Mini Seri Infrastruktur

Docker #01
Pengenalan, Instalasi dan Penggunaan Dasar

Edisi:
1.0
9 September 2017

Docker #01
Pengenalan, Instalasi dan Penggunaan Dasar

Yudha H Tejaningrat
yht@kalamangga.net

1.0
9 September 2017

Daftar Isi

Halaman

Pendahuluan

1.1 Tentang Kalamangga.Net

Proyek Kalamangga.Net di inisiasi pada Agustus 2011. Proyek ini awalnya merupakan bagian dari Proyek B2B.Web.ID yang dibangun sebagai media pengumpul data.

Pada perkembangannya, Tahun 2014, proyek Kalamangga.Net dipisahkan dengan fokus pada infrastruktur jaringan komputer. Infrastruktur B2B.Web.ID masih berjalan dengan dukungan Kalamangga.Net.

Pada Tahun 2015, seiring perkembangan teknologi mahadata, Kalamangga.Net mulai melanjutkan riset dengan mengadopsi teknologi mahadata dalam membangun infrastrukturnya.

Kalamangga.Net memiliki *tag-line* "*Research Facility on Computer Network and Big Data Technology*". Dengan *tag-line* ini kami mencoba mendukung pembelajaran dan pengadopsian teknologi jaringan komputer dan teknologi mahadata dengan memulai riset dalam pembangunan dan pengimplementasiannya.

Bila ingin mengetahui berita dan proses kerja silakan melalui media yang telah disediakan, yaitu Log Pengembangan atau

Channel Telegram. Dan bila ingin berdiskusi silakan menuju Grup Telegram dan mari berdiskusi.

1.2 Tentang Mini Seri Infrastruktur

Mini Seri Infrastruktur ini adalah kumpulan dokumentasi yang disusun sebagai sarana pembelajaran dan dukungan dalam rangka adopsi teknologi jaringan komputer dan teknologi mahadata.

1.2.1 Mengapa disebut Mini Seri?

Karena dalam dokumentasi ini kami berusaha agar seri ini bukan merupakan dokumen yang terlalu panjang sehingga melelahkan untuk dibaca. Dalam satu jilid bukunya diharapkan berisi tidak lebih dari 70 halaman saja.

Dengan batasan ini tentunya penyajian tidak akan memuaskan semua karena harus difokuskan pada beberapa hal saja. Silakan usulkan melalui media yang kami sediakan bila memiliki ide.

1.2.2 Mengapa infrastruktur yang difokuskan?

Sebuah rumah, tentunya harus dibangun diatas dasar atau pondasi yang kuat agar dapat bertahan terhadap segala jenis bencana yang mungkin terjadi. Demikian pula dalam membangun sebuah sistem yang kuat diperlukan dasar atau pondasi yang dijadikan acuan dalam membangun dan mengimplementasikannya.

Untuk itulah jilid pertama mini seri akan difokuskan pada dokumentasi infrastruktur yang digunakan dalam riset dan pengembangan yang telah kami lakukan.

1.3 Perangkat yang Digunakan

Dalam menyusun dokumentasi ini kami akan menggunakan ruang lingkup sesuai infrastruktur yang kami gunakan, yaitu Sistem Operasi GNU/Linux Debian 8.0 (jessie). Pilihan jatuh karena pengembangan Debian yang kami rasa cukup mapan untuk dijadikan sebagai acuan. Sehingga semua platform dan syntax (perintah) yang akan digunakan pada dokumentasi ini hanya berlaku pada lingkungan yang sama dan/atau mirip.

Untuk perangkat keras yang kami sarankan adalah sistem berarsitektur **amd64** dengan RAM minimal 4GB dan media penyimpanan minimal 80GB.

Docker

2.1 Apa Itu Docker?

Docker merupakan salah satu proyek FLOSS (*Free/Libre/Open Source Software*) yang mengembangkan platform teknologi virtualisasi berbasis kontainer. Mirip seperti mesin virtual (VM / *Virtual Machine*) namun diklaim lebih ringan.

Dengan *tag-line*, "*Build, Ship and Run Any App, Anywhere*", Docker dikembangkan untuk para *developer* maupun *sysadmin* untuk dapat membangun, mem-*bundle* dan menjalankan aplikasi dimanapun. Jadi, meskipun *bundle* aplikasi dikembangkan di Docker dalam sistem operasi berbasis Debian GNU/Linux, aplikasi dapat dijalankan di sistem operasi berbasis GNU/Linux lainnya, bahkan di sistem operasi Windows atau Mac OS.

2.2 VM vs *Container*

Dalam bidang virtualisasi, terdapat istilah VM (*Virtual Machine*) dan *Container*. Secara prinsip keduanya merupakan skema virtualisasi, namun terdapat beberapa perbedaan dalam konsep terutama dalam efisiensi sumber daya.

- VM menggunakan keseluruhan sumber daya perangkat keras dalam mesin induk sesuai dengan pengaturan spesifikasi mesin, sehingga layaknya sebuah mesin yang menjalankan beberapa sistem operasi sekaligus bersamaan. *container* berjalan layaknya aplikasi, sehingga hanya menggunakan sumber daya yang diperlukan saja.
- VM membutuhkan kernel tersendiri, sedangkan *container* dapat menggunakan kernel mesin induk. Dengan hanya 1 (satu) kernel, sumberdaya yang digunakan pada infrastruktur berbasis *container* akan lebih efisien.
- Alokasi pada pengaturan VM tidak dapat direalokasikan antar VM ketika sedang berjalan. Pada basis *container* sumber daya mesin *idle* sangat fleksibel direalokasikan ke mesin yang sedang sibuk.

2.3 *Container* Docker vs OpenVZ/LXC

Skema virtualisasi *container* juga diadopsi oleh OpenVZ/LXC yang sudah lebih dahulu dikembangkan sebelum Docker. Apa perbedaan keduanya?

- OpenVZ/LXC menggunakan *image template* yang berisi sebuah sistem operasi secara utuh yang berjalan di atas *container*. Pada Docker, *image template* yang digunakan hanya berisi aplikasi yang akan digunakan saja, sehingga tidak perlu melakukan konfigurasi layaknya sebuah sistem operasi.

- Karena berisi sebuah sistem operasi utuh, *image template* pada OpenVZ akan membutuhkan ruang penyimpanan lebih besar dan waktu *load* lebih lama dibandingkan Docker.

