Docker Simplified: Panduan Langsung untuk Pemula Mutlak

oleh Shahzan

Apakah Anda berencana untuk memulai karir Anda di DevOps, atau Anda sudah menyukainya, jika Anda tidak memiliki Docker yang terdaftar di resume Anda, tidak diragukan lagi saatnya bagi Anda untuk memikirkannya, karena Docker adalah salah satu keterampilan penting bagi siapa saja yang berada di arena DevOps.

Dalam posting ini, saya akan mencoba yang terbaik untuk menjelaskan Docker dengan cara paling sederhana yang saya bisa.

Sebelum kita menyelam dalam-dalam dan mulai menjelajahi Docker, mari kita lihat topik apa yang akan kita bahas sebagai bagian dari panduan pemula ini.

Mari kita mulai dengan memahami, Apa itu Docker?

Secara sederhana, Docker adalah platform perangkat lunak yang menyederhanakan proses membangun, menjalankan, mengelola, dan mendistribusikan aplikasi. Hal ini dilakukan dengan virtualisasi sistem operasi komputer di mana ia diinstal dan berjalan.

Edisi pertama Docker dirilis pada tahun 2013.

Docker dikembangkan menggunakan bahasa pemrograman GO.

> *Melihat serangkaian fungsi yang kaya yang ditawarkan Docker, telah diterima secara luas oleh beberapa organisasi dan universitas terkemuka di dunia, seperti **Visa, PayPal, Cornell University dan Indiana University** (hanya untuk beberapa nama) untuk menjalankan dan mengelola aplikasi mereka menggunakan Docker.*

Sekarang mari kita coba memahami masalahnya, dan solusi yang ditawarkan Docker.

Masalahnya

Katakanlah Anda memiliki tiga aplikasi berbasis Python yang berbeda yang Anda rencanakan untuk di-host pada satu server (yang bisa berupa mesin fisik atau virtual).

Masing-masing aplikasi ini menggunakan versi Python yang berbeda, serta pustaka dan dependensi terkait, berbeda dari satu aplikasi ke aplikasi lainnya.

Karena kita tidak dapat memiliki versi Python yang berbeda diinstal pada mesin yang sama, ini mencegah kita dari hosting ketiga aplikasi pada komputer yang sama.

Solusinya

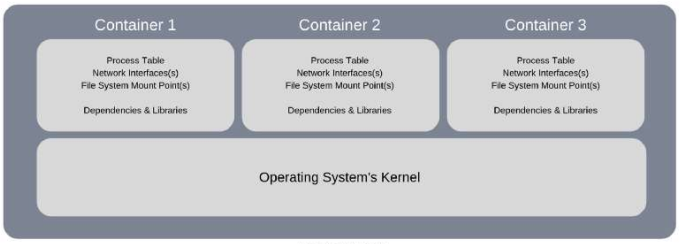
Mari kita lihat bagaimana kita bisa memecahkan masalah ini tanpa memanfaatkan Docker. Dalam skenario seperti itu, kita bisa memecahkan masalah ini baik dengan memiliki tiga mesin fisik, atau satu mesin fisik, yang cukup kuat untuk menjadi tuan rumah dan menjalankan tiga mesin virtual di atasnya.

Kedua opsi akan memungkinkan kami untuk menginstal versi Python yang berbeda pada masing-masing mesin ini, bersama dengan dependensi terkaitnya.

Terlepas dari solusi yang kami pilih, biaya yang terkait dengan pengadaan dan pemeliharaan perangkat keras cukup mahal.

Sekarang, mari kita lihat bagaimana Docker bisa menjadi solusi yang efisien dan hemat biaya untuk masalah ini.

Untuk memahami hal ini, kita perlu melihat bagaimana tepatnya fungsi Docker.



Mesin tempat Docker diinstal dan berjalan biasanya disebut sebagai Host atau Host Docker dalam istilah sederhana.

