



MODUL 6

IMPLEMENTASI DOCKER SEBAGAI CONTAINER

6.1 Topik Pembahasan

1. *Platform open-source docker.*
2. *Layanan container.*
3. *Fungsi docker dan container.*
4. *Prinsip kerja docker dan container.*

6.2 Tujuan Praktikum

1. *Praktikan dapat mengenali dan memahami konsep dasar Docker dan Containerization.*
2. *Praktikan dapat memahami perbedaan dari Container dan Virtual Machine.*
3. *Praktikan dapat memahami prinsip kerja dan fungsi Docker.*
4. *Praktikan dapat mengoperasikan penggunaan Docker sebagai Container.*

6.3 Alat dan Bahan

1. Laptop

6.4 Dasar Teori

6.4.1 Docker



Gambar 1 Docker

Docker merupakan sebuah platform yang menyediakan kemampuan untuk mengemas dan menjalankan sebuah aplikasi dalam sebuah lingkungan terisolasi yang disebut dengan container. Dengan adanya isolasi dan keamanan yang memadai memungkinkan kamu untuk menjalankan banyak container di waktu yang bersamaan pada *host* tertentu.

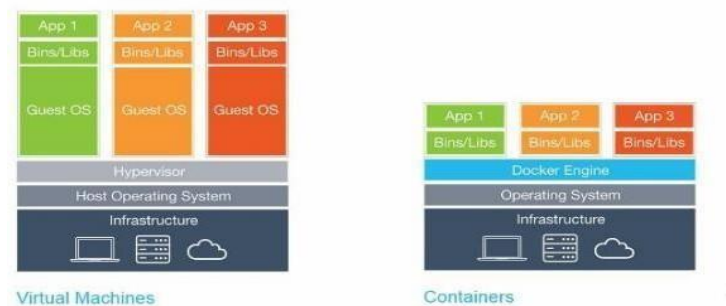
Docker diperkenalkan pada tahun 2013 oleh *Solomon Hykes* pada acara *PyCon*. Beberapa bulan setelahnya *docker* secara resmi diluncurkan, tepatnya pada tahun 2014. Semenjak itu *docker* menjadi sangat populer di kalangan *developer* luar negeri, tetapi belum terlalu populer di Indonesia. Sekarang, *docker* dapat dijalankan pada sistem operasi *MacOS*, *Windows*, dan *Linux*.

Kelebihan *docker* antara lain sistem container berbasis *open source*, mudah dibuat dan mendistribusikan *images*, dapat menjalankan *images* yang sama di setiap *container* yang telah dibuat, dan pemanfaatan sumber daya yang lebih baik. Kekurangannya, sulit mengelolah data persisten dalam sebuah wadah.

6.4.2 Container

Container merupakan sebuah istilah teknologi software yang mengemas dan mengisolasi aplikasi secara virtual untuk mempermudah software deployment. Berbeda dengan konsep tradisional *virtual machine*, *container* tidak membutuhkan *dedicated operating system* (OS Kernel) tetapi *container kernel* dapat di pergunakan secara bersamaan.

Container memungkinkan aplikasi dapat di pindah-pindahkan dengan cepat dan andal dari satu lingkungan komputasi ke lingkungan komputasi lainnya. *Container* merupakan paket *software* yang ringan, mandiri, dan dapat dieksekusi. Paket *container* mencakup semua yang diperlukan untuk menjalankan aplikasi: kode, runtime, alat sistem, pustaka sistem, dan pengaturan.



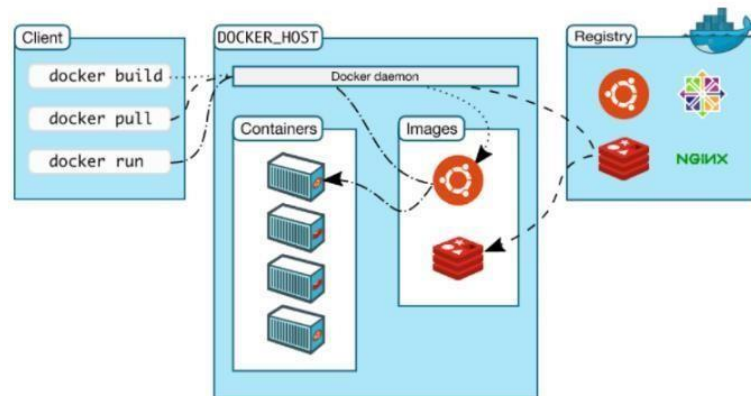
Gambar 2 Ilustrasi perbedaan Virtual Machine dan Contai

Pada VM, alokasi *resource* dilakukan pada awal instalasi sehingga ketika ada 2 atau lebih VM yang sudah ditentukan *resource*-nya dan salah satunya kehabisan *resource*, maka VM yang kehabisan *resource* tidak dapat mengambil *resource* dari VM lain. Jika di Container, alokasi *resource* dapat dilakukan oleh *host server* sehingga *host* dapat melakukan pengambilan *resource* pada *hardware* sesuai yang dibutuhkan container itu sendiri. Pada VM memiliki *Hypervisor* yang digunakan untuk menjalankan sistem di VM. Jika tidak ada *hypervisor*, maka VM tidak dapat berjalan. Di sisi lain, *container* dapat menjalankan program secara langsung di OS itu sendiri.

