Docker

1. -> Consitancy on all the devices
   - -> every one uses same command
2. -> Isolations
   - -> improve security
3. -> Portability
4. -> More Efficient
5. -> Version Control
   - -> track version
6. -> Scalibility
   - -> Make copy of machines
7. -> Integerate with Devops

How it Works ?

- Images
- Containers

Image → Lightweight
↳ Standalone
↳ Executable package

Runs a piece of software

Recipe of a software
↳ Run time for our application
System tools

Docker Container

↓

Runnable Instance of an Docker Image

↳ Includes
↓
Image → Code
→ Runtime
→ System Tools
→ Libraries

Eg:- Baked Cake → Docker Container
↳ Recipe used in it → Docker Image

Image

Can Run Multiple Docker Containers

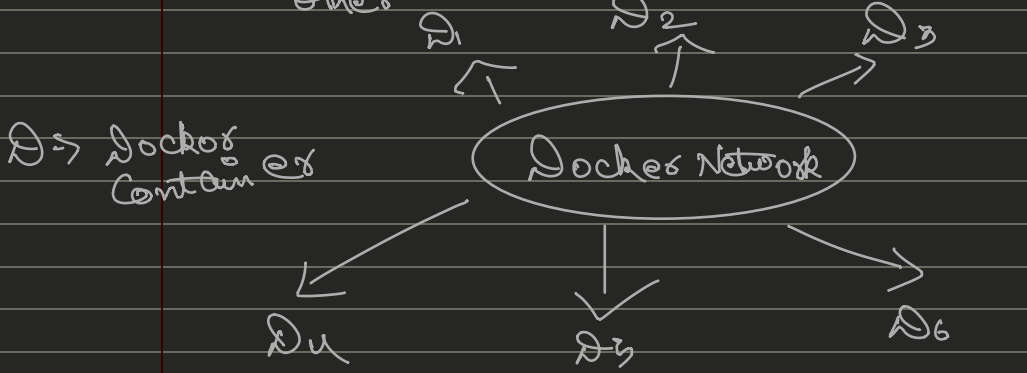
Volumes -> Persistent data storage mechanism

Docker Container ⟶ Volume ⟶ host Machin

Volume: shared folder or storage compartment outside Container

Network ->

Containers can Interact with each other



Docker work flow

Docker client

User Interface to use Docker

(GUI)

[Chef]

Docker host

(Docker Daemon)

Background Process Responsible for Managing containers on host system

[Master chef]

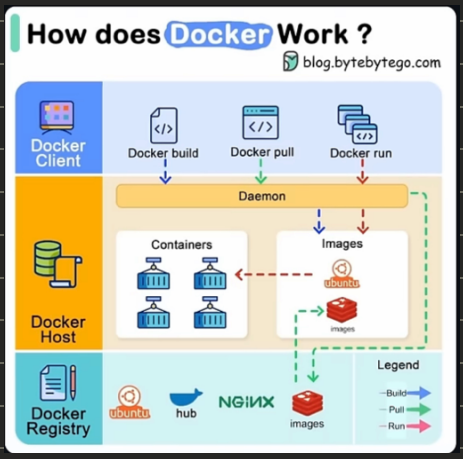
Docker Registry

Docker hub

Github for Docker

Git ⟶ Github

Docker ⟶ Dockerhub



Docker Image ->

(S-1) => Create Docker file

Some Commands

(1) -> Copy

(2) -> Run

(3) -> Expose -> Run on port

(4) -> Env

(5) -> Args -> Deciding what to use

(6) -> Volumes -> create external Mount point

(7) -> CMD -> when the container start

(8) -> Entry point
=> specifies Default executable

CMD ≠ Entry point

CMD -> flexible, can be overwritten

Entry point -> cannot be overwritten

↓

Fixed starting point

Docker File ->

① -> FROM node:20-alpine

② -> WORKDIR /app

③ -> COPY . . -> curr dir to Docker container

④ -> CMD node hello.js

-> Now Open Terminal move to folder

Run the command

↳ docker build -t hello-docker .

impt

for tag

Indicate the curr dir

-> If no tag provided then take tage -> latest

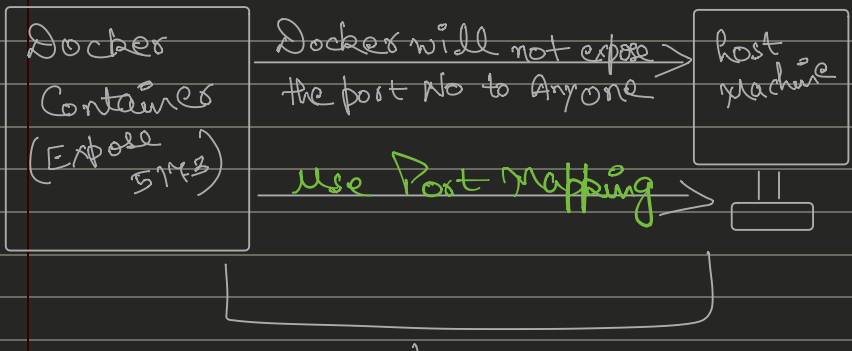
-> docker run -it file Name -> a command to run docker Image

