[*Docker*](https://www.docker.com/)*adalah sebuah project open source yang ditujukan untuk developer atau sysadmin untuk membangun, mengemas dan menjalankan aplikasi dimana pun di dalam sebuah container.* Docker berfungsi sebagai virtualisasi sebuah sistem operasi atau sebuah server atau sebuah web server atau bahkan sebuah database server, dimana dengan menggunakan virtualisasi ini, diharapkan developer dapat mengembangkan aplikasi sesuai dengan spesifikasi server

**Docker Images**

Docker images adalah sebuah template yang bersifat read only. Template ini sebenarnya adalah sebuah OS atau OS yang telah diinstall berbagai aplikasi. Docker images berfungsi untuk membuat docker container, dengan hanya 1 docker images kita dapat membuat banyak docker container.

**Docker Container**

Docker container bisa dikatakan sebagai sebuah folder, dimana docker container ini dibuat dengan menggunakan docker daemon. Setiap docker container disimpan maka akan terbentuk layer baru tepat diatas docker images atau base image diatasnya. Contohnya misalkan kita menggunakan image ubuntu, kemudian kita membuat sebuah container dari image ubuntu tersebut dengan nama ubuntuku, kemudian kita lakukan instalasi sebuah software misalnya nginx maka secara otomatis container ubuntuku akan berada diatas layer image atau base image ubuntu. Anda dapat membuat banyak docker container dari 1 docker images. Docker container ini nantinya dapat dibuild sehingga akan menghasilkan sebuah docker images, dan docker images yang dihasilkan dari docker container ini dapat kita gunakan kembali untuk membuat docker container yang baru.

**Docker Registry**

Docker registry adalah kumpulan docker image yang bersifat private maupun public yang dapat anda akses di [docker hub](https://hub.docker.com/). Dengan menggunakan docker registry, anda dapat menggunakan docker image yang telah dibuat oleh developer yang lain, sehingga mempermudahkan kita dalam pengembangan aplikasi.

[*Docker compose*](https://docs.docker.com/compose/)*berfungsi untuk menjalankan container docker secara bersamaan.*

docker compose ini sangat berguna ketika aplikasi kita terpisah - pisah pada komputer yang berbeda, contohnya adalah aplikasi yang dibuat berada pada 1 container sedangkan database yang akan digunakan oleh aplikasi tersebut berada pada container yang lain. Ketika menggunakan docker compose maka kita dapat menjalankan kedua container tersebut secara bersamaan dan bahkan kita dapat melakukan link ke container yang kita inginkan.

Compose adalah alat untuk mendefinisikan dan menjalankan aplikasi Docker multi-kontainer. Dengan Compose, Anda menggunakan file Compose untuk mengonfigurasi layanan aplikasi Anda. Kemudian, menggunakan satu perintah, Anda membuat dan memulai semua layanan dari konfigurasi Anda. Dengan docker compose kita bisa menyimpan konfigurasi dalam file, berarti semua perubahan dependency service, seperti versi database dan service lain dapat dimasukkan dalam VCS (Version Control System). Dengan VCS kita dapat lebih mudah men-debug jika terjadi error pada software.

Jalankan Docker Compose

docker-compose up -d

Install dependency dengan menjalankan perintah `composer install` di dalam docker container

$ docker-compose run --rm web composer install

Wait, tadi kita jalankan composer install dalam docker container tapi di directory local terdapat directory baru `vendor`? Itu karena directory local sudah dimount ke docker, jadi semua perubahan file pada container juga terjadi pada local directory, dan sebaliknya.

volumes:  
 - .:/app

Untuk melihat output yang dikeluarkan jalankan perintah

docker-compose logs -f

**Menambahkan Service**

Selanjutnya kita akan menambahkan service redis untuk presisten datanya. Tambahkan code berikut dalam block services di docker-compose.yml

redis:  
 image: redis:alpine

Tambahkan environment variable untuk service web pada block ‘environment’

web:  
 build: .  
 ports:  
 - 9090:8080  
 volumes:  
 - .:/app  
 environment:  
 - REDIS\_HOST=redis

Restart docker-compose

$ docker-compose up -d

Install library redis, di dalam docker container

$ docker-compose run --rm web composer require predis/predis

Edit index.php, tambahkan line berikut

<?php  
...Predis\Autoloader::register();  
$redis = new Predis\Client([  
 'host' => getenv('REDIS\_HOST')  
]);$app->get('/count', function($request, $response) use ($redis) {  
 $redis->incr('counter');  
 $counter = $redis->get('counter');  
 $response->getBody()->write($counter); return $response;  
});$app-run();

Lihat hasil

$ curl <http://localhost:9090/count>  
1$ curl <http://localhost:9090/count>  
2$ curl <http://localhost:9090/count>  
3

Dengan docker compose kita cukup mendeskripsikan environment yang kita inginkan pada file docker-compose.yml, proses bootstraping secara otomatis di handle oleh docker. File konfigurasi juga dapat dimasukkan dalam CSV sehingga perubahan dependency service bisa dipantau.

