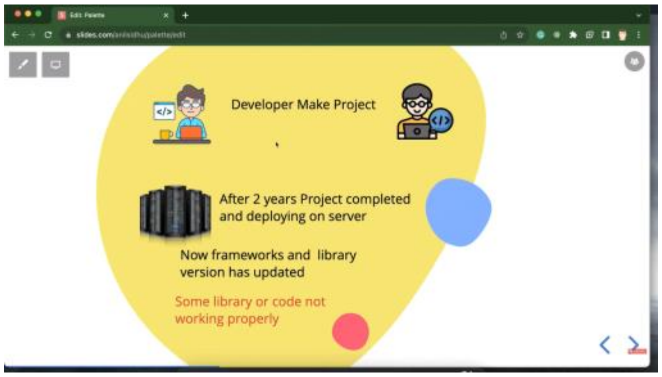
DOCKER

Wednesday, December 11, 2024 7:02 PM

What is Docker?

Official Definition :
Docker is a platform that helps develop build, share, and run applications with containers.

Use case:



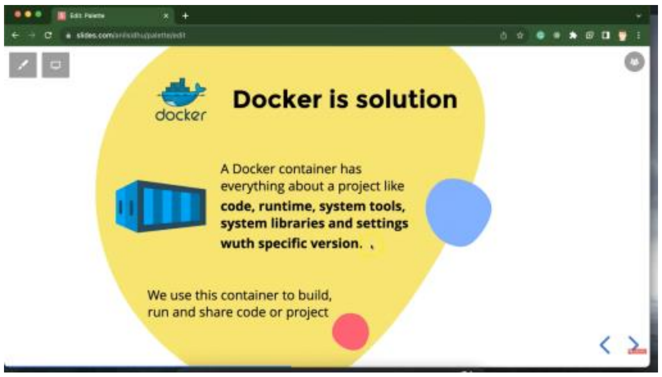
Developer Make Project

After 2 years Project completed and deploying on server

Now frameworks and library version has updated

Some library or code not working properly

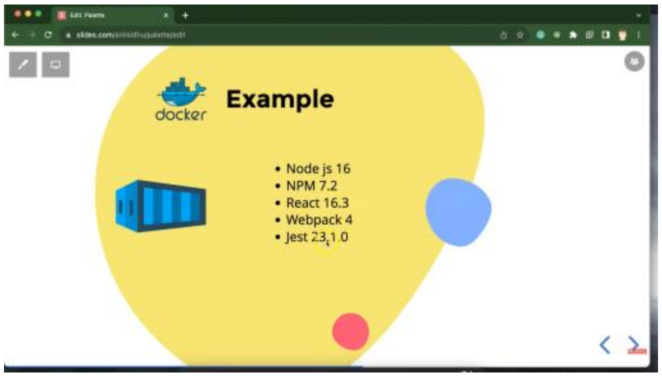
Solution :



Docker is solution

A Docker container has everything about a project like **code, runtime, system tools, system libraries and settings wuth specific version.**

We use this container to build, run and share code or project



Example

- Node js 16
- NPM 7.2
- React 16.3
- Webpack 4
- Jest 23.1.0

Key Features of Docker:

1. **Containerization**:
    - Encapsulates applications and their dependencies in containers, ensuring that the application works the same in development, testing, and production environments.
2. **Portability**:
    - Containers are platform-independent and can run on any system with Docker installed, whether it's a developer's laptop, a virtual machine, or a cloud server.
3. **Efficiency**:
    - Containers share the host system's operating system kernel, making them more lightweight and faster to start than traditional virtual machines (VMs).
4. **Version Control**:
    - Docker enables you to create, manage, and version your application images, allowing easy rollback to previous versions if necessary.
5. **Isolation**:
    - Each container operates in isolation, ensuring that changes in one container don't affect others.
6. **Scalability**:
    - Containers can be quickly scaled up or down based on demand.

