# pull, image, run contaner

docker pull nginx:latest

# create a container from an image

docker run nginx:latest

run in detached mode

docker run -d nginx:lastest

docker container ls

docker image ls

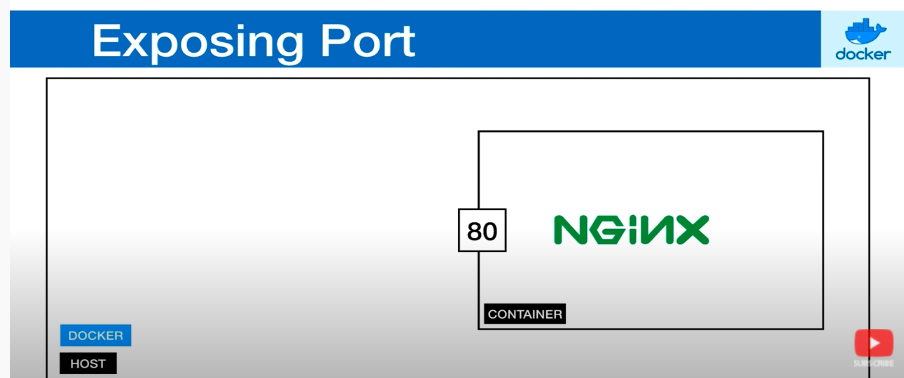
equivalent to

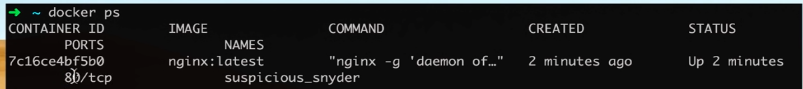
docker ps

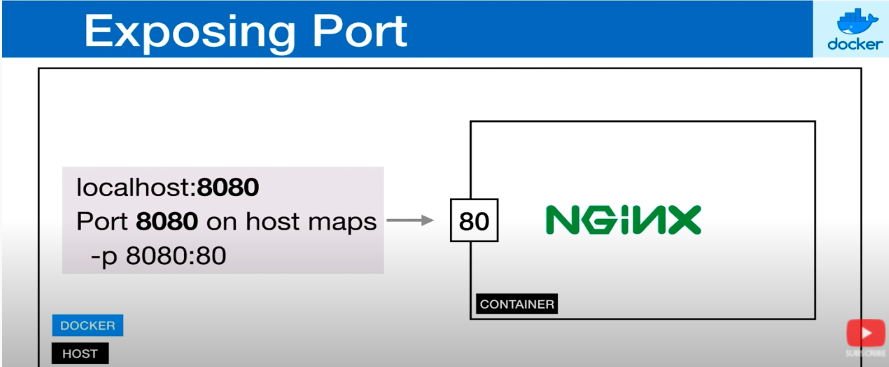
docker stop <container\_id>/<container\_name>

# Exposing port

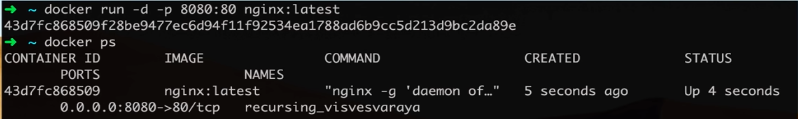
Map from host server to container



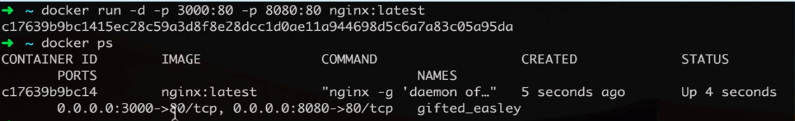




docker run -d -p 8080:80 nginx:latest



# Exposing multiple ports



# Managing containers

## help

docker ps --help

docker container ls --help

docker ps -> list all running containers

docker start <container\_id>/<container\_name>

docker stop <container\_id>/<container\_name>

docker container ls

docker container ls -a

docker container rm <container-id>/<container-name>

equivalent to

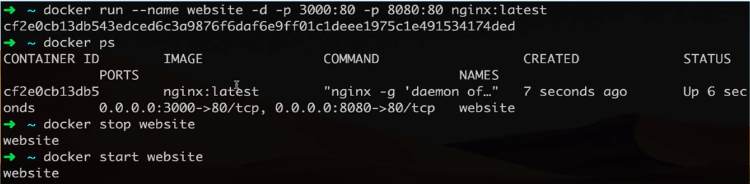
docker rm <container-id>/<container-name>

docker container rm $(docker ps -aq)

stop all containers before attempting removal or force removal

docker rm -f $(docker ps -aq)