**Instalasi Docker**

Untuk melakukan instalasi docker, silahkan update terlebih dahulu repository ubuntu anda dengan perintah.

sudo apt update

Kemudian lakukan update CA certificates dengan perintah.

sudo apt install apt-transport-https ca-certificates curl software-properties-common

Kemudian tambahkan GPG key dengan perintah.

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Lalu silahkan lakukan verifikasi fingerprint dengan perintah

sudo apt-key fingerprint 0EBFCD88

Jika berhasil maka akan muncul output seperti berikut

pub 4096R/0EBFCD88 2017-02-22

Key fingerprint = 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88

uid Docker Release (CE deb) <docker@docker.com>

sub 4096R/F273FCD8 2017-02-22

Lalu tambahan repo docker seperti berikut

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

Kemudian silahkan lakukan update kembali dengan perintah.

sudo apt update

Kemudian silahkan hapus docker yang lama jika anda pernah melakukan instalasi docker versi lama dengan perintah.

sudo apt purge lxc-docker

Lalu silahkan jalankan perintah berikut untuk melakukan instalasi docker.

sudo apt install docker-ce

lalu lakukan pengecekan status service docker dengan perintah.

sudo systemctl status docker

jika berhasil maka akan muncul output seperti berikut.

Agar kita dapat menggunakan docker tanpa sudo maka kita harus melakukan beberapa konfigurasi. Silahkan buat docker group dengan perintah berikut.

sudo groupadd docker

kemudian tambahkan user ke docker group dengan perintah.

sudo usermod -aG docker rizki

silahkan ganti rizki dengan user linux anda. Lalu silahkan restart komputer anda. Kita akan melakukan test docker dengan perintah.

docker run hello-world

Jika berhasil maka anda akan melihat output seperti gambar berikut.

### Instalasi Docker Compose

Untuk melakukan instalasi docker compose silahkan jalankan perintah berikut untuk melakukan akses root.

sudo -s

kemudian jalankan perintah berikut.

curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

Setelah selesai lalu beri hak akses eksekusi dengan perintah.

chmod +x /usr/local/bin/docker-compose

Lalu agar docker compose dapat diakses tanpa root, silahkan jalankan perintah berikut.

sudo chmod -R 777 /usr/local/bin/docker-compose

Kemudian lakukan pengecekan docker compose dengan perintah

docker-compose -version

Jika berhasil maka akan muncul output seperti berikut.

docker-compose version 1.22.0, build 9e633ef

## Membangun Image

Ada dua cara untuk membangun image. Anda dapat memodifikasi kontainer yang ada dan kemudian melakukannya sebagai image baru, atau Anda dapat menulis Dockerfile dan membangun image. Kita akan melakukan keduanya dan menjelaskan pro dan kontra.

### Membangun Manual

Dengan membangun manual, Anda memperlakukan kontainer Anda seperti komputer biasa. Anda menginstal paket, Anda menulis file, dan ketika itu semua dikatakan dan dilakukan, Anda melakukan commit dan berakhir dengan image baru yang Anda gunakan sebagai template untuk membuat lebih banyak kontainer identik atau bahkan dasar untuk image lain.

Mari kita mulai dengan [image alpine](https://hub.docker.com/_/alpine/), yang merupakan image yang sangat kecil dan sederhana berbasis Alpine Linux. Kita dapat menjalankannya dalam modus interaktif untuk masuk ke sebuah shell. Tujuan kami adalah untuk menambahkan sebuah file bernama "yeah" yang berisi teks "it works!" ke direktori root dan kemudian membuat image baru darinya yang disebut "yeah-alpine".

