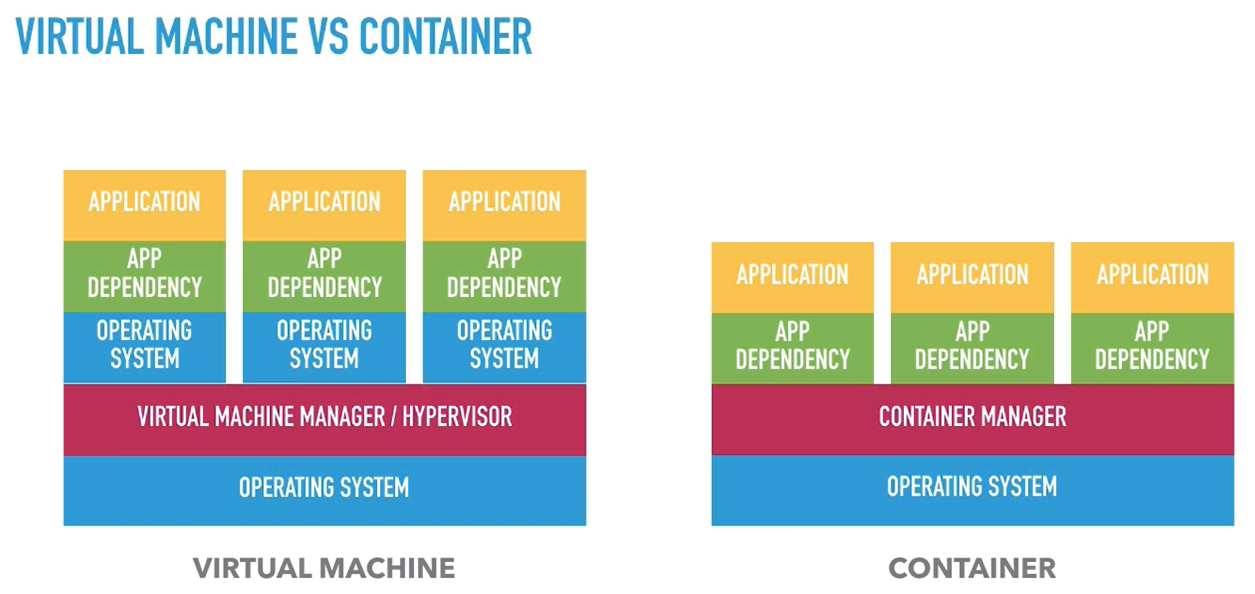
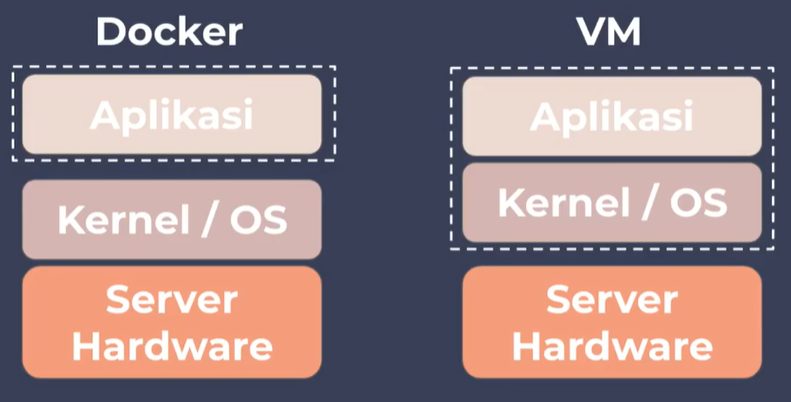
**Docker Documentation**

Docker adalah carauntuk package, deploy, distribusi app secara konsisten di environment manapun. Aplikasi dipackage dengan semua dependensi dan konfigurasi. Artifak portable, mudah dishare. Development dan deployment jadi mudah.

