Container Virtualization : Docker

Tujuan:

- Mahasantri mampu memasang (instal) perangkat lunak docker
- Mahasantri mampu menggunakan perangkat lunak docker untuk menerapkan virtualisasi sistem operasi atau virtualisasi berbasis container

Pengantar

Docker adalah sebuah mesin (*engine*) *open-source* yang mengotomatisasi penyebaran (*deployment*) aplikasi ke dalam *container*. Docker dibuat oleh tim di Docker, Inc (sebelumnya dotCloud Inc, pemain awal untuk pasar Platform-As-A-Service (PaaS)), dan dirilis oleh mereka di bawah lisensi Apache 2.0

Docker menambahkan engine penyebaran aplikasi di atas sebuah lingkungan eksekusi virtualiasi container. Docker dirancang untuk menyediakan lingkungan yang ringan dan cepat untuk menjalankan kode aplikasi Anda serta alur kerja yang efisien untuk mendapatkan kode dari laptop Anda ke lingkungan pengujian dan kemudian ke lingkungan produksi.

Ilustrasi kegunaan dan peran dari docker dapat dilihat pada gambar dibawa berikut ini.

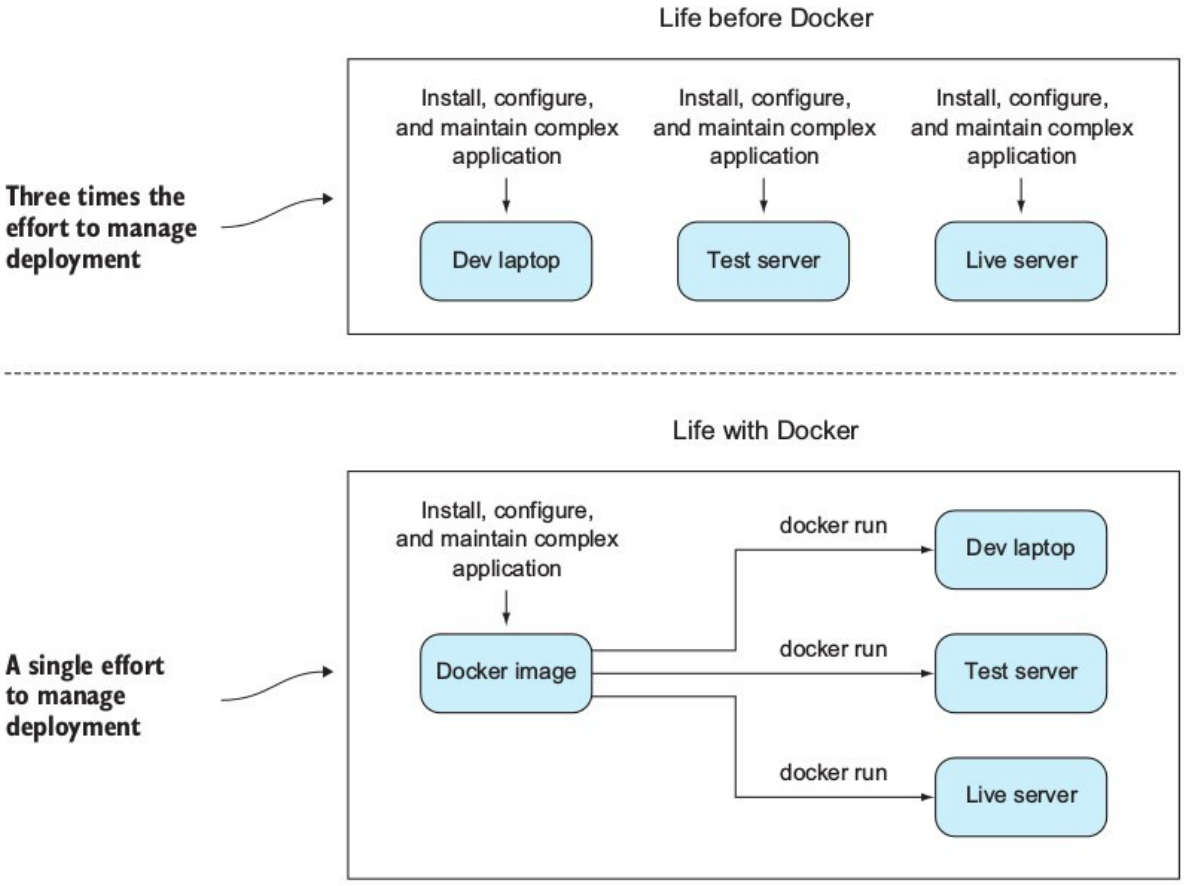

Figure 1.3 **Software delivery before and after Docker**

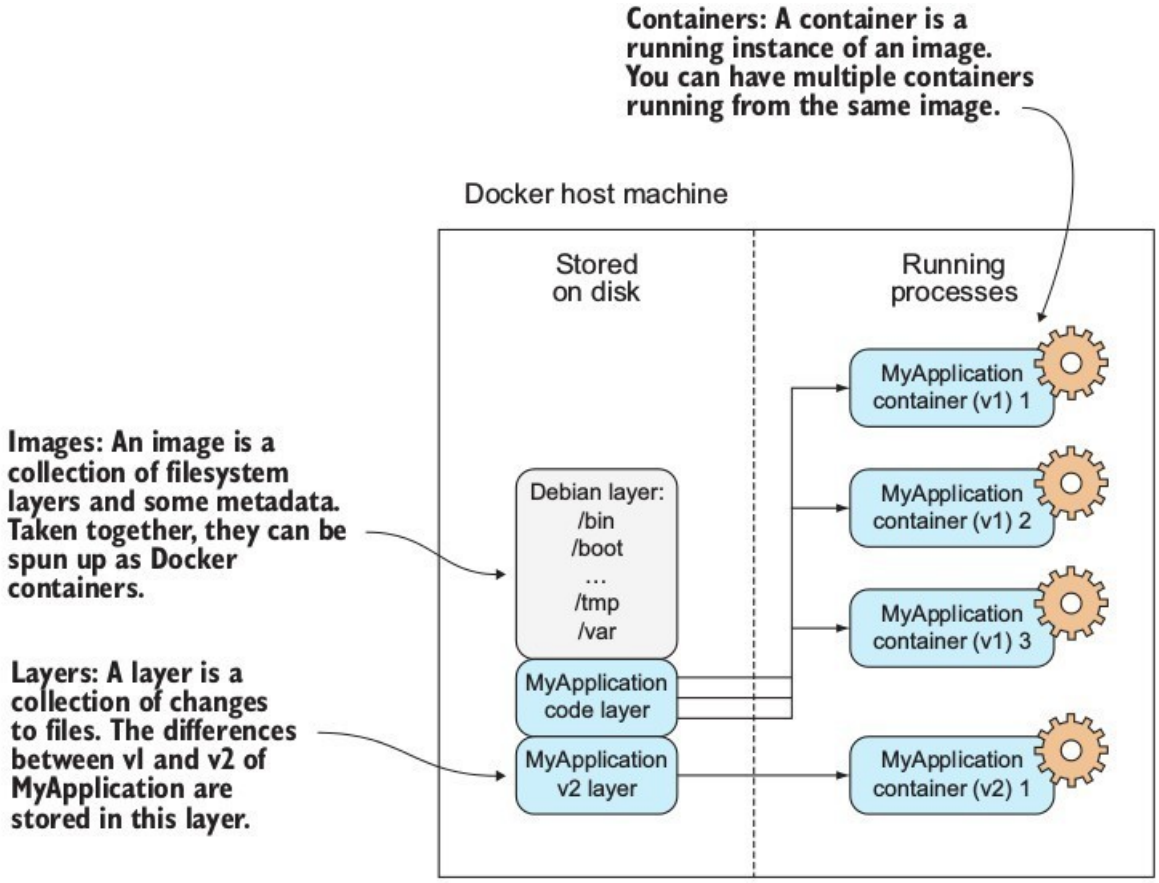

Figure 1.4 **Core Docker concepts**

A. Memasang (instal) perangkat lunak Docker

```
$ sudo su -

# curl -fsSL https://get.docker.com/ | sh
```

B. Menajamen Docker daemon

Setelah memasang (menginstal) Docker, kita perlu mengkonfirmasi bahwa daemon Docker berjalan. Docker berjalan sebagai proses (daemon) dengan privilege root untuk memungkinkan menangani operasi yang tidak dapat dieksekusi oleh pengguna normal (misalnya, pemasangan filesystem). Docker binari berjalan sebagai klien dari daemon ini, dan juga membutuhkan hak akses root untuk berjalannya.

