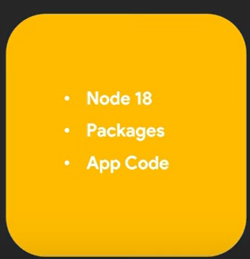
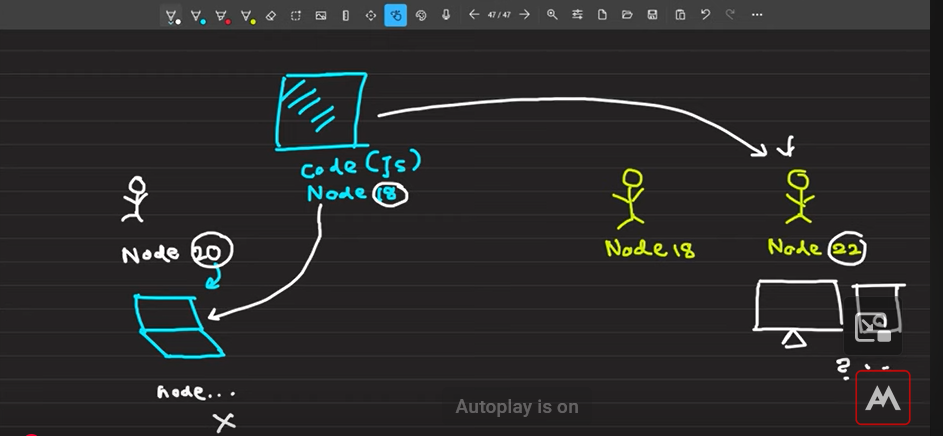
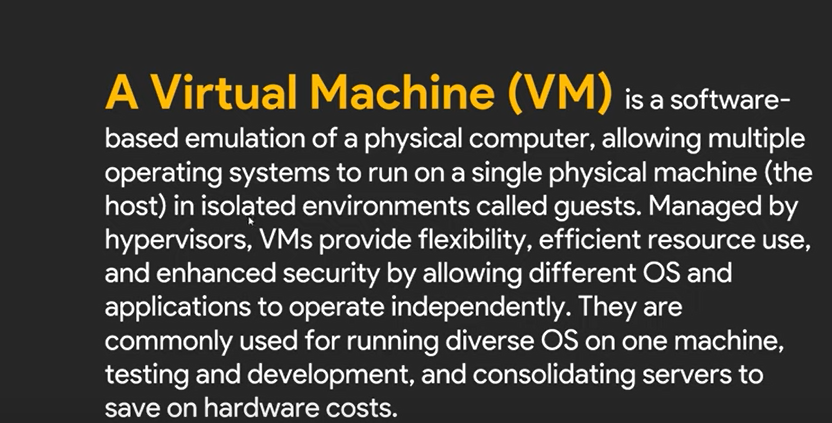
**Docker**

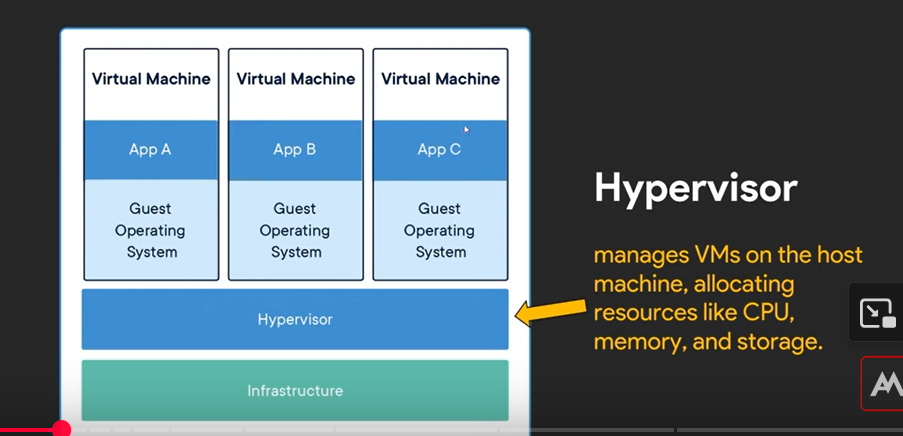
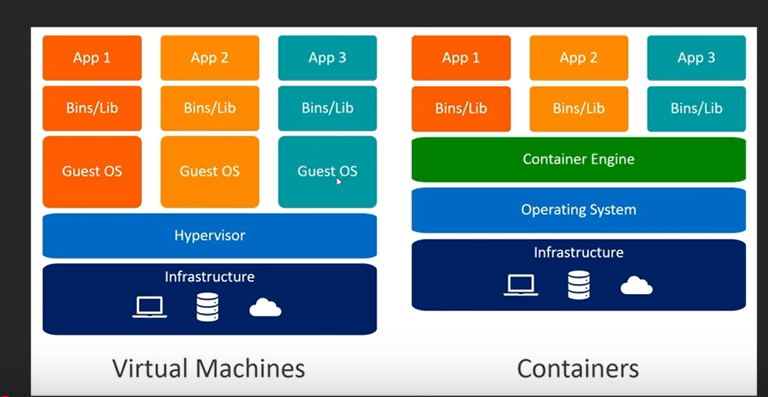
Using docker we can made the container including the environment Configuration

As an example your project used the node 18 but your friend has node 20 then the project don”t work on friend computer.

