LAPORAN KOMPUTASI AWAN
TUGAS 2

DEPARTEMEN TEKNIK INFORMATIKA

Muhammad Wafi Zaki Hanif

5025221039

1. Pendahuluan

Proyek ini bertujuan untuk mendemonstrasikan evolusi pengelolaan infrastruktur server menggunakan teknologi kontainerisasi Docker dan orkestrasi menggunakan docker-compose. Latihan dimulai dari pemahaman dasar tentang web server statis, kemudian meningkat ke aspek keamanan (SSL/HTTPS), integrasi sistem manajemen konten (CMS) berbasis Microservices, hingga pembuatan aplikasi kustom (Custom Image).

Puncaknya, proyek ini mengembangkan simulasi sistem Smart Farming yang menerapkan prinsip-prinsip Cloud Computing modern, yaitu *Scalability*, *High Availability* dan *Load Balancing* untuk memastikan sistem tetap berjalan meskipun terjadi kegagalan pada salah satu komponen.

2. Penjelasan Arsitektur Sistem

Sistem ini menggunakan arsitektur **Microservices** yang berjalan di atas satu jaringan kontainer terisolasi. Sistem terbagi menjadi 4 lapisan utama:

a. Layer Gateway & Security (Load Balancer)
   i. Komponen: Nginx Container (loadbalancer)
   ii. Fungsi: Bertindak sebagai pintu gerbang utama. Ia menerima request dari User melalui protokol aman HTTPS (Port 443) menggunakan sertifikat SSL.
   iii. Logika: Menggunakan algoritma Round Robin untuk mendistribusikan trafik pengunjung secara bergantian ke 3 node dashboard di belakangnya. Ini menjamin beban server terbagi rata.

b. Layer Aplikasi
   i. Komponen: 3x Python Flask Containers (dashboard-node-1, 2, 3).
   ii. Fungsi: Menyajikan antarmuka visual (Frontend) kepada pengguna.
   iii. Sifat: Stateless (tidak menyimpan data di dalam dirinya sendiri). Jika satu container mati, container lain bisa menggantikan tanpa kehilangan data karena data tersimpan di database terpisah.

c. Layer IoT Simulation (Data Producer)
   i. Komponen: Python Script Container (sensor-worker).
   ii. Fungsi: Mensimulasikan perangkat keras sensor di lapangan. Container ini berjalan di background (tanpa UI), menghasilkan data dummy (suhu/kelembaban), dan mengirimkannya (WRITE) ke database secara terus-menerus.