Jadi, setiap kali Anda berencana untuk menyebarkan aplikasi pada host, itu akan membuat entitas logis di atasnya untuk meng-host aplikasi itu. Dalam terminologi Docker, kita menyebut entitas logis ini sebagai Container atau Docker Container agar lebih tepat.

Docker Container tidak memiliki sistem operasi yang diinstal dan berjalan di atasnya. Tapi itu akan memiliki salinan virtual dari tabel proses, antarmuka jaringan (s), dan sistem file mount point (s). Ini telah diwarisi dari sistem operasi host di mana wadah di-host dan berjalan.

Sedangkan kernel dari sistem operasi host dibagi di semua kontainer yang berjalan di atasnya.

Hal ini memungkinkan setiap wadah untuk diisolasi dari yang lain hadir pada host yang sama. Dengan demikian mendukung beberapa kontainer dengan persyaratan aplikasi yang berbeda dan dependensi untuk berjalan pada host yang sama, selama mereka memiliki persyaratan sistem operasi yang sama.

Untuk memahami bagaimana Docker telah bermanfaat dalam memecahkan masalah ini, Anda perlu merujuk ke bagian berikutnya, yang membahas kelebihan dan kekurangan menggunakan Docker.

Singkatnya, Docker akan memvirtualisasi sistem operasi host di mana ia diinstal dan berjalan, daripada memvirtualisasi komponen perangkat keras.

Kelebihan dan Kekurangan Menggunakan Docker

Keuntungan menggunakan Docker

Beberapa manfaat utama menggunakan Docker tercantum di bawah ini:

- Docker mendukung beberapa aplikasi dengan persyaratan aplikasi dan dependensi yang berbeda, untuk dihosting bersama pada host yang sama, selama mereka memiliki persyaratan sistem operasi yang sama.
- Penyimpanan Dioptimalkan. Sejumlah besar aplikasi dapat di-host pada host yang sama, karena kontainer biasanya berukuran beberapa megabyte dan mengkonsumsi sangat sedikit ruang disk.
- Ketahanan. Sebuah wadah tidak memiliki sistem operasi yang terpasang di atasnya. Dengan demikian, ia mengkonsumsi sangat sedikit memori dibandingkan dengan mesin virtual (yang akan memiliki sistem operasi lengkap yang diinstal dan berjalan di atasnya).

Ini juga mengurangi waktu bootup menjadi hanya beberapa detik, dibandingkan dengan beberapa menit yang diperlukan untuk mem-boot mesin virtual.
- Mengurangi biaya. Docker kurang menuntut ketika datang ke perangkat keras yang diperlukan untuk menjalankannya.

Kekurangan menggunakan Docker

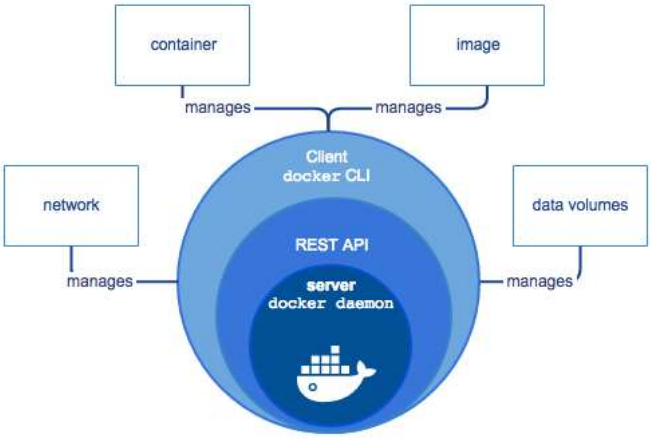
- Aplikasi dengan persyaratan sistem operasi yang berbeda tidak dapat dihosting bersama di Host Docker yang sama. Sebagai contoh, katakanlah kita memiliki 4 aplikasi yang berbeda, dari mana 3 aplikasi memerlukan sistem operasi berbasis Linux dan aplikasi lainnya memerlukan sistem operasi berbasis Windows. Dalam skenario seperti itu, 3 aplikasi yang memerlukan sistem operasi berbasis Linux dapat di-host pada satu Host Docker, sedangkan aplikasi yang membutuhkan sistem operasi berbasis Windows perlu di-host pada Host Docker yang berbeda.