Installation on window: https://docs.docker.com/desktop/setup/install/windows-install/
Installation on Mac: https://docs.docker.com/desktop/setup/install/mac-install/
Installation on Ubantu: https://docs.docker.com/engine/install/ubuntu/

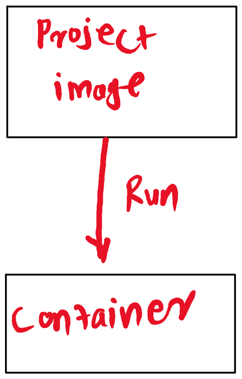
What is Image?

Images are templates for the project.

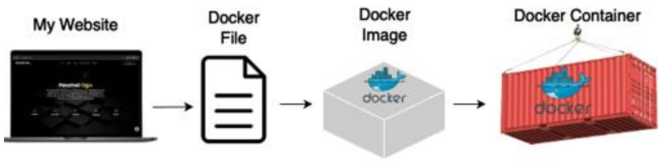
1. It is a kind of ready-to-use software read-only template.
2. The image is made with source code, libraries, external dependencies, and tools.
3. The image can not be updated.
4. If you want to make a change in the image you have to create a new image.
5. The image can not run directly.

What is a Container?

Container:- Running instance of image.



Container is an isolated process. (Mean container run indepndently on a computer).



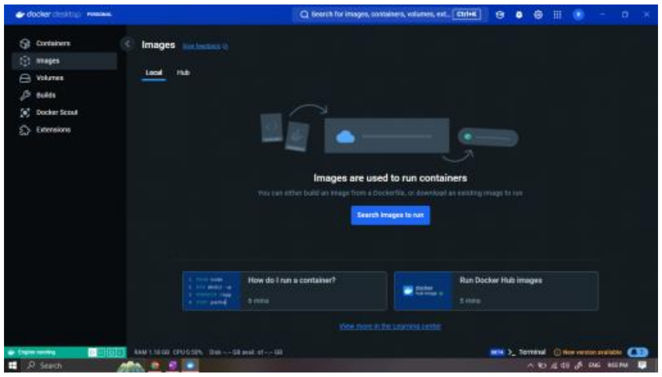
Docker Hub v/s Docker Desktop

Feature	**Docker Hub**	**Docker Desktop**
Type	Cloud-based platform	Local application
Primary Role	Image storage and distribution	Local container runtime
Functionality	Push/pull/share Docker images	Build, test, and run containers locally
Access Method	Web interface or Docker CLI	GUI and CLI on local machine
Dependency	No local installation needed	Requires installation on local machine
Purpose	Central repository for Docker images	Development and testing environment
Example Use	docker pull nginx or docker push my-app	docker build . or docker run my-app

What is Base Image or Parent Image?

A base image is the initial image in a Docker container that provides the environment needed to run applications.

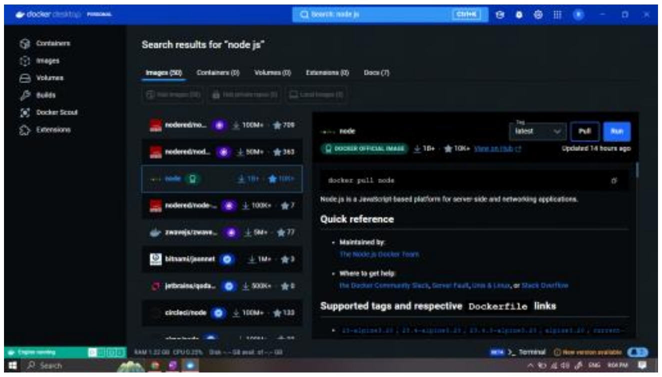
Docker Desktop UI:-



Images are used to run containers

You can either build an image from a Dockerfile, or download an existing image to run

