Muhammad Iqbal Ramadhan
5025221274

Cloud Computing
EF4704

---

# Tugas 2

Repository: github.com/LaptopMerah

## Pendahuluan

Tugas ini membahas implementasi konsep kontainerisasi menggunakan Docker Compose. Compose merupakan alat orkestrasi esensial yang memungkinkan pengguna untuk mendefinisikan dan menjalankan aplikasi multi-kontainer secara efisien melalui satu file konfigurasi berbasis YAML. Dalam ekosistem pengembangan modern, alat ini berfungsi untuk menyederhanakan manajemen tumpukan (*stack*) aplikasi yang kompleks, memastikan bahwa setiap komponen dapat beroperasi secara serentak dalam lingkungan yang terisolasi namun tetap saling terhubung. Penggunaan Docker Compose menjamin konsistensi lingkungan dari tahap pengembangan hingga produksi, menghilangkan masalah kompatibilitas sistem yang sering terjadi pada konfigurasi manual.

Tujuan dari tugas ini adalah untuk mempraktikkan penyusunan arsitektur *microservices* yang terintegrasi dan aman pada sebuah aplikasi yang saya buat yaitu *Link Shortener*. Secara spesifik, praktik ini difokuskan pada penggabungan empat layanan independen—*database*, *backend*, *frontend*, dan *gateway*—agar dapat bekerja sebagai satu kesatuan sistem yang harmonis. Melalui studi kasus ini, diterapkan konsep-konsep tingkat lanjut seperti penggunaan *reverse proxy* (gateway) sebagai satu-satunya akses publik untuk meningkatkan keamanan, mekanisme *healthcheck* untuk menjaga urutan *startup* layanan, serta manajemen volume untuk persistensi data, sehingga memberikan pemahaman menyeluruh tentang bagaimana membangun infrastruktur aplikasi yang *scalable* dan *maintainable*.

## Tugas yang Diberikan

- Menjalankan 4 Case yang disediakan pada repository github. Setiap case, diperlukan dokumentasi yang mencakup cara menjalankan, screenshot hasil, dan penjelasan proses.
- Membuat Case baru (case 5) berbasis pada case sebelumnya. Proyek ini harus dilengkapi dengan:
    - Penjelasan skenario atau kondisi di mana kreasi tersebut penting atau cocok untuk digunakan.
    - Gambar arsitektur sistem yang dibuat.
    - Script pendukung dan screenshot hasil serta penjelasannya.

## Case 1:

Pada case1 ini terdapat 2 file dan 1 folder html yakni config Nginx, Website HTML, dan Docker Compose.

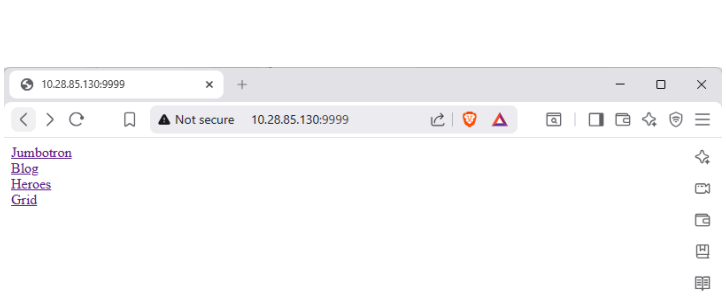
```
LaptopMerah@LaptopMerah:case1
> ls html/
assets  grid    jumbotron
blog    heroes  index.html
```

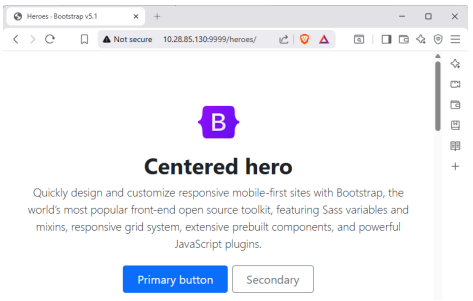
```
LaptopMerah@LaptopMerah:case1
> cat nginx-conf/nginx.conf
server {
        listen 80;
        listen [::]:80;

        server_name _;

        index index.html index.htm;

        root /var/www/html;

        location = /favicon.ico {
                log_not_found off; access_log off;
        }
        location = /robots.txt {
                log_not_found off; access_log off; allow all;
        }
        location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {
                expires max;
                log_not_found off;
        }
}
```

```
LaptopMerah@LaptopMerah:case1
> cat docker-compose.yml
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
```

