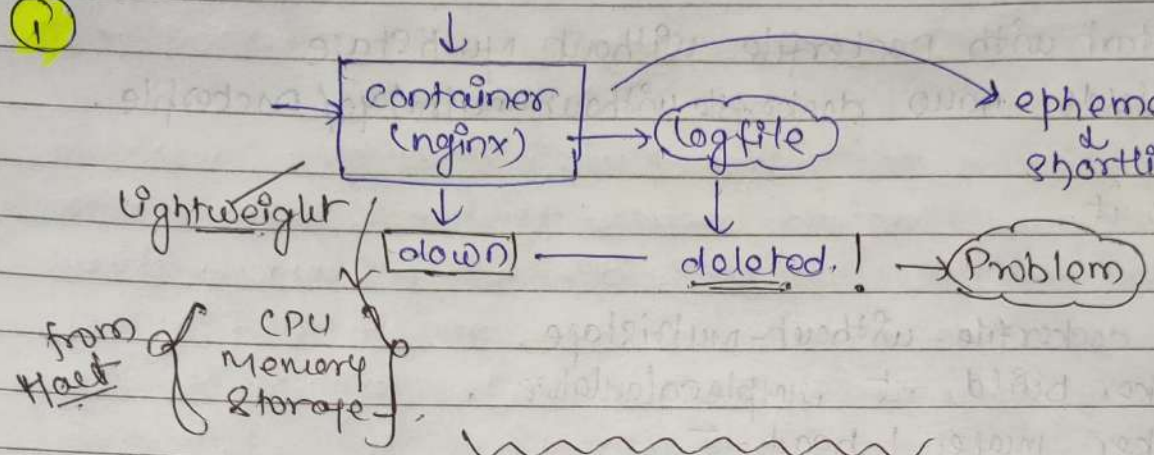
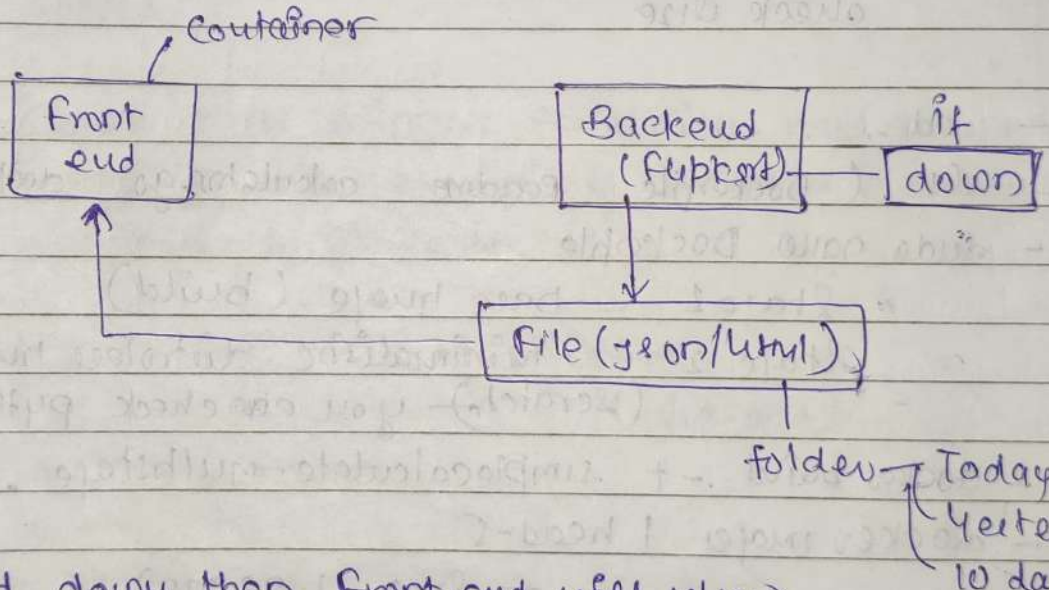
Date:
Page:

Docker Bind Mount & Volumes

1

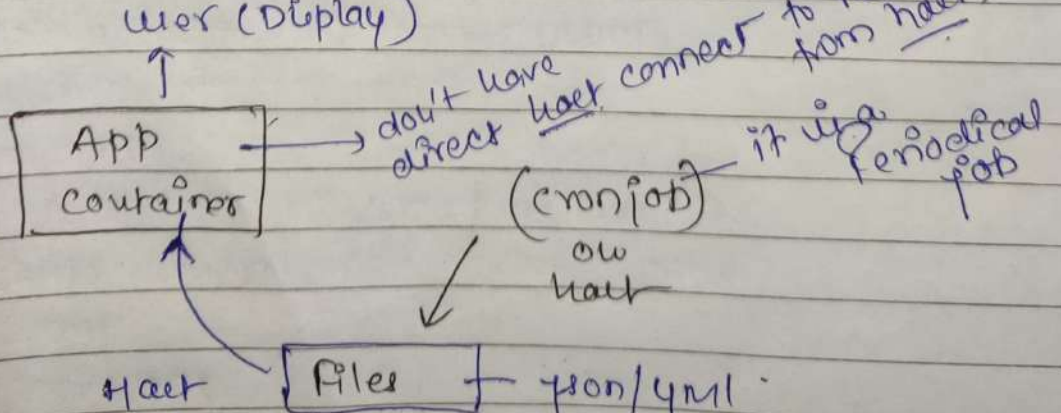


2



If Backend down then Front end user know only today's data.

3

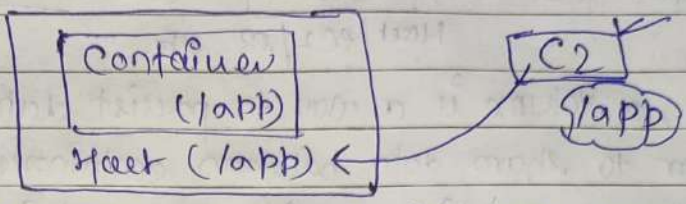


Solutions

Date:
Page:

1. Bind Mount
2. Volumes

① Bind Mount



If the container goes down host can we /app & create another container & bind it with /app so that information is not lost.

"In Docker, a bind mount is a way to share a file or directory from the host machine with a container. It allows you to ~~link~~ link a specific file or directory on your host system to a location within the container, creating a shared link between the two.

Here's a simple breakdown :-

1. Host Machine :- This is your computer or server where Docker is running.
2. Container :- This is an isolated environment created by docker to run application.
3. Bind Mount :- It's like making a symbolic link between a folder on your computer (host) and a folder inside the container.

When you use bind mount, changes made in the linked folder are reflected both on the host m/c & inside the container, & vice versa. This is useful when you want to provide data to a container or persist data generated by a container outside its runtime.

Date:
Page:

easy lifecycle

(2) Volumes

host external source.

Docker cli → [Volume] create/destroy.

Host/ec2/s3...etc.

In Docker, a volume is a way to persist data generated by a container or to share data between containers. It provides a separate, durable storage location outside the container itself. Volumes are used to overcome the ephemeral nature of containers, allowing data to survive container restarts or even when the container is removed.

here is simple breakdown,

1. **Container** :- This is an isolated environment where app runs.
2. **Volume** :- It's an external storage space that can be attached to one or more containers. This is storage is separate from the container's file system & is designed to persist data.
3. **Use Cases** :- Volumes are handy for scenarios where you want to store data that should persist beyond the life of single container, for e.g, storing database files, config files, or any other data that you want to preserve even if container is stopped or removed.

LAB :- Launch ec2 - ubuntu - connect
- sudo apt-get update
- sudo apt install docker.io -y
- sudo systemctl status docker
- sudo usermod -aG docker ubuntu
- logout
- login back.

Date:
Page:

- docker volume ls
- docker volume create techburner
- docker volume ls
- docker volume inspect techburner
- zsh
- docker volume rm techburner
- docker volume ls

To mount the volm, firstly create an container

- sudo nano Dockerfile
  FROM ubuntu
- ~~docker~~ git clone <DockerNew Repo>.
- cd first-docker-file
- ls (Dockerfile app.py)
- docker build -t volumedemo .
- docker volume create techburner
- docker run -d --mount source=techburner, target=/app ~~ngi~~ volumedemo
  or nginx:latest
- docker ps
- docker inspect <cont.ID>
  (mounts search)

→ To delete the volm, need to stop container.

- docker stop <cont. ID>
- docker volume rm <volm>