Docker daemon biasanya dijalankan sesaat setelah diinstal dan akan selalu dijalankan secara default saat sistem boot. Secara default, daemon

mendengarkan pada soket Unix di /var/run/docker.sock untuk permintaan Docker yang masuk. Jika pada sistem (host) terdapat grup **docker** , Docker akan menerapkan kepemilikan soket untuk grup tersebut. Oleh karena itu, setiap pengguna yang termasuk dalam grup docker dapat menjalankan Docker tanpa perlu menggunakan perintah sudo.

Untuk memeriksa apakah Docker daemon telah bejalan atau tidak , ketika perintah berikut:

```
 # service docker status
```

atau

```
# initctl status docker
```

Jika Docker daemon belum berjalan , maka lakukan perintah berikut:

```
 # service docker start
```

atau

```
# initctl start docker
```

Jika Docker daemon ingin dimatikan , maka lakukan perintah berikut:

```
 # service docker stop
```

atau

```
# initctl stop docker
```

C. Memeriksa docker client

Untuk memeriksa apakah program biner docker (docker client) berjalan , ketiklah perintah berikut ini:

```
# docker info
```

Perhatikan output dari perintah diatas. !

D. Membangun container pertama Anda

Anda akan mencoba membangun container pertama Anda, dengan cara mengetikkan perintah berikut ini (catatan: Anda harus terhubung ke internet) :

```
# docker run -i  -t  ubuntu debootstrap /bin/bash
```