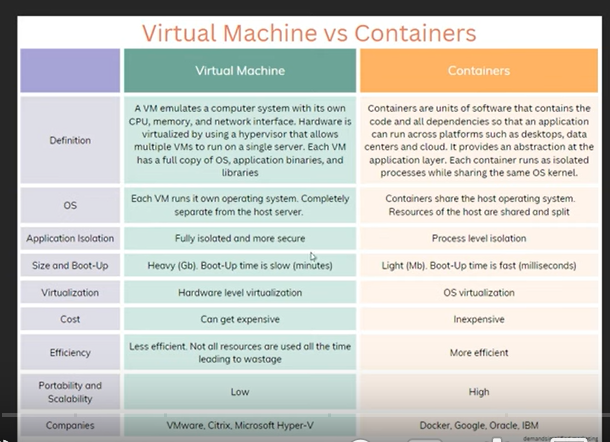




**Different Between the VM Containers**



**Compare of the Virtual Machine Vs Containers**

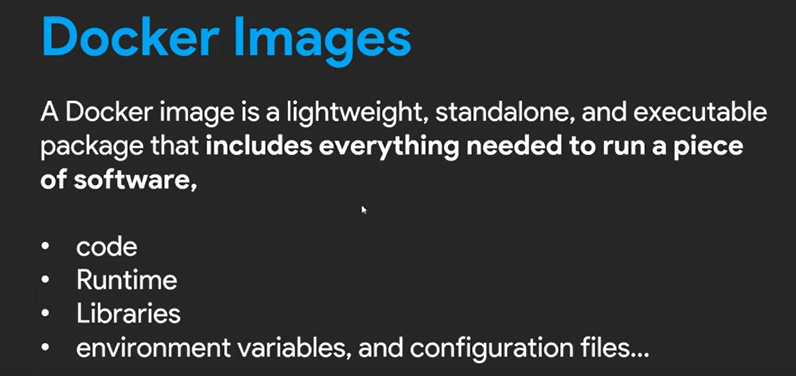


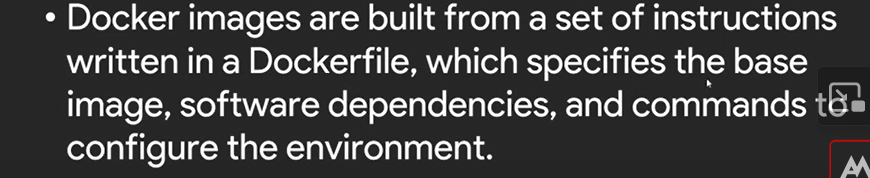
**Docker Images and Containers**

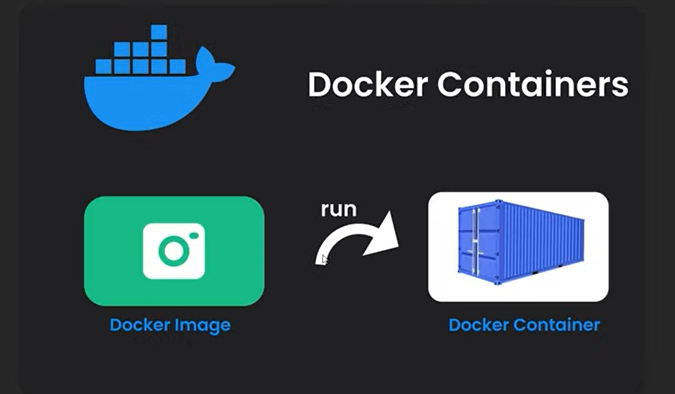
What is the Docker image ?