Dengan perbedaan tersebut, akan lebih mudah dan murah

bagi pengembang untuk mengadopsi Docker dibandingkan OpenVZ/LXC.

2.4 Kelebihan Docker

Sebagai salah satu aplikasi virtualisasi yang sedang 'hit', berikut Kelebihan yang bisa Anda dapatkan bila mengadopsi Docker.

2.4.1 Konfigurasi yang Sederhana

Di Docker, ketika kita butuh untuk men-*deploy* sebuah mesin virtual dengan *Container* hanya membutuhkan 1 (satu) baris perintah saja untuk menjalankan *images*. *Images* yang berisi sistem operasi dapat diunduh langsung dari *repository* dan langsung digunakan.

Fitur inilah yang membuat konfigurasi pada Docker menjadi sangat sederhana, mudah dan cukup efisien.

2.4.2 Optimalisasi Sumber Daya

Pada sub-bab sebelumnya, Docker diunggulkan dalam efisiensi penggunaan sumber daya, sehingga sebuah *host* Docker mampu menjalankan banyak *container* secara bersamaan. Selain berbagi pakai kernel sistem operasi, *images* yang digunakan terdiri dari beberapa lapis sistem file dimana lapisan yang sama akan dibagi pakai antar *container*. Dengan begitu, selain menggunakan lebih hemat RAM, Docker juga menghemat penggunaan media penyimpanan (hardisk) dan proses pengunduhan *image* dari *repository*.

2.4.3 Proteksi Sistem Berkas

Docker menggunakan sistem proteksi hanya baca (*read-only mount point*) dan salin saat tulis (*copy on write*) dengan tingkatan yang cukup variatif sesuai kebutuhan, dari tingkat rendah hingga tinggi. Dengan penggunaan sistem proteksi hanya baca ini, sistem dapat terlindungi dari berbagai gangguan sehingga terhindar dari resiko kegagalan sistem dan keamanan.

2.4.4 Sistem yang Terisolasi

Dari kesemuanya, mungkin alasan keamanan merupakan alasan terpenting. Seperti halnya proteksi keamanan *chroot-jail* yang sudah lama digunakan, sistem yang di-*deploy* merupakan sistem yang terisolasi. Sehingga bila terjadi gangguan pada layanan, tidak mengganggu sistem secara keseluruhan. Bila terjadi peretasan dan pengubahan data, misalnya, dapat dilakukan pemulihan dengan menggunakan *image* asal dengan mudah.

2.4.5 Portabilitas

Pada dasarnya, dengan penggunaan Docker, secara pribadi saya menjadi sangat mudah untuk men-*deploy* sistem tanpa harus mempelajari keseluruhan sistem. Karena setiap orang mempunyai preferensi sendiri dalam penggunaan sistem yang dibangunnya.

Misalkan saya yang sudah terbiasa menggunakan sistem berbasis Debian tidak harus mempelajari sistem berbasis Redhat terlebih dahulu untuk melakukan *deploy* sistem karena mendapatkan sistem RHEL. Dengan menggunakan Docker, saya dapat membangun sistem yang dibutuhkan dengan basis Debian dan menjalankannya di atas RHEL dengan mudah.

2.4.6 Adopsi di Platform Awan

Dengan kemudahan-kemudahan yang dikembangkan tersebut, kini banyak penyedia platform awan yang mengadopsi Docker sebagai produk. Bahkan ada yang menggunakan sebagai platform virtualisasi.

Silakan pilih penyedia platform awan yang makin menjamur, dan tanyakan bagaimana men-*deploy* sistem berbasis Docker.

2.5 Ruang Lingkup Buku

Dalam buku Docker 01 ini, batasan pembahasan hanya pada pengenalan dan penggunaan dasar yang ditujukan untuk pemula atau awam yang ingin mengenal sekilas apa itu Docker.

Arsitektur dan Komponen Docker

Setelah sekilas mengenal Docker, dalam bab ini akan dibahas arsitektur dan komponen yang ad dalam Docker.

3.1 Arsitektur

Docker menggunakan mekanisme *client* dan *server*, dimana terdapat komunikasi melalui *socket* dengan *Restful API*. Docker *client* mengirim permintaan ke Docker *server* untuk membangun, mendistribusikan dan menjalankan *container*. Keduanya, *client* dan *server*, dapat berjalan pada sistem yang sama.

3.2 Komponen

Docker memiliki beberapa komponen yang saling membutuhkan.

3.2.1 *Images*

Images pada dasarnya sebuah *template file-system* yang hanya bisa dibaca, sebagai dasar untuk menjalankan *container* seperti

pada OpenVZ. Perbedaan mendasar adalah pada cara membangun dan memanipulasi *image* sesuai kebutuhan cukup mudah dan sederhana.

Dengan kemudahan dan kesederhanaan itu di Docker *Index* banyak *images* yang dikontribusikan oleh pengguna. *Images* yang ter-*publish* dapat diunduh dan digunakan secara bebas dan *images* ini memungkinkan digunakan ulang dibanyak *container* tanpa mengulang instalasi dan konfigurasi dasar sistem, atau bahasa kerennya "*reinventing the wheel*".

Dalam *image* ini juga mendukung pengembangan *images* lanjutan dari basis *image* yang sama. Misal, untuk membangun *web server* dan *database server* kita dapat menggunakan *image* Debian sebagai basis dan membuat *images* lanjutan dengan memasang paket tambahan di atasnya. Teknologi ini juga dipakai VMWare dan dinamai *linked clone*.

3.2.2 *Container*

Bila *images* merupakan *template file-system* hanya baca, *container* bersifat *read-write* yang berjalan atau berbasis pada *images* sebagai layer baru. *File-system* Docker yang menggunakan *union file-system* sebagai *back-end* memungkinkan hal ini. Jadi *container* adalahnya layer dimana kita dapat melakukan perubahan, misal memasang aplikasi.

Masing-maing *container* terisolasi, sehingga tidak saling mengganggu meskipun dalam satu *host* yang sama.

3.2.3 *Registry*

Docker *Registry* adalah sebutan untuk *repository* untuk mendistribusikan *images* yang terpusat, baik itu bersifat publik maupun privat. *Registry* yang disediakan secara publik oleh Docker disebut Docker Hub.