Perintah diatas akan menyebabkan docker mencari image *ubuntu-debootstrap* pada lokal komputer. Jika tidak ada maka akan dilakukan pencarian dan download image yang ada di repository docker.io.

Image adalah kumpulan lapisan filesystem dan beberapa metadata. Secara bersama-sama, mereka dapat berlaku sebagai Docker Container.

Jika container berhasil dibuat, maka Anda akan melihat shell prompt seperti

berikut: **root@c9c74cc053b0:/#**

Lihatlah struktur direktori pada container, dengan perintah:

root@c9c74cc053b0:/# ls root@c9c74cc053b0:/#

ls /sbin

Filesystem struktur yang tampak adalah filesystem dari image yang sebelumnya diperoleh dari repository docker.io.

Menampilkan daftar interface network dan ip address, dengan perintah:

root@c9c74cc053b0:/# ip add ls

Keluar dari proses container,

root@c9c74cc053b0:/# exit

E. Melihat daftar images pada host

Seluruh image yang sudah terpasang dalam komputer, dapat dilihat dengan perintah berikut:

```
# docker images
```

F. Melihat daftar container

Untuk melihat daftar container yang sedang berjalan (aktif), Anda dapat menggunakan perintah:

```
# docker ps
```

Untuk melihat seluruh daftar container yang sedang berjalan (aktif) atau sedang mati (tidak aktif), Anda dapat menggunakan perintah:

```
# docker ps a
```

G. Mengaktifkan container

Untuk mengaktifkan atau menjalankan container yang sudah ada , dapat dilakukan dengan menjalankan perintah berikut ini.

```
# docker start c9c74cc053b0
```

dimana c9c74cc053b0 adalah container id (dapat Anda ketahui dengan perintah “docker ps -a”).

