Docker

What is Docker ?

Platform for building , running, and shipping applications.

Container Vs Virtual Machine

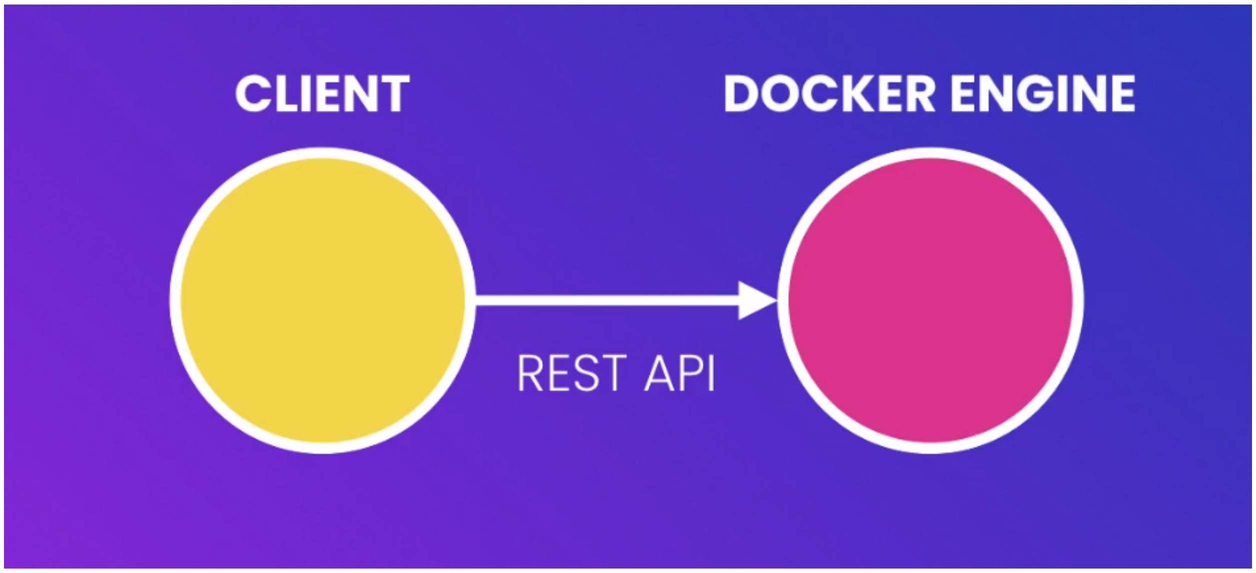
In virtual machine,

- different os
- intensive resources
- takes time

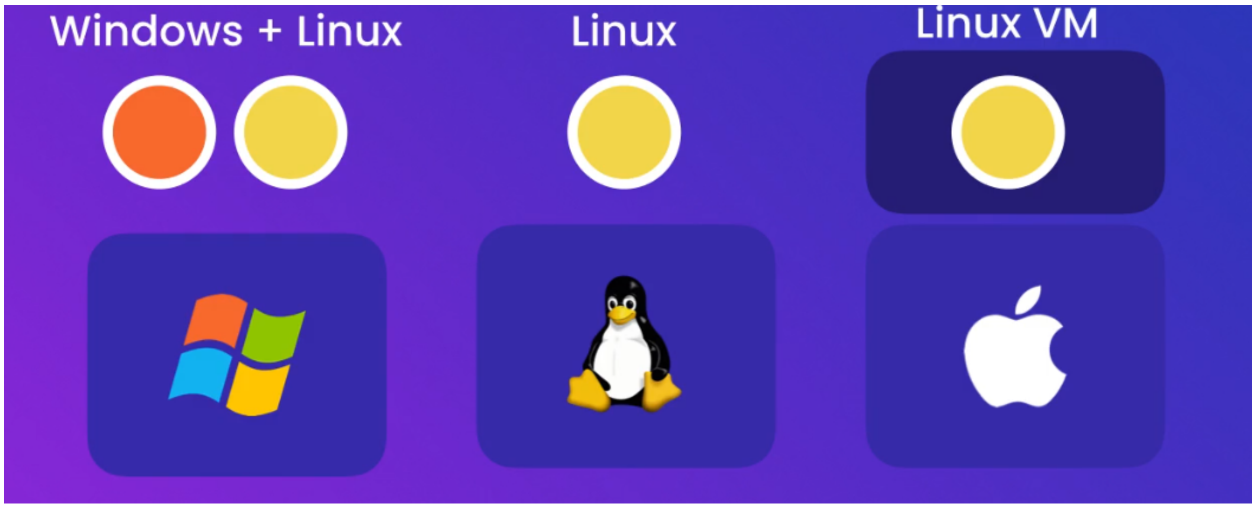
In container

- multiple app in isolation
- lightweight
- use os of host
- start quickly
- less hardware resources

