1. Go to docker hub -> explore -> search Node -> copy code(docker pull node)
2. Open cmd(for any location, download node image for remote docker hub) -> docker pull node
3. Create example project –> sample\_node\_app, inside it create docker file and other project base files
4. Create docker file
5. Using docker file, build the image –> docker build -t my-first-image . (it – tag, myfirs… is name, . is location (docker file eka thiyena thenama denna))
6. Run this command for view existing docker images -> docker images
7. Create docker container using image -> docker run my-first-image
8. Give name for this container -> docker run --name my-first-container my-first-image
9. What containers are running now -> docker ps
10. What containers are not running now –> docker ps -a
11. Package json file eka add kireema -> npm init -y
12. Nodemon install kireema(backend eke wenasak uu wita eya update weema(hotreaload wage)) -> npm i nodemon
13. Packeg.json wala start script ekak dagannwa -> "start": "nodemon index.js"
14. Dan run karagann puluwan start script eke start use karala -> npm start
15. Meka aye update karanna (docker ekka weda karddi meya awashyay) -> "start": "nodemon -L index.js" (Docker ekka mema script eka use karan nisa -L danawa )
16. Dan apita docker file eke CMD [“node”, “index.js”] wenuwata meya denna puluwan -> CMD [ "npm", "start" ]
17. Api hadagata package.json file eka api working dir Ekata copy karaganna one ->
18. Denata run wena okkoma images/containers/volumes iwath kireema -> docker system prune -a
19. Image ekak hadagnnw me wenaskam ekka -> docker build -t simple-image .

Image versioning - docker build -t my-image:v1 .

1. Hadagatta image eken container ekak hadagannwa volume ekath samagama -> docker run --name simple-container --rm -v /app/node\_modules -v ${PWD}:/app simple-image

20 n passe nodejs backend app ekak containerize karana project 2 video eke sita (21 step eka lesa eka danawa)

1. docker run --name node-backend-container2 -p 5000:5000 --rm -v ${PWD}:/app -v /app/node\_modules node-backend-image:v1

**How push docker image to docker hub**

1# docker login (username eka ahanawne eka docker hub eken balaganna puluwan)

2# docker tag node-backend-image:v1 malintha25/node-backend-image:v1

3# docker push malintha25/node-backend-image:v1

**How pull docker image to local machine**

1# Adala reposatary Ekata yanna, **public view** Ekata gihin pull karanna puluwan command eka ganna

2# docker pull malintha25/node-backend-image – me widiyata command eka thiyenne

Namuth, docker pull malintha25/node-backend-image:v1 lesa tag ekak yoda eya vs code Ekata awilla terminal eke run karanna

Dan wenasak karoth update wenawa container eketh

Container eka update unath image eka update wenne naha

Image Ekata wenasak krnn nam aluthrn image ekak hadanna one e wenaskam samaga

**For 20**

Mehidee –rm eken container eka stop karagtta gaman remove wela yanawa

-v /app/node\_modules – node\_modules walata wenama volume ekak awashya nathi nisa eya wenama hadagannwa

-v ${PWD}:/app – mehi ${PWD} kiyanne present working directory kiyana (${PWD} (which stands for "Present Working Directory") is a shell variable that references the current directory.) eka (H:\Docker\sample\_node\_app> me wage api denata innsa path kotasa e kiyane host eke file system location eka)

:/app -> meken kiyanne docker eke container eke specific location Ekata

**/app/node\_modules** is the **container directory**.

* When you run this command with **-v /app/node\_modules**, it creates a **volume** that stores the data of the node\_modules directory inside the container.
* It **does not** refer to a local directory on your machine, but instead tells Docker to **persist the node\_modules directory** data **separately** outside the container (in a volume).

What happens with -v /app/node\_modules?

This creates a named volume on the host system, which Docker manages.

 The purpose is to keep the node\_modules directory from being overwritten every time the container restarts.

 Since node\_modules can often be large and include platform-specific code (depending on your system), Docker handles this with a volume so that it stays persistent across container restarts or even rebuilds.

Using a volume for node\_modules prevents the installed packages from being lost when the container is stopped or restarted.

The -v /app/node\_modules in the command is an anonymous volume.

Anonymous volumes do not have a human-readable name or specific location on your system, but Docker still persists the data, and it can be reused by the container during restarts. However, it cannot be accessed easily by the user or linked to any specific path on the host system.