**nginx.conf** mengatur untuk listen pada port 80 dan serve HTML yang ada di direktori /var/www/html di dalam container yang di mounting dari host $(pwd)/html. **docker-compose.yml** dimana untuk membuat satu service dengan nama *webserver* dengan nginx:1.15.12-alpine. Service ini listen pada port 9999 pada host dan di forward ke port 80 di dalam container sesuai nginx.conf. Terdapat 2 volume yang dipasang, yakni di website pada ./html ke /var/www/html dan config nginx-nya pada ./nginx-conf ke /etc/nginx/conf.d.

```
LaptopMerah@LaptopMerah:case1
> docker compose up -d
WARN[0000] /home/LaptopMerah/Cloud-Computing/Containers/Task 2 Compose/case1/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 1/1
 ✔Container case1-webserver-1  Running                    0.0s
LaptopMerah@LaptopMerah:case1
> docker ps
CONTAINER ID   IMAGE                  COMMAND
        CREATED              STATUS              PORTS
                                   NAMES
067f28d84ba9   nginx:1.15.12-alpine   "nginx -g 'daemon of…"   About a minute ago   Up About a minute   0.0.0.0:9999→80/tcp, [::]:9999→80/tcp   case1-webserver-1
```




docker compose dijalankan dengan `**docker compose up -d**`, dan container *case1-webserver-1* sudah berhasil dijalankan. Lalu pada localhost:9999 yang saya jalankan di 10.28.85.130, Terlihat pada "/" terlihat 4 link sesuai dengan isi dari index.html. Ketika diklik salah satu dari link tersebut, user akan diarahkan sesuai dengan link tersebut. Pada dengan contoh pada link `*Heroes*` maka akan diarahkan ke */heroes* dan akan muncul tampilan heroes.

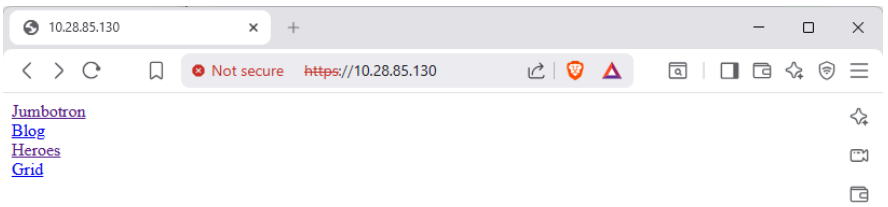
## Case 2:

Pada case2 ini sama dengan case1, dimana docker compose ditambahkan SSL dengan tambahan folder certs.

```
LaptopMerah@LaptopMerah:case2
> ls
certs  nginx-conf
html   docker-compose.yml
LaptopMerah@LaptopMerah:case2
> cat docker-compose.yml
version: '3'

services:
  webserver:
    image: nginx:1.15.12-alpine
    restart: unless-stopped
    privileged: true
    ports:
      - "443:443"
      - "80:80"
    volumes:
      - ./certs:/certs
      - ./html:/var/www/html
      - ./nginx-conf/nginx.secure.conf:/etc/nginx/conf.d/nginx.conf
    networks:
      - app-network
networks:
  app-network:
    driver: bridge
```

**docker-compose.yml** pada case 2 ini akan membuat service dengan nama *webserver* yang mem-binding port 443 pada host dan di forward ke port 443 juga, begitupun dengan port 80 untuk dalam container yang sesuai nginx.secure.conf. Pada case 2 ini juga bertambah mounting volumes untuk certs. Lalu pada **nginx.secure.conf** dilakukan Auto-Redirect (HTTP 301) untuk memaksa user yang mengakses website melalui protokol HTTP langsung dialihkan ke HTTPS.