Docker Architecture



Container use host operating system’s kernel to run applications



Docker Tutorial →

- Pull image from docker hub

```
docker pull ubuntu
```

- Run docker image if not available first pull

```
docker run ubuntu
```

- Show all container

```
docker ps -a
```

- Run container interactively

```
docker run -it ubuntu
```

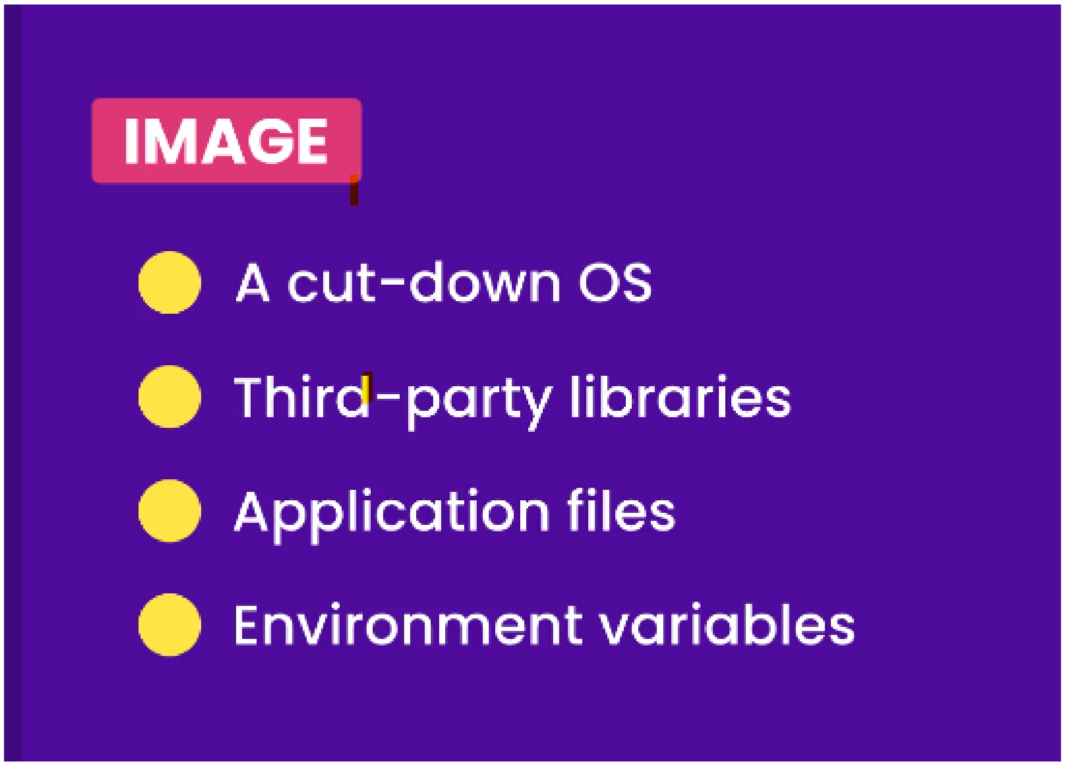
- Start Container

```
docker start -i container_id
```

- Start new bash session

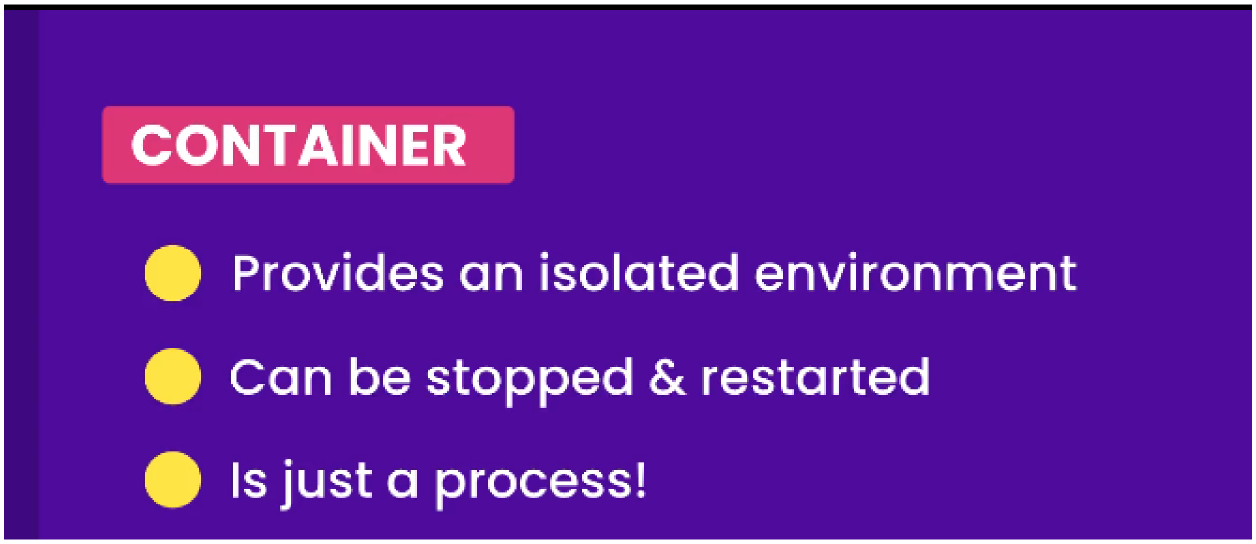
```
docker exec -it -u user_name container_id bash
```

Images And Container



IMAGE

- A cut-down OS
- Third-party libraries
- Application files
- Environment variables



CONTAINER

- Provides an isolated environment
- Can be stopped & restarted
- Is just a process!