# container naming

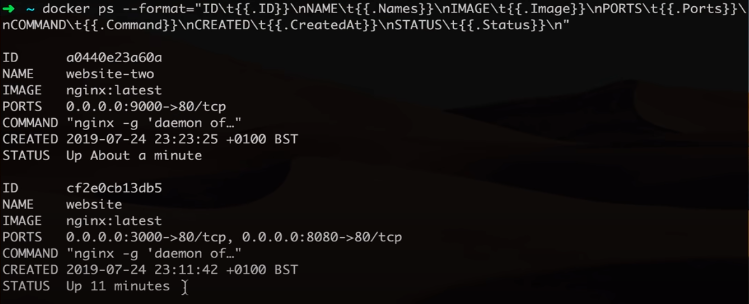
docker run --name website -d -p 3000:80 nginx:latest

# docker ps

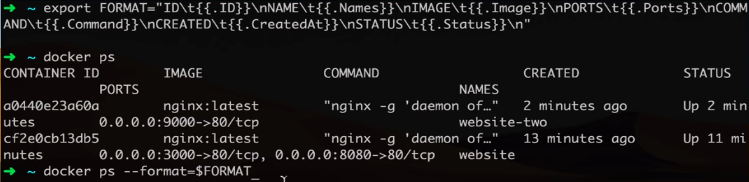
docker ps

docker ps -a

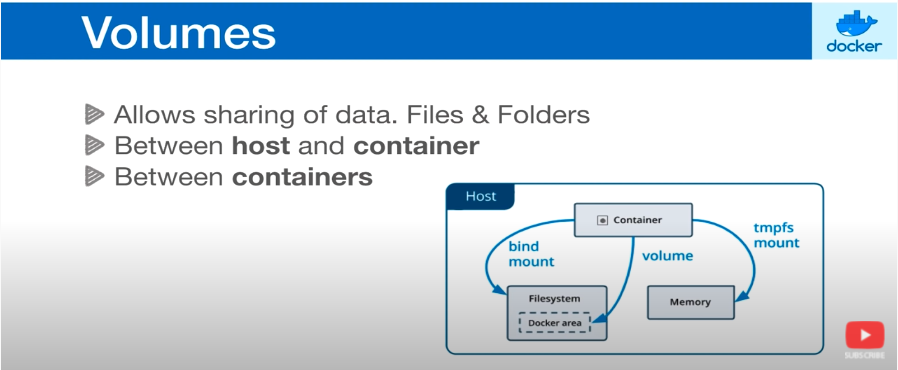
docker ps -aq

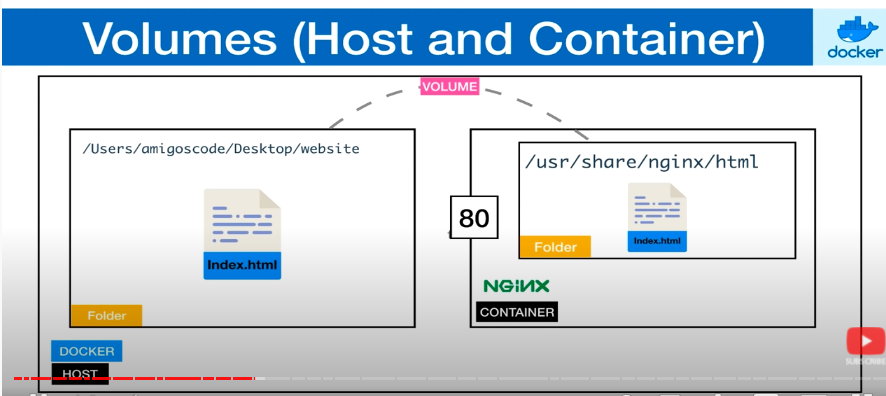


docker ps –format=”<your-format>”



