KOMPUTASI AWAN (T)

TUGAS 2

Dosen Pengampu:

Royyana Muslim Ijtihadie, S.Kom.,M.Kom., Ph.D.

Disusun Oleh:

Hilmi Fawwaz Sa’ad

NRP. 5025221103

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS

DEPARTEMEN TEKNIK INFORMATIKA

SURABAYA

2025/2026

Tugas 2

Repository Github: https://github.com/hilmifawwazsaad/CloudComp_Class_2025/tree/main/task-2

Pendahuluan

Docker Compose adalah tool yang sangat membantu untuk menjalankan aplikasi Docker yang terdiri dari banyak container sekaligus. Dengan Docker Compose, kita bisa mendefinisikan semua konfigurasi aplikasi seperti service, network, dan volumes hanya dalam satu file YAML/YML. Tugas ini mengeksplorasi penggunaan Docker Compose mulai dari yang paling sederhana seperti web server statis, database, reverse proxy, dan konfigurasi SSL/TLS untuk keamanan. Tujuan tugas ini adalah memahami cara kerja Docker Compose dalam mengelola aplikasi multi-container yang servicenya saling terhubung satu sama lain.

Tugas yang Diberikan

Tugas yang diberikan adalah menjalankan semua studi kasus (cases) yang ada pada repository https://github.com/rm77/cloud2023/tree/master/containers/compose/compose. Setiap studi kasus harus dijalankan menggunakan Docker Compose dengan konfigurasi dan pengelolaan layanan yang berbeda. Lingkup pengerjaan mencakup empat studi kasus utama dari repository tersebut, ditambah satu studi kasus tambahan sebagai bentuk eksplorasi dan pengembangan berdasarkan referensi yang sama. Adapun penjelasan masing-masing studi kasus adalah sebagai berikut:

1. Menjalankan web server Nginx dengan protocol http, tanpa sertifikat
2. Menjalankan web server Nginx dengan protocol https menggunakan sertifikat
3. Menjalankan WordPress menggunakan web server Nginx dengan protocol https dan sertifikat
4. Menjalankan aplikasi PHP dan phpMyAdmin dalam container terpisah dan menghubungkannya untuk pengelolaan database
5. [CASE TAMBAHAN] Menjalankan Layanan QR Code Generator berbasis Flask dengan Docker Compose

Setelah seluruh studi kasus berhasil dijalankan, dilakukan proses dokumentasi yang mencakup penjelasan terhadap script/code yang digunakan, langkah menjalankan tiap case, alur proses yang terjadi, serta hasil yang diperoleh dari masing-masing program.

Implementasi dan Pembahasan

- ***Case 1***

Pada studi kasus ini terdapat tiga komponen. Komponen pertama adalah file `docker-compose.yml` yang berfungsi sebagai orchestration configuration untuk mendefinisikan services dan network yang digunakan. Komponen kedua adalah file `nginx.conf` yang ada di folder `nginx-conf/` untuk mengatur behavior web server. Komponen terakhir adalah komponen statis content website di folder `html/` yang akan diserve oleh web server Nginx.

```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/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
```

Gambar 1. Script docker-compose.yml

Ketika script ini dijalankan, Docker Compose akan membuat satu kontainer Nginx menggunakan image `nginx:1.15.12-alpine` dengan format Compose versi 3. Kontainer dikonfigurasi agar otomatis menyala ulang dengan `restart: unless-stopped`, memetakan port 80 di dalam kontainer ke 9999 di host sehingga dapat diakses melalui `http://localhost:9999`. Selain itu, terdapat dua volume yang menghubungkan folder lokal `./html` dan `./nginx-conf` ke direktori web dan konfigurasi Nginx di dalam kontainer sehingga perubahan file langsung diterapkan tanpa rebuild. Service ini juga ditempatkan pada jaringan `app-network` bertipe `bridge` untuk memudahkan komunikasi antar kontainer.

```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/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;
        }
```