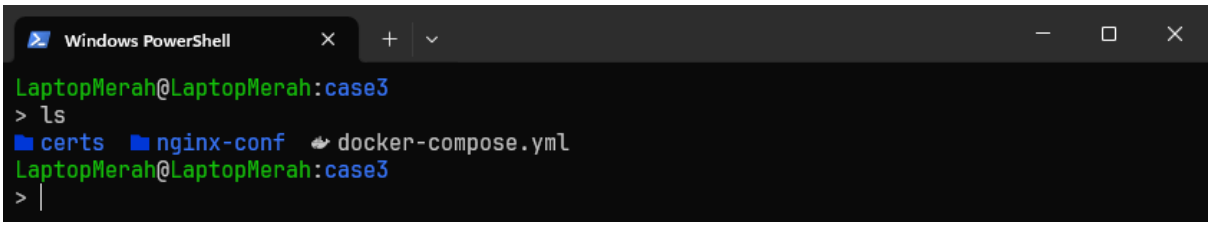
```
LaptopMerah@LaptopMerah:case2
> docker compose up -d
WARN[0000] /home/LaptopMerah/Cloud-Computing/Containers/Task 2 Compose/case2/docker-compose.yml: the
attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 2/2
 ✔Network case2_app-network    Created                                                    0.0s
 ✔Container case2-webserver-1  Started                                                    0.2s
LaptopMerah@LaptopMerah:case2
> docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED         STATUS         PORTS
                                                                        NAMES
dbe2f3aec1b5   nginx:1.15.12-alpine  "nginx -g 'daemon of…"   4 seconds ago   Up 3 seconds   0.0.0.0
:443→443/tcp, [::]:443→443/tcp, 0.0.0.0:8080→80/tcp, [::]:8080→80/tcp   case2-webserver-1
```



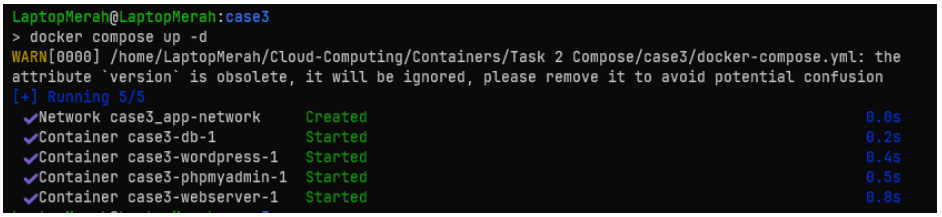

docker compose dijalankan dengan `**docker compose up -d**`, dan container *case2-webserver-1* sudah berhasil dijalankan. Lalu pada webserver ketika diakses akan redirect ke HTTPS

## Case 3:

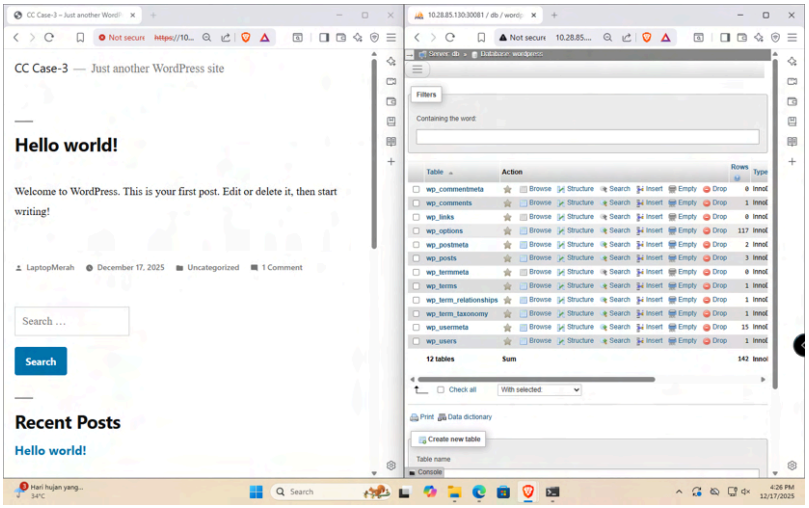
Pada case3 ini dilakukan instalasi wordpress dengan persistent data dengan mysql, phpmyadmin, nginx. dengans isi folder case3:




**docker-compose.yml** tersebut membangun aplikasi **WordPress** yang terdiri dari **4 service** yang terhubung dalam satu **app-network**. Service **db** menggunakan MySQL 8.0 sebagai database dengan konfigurasi penting seperti MYSQL_ROOT_PASSWORD dan MYSQL_DATABASE dari file .env, serta volume untuk penyimpanan data persisten. Service **phpmyadmin** diekspos melalui port PHPMYADMIN_PORT untuk memudahkan pengelolaan database, sedangkan service **wordpress** terhubung ke database menggunakan variabel WORDPRESS_DB_HOST, WORDPRESS_DB_USER, dan WORDPRESS_DB_PASSWORD dengan file aplikasi disimpan pada volume bersama. Service **webserver** menggunakan Nginx untuk menyajikan WordPress melalui HTTP dan HTTPS dengan dukungan SSL/TLS dari folder sertifikat dan konfigurasi yang disediakan.




docker compose dijalankan dengan `**docker compose up -d**`, dengan 4 container berjalan sesuai dengan yang telah didefinisikan dalam docker-compose.yml t

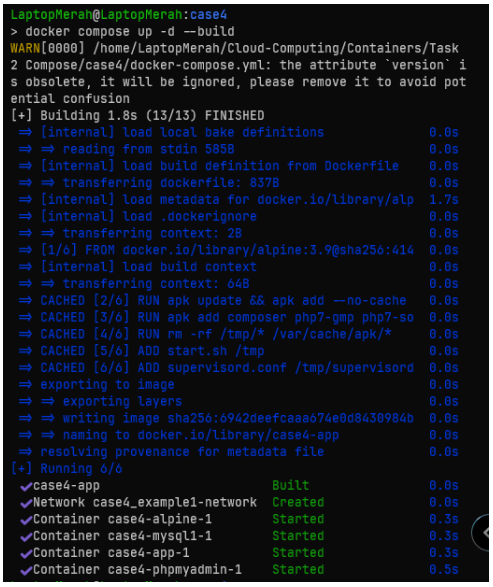



WordPress dapat di akses pada 10.28.85.130 yang membuktikan telah berhasil running dan bisa meng setup wordpress yang menghasilkan halaman pada gambar kiri dan pada PHPMyAdmin dapat diakses pada 10.28.85.130:30081 dengan credential sesuai pada .env, dan sudah terbuat database dari wordpress tersebut yang bisa dilihat pada gambar kanan.

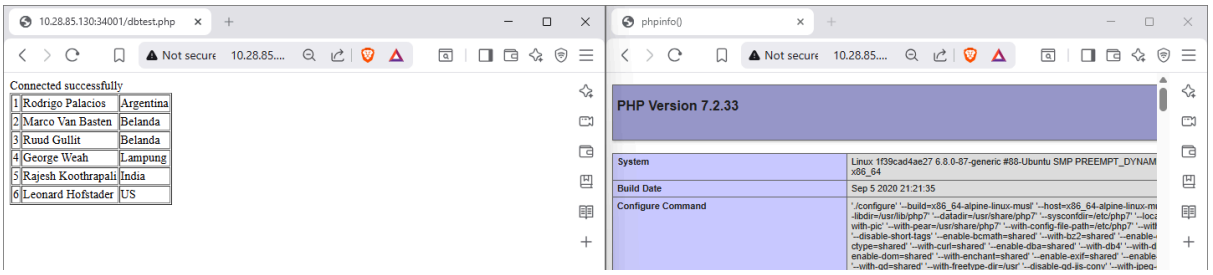
## Case 4:

Jika sebelumnya (case3) menggunakan **image yang sudah tersedia** seperti WordPress, MySQL, dan Nginx, sedangkan Docker Compose pada case4 ini **berfokus pada pembuatan custom image menggunakan Dockerfile** untuk menjalankan aplikasi web. Compose ini membangun aplikasi berbasis Apache yang terhubung dengan service **mysql1** sebagai database, dengan data yang disimpan secara persisten melalui volume. Service **app** dijalankan dari hasil build Dockerfile lokal, diekspos melalui port **34001**, dan dikonfigurasi menggunakan file Apache (httpd.conf), PHP (php.ini), serta SSL yang di-mount dari host.