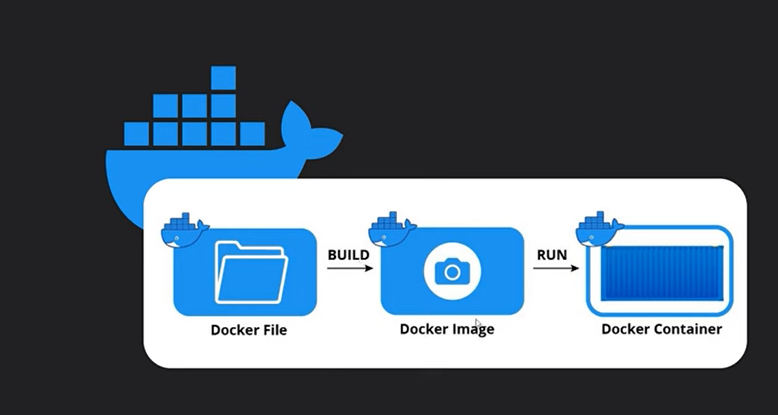
A Docker image is a lightweight, standalone , and executable package that includes everythings,

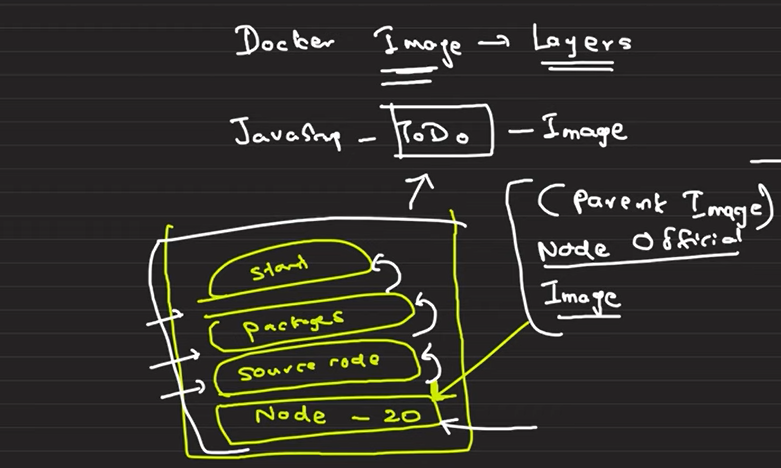
It is a template form which Docker Containers are Created

****

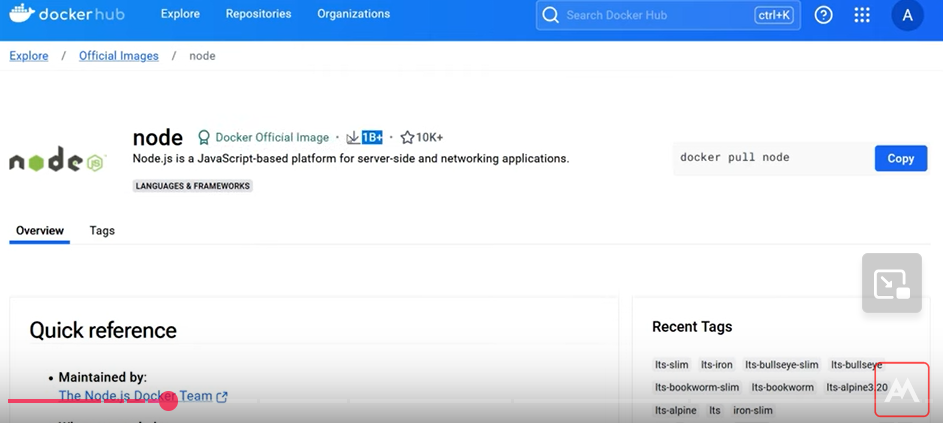
**Docker file**

****

****

**How to set the environment of the docker**

All the images are include in docker hub Then Using that we have the get the node , mongo images

