Penjelasan Lengkap: Deployment Diagram EazZy Project

Dokumen ini memberikan penjelasan mendetail untuk setiap komponen dan alur kerja yang digambarkan dalam Deployment Diagram proyek Anda.

Konsep Dasar: Apa yang Sedang Kita Lihat?

Jika ERD adalah blueprint untuk **database**, maka Deployment Diagram adalah **peta infrastruktur** aplikasi Anda. Diagram ini tidak peduli dengan *logika kode*, tetapi ia peduli dengan **di mana setiap bagian dari perangkat lunak Anda "hidup" dan bagaimana mereka saling "berbicara"**.

Bayangkan ini adalah denah sebuah gedung perkantoran modern:

- **Debian Server (Host):** Adalah tanah dan bangunan utama.
- **Docker Engine:** Adalah sistem manajemen gedung yang mengatur listrik, keamanan, dan akses antar ruangan.
- **Container:** Adalah setiap ruangan kantor (misal: Ruang Resepsionis, Ruang Kerja Utama, Ruang Arsip). Setiap ruangan terisolasi tetapi bisa berkomunikasi melalui jalur yang sudah ditentukan.
- **Jaringan Virtual:** Adalah koridor dan jalur telepon internal yang aman.
- **Volume:** Adalah brankas atau lemari arsip di luar ruangan kantor untuk menyimpan dokumen-dokumen penting secara permanen.

Penjelasan Setiap Komponen Diagram

1. Debian Server (Host)

- Ini adalah server utama Anda, "komputer" yang menjalankan semuanya. Di sinilah Docker diinstal.
- **Host Filesystem:** Ini adalah direktori di dalam server Debian Anda.
    - [/opt/eazzy-docker]: Tempat Anda menyimpan semua file proyek, seperti docker-compose.yml, kode aplikasi, dan file konfigurasi Nginx.
    - [/var/lib/docker/volumes]: Tempat Docker secara default menyimpan data persisten (seperti data database dan Redis) yang tidak boleh hilang jika kontainer dihapus.

2. Docker Engine

- Ini adalah platform virtualisasi yang berjalan di atas Debian. Tugasnya adalah membuat, menjalankan, dan mengelola semua kontainer dan jaringan virtual berdasarkan instruksi dari file docker-compose.yml.

3. Container (Layanan)

- Setiap kotak di dalam Docker Engine adalah sebuah **kontainer**. Ini adalah lingkungan yang terisolasi penuh yang berisi satu layanan spesifik.
- **Container: reverse-proxy (Nginx):** Ini adalah "Resepsionis" atau "Pintu Gerbang" utama. Hanya dia yang bisa dihubungi dari luar (Internet). Tugasnya adalah menyambut semua "tamu" dan mengarahkan mereka ke ruangan yang tepat.
- **Container: main-app (PHP-FPM):** Ini adalah "Ruang Kerja Utama" tempat semua logika bisnis aplikasi Anda diproses.
- **Container: postgres-db & redis-cache:** Ini adalah "Ruang Arsip" (database) dan "Ruang Penyimpanan Cepat" (cache) yang sangat aman. Hanya "staf internal" (kontainer PHP) yang diizinkan masuk.
- **Container: grafana & prometheus:** Ini adalah "Ruang Kontrol & CCTV" untuk memantau kinerja dan kesehatan semua ruangan lain.

4. Jaringan Virtual (frontend-net & backend-net)

- Ini adalah inovasi paling penting dalam arsitektur keamanan Anda.
- **[frontend-net]:** Jaringan "Publik" atau "Lobi". Hanya Nginx (reverse-proxy) yang terhubung ke sini agar bisa menerima tamu dari luar.
- **[backend-net]:** Jaringan "Internal" yang sangat aman dan terisolasi. Semua komunikasi sensitif (antara PHP, database, dan Redis) terjadi di sini. Tidak ada seorang pun dari luar yang bisa langsung mengakses jaringan ini.

5. Volume & Mounting (Garis Putus-putus)

- Garis putus-putus (..>) yang menghubungkan Host Filesystem ke kontainer menunjukkan proses **mounting**.
- **mounts config & mounts code:** Artinya, file konfigurasi Nginx dan kode PHP Anda yang ada di server host "dipinjamkan" ke dalam kontainer. Keuntungannya, Anda bisa mengedit kode atau konfigurasi di host, dan perubahannya akan langsung terlihat di dalam kontainer tanpa perlu membangun ulang *image*.
- **mounts data volume:** Artinya, data di dalam kontainer database dan Redis sebenarnya disimpan di lokasi yang aman di server host (/var/lib/docker/volumes). Ini krusial, karena jika kontainer database rusak atau dihapus, datanya tetap aman di *volume*.

Alur Kerja Permintaan (Contoh)

Mari kita lacak sebuah permintaan dari pengunjung:

1. **Pengunjung** dari **Internet** mengakses https://eazzy-project.com.
2. Permintaan tersebut masuk ke **Debian Server** Anda pada port 443 (HTTPS).
3. **Docker Engine** langsung meneruskan (port-mapping) permintaan ini ke **Container: reverse-proxy (Nginx)** karena hanya dia yang "mendengarkan" di port tersebut.
4. **Nginx** menerima permintaan. Berdasarkan file konfigurasinya, ia tahu bahwa permintaan ini harus diproses oleh PHP.
5. Nginx kemudian meneruskan permintaan tersebut ke **Container: main-app (PHP-FPM)** melalui jaringan internal yang aman, **backend-net**.
6. **PHP-FPM** memproses logika. Misalnya, ia perlu mengambil data produk.

7. Aplikasi PHP kemudian mengirim *query* SQL ke **Container: postgres-db**, juga melalui **backend-net**.
8. Database mengembalikan data produk ke PHP.
9. PHP merender halaman HTML dan mengirimkannya kembali ke Nginx.
10. Nginx mengirimkan halaman HTML final kembali ke pengunjung di Internet.

Seluruh proses ini terjadi dengan sangat cepat, dan yang terpenting, semua komponen sensitif (PHP, database, Redis) terlindungi di dalam jaringan backend-net, tidak pernah terekspos langsung ke internet.
Ini adalah penjelasan dari arsitektur modern dan aman yang telah Anda rancang.