How to install any image
Click on **pull** button



Search results for "node js"

```
docker pull node
```

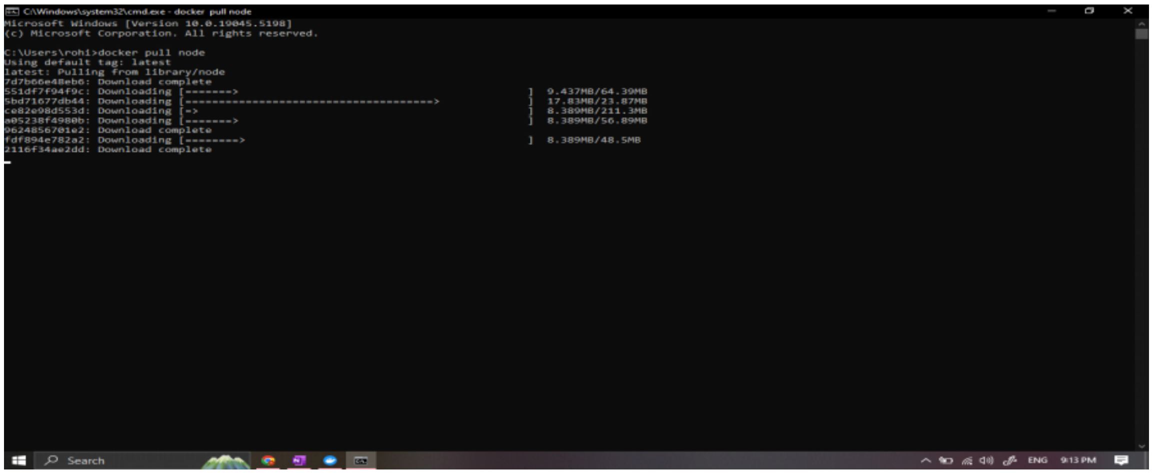
Node.js is a JavaScript-based platform for server-side and networking applications.

Quick reference

- Maintained by:
  The Node.js Docker Team
- Where to get help:
  the Docker Community Slack, Server Fault, Unix & Linux, or Stack Overflow

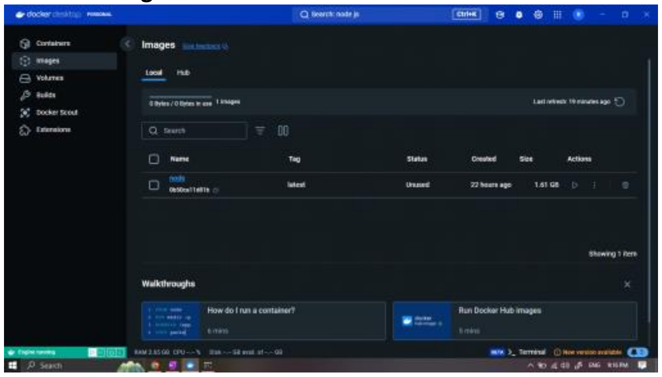
Supported tags and respective `Dockerfile` links

Using CLI : docker pull node



```
C:\Windows\system32\cmd.exe - docker pull node
Microsoft Windows [Version 10.0.19045.5198]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rohi>docker pull node
Using default tag: latest
latest: Pulling from library/node
7d7b66e48eb6: Download complete
551df7f94f9c: Downloading [=======>                                            ]  9.437MB/64.39MB
5bd71677db44: Downloading [=====================================>              ]  17.83MB/23.87MB
ce82e98d553d: Downloading [=>                                                  ]  8.389MB/211.3MB
a05238f4980b: Downloading [=======>                                            ]  8.389MB/56.89MB
9624856701e2: Download complete
fdf894e782a2: Downloading [========>                                           ]  8.389MB/48.5MB
2116f34ae2dd: Download complete
```

Installed Image:



Images

Local Hub

0 Bytes / 0 Bytes in use 1 images Last refresh: 19 minutes ago

Name	Tag	Status	Created	Size	Actions
node 0b50ea11e81b	latest	Unused	22 hours ago	1.61 GB	