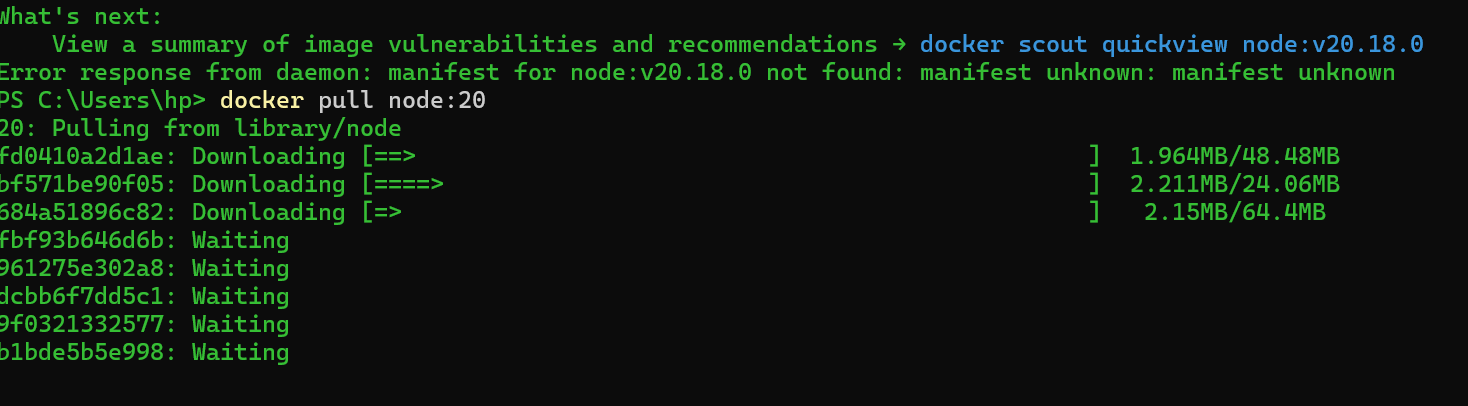


This is the docker hub images got tag and you can choose suitable images

**How to get the the Docker Hub images in to your Computer**

**Commad**

**>>Pull node**

****

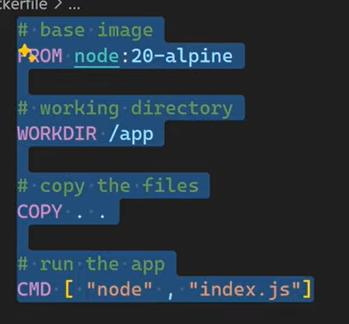
f

**How to make the Docker file**

Put the extention of docket on the vs code that provide snipping

**Docker file code**

**Simple Example**

#Base Image

**FROM node:20-alpine #( in this we are using light weight file)**

# Working directory

**WORKDIR /app**

#copy the Source files

**COPY . .**

#run the app

**CMD[“node”,”index.js”]**

**How to build a docker image**

Goto working directory

**Command>>**

docker build –t image\_name .

**How to make the Docker container**

Command> docker run simple\_image

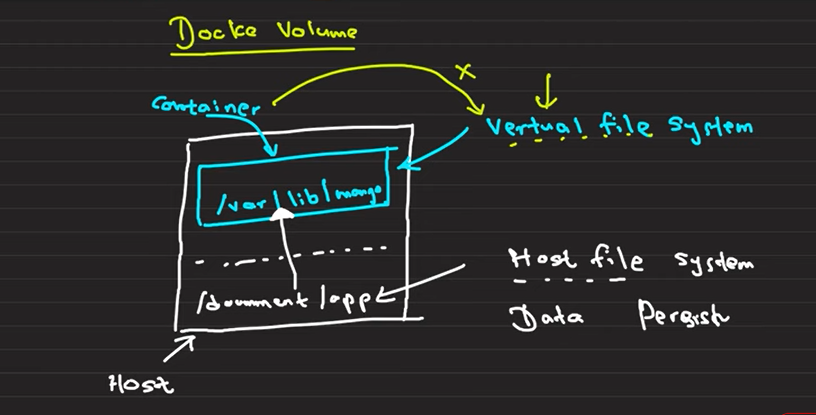
Command> docker run –name my-first-container simple\_image

docker images (we can see all images)

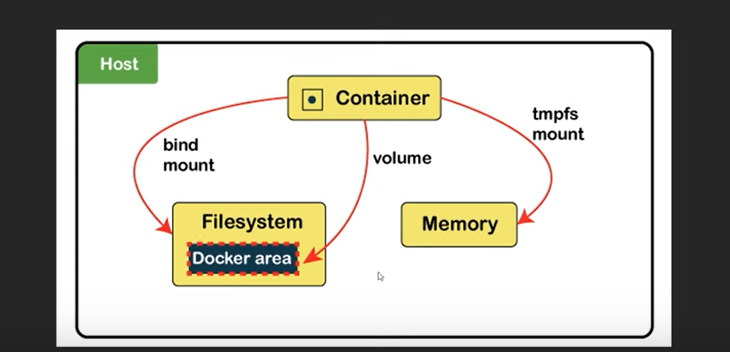
docker ps ( run images)

docker ps -a ( unused images)

Docker Volume



**Docker Volumes are a simple and efficient way to mange persistent data in Docker. They ensure data is safe , manageable, and easily shared between containers.**

****

**Folder in physical host file system is mounted in to the virtual file system**

**Types of Docker Volumes**

1. Anonymous Volumes: Created and managed by Docker but not given

a specific name. These volumes are typically used for temporary data

that doesn't need to be persisted beyond the container's lifecycle.

**docker run -d --name my\_container -v /app/data my\_image**

2. Named Volumes: Created with a specific name and can be

referenced by multiple containers. Named volumes are useful for

persisting data that needs to be shared between containers or across

container restarts.

**docker run -d --name my\_container -v my\_volume:/app/data my\_image**

3. Host Volumes (Bind Mounts): Directly map a directory or file on the

host to a directory or file in the container. Unlike managed volumes,

the host determines where the data is stored. Bind mounts provide

more control but less isolation from the host system.

