Politeknik Negeri Lhokseumawe
Jurusan Teknologi Informasi dan Komputer
Teknologi Rekayasa Komputer Jaringan

TRKJ-6435 — Cloud Computing

Dosen Pengampu: Muhammad Davi, S.Kom., M.Cs.
Gedung Terpadu Jurusan Teknologi Informasi dan Komputer
muhammad.davi@pnl.ac.id
2023-2024 Genap

Daftar Isi

Lab 1 — Docker

1 Lab 1.1: Docker Image "hello-world"

Pertama periksa dulu versi docker yang digunakan dengan perintah berikut:

```
$ docker version
```

Tuliskan versi docker disini: _____
Dari output diatas, Docker diimplementasikan dengan bahasa pemrograman: _____
Anda ingin menjalankan sebuah Image yang bernama hello-world. Image ini tersedia di Docker Hub. Namun sebelum menjalankan instruksi tersebut, lebih dahulu periksa apakah ada container dan image yang aktif.

```
$ docker images
```

Jika ternyata ada images yang sudah terdaftar, maka hapus lebih dahulu dengan instruksi dibawah ini.

```
// hentikan proses docker yang masih aktif
$ docker stop $(docker ps -a -q)


// hapus semua kontainer yang ada
$ docker rm $(docker ps -a -q)

// hapus semua images
$ docker rmi $(docker images -a -q)


// atau hapus semua image yang ada dan kontainer yg stop
$ docker system prune -a
```

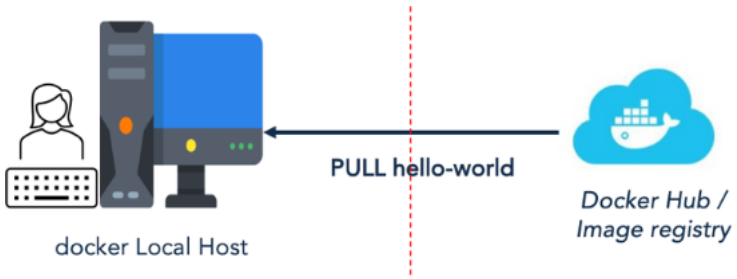
Opsi -a menampilkan semua elemen, sedangkan -q menampilkan ID nya saja. Ketiga instruksi ini dilakukan hanya jika ada proses tersebut yang aktif, urutannya penting, mulai dari proses, kontainer dan image. Selanjutnya lakukan instruksi berikut:

```
$ docker run hello-world
```

Pesan pertama yang muncul adalah:

```
Unable to find image 'hello-world:latest' locally
```

Docker berusaha untuk menjalankan image yang bernama hello-world, tapi tidak ditemukan di localhost. Oleh karena itu docker secara otomatis mencari hello-world di Docker Hub. Image tersebut ditemukan disana, dan selanjutnya docker mengambil (pull) image tersebut ke localhost, kemudian menjalankannya setelah selesai menarik image tersebut.



Bila Anda menjalankan untuk kedua kali, maka reaksi dari localhost lebih cepat karena image hello-world sudah berada di localhost, sehingga tidak lagi perlu mengunduh (pull) dari docker hub.

```
$ docker images
$ docker run hello-world
```

Tuliskan Image ID disini: _____
Kalau punya _____ Image ID-nya: _____

2 Lab 1.2: Shell

Anda ingin mengaktifkan Image seperti Lab 1, tapi kali ini Anda ingin masuk ke kontainer tersebut untuk bekerja diatas kontainer seperti bekerja diatas sebuah Sistem Operasi yang dengan menggunakan Shell.
Sebelumnya pilih dulu Sistem Operasi yang akan digunakan, umumnya versi Linux. Ada 3 versi yang populer, yaitu RedHat, Ubuntu, Suse dan turunan lainnya seperti CentOS, Alpine, Linux Mint, Manjaro dan lainnya. Masing-masing punya kelebihan atau konsentrasi di segment tertentu.
Kita akan coba Linux yang relatif paling ringan, yaitu Alpine. Cari dulu di Docker Hub, apakah image alpine tersedia.

```
$ docker search alpine
```

