Belajar Dasar Docker Dalam 30 Menit

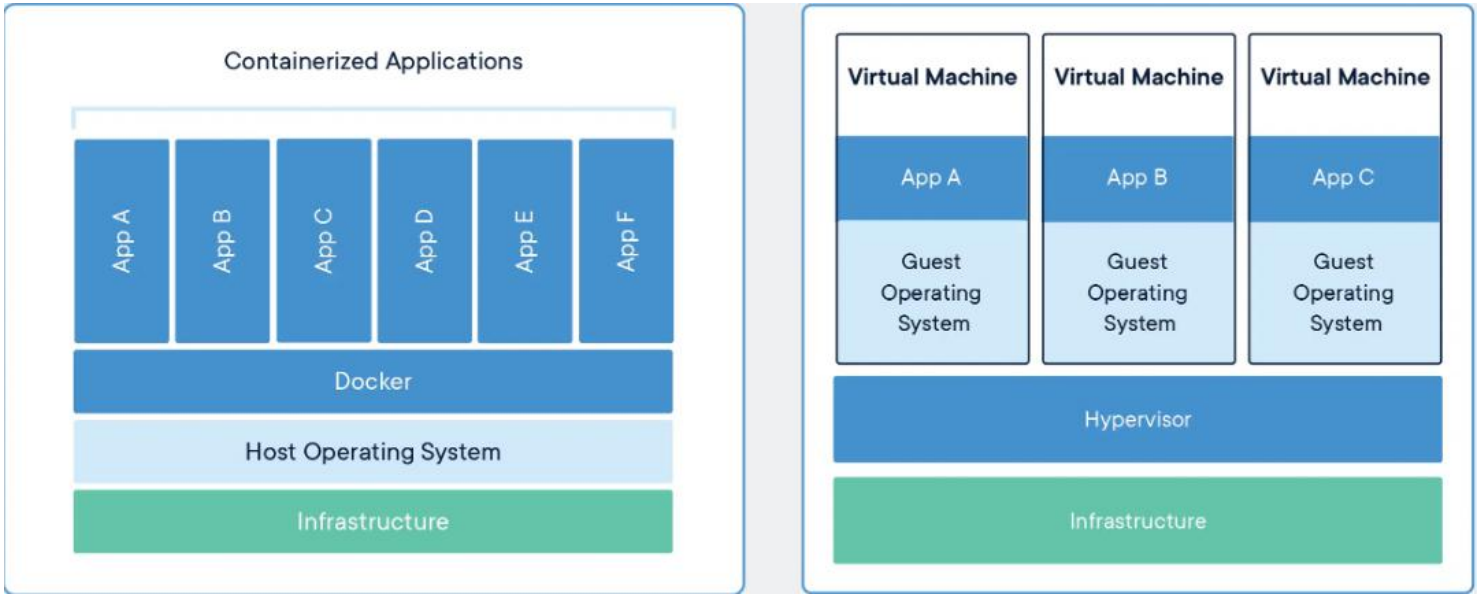
Muhammad Syukur Abadi

Kenapa *Container*?

Ada beberapa permasalahan yang diselesaikan oleh *container*

- Menyeragamkan Dependensi
  Seringkali, ketika menjalankan program yang sama, namun di komputer yang berbeda, salah satu komputer mengalami masalah karena dependensi yang tidak sesuai.
- Mengisolasi Aplikasi
  Aplikasi yang dibungkus oleh *container* akan terisolasi, sehingga bisa menghindari konflik jika terdapat *package* yang sama pada suatu aplikasi di dalam komputer yang sama.

Container Vs Virtual Machine



Container Vs Virtual Machine

Container

- Mengabstraksi di tingkat *application*
- Lebih cepat di *boot* (*spin*)
- Ukuran lebih kecil

Virtual Machine

- Mengabstraksi di tingkat *hardware*
- Lebih lambat di *boot*
- Ukuran lebih besar