**docker run -d --name my\_container -v /path/on/host:/path/in/container my\_image**

**Commands**

1.Create a docker volume

docker volume create my\_volume

2.Run a docker container with a volume

docker run -d --name my\_container -v my\_volume:/app/data

my\_image

docker run --name container\_name --rm -v /app/node\_modules -v

${PWD}:/app image\_name

3.List all docker volumes

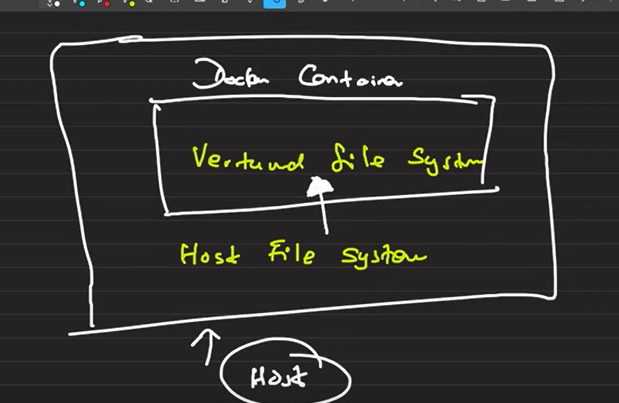
docker volume ls

4.Inspect docker volumes

docker volume inspect my\_volume

5.Remove

docker volume rm my\_volume

****

**For this we want to do two thing first one we should get the package json for that**

**Commad>> npm init –y**

**Command>> npm I nodemon**

Nodemon is a utility **depended on by over 3 million projects**, that will monitor for any changes in your source and automatically restart your server. Perfect for development.

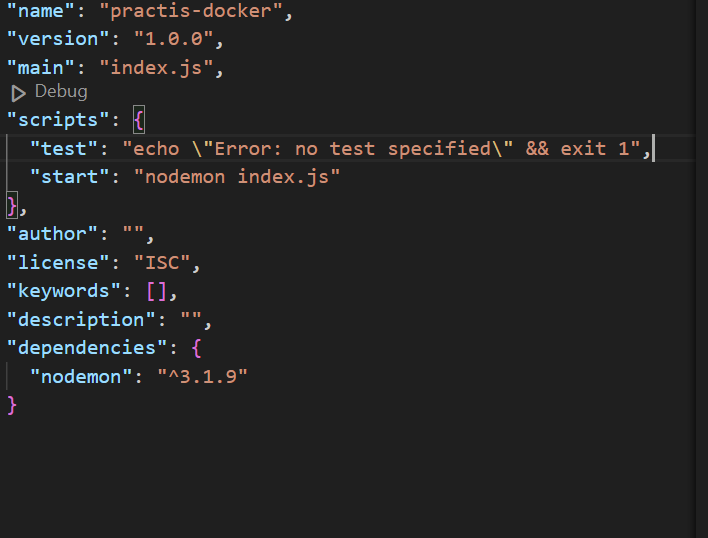
Swap **nodemon** instead of **node** to run your code, and now your process will automatically restart when your code changes. To install, get [Node.js](https://nodejs.org/), then from your terminal run:

npm install -g nodemon

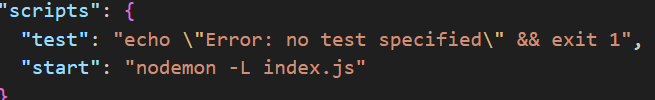
Features

* Automatic restarting of application.
* Detects default file extension to monitor.
* Default support for node but easy to run any executable, such as python, ruby, make, etc.
* Ignoring specific files or directories.
* Watch specific directories.
* Works with server applications or one time run utilities and REPLs.
* Scriptable through node require statements.
* Open source and available on [github](https://github.com/remy/nodemon/).

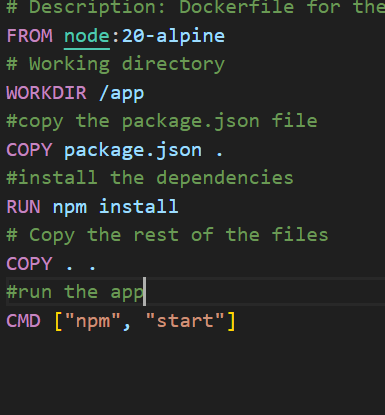
Read the full [documentation](https://github.com/remy/nodemon#nodemon) or visit the [FAQ](https://github.com/remy/nodemon/blob/master/faq.md)



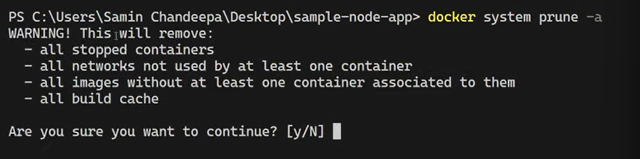
We want to change the script that should be start in nodemon



For the Docker usage that should change as “nodemon –L index.js”



**Change the Docker file content for that suitable**

****

This command for stop the running docker images we should stop running docker images.