A **Dockerfile** contains instructions for building an image

1. FROM → Specify base image

2. WORKDIR → Specify current working directory
3. COPY → Copying files and dir
4. ADD → Adding files and dir
5. RUN → running commands
6. ENV → setting environment variables
7. EXPOSE → telling on which port container is starting
8. USER → specify user which should run app
9. CMD
10. ENTRYPOINT

FROM :

Used to specify base image. It can be Windows or Linux or any other OS or node or python

```
FROM node:14.20.0-alpine3.16
```

Building Image and other Configurations :

-t → to give tag to image

. (next argument) → directory where it can find Dockerfile

```
docker build -t react-app .
```

- To see all images

```
docker image ls / docker images
```

- To start container with shell

```
docker run -it react-app sh
```

COPY :

```
COPY files_to_be_copied where_to_paste
# if file name contains space (array represent two arguments of COPY command)
COPY ["hello word.txt", "."]
```

WORKDIR :

Set Working dir

```
WORKDIR /app
```

ADD :

Same syntax as COPY. But it provides extra functionality

- Add file from url
- Can pass compressed file as argument it automatically uncompress it while copying.

To Exclude file from copying in build context i.e while building image

- Create file `.dockerignore` and add files to be excluded.

RUN :

- To run command inside container

```
RUN npm install
```

ENV :

- To add an environment variable like url of api

```
ENV API_URL=http://api.myapp.com/
```

EXPOSE :

A form of documentation that tells this container listens on port

```
EXPOSE 3000
```

- Add user to OS using RUN

USER :

- Set user

```
USER app
```

CMD :

- Set default command to be executed
- If multiple command instructions are present then recent will executed

```
# Shell Form -> Executed inside separate shell - /bin/sh or cmd
CMD npm start
# Exec Form -> Use this form since it not start new shell and execute command directly.
CMD ["npm", "start"]
```

- RUN is executed while building an image while CMD is run time instruction it is executed while starting container.

ENTRYPOINT :

```
# Shell Form -> Executed inside separate shell - /bin/sh or cmd
ENTRYPOINT npm start
```

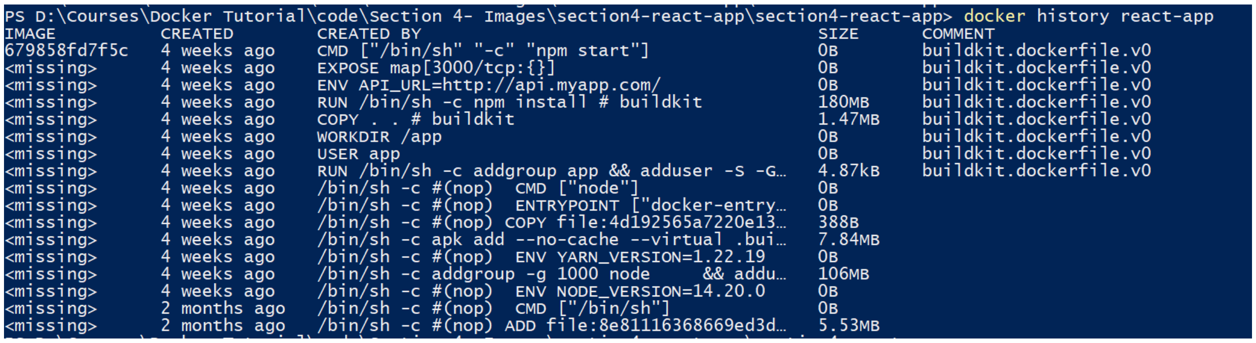
```
# Exec Form -> Use this form since it not start new shell and execute command directly.
ENTRYPOINT ["npm", "start"]
```

- ENTRYPOINT instruction same as CMD but cannot be overridden easily (to override use —entrypoint)

```
docker run react-app sh -> will override CMD (but not ENTRYPOINT)
```