# Docker volumes: Allows to share data between host and container or between containers

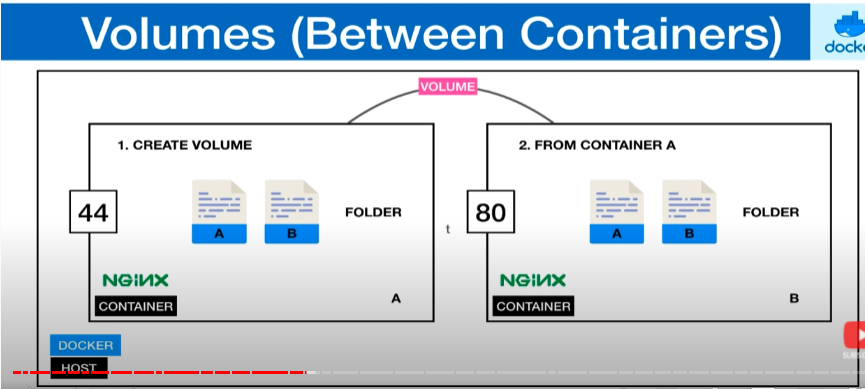


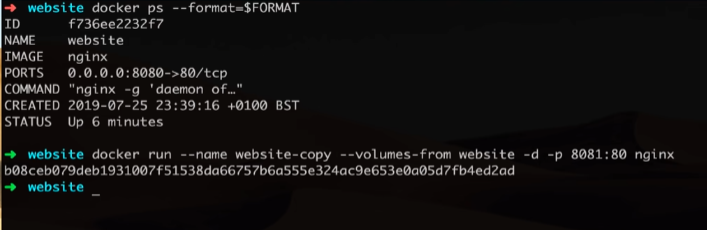


docker run –name website -d -p 8080:80 -v <host-src>:<container-dest>:ro nginx:latest

then <host-src> is mounted to <container-dest> as read only, they two are synced

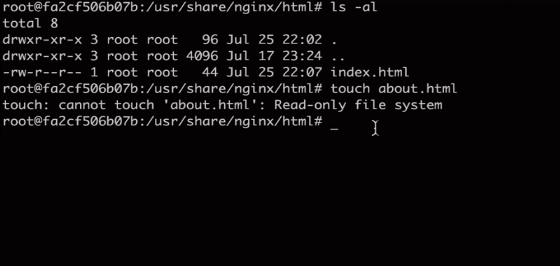
# share volumes between containers





# exec interactively inside a container

docker exec -it website bash



# Dockerfile to build image

Dockerfile reference: <https://docs.docker.com/engine/reference/builder/>

Define a series of steps of how to build your image

docker build -t (--tag) website:latest .

this will look for the docker file under .

example Dockerfile:

FROM node:latest

WORKDIR /app

ADD . .

RUN npm install

CMD node index.js

(default working dir is / if not specified)

# .dockerignore

Files that don’t need to be included when creating the image

Example .dockerignore:

node\_modules

Dockerfile

.git

\*.gulp.js

<folder\_name>/\*\*

# caching and layers

Example .dockerignore:

FROM node:latest

WORKDIR /app

ADD package\*.json ./

RUN npm install

ADD . .

CMD node index.js

If the content of a step is not changed, then docker will use the cached layer instead of running that step again. If the content of one step is modified, then all following steps will be re-built.

# alpine version

node:alpine

nginx:alpine

# tags, versioning

node:10.16.1-alpine

docker tag nginx:latest nginx:1.00

then nginx:latest and nginx:1.00 will refer to the same image id

# Docker registry

Like a repo

Docker Hub

Quay.io

Amazon ECR

## take docker hub for instance

docker login

docker tag <local-image> <standard-image-name>

docker push <account-name>/<repo-name>:<tag-name>

docker pull <account-name>/<repo-name>[:<tag-name>]

(default tag is latest)

# docker inspect

docker inspect <docker-id> or <docker-name>

# view container logs

docker logs <container-name> or <container-id>

docker logs -f

--since

--tail

--until

# entrypoint vs cmd

entrypoint + cmd is the full start command