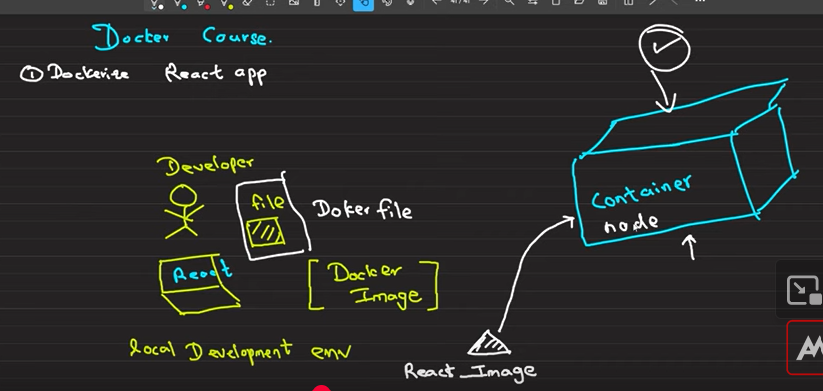
Now, we have to create a new image “docker build –t simpleimage .”

In this the image does not update only update the container

Now run the docker run --name simple-container --rm -v /app/node\_modules -v ${PWD}:/app simpleimage

**Docker Project**

Dockerize React app



WORKDIR /app- working directory

The project is create in the /app folder that

**The sequence of the dockerfile for create the React application**

FROM node:20-alpine

WORKDIR /app

COPY package\*.json ./ ---------------------- /app—use \* to copy the all json files

RUN npm install **“adding dependencies”**

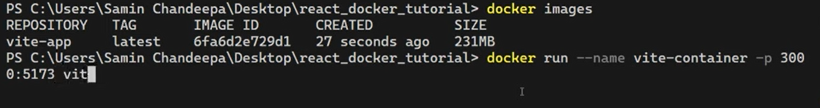
COPY . . **”copy the remaining files”**

**Now I want to expose the this container**

EXPOSE 3000

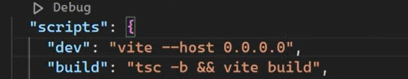
CMD[ “npm”,”run”,”dev”] **“start the application”**

**Then make the image and after containerize**

****

**We can see run command “docker ps”**

**In vite some problem come because of the Ip address then you can make correct that in pakege.json**

****

You should stop the docker and again run

**Command>> docker stop vite-container**

**Again run the docker with port**

**If we want to remove the image**

**Command>> docker image rm vite-app**

In this give the error because we should firstly delete the **container**

**Commad>>docker ps**

**Commad>> docker ps –a**

**Commad>> docker container rm vite-app**

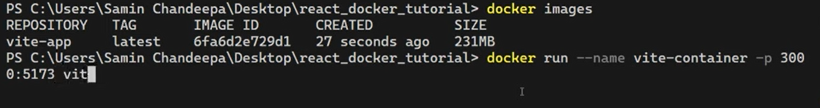
**Now again run the image remove command**

**Command>> docker image rm vite-app**

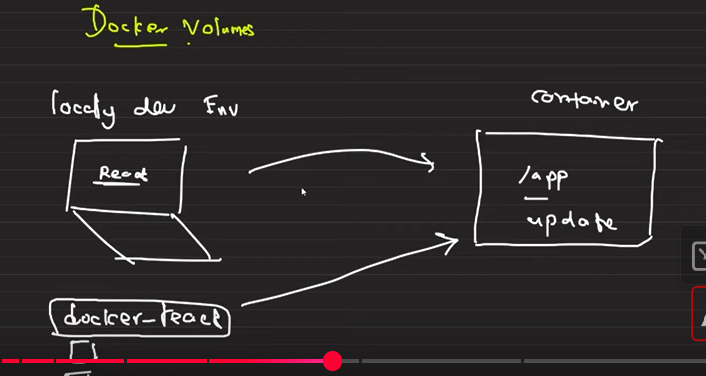
**Docker volume for React**

Commd>> docker build –t vite-image

**Commad>>docker run (port 4000)**

****

**We used volume in small change code in instant change on the ouput**



1.Docker Volumes

This Docker run command sets up a container (vite\_container)

based on the vite-app image, with port mappings, volume mounts,

environment variable settings, and automatic removal (--rm) when

the container stops. It's configured to facilitate local development with

live code updates (${PWD}:/app), efficient node\_modules

management (/app/node\_modules), and reliable file change

detection (CHOKIDAR\_USEPOLLING=true). Adjustments can be

made based on specific project requirements or environment

configurations.