Dalam *registry* ini dapat disimpan docker *images* yang telah dibangun maupun menggunakan docker *images* yang sudah ada. Karena tersedia secara publik, dalam docker *registry* publik terdapat banyak sekali *images*, baik *official* maupun dibangun oleh personal.

3.2.4 *Dockerfile*

Dockerfile pada dasarnya adalah sebuah *script* otomasi untuk membangun sebuah *images*. *Dockerfile* ini berupa berkas teks atau *script* yang berisi perintah-perintah yang biasanya dijalankan untuk membangun sistem secara manual.

Bagaimana cara membuat dan membangun *images* dengan *dockerfile* ini akan dilanjutkan di perintah dasar *docker build*.

3.2.5 *Repository*

Fungsi *repository* pada Docker layaknya pada Git dan SCM (*Source Code Management*) atau VCS (*Version Control System*) dimana kita dapat memberi nama pada proyek, dalam hal ini *images*, berupa ID dan menyimpannya pada *registry*. ID yang digunakan seperti di Github, namauser/namaproyek, tentunya di Docker namaproyek diganti dengan namaimage. Penamaan ini berlaku untuk *registry* publik maupun privat. Penamaan ini menunjukkan pengembang dan identitas *images* sehingga mudah dipahami fungsinya.

3.2.6 *Index*

Docker *Index* berfungsi untuk mengatur *user account*, hak akses (*permission*), *search*, *tagging* dan segala sesuatu yang tersimpan sebagai basis data antar muka web *repository*. Identitas yang kita

panggil dalam *docker run* atau *docker pull* dicari dalam *index*. Berdasar hak akses dalam *index*, *registry* dapat menyediakan dan memberikan ijin untuk mengakses atau memodifikasi *images*.

Pemasangan dan Perintah Dasar

4.1 Pemasangan

Pemasangan Docker sama seperti pemasangan aplikasi lain yang ada di GNU/Linux. Pada bab ini akan coba dibahas bagaimana pemasangan Docker di sistem operasi Debian GNU/Linux 8.0 (jessie).

4.1.1 Pemasangan Manual

Proses pemasangan manual ini adalah prosedur normal yang dilakukan untuk memasang aplikasi Docker. Langkah-langkah yang perlu dijalankan, akan kami rinci. Sebagai catatan, semua perintah dijalankan oleh pengguna `root`.

- Perbaharui daftar aplikasi.
  `apt-get update; apt-get upgrade`
- Pasang `apt-transport-https` dan `ca-certificates`.
  `apt-get install apt-transport-https ca-certificates curl gnupg2 software-properties-common`

- Pasang gpg-key untuk *repository* Docker.
  `curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -`
- Pastikan kunci yang benar terpasang.
  `apt-key fingerprint 0EBFCD88`
- Tambahkan *repository* paket Docker.
  `echo "deb https://download.docker.com/linux/debian jessie stable" | tee /etc/apt/sources.list.d/docker.list`
- Perbaharui daftar aplikasi dan pasang *docker-ce*.
  `apt-get update; apt-get install docker-ce`

Dan Docker sudah terpasang di sistem Anda.

4.1.2 Pemasangan dengan *script*

Sebelum memasang dengan cara ini harap membaca terlebih dahulu kode yang diunduh dari laman web. Segala kerusakan yang terjadi karena hal ini menjadi tanggung jawab pembuat kode dan yang menjalankannya.

- Unduh skrip kode pemasangan.
  `curl -fsSL get.docker.com -o get-docker.sh`
- Jalankan kode tersebut.
  `sh get-docker.sh`

Kode tersebut akan memilih dan mengunduh serta memasang paket-paket yang dibutuhkan secara otomatis.

4.1.3 Group '*docker*'

Dalam manajemen Docker semua pengguna yang masuk dalam group '*docker*' akan memiliki hak untuk mengakses dan menjalankan Docker. Tambahkan akun pengguna yang diberikan hak pada grup ini. Perintah untuk menambahkan:

```
usermod -aG docker [namapengguna]
```

4.2 Perintah Dasar

Setelah Docker terpasang, mari mengenal perintah dasar yang digunakan dalam operasional harian.

4.2.1 *docker version*

Perintah pertama ini untuk mengetahui versi Docker yang terpasang.

```
yht@debian:~$ docker version
Client:
 Version:      17.05.0-ce
 API version:  1.29
 Go version:   go1.7.5
 Git commit:   89658be
 Built:        Thu May  4 22:04:27 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.05.0-ce
 API version:  1.29 (minimum version 1.12)
 Go version:   go1.7.5
```

```
 Git commit:   89658be
 Built:        Thu May  4 22:04:27 2017
 OS/Arch:      linux/amd64
 Experimental: false
yht@debian:~$
```

4.2.2 *docker info*

Perintah kedua ini untuk melihat informasi sekilas mengenai mesin dan layanan Docker yang terpasang dalam mesin. Jumlah *container* yang berjalan, *images* yang sudah diunduh atau dibangun dilokal, dan beberapa informasi yang lain.

```
yht@debian:~$ docker info
Containers: 1
 Running: 0
 Paused: 0
 Stopped: 1
Images: 77
Server Version: 17.05.0-ce
Storage Driver: aufs
 Root Dir: /mnt/docker/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 179
 Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
```

```
containerd version: 9048e5e50717ea4497b757314bad98ea3763c145
runc version: 9c2d8d184e5da67c95d601382adf14862e4f2228
init version: 949e6fa
Kernel Version: 3.16.0-4-amd64
Operating System: Debian GNU/Linux 8 (jessie)
OSType: linux
Architecture: x86_64
CPUs: 2
Total Memory: 11.66GiB
Name: debian
ID: 43NB:JDDT:V4VS:7M5H:3KD6:UB7W:R7MJ:J3SE:JS4O:H7Q7:D3RJ:VU5C
Docker Root Dir: /mnt/docker/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Username: ryht
Registry: https://index.docker.io/v1/
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false

WARNING: No memory limit support
WARNING: No swap limit support
WARNING: No kernel memory limit support
WARNING: No oom kill disable support
WARNING: No cpu cfs quota support
WARNING: No cpu cfs period support
yht@debian:~$
```

Dari informasi di atas ada beberapa informasi. Terdapat 1 buah *container* dengan status tidak beroperasi, *images* yang siap pakai ada 77 buah, dan informasi mesin *dual-core* dengan RAM 12GB ber-OS Debian Jessie.