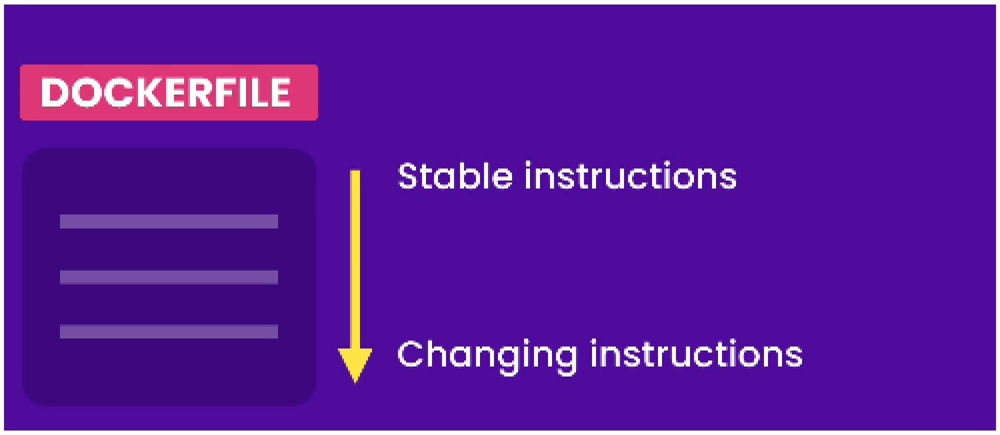
- Images are created by creating layer over them so to keep track of modified files

```
# To see history while creating image
docker history react-app (image name)
```



```
PS D:\Courses\Docker Tutorial\code\Section 4- Images\section4-react-app\section4-react-app> docker history react-app
IMAGE          CREATED        CREATED BY                                      SIZE      COMMENT
679858fd7f5c   4 weeks ago    CMD ["/bin/sh" "-c" "npm start"]                0B        buildkit.dockerfile.v0
<missing>      4 weeks ago    EXPOSE map[3000/tcp:{}]                         0B        buildkit.dockerfile.v0
<missing>      4 weeks ago    ENV API_URL=http://api.myapp.com/               0B        buildkit.dockerfile.v0
<missing>      4 weeks ago    RUN /bin/sh -c npm install # buildkit           180MB     buildkit.dockerfile.v0
<missing>      4 weeks ago    COPY . . # buildkit                             1.47MB    buildkit.dockerfile.v0
<missing>      4 weeks ago    WORKDIR /app                                    0B        buildkit.dockerfile.v0
<missing>      4 weeks ago    USER app                                        0B        buildkit.dockerfile.v0
<missing>      4 weeks ago    RUN /bin/sh -c addgroup app && adduser -S -G…   4.87kB    buildkit.dockerfile.v0
<missing>      4 weeks ago    /bin/sh -c #(nop)  CMD ["node"]                 0B
<missing>      4 weeks ago    /bin/sh -c #(nop)  ENTRYPOINT ["docker-entry…   0B
<missing>      4 weeks ago    /bin/sh -c #(nop) COPY file:4d192565a7220e13…   388B
<missing>      4 weeks ago    /bin/sh -c apk add --no-cache --virtual .bui…   7.84MB
<missing>      4 weeks ago    /bin/sh -c #(nop)  ENV YARN_VERSION=1.22.19     0B
<missing>      4 weeks ago    /bin/sh -c addgroup -g 1000 node     && addu…   106MB
<missing>      4 weeks ago    /bin/sh -c #(nop)  ENV NODE_VERSION=14.20.0     0B
<missing>      2 months ago   /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      2 months ago   /bin/sh -c #(nop) ADD file:8e81116368669ed3d…   5.53MB
```

- If instruction is not changed then docker will not create new layer for it instead will use existing layer from cache
- If for any instruction new layer created then following instructions also need to be rebuild
- So first COPY package*.json and RUN npm install and then COPY . .



Deleting Images and Containers :

> Adding User to Alpine ⇒ adduser -S → create system user -G → group name

- Deleting images which are dangling means not pulled but created as a part of layering.

```
docker image prune
```

- Deleting Containers which are exited.

```
docker container prune
```

- Deleting specific image

```
docker image rm <container-id/name> .. .. ..
```

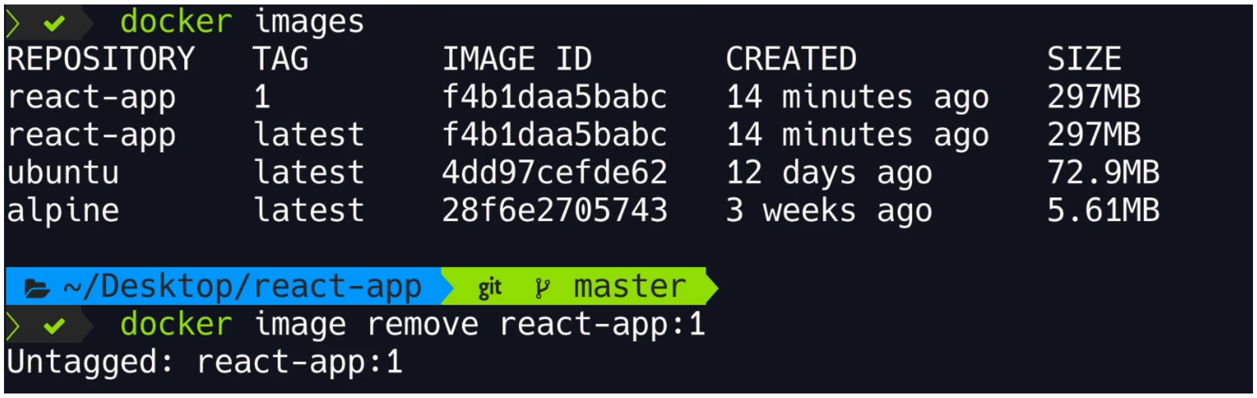
Tags :