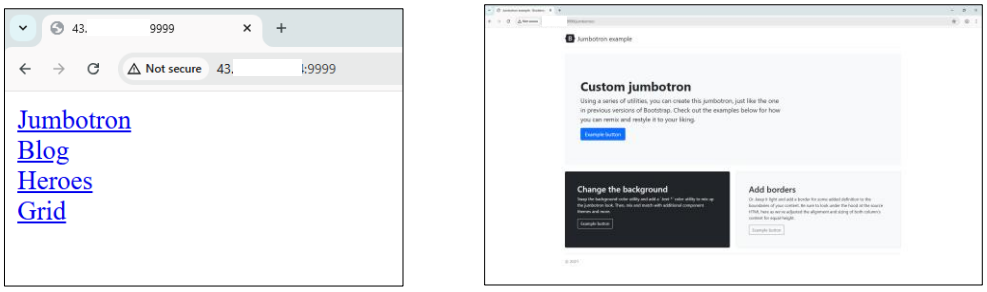
Gambar 2. Script nginx.conf

Selain konfigurasi pada Docker Compose, file `nginx.conf` juga mengatur bagaimana Nginx menangani request di dalam kontainer. Pada konfigurasi ini, Nginx diset untuk mendengarkan permintaan melalui port 80 dan menerima seluruh domain menggunakan `server_name _`. Direktori utama aplikasi ditetapkan pada `/var/www/html`, yang terhubung langsung dengan volume `./html` dari host. Ketika pengguna mengakses root path, Nginx akan mencari file `index.html` atau `index.htm` sebagai halaman utama. Beberapa location block juga disertakan, misalnya untuk `/favicon.ico` agar tidak menampilkan error jika file tidak ada, serta `/robots.txt` yang tetap dapat diakses tanpa dicatat dalam log.

```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case1$ sudo docker compose up -d
[sudo] password for hfwxyz:
WARN[0000] /home/hfwxyz/cc25/task-2/case1/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] up 7/7
 ✓ Image nginx:1.15.12-alpine  Pulled                                    6.3s
 ✓ Network case1_app-network   Created                                   0.1s
 ✓ Container case1-webserver-1 Created                                   0.1s
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case1$ docker ps
permission denied while trying to connect to the docker API at unix:///var/run/docker.sock
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case1$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
ef1115ceb88b   nginx:1.15.12-alpine  "nginx -g 'daemon of…"   16 seconds ago   Up 16 seconds   0.0.0.0:9999->80/tcp, [::]:9999->80/tcp   case1-webserver-1
```

Gambar 3. Menjalankan Docker Compose

Docker Compose dijalankan dengan command `sudo docker compose up -d` yang berhasil melakukan pull image `nginx:1.15.12-alpine`, membuat network `case1_app-network`, dan menjalankan container `case1-webserver-1`.



Gambar 4. Hasil localhost pada port 9999

Halaman web ditampilkan melalui `http://localhost:9999`. Jika kita mengeklik "Jumbotron", browser akan diarahkan ke `/jumbotron` dan menampilkan file `index.html` yang ada di folder jumbotron.

- ***Case 2***

Studi kasus ini memiliki empat komponen utama yang membentuk arsitektur web server dengan SSL/TLS. Komponen pertama adalah `file docker-compose.yml` yang berfungsi sebagai orchestration configuration. Komponen kedua berupa direktori `nginx-conf/` yang berisi dua file konfigurasi yaitu `nginx.conf` untuk HTTP standard configuration dan `nginx.secure.conf` untuk HTTPS configuration dengan SSL/TLS enabled. Komponen ketiga adalah direktori `html/` yang menyimpan static web content. Komponen keempat adalah direktori `certs/` yang berisi SSL certificate dan private key untuk enkripsi HTTPS connection.

```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/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
```

Gambar 5. Script docker-compose.yml

Ketika script `docker-compose.yml` ini dijalankan, alurnya sama dengan studi kasus yang pertama. Namun, terdapat perbedaan pada port dan volume. Untuk port, terdapat dua port utama diekspos, yaitu 80 untuk HTTP dan 443 untuk HTTPS, sehingga layanan dapat diakses melalui `http://localhost` maupun `https://localhost`. Kemudian untuk volume beberapa juga di-mount dari host, termasuk folder `./certs` yang berisi file SSL, `./html` sebagai direktori file web, serta berkas konfigurasi khusus `nginx.secure.conf` yang menggantikan konfigurasi default Nginx di dalam kontainer.

Konfigurasi Nginx pada case ini tetap menggunakan `nginx.conf` yang sama dengan case pertama, namun ditambahkan `nginx.secure.conf` untuk mengaktifkan HTTPS dengan SSL/TLS.

```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case2$ cat nginx-conf/nginx.secure.conf
server {
    listen 80 default_server;
    server_name _;
    return 301 https://$host$request_uri;
}


server {
        listen 443 ssl;

        server_name _;
        ssl_certificate                 /certs/MyCert.crt;
        ssl_certificate_key             /certs/MyPrivate.key;
        ssl_protocols TLSv1.1 TLSv1.2;
        ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
        ssl_prefer_server_ciphers on;
        ssl_session_cache shared:SSL:10m;

        client_max_body_size        0;
        chunked_transfer_encoding   on;

        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;
        }
}
```