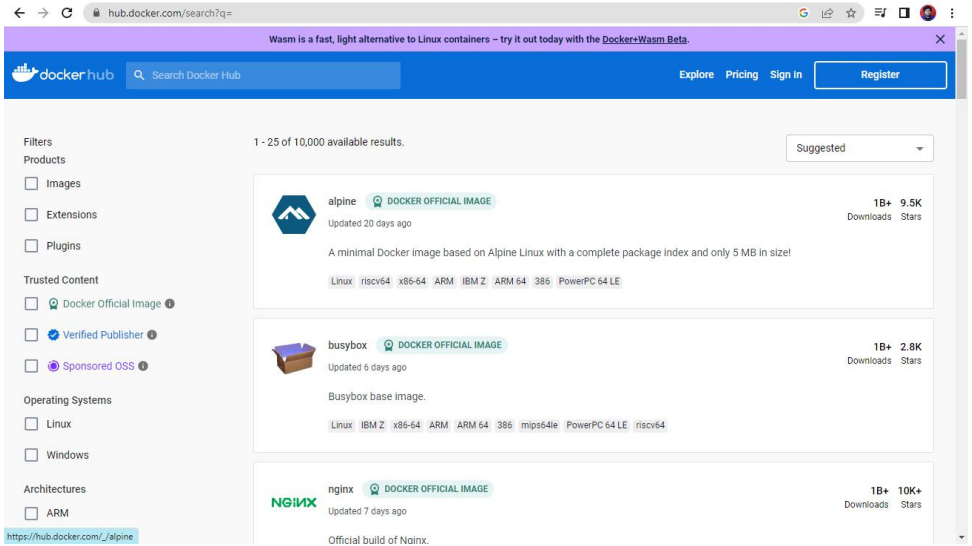
Docker

Docker merupakan salah satu jenis *container* yang saat ini banyak digunakan. Docker digunakan untuk membungkus dan men-*deploy* aplikasi/dependensi dalam bentuk *docker image*. *Docker image* bisa diunduh atau diunggah di *docker hub*

Untuk meng-*install* Docker di windows, teman-teman bisa mengikuti panduannya di https://docs.docker.com/desktop/windows/wsl/

Docker Hub

Docker hub merupakan repositori yang digunkan untuk melakukan *pull* dan *push Docker Image*.



Perintah-Perintah Dasar Docker

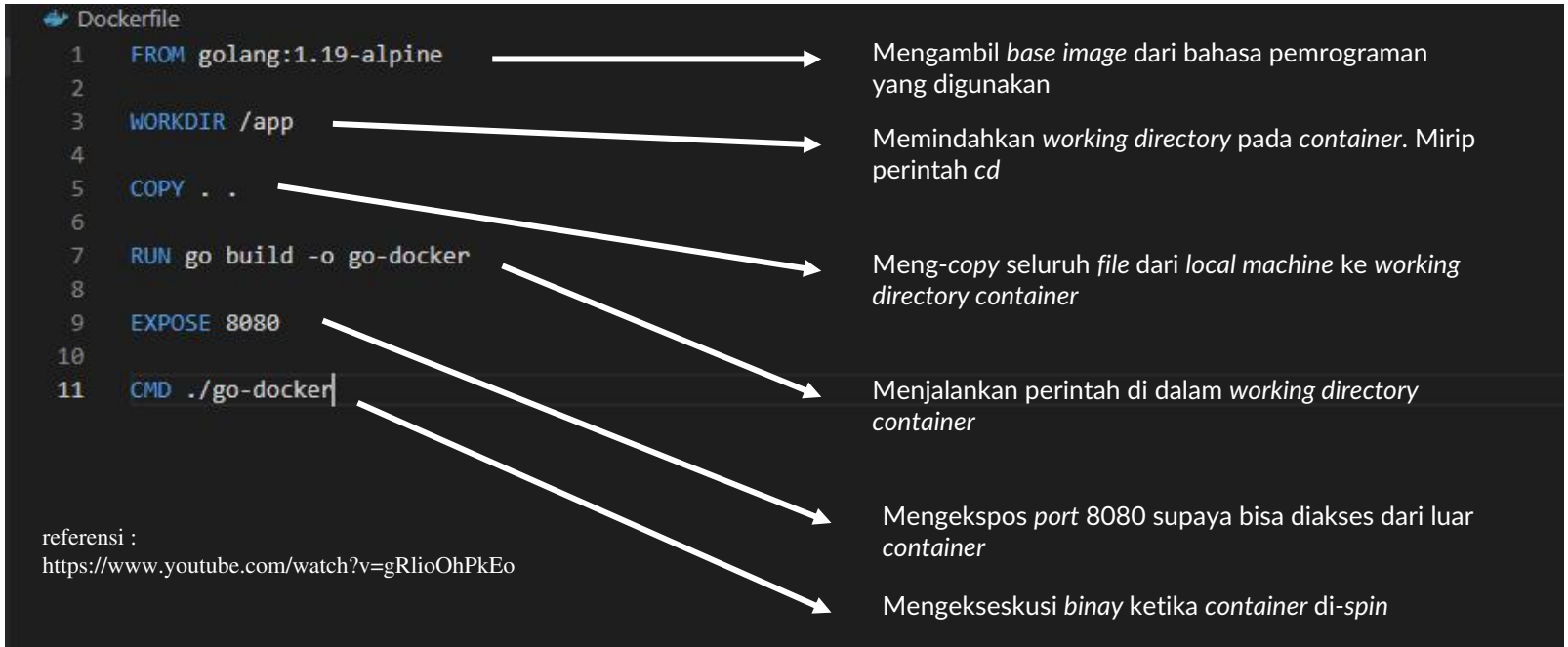
Setiap perintah *docker* diawali dengan *docker*. Berikut perintah dasar yang biasanya digunakan beserta fungsinya

