

Tugas Akhir

“Menerapkan Aplikasi Web PHP dengan Docker Compose, Nginx, dan MariaDB”

Dosen Pengampu : R. Dimas Adityo, MT



Disusun Oleh :

Rayhan Gimnastiar (2314311015)

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BHAYANGKARA SURABAYA**

2024

KATA PENGANTAR

Puji syukur saya ucapkan kehadiran Allah SWT yang telah memberikan rahmat serta hidayah kepada kita semua, sehingga berkat Karunia-Nya penulis dapat menyelesaikan makalah ini guna memenuhi Tugas Akhir UAS untuk mata kuliah Jaringan Komputer, dengan judul *“Instalasi Mesin Virtualisasi Docker dan Web Ecommerce”*.

Saya menyadari bahwa dalam penulisan makalah ini tidak terlepas dari bantuan banyak pihak yang dengan tulus memberikan doa, saran dan kritik sehingga makalah ini dapat terselesaikan.

Saya menyadari sepenuhnya bahwa makalah ini masih jauh dari sempurna dikarenakan terbatasnya pengetahuan yang saya miliki. Oleh karena itu, kami mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Dalam penyusunan makalah ini penulis berharap semoga makalah ini dapat bermanfaat bagi penulis sendiri maupun kepada pembaca umumnya.

Penyusun

Rayhan Gimnatiar

Daftar Isi

<i>PROGRAM STUDI TEKNIK INFORMATIKA FAKULTAS TEKNIK</i>	1
KATA PENGANTAR	2
<i>BAB I</i>	4
<i>PENDAHULUAN</i>	4
<i>1.1 Latar Belakang</i>	4
<i>1.2 Rumusan Masalah</i>	4
<i>1.3 Tujuan Masalah</i>	4
<i>BAB II</i>	5
<i>PEMBAHASAN</i>	5
<i>2.1. Instalasi Docker di Windows</i>	5
<i>2.2. Buat Kontainer Nginx</i>	5
<i>2.3. Membuat PHP Container</i>	6
<i>2.4. Buat kontainer MariaDB</i>	9
<i>BAB III PENUTUP</i>	11
<i>3.1. Kesimpulan</i>	11
<i>3.2. Saran</i>	11

BAB I

PENDAHULUAN

1.1 Latar Belakang

Aplikasi berbasis web merupakan aplikasi yang mempermudah seseorang untuk mencari informasi, bertransaksi, atau melakukan hal lainnya melalui layanan internet. Aplikasi berbasis web sangat mudah untuk digunakan karena dapat diakses di berbagai platform computer hanya dengan menjalankan web browser.

Dengan kemajuan pengetahuan dalam ilmu komputer, berbagai macam bahasa pemrograman pun telah banyak dibuat untuk menjalankan sebuah aplikasi berbasis web. Bahasa pemrograman yang digunakan antara lain yaitu *php*.

Dengan perkembangan teknologi yang begitu pesat ini, pembuatan aplikasi berbasis web pun ikut mengalami perkembangan dalam hal arsitektur. Arsitektur yang digunakan dalam pembuatan web bernama *Microservice*. *Microservice* adalah sebuah arsitektur dimana aplikasi dipecah-pecah menjadi service-service kecil dengan menggunakan bahasa pemrograman yang sama atau berbeda. Tujuan dari *Microservice* adalah untuk membuat sebuah aplikasi yang mampu berjalan secara independent.

Docker sangat mendukung untuk pembuatan *Microservice*, dengan menggunakan *docker* pembuatan aplikasi dan memasukkannya menjadi sebuah image yang nanti dijalankan menjadi sebuah *container*. Adanya *Microservice* membuat aplikasi *website* lebih mudah karena dapat membagi service menjadi kecil-kecil. Akan tetapi dengan adanya hal itu akan mempengaruhi *server* yang bekerja sebagai penyedia layanan dari aplikasi web tersebut serta bahasa pemrograman yang digunakan dalam pembuatan *website* akan mempengaruhi juga.