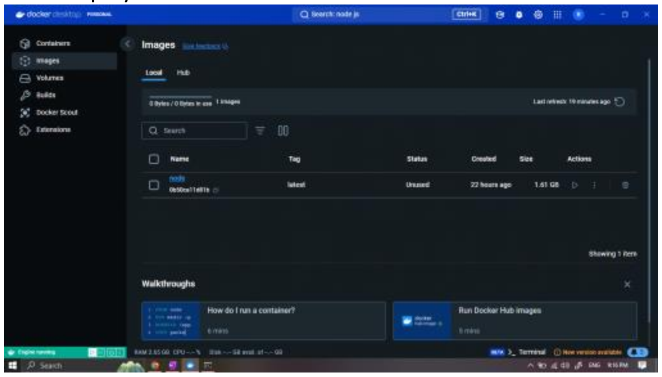
Showing 1 item

Walkthroughs

How do I run a container?

Run Docker Hub images

How to run image:-
Click on play button



Images

Local Hub

0 Bytes / 0 Bytes in use 1 images Last refresh: 19 minutes ago

Name	Tag	Status	Created	Size	Actions
node 0b50ea11e81b	latest	Unused	22 hours ago	1.61 GB	

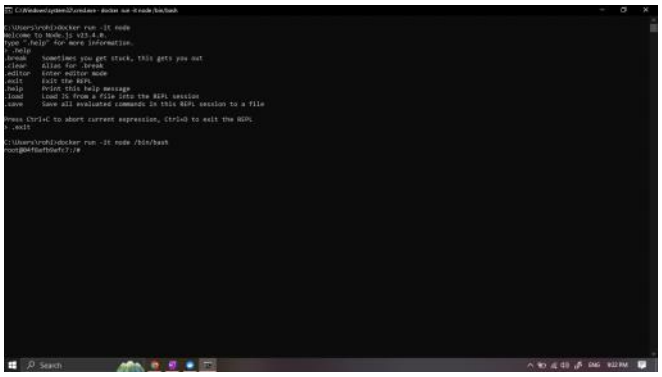
Showing 1 item

Walkthroughs

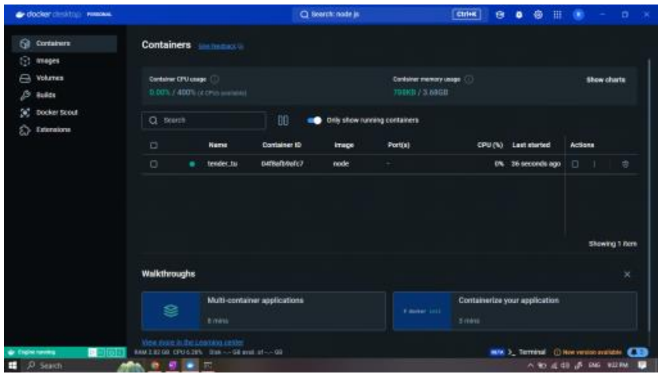
How do I run a container?

Run Docker Hub images

Using CLI : docker run -it node /bin/bash   or docker run -it node



```
C:\Users\rohi>docker run -it node
Welcome to Node.js v23.4.0.
Type ".help" for more information.
> .help
.break    Sometimes you get stuck, this gets you out
.clear    Alias for .break
.editor   Enter editor mode
.exit     Exit the REPL
.help     Print this help message
.load     Load JS from a file into the REPL session
.save     Save all evaluated commands in this REPL session to a file

Press Ctrl+C to abort current expression, Ctrl+D to exit the REPL
> .exit

C:\Users\rohi>docker run -it node /bin/bash
root@04f0afb0efc7:/#
```



Containers

Container CPU usage 0.00% / 400% (4 CPUs available)
Container memory usage 708KB / 3.69GB

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
	tender_tu	04f0afb0efc7	node	-	0%	36 seconds ago	