4.2.3 *docker search*

Perintah yang ketiga ini membutuhkan akses ke dalam Docker *index*. Perintah ini mengunduh daftar *images* dari *index* sesuai dengan yang dicari.

```
yht@debian:~$ docker search debian
NAME                                    DESCRIPTION
ubuntu                                  Ubuntu is a Debian-based Linux
debian                                  Debian is a Linux distribution
google/debian
neurodebian                             NeuroDebian provides neuroscien
armhf/debian                            Debian is a Linux distribution
itscaro/debian-ssh                      debian:jessie
arm32v7/debian                          Debian is a Linux distribution
resin/armv7hf-debian                    Debian is a Linux distro compos
samueldebruyn/debian-git                a minimal docker container with
armbuild/debian                         ARMHF port of debian
eboraas/debian                          Debian base images, for all cur
arm64v8/debian                          Debian is a Linux distribution
i386/debian                             Debian is a Linux distribution
aarch64/debian                          Debian is a Linux distribution
rockyluke/debian                        Docker images of Debian.
vicamo/debian                           Debian docker images for all ve
ppc64le/debian                          Debian is a Linux distribution
dockershelf/debian                      Repository for docker images of
s390x/debian                            Debian is a Linux distribution
vergissberlin/debian-development        Docker debian image to use for
igneoussystems/base-debian-client       Base image for debian clients
trollin/debian
casept/debian-amd64                     A debian image built from scrat
smartentry/debian                       debian with smartentry
jdub/debian-sources-resource            Concourse CI resource to check
yht@debian:~$
```

4.2.4 *docker pull*

Perintah ini digunakan untuk mengunguh *images* dari *repository* publik (Docker Hub).

```
yht@debian:~$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
06b22ddb1913: Pull complete
Digest: sha256:6ccbcbf362dbc4add74711cb774751b59cdfd7aed16c3c29aaec
Status: Downloaded newer image for debian:latest
yht@debian:~$
```

Bila *tag* tidak diberikan, secara otomatis Docker akan mengarahkan untuk menggunakan *tag latest* (terakhir).

Selain untuk mengunduh *images*, perintah ini digunakan juga untuk memperbaharui *images* yang sudah ada dari *index* dan mengunduh pembaharuannya dari *repository*.

4.2.5 *docker run*

Sekarang kita belajar masuk ke mesin virtual dengan perintah *run* dengan opsi *-it* yang meminta akses interaktif ke mesin, sehingga kita dapat menjalankan perintah layaknya di *console*.

```
yht@debian:~$ uname -a
Linux debian 3.16.0-4-amd64 #1 SMP Debian 3.16.43-2+deb8u3 (2017-08
yht@debian:~$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 8 (jessie)"
NAME="Debian GNU/Linux"
VERSION_ID="8"
VERSION="8 (jessie)"
ID=debian
```

```
HOME_URL="http://www.debian.org/"
SUPPORT_URL="http://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
yht@debian:~$ docker run -it debian:latest
root@240ead5e3e3f:/# uname -a
Linux 240ead5e3e3f 3.16.0-4-amd64 #1 SMP Debian 3.16.43-2+deb8u3 (2
root@240ead5e3e3f:/# cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
NAME="Debian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
root@240ead5e3e3f:/# exit
exit
yht@debian:~$
```

Dapat dilihat perbedaan dimana *host* merupakan Debian Jessie sedangkan *guest* merupakan *images* Debian *latest* atau Debian Stretch.

4.2.6 *docker ps*

Sekarang kita akan coba melihat daftar *container* yang ada pada mesin. Coba perintah berikut:

```
yht@debian:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATE
240ead5e3e3f        debian:latest       "bash"              18 min
yht@debian:~$
```

Ternyata ada 1 *container* yang dibuat 18 menit yang lalu.

4.2.7 *docker images*

Bila *ps* dipakai untuk melihat *container* yang telah dibuat, *docker images* merupakan perintah untuk melihat *images* yang telah diunduh atau dibuat di sistem.

```
yht@debian:~$ docker images
REPOSITORY     TAG            IMAGE ID            CREATED
python         2.7.13         fa8e55b2235d        5 weeks ago
python         latest         968120d8cbe8        5 weeks ago
debian         latest         a20fd0d59cf1        5 weeks ago
debian         jessie         86baf4e8cde9        5 weeks ago
yht@debian:~$
```

Empat *images* yang ada dalam daftar siap dipakai tanpa perlu koneksi ke *index* maupun *repository*.

4.2.8 *docker start*

Seperti namanya, perintah ini meminta Docker untuk menjalankan *container* yang telah dimatikan atau dengan status *Exited (0)*.

```
yht@debian:~$ docker ps -a
CONTAINER ID    IMAGE           COMMAND     CREATED          STATUS
33cc15abecc3    debian          "bash"      7 seconds ago    Exited
9906f0f91831    debian:jessie   "bash"      2 hours ago      Up 2 h
yht@debian:~$ docker start 33cc15abecc3
33cc15abecc3
yht@debian:~$ docker ps -a
CONTAINER ID    IMAGE           COMMAND     CREATED          STATUS
33cc15abecc3    debian          "bash"      31 seconds ago   Up 2 s
9906f0f91831    debian:jessie   "bash"      2 hours ago      Up 2 h
yht@debian:~$
```

4.2.9 *docker attach*

Perintah ini, sama halnya dengan *chroot* atau masuk ke dalam *container* yang sedang berjalan. Untuk melakukannya pastikan mesin virtual (*container*) dalam posisi *Up*.

```
yht@debian:~$ docker ps -a
CONTAINER ID    IMAGE           COMMAND     CREATED          STATUS
33cc15abecc3    debian          "bash"      31 seconds ago   Up 2 s
9906f0f91831    debian:jessie   "bash"      2 hours ago      Up 2 h
yht@debian:~$ docker attach 33cc15abecc3
root@33cc15abecc3:/# exit
exit
yht@debian:~$ docker ps -a
CONTAINER ID    IMAGE           COMMAND   CREATED          STATUS
33cc15abecc3    debian          "bash"    3 minutes ago    Exited (
9906f0f91831    debian:jessie   "bash"    2 hours ago      Up 2 hou
yht@debian:~$ docker attach 33cc15abecc3
You cannot attach to a stopped container, start it first
yht@debian:~$
```

Bila posisi *container* mati, galat pada baris perintah terakhir akan muncul.