Cara untuk melalui proses penerapan aplikasi web PHP menggunakan Docker Compose, Nginx sebagai server web, dan MariaDB sebagai database. Docker Compose menyederhanakan pengelolaan aplikasi multi-kontainer, membuatnya mudah untuk mengatur dan menjalankan aplikasi web PHP Anda dengan semua dependensinya..

1.2 Rumusan Masalah

1. Bagaimana cara instalasi *Docker* di Windows?
2. Bagaimana cara menjalankan *Container* di *Docker* ?
3. Bagaimana cara menggabungkan *Container Docker*?

1.3 Tujuan Masalah

1. Untuk mengetahui cara instalasi Docker di windows
2. Untuk mengetahui bagaimana menjalankan *container* di *docker*.
3. Untuk menggabungkan *Container docker*

BAB II PEMBAHASAN

2.1. Instalasi Docker di Windows

1. Unduh penginstal menggunakan tombol unduh link berikut [Instal Docker Desktop di Windows | Dokumen Docker](#)
2. Klik dua kali Docker Desktop Installer.exe untuk menjalankan penginstal. Secara default, Docker Desktop diinstal pada C:\Program Files\Docker\Docker.
3. Saat diminta, pastikan opsi Gunakan WSL 2 alih-alih Hyper-V pada halaman Konfigurasi dipilih atau tidak tergantung pada pilihan backend Anda.
4. Jika sistem Anda hanya mendukung salah satu dari dua opsi tersebut, Anda tidak akan dapat memilih backend mana yang akan digunakan.
5. Ikuti petunjuk pada wizard penginstalan untuk mengotorisasi penginstal dan melanjutkan penginstalan.
6. Jika instalasi berhasil, pilih Close untuk menyelesaikan proses instalasi.:

2.2. Buat Kontainer Nginx

Sebelum memulai, kita akan menyiapkan container Nginx untuk menghosting aplikasi PHP kita. Ikuti langkah ini:

1. Buat direktori untuk proyek dan navigasikannya:

```
mkdir ~/docker-project  
cd ~/docker-project
```

2. Buat docker-compose.yml file untuk meluncurkan container Nginx:

```
nano docker-compose. yml
```

Misal di laptop tidak ada nano bisa menggunakan Visul Studio Code atau aplikasi lainnya yang sama.

3. Tambahkan konfigurasi berikut ke docker-compose.yml file:

```
version: "3.9"  
services:  
  nginx:  
    image: nginx:latest  
    container_name: nginx-container  
    ports:  
      - 80:80
```

Konfigurasi ini memastikan bahwa container Nginx berjalan pada port 80. Simpan dan tutup.

4. Luncurkan Container Nginx:

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

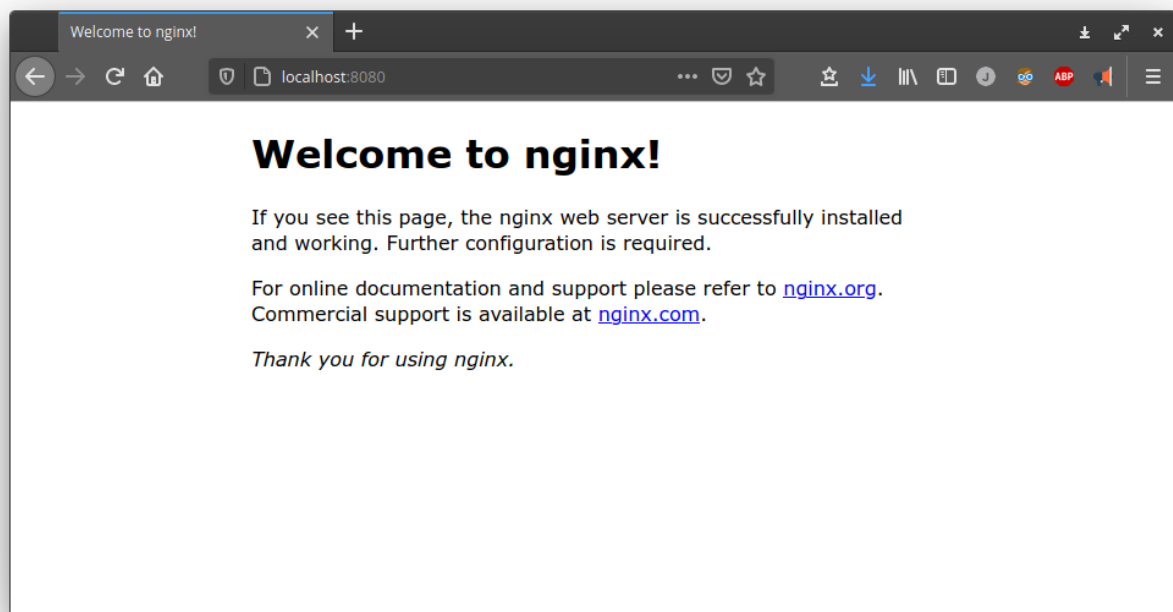
5. Verifikasi bahwa container Nginx sedang berjalan:

```
docker ps
```