Showing 1 item

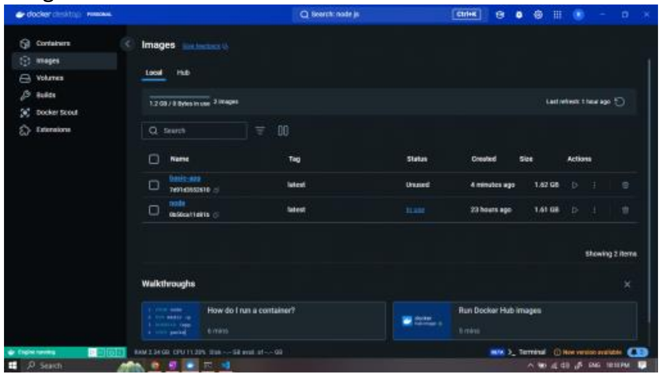
Create sample project:

```
const express = require("express");
const app = express();
app.get("/api",(req,res)=>{
    res.json(
        [
            {
                id:1,
                name:"Rohit",
                Age : 18
            },
            {
                id:2,
                name:"Ram",
                Age : 20
            }
        ]
    )
});
app.listen(3000,()=>{
    console.log("Server Started");
})
```

build a Docker image from a Dockerfile

docker build -t basic-app .

- **docker build**: This is the command to build a Docker image from a Dockerfile.
- **-t basic-app**: The -t flag is used to tag the image with a name. In this case, the image will be named basic-app.
- **.** : This specifies the build context, which is the current directory. Docker will look for a Dockerfile in the current directory to build the image.



Images

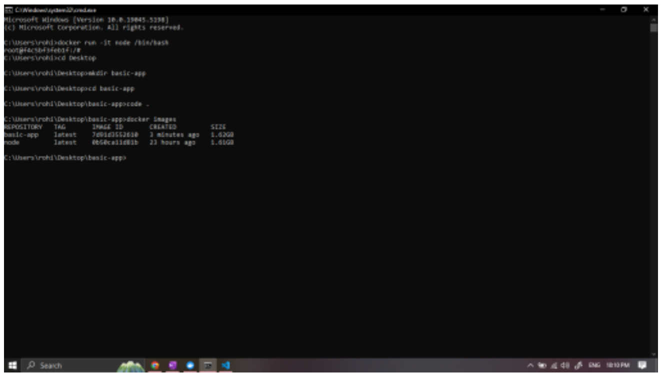
1.2 GB / 0 Bytes in use 2 images

Last refresh: 1 hour ago

	Name	Tag	Status	Created	Size	Actions
	basic-app 7e91d3552610	latest	Unused	4 minutes ago	1.62 GB	
	node 0b56ca11d815	latest	In use	23 hours ago	1.61 GB	

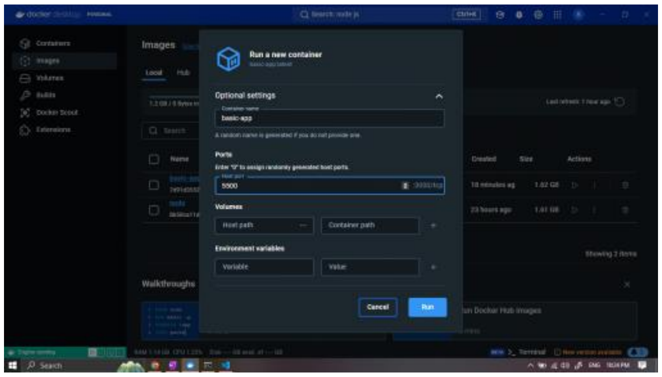
Showing 2 items

Check docker images : docker images



```
Microsoft Windows [Version 10.0.19045.5198]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rohi>docker run -it node /bin/bash
root@f4c5bf3feb1f:/#
C:\Users\rohi>cd Desktop

C:\Users\rohi\Desktop>mkdir basic-app

C:\Users\rohi\Desktop>cd basic-app

C:\Users\rohi\Desktop\basic-app>code .

C:\Users\rohi\Desktop\basic-app>docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
basic-app    latest    7d91d3552610   3 minutes ago    1.62GB
node         latest    0b50ca11d81b   23 hours ago     1.61GB

C:\Users\rohi\Desktop\basic-app>
```