Selain itu, service **phpmyadmin** disediakan untuk memudahkan pengelolaan database melalui browser pada port **10000**, dan service **alpine** digunakan sebagai container ringan untuk kebutuhan testing dan debugging. Seluruh service berjalan dalam satu jaringan **example1-network** sehingga dapat saling berkomunikasi secara internal dengan aman dan terisolasi.



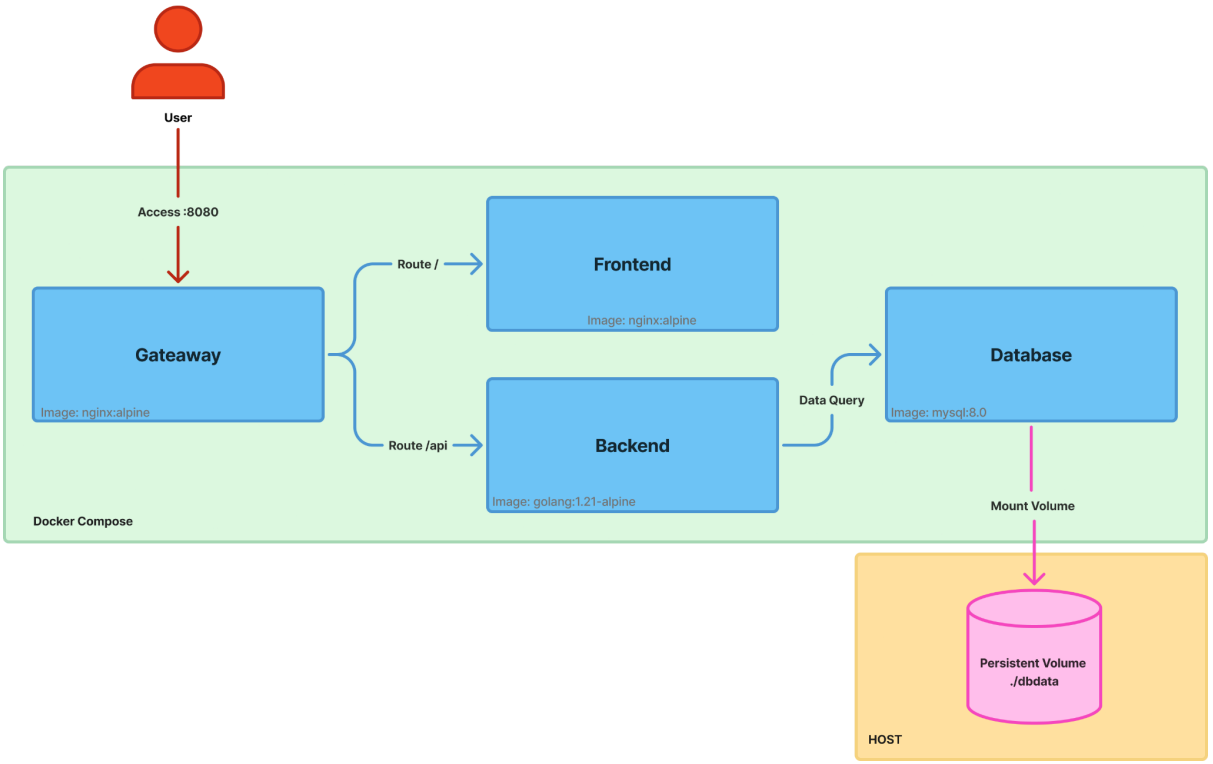

docker compose dijalankan dengan perintah **`docker compose up -d --build`** digunakan untuk membangun image dari Dockerfile dan menjalankan container case4-app secara bersamaan.



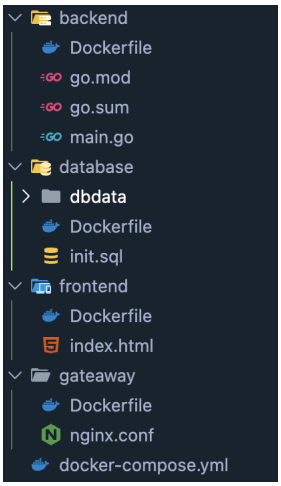

Setelah Docker Compose dijalankan, aplikasi bisa diakses melalui **10.28.85.130:34001**. Saat dibuka akan muncul halaman phpinfo yang menandakan PHP sudah berjalan sesuai dengan versi yang didefinisikan di Dockerfile seperti gambar kanan. Selanjutnya, dilakukan pengecekan database menggunakan script **restoredb.sh** yang sudah tersedia di folder /dbscripts untuk me-restore database. Hasilnya dapat dilihat melalui **10.28.85.130:34001/dbtest.php**, di mana data dari database berhasil ditampilkan, menandakan koneksi antara aplikasi dan database berjalan dengan baik seperti gambar kiri.