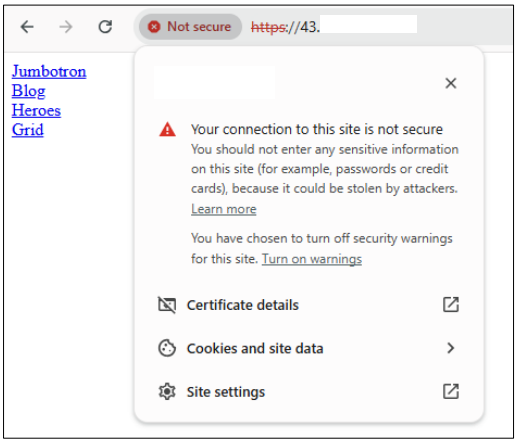
Gambar 6. Script nginx.secure.conf

Pada bagian pertamanya, Nginx mendengarkan request pada port 80 dan langsung melakukan redirect permanen (301) ke HTTPS, sehingga setiap akses melalui `http://` otomatis diarahkan ke `https://` sebagai praktik terbaik keamanan. Bagian kedua mendefinisikan server HTTPS yang berjalan di port 443, menggunakan sertifikat dan private key yang di-mount dari folder `/certs`. Konfigurasi SSL mencakup pengaturan protokol TLS, daftar cipher, serta cache session untuk meningkatkan keamanan dan performa. Selain itu, batas ukuran unggahan diatur tanpa batas melalui `client_max_body_size 0` dan direktori root diarahkan ke `/var/www/html`.

```
hfwxyz@M-5-14-ubuntu:~/cc25/task-2/case2$ sudo docker compose up -d
WARN[0000] /home/hfwxyz/cc25/task-2/case2/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] up 2/2
 ✓ Network case2_app-network    Created                                                                                    0.0s
 ✓ Container case2-webserver-1  Created                                                                                    0.0s
hfwxyz@M-5-14-ubuntu:~/cc25/task-2/case2$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED         STATUS         PORTS                                                                                  NAMES
4ae3734ae2c0   nginx:1.15.12-alpine  "nginx -g 'daemon of…"   7 seconds ago   Up 6 seconds   0.0.0.0:80->80/tcp, [::]:80->80/tcp, 0.0.0.0:443->443/tcp, [::]:443->443/tcp   case2-webserver-1
```

Gambar 7. Menjalankan Docker Compose

Container berhasil dijalankan dengan nama `case2-webserver-1`. Sama seperti studi kasus pertama, halaman website dapat diakses melalui `https://localhost` menggunakan protokol HTTPS dengan memanfaatkan sertifikat yang berada di direktori `certs/` dan dipetakan ke dalam container. Untuk hasilnya bisa dilihat pada gambar di bawah.



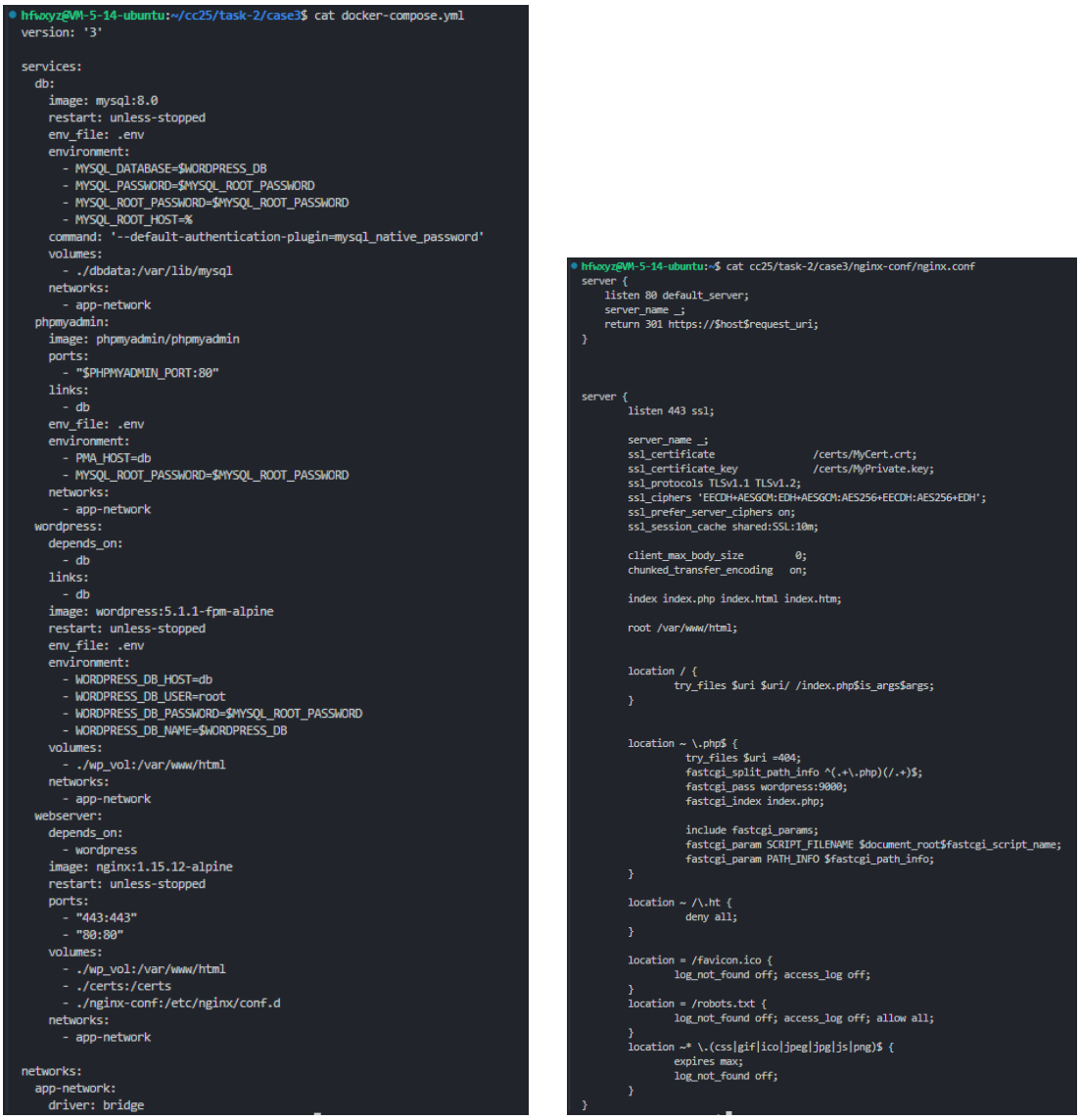
Gambar 8. Hasil dari https://localhost

Gambar tersebut menunjukkan bahwa pemasangan ssl berhasil berhasil dan bisa di cek pada “Certificate details”.