atau

```
# docker start suspicious_gooda
```

dimana suspicious_gooda adalah nama container

Selanjutnya jika container telah aktif , Anda dapat memeriksanya dengan perintah:

```
# docker ps
```

H. Menonaktifkan container

Untuk menonaktifkan atau mematikan container , dapat dilakukan dengan menjalankan perintah berikut ini.

```
# docker stop c9c74cc053b0
```

dimana c9c74cc053b0 adalah container id (dapat Anda ketahui dengan perintah “docker ps -a”).

Selanjutnya jika container telah aktif , Anda dapat memeriksanya dengan perintah:

```
# docker ps
```

I. Menjalankan command pada sebuah container aktif

Periksa daftar container aktif dengan perintah:

```
# docker ps
```

Jika tidak ada container yang aktif, maka aktifkan container yang ada,dengan perintah:

```
# docker start c9c74cc053b0
```

dimana c9c74cc053b0 adalah container id (dapat Anda ketahui dengan perintah “docker ps -a”).

Selanjutnya periksa container tersebut apakah sudah aktif atau belum dengan perintah:

```
# docker ps
```

Kemudian eksekusi perintah “ls -al” pada container tersebut, dengan menjalankan perintah berikut:

```
# docker exec -i -t c9c74cc053b0 ls - al
```

Perhatikan output dari perintah tersebut, apakah hasilnya ?

Eksekusi kembali perintah “ps axf” pada container tersebut, dengan menjalankan perintah berikut:

```
# docker exec -i -t c9c74cc053b0 ps axf
```

Anda dapat juga mengakses atau kembali ke shell prompt container aktif dengan perintah berikut:

```
# docker attach  c9c74cc053b0
```

J. Container naming

Docker secara otomatis akan menghasilkan nama secara acak untuk setiap kontainer yang Anda buat. Anda bisa melihat bahwa container yang baru saja kita buat memiliki nama misalnya ***suspicious_gooda***. Jika Anda ingin menentukan nama container tertentu, Anda dapat melakukannya dengan menggunakan flag –**name** .

Anda ingin membuat container dengan nama **my_webapp1** , ketik perintah berikut ini:

```
# docker run -i -t -name my_webapp1  ubuntu  /bin/bash
root@b1e975395f87:/# ps axf root@b1e975395f87:/# exit
```

Perhatikan output dari perintah berikut:

```
# docker ps -a
```

Bagaimanakah hasilnya ?

K. Menginstal apache web server pada container

Diasumsikan container bernama my_webapp1 sudah aktif/berjalan.

```
# docker exec -i -t my_webapp1 /bin/bash
root@b1e975395f87:/# apt update
```

```
root@b1e975395f87:/# apt install apache2

root@b1e975395f87:/# /etc/init.d/apache2 start
```

Selanjutnya periksa dan coba akses layanan web server apache yang berjalan pada container b1e975395f87 !

Perhatikan perbedaan lapisan filesystem yang ada pada container tersebut terhadap filesystem awal sebelum menginstall apache2 , ketik perintah berikut untuk melihat perubahan perubahan file yang telah terjadi pada container b1e975395f87 :

```
# docker diff  b1e975395f87
```

L. Mendapatkan informasi lain dari container

Selain informasi yang diperoleh menggunakan perintah docker ps, kita bisa mendapatkan informasi lebih banyak lagi menggunakan perintah docker inspect.

```
# docker inspect  b1e975395f87_

atau

# docker inspect my_webapp1
```

Untuk menampilkan informasi spesifik seperti informasi IP Address dan Gateway dari container, ketikkan perintah berikut:

```
# docker inspect -format '{{ .NetworkSettings.IPAddress }}'
my_webapp1

# docker inspect -format '{{ .NetworkSettings.Gateway }}'
my_webapp1
```