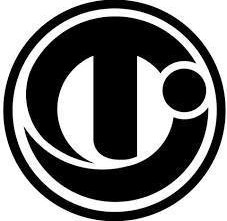
# Laporan Praktek Topik Khusus 6

Docker File



## SEMESTER VI

**Muhammad Abel Al-fahrezi 2211083034**

# PROGRAM STUDI D4 TEKNOLOGI REKAYASA PERANGKAT LUNAK

**JURUSAN TEKNOLOGI INFORMASI**

## Landasan Teori

* 1. **Windows Subsystem for Linux (WSL) :**

WSL adalah fitur dari sistem operasi Windows yang memungkinkan pengguna untuk menjalankan lingkungan GNU/Linux secara langsung di Windows, tanpa harus menggunakan mesin virtual (Virtual Machine) secara penuh. Versi terbaru, **WSL2**, menggunakan kernel Linux asli yang mendukung sistem file yang lebih cepat dan kompatibilitas lebih baik dengan tool berbasis Linux seperti Docker. Dengan WSL2, pengguna Windows dapat menjalankan container Docker **secara native**, tanpa harus bergantung pada teknologi virtualisasi berat seperti Hyper-V atau VirtualBox.

* 1. **Docker dan Container :**

Nginx Docker adalah platform open-source yang dirancang untuk mengembangkan, mengirimkan, dan menjalankan aplikasi dalam unit terisolasi yang disebut container. Container berisi semua dependensi aplikasi, seperti kode, pustaka, dan konfigurasi, sehingga menjamin konsistensi aplikasi saat dipindahkan antar lingkungan. Docker bekerja sangat baik di lingkungan Linux karena container dibangun di atas fitur kernel Linux seperti cgroups dan namespaces. Oleh karena itu, menjalankan Docker melalui WSL2 menciptakan kondisi ideal untuk penggunaan container di sistem operasi Windows.

* 1. **Dockerfile**

Dockerfile adalah sebuah file teks yang berfungsi sebagai skrip instruksi untuk membangun image Docker. File ini mendeskripsikan secara berurutan perintah-perintah yang harus dijalankan Docker untuk membentuk sebuah lingkungan aplikasi yang dapat dijalankan secara terisolasi dalam container. Instruksi-instruksi dalam Dockerfile mencakup pemilihan image dasar (FROM), penyalinan file dari sistem lokal ke dalam image (COPY atau ADD), pengaturan direktori kerja (WORKDIR), instalasi dependensi (RUN), deklarasi variabel lingkungan (ENV), hingga perintah yang akan dijalankan saat container aktif (CMD atau ENTRYPOINT). Dengan menggunakan Dockerfile, proses pembuatan dan pengaturan lingkungan aplikasi menjadi lebih otomatis, konsisten, dan portabel, karena image yang dihasilkan dapat dijalankan di berbagai sistem tanpa perlu konfigurasi ulang. Hal ini sangat berguna dalam pengembangan perangkat lunak modern, terutama pada pendekatan DevOps dan microservices, di mana konsistensi dan kecepatan deployment menjadi sangat penting.

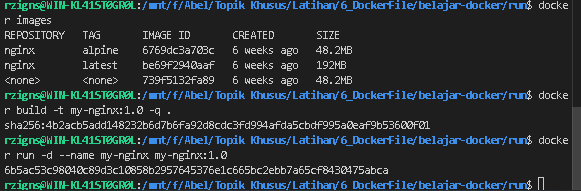
## Tools

* Ubuntu (melalui WSL2)
* VSCode

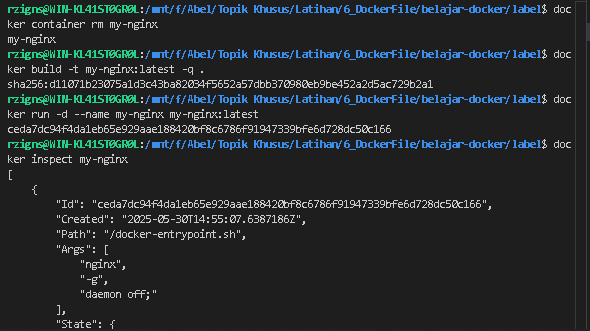
## Langkah Kerja

* 1. **RUN Instruction**
* Instruksi RUN digunakan untuk mengeksekusi perintah shell selama proses build image.
* Instruksi ini umumnya digunakan untuk menginstal paket atau melakukan modifikasi pada image selama proses build berlangsung.
* *Docker run -d –name my-nginx:1.0*

