

**IMPLEMENTASI *DOCKER* PADA *WEBSITE* TO-DO-LIST
BERBASIS *LARAVEL***



Disusun Oleh :

NAMA : DAFAVICO ASSECHAN

NIM : 32602200050

**PRODI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM SULTAN AGUNG**

2025

DAFTAR ISI

HALAMAN JUDUL	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR.....	iii
PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Tujuan Penulisan	1
PEMBAHASAN	2
2.1 <i>Laravel</i>	2
2.2 <i>Docker</i>	2
2.3 <i>Docker Compose</i>	2
IMPLEMENTASI.....	3
3.1 Struktur <i>Project</i>	3
3.2 <i>Dockerfile</i>	3
3.3 Docker-compose.yml.....	3
3.4 Konfigurasi <i>Environment</i> (.env).....	3
HASIL DAN PEMBAHASAN	4
4.1 Proses Eksekusi	4
4.2 Tampilan Aplikasi	4
PENUTUP.....	5
5.1 Kesimpulan.....	5
REFERENSI.....	
LAMPIRAN.....	

DAFTAR GAMBAR

Gambar 3. 1 Struktur Folder <i>Project</i>	3
Gambar 4. 1 Daftar <i>Container</i> Aktif	4
Gambar 4. 2 Tampilan Halaman Utama <i>Website</i> To-Do-List.....	4

PENDAHULUAN

1.1 Latar Belakang

Seiring dengan berkembangnya teknologi informasi, kebutuhan akan proses pengembangan perangkat lunak yang cepat, efisien, dan portabel semakin meningkat. *Laravel* sebagai salah satu *framework* PHP yang populer menawarkan berbagai kemudahan dalam membangun aplikasi *web* modern. Namun demikian, mengatur *environment Laravel* secara manual di setiap perangkat dapat menjadi tantangan tersendiri, terutama ketika harus mengatur dependensi seperti PHP, *Composer*, MySQL, dan Nginx secara terpisah.

Docker hadir sebagai solusi atas permasalahan tersebut. Dengan teknologi *containerization*, *Docker* memungkinkan *developer* untuk mengemas seluruh *environment* aplikasi ke dalam *container* yang ringan dan terisolasi. Hal ini memungkinkan *project Laravel* dijalankan secara konsisten di berbagai sistem tanpa perlu pengaturan ulang *environment*.

Dalam laporan ini, penulis melakukan implementasi *Docker* pada sebuah aplikasi *Laravel* berbasis *website* To-Do-List. Aplikasi tersebut berasal dari *repository* pihak ketiga yang kemudian *dideploy* menggunakan *Docker multi-container* dengan konfigurasi *Laravel*, MySQL, dan Nginx. Fokus laporan ini bukan pada pengembangan fitur *Laravel*, melainkan pada proses integrasi dan *deployment Laravel* ke dalam lingkungan *container* menggunakan *Docker* dan *Docker Compose*.

1.2 Tujuan Penulisan

Tujuan dari penulisan makalah ini adalah sebagai berikut:

1. Menjelaskan proses implementasi *Docker* pada *project Laravel*.
2. Menyajikan tahapan konfigurasi *Dockerfile* dan *docker-compose.yml* pada *project Laravel* yang sudah ada.
3. Menunjukkan keberhasilan *deployment* aplikasi *Laravel* ke dalam *environment Docker multi-container*

PEMBAHASAN

2.1 *Laravel*

Laravel adalah *framework* PHP berbasis *Model-View-Controller* (MVC) yang dirancang untuk mempermudah pengembangan aplikasi *web* modern. *Laravel* menyediakan fitur-fitur penting seperti *routing*, *middleware*, Eloquent ORM, *migration*, serta *Artisan CLI* yang membantu *developer* mempercepat proses pembuatan dan pengelolaan aplikasi *web*.

2.2 *Docker*

Docker adalah *platform open-source* yang digunakan untuk membangun, mendistribusikan, dan menjalankan aplikasi di dalam *container*. *Container* merupakan unit terisolasi yang berisi seluruh komponen aplikasi beserta dependensinya, sehingga menjamin aplikasi dapat dijalankan secara konsisten di berbagai *environment* tanpa perlu instalasi manual.

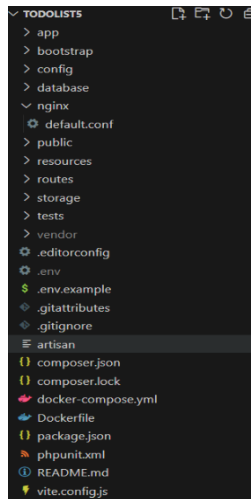
2.3 *Docker Compose*

Docker Compose adalah *tool* tambahan dari *Docker* yang memungkinkan *developer* menjalankan beberapa *container* sekaligus melalui satu *file* konfigurasi bernama *Docker-compose.yml*. Dengan *Docker Compose*, *developer* dapat mendefinisikan bagaimana *container Laravel*, *MySQL*, dan *Nginx* saling terhubung dan berjalan secara otomatis hanya dengan satu perintah.

IMPLEMENTASI

3.1 Struktur *Project*

Project Laravel To-Do-List ini di-clone dari *repository* pihak ketiga dan kemudian dikonfigurasi ke dalam *environment Docker*.



Gambar 3. 1 Struktur Folder *Project*

3.2 *Dockerfile*

Dockerfile berfungsi untuk membangun *image Laravel* berbasis PHP-FPM. *Dockerfile* ini bertanggung jawab untuk menyiapkan dependency PHP, *Composer*, dan mengatur *permission* folder aplikasi..