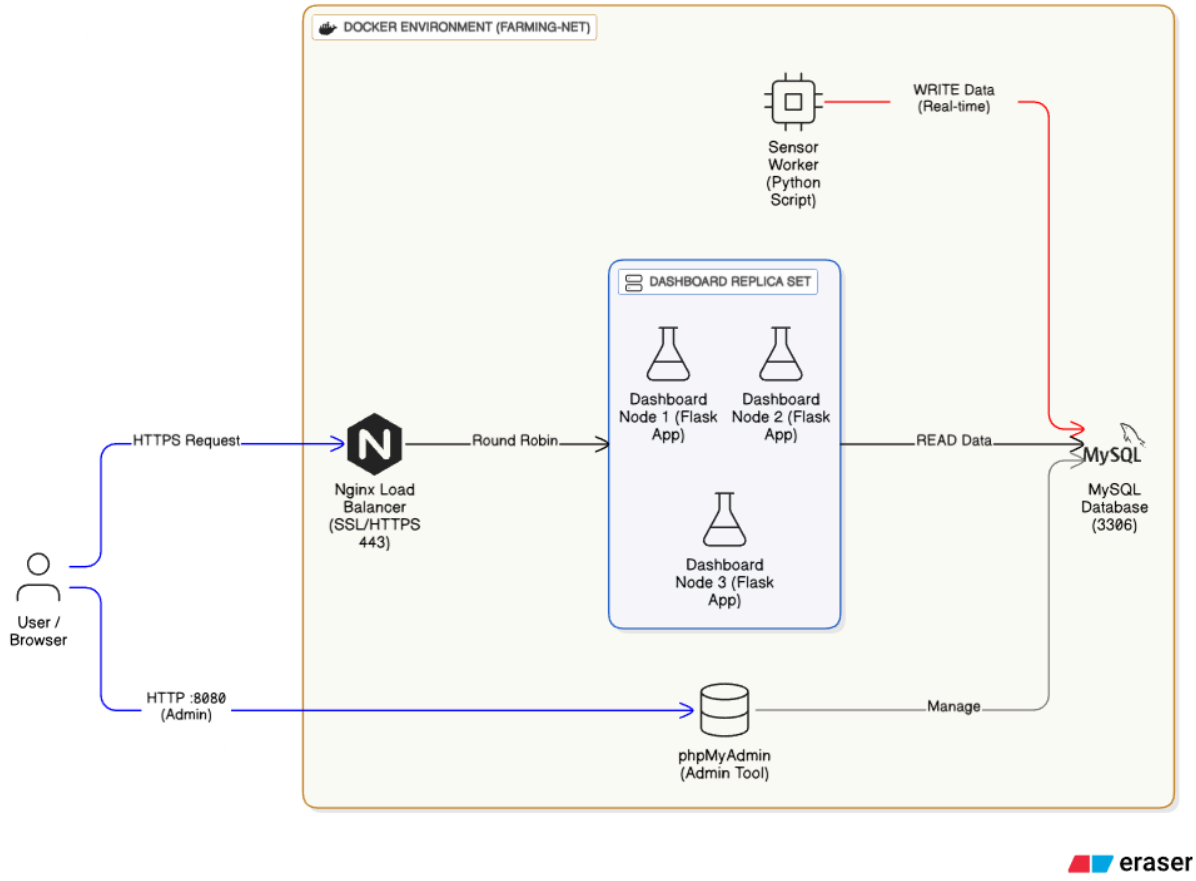
d. Layer Data & Management (Database)

i. Komponen: MySQL 5.7 (database) & phpMyAdmin (phpmyadmin).

ii. Fungsi:

1. MySQL: Pusat penyimpanan data persisten. Semua node dashboard (membaca) dan sensor (menulis) terhubung ke sini.
2. phpMyAdmin: Alat bantu admin untuk memantau data secara manual lewat Port 8080.

Diagram Arsitektur



3. Pembahasan Tugas

a. Pembahasan Case 1

Pada case ini, tujuan utamanya adalah memahami dasar web server Nginx dan konsep Volume Mounting pada Docker.