Run project with container :



Run a new container

Optional settings

Container name: basic-app

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port: 5500

Volumes

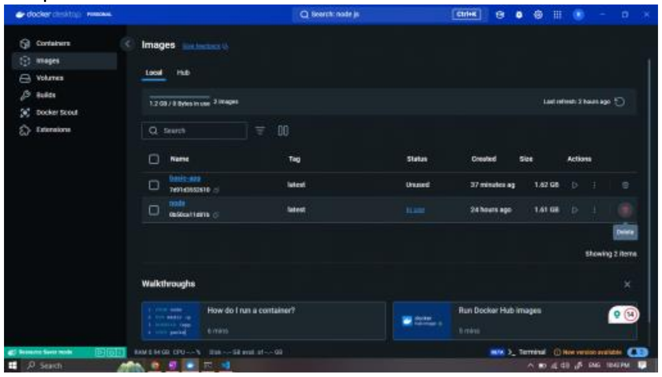
Environment variables

.dockerignore file used to not add unnecessary files.(like node_modules)

Delete Images and Container with terminal

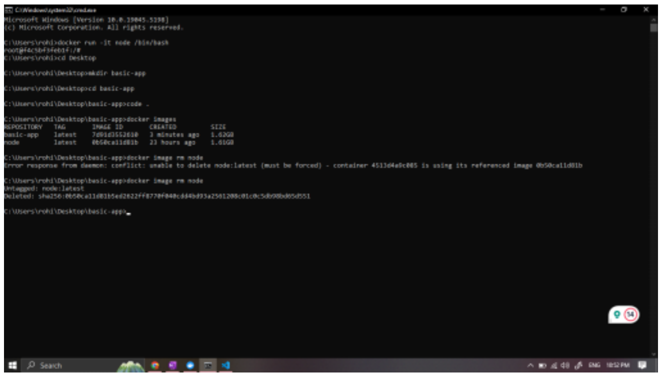
Keypoint : if any image **in use** hai tho. First delete that image container then delete image

Simple click on delete icon



Using CLI :

Docker image rm image_name



```
Microsoft Windows [Version 10.0.19045.5198]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rohi>docker run -it node /bin/bash
root@f4c5bf3feb1f:/#
C:\Users\rohi>cd Desktop

C:\Users\rohi\Desktop>mkdir basic-app

C:\Users\rohi\Desktop>cd basic-app

C:\Users\rohi\Desktop\basic-app>code .

C:\Users\rohi\Desktop\basic-app>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
basic-app    latest    7d91d3552610   3 minutes ago   1.62GB
node         latest    0b50ca11d81b   23 hours ago    1.61GB

C:\Users\rohi\Desktop\basic-app>docker image rm node
Error response from daemon: conflict: unable to delete node:latest (must be forced) - container 4513d4a0c085 is using its referenced image 0b50ca11d81b

C:\Users\rohi\Desktop\basic-app>docker image rm node
Untagged: node:latest
Deleted: sha256:0b50ca11d81b5ed2622ff8770f040cdd4bd93a2561208c01c0c5db98bd05d551

C:\Users\rohi\Desktop\basic-app>_
```