docker run --name container\_name -p 3000:5173 --rm –v(“if the stop the docker container “rm” remove the volume”)

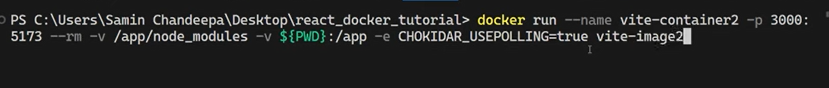
/app/node\_modules -v ${PWD}:/app -e

CHOKIDAR\_USEPOLLING=true image\_name

The --rm flag in the docker run command stands for "remove".

When you use --rm, Docker automatically removes the container and

its filesystem when the container exits (stops running



In this don’t work nodemon instead of this we use “CHOKIDAR\_USEPOLLING=true image\_name”this command

**Dockerize node Application**

Make the node application

Commad>> npm init –y

Commad>>npm install express

Commad>>npm install nodemon

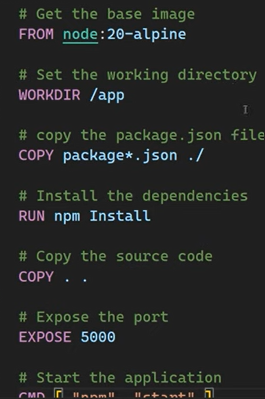
Then make the server.js

Commad>>npm install dotenv(we get data from env)

Make the gitignore ( node\_modules,.env)

Make the dokcerfile , dockeignore

Now dockerfile create



Last commad CMD[ “npm”,”start”]

Install should change install

You should include the node\_modules in the dockerignore

Now you should make the image and container

**Dockerignore**

In Docker, the .dockerignore file is used to specify which files and

directories should be excluded from the Docker build context. When

you build a Docker image, Docker uses a "build context" which

includes all files and directories in the current directory (where your

Dockerfile resides) and its subdirectories. The .dockerignore file

helps optimize the build process by preventing unnecessary or

sensitive files from being sent to the Docker daemon as part of the

build context.

**Commands**

1.Create an docker image using the Dockerfile

docker build -t image\_name .

2.To see all the images

docker images

3.To create a container from that image

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

docker run –name container\_name -p 3000:5173 -d image\_name (to

run in detached mode)

4.To see all the running services(containers)

docker ps

5.To see all the containers (running ones or not)

docker ps -a

6.To stop a running container

docker stop container\_name

or

docker stop container\_id

7.To restart a docker container

Docker start container\_name

(here we don't need to re configure the port mappings as we did that

earlier)

‘

8.Remove a docker container

docker ps -a

docker container rm CONTAINER\_ID\_OR\_NAME

Remove multiple containers

docker container rm CONTAINER\_ID\_OR\_NAME

CONTAINER\_ID\_OR\_NAME

9.Remove a docker image

docker images

docker image rm IMAGE\_ID\_OR\_TAG

10.Remove all containers all images and all volumes

docker system prune -a

11.Docker Volumes

This Docker run command sets up a container (vite\_container)

based on the vite-app image, with port mappings, volume mounts,

environment variable settings, and automatic removal (--rm) when

the container stops. It's configured to facilitate local development with

live code updates (${PWD}:/app), efficient node\_modules

management (/app/node\_modules), and reliable file change

detection (CHOKIDAR\_USEPOLLING=true). Adjustments can be

made based on specific project requirements or environment

configurations.

docker run --name container\_name -p 5000:5000 --rm -v

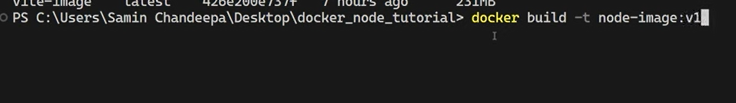
/app/node\_modules -v ${PWD}:/app image\_name

The --rm flag in the docker run command stands for "remove".

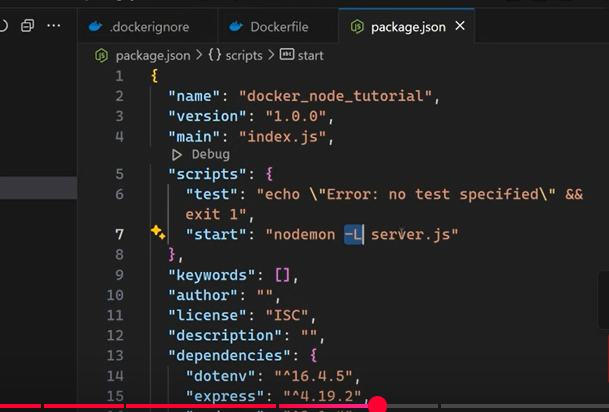
When you use --rm, Docker automatically removes the container and