Komponen Inti dari Docker

Docker Engine adalah salah satu komponen inti dari Docker. Hal ini bertanggung jawab untuk fungsi keseluruhan dari platform Docker.

Docker Engine adalah aplikasi berbasis client-server dan terdiri dari 3 komponen utama.

1. Peladen
2. REST API
3. Klien



Sumber gambar: https://docs.docker.com

Server menjalankan daemon yang dikenal sebagai **dockerd (Docker Daemon),**yang tidak lain adalah sebuah proses. Hal ini bertanggung jawab untuk membuat dan mengelola Gambar Docker, Kontainer, Jaringan dan Volume pada platform Docker.

REST API menentukan bagaimana aplikasi dapat berinteraksi dengan Server, dan menginstruksikannya untuk menyelesaikan pekerjaan mereka.

Klien tidak lain adalah antarmuka baris perintah, yang memungkinkan pengguna untuk berinteraksi dengan **Docker** menggunakan perintah.

Terminologi Docker

Mari kita lihat sekilas beberapa terminologi yang terkait dengan Docker.

Docker Images dan **Docker Containers** adalah dua hal penting yang akan Anda temui setiap hari saat bekerja dengan **Docker.**

Secara sederhana, **Gambar Docker** adalah template yang berisi aplikasi, dan semua dependensi yang diperlukan untuk menjalankan aplikasi itu pada Docker.

Di sisi lain, seperti yang dinyatakan sebelumnya, **Wadah Docker** adalah entitas logis. Dalam istilah yang lebih tepat, ini adalah contoh yang berjalan dari Gambar Docker.

Apa itu Docker Hub?

Docker Hub adalah repositori online resmi di mana Anda dapat menemukan semua Gambar Docker yang tersedia untuk kami gunakan.

Docker Hub juga memungkinkan kami untuk menyimpan dan mendistribusikan gambar kustom kami juga jika kami ingin melakukannya. Kita juga bisa membuatnya baik publik atau swasta, berdasarkan kebutuhan kita.

Harap Dicatat: Pengguna gratis hanya diperbolehkan untuk menyimpan satu Gambar Docker sebagai pribadi. Jika kita ingin menyimpan lebih dari satu Gambar Docker sebagai pribadi, kita perlu berlangganan paket berlangganan berbayar.

Edisi Docker

Docker tersedia dalam 2 edisi yang berbeda, seperti yang tercantum di bawah ini:

- **Edisi Komunitas (CE)**
- **Edisi Perusahaan (EE)**

Community Edition cocok untuk pengembang individu dan tim kecil. Ini menawarkan fungsionalitas terbatas, dibandingkan dengan Enterprise Edition.

Enterprise Edition, di sisi lain, cocok untuk tim besar dan untuk menggunakan Docker di lingkungan produksi.

Enterprise Edition selanjutnya dikategorikan ke dalam tiga edisi yang berbeda, seperti yang tercantum di bawah ini:

- **Edisi Dasar**
- **Edisi Standar**
- **Edisi Lanjutan**

Menginstal Docker

Satu hal terakhir yang perlu kita ketahui sebelum kita melanjutkan dan membuat tangan kita kotor dengan Docker sebenarnya adalah memasang Docker.

Di bawah ini adalah tautan ke panduan instalasi Docker CE resmi. Anda dapat mengikuti panduan ini untuk menginstal Docker di mesin Anda, karena sederhana dan mudah.

Ingin melewatkan instalasi dan langsung menuju berlatih Docker?

Untuk berjaga-jaga jika Anda merasa terlalu malas untuk menginstal Docker, atau Anda tidak memiliki cukup sumber daya yang tersedia di komputer Anda, Anda tidak perlu khawatir - inilah solusi untuk masalah Anda.

