NAMA : FIDELIA PING
NIM : 245410012
KELAS : INFORMATIKA 1

MODUL 11
Simulasi Arsitektur Client-Server Menggunakan Docker di AWS EC2

DASAR TEORI

1. Arsitektur Client-Server

Arsitektur Client-Server adalah model pendistribusian aplikasi yang membagi tugas atau beban kerja antara penyedia sumber daya atau layanan, yang disebut sebagai **Server**, dan peminta layanan, yang disebut sebagai **Client**.

Dalam model ini, komunikasi biasanya mengikuti pola *request-response* (permintaan-tanggapan).

- **Client:** Perangkat atau program yang menginisiasi komunikasi dengan mengirimkan permintaan data atau layanan. Contohnya adalah web browser atau aplikasi mobile.
- **Server:** Perangkat atau program yang menunggu permintaan masuk, memproses data, dan mengirimkan kembali jawaban (*response*) ke client. Server harus memiliki sumber daya yang lebih kuat dan ketersediaan (availability) yang tinggi.

2. Amazon Web Services (AWS) dan EC2

Cloud Computing adalah penyediaan sumber daya IT (seperti server, penyimpanan, database) sesuai permintaan melalui internet dengan skema pembayaran *pay-as-you-go*.
Amazon Elastic Compute Cloud (Amazon EC2) adalah layanan web dari AWS yang menyediakan kapasitas komputasi yang aman dan dapat diubah ukurannya di cloud.

- **Virtualisasi:** EC2 memungkinkan pengguna menyewa komputer virtual (instance) untuk menjalankan aplikasi server tanpa perlu membeli perangkat keras fisik.
- **Skalabilitas:** Pengguna dapat menaikkan (scale up) atau menurunkan (scale down) kapasitas server sesuai kebutuhan trafik.
- **Peran dalam Praktikum:** Dalam simulasi ini, instance EC2 bertindak sebagai *host* fisik bagi server yang dapat diakses melalui alamat IP publik (Public IP) dari mana saja.

3. Docker dan Kontainerisasi

Docker adalah platform perangkat lunak terbuka yang digunakan untuk mengembangkan, mengirim, dan menjalankan aplikasi di dalam lingkungan terisolasi yang disebut **Container**.

Berbeda dengan Virtual Machine (VM) tradisional yang membutuhkan sistem operasi tamu (Guest OS) penuh untuk setiap aplikasi, Docker menggunakan kernel sistem operasi *host* secara bersama-sama namun tetap menjaga isolasi antar aplikasi. Ini membuat Docker jauh lebih ringan dan cepat.

Komponen utama Docker meliputi:

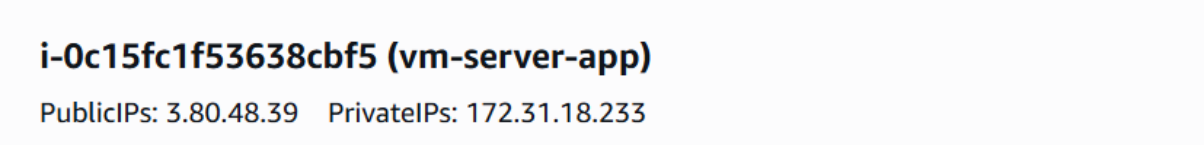
- **Docker Image:** Cetak biru (blueprint) atau template *read-only* yang berisi kode aplikasi, *library*, dan dependensi yang dibutuhkan agar aplikasi berjalan.
- **Docker Container:** Instance yang berjalan dari sebuah Docker Image. Container bersifat *portable*, artinya dapat berjalan dengan cara yang sama di laptop pengembang, server on-premise, maupun di cloud (AWS EC2).
- **Docker Hub:** Registry berbasis cloud untuk menyimpan dan berbagi Docker Image.

4. **Integrasi: Client-Server dengan Docker di AWS**
   Menggabungkan ketiga konsep di atas menciptakan infrastruktur modern yang efisien:
   1. Aplikasi server dikemas (packaged) menjadi **Docker Image** untuk memastikan konsistensi lingkungan (*environment consistency*).
   2. Image tersebut dijalankan sebagai **Container** di dalam **AWS EC2** (Cloud Server).
   3. Port pada container dipetakan (exposed) ke port publik EC2, sehingga **Client** dari jaringan eksternal dapat mengirim *request* ke aplikasi tersebut.