- ***Case 3***

Pada studi kasus ini, kita diminta untuk menyediakan layanan WordPress dan phpMyAdmin yang berjalan di dalam container terpisah.

Script `docker-compose.yml` ini menjalankan WordPress melalui empat service yang terhubung dalam satu bridge network app-network. Service `db` memakai MySQL 8.0 dengan penyimpanan persisten di `dbdata` dan konfigurasi `mysql_native_password` untuk kompatibilitas, meskipun penggunaan `MYSQL_ROOT_HOST=%` kurang aman untuk produksi. Service phpmyadmin menyediakan antarmuka GUI untuk mengelola database melalui port yang ditentukan di `.env`. Service `wordpress` menggunakan WordPress 5.1.1 berbasis PHP-FPM Alpine, menyimpan file di `wp_vol`, dan terhubung ke MySQL menggunakan hostname `db`. Service `webserver` menggunakan Nginx sebagai reverse proxy untuk HTTP/HTTPS, meneruskan request PHP ke WordPress melalui FastCGI, serta memuat sertifikat SSL dan konfigurasi dari `certs` dan `nginx-conf`. Semua service saling berkomunikasi melalui hostname masing-masing melalui bridge network. Untuk lebih jelasnya bisa dilihat pada gambar di bawah ini.

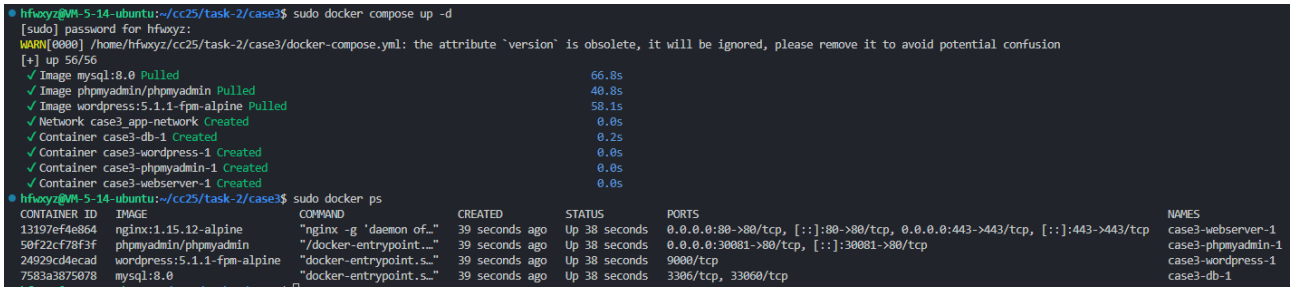


```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case3$ cat docker-compose.yml
version: '3'

services:
  db:
    image: mysql:8.0
    restart: unless-stopped
    env_file: .env
    environment:
      - MYSQL_DATABASE=$WORDPRESS_DB
      - MYSQL_PASSWORD=$MYSQL_ROOT_PASSWORD
      - MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD
      - MYSQL_ROOT_HOST=%
    command: '--default-authentication-plugin=mysql_native_password'
    volumes:
      - ./dbdata:/var/lib/mysql
    networks:
      - app-network
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    ports:
      - "$PHPMYADMIN_PORT:80"
    links:
      - db
    env_file: .env
    environment:
      - PMA_HOST=db
      - MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD
    networks:
      - app-network
  wordpress:
    depends_on:
      - db
    links:
      - db
    image: wordpress:5.1.1-fpm-alpine
    restart: unless-stopped
    env_file: .env
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=root
      - WORDPRESS_DB_PASSWORD=$MYSQL_ROOT_PASSWORD
      - WORDPRESS_DB_NAME=$WORDPRESS_DB
    volumes:
      - ./wp_vol:/var/www/html
    networks:
      - app-network
  webserver:
    depends_on:
      - wordpress
    image: nginx:1.15.12-alpine
    restart: unless-stopped
    ports:
      - "443:443"
      - "80:80"
    volumes:
      - ./wp_vol:/var/www/html
      - ./certs:/certs
      - ./nginx-conf:/etc/nginx/conf.d
    networks:
      - app-network

networks:
  app-network:
    driver: bridge
```

```
hfwxyz@VM-5-14-ubuntu:~$ cat cc25/task-2/case3/nginx-conf/nginx.conf
server {
    listen 80 default_server;
    server_name _;
    return 301 https://$host$request_uri;
}



server {
        listen 443 ssl;

        server_name _;
        ssl_certificate                 /certs/MyCert.crt;
        ssl_certificate_key             /certs/MyPrivate.key;
        ssl_protocols TLSv1.1 TLSv1.2;
        ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
        ssl_prefer_server_ciphers on;
        ssl_session_cache shared:SSL:10m;

        client_max_body_size        0;
        chunked_transfer_encoding   on;

        index index.php index.html index.htm;

        root /var/www/html;


        location / {
                try_files $uri $uri/ /index.php$is_args$args;
        }


        location ~ \.php$ {
                  try_files $uri =404;
                  fastcgi_split_path_info ^(.+\.php)(/.+)$;
                  fastcgi_pass wordpress:9000;
                  fastcgi_index index.php;

                  include fastcgi_params;
                  fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
                  fastcgi_param PATH_INFO $fastcgi_path_info;
        }

        location ~ /\.ht {
                  deny all;
        }

        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;
        }
}
```