 When you use the -v /app/node\_modules volume, Docker ensures that the node\_modules directory in the container is persistent. The data in this directory will persist even if the container is stopped or removed.

 The benefit: Since the node\_modules are mounted as a volume, any modifications (such as when you run npm install or yarn install inside the container) will be retained across restarts or re-creations of the container. This avoids the need to reinstall dependencies every time you rebuild the container.

 Without this volume: If you didn't mount the volume, every time you rebuild the container, Docker would copy over the node\_modules directory from your project directory (or leave it out if you're using the COPY . . instruction). You would need to reinstall all the dependencies every time you create a new container.

**For 20**

* 1\_We can use image name or image id for create container
* 2\_Even if we make changes to the locally existing project, the same changes will not be made to the container created from the previously created image.

Eg –

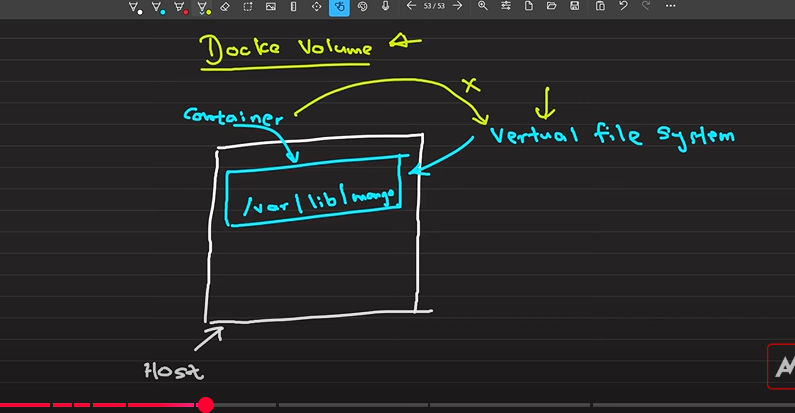
PS H:\Docker\sample\_node\_app> node index.js

Sum is: 12

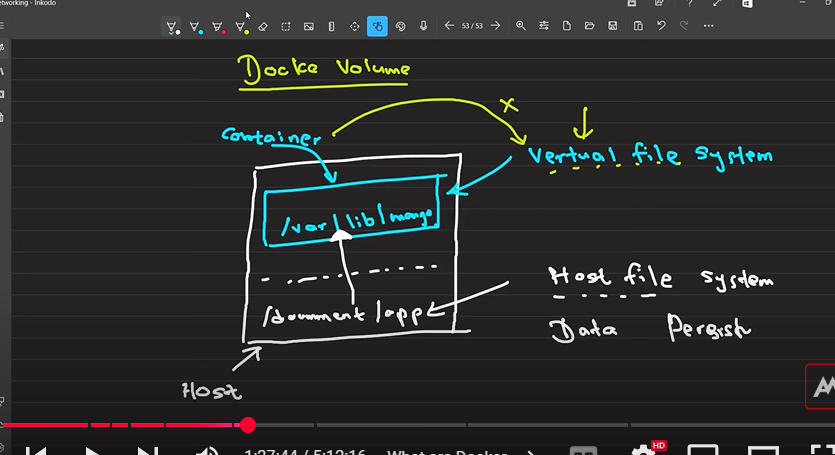
PS H:\Docker\sample\_node\_app> docker run --name test-container my-first-image

Sum is: 3 (passe code eke value change karata passe image ekak e wenuwen hadapu nathi nisa ihatha 12 methana penwanne na, parana image eken hadapu container ekak create wela parana value eka penwanne methana)

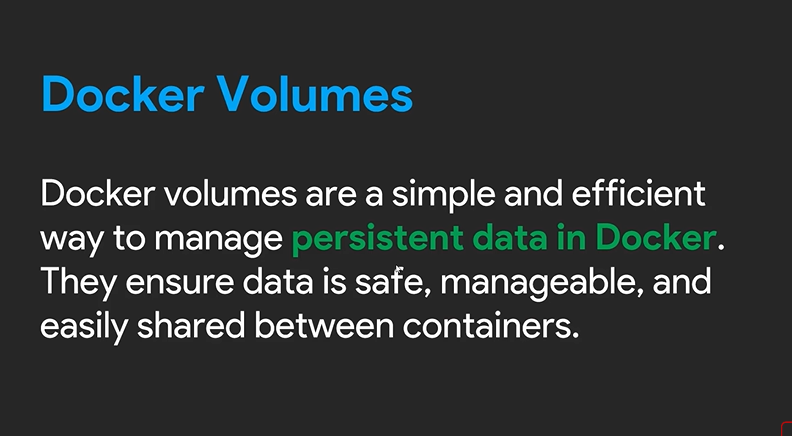
-> Ihatha error eka hadaganna nam docker volume awashya wenawa (siduwana wenasweem image eke update weemata)