Ada banyak image alpine terdaftar disitu, lengkap dengan utilitas lainnya. Yang kita butuhkan adalah alpine.

```
$ docker pull alpine
$ docker images
```

Perhatikan bahwa Tag adalah versi, dan bila tidak diberikan, nilai default adalah "latest". Sehingga instruksi berikut ekuivalen dengan sebelumnya.

```
$ docker pull alpine:latest
```

Jalankan sebagai kontainer dan bekerja dengan shell (bash):

```
$ docker run -it alpine
```

Opsi -it artinya jalankan sebagai interactive terminal (sebagai terminal). Jalankan beberapa instruksi shell untuk mencoba:

```
# id
# date
# hostname
# netstat -r
# exit
```

Anda bisa melihat proses aktif di kontainer dengan ps (proses status)

```
$ docker ps
```

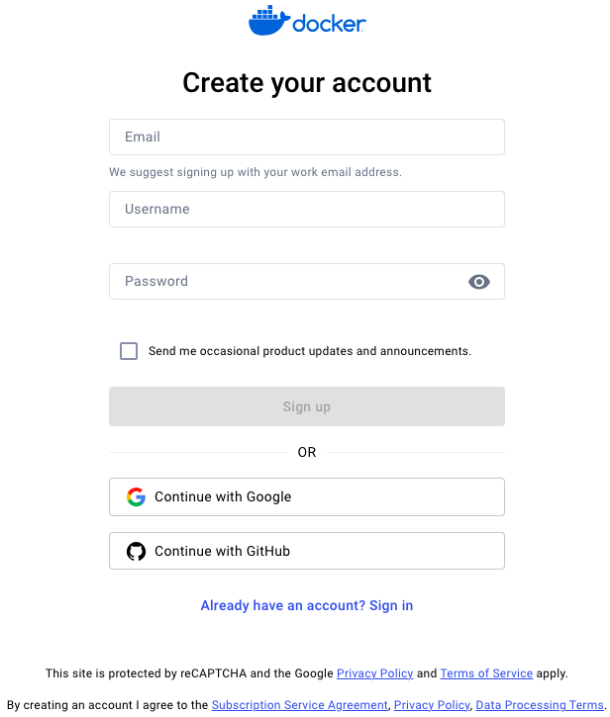
Tapi tidak ada kontainer yang aktif. Untuk melihat jejak proses sebelumnya, gunakan opsi -a (all).

```
$ docker ps -a
```

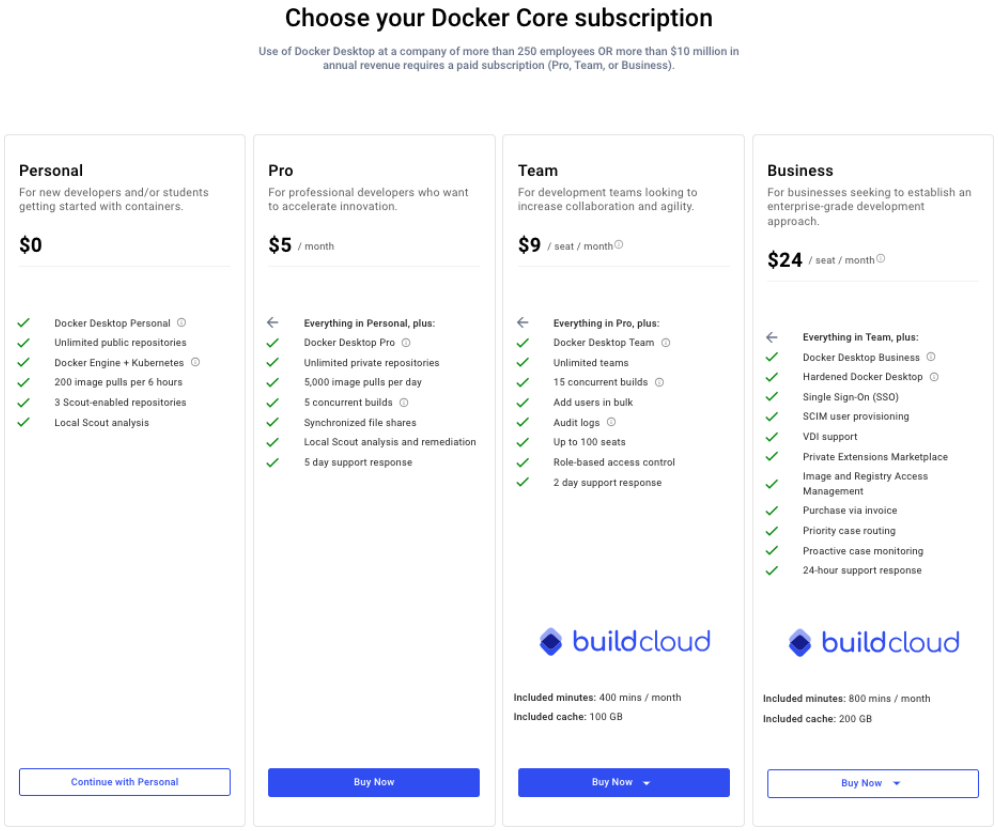
Banyak proses yang sudah berjalan dan selesai dengan status Exited.