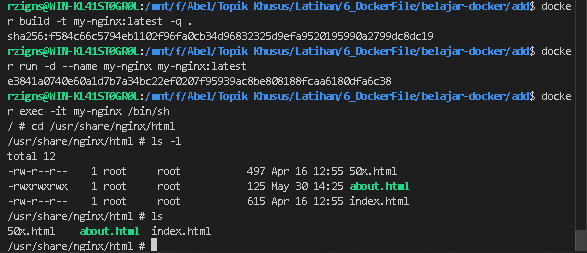
Untuk mengeksekusi dockerfile



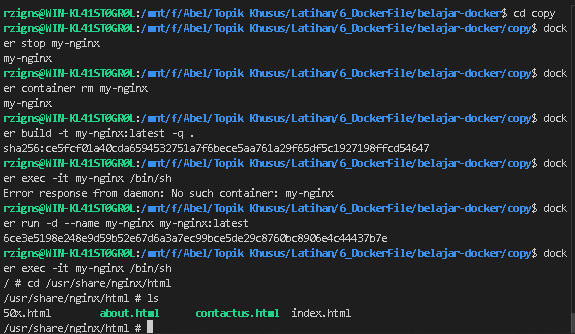
* 1. **Label Instruction**
* Instruksi LABEL digunakan untuk menambahkan metadata atau informasi tambahan ke dalam sebuah image.
* Label didefinisikan menggunakan format pasangan *key-value* (kunci-nilai).
* Label dapat berisi informasi seperti versi, pembuat (author), tujuan penggunaan, atau informasi khusus lainnya.
* melihat label dari sebuah image dengan perintah:
* *Docker inspect my-nginx*



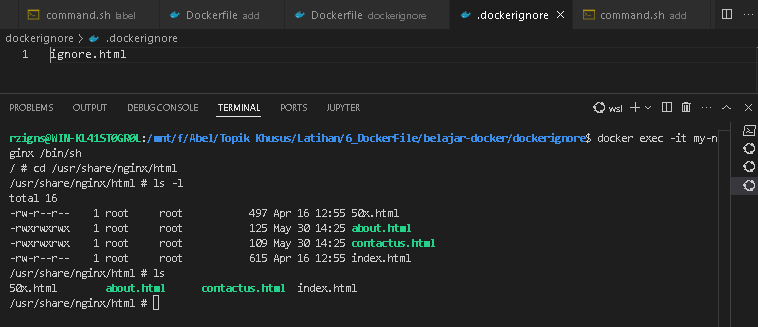
* 1. **Add Instruction**
* ADD digunakan untuk menyalin file, folder, atau URL dari jarak jauh ke dalam image Docker.
* Sintaksnya adalah: ADD <src> <dest>
* Jika file sumber berupa arsip terkompresi (seperti .tar, .gz), Docker akan mengekstraknya secara otomatis.



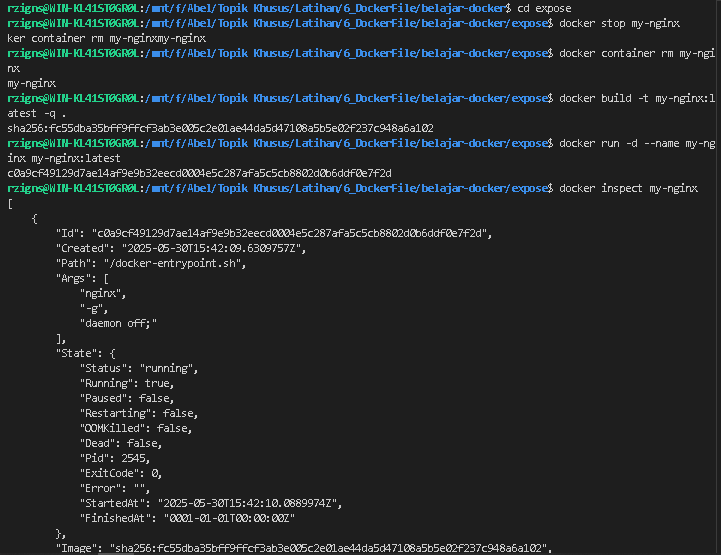
* 1. **COPY Instruction**
* COPY memiliki fungsi yang sama dengan ADD, yaitu menyalin file atau folder ke dalam image.
* Namun, berbeda dengan ADD, COPY tidak mendukung ekstraksi otomatis file arsip atau pengunduhan dari URL.
* COPY lebih direkomendasikan dibandingkan ADD karena lebih sederhana dan cenderung menghasilkan lebih sedikit kesalahan.



