PRAKTIKUM DEVOPS ENGINEERING

TUGAS 1

Oleh:

Muh. Bayu Radithya Pradana D. | 235013

PROGRAM STUDI REKAYASA PERANGKAT LUNAK
UNIVERSITAS DIPA MAKASSAR
2026

Eksperimen 1 - Tanpa Persistence

1. File tersebut hilang.
2. Container tempat file tersebut terhapus dan volume-nya tidak tersimpan secara persisten di host/docker

Eksperimen 2 - Bind Mount

1. File tetap ada setelah container dihapus.
2. Container membaca file dari folder host yang sudah di-mount. Misalkan directory dimana file-file aplikasi terletak di `/home/$USER/.web` di-mount ke `/var/www/html` maka container tersebut akan membaca directory `~/.web` melalui `/var/www/html`.
3. File secara persisten tersimpan di sistem host, jadi jika terjadi hal yang tidak diinginkan kepada container tersebut developer dapat menghapus container tersebut dengan tanpa dampak yang signifikan.

Eksperimen 3 - Docker Volume

1. File masih ada.
2. Di Linux, volume yang dibuat melalu docker terletak di `/var/lib/docker/volumes/<volume_name>`, dimana `volume_name` adalah nama volume yang dibuat.
3. Ada beberapa perbedaan docker volume dan bind mount, diantaranya:
   - Volume docker lebih mudah di back-up dan migrate dibandingkan bind mount.
   - Volume dapat digunakan oleh beberapa container dengan lebih aman.
   - Volume baru dapat diisi terlebih dahulu oleh sebuah container/build.
4. Bind mount berguna saat developer ingin berbagi file konfigurasi dari sistem host dan container atau jika developer ingin berbagi source code atau build artifact diantara developer environment container dan host.

5. Docker volume dapat digunakan saat aplikasi membutuhkan operasi I/O dengan performa tinggi.
6. Berdasarkan jawaban pertanyaan sebelumnya, menggunakan docker volume lebih menguntungkan dibandingkan bind mount sebab dengan menggunakan docker volume developer dapat me-recover data dan maintenance lebih mudah di tahap produksi.
7. Sebenarnya docker volume dan bind mounting sudah cukup untuk mengatasi masalah persistensi data. Namun, menurut saya dapat ditingkatkan dengan mengimplementasikan "snapshot" untuk volume. Mirip dengan implementasi filesystem btrfs/zfs di Linux dan NixOS dimana volume dapat membuat snapshot dengan mekanisme CoW (copy-on-write), membuat salinan data baru saat terjadi perubahan daripada menimpa/overwrite data lama. Jadi, tiap volume merupakan container-nya sendiri. Beberapa keuntungan dari implementasi ini, yaitu:
   - Memungkinkan restorasi data dengan instan jika data tiba-tiba hilang dengan hanya satu perintah/command.
   - Developer data me-rollback perubahan yang terjadi pada sistem ke snapshot sebelumnya. Misalnya snapshot λ mengalami sebuah error, developer dapat me-undo kerusakan tersebut dengan me-rollback ke snapshot β.
   - Developer juga dapat mengisolasi lingkungan filesystem/ volume tanpa konten volume tersebut mempengaruhi sistem host secara satu arah.

   Secara konsep, ini merupakan gabungan dari filosofi Nix, dimana Implementasi ini dasarnya adalah salah satu bentuk containerisation, namun bedanya container ini ekslusif untuk penyimpanan data dan tidak berupa suatu sistem operasi yang utuh. Implementasi ini sebenarnya kurang praktis untuk penggunaan di dunia nyata, karena sistem ini lebih kompleks dibandingkan metode volume dan binding, dan bisa jadi performa implementasi ini lebih buruk dibandingkan data storage konvensional.