3 Lab 1.3: Membuat Akun di Docker Hub

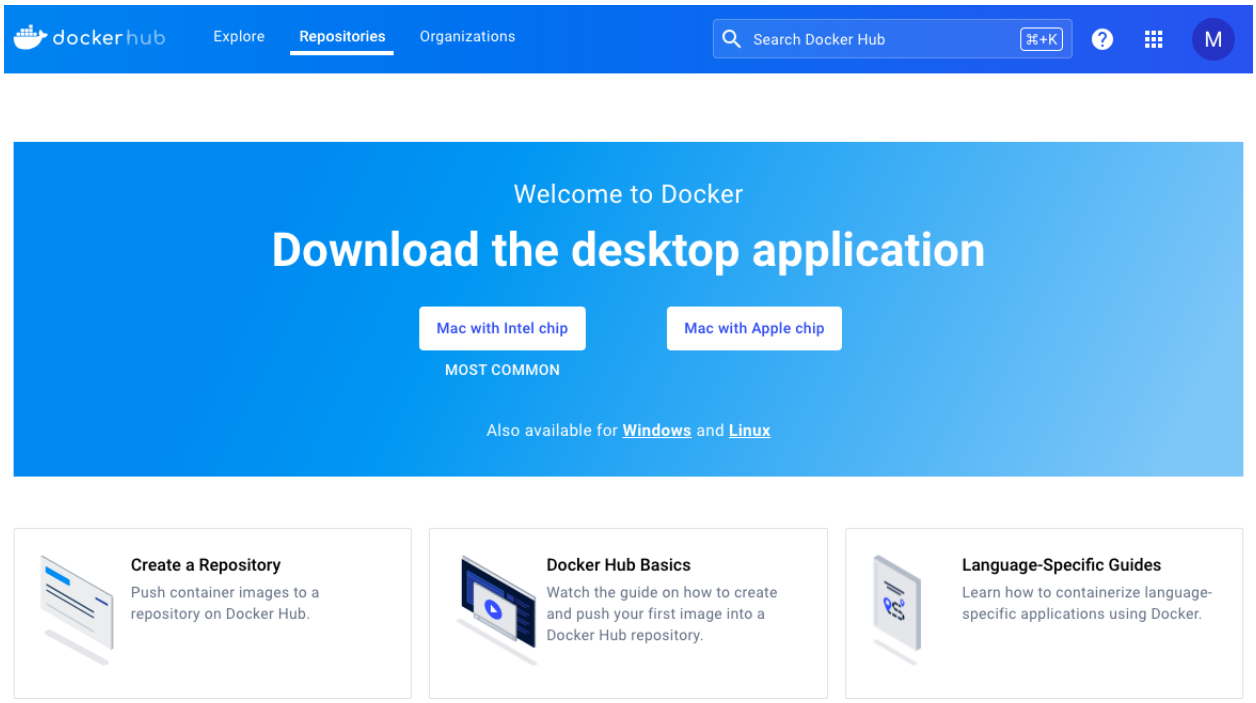
Persiapkan email yang valid yang dapat segera diakses. Validasi token akan diberikan melalui akun email tersebut. Arahkan browser ke: `https://hub.docker.com`. Pilih Sign up untuk membuat akun baru.