4.2.10 *docker rm*

Bila kita memiliki *container* yang sudah tidak lagi terpakai, kita dapat membersihkannya dengan menggunakan *rm* (hapus / *remove*). Misalkan kita ingin menghapus *container* **33cc15abecc3** yang telah dibuat dari *images* debian:latest.

```
yht@debian:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
33cc15abecc3        debian              "bash"              28 minu
```

```
9906f0f91831        debian:jessie       "bash"              2 hours
yht@debian:~$ docker rm 33cc15abecc3
33cc15abecc3
yht@debian:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
9906f0f91831        debian:jessie       "bash"              2 hours
yht@debian:~$
```

4.2.11 *docker rmi*

Nah, perintah terakhir ini sangat berguna bila ingin menghemat penggunaan ruang disk. Terutama bila terdapat banyak *images* yang sudah tidak terpakai, perintah ini akan menghapusnya.

```
yht@debian:~$ docker images | grep debian
debian         latest              a20fd0d59cf1        5 weeks ago
debian         jessie              86baf4e8cde9        5 weeks ago
debian         wheezy              bbd62956fac7        2 months ago
debian         <none>              a2ff708b7413        2 months ago
debian         <none>              62a932a5c143        2 months ago
yht@debian:~$ docker rmi a2ff708b7413
Untagged: debian@sha256:7d067f77d2ae5a23fe6920f8fbc2936c4b0d417e9d0
Deleted: sha256:a2ff708b74137df70ad546f35d6034ef3b3b354b3020c318d79
yht@debian:~$ docker rmi 62a932a5c143
Untagged: debian@sha256:4bc62f74d246e8428be8dd3833461ba2cfd135064ae
Deleted: sha256:62a932a5c143fb4cb0fe444ff5eded246015d86a169dffd77a9
yht@debian:~$ docker images | grep debian
debian         latest              a20fd0d59cf1        5 weeks ago
debian         jessie              86baf4e8cde9        5 weeks ago
debian         wheezy              bbd62956fac7        2 months ago
yht@debian:~$
```

Debian di Docker

Setelah Docker terpasang dan mengenal perintah-perintah dasarnya, kini kita masuk pada implementasi atau menggunakan sebagai media virtualisasi. Dan mengingat semua platform yang kami gunakan adalah GNU/Linux Debian 8.0 (jessie) maka sebagai contoh akan digunakan pula dalam sistem di dalam Docker.

5.1 Pemasangan Debian di Docker

Bila Anda sudah membaca bab sebelumnya tentunya sudah tidak bingung lagi perintah apa yang harus dijalankan. Silakan ke bab sebelumnya pada bagian perintah dasar, *docker pull* dan *docker run* bila masih bingung.

Namun sebagai catatan, pada bagian ini kita tidak akan menggunakan *images latest*. Kita akan menggunakan Debian Jessie, dan tentunya ini akan membutuhkan argumen tambahan.

```
yht@debian:~$ docker pull debian:jessie
jessie: Pulling from library/debian
ad74af05f5a2: Already exists
```

```
Digest: sha256:51cd80bb935b76fbbf49640750736abc63ab7084d5331e198326
Status: Downloaded newer image for debian:jessie
yht@debian:~$ docker run -it debian:jessie
root@9906f0f91831:/# cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 8 (jessie)"
NAME="Debian GNU/Linux"
VERSION_ID="8"
VERSION="8 (jessie)"
ID=debian
HOME_URL="http://www.debian.org/"
SUPPORT_URL="http://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
root@9906f0f91831:/#
```

Ok, kita sudah masuk sebagai *root* ke mesin virtual dan menggunakan *images* Debian Jessie.

5.2 Pemasangan Aplikasi Editor Basis Teks

Pertama-tama kita akan melakukan instalasi paket editor basis teks. Untuk pengguna awam disarankan menggunakan *nano*, atau *vim* untuk lanjutan. Namun sebelum memasang paket jangan lupa untuk memperbaharui daftar paket.

Mari kita mulai.

```
root@9906f0f91831:/# apt-get update
Get:1 http://security.debian.org jessie/updates InRelease [63.1 kB]
Get:2 http://security.debian.org jessie/updates/main amd64 Packages
Ign http://deb.debian.org jessie InRelease
Get:3 http://deb.debian.org jessie-updates InRelease [145 kB]
Get:4 http://deb.debian.org jessie Release.gpg [2373 B]
```

```
Get:5 http://deb.debian.org jessie Release [148 kB]
Get:6 http://deb.debian.org jessie-updates/main amd64 Packages [17.
Get:7 http://deb.debian.org jessie/main amd64 Packages [9063 kB]
Fetched 10.0 MB in 22s (440 kB/s)
Reading package lists... Done
root@9906f0f91831:/# apt-get install nano
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  spell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 369 kB of archives.
After this operation, 1707 kB of additional disk space will be used
Get:1 http://deb.debian.org/debian/ jessie/main nano amd64 2.2.6-3
Fetched 369 kB in 2s (165 kB/s)
debconf: delaying package configuration, since apt-utils is not ins
Selecting previously unselected package nano.
(Reading database ... 7566 files and directories currently installe
Preparing to unpack .../nano_2.2.6-3_amd64.deb ...
Unpacking nano (2.2.6-3) ...
Setting up nano (2.2.6-3) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (ed
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico
root@9906f0f91831:/#
```

Nah, untuk memanggil editor ini bisa menggunakan perintah *nano*, *editor* atau *pico* yang semuanya sama saja untuk memanggil editor teks *nano*.

5.3 Mengubah Lumbung Paket (*Repository*)

Setelah teks editor terpasang, kita dapat menggunakannya untuk mengubah berkas teks. Pada sub-bab ini akan mengubah berkas konfigurasi cermin Debian agar mengarah ke *repository* Debian yang disediakan oleh lokal, misal Kambing.UI.AC.ID atau Kartolo.SBY.DataUtama.Net.ID.