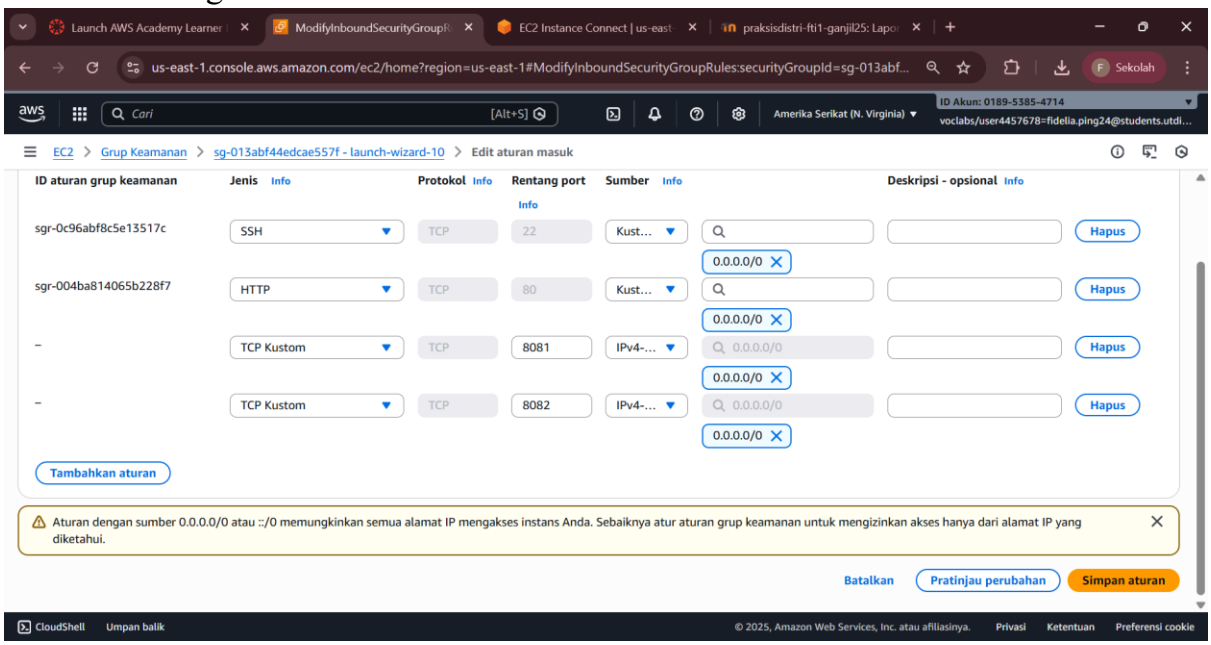
PRAKTIK

Hentikan dan hapus container web-app-ku yang sedang berjalan menggunakan perintah `sudo docker stop 374d22345d97 362fe90e9266` dan `sudo docker rm 374d22345d97 362fe90e9266`.

```
ubuntu@ip-172-31-18-233:~$ ^C
ubuntu@ip-172-31-18-233:~$ sudo docker rm 374d22345d97 362fe90e9266
374d22345d97
362fe90e9266
ubuntu@ip-172-31-18-233:~$
```



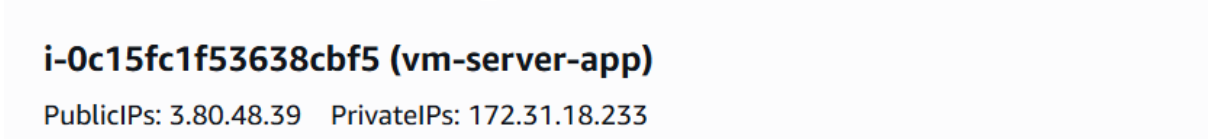
Memastikan port 8081 dan 8082 sudah dibuka pada Security Group (Inbound Rules) di AWS EC2 agar bisa diakses.