Kita mulai. Bagus, kami sudah dalam dir root. Mari kita lihat apa yang ada di sana.

|  |  |
| --- | --- |
| 1  2  3 | > docker run -it alpine /bin/sh  / # ls  bin      dev      etc      home     lib      linuxrc  media    mnt      proc     root     run      sbin     srv      sys      tmp      usr      var |

Editor apa tersedia? Tidak ada vim, tidak ada nano?

|  |  |
| --- | --- |
| 1  2  3  4 | / # vim  /bin/sh: vim: not found  / # nano  /bin/sh: nano: not found |

Oh, baiklah. Kami hanya ingin membuat file:

|  |  |
| --- | --- |
| 1  2  3 | / # echo "it works!" > yeah  / # cat yeah  it works! |

Saya keluar dari shell interaktif, dan saya bisa melihat kontainer yang bernama "vibrant\_spenc" dengan docker ps --all. Flag --all penting karena kontainer sudah tidak berjalan lagi.

|  |  |
| --- | --- |
| 1  2  3 | > docker ps --all  CONTAINER ID IMAGE  COMMAND   CREATED       STATUS NAMES  c8faeb05de5f alpine "/bin/sh" 6 minutes ago Exited vibrant\_spence |

Di sini, saya membuat image baru dari kontainer "vibrate\_spence". Saya menambahkan pesan commit "mine, mine, mine" untuk pengukuran yang baik.

|  |  |
| --- | --- |
| 1  2 | > docker commit -m "mine, mine, mine" vibrant\_spence yeah-alpine  sha256:e3c98cd21f4d85a1428...e220da99995fd8bf6b49aa |

Mari kita periksa. Yep, ada image baru, dan dalam history-nya, Anda dapat melihat lapisan baru dengan komentar "mine, mine, mine".

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | > docker images  REPOSITORY       TAG    IMAGE ID      SIZE  yeah-alpine      latest e3c98cd21f4d  4.8 MB  python           latest 775dae9b960e  687 MB  d4w/nsenter      latest 9e4f13a0901e  83.8 kB  ubuntu-with-ssh  latest 87391dca396d  221 MB  ubuntu           latest bd3d4369aebc  127 MB  hello-world      latest c54a2cc56cbb  1.85 kB  alpine           latest 4e38e38c8ce0  4.8 MB  nsqio/nsq        latest 2a82c70fe5e3  70.7 MB    > docker history yeah-alpine  IMAGE        CREATED         SIZE   COMMENT  e3c98cd21f4d 40 seconds ago  66 B   mine, mine, mine  4e38e38c8ce0 7 months ago    4.8 MB |

Sekarang untuk ujian yang sebenarnya. Mari kita menghapus kontainernya dan membuat kontainer baru dari image. Hasil yang diharapkan adalah bahwa file "yeah" akan hadir dalam kontainer baru.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | > docker rm vibrant\_spence  vibrant\_spence    > docker run -it yeah-alpine /bin/sh  / # cat yeah  it works!  / # |

Apa yang bisa saya katakan? Ya, itu berhasil!

### Menggunakan Dockerfile

Menciptakan image dari memodifikasi kontainer itu keren, tapi ada tidak ada akuntabilitas. Sulit untuk melacak perubahan dan mengetahui manakah modifikasi tertentu. Cara disiplin untuk membuat image adalah membangun mereka menggunakan Dockerfile.

Dockerfile adalah file teks yang mirip dengan skrip shell, tapi mendukung beberapa perintah. Setiap perintah yang memodifikasi sistem file membuat lapisan baru. Pada bagian satu kita membahas pentingnya membagi image Anda menjadi lapisan dengan benar. Dockerfile adalah topik yang besar dengan dirinya.

Di sini, saya akan hanya menunjukkan beberapa perintah untuk membuat image yang lain, "oh-yeah-alpine", berdasarkan Dockerfile. Selain menciptakan file "yeah" yang terkenal, mari kita juga menginstal vim. Distribusi Linux alpine menggunakan sistem manajemen paket yang disebut "apk". Berikut adalah Dockerfile-nya:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | FROM alpine    # Copy the "yeah" file from the host  COPY yeah /yeah    # Update and install vim using apk  RUN apk update && apk add vim    CMD cat /yeah |

Image dasarnya alpine. Itu menyalin file "yeah" dari direktori host yang sama di mana Dockerfile berada (jalur konteks membangun). Kemudian, dia menjalankan apk update dan menginstal vim. Akhirnya, dia menetapkan perintah yang dijalankan ketika kontainer berjalan. Dalam kasus ini dia akan mencetak ke layar isi file "yeah".