- By default Docker uses `latest` tag (it doesn't mean that a latest image it may point to old image)

```
docker build -t react-app:<tag> .
```

> Two images with same name and different tag can exist

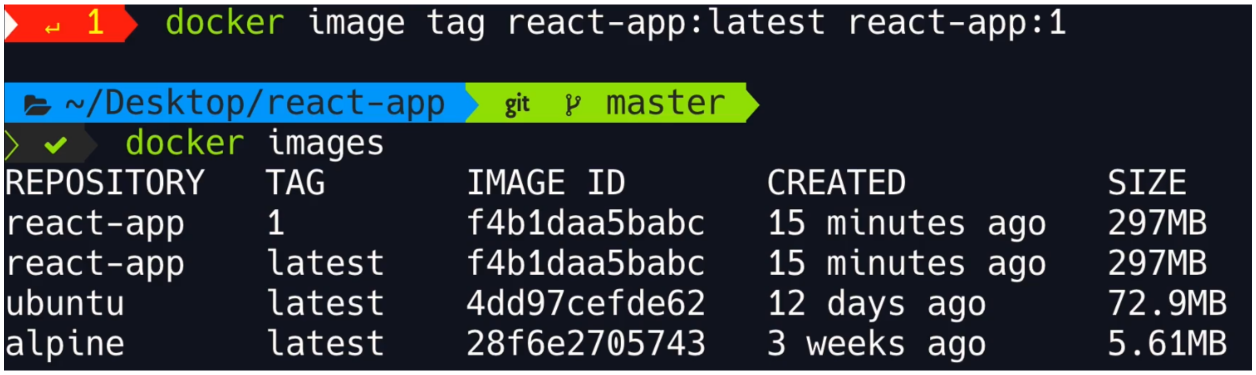
```
# Remove specific image with tag
docker image rm <image_name>:<tag>
```



```
❯ ✔  docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
react-app     1         f4b1daa5babc   14 minutes ago   297MB
react-app     latest    f4b1daa5babc   14 minutes ago   297MB
ubuntu        latest    4dd97cefde62   12 days ago      72.9MB
alpine        latest    28f6e2705743   3 weeks ago      5.61MB

 ~/Desktop/react-app  git  master
❯ ✔  docker image remove react-app:1
Untagged: react-app:1
```

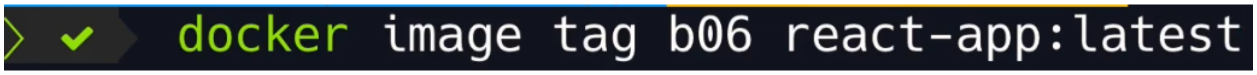
Adding Tag After Building it

```
docker image tag <pre_image_name>:<pre_tag> <new_image_name>:<new_tag>
```



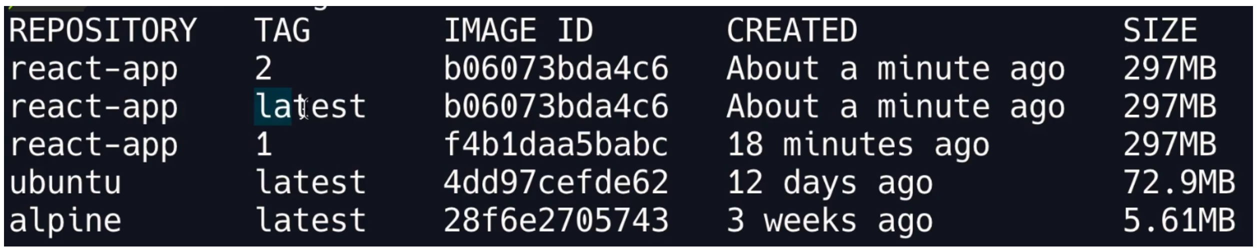
```
↵ 1  docker image tag react-app:latest react-app:1

 ~/Desktop/react-app  git  master
❯ ✔  docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
react-app     1         f4b1daa5babc   15 minutes ago   297MB
react-app     latest    f4b1daa5babc   15 minutes ago   297MB
ubuntu        latest    4dd97cefde62   12 days ago      72.9MB
alpine        latest    28f6e2705743   3 weeks ago      5.61MB
```

> Latest tag may point to old image we need explicitly change current tag to latest



```
❯ ✔  docker image tag b06 react-app:latest
```

⇒


```
REPOSITORY   TAG      IMAGE ID       CREATED              SIZE
react-app    2        b06073bda4c6   About a minute ago   297MB
react-app    latest   b06073bda4c6   About a minute ago   297MB
react-app    1        f4b1daa5babc   18 minutes ago       297MB
ubuntu       latest   4dd97cefde62   12 days ago          72.9MB
alpine       latest   28f6e2705743   3 weeks ago          5.61MB
```

Sharing images :

- Give explicitly tag to image

```
docker image tag react-app:2 kunalchaudhari1/react-app:2
```