Anda dapat menuju ke Play with Docker,yang merupakan taman bermain online untuk Docker. Hal ini memungkinkan pengguna untuk berlatih perintah Docker segera, tanpa harus menginstal apa pun pada mesin Anda. Bagian yang terbaik adalah mudah digunakan dan tersedia secara gratis.

Perintah Docker

Sekarang saatnya untuk membuat tangan kita kotor dengan perintah Docker, yang kita semua telah menunggu sampai sekarang.

docker membuat

Perintah pertama yang akan kita lihat adalah **docker membuat** perintah.

Perintah ini memungkinkan kita untuk membuat wadah baru.

Sintaks untuk perintah ini seperti yang ditunjukkan di bawah ini:

```
docker create [options] IMAGE [commands] [arguments]
```

Harap Dicatat: Apa pun yang tertutup dalam tanda kurung siku adalah opsional. Ini berlaku untuk semua perintah yang akan Anda lihat di panduan ini.

Beberapa contoh penggunaan perintah ini ditunjukkan di bawah ini:

```
$ docker create fedora
02576e880a2ccbb4ce5c51032ea3b3bb8316e5b626861fc87d28627c810af03
```

Dalam contoh di atas, perintah docker create akan membuat wadah baru menggunakan gambar Fedora terbaru.

Sebelum membuat wadah, ia akan memeriksa apakah gambar resmi terbaru dari Fedora tersedia di Docker Host atau tidak. Jika gambar terbaru tidak tersedia di Docker Host, maka akan melanjutkan dan mengunduh gambar Fedora dari Docker Hub sebelum membuat wadah. Jika gambar Fedora sudah ada di Docker Host, itu akan menggunakan gambar itu dan membuat wadah.

Jika wadah berhasil dibuat, Docker akan mengembalikan ID kontainer. Misalnya, dalam contoh di atas
02576e880a2ccbb4ce5c51032ea3b3bb8316e5b626861fc87d28627c810af03
adalah ID kontainer yang dikembalikan oleh Docker.

Setiap wadah memiliki ID kontainer yang unik. Kami mengacu pada wadah menggunakan ID kontainer untuk melakukan berbagai operasi pada wadah, seperti memulai, menghentikan, memulai ulang, dan sebagainya.

Sekarang, mari kita merujuk ke contoh lain dari docker membuat perintah, yang memiliki opsi dan perintah yang diteruskan ke sana.

```
$ docker create -t -i ubuntu bash
30986b73dc0022dbba81648d9e35e6e866b4356f026e75660460c3474f1ca005
```

Dalam contoh di atas, perintah docker create membuat wadah menggunakan gambar Ubuntu (Seperti yang dinyatakan sebelumnya, jika gambar tidak tersedia di Host Docker, itu akan melanjutkan dan mengunduh gambar terbaru dari Docker Hub sebelum membuat wadah).

Opsi -t dan -i menginstruksikan Docker untuk mengalokasikan terminal ke kontainer sehingga pengguna dapat berinteraksi dengan kontainer. Ini juga menginstruksikan Docker untuk melaksanakan perintah bash setiap kali wadah dimulai.

docker ps

Perintah berikutnya yang akan kita lihat adalah perintah **docker ps.**

Perintah **docker ps** memungkinkan kita untuk melihat semua kontainer yang berjalan di Host Docker.

```
$ docker ps
CONTAINER ID IMAGE  COMMAND CREATED        STATUS            PORTS NAMES30986b73dc00 ubuntu "bash"  45 minutes ago
```

Ini hanya menampilkan wadah yang saat ini berjalan di Docker Host.

Jika Anda ingin melihat semua kontainer yang dibuat di Host Docker ini, terlepas dari statusnya saat ini, seperti apakah mereka berjalan atau keluar, maka Anda perlu menyertakan opsi -a, yang pada gilirannya akan menampilkan semua kontainer yang dibuat di Host Docker ini.

```
$ docker ps -a
CONTAINER ID IMAGE  COMMAND     CREATED           STATUS       PORTS NAMES30986b73dc00 ubuntu “bash”      About an
```

Sebelum kita melanjutkan lebih jauh, mari kita coba untuk memecahkan kode dan memahami output dari perintah **docker ps.**

ID KONTAINER: String unik yang terdiri dari karakter alfanumerik, terkait dengan setiap wadah.