Container eke store wena files wage dewal thiyaganne docker ekka yana virtual file system ekaka, api container eka update karoth ema files nathi wela yanwa, e sadahath volumes apita yodaganna puluwan.



Mehidee virtual file system eke data store karanwa wenuwata volumes magin host machine eke file system eka map karagannwa (mount karanawa) virtual file system ekath ekka(host machine eke data remove wenne nane). Mehidee host file system eke code eka update karagnnkota virtual file system ekath update wenawa.



There are three volumes,

1- annonymouse volume – api specify krnne virtual file system eka witaray, eya host file system eke kohetada mount wela tyenne kiyala api danne naha

2-host volumes – mehidee virtule saha host file system wala location specify karanna puluwan

3-named volumes – mehidee api kalin hadagatta location ekakata/volume ekakata/host file system ekakata, virtually tyena file system eka/volume eka mount karaganna puluwan

Mehidee project eke wena wenasak aduragann system ekak awashy, e sadaha ‘nodemon’ use karanna puluwan

* Command run karaganeemata RUN npm install wage denna pulwan

**For react project**

**1. setup docker file**

# Use the base image from Docker Hub

FROM node:20-alpine

# Set the working directory (me app directory ekata tamay api project eka copy karagnne)

WORKDIR /app

# Copy package.json and package-lock.json to the working directory

COPY package\*.json ./

# (above) package\* kiyanne e wachanen patan ganna siyalu file kiyana eka, WORKDIR /app kiyana line eka dunnama api hithanna one api dan app folder eka thulata awilla kiyala

# Install the dependencies (api package.json ekata thiyena dependencies tika install karanawa)

RUN npm install

# (above) api localy nam mema package.json ekata thiyena dependencies tika install karanawa, e wagema api mehideeth karanna one

# copy the rest of the application files to the app

COPY . .

# dan api container eka expose karaganna one,

# me container eka kiyanne ape lap eka athule thiyena thawath lap ekak wage, namuth api meka run karagnne main environment eke idan, main eke thiyena localhost use karala

# e nisa api run wena port eka specify karanna one(main environment ekata container eken dena port eka access karanna puluwan wenna one, enam me port eka api eliyata expose karanna one, ewita tmy eliye main computer ekata app eka access karanna puluwan wenne)

EXPOSE 5173

# api ihatha port eka expose karapu nisa apita e port eka use karala dan app eka run karaganna puluwan

CMD [ "npm", "run", "dev" ]

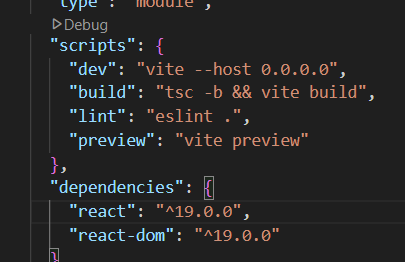
**2. Create docker image**

* docker run –name container\_name -p 3000:5173 image\_name ->

**-p 3000:5173**

* This is port forwarding.
* 3000 (Host Port): Port on your local machine (e.g., your browser or Postman will connect to this port).
* 5173 (Container Port): Port exposed by the application running inside the Docker container.
* So, when you access localhost:3000 on your machine, it gets redirected to 5173 on the Docker container.

Api me command eka dunnath 3000 port eka 5173 th ekka connect wela na, e sadaha package.json wala pahata de karanna one



* **Container eka volume ekath ekka hadaddi, file update listn karan inna widiya setup kireema**
  + docker run --name container\_name -p 3000:5173 --rm -v /app/node\_modules -v ${PWD}:/app -e CHOKIDAR\_USEPOLLING=true image\_name , CHOKIDAR\_USEPOLLING=true meya magin eya siduwe