- Login to docker

```
docker login
```

- Push docker image (push happens layerwise)

```
docker push kunalchaudhari1/react-app:1
```

Saving and Loading Images

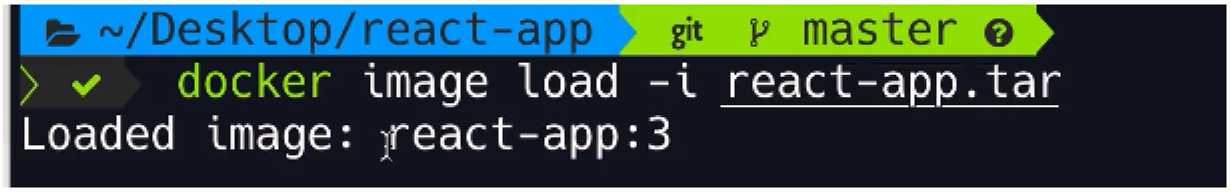
o, --output string Write to a file, instead of STDOUT

```
docker image save -o react-app.tar react-app:3 #(single or multiple images)
```

images is stored layer wise each layer contain specific file to that layer

i, --input string Read from tar archive file, instead of STDIN q, --quiet Suppress the load output

```
docker image load [OPTIONS]
```



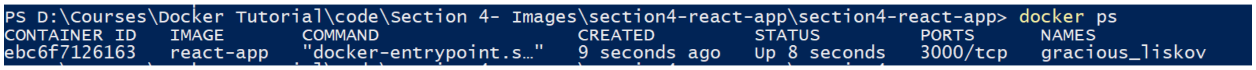
```
~/Desktop/react-app  git  master
docker image load -i react-app.tar
Loaded image: react-app:3
```

Containers :

> Container is a special type of process which has its own file system provided by image

- To run Container in background

```
docker run -d react-app
```

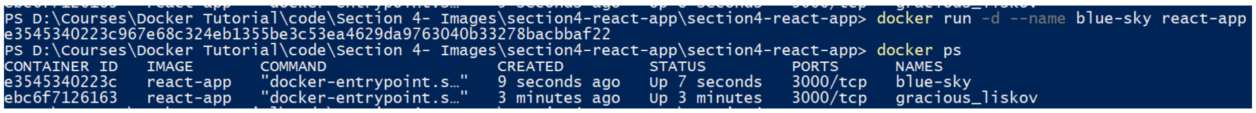


```
PS D:\Courses\Docker Tutorial\code\Section 4- Images\section4-react-app\section4-react-app> docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS         PORTS      NAMES
ebc6f7126163   react-app   "docker-entrypoint.s…"   9 seconds ago    Up 8 seconds   3000/tcp   gracious_liskov
```

Docker automatically associates each container with random name

- To give custom name use —name

```
docker run -d --name blue-sky react-app
```



```
PS D:\Courses\Docker Tutorial\code\Section 4- Images\section4-react-app\section4-react-app> docker run -d --name blue-sky react-app
e3545340223c967e68c324eb1355be3c53ea4629da9763040b33278bacbbaf22
PS D:\Courses\Docker Tutorial\code\Section 4- Images\section4-react-app\section4-react-app> docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS      NAMES
e3545340223c   react-app   "docker-entrypoint.s…"   9 seconds ago   Up 7 seconds   3000/tcp   blue-sky
ebc6f7126163   react-app   "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   3000/tcp   gracious_liskov
```

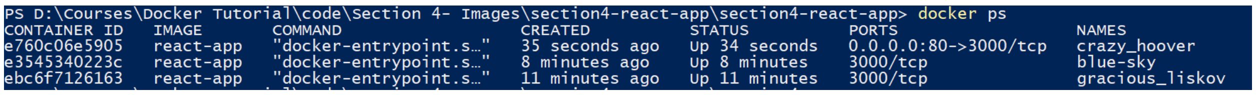
- To see whats going on in container in bg use logs

```
docker logs <container-id/name>
# to see last 5 lines
docker logs -n 5 <container-id/name>
# to see timestamp infront of each line
docker logs -t <container-id/name>
```

Publish Port:

For actual working of node, we need to publish port of container to port of host

```
docker run -d -p <host-port>:<container-port>
#example
docker run -d -p 80:3000 react-app
```



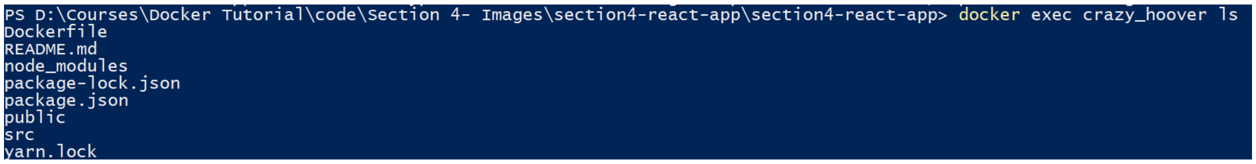
```
PS D:\Courses\Docker Tutorial\code\Section 4- Images\section4-react-app\section4-react-app> docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                  NAMES
e760c06e5905   react-app   "docker-entrypoint.s…"   35 seconds ago   Up 34 seconds   0.0.0.0:80->3000/tcp   crazy_hoover
e3545340223c   react-app   "docker-entrypoint.s…"   8 minutes ago    Up 8 minutes    3000/tcp               blue-sky
ebc6f7126163   react-app   "docker-entrypoint.s…"   11 minutes ago   Up 11 minutes   3000/tcp               gracious_liskov
```