6.4.3 Fungsi Docker

1. Mempermudah Pengembangan Aplikasi. *Docker* bisa mempermudah pekerjaan developer ketika mengembangkan aplikasi. Alasannya, *Docker* lebih hemat *resource* dan mampu menyediakan *environment* yang stabil untuk dijalankan di perangkat apapun, mulai dari *cloud server* hingga komputer pribadi.
2. Menyederhanakan Konfigurasi. *Docker* tidak memiliki *overhead* sehingga *developer* bisa menjalankan aplikasi yang diuji tanpa konfigurasi tambahan.
3. Mendukung *Multitenancy*. *Docker* cocok digunakan untuk membuat aplikasi berstruktur *multitenancy* seperti *Software as a Service* (SaaS). Anda bisa membuat lebih dari satu *environment* yang terisolasi dan menjalankan objek aplikasi untuk setiap tenant.
4. Meningkatkan Sumber Daya dengan Cepat. Dengan *docker*, peningkatan sumber daya perangkat dapat dilakukan dengan cepat sehingga durasi pengembangan software lebih singkat.

6.4.4 Prinsip Kerja Docker



Gambar 3 Ilustrasi arsitektur client-server dalam prinsip kerja Docker

AWS Docker bertindak sebagai alat yang digunakan untuk menjalankan container. Container ini bertindak seperti virtual machine, yang seperti simulasi komputer yang berjalan di dalam komputer asli pengembang. Pada *virtual machine* ini, nantinya semua kode sistem tersimpan untuk menjalankan simulasi seolah-olah adalah operasi sistem utama. Docker bertindak melakukan virtualisasi sistem operasi di dalam sistem operasi *host*. Docker membangun *container* berdasarkan gambar yang berisi kode program. Gambar atau *images* ini ditumpuk satu sama lain untuk lantas membangun pengaturan yang lengkap. Gambar bertumpuk dapat berbagi gambar inti yang sama, seperti cabang-cabang dari batang pohon yang sama.

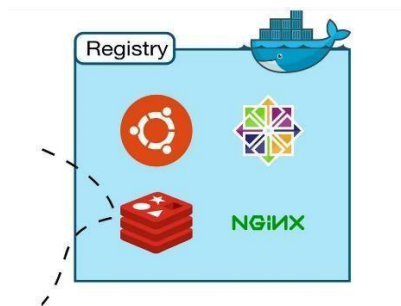
Proses ini dapat dimisalkan ketika seorang pengembang ingin menguji tampilan situs *web* baru di sebuah *browser web* berbeda, tetapi pengembang tidak ingin langsung melakukan instalasi setiap browser ke komputernya. Melakukan hal ini dapat menyebabkan masalah dengan peramban pribadi pengembang. Dalam pengujian inilah nantinya *docker* berguna sebagai tempat uji coba. Docker bekerja menggunakan sistem arsitektur *client-server*. Dalam hal ini nantinya klien akan berkomunikasi dengan apa yang disebut *daemon docker* atau proses pengelolaan *docker images*, *container*, *network*, dan *volume* penyimpanan. *Docker daemon* nantinya akan menerima permintaan pemrosesan dari *Docker Engine Rest API* yang berguna untuk interaksi dan bisa diakses oleh klien melalui *hypertext transfer protocol* (HTTP).

Klien *docker* yang lain adalah *docker compose* yang dapat memungkinkan pengembang untuk bekerja dengan aplikasi yang terdiri dari sekumpulan container. Dalam ilustrasi di atas container yang dipakai adalah Ubuntu, sehingga nantinya akan menempel secara interaktif ke sesi baris perintah yang dimiliki pengembang. Saat pengembang mulai menjalankan perintah terhadap container *Ubuntu* maka akan terjadi konfigurasi oleh *docker* dan pembuatan *container* baru. Selanjutnya, *docker* akan mengalokasikan dokumen sistem ke dalam container sebagai lapisan terakhirnya. Hal ini memungkinkan container yang sedang berjalan untuk membuat atau memodifikasi dokumen dan direktori ke sistem dokumen lokal.

6.4.5 Docker File

Dockerfile adalah file teks yang berisi instruksi yang diperlukan untuk membuat *image container* baru. Instruksi ini termasuk identifikasi image yang ada untuk digunakan sebagai dasar, perintah yang akan dijalankan selama proses pembuatan *image*, dan perintah yang akan berjalan ketika instans baru image kontainer disebarkan.

6.4.6 Docker Registry



Gambar 4 Docker Registry

Registry adalah sebuah *repository* atau sebuah penyimpanan image terpusat, kasarnya *image* yang sudah dibuat oleh orang orang atau mungkin kalian ikut membuat sebuah image, maka *registry* adalah sebuah tempat yang tepat untuk menyimpan *images* kalian disebarkan.

Contoh dari *Docker Registry* sebagai berikut:

1. *Harbour*, opsi sumber terbuka lainnya.
2. *Docker Hub*, *registry* resmi berbasis *cloud* dan *docker*. Berbeda dengan *docker registry open source*, *Docker Hub* adalah layanan terkelola sepenuhnya yang hanya tersedia dari *Docker*.
3. *JBfrog Artifactory*, manajer biner yang dapat digunakan untuk *menghosting* gambar *container* baik lokal maupun di *cloud*.
4. *Amazon Elastic Container Registry*, layanan *registri docker* di *cloud* AWS.
5. *Azure Container Registry*, solusi *registrer* *kontainer* utama yang *dihosting* di *Azure*.
6. *Registry Container Google Cloud*.