→ To verify Image → docker images

→ To Run Image → docker run file name

→ To go inside Docker container through terminal

↳ docker run -it filename sh



To solve this we use Port Mapping

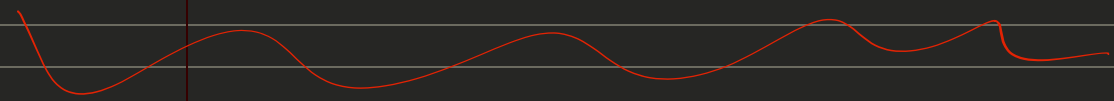
→ docker run -p 5173:5173 file/folder name

→ Also in package.json

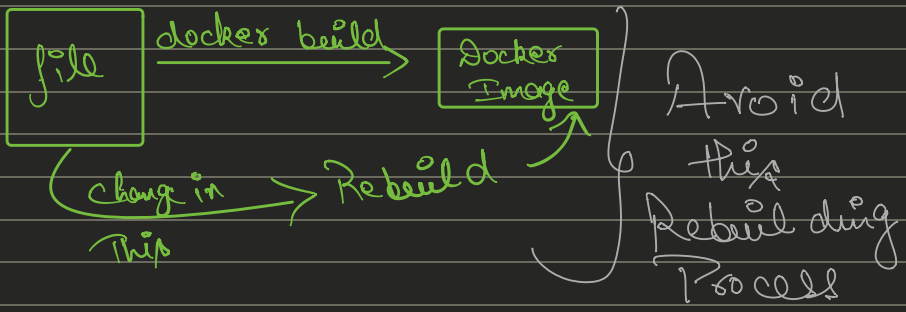
dev : 'vite --host'

Some Docker Commands

1. -> docker ps -> Active running containers
2. -> docker ps -a -> all containers
3. -> docker stop container Id -> stop container
4. -> docker container prune -> delete all inactive container
5. -> docker rm container Id -> To delete specific container



=> Now problem kya hai jab bhi hum kuch bhi update kr rhe hai vo update nhi ho rha hai docker me tb vo krne ke liye re build krna padega



=> We want like jaise Github pe kuch bhi change hua toh Vercel khud hi redeploy kr deta vaise hi docker automatically Rebuild krle hume manually na krna pdle

→ To solve this we will do

docker run -p 5173:5173 -v "$(pwd):/app" -v /app/node_modules folder name

volume

our local code is linked to Docker Container

Impt → we have to create a new volume for node_modules within the directory, we do this to ensure that volume mount is available in the container, now we will run the the container it will run on existing node_modules change in the dependencies, won't require re-install in the container.

How To Publish Image →

(s-1) → docker login

(s-2) → docker tag folder Name username/folder Name

(s-3) → docker push username/folder Name

Now you can check your image on Docker hub

Docker Compose ->

-> Jb itne saare Command kon Run kre just use Docker compose to avoid so, many steps

☆ docker compose up

-> Define & Manage Multi containers Docker apps.

-> Use .yaml file

-> Write all things in a one file & run compose command

-> Automatically trigger Containers

-> Ab file bhi manually create kr skte ho But easy way jo Docker Provide krta hai -> docker init