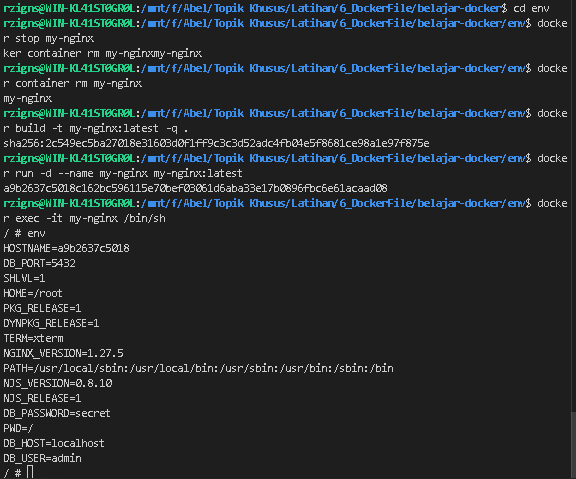
* 1. **.dockerignore File**
* dockerignore digunakan untuk mengecualikan file atau folder tertentu saat menggunakan perintah COPY atau ADD dalam Dockerfile.
* Fungsinya mirip dengan .gitignore: memberi tahu Docker file atau folder apa saja yang harus diabaikan saat membangun image.
* Hal ini membantu untuk:
  + Mengurangi ukuran image
  + Menghindari penyalinan file yang tidak diperlukan atau bersifat sensitif



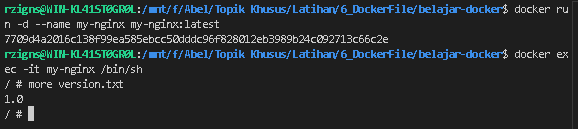
* 1. **EXPOSE Insturction**
* Instruksi EXPOSE digunakan untuk menunjukkan bahwa image akan menggunakan port tertentu saat dijalankan sebagai container.
* Kamu juga dapat menentukan protokol yang digunakan (TCP/UDP). Jika tidak disebutkan, Docker secara default akan menganggapnya sebagai TCP.
* Untuk memeriksa port yang diekspos dalam sebuah image, kamu dapat menggunakan perintah:
  + *Docker inspect my-nginx*



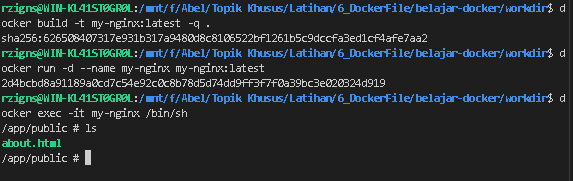
* 1. **ENV Instruction**
* Instruksi ENV digunakan untuk mendefinisikan variabel lingkungan (environment variables) yang akan tersedia saat container dijalankan.
* Variabel lingkungan digunakan untuk membuat konfigurasi aplikasi menjadi lebih dinamis.
* Kamu dapat menggunakan kembali variabel lingkungan dengan format: $(NAMA\_ENV)



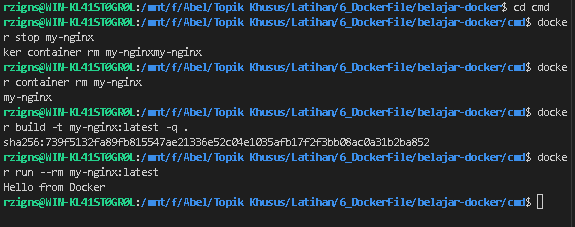
* 1. **ARG Instruction**
* ARG bekerja mirip dengan ENV, tetapi hanya tersedia selama proses build image.
* Gunakan ARG saat kamu perlu mengatur nilai statis yang tidak berubah saat container dijalankan (runtime).



* 1. **WORKDIR Instruction**
* WORKDIR (working directory) adalah folder kerja di dalam image.
* Jika folder tersebut belum ada, maka akan dibuat secara otomatis.
* WORKDIR dapat didefinisikan menggunakan path relatif maupun absolut.



* 1. **CMD Instruction**
* CMD mendefinisikan perintah default yang akan dijalankan saat sebuah container dimulai.
* Hanya satu perintah CMD yang diperbolehkan dalam sebuah Dockerfile — perintah terakhir akan menggantikan yang sebelumnya.
* CMD dapat didefinisikan dalam dua bentuk:
  + Bentuk Exec (direkomendasikan): CMD ["executable", "param1", "param2"]
  + Bentuk Shell: CMD command param1 param2



## Kesimpulan

Praktik pembuatan Dockerfile memberikan pemahaman yang mendalam tentang cara otomatisasi pembuatan image Docker yang konsisten dan portabel. Dengan menggunakan instruksi seperti FROM, COPY, RUN, WORKDIR, dan CMD, kita dapat membangun lingkungan aplikasi secara terstruktur dan dapat direproduksi di berbagai sistem tanpa perlu konfigurasi manual ulang. Proses build yang otomatis juga memudahkan instalasi dependensi dan pengaturan aplikasi secara efisien. Selain itu, pemahaman tentang instruksi tambahan seperti ENV, ARG, dan EXPOSE membantu mengelola konfigurasi dan komunikasi container dengan lebih baik. Dengan demikian, Dockerfile menjadi alat penting dalam pengembangan aplikasi modern, khususnya dalam praktik DevOps dan deployment berbasis container.