- **pull** : melakukan *pull* terhadap *image* dari *Docker Hub*
- **run** : menjalankan *image* dan mengubahnya menjadi *container*. Jika belum ada *image* di *local*, *docker* akan melakukan *pull* terlebih dahulu
- **ps** : melihat *container* yang sedang berjalan
- **exec** : menjalankan perintah di dalam *container* yang sedang berjalan
- **stop** : menghentikan *container* yang sedang berjalan
- **start** : menjalankan *container* yang sudah berhenti
- **rm** : menghapus *container*

Men-*Dockerize* Aplikasi

Kita juga dapat membungkus aplikasi kita menjadi sebuah *image* yang bisa di *push* ke *Docker Hub* dan digunakan oleh developer lain.

Men-*Dockerize* Aplikasi



referensi :
https://www.youtube.com/watch?v=gRlioOhPkEo

Mengintegrasikan *Container*

Supaya satu *container* bisa terhubung dengan *container* lain, kita perlu menghubungkan *container-container* tersebut dengan *network*.

Mengintegrasikan *Container*

```
PS C:\Users\HP\Documents\go-nsq> docker network create go-docker
```

Membuat *network* dengan nama *go-docker*

```
PS C:\Users\HP\Documents\go-nsq> docker run -p 8080:8080 --name go-docker --network go-docker ./app-backend
```

Menyambungkan *container go-docker* dengan *network go-docker*

```
PS C:\Users\HP\Documents\go-nsq> docker run -p 27017:27017 --name mongodb --network go-docker mongo:6.0.2
```

Menyambungkan *container mongo* dengan *network go-docker*

Menjalankan Banyak *Container* Sekaligus

Apakah kita bisa menjalankan banyak *container* sekaligus dengan satu *command*?

Tentu bisa!. Dengan menggunakan *docker compose*, kita bisa mengotomasi pembuatan banyak *container* sekaligus. Yang kita butuhkan hanyalah *file docker compose* dan perintah *docker compose*.

Menjalankan Banyak *Container* Sekaligus

File docker compose adalah file dengan ekstensi *yml* atau *yaml*. Berikut contoh *file docker compose*

docker-compose.yml

You, 3 weeks ago | 2 authors (You and others)

```yaml
version: '3'
services:
  minio:
    container_name: minio
    image: quay.io/minio/minio
    ports:
      - "9001:9001"
      - "9000:9000"
    environment:
      - MINIO_ROOT_USER=${MINIO_ACCESS_KEY_ID}
      - MINIO_ROOT_PASSWORD=${MINIO_SECRET_ACCESS_KEY}
    command: server /data --console-address ":9001"
  nsqlookupd:        You, 3 months ago • rename prefalsif
    image: nsqio/nsq
    container_name: nsqlookupd
    command: /nsqlookupd
    ports:
      - "4160:4160"
      - "4161:4161"
```

Menjalankan Banyak *Container* Sekaligus

Kembali ke pembahasan *network*, jika kita kita punya *docker compose* seperti ini tanpa atribut *network*, ketika kita melakukan *up* pada *docker compose* tersebut, *docker* akan membuat *network* secara otomatis

docker-compose.yml

```yaml
...
version: "3.8"

services:
  mongo:
    image: mongo:6.0.2
    container_name: mongodb
    ports:
      - 27017:27017

  app-backend:
    build: ./app-backend
    ports:
      - 8080:8080
    depends_on:
      - mongo

```

network akan langsung dibuat ketika kita meng-*up*

```
PS C:\Users\HP\Documents\code\go-docker-demo> docker compose up
[+] Running 3/3
 - Network go-docker-demo_default          Created
 - Container mongodb                       Created
 - Container go-docker-demo-app-backend-1  Created
Attaching to go-docker-demo-app-backend-1, mongodb
```

Terima Kasih

sykrabadi

Muhammad Syukur Abadi

sykrabadi