GAMBAR: Nama Gambar Docker yang digunakan untuk membuat wadah ini.

PERINTAH: Setiap perintah khusus aplikasi yang perlu dijalankan ketika kontainer dimulai.

DIBUAT: Ini menunjukkan waktu berlalu sejak wadah ini telah dibuat.

STATUS: Ini menunjukkan status kontainer saat ini, bersama dengan waktu berlalu, dalam keadaan sekarang.

Jika wadah berjalan, itu akan ditampilkan sebagai Up bersama dengan periode waktu berlalu (misalnya, Up Sekitar satu jam atau Up 3 menit).

Jika kontainer dihentikan, maka akan ditampilkan sebagai Keluar diikuti oleh kode status keluar dalam tanda kurung bulat, bersama dengan periode waktu berlalu (misalnya, Keluar (0) 3 minggu yang lalu atau Keluar (137) 15 detik yang lalu, di mana 0 dan 137 adalah kode keluar).

PORT: Ini menampilkan pemetaan port yang ditentukan untuk kontainer.

NAMA: Selain CONTAINER ID, setiap kontainer juga diberi nama yang unik. Kita dapat merujuk ke wadah baik menggunakan ID kontainer atau nama yang unik. Docker secara otomatis menetapkan nama konyol unik untuk setiap wadah yang dibuatnya. Tetapi jika Anda ingin menentukan nama Anda sendiri ke wadah, Anda dapat melakukannya dengan memasukkan opsi (nama tanda hubung ganda) ke docker create atau docker run (kita akan melihat perintah docker run nanti).– –
name

Saya harap ini memberi Anda pemahaman yang lebih baik tentang output dari perintah docker ps.

docker mulai

Perintah berikutnya yang akan kita lihat, adalah perintah **awal docker.**

Perintah ini memulai setiap wadah yang berhenti(s).

Sintaks untuk perintah ini seperti yang ditunjukkan di bawah ini:
```
docker start [options] CONTAINER ID/NAME [CONTAINER ID/NAME…]
```

Kita dapat memulai wadah baik dengan menentukan beberapa karakter unik pertama dari ID kontainernya atau dengan menentukan namanya.

Beberapa contoh penggunaan perintah ini ditunjukkan di bawah ini:
```
$ docker start 30986
```

Dalam contoh di atas, Docker memulai wadah dimulai dengan ID kontainer 30986.
```
$ docker start elated_franklin
```

Sedangkan dalam contoh ini, Docker memulai wadah bernama elated_franklin.

docker berhenti

Perintah berikutnya dalam daftar adalah perintah **docker stop.**

Perintah ini menghentikan setiap kontainer yang berjalan(s).

Sintaks untuk perintah ini seperti yang ditunjukkan di bawah ini:
```
docker stop [options] CONTAINER ID/NAME [CONTAINER ID/NAME…]
```

Hal ini mirip dengan perintah docker start.

Kita dapat menghentikan wadah baik dengan menentukan beberapa karakter unik pertama dari ID kontainer atau dengan menentukan namanya.

Beberapa contoh penggunaan perintah ini ditunjukkan di bawah ini:
```
$ docker stop 30986
```

Dalam contoh di atas, Docker akan menghentikan wadah dimulai dengan ID kontainer 30986.
```
$ docker stop elated_franklin
```

Sedangkan dalam contoh ini, Docker akan menghentikan wadah bernama elated_franklin.

docker restart

Perintah berikutnya yang akan kita lihat adalah perintah **restart docker.**

Perintah ini memulai ulang setiap kontainer yang sedang berjalan.