Running Command into already running Container

run → start new container and run a command

exec → execute a command on already running a container

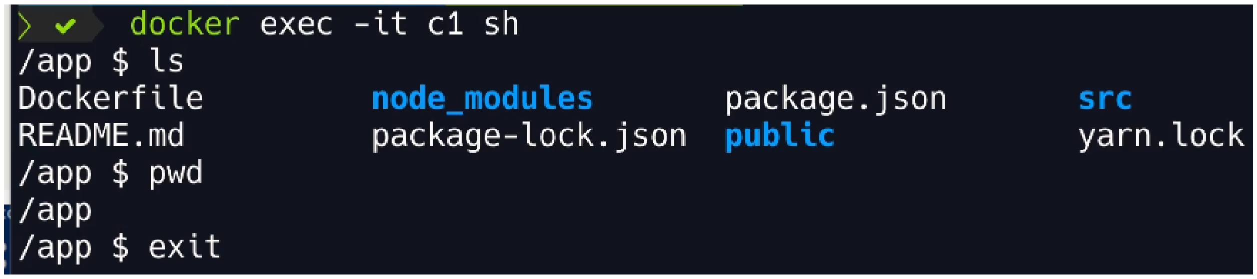
```
docker exec blue-sky ls
            (container) (command)
```



```
PS D:\Courses\Docker Tutorial\code\Section 4- Images\section4-react-app\section4-react-app> docker exec crazy_hoover ls
Dockerfile
README.md
node_modules
package-lock.json
package.json
public
src
yarn.lock
```

- For interactive mode

```
docker exec -it <container> sh
```



```
docker exec -it c1 sh
/app $ ls
Dockerfile          node_modules        package.json        src
README.md           package-lock.json   public              yarn.lock
/app $ pwd
/app
/app $ exit
```

Stopping and Starting Container

```
# Stop container
docker stop blue-sky
# Start container
docker start blue-sky
```

Remove Container

```
docker container rm <container-name>
#or
docker rm <container-name>
docker rm -f <container-name>
```

Copying Files

- From Computer to Host

```
docker cp <container-id:<absolute_path>> .
            (Source)                      (Dest)
```

- From Host to Computer

  Reverse above command

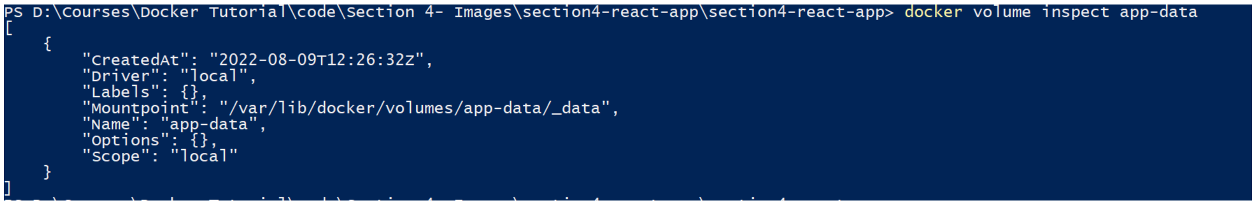
Persisting Data Using Volumes

- Volume : It is storage outside container
- Create Volume

```
docker volume create app-data
# list all volumes
docker volume ls
```

