

- 3 main elements:

- 1° Dockerfile
- 2° image
- 3° container

- VM vs. Container

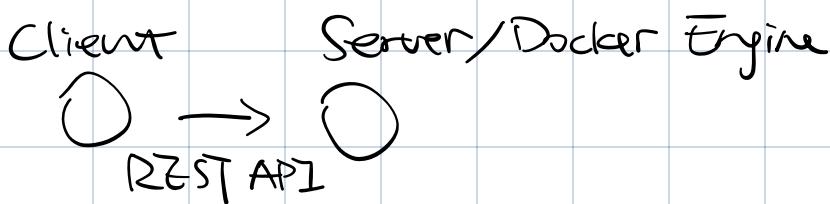
- VM cons:

- 1° each VM needs a full-blown OS
- 2° slow to start
- 3° resource intensive (RAM, CPU...)

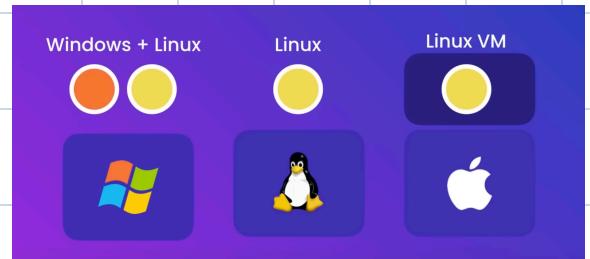
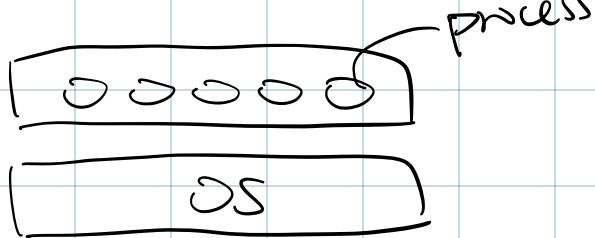
- Containers pros:

- 1° all running multiple Apps in isolation
- 2° lightweight
- 3° use OS of the host
- 4° quick to start
- 5° need less hardware resource

