IMAGES & CONTAINERS

What is an Image

- It has
  - Lite version OS
  - 3rd party libraries
  - Appln files
  - Environment variables

What is Container

- Provides an Isolated Environment
- Can be started & Stopped
- It is just an process

Sample web application.

Steps to start an app normally

- Install node
- npm install
- npm start

Docker file instructions

FROM

WORKDIR

COPY

ADD

RUN

ENV

EXPOSE

USER

CMD

ENTRY POINT

Choosing right Base image.

FROM → which image you want to use?

Eg: DJANGO
FROM python:3

∴ Always use specific version. Dont use latest

ALPINE

FROM node:14.16.0-alpine3.13

docker build -t react-app .

docker images

docker run -it react-app

docker run -it react-app sh

node <- version.

Copying files and directories

Copy application files into image

FROM .....

COPY
ADD
} One of these can be used

COPY

COPY has 2 arguments
1st - Source
2nd - Destination.

- COPY package.json /app
OR
- COPY package*.json /app
OR
if there is space in file name
COPY ["hello world", "."]

Each argument can be put in this array

or
COPY

ADD has 2 added features

1) Can copy from website url
ADD http://.../file.json .
2) If added Compress file, ADD will decompress

ADD file.zip .

Checking if files copied

docker build -t react-app .

docker run -it react-app sh

/app #

↳ we are inside app because it is active work direc

Excluding files & directories

In our previous build the build context was 150mb because docker client copied all files to docker engine

It was because of node modules

This is a problem we dont need to transfer node modules.

2 benefits

1) build context is smaller
   - less data to be transfered over network
2) Build will be faster.

how to exclude	Similar to .gitignore we have .dockerignore
Inside .dockerignore	node_modules/
Build context now	reduced to 10.16 kb from 150MB.
Now what?	Since we dont have node_modules folder, we need to run NPM install

Running Commands

Current dockerfile

```
FROM node:14.....alpine2.13
WORKDIR /app
COPY . .
```

↳ RUN npm install

Rebuilding image:

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

This now runs npm install

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

```
/app # ls -l
```

It now has node modules directory.

Setting Environment Variable

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

Rebuild img.
Start new container.

```
/app #printenv
    or
printenv API_URL
    or
echo $API_URL
```

Exposing ports.

The port 3000 is exposed in container but the host is not exposed yet. We will in next lesson.

EXPOSE 3000

Setting USER

Default — Runs in Root user.

Best practice — Create a user with limited privileges.

```
docker run -it alpine
/ # Useradd
  Useradd: not found
/ # adduser
```

Create System User.

1. Create group.
   addgroup app.
2. adduser -S -G app app

Verify.

groups app

app

addgroup nagz
adduser -S -G nagz nagz

Current

```
FROM node:14.16.0-alpine3.13
WORKDIR /app
COPY . .
RUN npm install
ENV API_URL = ....
EXPOSE 3000
RUN addgroup app &&
    adduser -S -G app
    app
USER app
```

Buied

Run shell comm.

$ whoami

Defining entrypoints

If we do

```
docker run react-app
```

then

it immediately stops because we didn't specify program to execute.

now lets do.

```
docker run react-app
    npm start
```

Error thrown

Permission error

`mkdir '/app/node_modules/.cache'`

Reason

We set user at end
All instructions executed by root but then we switched to regular user

moving it

FROM
RUN addgroup.....
USER app
WORKDIR /app
:
:
:

Rebuild & Start new Container.

App Started but cant see in browser because host not mapped to Container.

Problem

Need to specify npm Start all the time.
docker run react-app npm start

Solution.

We can use Command instruction.

At end of dockerfile.

CMD npm start

Rebuild & start

docker run react-app.

Great!

What is difference between CMD & RUN

Both are used to execute Commands.

CMD	RUN
Run time instruction.	Build time instruction.
while starting Container.	while building image.

CMD has 2 forms.

1] Shell form
CMD npm start
[Executes inside seperate shell]

2] Exec form.
CMD ["npm", "start"]
[Executes directly, no need seperate shell process]

So exec form is recommended.

Another instruction.

ENTRYPOINT.

Similar to CMD
2 forms
1] Shell form
2] Exec form.

Difference between CMD & ENTRYPOINT

We can override CMD Command when starting Container.

docker run react-app
echo hello
→ This overrides.

But
we cannot easily override ENTRYPOINT Command.

docker run react-app
--entrypoint echo hello

CMD or ENTRYPOINT?

Personal preference.

Speeding up builds

Now — builds are layers.

Image — Image is a collection of layers.

Layers — FROM node:14.16.0-alpine3.13
↳ Node image will be in a layer.
RUN addgroup ....
↳ another layer.

How to see layers — docker history react-app

→ check created by & size
npm install - 178MB

What can be done — Docker has optimization mechanism.

→ docker checks instruction in dockerfile & sees if it is changed

Rebuild. Build from Cache.

COPY . .

This Command docker has to look at files to see if anything changed

Seperate installion from Copying.

→ First Copy package-lock.json & package.json.

COPY package*.json
RUN Npm install
COPY . .

Now builds are faster.

Dockerfile

Stable instruction

Changing instruction

Removing Images

How to remove images

docker images

↳ Shows list of images
- Some are not having Repository name, or tag, they are called dangling images

Use docker prune command.

Nothing got deleted because there are containers running an older react app image.

docker ps -a

lot of Container comes up which are in stopped state

Remove Containers

docker Container prune
-> all stopped Containers deleted.

now run
- docker image prune
all images removed.

Remove a non prune Container

docker image rm react-ap

Tagging images.

Don't use 'latest' tag in production.

Repository	Tag	ImageID
react-app	latest	f4b1daa
Ubuntu	latest	4dd97ce
alpine	latest	28fbe2

docker build -t react-app:1 .

react-app	1	f4b1daa

How to remove tag

docker image remove react-app:1

Adding a tag after creating an image.

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

Sharing images

hub.docker.com.

1st step

Give tag
raghavan1812/react-app

docker image tag
bob raghavan1812/
react-app:1

Image created

docker login
Username:
Password:

docker push raghavan1812
/react-app:2

1st time

takes time.

Now change in Readme file

Build & push as version 3

Push this time was quick.

Save & load Images

without docker hub

Save image

```
docker image save
-o react-app.tar
   react-app:3
```

Image saved in folder. The folder has several layers. Each layer represents either app... etc.

Load image

```
docker image load
-i react-app.tar
```

Image loaded