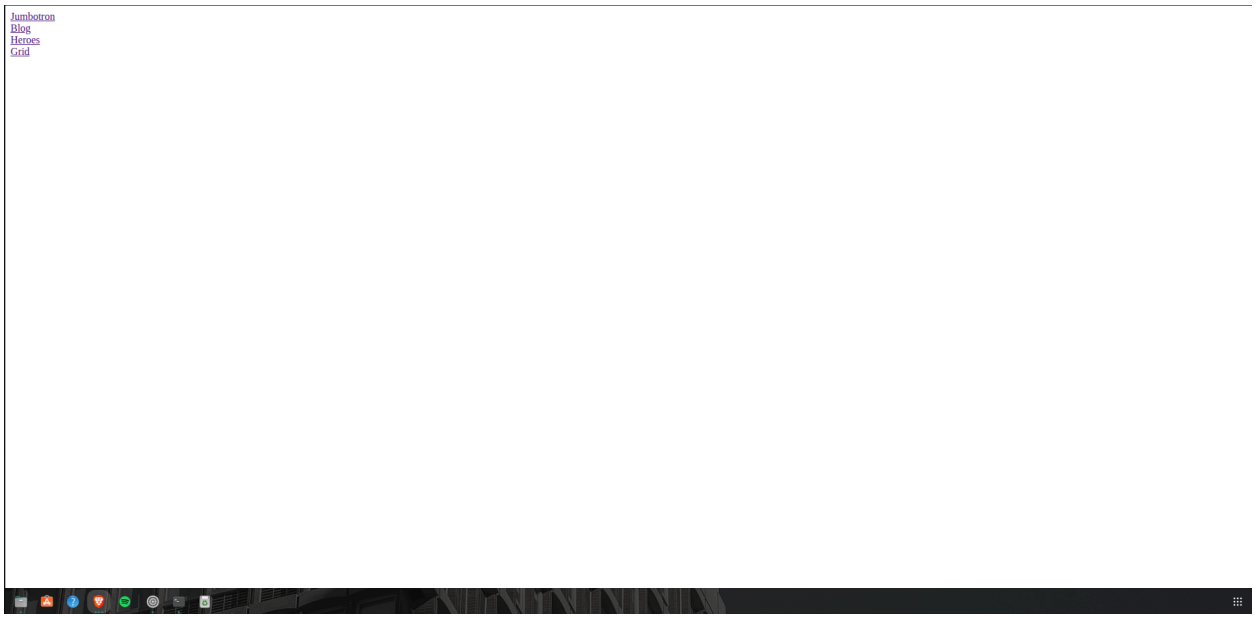
Sistem dikonfigurasi menggunakan docker-compose untuk menjalankan image Nginx. Fitur kunci yang dipelajari adalah penggunaan Docker Volume (-v). Dengan memetakan folder lokal komputer ke folder /var/www/html di dalam container, perubahan file HTML yang dilakukan di laptop (Host) dapat langsung terlihat di server (Container) secara real-time tanpa perlu melakukan rebuild image.

Cara Run: Menjalankan perintah “sudo docker compose up -d”

```
version: '3'

services:
  webserver:
    image: nginx:1.15.12-alpine
    restart: unless-stopped
    ports:
      - "9999:80"
    volumes:
      - ./html:/var/www/html
      - ./nginx-conf:/etc/nginx/conf.d
    networks:
      - app-network
networks:
  app-network:
    driver: bridge
```

Gambar 3.1: Konfigurasi Docker Compose Case 1



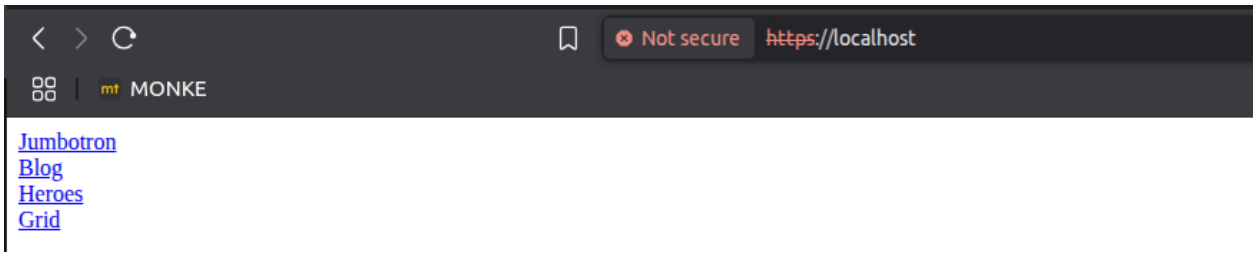
Gambar 3.2 : Tampilan Web Statis

b. Pembahasan Case 2

Case 2 berfokus pada aspek keamanan jaringan. Pada tahap ini, server Nginx ditingkatkan kemampuannya untuk menangani protokol HTTPS (Port 443) menggunakan sertifikat SSL/TLS (Self-Signed).

Selain itu, dipelajari juga konfigurasi Nginx (nginx.conf) untuk melakukan Auto-Redirect (HTTP 301). Mekanisme ini memaksa pengguna yang mengakses website melalui protokol HTTP biasa (Port 80) untuk dialihkan secara otomatis ke jalur aman HTTPS. Ini mensimulasikan standar keamanan web modern.