Gambar 9. Script docker-compose.yml (kiri) dan nginx.conf (kanan)

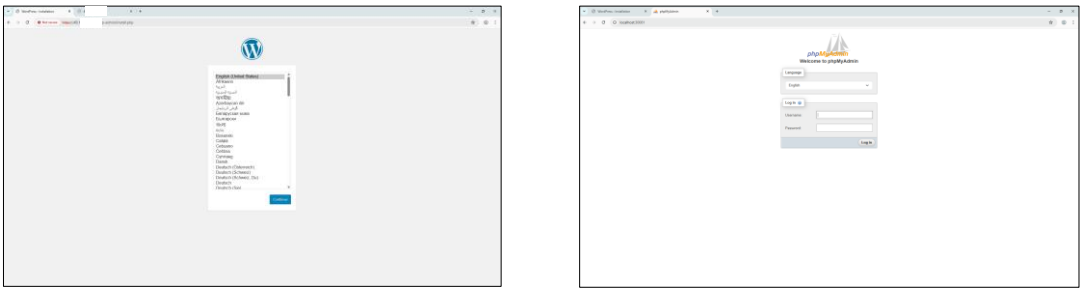
Untuk konfigurasi Nginx memiliki dua server block port 80 untuk redirect seluruh request ke HTTPS, dan port 443 untuk melayani WordPress dengan SSL yang mengambil sertifikat dari folder `/certs`. Server HTTPS menggunakan `client_max_body_size 0` untuk upload tanpa batas, root `/var/www/html`, serta `try_files…` untuk memastikan permalink WordPress berjalan dengan mengarahkan request non-statis ke `index.php`. Bagian utama adalah blok location `~ \.php$` yang meneruskan request PHP ke container WordPress melalui `fastcgi_pass wordpress:9000` dengan parameter FastCGI yang benar. Keamanan ditingkatkan dengan memblokir akses ke `.htaccess`.



```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case3$ sudo docker compose up -d
[sudo] password for hfwxyz:
WARN[0000] /home/hfwxyz/cc25/task-2/case3/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] up 56/56
 ✓ Image mysql:8.0 Pulled                                    66.8s
 ✓ Image phpmyadmin/phpmyadmin Pulled                        40.8s
 ✓ Image wordpress:5.1.1-fpm-alpine Pulled                   58.1s
 ✓ Network case3_app-network Created                          0.0s
 ✓ Container case3-db-1 Created                               0.2s
 ✓ Container case3-wordpress-1 Created                        0.0s
 ✓ Container case3-phpmyadmin-1 Created                       0.0s
 ✓ Container case3-webserver-1 Created                        0.0s
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case3$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED         STATUS         PORTS                                                                                  NAMES
13197ef4e864   nginx:1.15.12-alpine  "nginx -g 'daemon of…"   39 seconds ago  Up 38 seconds  0.0.0.0:80->80/tcp, [::]:80->80/tcp, 0.0.0.0:443->443/tcp, [::]:443->443/tcp          case3-webserver-1
50f22cf78f3f   phpmyadmin/phpmyadmin "/docker-entrypoint.…"   39 seconds ago  Up 38 seconds  0.0.0.0:30081->80/tcp, [::]:30081->80/tcp                                              case3-phpmyadmin-1
24929cd4ecad   wordpress:5.1.1-fpm-alpine "docker-entrypoint.s…" 39 seconds ago Up 38 seconds  9000/tcp                                                                              case3-wordpress-1
7583a3875078   mysql:8.0             "docker-entrypoint.s…"   39 seconds ago  Up 38 seconds  3306/tcp, 33060/tcp                                                                    case3-db-1
```

Gambar 10. Menjalankan container

Empat container berhasil berjalan sesuai dengan yang telah didefinisikan dalam `docker-compose.yml`. Untuk WordPress dapat diakses melalui port 80 dan untuk phpMyAdmin bisa diakses melalui port 30081.



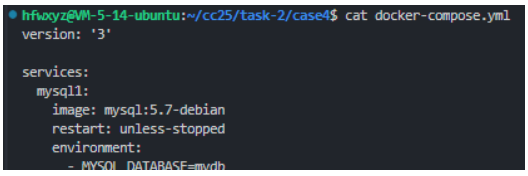
Gambar 11. Halaman WordPress (kiri) dan phpMyAdmin (kanan)

Pada gambar terlihat bahwa masing-masing layanan dapat diakses melalui port yang sesuai. Hal ini menunjukkan bahwa web server Nginx telah dikonfigurasi dengan benar untuk menampilkan halaman WordPress, serta phpMyAdmin dapat diakses dengan baik sehingga menandakan bahwa database MySQL berhasil terhubung dan dikelola.