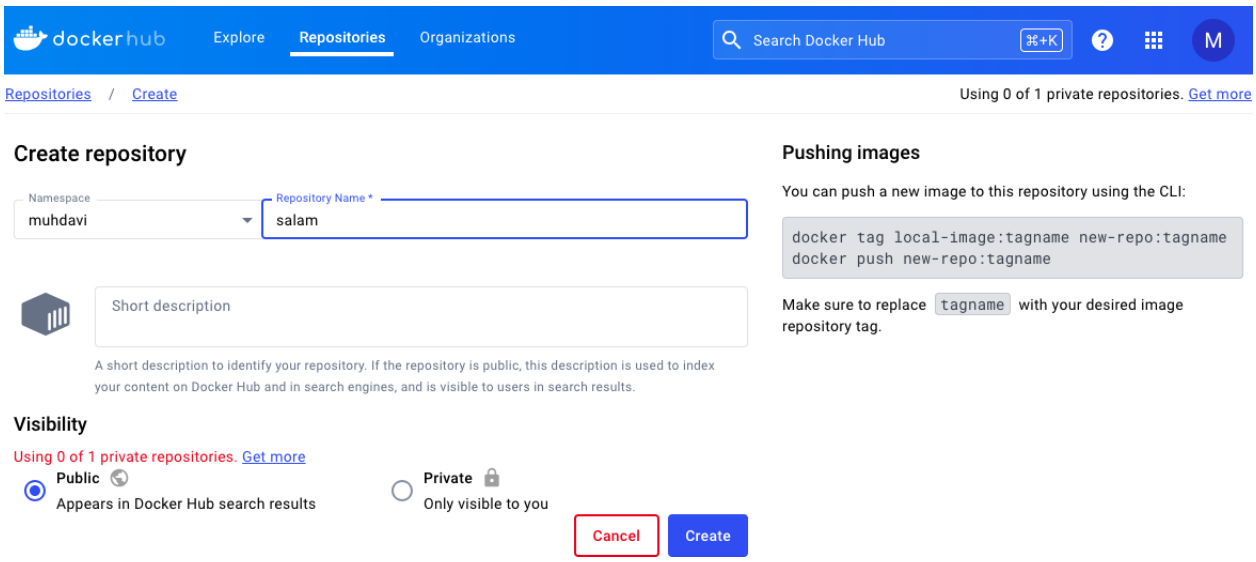
Tulis identitas, alamat email, dan password untuk login ke docker hub. Ikuti instruksi untuk validasi dan verifikasi (biasanya melalui pertanyaan pada sebuah gambar yang diberikan). Pilih Skema yang Free ($0), **"Continue with Personal"**.