Panggil dan ubah berkas *sources.list*.

```
root@9906f0f91831:/# nano /etc/apt/sources.list
```

Berikut konfigurasi untuk menggunakan Kartolo.SBY.DataUtama.Net.ID sebagai cermin.

```
deb http://kartolo.sby.datautama.net.id/debian jessie main
deb http://kartolo.sby.datautama.net.id/debian jessie-updates main
deb http://kartolo.sby.datautama.net.id/debian-security jessie/upda
```

5.4 Update / Upgrade Sistem

OK, setelah arah cermin diubah, saatnya memperbaharui paket.

```
root@9906f0f91831:/# apt-get update
Ign http://kartolo.sby.datautama.net.id jessie InRelease
Get:1 http://kartolo.sby.datautama.net.id jessie-updates InRelease
Get:2 http://kartolo.sby.datautama.net.id jessie/updates InRelease
Get:3 http://kartolo.sby.datautama.net.id jessie Release.gpg [2373
Get:4 http://kartolo.sby.datautama.net.id jessie Release [148 kB]
Get:5 http://kartolo.sby.datautama.net.id jessie-updates/main amd64
Get:6 http://kartolo.sby.datautama.net.id jessie/updates/main amd64
```

```
Get:7 http://kartolo.sby.datautama.net.id jessie/main amd64 Package
Fetched 9999 kB in 28s (352 kB/s)
Reading package lists... Done
root@9906f0f91831:/# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@9906f0f91831:/#
```

Selamat, tidak perlu mengunduh apapun. Hemat *bandwidth.* :D

5.5 Jaringan

Setelah arah cermin ke lokal, yang tentunya lebih hemat dan cepat. Saatnya kita melakukan konfigurasi untuk jaringan.

5.5.1 Pemasangan Aplikasi Jaringan

Mula-mula kita pasang terlebih dahulu paket-paket yang merupakan aplikasi jaringan komputer standar, yaitu *nettools* dan *nmap*.

```
root@9906f0f91831:/# apt-get install net-tools nmap
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
.... [truncated] ....
0 upgraded, 60 newly installed, 0 to remove and 0 not upgraded.
Need to get 19.7 MB of archives.
```

```
After this operation, 89.7 MB of additional disk space will be used
Do you want to continue? [Y/n]
.... [truncated] ....
Fetched 19.7 MB in 1min 54s (172kB/s)
debconf: delaying package configuration, since apt-utils is not ins
Selecting previously unselected package libgdbm3:amd64.
(Reading database ... 7651 files and directories currently installe
.... [truncated] ....
Unpacking nmap (6.47-3+deb8u2) ...
Setting up libgdbm3:amd64 (1.8.3-13.1) ...
Setting up libssl1.0.0:amd64 (1.0.1t-1+deb8u6) ...
debconf: unable to initialize frontend: Dialog
.... [truncated] ....
Setting up ndiff (6.47-3+deb8u2) ...
Setting up nmap (6.47-3+deb8u2) ...
Processing triggers for libc-bin (2.19-18+deb8u10) ...
Processing triggers for sgml-base (1.26+nmu4) ...
root@9906f0f91831:/#
```

Untuk mencoba apakah sudah benar terpasang, gunakan perintah *ifconfig*.

5.5.2 Jaringan *Bridge*

Standarnya, semua *container* yang ada di Docker menggunakan jaringan *bridge* dengan subnet 172.17.0.0/16 dengan *host* beralamat ip 172.17.0.1 pada *ethernet* virtual bernama *docker0*.

```
yht@debian:~$ /sbin/ifconfig docker0
docker0   Link encap:Ethernet  HWaddr 02:42:9d:f9:60:d3
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:9dff:fef9:60d3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9294 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:14811 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:462026 (451.1 KiB)  TX bytes:21294918 (20.3 MiB)

yht@debian:~$
```

Dan bila kita cek pada *container* yang sedang berjalan, akan secara otomatis bergabung pada subnet.

```
root@9906f0f91831:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:02
          inet addr:172.17.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:21284 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11782 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:30552160 (29.1 MiB)  TX bytes:662996 (647.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@9906f0f91831:/#
```

5.6 Volume

Ketika kita membuat *container*, sumber daya yang ada pada dasarnya sama dengan *host*. Misal, pada *host* memiliki hardisk 80 GB untuk / (*root*) maka *container* pun memiliki spesifikasi yang

sama. Secara *default* Docker akan menggunakan *free-space* partisi / (root), atau partisi dimana berkas-berkas Docker tersimpan. Berkas-berkas ini ada di **/var/lib/docker**.

Lalu bagaimana bila kita memiliki partisi atau disk yang disediakan untuk Docker? Cara pertama adalah membuat berkas *link* **/var/lib/docker** dan mengarahkan ke partisi khusus tersebut. Cara kedua adalah dengan menggunakan fitur **volume**.

Misalkan kita memiliki disk 1TB dan telah di-*mount* di **/mnt/s1tb**. Kita dapat menggunakan partisi tersebut dengan mengarahkan *mount* di *guest* agar menggunakannya.

Sebagai contoh, berikut spesifikasi partisi di *host*.

```
root@debian:~# df
Sistem Berkas     1K-blok      Isi      Sisa Isi% Dipasang di
/dev/sda7        57542652  27273084  27323520  50% /
udev                10240         0     10240   0% /dev
tmpfs             2445088      9284   2435804   1% /run
tmpfs             6112716     14632   6098084   1% /dev/shm
tmpfs                5120         4      5116   1% /run/lock
tmpfs             6112716         0   6112716   0% /sys/fs/cgroup
/dev/sda6        61796348  29279104  29355132  50% /home
/dev/sda9       121268680 112830784   2254576  99% /mnt
tmpfs             1222544         4   1222540   1% /run/user/109
tmpfs             1222544         8   1222536   1% /run/user/1000
/dev/sdb1      1922728752 882837200 942199496  49% /mnt/s1tb
root@debian:~#
```

Dan bila kita ingin menggunakan partisi tersebut kita bisa menggunakan opsi **-v** yang berarti *volume*. Misalkan kita mount di *folder* **/mnt**, berikut cara penggunaannya.

```
root@debian:~# docker run -it -v /mnt/s1tb:/mnt debian:jessie
```

Dan di *guest* partisi / disk tersebut sudah dapat digunakan. Berikut spesifikasi partisi pada *guest* dengan perintah tersebut.

```
root@6d4dc52eab18:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
none            116G  108G  2.2G  99% /
tmpfs           5.9G     0  5.9G   0% /dev
tmpfs           5.9G     0  5.9G   0% /sys/fs/cgroup
/dev/sdb1       1.8T  842G  899G  49% /mnt
/dev/sda9       116G  108G  2.2G  99% /etc/hosts
shm              64M     0   64M   0% /dev/shm
tmpfs           5.9G     0  5.9G   0% /sys/firmware
root@6d4dc52eab18:/#
```

Folder **/mnt** siap digunakan.