Sintaks untuk perintah ini seperti yang ditunjukkan di bawah ini:
```
docker restart [options] CONTAINER ID/NAME [CONTAINER ID/NAME…]
```

Kita dapat me-restart wadah baik dengan menentukan beberapa karakter unik pertama dari ID kontainer atau dengan menentukan namanya.

Beberapa contoh penggunaan perintah ini ditunjukkan di bawah ini:
```
$ docker restart 30986
```

Pada contoh di atas, Docker akan memulai kembali kontainer yang dimulai dengan ID kontainer 30986.
```
$ docker restart elated_franklin
```

Sedangkan dalam contoh ini, Docker akan memulai ulang wadah bernama elated_franklin.

docker run

Perintah berikutnya yang akan kita lihat adalah perintah **docker run.**

Perintah ini pertama-tama membuat wadah, dan kemudian memulai wadah. Singkatnya, perintah ini adalah kombinasi dari docker create dan perintah docker start.

Sintaks untuk perintah ini seperti yang ditunjukkan di bawah ini:
```
docker run [options] IMAGE [commands] [arguments]
```

Ini memiliki sintaks yang mirip dengan perintah docker create.

Beberapa contoh penggunaan perintah ini ditunjukkan di bawah ini:
```
$ docker run ubuntu
30fa018c72682d78cf168626b5e6138bb3b3ae23015c5ec4bbcc2a088e67520
```

Pada contoh di atas, Docker akan membuat wadah menggunakan gambar Ubuntu terbaru dan kemudian segera memulai wadah.

Jika kita menjalankan perintah di atas, itu akan memulai wadah dan segera menghentikannya - kita tidak akan mendapatkan kesempatan untuk berinteraksi dengan wadah sama sekali.

Jika kita ingin berinteraksi dengan wadah, maka kita perlu menentukan opsi: -it (tanda hubung diikuti oleh i dan t) ke perintah docker run menyajikan kita dengan terminal, yang menggunakannya kita dapat berinteraksi dengan wadah dengan mengetikkan perintah yang sesuai. Di bawah ini adalah contoh yang sama.
```
$ docker run -it ubuntu
root@e4e633428474:/#
```

Untuk keluar dari kontainer, Anda perlu mengetik keluar di terminal.

docker rm

Beralih ke perintah berikutnya — jika kami ingin menghapus kontainer, kami menggunakan perintah **docker rm.**

Sintaks untuk perintah ini seperti yang ditunjukkan di bawah ini:
```
docker rm [options] CONTAINER ID/NAME [CONTAINER ID/NAME...]
```

Beberapa contoh penggunaan perintah ini ditunjukkan di bawah ini:
```
$ docker rm 30fa elated_franklin
```

Dalam contoh di atas, kami menginstruksikan Docker untuk menghapus 2 kontainer dalam satu perintah. Kontainer pertama yang akan dihapus ditentukan menggunakan ID kontainernya, dan kontainer kedua yang akan dihapus ditentukan menggunakan namanya.

Harap Dicatat: Kontainer harus dalam keadaan berhenti agar dapat dihapus.

gambar docker

gambar docker adalah perintah berikutnya dalam daftar.

Perintah ini mencantumkan semua Gambar Docker yang ada di Host Docker Anda.
```
$ docker images
REPOSITORY  TAG      IMAGE          CREATED        SIZEmysql       latest   7bb2586065cd   38 hours ago   477MBhttp
```

Mari kita decode output dari perintah **gambar docker.**

REPOSITORI: Ini mewakili nama unik dari Gambar Docker.

TAG: Setiap gambar dikaitkan dengan tag yang unik. Tag pada dasarnya mewakili versi gambar.

Tag biasanya diwakili baik menggunakan kata atau set angka atau kombinasi karakter alfanumerik.

ID GAMBAR: String unik yang terdiri dari karakter alfanumerik, terkait dengan setiap gambar.

DIBUAT: Ini menunjukkan waktu berlalu sejak gambar ini telah dibuat.