- ***Case 4***

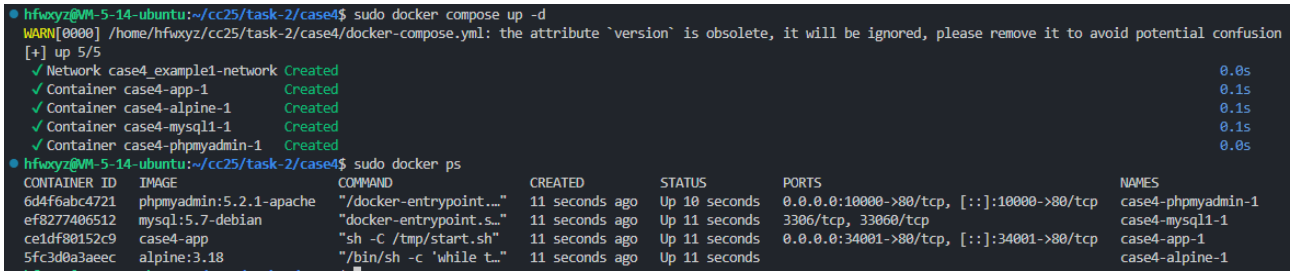
Pada studi kasus ini kita diminta untuk menyediakan aplikasi berbasis PHP yang berjalan bersama phpMyAdmin di dalam container terpisah.

Untuk konfigurasi Docker Compose versi 3 ini menjalankan empat service yang terhubung dalam `example1-network` bridge untuk mengisolasi komunikasi internal. Service `mysql1` menggunakan MySQL 5.7 Debian dengan database `mydb`, menyimpan data persisten di folder `dbdata` dan script SQL di `dbscripts`, dikonfigurasi dengan `MYSQL_ROOT_HOST=%` agar root bisa akses dari host manapun. Service `app` adalah container utama yang dibangun dari Dockerfile custom di folder platform, menjalankan web server yang dapat diakses via port 34001, dengan source code dimount dari `./src ke /var/www/localhost/htdocs/`. Service `alpine` adalah container Alpine Linux ringan yang dibuat tetap hidup dengan infinite loop sleep 10000 sebagai helper untuk debugging network seperti ping atau masuk console via docker exec. Service `phpmyadmin` menyediakan GUI manajemen database yang dapat diakses di port 10000, terhubung langsung ke `mysql1` via environment variable `PMA_HOST=mysql1`. Untuk script detailnya, bisa dilihat pada gambar di bawah ini.



```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case4$ cat docker-compose.yml
version: '3'

services:
  mysql1:
    image: mysql:5.7-debian
    restart: unless-stopped
    environment:
      - MYSQL_DATABASE=mydb
```

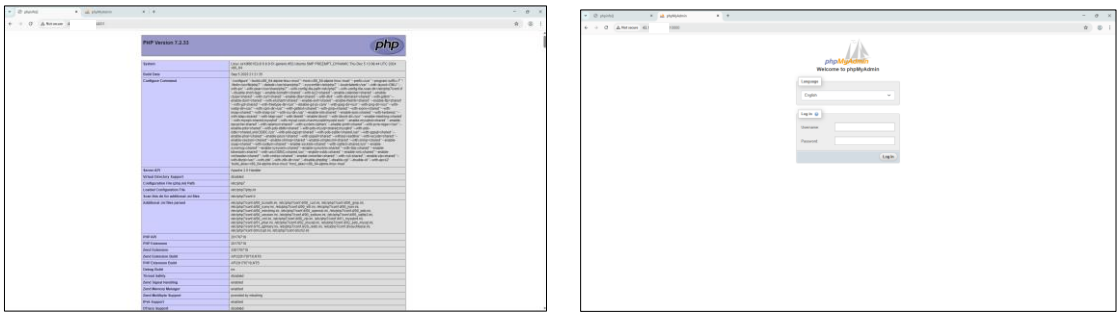
Gambar 12. Potongan script docker-compose.yml



```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case4$ sudo docker compose up -d
WARN[0000] /home/hfwxyz/cc25/task-2/case4/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] up 5/5
 ✓ Network case4_example1-network Created                                                                                                    0.0s
 ✓ Container case4-app-1          Created                                                                                                    0.1s
 ✓ Container case4-alpine-1       Created                                                                                                    0.1s
 ✓ Container case4-mysql1-1       Created                                                                                                    0.1s
 ✓ Container case4-phpmyadmin-1   Created                                                                                                    0.0s
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case4$ sudo docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                                          NAMES
6d4f6abc4721   phpmyadmin:5.2.1-apache  "/docker-entrypoint.…"   11 seconds ago   Up 10 seconds   0.0.0.0:10000->80/tcp, [::]:10000->80/tcp      case4-phpmyadmin-1
ef8277406512   mysql:5.7-debian         "docker-entrypoint.s…"   11 seconds ago   Up 11 seconds   3306/tcp, 33060/tcp                            case4-mysql1-1
ce1df80152c9   case4-app                "sh -C /tmp/start.sh"    11 seconds ago   Up 11 seconds   0.0.0.0:34001->80/tcp, [::]:34001->80/tcp      case4-app-1
5fc3d0a3aeec   alpine:3.18              "/bin/sh -c 'while t…"   11 seconds ago   Up 11 seconds                                                  case4-alpine-1
```