Cara Run: Menjalankan perintah “sudo docker compose up -d”



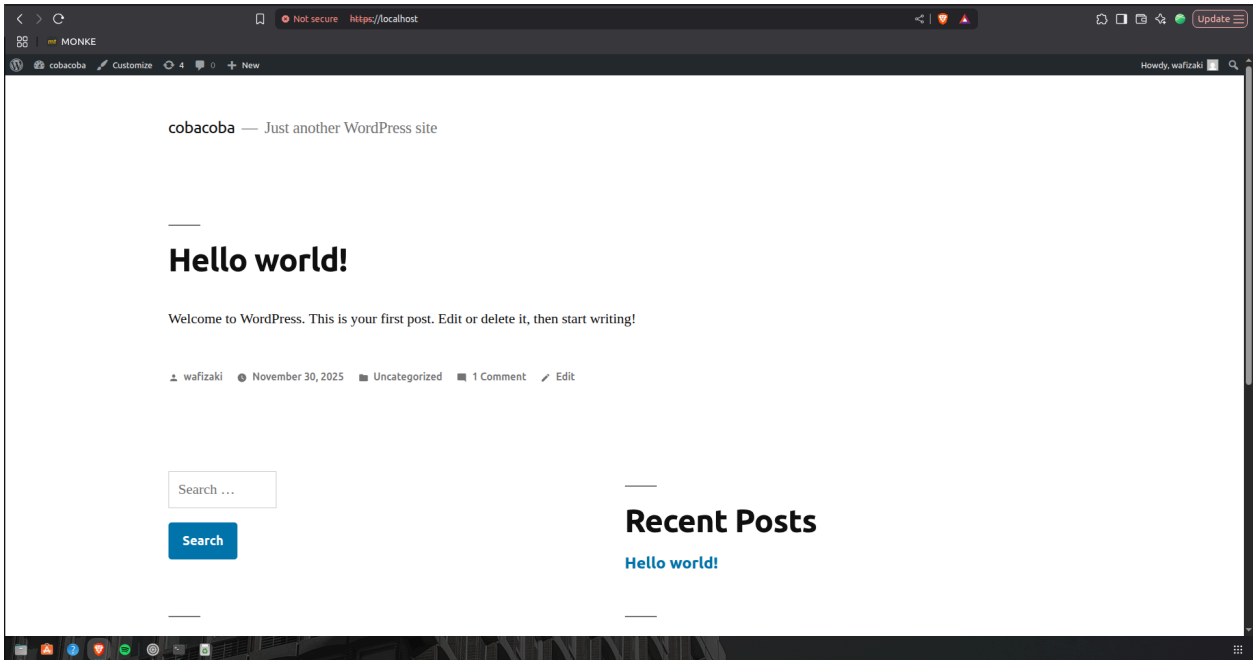
Gambar 3.3: Bukti Keamanan HTTPS

c. Pembahasan Case 3

Case 3 merupakan implementasi arsitektur Microservices yang umum digunakan di industri. Sistem memisahkan antara aplikasi Frontend dan Backend penyimpanan data.

Kami membangun sistem CMS WordPress di mana container WordPress bertindak sebagai aplikasi, container MySQL bertindak sebagai database, dan Nginx bertindak sebagai Reverse Proxy di depannya. Kedua container (WP dan DB) saling berkomunikasi melalui jaringan internal Docker, membuktikan bahwa aplikasi kompleks dapat dibangun dari layanan-layanan kecil yang terpisah namun saling "mengobrol".

Cara Run : Menjalankan perintah “sudo docker compose up -d”



Gambar 3.4: Tampilan Dashboard WordPress

d. Pembahasan Case 4

Jika case sebelumnya menggunakan image yang sudah jadi, Case 4 berfokus pada pembuatan Custom Image menggunakan Dockerfile.