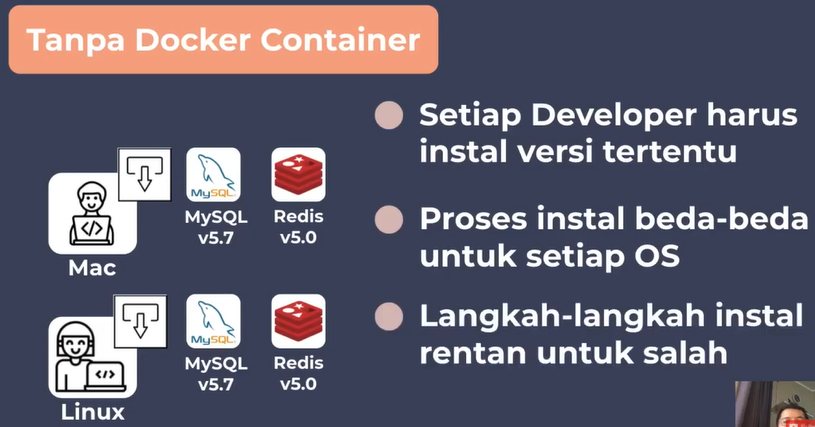
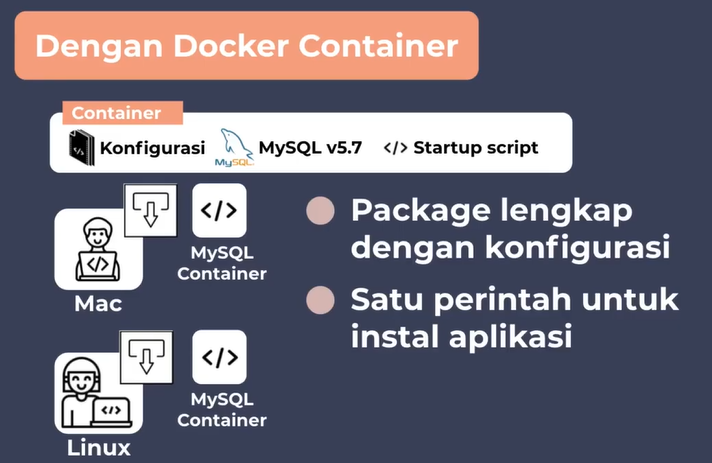
**Virtual Machine vs Docker**



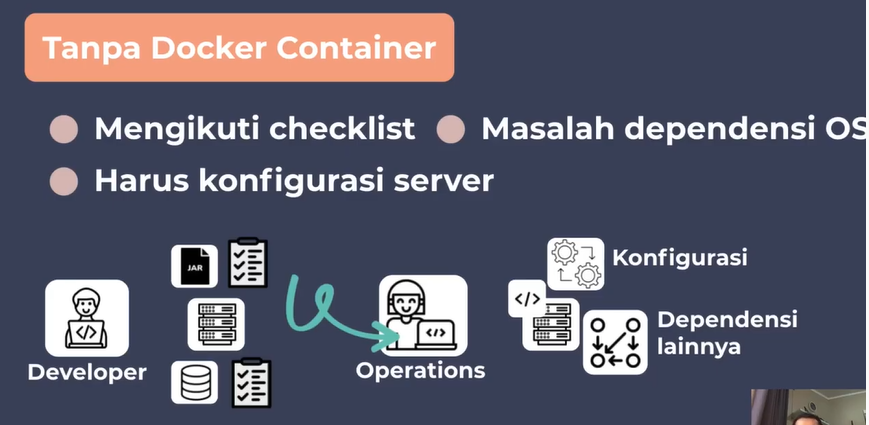


Docker is faster, lighter, not needing installing new OS like in VM.