## Case 5:

**Spesifikasi VM** yang digunakan: 4096 MB Memory, 2 vCPU, 32GB Disk Size, IP address 10.21.85.130. Port yang digunakan pada case ini utamanya port 8080 dan 443.



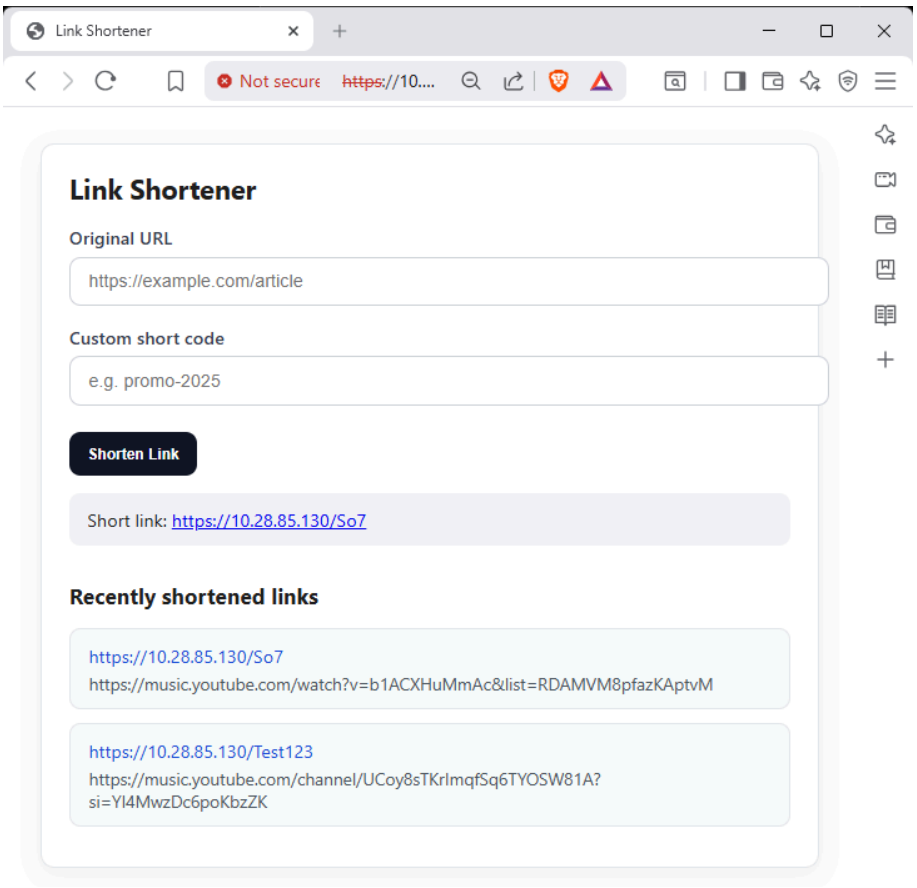

Arsitektur ini sangat ideal diterapkan pada pengembangan aplikasi modern berbasis *microservices* yang memisahkan sisi antarmuka (*Frontend*) dan logika bisnis (*Backend*), terutama di lingkungan produksi yang menuntut standar keamanan tinggi dan efisiensi manajemen jaringan. Penggunaan *Gateway* sebagai satu-satunya pintu masuk yang menyatukan seluruh lalu lintas data di bawah satu domain, sekaligus bertindak sebagai tameng keamanan yang menyembunyikan layanan internal sensitif (seperti Database dan API) dari akses publik secara langsung. Selain itu, skenario ini memungkinkan sentralisasi konfigurasi SSL/HTTPS dan memberikan fleksibilitas pemeliharaan, dimana layanan *Backend* dapat diperbarui atau dipulihkan secara independen tanpa memutus akses pengguna terhadap tampilan *Frontend* aplikasi.




```yaml
version: "3.9"

name: case5-link-shortener

services:
  database:
    build: ./database
    container_name: case5-db
    environment:
      MYSQL_ROOT_PASSWORD: "case5rootpass"
      MYSQL_DATABASE: "shorten_link"
      MYSQL_USER: "case5user"
      MYSQL_PASSWORD: "case5pass"
    volumes:
      - ./database/dbdata:/var/lib/mysql
    restart: unless-stopped
    healthcheck:
      test:
        [
          "CMD-SHELL",
          "mysqladmin ping -h 127.0.0.1 -p$$MYSQL_ROOT_PASSWORD --silent",
        ]
      interval: 10s
      timeout: 5s
      retries: 10

  backend:
    build: ./backend
    container_name: case5-backend
    environment:
      DB_HOST: database
      DB_PORT: "3306"
      DB_USER: case5user
      DB_PASS: case5pass
      DB_NAME: shorten_link
      DB_MAX_RETRIES: "60"
      DB_RETRY_DELAY: "2s"
    depends_on:
      database:
        condition: service_healthy
    restart: unless-stopped

  frontend:
    build: ./frontend
    container_name: case5-frontend
    restart: unless-stopped

  gateway:
    build: ./gateaway
    container_name: case5-gateway
    ports:
      - "8080:80"
      - "443:443"
    depends_on:
      - frontend
      - backend
    restart: unless-stopped
```