UKURAN: Ini menunjukkan ukuran gambar.

docker rmi

Perintah berikutnya dalam daftar adalah perintah **docker rmi.**

Perintah **docker rmi** memungkinkan kita untuk menghapus gambar (s) dari Host Docker.

Sintaks untuk perintah ini seperti yang ditunjukkan di bawah ini:
```
docker rmi [options] IMAGE NAME/ID [IMAGE NAME/ID...]
```

Beberapa contoh penggunaan perintah ini ditunjukkan di bawah ini:
```
docker rmi mysql
```

Perintah di atas menghapus gambar bernama mysql dari Host Docker.
```
docker rmi httpd fedora
```

Perintah di atas menghapus gambar bernama httpd dan fedora dari Docker Host.
`docker rmi 94e81`

Perintah di atas menghapus gambar dimulai dengan ID gambar 94e81 dari Host Docker.
`docker rmi ubuntu:trusty`

Perintah di atas menghapus gambar bernama ubuntu, dengan tag terpercaya dari Host Docker.

Ini adalah beberapa perintah Dasar Docker yang akan Anda lihat. Ada banyak lagi perintah Docker untuk dijelajahi.

Wrap-up

Containerization baru-baru ini mendapat perhatian yang layak, meskipun telah ada sejak lama. Beberapa perusahaan teknologi top seperti Google, Amazon Web Services (AWS), Intel, Tesla, dan Juniper Networks memiliki versi mesin kontainer khusus mereka sendiri. Mereka sangat bergantung pada mereka untuk membangun, menjalankan, mengelola, dan mendistribusikan aplikasi mereka.

> ***Docker*** *adalah mesin containerization yang sangat kuat, dan memiliki banyak hal untuk ditawarkan ketika datang untuk membangun, menjalankan, mengelola dan mendistribusikan aplikasi Anda secara efisien.*

Anda baru saja melihat Docker pada tingkat yang sangat tinggi. Ada banyak lagi yang harus dipelajari tentang Docker, seperti:

- Perintah Docker (perintah yang lebih kuat)
- Gambar Docker (Buat gambar kustom Anda sendiri)
- Docker Networking (Setup dan mengkonfigurasi jaringan)
- Layanan Docker (Mengelompokkan kontainer yang menggunakan gambar yang sama)
- Docker Stack (Layanan pengelompokan yang diperlukan oleh aplikasi)
- Docker Compose (Alat untuk mengelola dan menjalankan beberapa kontainer)
- Docker Swarm (Pengelompokan dan mengelola satu atau lebih mesin tempat docker berjalan)
- Dan masih banyak lagi...

Jika Anda telah menemukan Docker menarik, dan tertarik untuk mempelajari lebih lanjut tentang hal itu, maka saya akan merekomendasikan agar Anda mendaftar di kursus yang tercantum di bawah ini. Saya menemukan mereka sangat informatif dan langsung ke intinya.

Jika Anda seorang pemula mutlak, maka saya sarankan Anda mendaftar di kursus ini,yang telah dirancang untuk pemula.

Jika Anda memiliki pengetahuan yang baik tentang Docker, dan cukup percaya diri dengan hal-hal dasar dan ingin memperluas pengetahuan Anda, maka saya sarankan Anda harus mendaftar ke kursus ini,yang lebih ditujukan untuk topik lanjutan yang terkait dengan Docker.

Docker adalah keterampilan yang terbukti di masa depan dan hanya mengambil momentum.

Menginvestasikan waktu dan uang Anda untuk mempelajari Docker tidak akan menjadi sesuatu yang akan Anda tolak.

> *Semoga artikel ini informatif. Jangan ragu untuk membaginya. Ini benar-benar sangat berarti bagiku.*

Sebelum anda mengucapkan selamat tinggal...

Mari kita tetap berhubungan, klik di sini untuk memasukkan alamat email Anda (Gunakan tautan ini jika widget di atas tidak muncul di layar Anda).

Terima kasih banyak telah meluangkan waktu berharga Anda untuk membaca artikel ini.