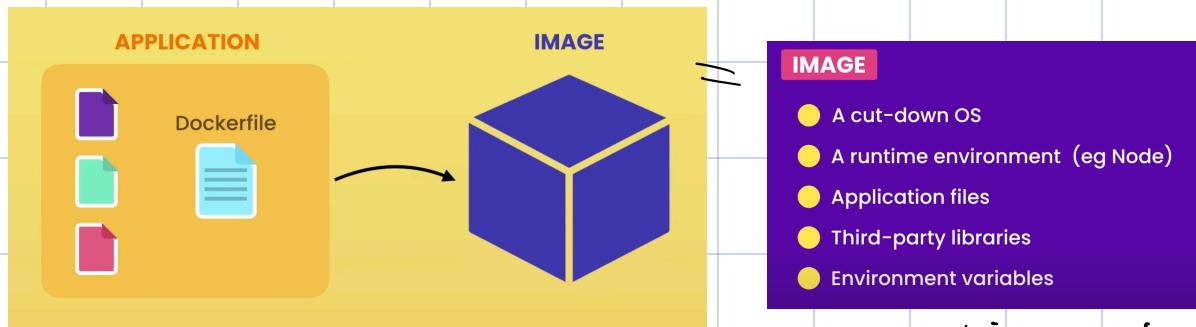
- Docker Architecture



Containers



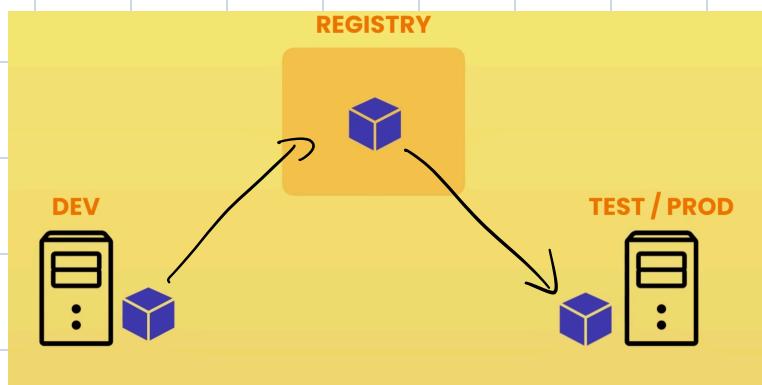
Development workflow



↓
load image into container

run docker inside a container

everything need to compile

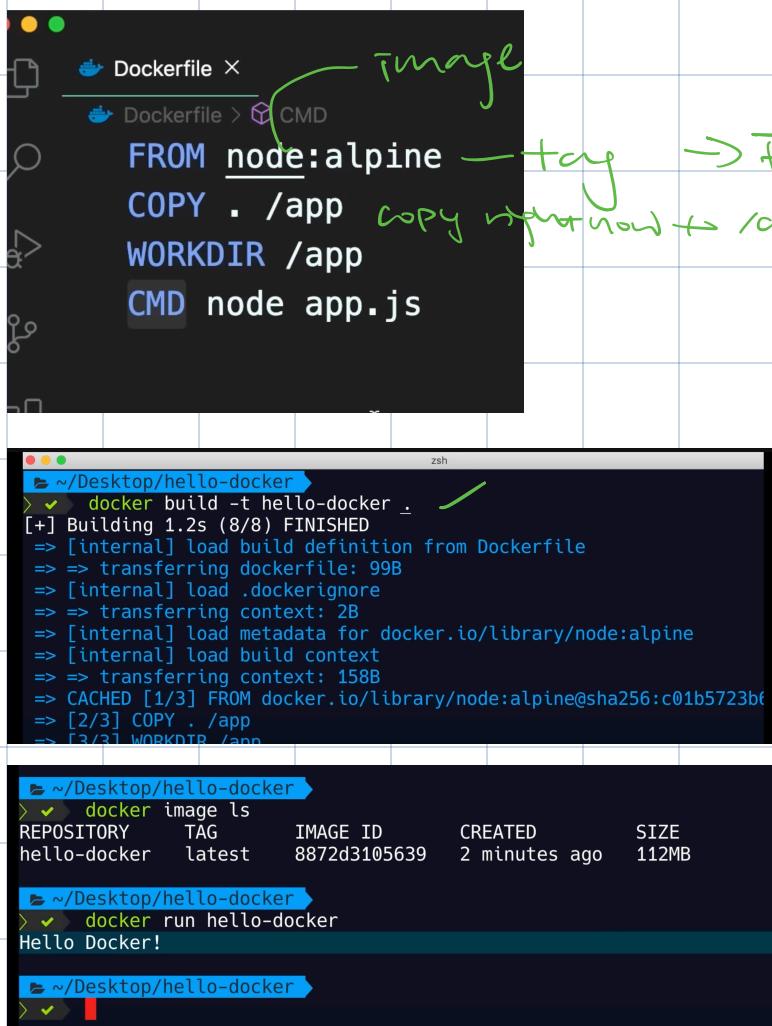


Demo (Docker in Action)

INSTRUCTIONS

- Start with an OS
- Install Node
- Copy app files
- Run node app.js

- Dockerfile



The image shows a terminal window with a Dockerfile and its build process. The Dockerfile content is:

```
FROM node:alpine
COPY . /app
WORKDIR /app
CMD node app.js
```

Annotations in green highlight the 'FROM' command as 'Image' and the 'tag' as 'tag'. A green arrow points from 'FROM' to 'FROM image:tag'. Another green arrow points from 'COPY' to 'copy instruction to /app'.

The terminal output shows the build process:

```
zsh
~/Desktop/hello-docker> docker build -t hello-docker .
[+] Building 1.2s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 99B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:alpine
=> [internal] load build context
=> => transferring context: 158B
=> CACHED [1/3] FROM docker.io/library/node:alpine@sha256:c01b5723bf
=> [2/3] COPY . /app
=> [3/3] WORKDIR /app
```

Then, the image is listed:

```
~/Desktop/hello-docker> docker image ls
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-docker    latest   8872d3105639  2 minutes ago  112MB
```

Finally, the container is run:

```
~/Desktop/hello-docker> docker run hello-docker
Hello Docker!
```

- Linux-Distribution

- Ubuntu