Initialize our app & create 3 files after some questions

1. -> Docker file
2. -> .dockerignore
3. -> ☆ Compose.yaml ☆

Steps =>

1 => docker init

2 => docker compose up

} Building + Running

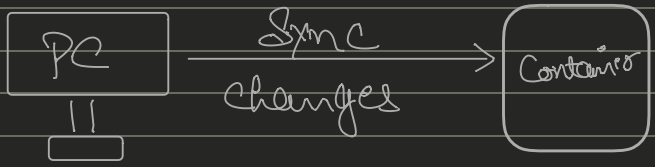
Docker Compose Watch

Listen to the changes

-> Rebuilding our app

-> Re-Running the Containers

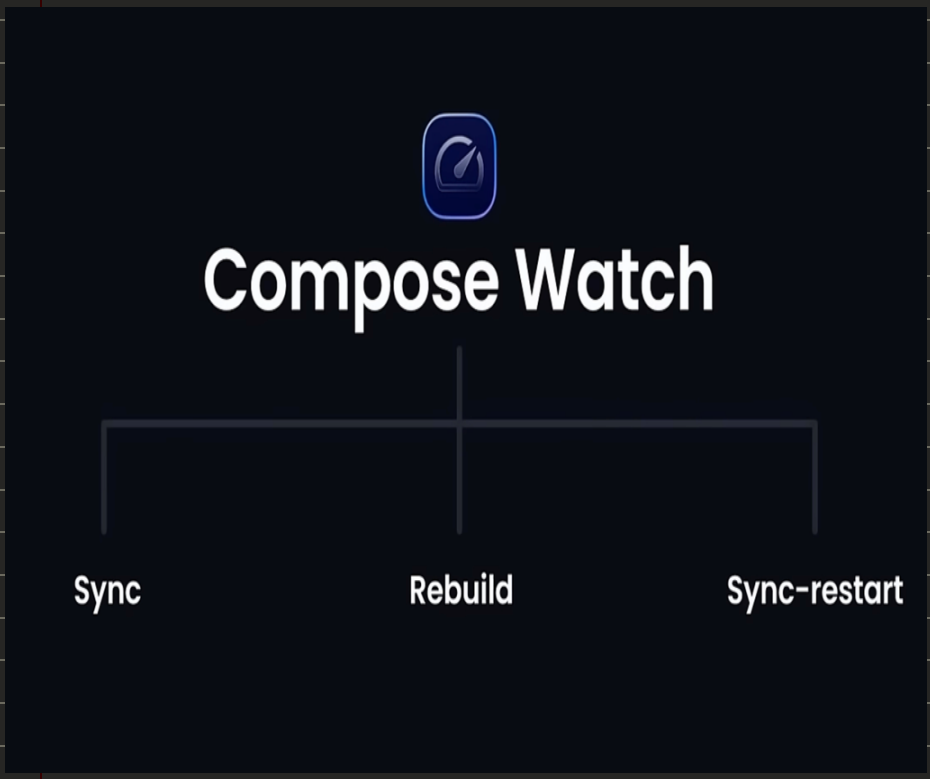
1 =>

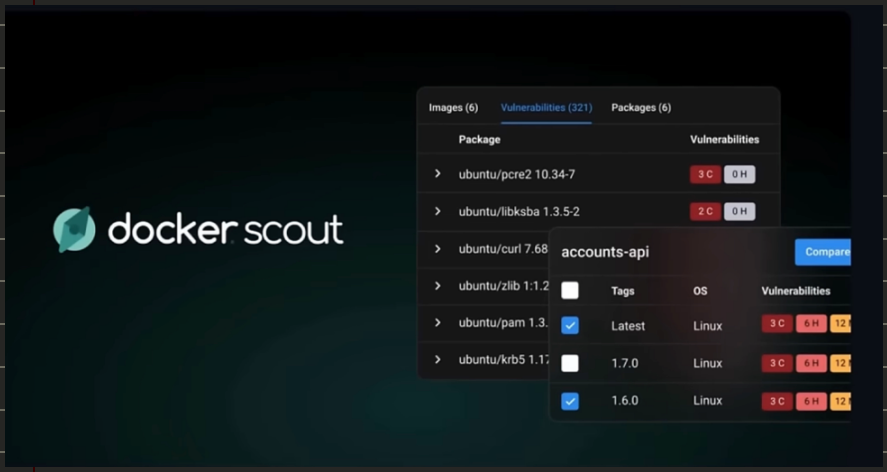


2 => Can do Rebuild & update the service

3 => Sync-restart => Merge sync & rebuild

↳ used in testing & Development


Compose Watch
Sync
Rebuild
Sync-restart


docker scout
Images (6)
Vulnerabilities (321)
Packages (6)
Package
Vulnerabilities
ubuntu/pcre2 10.34-7
3 C
0 H
ubuntu/libksba 1.3.5-2
2 C
0 H
ubuntu/curl 7.68
ubuntu/zlib 1:1.2
ubuntu/pam 1.3.
ubuntu/krb5 1.17
accounts-api
Compare
Tags
OS
Vulnerabilities
Latest
Linux
3 C
6 H
1.7.0
Linux
3 C
6 H
1.6.0
Linux
3 C
6 H

Docker Scout

↳ check vulnerabilities & tell us about that

Main step to dockerize (MEERN)

(S-1) -> Create Dockerfile & .dockerignore in the frontend & backend

(S-2) -> Create a .yaml file

Add watch configuration

Docker init

(S-3) -> docker compose up

(S-4) -> docker compose watch

Next js ->

(1) -> docker init

(2) -> change Dockerfile & docker.yaml according to you (add watch (develop))

③ -> docker compose up -> To start

④ -> docker compose watch -> To sync & rebuild container