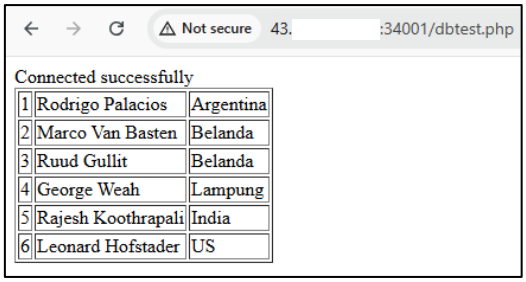
Gambar 13. Menjalankan container

Terlihat pada gambar bahwa empat container berhasil dijalankan. Untuk aplikasi PHP bisa diakses melalui port 34001 dan untuk phpMyAdmin bisa diakses melalui port 10000 untuk pengelolaan database MySQL.



Gambar 14. Halaman aplikasi PHP (kiri) dan phpMyAdmin (kanan)

Gambar di atas menunjukkan halaman aplikasi PHP yang diakses melalui `localhost:34001` setelah menjalankan studi kasus keempat, menampilkan layanan aplikasi beserta dependensinya. Selain itu, tampak juga antarmuka phpMyAdmin di `localhost:10000`, yang menegaskan bahwa database MySQL berhasil terhubung dan dapat dikelola dengan baik.

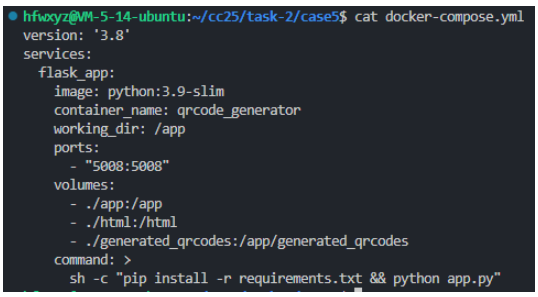


Gambar 15. Tes koneksi db

Apabila kita mencoba menjalankan command `docker compose exec -it mysql1 sh dbscripts/restoredb.sh`, maka akan memunculkan list data dari `./dbscripts/backup.sql` dan hasilnya bisa dilihat pada `localhost:34001/dbtest.php`.

- ***Case 5***

Pada studi kasus eksplorasi ini, saya mengembangkan aplikasi QR Code Generator berbasis Flask yang dijalankan menggunakan Docker melalui file `docker-compose.yml`. Antarmukanya disediakan oleh `html/index.html` untuk menerima input teks atau URL dari pengguna, lalu diproses oleh Flask melalui `app/app.py`, sementara `requirements.txt` berisi seluruh dependensi Python yang diperlukan.



Gambar 16. Script docker-compose.yml

Docker Compose digunakan untuk menjalankan sebuah aplikasi Flask dalam container. Service `flaskapp` menggunakan image `python:3.9-slim`, kemudian container diberi nama `qrcode_generator`. Terdapat tiga volume mounting, yaitu `app/` berisi source code aplikasi, `html/` untuk file HTML statis, serta folder `generated_qrcodes` untuk menyimpan hasil QR code yang

dihasilkan oleh program. Ketika container dijalankan, perintah `pip install -r requirements.txt` digunakan untuk menginstal dependensi Python secara otomatis, dan dilanjutkan dengan menjalankan aplikasi melalui `python app.py`.

```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case5$ cat app/app.py
from flask import Flask, request, jsonify, Response, send_file, send_from_directory
import qrcode
import os
from io import BytesIO
import logging
import time

app = Flask(__name__)

UPLOAD_FOLDER = os.path.expanduser("/app/generated_qrcodes")
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

logging.basicConfig(level=logging.DEBUG)

@app.route('/generate', methods=['POST'])
def generate_qrcode():
    try:
        data = request.json.get("data")
        if not data:
            app.logger.error("No data provided in request")
            return jsonify({"error": "No data provided"}), 400

        qr = qrcode.make(data)
        buffer = BytesIO()
        qr.save(buffer, format="PNG")
        buffer.seek(0)

        timestamp = int(time.time())
        qr_code_filename = f"{timestamp}.png"
        file_path = os.path.join(UPLOAD_FOLDER, qr_code_filename)

        app.logger.info(f"Saving QR code to {file_path}")

        with open(file_path, 'wb') as f:
            f.write(buffer.getvalue())

        app.logger.info(f"QR Code saved successfully: {file_path}")

        return jsonify({"message": "QR Code generated and saved", "file": qr_code_filename})

    except Exception as e:
        app.logger.error(f"Error generating QR Code: {e}")
        return jsonify({"error": "Internal Server Error"}), 500

@app.route('/')
def home():
   return send_from_directory('/html', 'index.html')

@app.route('/qrcode/<filename>')
def get_qrcode(filename):
    try:
        file_path = os.path.join(UPLOAD_FOLDER, filename)
        if os.path.exists(file_path):
            return send_file(file_path, mimetype='image/png')
        else:
            return jsonify({"error": "File not found"}), 404
    except Exception as e:
        app.logger.error(f"Error serving QR Code: {e}")
        return jsonify({"error": "Internal Server Error"}), 500

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5008)
```