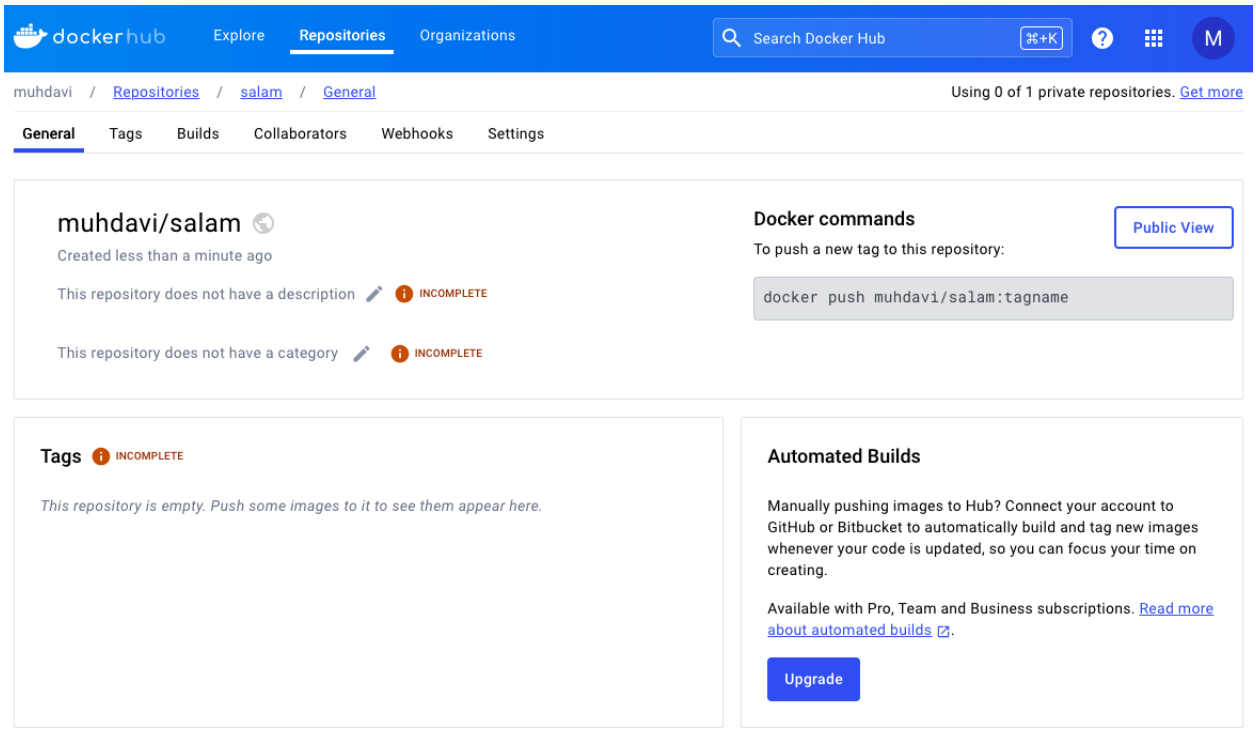
Abaikan Download the desktop application. Pilih **Create a Repository**.



Pada Menu **Create repositori**, tulis nama repository - pada contoh adalah "salam". Berikan deskripisi terkait. Visibility adalah "public", yang mana repositori ini dapat diakses oleh siapa saja di Internet.



Hasil Pembuatan repositori. Disebelah kanan adalah informasi untuk melakukan "push" yaitu menuyimpan docker image di docker hub. Instruksi tersebut dilakukan dari komputer client setelah mebuat image docker baru.



4 Lab 1.4: Membuat Image baru dan Menyimpan di Docker Hub

1. Buar folder baru untuk image baru

```
$ mkdir salam
$ cd salam
```

2. Buat sebuah file Golang sebagai berikut dengan nama salam.go:

```
package main
import "fmt"
func main() {
    fmt.Println("Salam dari Faiza Kelompok 6")
}
```

   Ganti Faiza dengan nama anda sendiri dan kelompok dengan kelompok Anda.

3. Buat Dockerfile seperti berikut:

```
FROM golang:latest
# Faiza - faiza@gmail.com
WORKDIR /app
COPY . .
RUN go build salam.go
CMD ["/app/salam"]
```

4. Bentuk Image baru dengan docker build
   Perhatikan, ganti nama muhdavi dengan nama akun Anda di Docker Hub. Jangan lupa ada titik pada akhir instruksi, yang berarti mengolah atau membaca Dockerfile dari Folder Aktual (current directory).

```
$ docker build -t muhdavi/salam .
```

5. Periksa hasilnya dengan:

```
$ docker images
```

6. Simpan image tersebut di Docker Hub. Sebelumnya login terlebih dahulu ke repositori.

```
$ docker login -u muhdavi
password:

$ docker push muhdavi/salam
```

7. Minta partner ada untuk mengambil dan menjalankan program tersebut dengan perintah berikut:

```
$ docker pull muhdavi/salam
```

Atau langsung dengan run (otomatis menjalankan pull)

```
$ docker run muhdavi/salam
```

Perhatikan: ganti nama muhdavi dengan user name Anda di docker hub.

