container Volumes

Persistent storage for volatile containers

- Containers are volatile in nature because they are disposable, making changes in container(adding packages, configurations) are done through image
- The data doesn't persist when that container no longer exists and it can be difficult to get the data of the container if another process needs it.
- A container's writable layer is tightly coupled to the host machine where the container is running. You can't easily move the data somewhere else.
- Removing container deletes the data
- In the case of a stateful container such as mysql, stores database and reads from database, in such a case we have container volumes.

Docker has two options for containers to store files in the host machine
so that the files are persisted even after the container stops

Volumes
Bind Mounts

Use of Volumes

1. Decoupling container from storage
2. Share volume (storage/data) among different containers
3. Attach volume to container
4. On deleting container volume does not delete

By default all files created inside a container are stored on a writable container layer

Volumes and BIND mounts

- Volumes are stored in a part of the host filesystem which is managed by Docker

- Non-Docker processes should not modify this part of the filesystem
- Bind mounts may be stored anywhere on the host system
- Non-Docker processes on the Docker host or a Docker container can modify them at any time
- In Bind Mounts, the file or directory is referenced by its full path on the host machine.
- Volumes are the best way to persist data in Docker
- volumes are managed by Docker and are isolated from the core functionality of the host machine
- A given volume can be mounted into multiple containers simultaneously.
- When no running container is using a volume, the volume is still available to Docker and is not removed automatically. You can remove unused volumes using docker volume prune.
- When you mount a volume, it may be named or anonymous.
- Anonymous volumes are not given an explicit name when they are first mounted into a container
- Volumes also support the use of volume drivers, which allow you to store your data on remote hosts or cloud providers, among other possibilities.

docker volume {options} *volumeName*
(ls, create, inspect, prune, rm)

docker volume create devOpsvol1

```
docker run --name jenkins -p 8080:8080 -p 50000:50000 --restart=on-failure -v
devOpsvol1:/var/jenkins_home jenkins/jenkins:lts-jdk11

docker run --name jenkins1 -p 8081:8080 -p 50001:50000 --restart=on-failure -v
devOpsvol1:/var/jenkins_home jenkins/jenkins:lts-jdk11

docker run --name jenkins2 -p 8082:8080 -p 50002:50000 --restart=on-failure -v
/var/jenkins_home jenkins/jenkins:lts-jdk11

 docker run --name jenkins3 -p 8083:8080 -p 50003:50000 --restart=on-failure -v
/opt/docker/vol2:/var/jenkins_home jenkins/jenkins:lts-jdk11
```

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