Struktur case5 merepresentasikan aplikasi Link Shortener berbasis microservices yang terbagi menjadi empat sub-direktori independen—**backend**, **frontend**, **database** (persisten), dan **gateway** (reverse proxy)—di mana file docker-compose.yml di root berfungsi sebagai orkestrator pusat yang mengintegrasikan seluruh stack tersebut. Konfigurasi ini tidak hanya menyatukan arsitektur dengan menjadikan gateway sebagai satu-satunya pintu masuk publik pada port 8080 dan 443, tetapi juga menjamin stabilitas sistem melalui manajemen jaringan terisolasi serta mekanisme healthcheck otomatis yang memastikan layanan berjalan dalam urutan yang tepat dan aman.

Pengoperasian cukup mengeksekusi **docker compose build** untuk membangun seluruh image layanan secara paralel, diikuti dengan **docker compose up -d** untuk mengaktifkan kontainer di latar belakang. Proses ini meniadakan kerumitan konfigurasi manual per layanan, memungkinkan seluruh lingkungan sistem siap digunakan dalam hitungan detik hanya dengan dua baris perintah sederhana.




Hasil dari implementasi arsitektur *microservices*, menampilkan Frontend yang diakses melalui *Gateway* (terindikasi dari alamat IP pada bilah alamat browser). Pengguna dapat memasukkan URL panjang ke dalam formulir, memicu logika *Backend* untuk menghasilkan kode unik sesuai input (seperti `/So7`) saat tombol "Shorten Link" ditekan, serta memvalidasi konektivitas *Database* yang terbukti dari munculnya daftar "Recently shortened links" yang menampilkan riwayat data tautan yang telah dipersistenkan secara *real-time*.

## Kesimpulan

Dengan tugas ini yang mempelajari kontainerisasi dengan Docker Compose, saya mendapatkan pemahaman mendalam mengenai orkestrasi arsitektur *microservices* melalui implementasi studi kasus aplikasi **Link Shortener**. Melalui konfigurasi file YAML yang terpusat, saya berhasil mengintegrasikan empat layanan terpisah (Database, Backend, Frontend, dan Gateway) ke dalam satu ekosistem jaringan yang aman dan terisolasi, di mana penggunaan *Gateway* sebagai satu-satunya akses publik serta penerapan *healthcheck* dan *volume* menjamin stabilitas sistem serta persistensi data. Praktik ini membuktikan bahwa Docker Compose tidak hanya menyederhanakan kompleksitas *deployment* sistem terdistribusi, tetapi juga menawarkan efisiensi tinggi dalam membangun aplikasi yang *scalable*, konsisten, dan mudah dipelihara layaknya standar lingkungan production.