Lab 2 — Docker

5 Docker Image "node_project"

Buat direktori tempat kerja Anda di direktori lokal dengan **mkdir** dan kemudian pindah ke direktori tersebut dengan **cd** (change directory).

```
$ mkdir node_project
$ cd node_project
```

Cek instalasi npm dengan perintah berikut:

```
$ npm -v
```

Bila belum ada, download dan install melalui link https://nodejs.org/en/download/. Selesai instalasi npm, lakukan `npm init` dah hasilnya sebuah file teks.

```
$ npm init
```

Periksa file yang terbentuk, dengan:

```
$ cat package.json
```

Hasilnya akan mirip tampilan disini:

```
{
  "name": "node_project",
  "version": "1.0.0",
  "description": "Cloud Computing",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Muhammad Davi",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2"
  }
}
```

File `package.json` akan ditambahkan modul "express" dengan perintah berikut:

```
$ npm install express
```

Hasil dari penambahan modul di file `package.json` sebagai berikut:

```
  "author": "Muhammad Davi",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2"
  }
}
```

Selanjutnya buat sebuah file dengan nama index.js dan isi dengan kode berikut ini:

```
const express = require("express")
const app = express()
const port = 7000

app.get("/", (req, resp) => {
    // identitas
```

```
    resp.send("Nama saya Faiza dari Kelompok 6");
})

app.listen(port, ()=> console.log(`listening on port ${port}`))
```

Perhatikan bahwa tanda petik di string listening adalah tanda kutip terbalik (`), bukan tandan petik ('). Tanda ini akan melakukan substitusi $port menjadi 7000.

```
// Terminal 1
$ node index.js
Listening on port 7000
```

Kemudian aktifkan terminal 2 dan jalankan curl:

```
// Terminal 2
$ curl localhost:7000
Nama saya Faiza dari Kelompok 6
```

Dan coba buka di web browser dengan url http://localhost:7000 apakah menghasilkan output yang sama?
Selanjutnya coba buat Dockerfile menggunakan kode berikut ini:

```
FROM node:latest
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
EXPOSE 7000
CMD ["node", "index.js"]
```

Buat docker image dengan perintah `docker build .` ← jangan lupa tanda titik di-belakang.
Kemudian periksa apakah image tersebut sudah ada?
Tuliskan Image ID disini: ————————
Kalau punya ———————— Image ID-nya: ————————
Perbaiki image tersebut dari <none> dengan nama image menggunakan opsi -t dan cantumkan nomor versi 1.0 (node_project:1.0), jangan menggunakan latest.

```
$ docker build -t muhdavi/node_project:1.0 .
```

Lihat hasilnya dengan `docker images` dan periksa status dengan `docker ps`.
Jalankan docker images tersebut sebagai proses daemon dengan menggunakan opsi -d (detach). Berikan nama kontainer dengan menggunakan opsi –name dan lakukan port mapping dengan opsi -p agar apalikasi dapat diakses. Coba akses menggunakan web browser.

```
$ docker run -d --name node_project -p 80:7000 muhdavi/node_project:1.0
```

6 Simpan Image di Dockerhub

Siapkan image yang sudah dibuat untuk dikirim ke docker hub. Langkah pertama adalah menggunakan login ke dockerhub lengkap dengan password. Langkah kedua adalah melakukan docker push.

7 Jalankan docker image teman Anda dari docker hub

Untuk lab ini dibutuhkan nama akun teman Anda dan nama repositori beserta nomor versi nya.

```
$ docker pull nama_akun/nama_repositori:nomor_versi
```

Atau dengan `docker run` untuk menjalankan secara otomatis.

```
$ docker run nama_akun/nama_repositori:nomor_versi
```

Perhatikan: kemungkinan Anda harus membuat port-mapper dan sebaiknya juga memberi nama pada kontainer tersebut.

Lab 3 — Docker

8 Docker Volume

Docker volume dibutuhkan oleh kontainer karena kontainer tidak dapat menyimpan data secara persisten. Docker volume menawarkan fisikal disk yang ada di Host sebagai media penyimpanan untuk kontainer. Dengan demikian jika kontainer selesai bekerja dan hilang dari Host, maka data tersebut tetap ada.
Keuntungan lain dari docker volume adalah menghemat media penyimpanan yang mana Multiple Container dapat mengakses folder dan data yang sama. Hal ini dapat direalisasikan dengan membuat "named volume" artinya volume yang secara eksplisit diberi nama.
Terdapat beberapa instruksi shell primitif yang akan digunakan.