Oke. Sekarang kita tahu apa yang kita hadapi, mari kita membangun ini. Pilihan "-t" mengatur repositori. Saya tidak menentukan tag, sehingga akan menggunakan default "latest".

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | >  docker build -t oh-yeah-alpine .  Sending build context to Docker daemon 3.072 kB  Step 1/4 : FROM alpine   ---> 4e38e38c8ce0  Step 2/4 : COPY yeah /yeah   ---> 1b2a228cc2a5  Removing intermediate container a6221f725845  Step 3/4 : RUN apk update && apk add vim   ---> Running in e2c0524bd792  fetch http://dl-cdn.alpinelinux.org/.../APKINDEX.tar.gz  fetch http://dl-cdn.alpinelinux.org.../x86\_64/APKINDEX.tar.gz  v3.4.6-60-gc61f5bf [http://dl-cdn.alpinelinux.org/alpine/v3.4/main]  v3.4.6-33-g38ef2d2 [http://dl-cdn.alpinelinux.org/.../v3.4/community]  OK: 5977 distinct packages available  (1/5) Installing lua5.2-libs (5.2.4-r2)  (2/5) Installing ncurses-terminfo-base (6.0-r7)  (3/5) Installing ncurses-terminfo (6.0-r7)  (4/5) Installing ncurses-libs (6.0-r7)  (5/5) Installing vim (7.4.1831-r2)  Executing busybox-1.24.2-r9.trigger  OK: 37 MiB in 16 packages   ---> 7fa4cba6d14f  Removing intermediate container e2c0524bd792  Step 4/4 : CMD cat /yeah   ---> Running in 351b4f1c1eb1   ---> e124405f28f4  Removing intermediate container 351b4f1c1eb1  Successfully built e124405f28f4 |

Terlihat baik. Mari kita pastikan image yang sudah dibuat:

|  |  |
| --- | --- |
| 1  2  3 | > docker images | grep oh-yeah    oh-yeah-alpine latest e124405f28f4 About a minute ago 30.5 MB |

Perhatikan bagaimana menginstal vim dan dependensinya membengkakkan ukuran kontainer dari 4.8MB basis image alpine basis menjadi 30,5MB yang masif!

Hal ini semua sangat bagus. Tapi apakah itu bekerja?

|  |  |
| --- | --- |
| 1  2 | > docker run oh-yeah-alpine  it works! |

Oh ya, ia bekerja!

Apabila Anda masih penasaran, mari kita pergi ke dalam kontainer dan memeriksa file "yeah" dengan vim kami yang baru saja diinstal.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | > docker run -it oh-yeah-alpine /bin/sh  / # vim yeah    it works!  ~  ~  .  .  .  ~  "yeah" 1L, 10C |

### Konteks Membangun dan file .dockerignore

Saya tidak memberitahu Anda, tapi awalnya ketika saya mencoba untuk membangun image oh-yeah-alpine, itu hanya tergantung selama beberapa menit. Masalahnya adalah bahwa saya hanya meletakkan Dockerfile di direktori home saya. Ketika Docker membangun sebuah image, itu pertama kali membungkus seluruh direktori di mana Dockerfile berada (termasuk sub-direktori) dan membuatnya tersedia untuk perintah COPY di Dockerfile.

Docker tidak mencoba untuk menjadi pintar dan menganalisis perintah COPY Anda. Dia hanya membungkus segala sesuatu. Perhatikan bahwa membangun konten tidak akan berakhir dalam image Anda, tapi itu akan memperlambat perintah membangun Anda jika konteks membangun Anda tidak perlu besar.

Dalam hal ini, saya hanya menyalin Dockerfile dan "yeah" ke sub-direktori dan menjalankan perintah docker build di sub-direktori itu. Tetapi kadang-kadang Anda memiliki pohon direktori rumit yang Anda ingin menyalin file dan sub-direktori tertentu dan mengabaikan lainnya. Masukkan ke file .dockerignore.

File ini memungkinkan Anda mengontrol persis apa yang terjadi ke dalam konteks membangun. Trik favorit saya adalah pertama mengecualikan segala sesuatu dan kemudian mulai memasukkan sedikit dan potongan-potongan yang saya butuhkan. Sebagai contoh, dalam kasus ini saya bisa membuat file .dockerignore berikut dan menyimpan file Docker dan "yeah" dalam direktori home saya:

|  |  |
| --- | --- |
| 1  2  3  4  5 | # Exclude EVERYTHING first  \*    # Now selectively include stuff  !yeah |

Ada tidak perlu memasukkan "Dockerfile" itu sendiri atau file ".dockerignore" dalam konteks membangun.