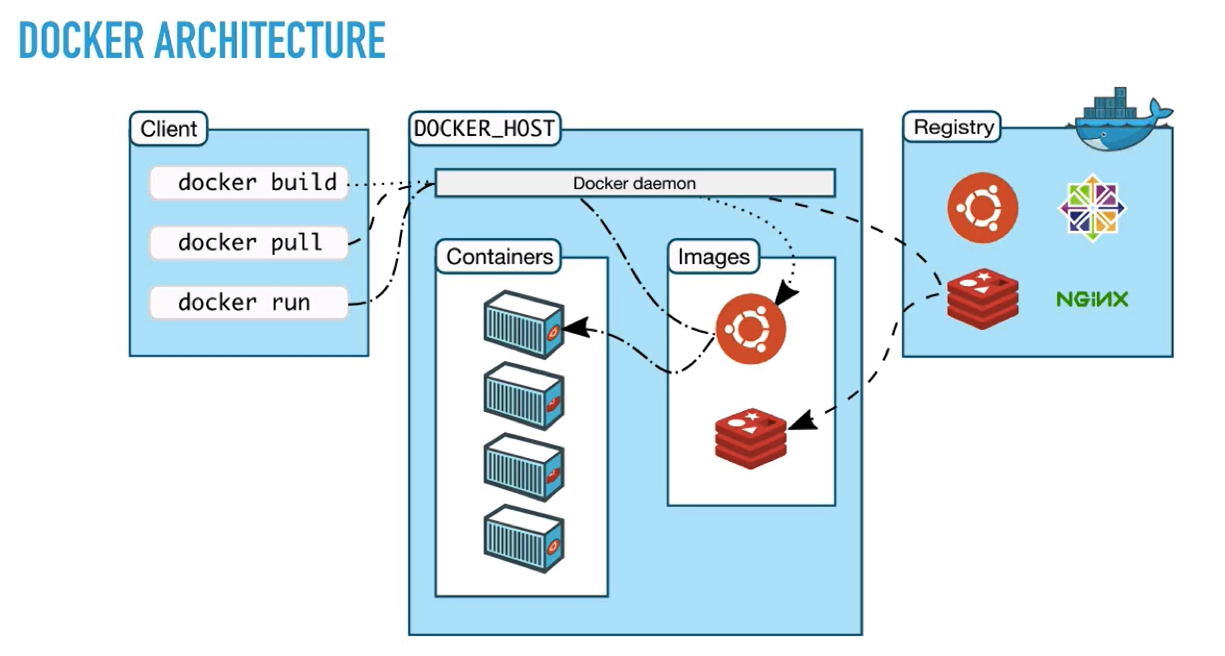
**Develop aplikasi tanpa dan dengan docker**

**Deploy aplikasi tanpa dan dengan docker**

**Docker Architechture**



Docker server akan diinstal di server production kita, tapi pada saat development docker server ada di local kita juga

Registry = tempat untuk menyimpan docker image

Image = hasil distribution file/package/bundle dari aplikasi kita, udah siap dijalankan

Container = instantiasi dari image



Base layer adalah OS linux, biasanya alpine karena yang paling ringan sehingga file docker menjadi kecil

**Docker command**

General command:

docker info

docker version

Melihat list images:

docker images

Download images: docker pull imageName:tag

docker pull mongo:4.1

Menghapus images: docker image rm imageName:tag 🡪 pastikan tidak sedang dipakai container

docker image rm mongo:4.1

Melihat list container:

docker container ls –a 🡪 untuk semua yang running dan tidak running

docker container ls 🡪 untuk semua yang running

Membuat container: docker container create --name namaContainer -p hostPort:containerPort imageName:tag

docker container create –name mongoserver1 8080:27017 mongo:4.1

Menjalankan container: docker container start containerName

docker container start mongoserver1

Mematikan container: docker container stop containerName

docker container stop mongoserver1

Menghapus container: docker container rm containerName 🡪 harus mati dulu

docker container rm mongoserver1

docker pull (untuk download dan lihat docker image)

docker run --name containerName (untuk menjalankan container)

docker ps (untuk melihat list container yang sedang berjalan)

docker run --options (untuk menjalankan container dengan opsi)

docker stop (untuk menghentikan container)

docker start (untuk menjalankan container)

docker port dan port binding (untuk mengakses container)

docker logs (untuk melihat logs yang dihasilkan container)

docker exec -it containerName (untuk masuk ke dalam container, --it = interactive)

flag pada command:

-d = detach

-p = konfigurasi port

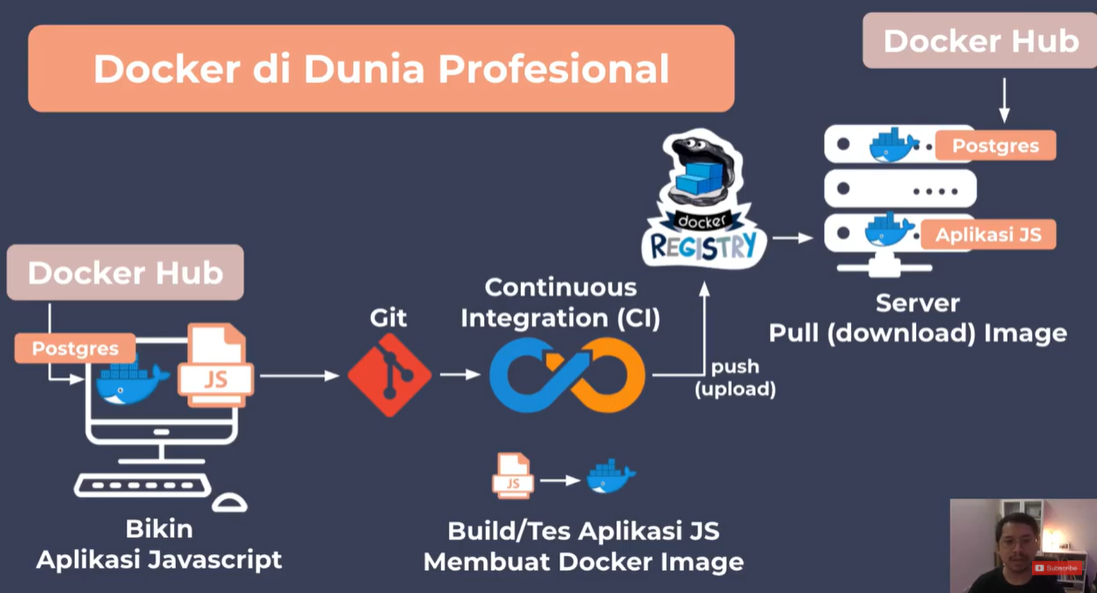
-a = all

--name = beri nama container

-f = untuk ngetail/ngikutin logs baru secara realtime

-it = interactive terminal

**Menggunakan Docker di sebuah project professional**



Gunakan Postgre image yg dipull dari dockerhub di pc kita saat development

Buat project javascript, push ke git, test oleh QA, ubah menjadi docker image, push ke docker registry

DevOps akan menggunakan Postgre image yg dipull dari dockerhub dan project javascript image kita saat deploy

**Membuat projek web dengan docker**

Aplikasu terdiri atas backend (golang), frontend (VueJS), dan database postgres. Semua aplikasi menggunakan docker image.

1. Membuat docker container untuk postgres.

Buka cmd, masuk ke direktori projek kita:

cd C:\Users\ASUS\Desktop\github\belajar-docker-pemula

Buat container postgres:

docker run -d \ 🡪 -d agar container jalan di background

-p 5432:5432 \ 🡪 port default container postgres

--name todo-postgres \ 🡪 nama container

-e POSTGRES\_USER=postgres \ 🡪 -e untuk input env variable, bisa dilihat apa saja di dockerhub

-e POSTGRES\_PASSWORD=rahasia \ 🡪 password

-e POSTGRES\_DB=belajar 🡪 nama database di postgres

-v %cd%/postgres/init.sql:/docker-entrypoint-initdb.d/init.sql \ 🡪 -v untuk mount file ke container

postgres 🡪 nama imagenya

Cek postgres container:

docker exec -it todo-postgres psql -U postgres -W belajar

isi password: rahasia

\dt 🡪 lihat database

SELECT \* FROM todo; 🡪 melihat isi table todo

Exit 🡪 keluar dr command postgres

1. Membuat docker file untuk frontend.

Buka cmd, masuk ke direktori projek kita:

cd C:\Users\ASUS\Desktop\github\belajar-docker-pemula\frontend

isi docker image frontend:

FROM node:16-alpine 🡪 install node.js di image

ENV VUE\_APP\_API\_URL=http://localhost:8080 🡪 set env variable di image

RUN mkdir /app 🡪 membuat directory baru /app di image

COPY . /app 🡪 copy semua file di pc ke image di /app

RUN yarn --cwd /app install 🡪 install dependensi di image di /app

CMD ["yarn", "--cwd" ,"/app", "serve"] 🡪 run command frontend di image di /app

Di terminal ketik:

docker build -t todo-frontend:v1 . 🡪 -t = tag, lalu namaImage, . = direktori dimana file berada

Jalankan container berisi image frontend:

docker run -d -p 8080:8080 todo-frontend:v1 🡪 -d = menjalankan di background

melihat log untuk mengecek error:

docker logs containerId

1. Membuat docker file untuk backend.

Buka cmd, masuk ke direktori projek kita:

cd C:\Users\ASUS\Desktop\github\belajar-docker-pemula\backend

isi docker image frontend:

FROM golang:1.16-alpine 🡪 install golag environment di image

WORKDIR /app 🡪 set direktori untuk command setelahnya dari direktori /app

COPY . . 🡪 copy semua file di pc/host ke direktori di image (sudah diset ke /app)

RUN go build -o todo-api 🡪 run go build di image, menjadi binary dengan nama todo-api

EXPOSE 8080 🡪 dokumentasi menunjukkan aplikasi menerima request di PORT ini

CMD ./todo-api 🡪 run binary yg sudah dibuild di image

Jalankan container berisi image backend:

docker build -t todo-backend:v1 . 🡪 -t = tag, lalu namaImage, . = direktori dimana file berada

Jalankan container berisi image backend:

docker run -d -p 8080:8080 todo-backend:v1

1. Menggunakan docker network.

buat docker network:

docker network create todo 🡪 semua container yg konek disini bisa saling connect port

jalankan container postgres pada network todo:

docker run -d \

-p 5432:5432 \

--name todo-postgres \

-e POSTGRES\_USER=postgres \

-e POSTGRES\_PASSWORD=rahasia \

-e POSTGRES\_DB=belajar \

-v %cd%/postgres/init.sql:/docker-entrypoint-initdb.d/init.sql \

--network todo \ 🡪 nama network yang dipakai

postgres

Jalankan container backend pada network todo:

docker run -d \

-p 8080:8080 \

--name todo-backend \

-e DB\_USER=postgres \

-e DB\_PASSWORD=rahasia \

-e DB\_HOST=todo-postgres \ 🡪 di network yg sama, nama container bisa jadi nama domain

-e DB\_PORT=5432\

-e DB\_DATABASE=belajar \

-v %cd%/postgres/init.sql:/docker-entrypoint-initdb.d/init.sql \

--network todo \ 🡪 nama network yang dipakai

postgres

Jalankan container frontend pada host:

docker run -d \

-p 8081:8080 \

--name todo-frontend \

todo-frontend:v1

buka di localhost:8081

1. Menggunakan docker compose, otomatis akan membuat docker network untuk container2 di dalamnya

buat file docker compose:

version: '3' 🡪 versi docker-compose, isi 1/2/3

services:

todo-postgres: 🡪 --name

image: postgres

ports:

- 5432:5432 🡪 -p

environment:

- POSTGRES\_USER=postgres 🡪 -e

- POSTGRES\_PASSWORD=rahasia 🡪 -e

- POSTGRES\_DB=belajar 🡪 -e

volumes:

- ./postgres/init.sql:/docker-entrypoint-initdb.d/init.sql 🡪 -v

todo-backend:

build: ./backend 🡪 cd backend, docker build -t todo-backend .

ports:

- 8080:8080 🡪 -p

environment:

- DB\_USER=postgres 🡪 -e

- DB\_PASSWORD=rahasia 🡪 -e

- DB\_HOST=todo-postgres 🡪 -e

- DB\_PORT=5432 🡪 -e

- DB\_DATABASE=belajar 🡪 -e

depends\_on:

- todo-postgres 🡪 jalankan container backend setelah container postgres berjalan

todo-frontend:

build: ./frontend 🡪 cd frontend, docker build -t todo-frontend .

ports:

- 8081:8080 🡪 -p

Jalankan docker compose file:

docker compose up

ctrl+c untuk exit dan stop semua container