Gambar 17. Script app.py

Script ini berfungsi sebagai backend untuk menghasilkan dan menyajikan QR Code. Endpoint `/generate` menerima data melalui POST JSON, kemudian membuat QR Code menggunakan library `qrcode`, menyimpannya ke folder `/app/generated_qrcodes`, dan mengembalikan nama file hasil. Endpoint `/` digunakan untuk menampilkan halaman utama dari folder `/html` dan endpoint `/qrcode/` digunakan untuk mengunduh QR Code yang sudah tersimpan.

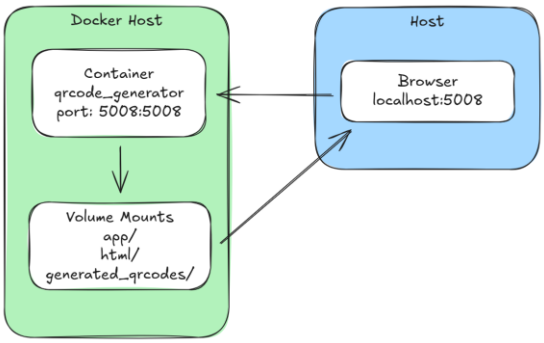
```
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case5$ sudo docker compose up -d
WARN[0000] /home/hfwxyz/cc25/task-2/case5/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] up 2/2
 ✓ Network case5_default       Created                                                                                                    0.1s
 ✓ Container qrcode_generator Created                                                                                                    0.0s
hfwxyz@VM-5-14-ubuntu:~/cc25/task-2/case5$ sudo docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS         PORTS                                       NAMES
c0b07872e724   python:3.9-slim  "sh -c 'pip install …"   7 seconds ago   Up 7 seconds   0.0.0.0:5008->5008/tcp, [::]:5008->5008/tcp   qrcode_generator
```

Gambar 18. Menjalankan container

Terlihat bahwa kontainer `qrcode_generator` berhasil dijalankan dengan `docker compose up -d` dan muncul di daftar `docker ps`. Untuk aksesnya bisa di `localhost:5008`.

Gambar 19. Halaman localhost:5008

Container berhasil berjalan, dapat diakses, serta mampu menghasilkan dan mengunduh QR Code dengan benar, yang menandakan bahwa konfigurasi orkestrasi menggunakan Docker Compose telah berhasil.



Gambar 20. Arsitektur Case 5

Untuk alurnya sendiri, pengguna membuka aplikasi melalui browser di `localhost:5008`, kemudian permintaan tersebut dikirim ke container `qrcode_generator` di Docker yang memproses input menggunakan Flask. Setelah QR code berhasil dibuat, file hasilnya disimpan melalui volume mounts ke direktori `app/`, `html/`, dan `generated_qrcodes/` di host, sehingga output dapat diakses dan tersimpan secara persisten.

Komputer atau VM yang digunakan memiliki spesifikasi 4 GB RAM, 2 vCPU, dan 60 GB storage, dengan IP Address 43.xxx.xxx.xxx atau localhost. Pada studi kasus ini, port yang digunakan adalah 5008 sebagai port akses aplikasi Flask di dalam container Docker, yang kemudian di forward ke host sehingga dapat dibuka melalui browser pada alamat localhost:5008.

Arsitektur containerized seperti pada QR Code Generator ini cocok diterapkan ketika aplikasi membutuhkan lingkungan yang konsisten di berbagai sistem, skalabilitas yang mudah saat beban meningkat, serta integrasi yang fleksibel dalam arsitektur microservices. Pendekatan ini juga ideal untuk deployment di berbagai environment karena konfigurasi dapat diatur melalui environment variables dan volume, sekaligus memudahkan kontrol resource serta monitoring tiap service. Selain itu, penggunaan Docker Compose mempermudah proses CI/CD dan onboarding tim yang terdistribusi, karena seluruh environment dapat dijalankan secara instan hanya dengan satu perintah.

Kesimpulan

Dalam laporan ini, saya berhasil menjalankan empat studi kasus dari repository yang diberikan serta mengembangkan Case 5, yaitu layanan Flask untuk generator QR Code menggunakan Docker Compose. Seluruh script, diagram alur, dan screenshot yang disertakan menunjukkan bagaimana setiap layanan dikonfigurasi dan dijalankan dengan benar. Penggunaan Docker Compose mempermudah pengelolaan container dan integrasi antarkomponen, sekaligus memungkinkan pengembangan lebih lanjut seperti pada Case 5 yang memanfaatkan Flask untuk menghasilkan QR Code secara dinamis dan dapat diakses melalui browser.