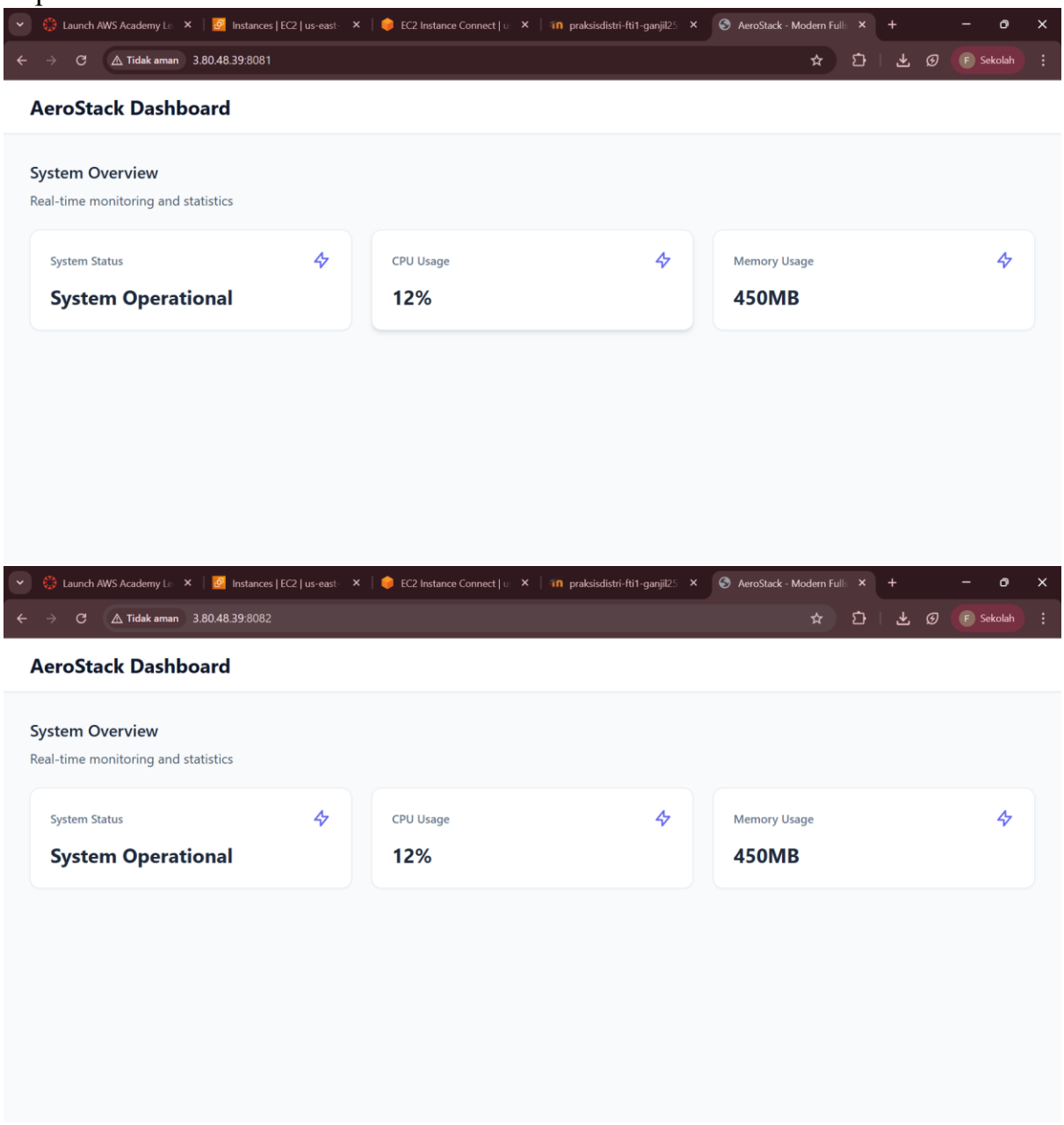
Jalankan dua container berbeda dari image yang sama (web-app-ku), namun menggunakan port yang berbeda pada VM 1 (port 8081 dan port 8082):

```
sudo docker run -d -p 8081:80 web-app-ku
sudo docker run -d -p 8082:80 web-app-ku
```

```
ubuntu@ip-172-31-18-233:~$ sudo docker run -d -p 8081:80 web-app-ku
045a06a80836e13197ead4801a8965205ebcf7454601caeb454816f93f3e58b5
ubuntu@ip-172-31-18-233:~$ sudo docker run -d -p 8082:80 web-app-ku
a5548cc026e2f99c68173c7c33785ad943743ec5f4cdd99d7f3fe2d3ba037e4b
ubuntu@ip-172-31-18-233:~$
```



Cek di browser dengan
http://3.80.48.39:8081/
http://3.80.48.39:8082/



KESIMPULAN

Jadi saya paham bagaimana cara hentikan dan hapus container web-app-ku. Kemudian bagaimana menambahkan lagi port yang baru 8081 dan 8082.