Skenario ini menjalankan aplikasi buatan sendiri (case4-app) dan sebuah container Alpine Linux yang menjalankan script otomatis (while true). Di sini dipelajari bagaimana memasukkan kode program sendiri ke dalam container (Build process) dan mensimulasikan proses background (Worker) yang berjalan terus menerus. Ini merepresentasikan pembuatan logika bisnis spesifik yang tidak tersedia pada image publik.

Cara Run: Karena menggunakan custom image maka perintah menjadi “sudo docker compose up -d –build”

Gambar 3.8: Proses Build Custom Image

e. Pembahasan Case 5

1. Latar Belakang Pengembangan

Case 5 adalah pengembangan mandiri yang menggabungkan seluruh pembelajaran dari Case 1-4 menjadi satu sistem utuh yang merepresentasikan Cloud Computing Sebenarnya.

Jika Case 3 dan 4 masih memiliki kelemahan berupa *Single Point of Failure*, Case 5 dirancang dengan arsitektur *High Availability Cluster*. Sistem ini mensimulasikan infrastruktur IoT untuk pertanian cerdas (*Smart Farming*).

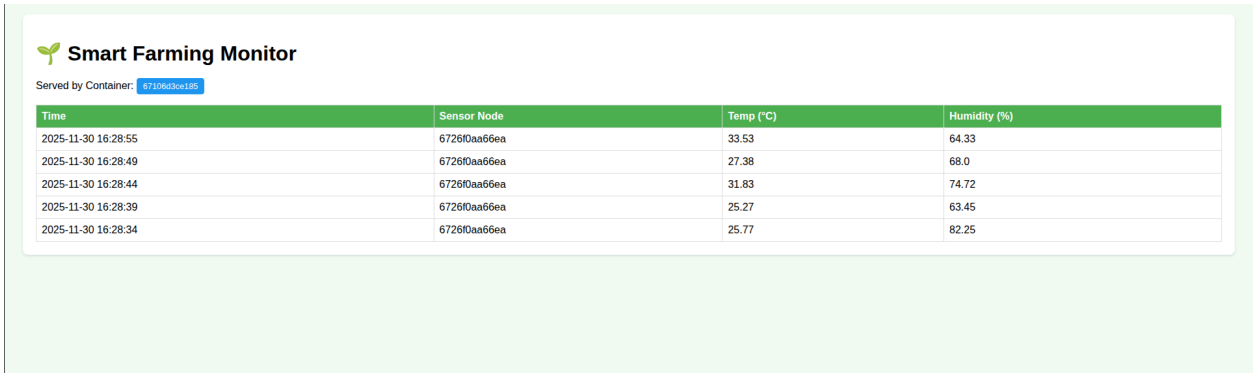
Komponen & Arsitektur:

1. Sensor Worker (IoT Simulator): Menggunakan teknik dari Case 4 (Custom Scripting), dibuat container Python yang bertindak sebagai sensor. Ia men-generate data suhu/kelembaban dummy dan mengirimkannya ke database secara *real-time*.
2. Dashboard Cluster (Replica Set): Alih-alih satu server, sistem menjalankan 3 Container Dashboard (Web App) secara bersamaan.
3. Secure Load Balancer: Menggunakan teknik dari Case 2 (Nginx HTTPS), load balancer ini bertugas membagi trafik pengunjung ke 3 container dashboard tersebut secara bergantian (*Round Robin*).