melihat Output yang mirip dengan ini:

```
C:\Users\RayG\my-app>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
8fea7fa100e6   nginx:latest   "/docker-entrypoint..." 2 hours ago   Up 57 seconds  0.0.0.0:8081->80/tcp     my-app-nginx-1
```

6. Buka browser web Anda dan akses container Nginx Anda menggunakan URL `http://your-server-ip`. Anda akan melihat halaman pengujian Nginx.



2.3. Membuat PHP Container

Pada langkah ini, kita akan menyiapkan container PHP untuk menghosting aplikasi PHP kita. Ikuti langkah ini:

1. Buat direktori untuk kode PHP Anda di dalam proyek:

```
mkdir ~/docker-project/php_code
```

2. Kloning kode PHP Anda ke dalam php_codedirektori. Ganti [GitHub URL] dengan URL sebenarnya dari kode PHP:

```
git clone php-mysql-crud/db.php at master · FaztWeb/php-mysql-crud \(github.com\)/ docker-project/php_code/
```

3. Buat Dockerfile untuk container PHP:

```
nano ~ /docker-project/ php_code/ Dockerfile
```

4. Tambahkan baris berikut ke Dockerfile:

```
FROM php:7.0-fpm
RUN docker-php-ext-install mysqli pdo pdo_mysql
RUN docker-php-ext-enable mysqli
```

Bisa diganti dengan versi terbaru. Simpan dan tutup file.

5. Buat direktori untuk Nginx di dalam direktori proyek Anda:

```
mkdir ~/docker-project/nginx
```

6. Buat file konfigurasi default Nginx untuk menjalankan aplikasi PHP Anda:

```
nano ~ /docker-project/ nginx/ default . konf
```

7. Tambahkan konfigurasi Nginx berikut ke default.conf file:

```
server {

    listen 80 default_server;
    root /var/www/html;
    index index.html index.php;

    charset utf-8;

    location / {
        try_files $uri $uri /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt { access_log off; log_not_found off; }

    access_log off;
    error_log /var/log/nginx/error.log error;

    sendfile off;

    client_max_body_size 100m;
```

```
location ~ .php$ {
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
```

```

fastcgi_pass php:9000;
fastcgi_index index.php;
include fastcgi_params;
fastcgi_read_timeout 300;
fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
fastcgi_intercept_errors off;
fastcgi_buffer_size 16k;
fastcgi_buffers 4 16k;
}

location ~ /\.ht {
    deny all;
}
}

```

Simpan dan tutup file.