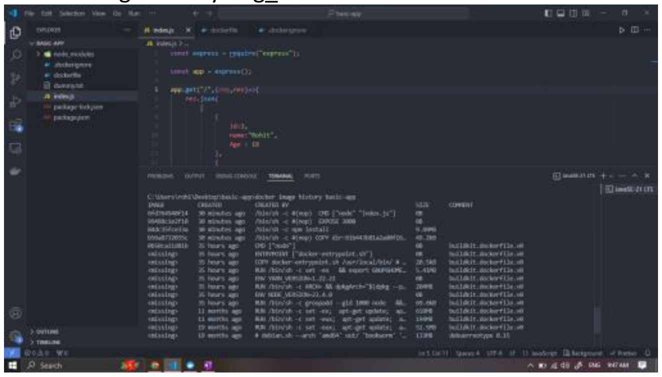
If any image is in use : docker image rm img_name -f

See all container : docker ps -p

Delete container : docker container rm containe_name -f

How to check which version docker image is built upon

docker image history img_name

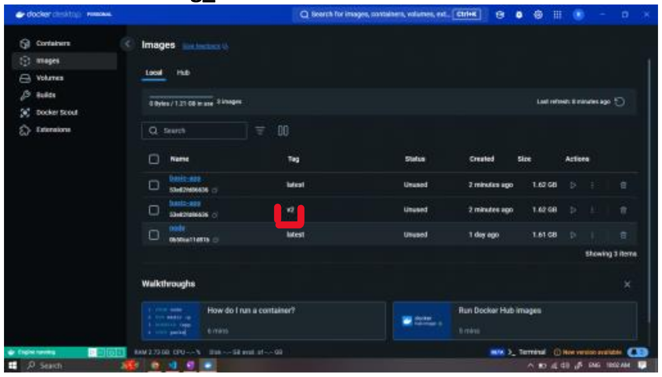


```
const express = require("express");

const app = express();

app.get("/",(req,res)=>{
    res.json(
        [
            {
                id:1,
                name:"Rohit",
                Age : 18
            },
            {
```

```
C:\Users\rohi\Desktop\basic-app>docker image history basic-app
IMAGE          CREATED          CREATED BY                                      SIZE      COMMENT
0fd7049400f14  30 minutes ago   /bin/sh -c #(nop)  CMD ["node" "index.js"]      0B
99408c1a2f10   30 minutes ago   /bin/sh -c #(nop)  EXPOSE 3000                  0B
8ddc35fce13a   30 minutes ago   /bin/sh -c npm install                          9.09MB
b55a8732035c   30 minutes ago   /bin/sh -c #(nop) COPY dir:019443b81a2a00f05…   45.2kB
0b50ca11d81b   35 hours ago     CMD ["node"]                                    0B        buildkit.dockerfile.v0
<missing>      35 hours ago     ENTRYPOINT ["docker-entrypoint.sh"]             0B        buildkit.dockerfile.v0
<missing>      35 hours ago     COPY docker-entrypoint.sh /usr/local/bin/ # …   20.5kB    buildkit.dockerfile.v0
<missing>      35 hours ago     RUN /bin/sh -c set -ex   && export GNUPGHOME…   5.41MB    buildkit.dockerfile.v0
<missing>      35 hours ago     ENV YARN_VERSION=1.22.22                        0B        buildkit.dockerfile.v0
<missing>      35 hours ago     RUN /bin/sh -c ARCH= && dpkgArch="$(dpkg --p…   204MB     buildkit.dockerfile.v0
<missing>      35 hours ago     ENV NODE_VERSION=23.4.0                         0B        buildkit.dockerfile.v0
<missing>      35 hours ago     RUN /bin/sh -c groupadd --gid 1000 node   &&…   09.6kB    buildkit.dockerfile.v0
<missing>      11 months ago    RUN /bin/sh -c set -ex;  apt-get update;  ap…   610MB     buildkit.dockerfile.v0
<missing>      11 months ago    RUN /bin/sh -c set -eux;  apt-get update;  a…   184MB     buildkit.dockerfile.v0
<missing>      13 months ago    RUN /bin/sh -c set -eux;  apt-get update;  a…   51.5MB    buildkit.dockerfile.v0
<missing>      13 months ago    # debian.sh --arch 'amd64' out/ 'bookworm' '…   133MB     debuerreotype 0.15
```

How to delete all images from docker
docker system prune -a

Create image with version:
docker build -t img_name:version .