its filesystem when the container exits (stops running

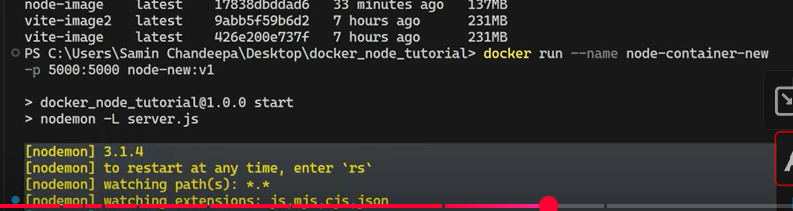
Versioning



Nodemon using that change can do in that instant



Create new image and the container

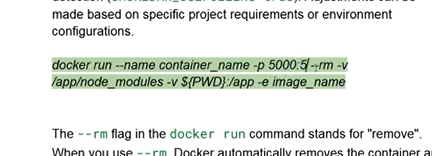


Add the version definitely

Now

docker stop container\_name

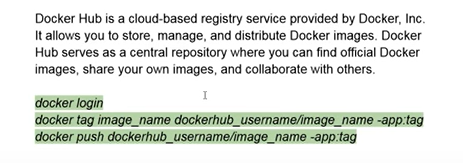
create the volume

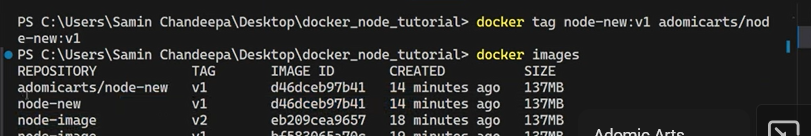


**${PWD}:/app –v present working directry map to the current working directory**

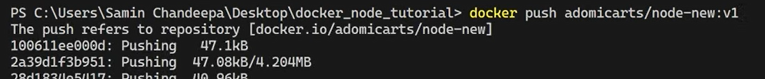


**Push Images to Docker Hub**

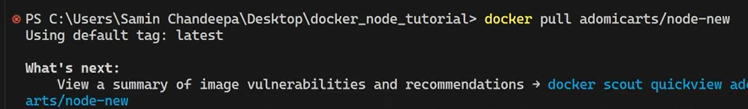
****



Change the image name

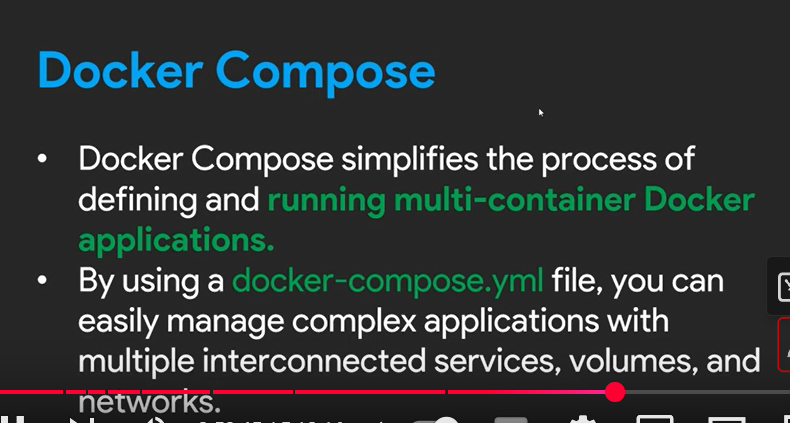


Push the image push the docker hub

**Image pull in the docker hub**

**Add the version otherwise show errors**

**Docker Compose**

****

**Containerizing a Full-Stack MERN Application**

**Sever folder**

**Commmad>>npm init -y**

****

**Docker-compose.yml**

# This file is used to define the services that will be used in the application.

services:

  mongo:

    image: mongo:latest

    container\_name: mongo\_container

    volumes:

      - mongo\_data:/data/db

    ports:

      - "27017:27017"

  api:

    build: ./api

    container\_name: api\_container

    ports:

      - "5000:5000"

    depends\_on:

      - mongo

    volumes:

      - ./api:/app

      - /app/node\_modules

    environment:

      - " MONGO\_URL=mongodb://mongo:27017/test-users”

  client:

    build: ./client

    container\_name: client\_container

    ports:

      - "3000:3000"

    depends\_on:

      - api

    stdin\_open: true

    tty: true

    volumes:

      - ./client:/app

      - /client/node\_modules

volumes:

  mongo\_data:

**Database localhost and container are two different ones**

MONGO\_URL=mongodb://mongo:27017/test-user

**This is the container one the not a local host”**

**Commad >>docker compose up**