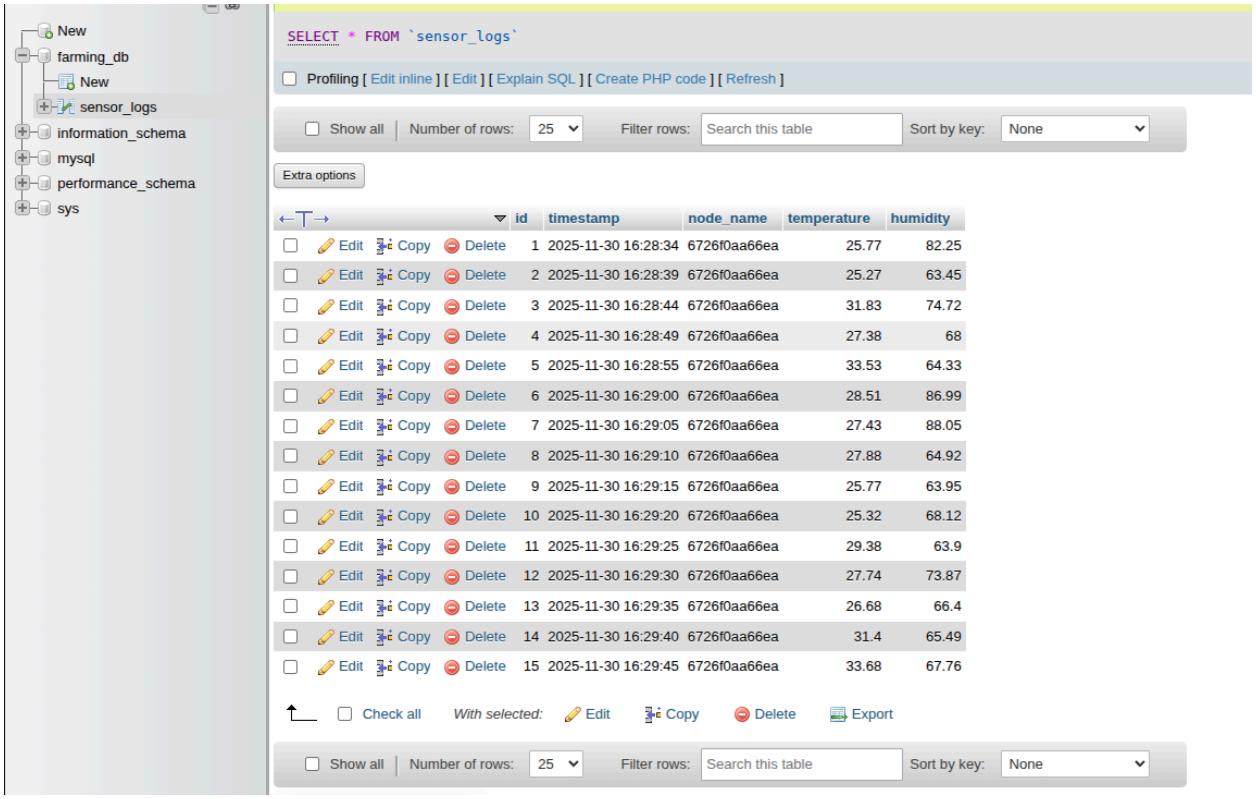
Kenapa ini perkembangan baru? Ini merepresentasikan lompatan dari sekadar "menjalankan aplikasi" menjadi "mendesain infrastruktur berskala besar".

- **Resiliency:** Jika Dashboard Node 1 mati, Nginx otomatis mengalihkan trafik ke Node 2 atau 3. Sistem tidak pernah *down*.
- **Scalability:** Sistem dapat menangani beban lebih tinggi dengan menambah jumlah container dashboard (Horizontal Scaling).

Cara Run: Menjalankan perintah “sudo docker compose up -d –build”



Gambar 4.1: Demo Load Balancing



Gambar 4.2: Monitoring Database

4. Penutup

Melalui rangkaian tugas ini, telah dipelajari evolusi pengelolaan server dari yang paling dasar (statis) hingga kompleks (terdistribusi). Penerapan Docker Compose mempermudah orkestrasi layanan yang terdiri dari banyak komponen. Implementasi Case 5 membuktikan bahwa dengan teknologi container, kita dapat membangun sistem yang aman, reliabel, dan skalabel layaknya arsitektur yang digunakan oleh perusahaan teknologi besar.