- **touch file.txt**
  Membuat file kosong, hanya menciptakan nama file.
- **echo "string" > file.txt**
  Mengisi konten file.txt, bila sebelumnya sudah ada isi, maka isi tersebut akan dihapus. Bila belum pernah ada, file.txt akan diciptakan.
- **echo "string juga" » file.txt**
  Mengisi konten file.txt pada akhir file, menyambung dengan isi sebelumnya.

Beberapa perintah untuk melihat environment kerja

```
$ docker info
$ docker images
$ docker ps
$ docker volume ls
```

Agar bisa dimulai dengan data kosong, lakukan hal berikut. Pastikan tidak ada container yang "running" saat Lab dimulai.

```
$ docker system prune -a
$ docker volume prune
```

Docker Image "alpine" adalah sistem operasi Linux yang sangat ringan, tidak seberat Ubuntu atau lainnya. Alpine digunakan sebagai basis OS untuk Lab berikut. Opsi -i menandakan bahwa input akan diberikan dari keyboard secara interaktif dan opsi -t menandakan modus terminal. Opsi -i -t atau -it mengaktifkan OS sebagai terminal shell. Instruksi shell dapat diberikan langsung. Opsi -v digunakan sebagai opsi untuk volume, -v nama folder, folder tersebut digunakan sebagai volume.

```
$ mkdir -p /home/muhdavi/localdata
$ docker run -it -v /home/muhdavi/localdata/:/mydata --name k1 alpine
$ cd /mydata
$ touch f1 f2 f3
$ ls -al
$ ctrl+d (^d)
$ docker ps
$ ls /home/muhdavi/localdata
```

Kembali aktifkan kontainer k1 dengan interactive mode.

```
$ docker start -i k1
$ ls mydata
$ docker volume ls
```

Volume anonymous tidak menggunakan nama untuk mengakses volume tersebut. Docker otomatis memberi hash-number sebagai nama folder. Sebelum memasuki Lab, ada baiknya hapus semua kontainer dan image agar tidak tercampur image, kontainer dan obyek lain hasil dari lab sebelumnya.

```
$ docker system prune -a
$ docker volume prune
```

Intruksi untuk menghasilkan volume anonymous adalah sebagai berikut:

```
$ docker run -it -v /myfolder --name k2 alpine
$ cd myfolder
$ touch f4 f5 f6
```

Aktifkan Terminal 2, lakukan instruksi berikut:

```
$ docker ps
$ docker volume ls
```

Nama folder local yang diberikan berupa random hash number. Docker malakukan random hash number ini agar tidak terjadi duplikasi dengan nama folder yang sudah diciptakan sebelumnya. Berikut ini inspeksi kontainer k2 untuk melihat informasi details, dimana folder tersebut berada.

```
$ docker inspect k2
$ docker volume ls
$ docker volume inspect
```

Asynchronous volume ini dapat diakses bersama dengan kontainer lain. Buka dari terminal 2, buat kontainer baru dengan nama k3 menggunakan volume dari kontainer k2 tersebut.

```
$ docker run -it --volumes-from k2 --name k3 alpine
$ cd myfolder
$ echo "salam dari kontainer 3" > pesan
```

Dari terminal 1 lakukan hal berikut.

```
$ cat pesan
$ echo "salam kembali untuk kontainer 3 dari kontainer 2" >> pesan
```

Kembali ke Terminal 2 untuk melihat hasilnya. Berdasarkan hasil yang ditampilkan dua kontainer ini melakukan shared file.

```
$ cat pesan
```

Kali ini volume diberikan nama, sehingga kontainer dapat menggunakan nama tersebut sebagai referensi, tidak lagi menggunakan random-hash. Pemberian nama akan memudahkan multiple kontainer untuk mengkases volume tersebut.
Sebelum memasuki lab, sebaiknya dihapus semua kontainer dan image agar tidak tercampur image, kontainer dan obyek lain hasil dari lab sebelumnya.

```
$ docker system prune -a
$ docker volume prune
```