**Men-tag Image**

Men-tag image ini sangat penting jika Anda mengembangkan sistem berbasis microservices dan Anda menghasilkan banyak image yang kadang-kadang harus dikaitkan dengan satu sama lain. Anda dapat menambahkan tag sebanyak yang Anda inginkan ke image.

Anda telah melihat tag default "latest". Kadang-kadang, masuk akal untuk menambahkan tag lain, seperti "tested", "release-1.4" atau git commit yang sesuai dengan image.

Anda dapat men-tag image saat membangun atau setelahnya. Berikut adalah cara untuk menambahkan tag ke image yang sudah ada. Perhatikan bahwa sementara ini disebut tag, Anda juga dapat menetapkan repositori baru.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | > docker tag oh-yeah-alpine oh-yeah-alpine:cool-tag  > docker tag oh-yeah-alpine oh-yeah-alpine-2    > docker images | grep oh-yeah  oh-yeah-alpine-2 latest    e124405f28f4 30.5 MB  oh-yeah-alpine   cool-tag  e124405f28f4 30.5 MB  oh-yeah-alpine   latest    e124405f28f4 30.5 MB |

Anda juga dapat untag dengan menghapus image berdasarkan nama tag-nya. Ini sedikit menakutkan karena jika Anda menghapus tag terakhir karena ketidaksengajaan, Anda akan kehilangan image. Tapi jika Anda membangun image dari Dockerfile, Anda dapat hanya membangun ulang image.

|  |  |
| --- | --- |
| 1  2  3  4  5 | > docker rmi oh-yeah-alpine-2  Untagged: oh-yeah-alpine-2:latest    > docker rmi oh-yeah-alpine:cool-tag  Untagged: oh-yeah-alpine:cool-tag |

Jika saya mencoba untuk menghapus image yang di-tag yang terakhir tersisa, saya mendapatkan error karena digunakan oleh sebuah kontainer.

|  |  |
| --- | --- |
| 1  2  3  4  5 | > docker rmi oh-yeah-alpine    Error response from daemon: conflict: unable to remove repository  reference "oh-yeah-alpine" (must force) -  container a1443a7ca9d2 is using its referenced image e124405f28f4 |

Tetapi jika saya menghapus kontainer...

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | > docker rmi oh-yeah-alpine  Untagged: oh-yeah-alpine:latest  Deleted: sha256:e124405f28f48e...441d774d9413139e22386c4820df  Deleted: sha256:7fa4cba6d14fdf...d8940e6c50d30a157483de06fc59  Deleted: sha256:283d461dadfa6c...dbff864c6557af23bc5aff9d66de  Deleted: sha256:1b2a228cc2a5b4...23c80a41a41da4ff92fcac95101e  Deleted: sha256:fe5fe2290c63a0...8af394bb4bf15841661f71c71e9a    > docker images | grep oh-yeah |

Yap. Itu hilang. Tapi jangan khawatir. Kita dapat membangunnya kembali:

|  |  |
| --- | --- |
| 1  2  3  4 | > docker build -t oh-yeah-alpine .    > docker images | grep oh-yeah  oh-yeah-alpine latest 1e831ce8afe1 1 minutes ago 30.5 MB |

Yay, dia telah kembali. Dockerfile untuk kemenangan!

### Mem-publish Image

Mem-push image agak membutuhkan usaha. Pertama Anda perlu membuat akun di Docker Hub (atau registri lainnya). Selanjutnya, Anda log in. Kemudian Anda perlu untuk men-tag image yang ingin Anda push menurut nama akun Anda ("g1g1" dalam kasus saya).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | > docker login -u g1g1 -p <password>  Login Succeeded    > docker tag hello-world g1g1/hello-world    > docker images | grep hello    g1g1/hello-world latest 48b5124b2768 2 weeks ago 1.84 kB  hello-world      latest 48b5124b2768 2 weeks ago 1.84 kB |

Sekarang, saya dapat mem-push image yang di-tag g1g1/hello-world.

|  |  |
| --- | --- |
| 1  2  3  4 | > docker push g1g1/hello-world  The push refers to a repository [docker.io/g1g1/hello-world]  98c944e98de8: Mounted from library/hello-world  latest: digest: sha256:c5515758d4c5e...f6d12a1ac8d7 size: 524 |