5.7 *Inspect*

Sesuai arti katanya, fungsi dari opsi ini adalah melakukan pengecekan terhadap *container*, *images*, ataupun *network* pada Docker. Dimana hasil keluaran ber-format JSON.

```
yht@debian:~$ docker inspect
"docker inspect" requires at least 1 argument(s).
See 'docker inspect --help'.

Usage:  docker inspect [OPTIONS] NAME|ID [NAME|ID...]

Return low-level information on Docker objects
yht@debian:~$
```

Dari keluaran galat di atas bisa diambil informasi bahwa untuk menjalankan pengecekan ini diperlukan argumen yang berisi

NAME atau *ID* dari komponen yang akan dicek. Misalkan kita akan mengecek salah satu *container*.

```
yht@debian:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
6d4dc52eab18        debian:jessie       "bash"              10 minu
9906f0f91831        debian:jessie       "bash"              10 days
yht@debian:~$ docker inspect 6d4dc52eab18
[
    {
        "Id": "6d4dc52eab18964451c245a8b67e325898fb0e197502193444d9
        "Created": "2017-09-08T08:20:42.33509391Z",
        "Path": "bash",
        "Args": [],
        "State": {
            "Status": "exited",
            "Running": false,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 0,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2017-09-08T08:20:43.774090326Z",
            "FinishedAt": "2017-09-08T08:27:01.050795678Z"
        },
        "Image": "sha256:86baf4e8cde94242e5ce75d9fb913eabd088c32f28
[... truncated ...]
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "NetworkID": "b682e9760b6df8522cc2aede096973548
```

```
                    "EndpointID": "",
                    "Gateway": "",
                    "IPAddress": "",
                    "IPPrefixLen": 0,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": ""
                }
            }
        }
    }
]
yht@debian:~$
```

Maaf, harus dipotong. Mengingat keluaran JSON ini terlalu panjang. Untuk membuat keluaran hanya pada bagian diinginkan bisa menggunakan *–format*. Mengenai hal ini akan dimasukkan dalam buku berikutnya.

5.8 *Build Script*, Dockerfile

Pada pemasangan Debian di Docker (sub-bab pertama), perintah untuk melakukan instalasi dimasukkan baris per baris pada *container* yang berjalan. Untuk mempermudah pembuatan, Docker menyediakan fitur untuk membuat *images* sebagai basis sistem Docker yang diinginkan masing-masing pengguna, yaitu *Dockerfile*.

Dockerfile ini pada prinsipnya berisi perintah-perintah *console* yang biasa dijalankan untuk membangun sistem. Sesuai pada sub-bab pertama, berikut isi *dockerfile* untuk membuat *images* Debian dengan **nano editor** di dalamnya.

```
FROM debian:jessie
MAINTAINER Kalamangga.Net R&D
RUN apt-get update && \
    apt-get upgrade -y --force-yes && \
    apt-get clean -y --force-yes
RUN apt-get install -y --force-yes nano && \
    apt-get clean -y --force-yes
CMD ["/bin/bash"]
```

Berkas *dockerfile* ini diambil dari sini dimana berkas tersebut adalah konfigurasi standar sistem dengan Debian Jessie dalam infrastruktur Kalamangga.Net.

Hanya satu baris yang dihapus yaitu penyalinan berkas **sources.list**. Berkas ini sebenarnya hanya mengarahkan *repository* Debian agar menggunakan *repository* lokal dalam jaringan Kalamangga.Net HQ.

Dan untuk menjalankan pembangunan *images* digunakan perintah *docker build*. Jangan lupa memberi nama *images* untuk membedakan dengan yang lain. Sebagai langkah untuk membuat *images* dengan nama **jessie-nano** dijalankan perintah sebagai berikut.

```
yht@debian:~ $ docker build -t jessie-nano .
```

Berjalan? Semoga. :D

Tips dan Trik

Pada bab ini dicoba untuk memberikan tips dan trik sederhana yang diperlukan terutama untuk sarana *back-up* dan jaringan, fitur standar yang pasti dibutuhkan oleh *System Administrator*.

6.1 *BackUp*

Apakah ada yang tahu kapan sistem akan mengalami *crash*? Tentunya tak satu pun *SysAdmin* yang ingin mengalaminya. Prinsip *SysAdmin* yang cerdas, katanya, adalah selalu melakukan *back-up*. Sedia payung sebelum hujan.

Untuk melakukan *back-up* ini ada 2 (dua) perintah yang digunakan, yaitu *commit* dan *save*. Namun sebelumnya pastikan Anda sudah mengetahui *container* ID yang akan di *back-up*.

Ok, mari kita lihat daftar *container*.

```
yht@debian:~$ docker ps -a
CONTAINER ID  IMAGE          COMMAND  CREATED             STATUS
6d4dc52eab18  debian:jessie  "bash"   About an hour ago  Exited (0)
9906f0f91831  debian:jessie  "bash"   10 days ago        Exited (0)
```

Untuk melakukan *back-up* dari *container* yang 1 (satu) jam yang lalu dimatikan dengan ID **6d4dc52eab18**, pertama-tama kita buat dulu *images* dari *container* itu dengan perintah *commit*.

```
yht@debian:~$ docker commit -p 6d4dc52eab18 jessie-backup
sha256:572ff98aa9c5496982c15cc8befafdc422952625b9ff529034806abb8b2b
yht@debian:~$
```

Cek hasil dengan melihat daftar *images*.

```
yht@debian:~$ docker images
REPOSITORY      TAG                 IMAGE ID            CREATED
jessie-backup   latest              572ff98aa9c5        2 minutes ago
python          2.7.13              fa8e55b2235d        6 weeks ago
python          latest              968120d8cbe8        6 weeks ago
debian          latest              a20fd0d59cf1        6 weeks ago
debian          jessie              86baf4e8cde9        6 weeks ago
```

Terlihat *images* **jessie-backup** pada baris teratas.