Images

0 Bytes / 1.21 GB in use 3 images Last refresh: 8 minutes ago

Name	Tag	Status	Created	Size
basic-app 53e82f686636	latest	Unused	2 minutes ago	1.62 GB
basic-app 53e82f686636	v2	Unused	2 minutes ago	1.62 GB
node 0b50ca11d81b	latest	Unused	1 day ago	1.61 GB

Showing 3 items

How to run docker container with version
docker run --name basic-app-container-v2 -p 3000:3000 basic-app:v2

What is volume in docker?

a volume is a mechanism for persisting data generated and used by Docker containers.

1. **docker build -t img_name .**
2. **docker run --name container_name -p portno:protno --rm -v path:workdir img_name**

What is compose file?

A **Docker Compose file** is a YAML file (docker-compose.yml) that defines how to configure and run multi-container Docker applications.
It provides a convenient way to manage complex setups by describing services, networks, and volumes required for the application.
Instead of running multiple docker run commands, you can define all your containers and configurations in one file and use a single docker-compose command to manage them.

Key Features of a Compose File

1. **Services**: Defines the containers to be run, including their images, ports, and commands.
2. **Networks**: Allows containers to communicate with each other in an isolated environment.
3. **Volumes**: Manages persistent data storage for containers.
4. **Environment Variables**: Defines environment variables for each container.
5. **Scalability**: Easily scale services to run multiple container instances.

Example docker-compose.yml

Here's a simple example of a docker-compose.yml for a Node.js application and a MongoDB database:

```
version: '3.9'  # Compose file format version
services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    volumes:
      - .:/app
    environment:
      - NODE_ENV=development
    depends_on:
      - db
db:
    image: mongo:latest
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db
volumes:
  mongo-data:
```

Explanation of the Compose File

1. **Version**:
   - Specifies the Docker Compose file format version.
   - Latest version as of now is 3.9.
2. **Services**:
   - Defines the containers in the application.
   - **app**: The Node.js service:
     - build: Builds the image from the specified Dockerfile in the current directory (.).
     - ports: Maps port 3000 on the host to port 3000 in the container.
     - volumes: Mounts the current directory to /app in the container for hot reloading.
     - environment: Passes environment variables (e.g., NODE_ENV=development).
     - depends_on: Ensures the db service starts before app.
   - **db**: The MongoDB service:
     - Uses the official MongoDB image.
     - Maps port 27017 for database access.
     - Uses a volume for persistent storage.
3. **Volumes**:
   - mongo-data: A named volume for persisting MongoDB data.

How to Use the Compose File

1. **Save the File**: Save the above YAML configuration as docker-compose.yml.
2. **Start the Application**: Use the docker-compose up command to start the application:

docker-compose up
3. **Run in Detached Mode**: Run containers in the background with -d:
   docker-compose up -d
4. **Stop the Application**: Stop and remove the containers:
   docker-compose down
5. **Build the Images**: Force a rebuild of the images:
   docker-compose up --build

Benefits of Docker Compose
- Simplifies multi-container setups.
- Reduces repetitive command-line instructions.
- Enables easy configuration sharing through the YAML file.
- Allows for version-controlled, reproducible environments.
- Supports both development and production configurations with different Compose files.

This makes docker-compose an essential tool for managing Docker-based applications.

How to share image to docker hub
docker build -t repo_name .
docker login
docker push repo_name
Docker pull repo_name

Upload Image on server | play-with-docker

1. docker buildx build --platform linux/amd64 -t rrkatkar2024/node-demo .
2. docker push rrkatkar2024/node-demo
3. docker push rrkatkar2024/node-demo
4. docker push rrkatkar2024/node-demo

rrkatkar2024/node-demo --> repo name