8. Buat Dockerfile di dalam nginxdirektori untuk menyalin file konfigurasi default Nginx:

```
nano ~ /docker-project/ nginx/ Dockerfile
```

9. Tambahkan baris berikut ke Dockerfile:

```
FROM nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

10. Update docker-compose.ymlfile dengan isi sebagai berikut:

```

version: "3.9"
services:
  nginx:
    build: ./nginx/
    ports:
      - 80:80

  volumes:
    - ./php_code/:/var/www/html/

  php:
    build: ./php_code/
    expose:
      - 9000
    volumes:
      - ./php_code/:/var/www/html/

```

11. Luncurkan kontainer:


```
cd ~/docker-project
docker-compose up -d
```

Buka browser web dan akses URL `http://your-server-ipatau localhost`. sekarang akan melihat konten web PHP.

2.4. Buat kontainer MariaDB

Pada langkah terakhir ini, kita akan menyiapkan wadah database MariaDB dan mengonfigurasinya agar berfungsi dengan aplikasi PHP kita.

Ikuti langkah ini:

1. Edit `docker-compose.yml` file untuk menambahkan entry untuk kontainer MariaDB:

```
nano ~ /docker-project/ docker-compose. Yml
```

2. Perbarui `docker-compose.yml` file dengan konfigurasi MariaDB:

```
version: "3.9"
services:
  nginx:
    build: ./nginx/
    ports:
      - 80:80

    volumes:
      - ./php_code:/var/www/html/

  php:
    build: ./php_code/
    expose:
      - 9000
    volumes:
      - ./php_code:/var/www/html/

  db:
    image: mariadb
    volumes:
      - mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: mariadb
      MYSQL_DATABASE: ecomdb

volumes:
  mysql-data:
```

Jalankan perintah berikut:

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

3. Buat sesi CLI di dalam container MariaDB:

```
docker exec -it [ id atau nama wadah db ] /bin/sh
```

4. Akses MariaDB sebagai pengguna root:

```
mariadb -u root -pmariadb
```

5. Buat pengguna baru untuk database:

```
CREATE USER 'rapidcode'@'%' IDENTIFIED BY "rapidcode123";
```

6. Berikan semua hak istimewa kepada pengguna baru:


```
GRANT ALL PRIVILEGES ON *.* TO 'rapidcode'@'%';  
FLUSH PRIVILEGES;
```

7. Keluar dari wadah MariaDB:

```
Exit
```

8. Pastikan index.phpfile dalam kode PHP Anda sudah dikonfigurasi dengan benar dengan nama pengguna dan kata sandi yang telah kita buat di atas untuk terhubung ke database MariaDB
9. Periksa kembali URL-nya dan *refresh*. Anda sekarang akan melihat aplikasi web PHP Anda mengambil data dari database MariaDB.

PHP MySQL CRUD

Title	Description	Created At	Action
To do my home	test	2018-12-31 16:56:35	 

BAB III

PENUTUP

3.1. Kesimpulan

Dalam hal ini kita telah berhasil membuat aplikasi web PHP menggunakan Docker Compose, Nginx sebagai server web, dan MariaDB sebagai database. Prosesnya melibatkan langkah-langkah berikut:

1. Persiapan struktur proyek dengan direktori untuk Docker Compose, Nginx, PHP, dan file-file aplikasi PHP.
2. Konfigurasi Docker Compose untuk mendefinisikan layanan PHP-FPM, Nginx, dan MariaDB, serta jaringan yang diperlukan.
3. Konfigurasi Nginx untuk mengarahkan permintaan ke server PHP-FPM.
4. Pembuatan skrip PHP sederhana untuk menguji aplikasi.

Setelah itu, aplikasi dapat dijalankan dengan perintah `docker-compose up`, dan kita dapat mengaksesnya melalui <http://localhost:80:80>.

3.2. Saran

Bagi pembaca makalah ini, semoga makalah ini dapat menjadi bermanfaat bagi semua. Itulah langkah-langkah penerapan aplikasi web PHP menggunakan Docker Compose, Nginx, dan MariaDB. Dengan menggunakan Docker Compose, pengelolaan aplikasi multi-kontainer menjadi lebih mudah dan terstruktur. Sesuaikan konfigurasi Nginx, PHP, dan MariaDB sesuai kebutuhan proyek Anda. Misalnya, tambahkan ekstensi PHP tertentu atau atur pengaturan Nginx yang lebih kompleks. Kepada generasi muda untuk lebih bijak dalam penggunaan perkembangan teknologi saat ini.