3.3 *Docker-compose.yml*

File *Docker-compose.yml* berfungsi untuk mengatur dan menjalankan beberapa *container* sekaligus dalam satu perintah. Terdapat 3 *service* utama:

1. *Laravel App* sebagai *backend* PHP-FPM,
2. *MySQL* sebagai *database server*,
3. *Nginx* sebagai *web server* yang berperan menerima *request* dari *user*.

3.4 Konfigurasi *Environment (.env)*

File *.env* pada *Laravel* disesuaikan agar dapat terhubung dengan *database* yang berjalan di dalam *container* *MySQL*. Pengaturan ini meliputi *host database*, *port*, nama *database*, *user*, dan *password* sesuai konfigurasi di *Docker-compose.yml*.

HASIL DAN PEMBAHASAN

4.1 Proses Eksekusi

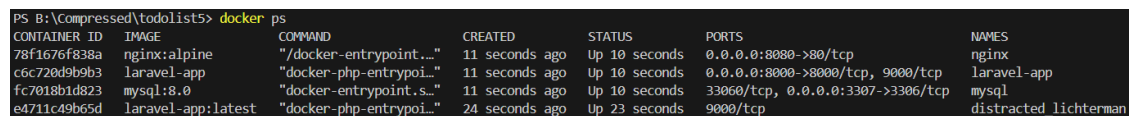
Setelah konfigurasi *Docker* selesai, jalankan:

```
Docker-compose up -d
```

Perintah ini memanggil *Dockerfile* dan *Docker-compose.yml* untuk membuat dan menjalankan 3 *service*: *Laravel App*, *MySQL*, dan *Nginx*. Cek status container dengan *Status container* dapat diperiksa dengan perintah:

```
Docker ps
```

Hasilnya menampilkan daftar container aktif lengkap dengan nama, status, dan *port mapping*.



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
78f1676f838a	nginx:alpine	"/docker-entrypoint..."	11 seconds ago	Up 10 seconds	0.0.0.0:8080->80/tcp	nginx
c6c720d9b9b3	laravel-app	"docker-php-entrypoi..."	11 seconds ago	Up 10 seconds	0.0.0.0:8000->8000/tcp, 9000/tcp	laravel-app
fc7018b1d823	mysql:8.0	"docker-entrypoint.s..."	11 seconds ago	Up 10 seconds	33060/tcp, 0.0.0.0:3307->3306/tcp	mysql
e4711c49b65d	laravel-app:latest	"docker-php-entrypoi..."	24 seconds ago	Up 23 seconds	9000/tcp	distracted_lichterman

Gambar 4. 1 Daftar *Container* Aktif

Setelah itu, untuk melakukan migrasi *database*, dijalankan perintah:

```
Docker exec -it Laravel-app php artisan migrate
```

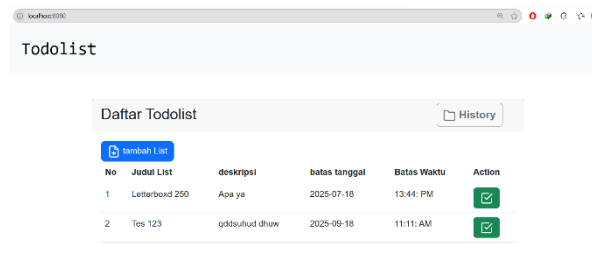
Proses ini memastikan semua tabel *Laravel* berhasil dibuat di dalam container *MySQL*.

4.2 Tampilan Aplikasi

Setelah *container* berjalan dan migrasi sukses, aplikasi *Laravel* dapat diakses melalui *browser* pada alamat:

```
http://localhost:8080
```

Halaman utama *website To-Do-List* akan tampil, menandakan *deployment* berhasil dan *Laravel* berjalan normal di *Docker*.



Gambar 4. 2 Tampilan Halaman Utama *Website To-Do-List*

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi yang dilakukan, dapat disimpulkan bahwa penggunaan *Docker* pada proyek website To-Do-List berbasis *Laravel* memberikan banyak keuntungan. Dengan teknologi *containerization*, seluruh komponen seperti *Laravel* App, MySQL, dan Nginx dapat dikemas dalam satu lingkungan yang terisolasi dan portabel.

Proses *setup* yang biasanya rumit dapat disederhanakan dengan *Docker-compose*, sehingga mempermudah pengembang untuk menjalankan aplikasi tanpa harus mengatur *environment* secara *manual*. Selain itu, implementasi ini juga mendukung kolaborasi tim karena *project* dapat dijalankan dengan konfigurasi yang sama di berbagai perangkat. Dengan demikian, *Docker* terbukti menjadi solusi praktis untuk membangun, menjalankan, dan mendistribusikan aplikasi *Laravel* secara konsisten dan efisien.

REFERENSI

- [1] Dhevan M. Anthareza. *Pengenalan Teknologi Kontainer Docker*. Universitas Dian Nuswantoro, 2019.
- [2] Dokumentasi *Laravel*. *Laravel Documentation*. <https://Laravel.com/docs>
- [3] Dokumentasi *Docker*. *Docker Docs*. <https://docs.Docker.com>
- [4] Repository Asli *Website To-Do-List*. <https://github.com/Igprad01/To-Do-List-Website>
- [5] Repository Implementasi *Docker*
https://github.com/pockypiko/Tugas_Docker_Cloud-Computing-B_DafavicoAssechan

LAMPIRAN

Nama: Dafavico Assechan

NIM: 32602200050

Link Project Github:

https://github.com/pockypiko/Tugas_Docker_Cloud-Computing-B_DafavicoAssechan