Lakukan instruksi berikut untuk menghasilkan volume bernama. Namakan secara logical "ourdata" karena volume tersebut akan diakses oleh banyak kontainer.

```
$ docker volume create ourdata
$ docker volume ls
$ docker run -it -v ourdata:/projectdata --name k4 alpine
$ ls
$ cd projectdata
$ echo "Ini pesan dari k4 alpine" > pesan
$ ls
```

Dari terminal 2

```
$ docker run -it -v ourdata:/projectdata --name k5 alpine
$ cd projectdata
$ echo "Ini pesan dari k5 alpine" >> pesan
$ cat pesan
```

Dari terminal 3 lihat status kontainer tersebut

```
$ docker ps
$ docker volume ls
$ docker images
```

9 Multi kontainer dengan docker-compose

Docker compose merupakan alat (tools) untuk menjalankan kontainer docker secara bersamaan. File compose digunakan untuk melakukan konfigurasi layanan aplikasi. Docker compose sering digunakan untuk development dan testing workflows karena seluruh kontainer dapat berjalan menggunakan single host secara default. Kali ini akan membuat multi services menggunakan docker compose dengan wordpress dan database.

1. Membuat multi kontainer secara otomatis dengan menggunakan tools docker compose dengan image wordpress dan mariadb. Buatkan direktori untuk menyimpan data mariadb dan wordpress persistent dengan nama $HOME/wp-compose/database dan $HOME/wp-compose/html dan masuk ke direktori $HOME/wp-compose.

```
$ mkdir -p $HOME/wp-compose/{database, html}
$ cd $HOME/wp-compose
```

2. Verifikasi versi docker compose yang digunakan saat ini

```
$ docker-compose --version
```

3. Membuat multi kontainer aplikasi Docker menggunakan docker-compose. Buatkan file docker-compose.yml dengan menggunakan teks editor apapun yang Anda sukai, file ini digunakan untuk melakukan konfigurasi seluruh layanan aplikasi (wordpress dan mariadb). Tambahkan baris code berikut ke dalam file docker-compose.yml.

```
$ vi docker-compose.yml
```

```
version: '3.3'

services:
  mariadb:
```

```
    image: mariadb
    volumes:
      - ./database:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - mariadb
    images: wordpress:latest
    ports:
      - "8001:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: mariadb:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - ./html:/var/www/html
```

Catatan
Service wordpress tidak akan dijalankan sebelum service mariadb berjalan (depends_on: mariadb). Image untuk mendefinisikan iamge yang akan digunakan pada kontainer. Ports untuk mendefinisikan port mana yang akan di expose sebagai host. Sama halnya dengan parameter -p pada perintah docker run. Setelah memiliki images yang dibutuhkan maka tidak perlu menambahkan images baru.

4. Menjalankan multi service kontainer dengan menggunakan docker-compose. Untuk menjalankannya masuk ke dalam direktori dimana lokasi dari docker-compose.yml berada dengan perintah `docker-compose up -d`.

```
$ docker-compose up -d
```

Untuk melihat log aplikasi, dapat menggunakan perintah `docker-compose logs`.

```
$ docker-compose logs -f wordpress
```

Untuk melihat informasi aplikasi yang dibuat dengan docker-compose, dapat menggunakan perintah `docker-compose ps`.

```
$ docker-compose ps
```

5. Setelah proses *docker-compose* berhasil maka, web dapat diakses melalui browser ke http://localhost:8001, sehingga muncul tampilan instalasi wordpress.
6. Untuk stop dan menghapus kontainer dari service saat ini, dapat menggunakan perintah `docker-compose down`.

```
$ docker-compose down
```

Catatan
Instalasi docker-compose di ubuntu

1. Melakukan instalasi docker-compose dengan mengunduh dan menyimpan file di **/usr/local/bin/docker-compose** dengan menjalankan perintah berikut:

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/v2.28.1/↩
    docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2. Ubah ijin akses agar file `/usr/local/bin/docker-compose` dapat di eksekusi dengan perintah berikut ini:

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

3. Verifikasi instalasi berhasil dengan menjalankan perintah berikut ini:

```
$ docker-compose --version
```