- Inspecting Volume

```
docker volume inspect <volume-name>
```



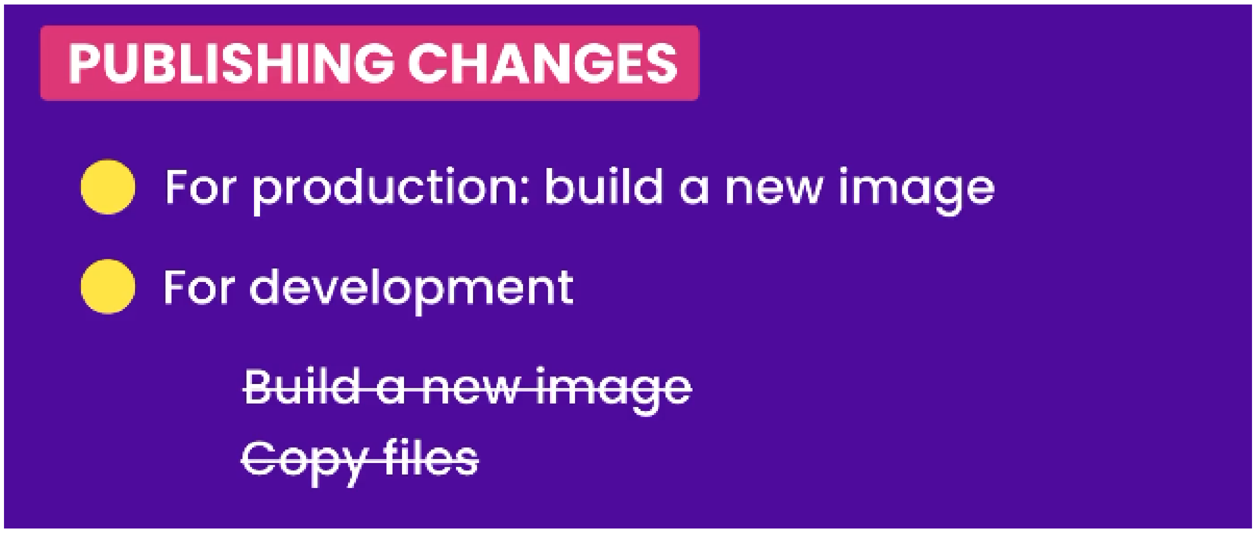
```
PS D:\Courses\Docker Tutorial\code\Section 4- Images\section4-react-app\section4-react-app> docker volume inspect app-data
[
    {
        "CreatedAt": "2022-08-09T12:26:32Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/app-data/_data",
        "Name": "app-data",
        "Options": {},
        "Scope": "local"
    }
]
```

- Mapping volume to directory in filesystem of container
    - if volume name or dir in container not exist it will create it
    - → Problem with above is →
        - Dir that automatically created by docker doesn’t have write permission for other users
        - So create dir using docker file

```
docker run -d -p 5001:3000 -v app-data:/app/data react-app
```

- Deleting container not delete files from volume folder
- So it act as persistent storage which can be shared across multiple container

Sharing Source Code with Container



PUBLISHING CHANGES

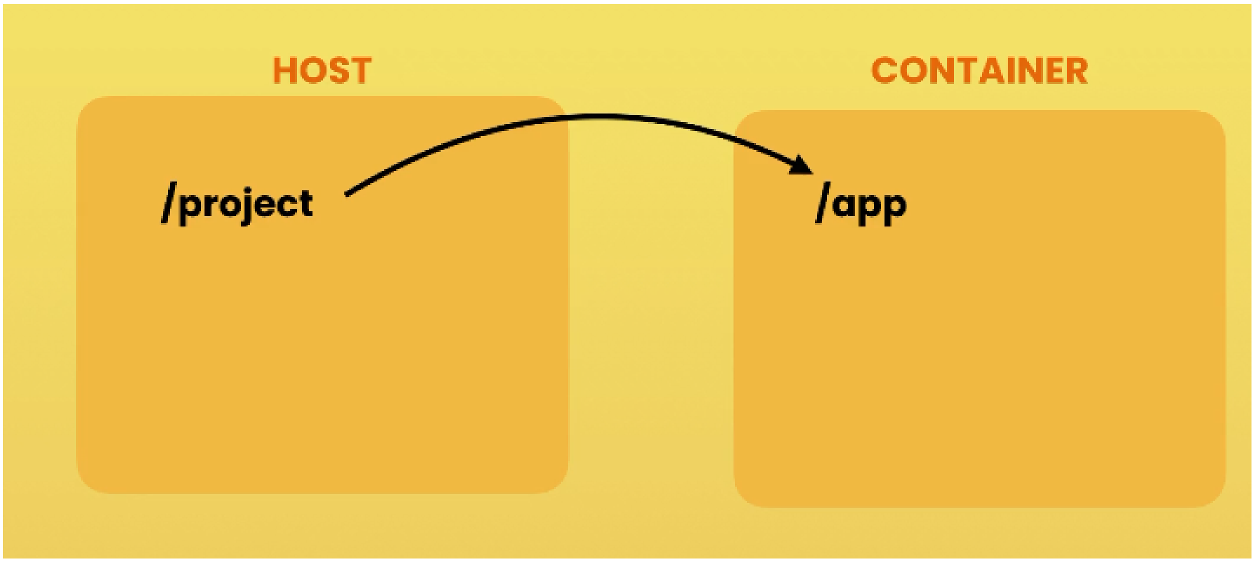
- For production: build a new image
- For development

  ~~Build a new image~~

  ~~Copy files~~

Binding source code directory with container director

It is important so that any changes made in source code will be available at container



```
# Map dir to container dir
docker run -d -p 5001:3000 -v $(pwd):/app react-app
```

Multi-Container Application

```
# get all images id
docker image ls -q
# Remove all images
docker image rm $(docker image ls -q)
# Remove all containers
docker container rm -f $(docker container ls -aq)
```

docker-compose.yml

```
docker-compose up
```

> Parsing YAML file is slower than JSON

YAML →

- No double quotes
- No curly braces
- Array represented using indentation and hyphen -
- Hierarchy represented using indentation

YAML → Used for cofiguration

JSON → Exchanging data between hosts