Berikutnya adalah membuat berkas *back-up* berupa *tape-archive* (tar) dengan perintah *docker save*. Berikut perintahnya.

```
yht@debian:~$ docker save -o jessie-backup.tar jessie-backup
yht@debian:~$ ls -l *.tar
-rw------- 1 yht yht 129310720 Sep  8 16:31 jessie-backup.tar
yht@debian:~$
```

Untuk menghemat tempat berkas ini dapat dikompres dengan perangkat favorit Anda, misal *gzip*, *bzip2* atau *xz*. Misalkan untuk menjadikannya terkompresi *gzip* bisa menggunakan perintah berikut.

```
yht@debian:~$ gzip -9 jessie-backup.tar
```

```
yht@debian:~$ ls -l *.tar.gz
-rw-r--r-- 1 yht yht   780308 Mar 13 04:50 etc.tar.gz
-rw------- 1 yht yht 51343177 Sep  8 16:45 jessie-backup.tar.gz
-rw-r--r-- 1 yht yht  7136508 Mar 13 04:51 log.tar.gz
yht@debian:~$
```

Menghemat tempat sebanyak 78MB.

6.2 *Restore*

Bagaimana bila hal yang tidak diinginkan itu terjadi? Beruntung kita telah melakukan *back-up*. Untuk membuat layaknya kita menginstall ulang mesin, kita hapus dulu *images* yang tadi telah dibuat dengan perintah *rmi*.

```
yht@debian:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
jessie-backup       latest              572ff98aa9c5        33 minutes ago
python              2.7.13              fa8e55b2235d        6 weeks ago
python              latest              968120d8cbe8        6 weeks ago
debian              latest              a20fd0d59cf1        6 weeks ago
yht@debian:~$ docker rmi 572ff98aa9c5
Untagged: jessie-backup:latest
Deleted: sha256:572ff98aa9c5496982c15cc8befafdc422952625b9ff5290348
Deleted: sha256:05c3827260888ad32d20407dbdab3a86f451a637a76e2087f59
yht@debian:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
python              2.7.13              fa8e55b2235d        6 weeks ago
python              latest              968120d8cbe8        6 weeks ago
debian              latest              a20fd0d59cf1        6 weeks ago
yht@debian:~$
```

Untuk me-*restore images* yang telah kita *back-up*, digunakan perintah *load*. Dan cek apakah ada dalam daftar *images*.

```
yht@debian:~$ docker load -i jessie-backup.tar
0efca2e42300: Loading layer [=======================================
Loaded image: jessie-backup:latest
yht@debian:~$ docker images
REPOSITORY          TAG           IMAGE ID            CREATED
jessie-backup       latest        572ff98aa9c5        43 minutes ag
python              2.7.13        fa8e55b2235d        6 weeks ago
python              latest        968120d8cbe8        6 weeks ago
debian              latest        a20fd0d59cf1        6 weeks ago
```

Silakan gunakan *images* tersebut.

6.3 *Save* atau *Export*?

Pada sub-bab pertama Tips dan Trik diperkenalkan perintah *save* untuk menyimpan berkas *back-up* dari *images*. Sedangkan masih ada 1 (satu) perintah lain yang bisa juga digunakan untuk melakukan *back-up* dari *container* yang sedang berjalan, yaitu *export*. Perbedaan implementasinya sudah jelas, penerapan bergantung preferensi *SysAdmin* sendiri.

Untuk melakukan *back-up* dari *container*, silakan pilih dari daftar di mesin Anda.

```
yht@debian:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND   CREATED       STATUS
6d4dc52eab18   debian:jessie  "bash"    2 hours ago   Exited (0) 2
9906f0f91831   debian:jessie  "bash"    10 days ago   Exited (0) 9
yht@debian:~$ docker export 6d4dc52eab18 > jessie-export.tar
yht@debian:~$ ls -l *.tar
-rw------- 1 yht yht 129310720 Sep  8 16:45 jessie-backup.tar
-rw-r--r-- 1 yht yht 129297920 Sep  8 17:31 jessie-export.tar
yht@debian:~$
```

Untuk melakukan *restore* digunakan perintah *import*.

```
yht@debian:~$ docker import
"docker import" requires at least 1 argument(s).
See 'docker import --help'.

Usage:  docker import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]

Import the contents from a tarball to create a filesystem image
```

Dari galat, perintah ini membutuhkan argumen berkas dan *repository*. Ternyata perintah ini akan membuat *images* dari berkas *back-up*. Untuk sementara *repository* atau nama *images* dikosongkan, dan lihat apa yang terjadi.

```
yht@debian:~$ docker import jessie-export.tar
sha256:f2dc080f72f1ae886a8c8fe92441d8ea6207793e62026c0bb9ae6266a474
yht@debian:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND   CREATED       STATUS
6d4dc52eab18   debian:jessie  "bash"    2 hours ago   Exited (0) 2
9906f0f91831   debian:jessie  "bash"    10 days ago   Exited (0) 9
yht@debian:~$ docker images
REPOSITORY       TAG            IMAGE ID            CREATED
<none>           <none>         f2dc080f72f1        10 seconds ago
jessie-backup    latest         572ff98aa9c5        About an hour a
python           2.7.13         fa8e55b2235d        6 weeks ago
python           latest         968120d8cbe8        6 weeks ago
debian           latest         a20fd0d59cf1        6 weeks ago
debian           jessie         86baf4e8cde9        6 weeks ago
```

Apakah Anda menemukan dimana berkas yang di-*import* tadi berada? Anda melihat *images* dengan nama atau *images* ID **f2dc080f72f1**?

Penutup

Buku yang dibuat ini masih terdapat banyak kekurangan. Kami sadar masih banyak yang belum tercakup dalam buku ini maupun yang harus dibenahi. Oleh karena itu segala saran dan kritik atas penulisan buku ini sangat kami harapkan.

Segala saran dan kritik atas buku ini silakan disampaikan melalui Grup Telegram atau melalui pos-el penulis di yht@kalamangga.net dan mari berdiskusi.

Akhir kata, terima kasih telah mengunduh dan membaca buku ini. Semoga bermanfaat.