Hướng dẫn cài đặt Docker

MỤC LỤC

Cài đặt docker engine

Cài bản stable mới nhất

- Các OS áp dụng: CentOS 7.3 64bit, Ubuntu 14.04 64bit, Ubuntu 16.04 64bit
- Đăng nhập với quyên `root` và thực hiện lệnh dưới để cài đặt. Đảm bảo máy có kết nối internet.`su -`

```
curl -sSL https://get.docker.com/ | sudo sh
```

  - Kết quả của lệnh trên như bên dưới

```
If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

    sudo usermod -aG docker your-user

Remember that you will have to log out and back in for this to take effect!

WARNING: Adding a user to the "docker" group will grant the ability to run
                 containers which can be used to obtain root privileges on the
                 docker host.
                 Refer to
https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface
                 for more information.
```

- Thực hiện lệnh dưới để phân quyền cho user hiện tại thuộc group `docker` `sudo usermod -aG docker `whoami``
- Kích hoạt docker sau khi cài đặt xong và cho phép khởi động cùng OS `systemctl start docker.service`

```
systemctl enable docker.service
```

- Kiểm tra trạng thái của docker sau khi khởi động `systemctl status docker.service`
  - Kết quả lệnh trên như dưới là ok `docker.service - Docker Application Container Engine`

```
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset:
disabled)
     Active: active (running) since Wed 2017-09-06 21:54:07 +07; 31s ago
         Docs: https://docs.docker.com
 Main PID: 2194 (dockerd)
     CGroup: /system.slice/docker.service
                     ├─2194 /usr/bin/dockerd
                     └─2200 docker-containerd -l
unix:///var/run/docker/libcontainerd/docker-containerd.sock --metrics-interval=0
--start-timeout 2m --state-dir /var/run/do...

Sep 06 21:54:05 docker1 dockerd[2194]: time="2017-09-06T21:54:05.324752580+07:00"
level=warning msg="failed to rename /var/lib/docker/tmp for background...hronously"
Sep 06 21:54:05 docker1 dockerd[2194]: time="2017-09-06T21:54:05.421102352+07:00"
level=warning msg="overlay: the backing xfs filesystem is formatted without d_ty...
Sep 06 21:54:05 docker1 dockerd[2194]: time="2017-09-06T21:54:05.541746640+07:00"
level=info msg="Graph migration to content-addressability took 0.00 seconds"
Sep 06 21:54:05 docker1 dockerd[2194]: time="2017-09-06T21:54:05.544930723+07:00"
level=info msg="Loading containers: start."
Sep 06 21:54:06 docker1 dockerd[2194]: time="2017-09-06T21:54:06.931709192+07:00"
level=info msg="Default bridge (docker0) is assigned with an IP addres...P address"
Sep 06 21:54:07 docker1 dockerd[2194]: time="2017-09-06T21:54:07.880341611+07:00"
level=info msg="Loading containers: done."
Sep 06 21:54:07 docker1 dockerd[2194]: time="2017-09-06T21:54:07.934858409+07:00"
level=info msg="Docker daemon" commit=8784753 graphdriver(s)=overlay v...17.07.0-ce
Sep 06 21:54:07 docker1 dockerd[2194]: time="2017-09-06T21:54:07.936237331+07:00"
level=info msg="Daemon has completed initialization"
Sep 06 21:54:07 docker1 dockerd[2194]: time="2017-09-06T21:54:07.983301268+07:00"
level=info msg="API listen on /var/run/docker.sock"
Sep 06 21:54:07 docker1 systemd[1]: Started Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
```

- Kiểm tra phiên bản của docker sau khi cài đặt, trong hướng dẫn này là bản 17.07.0-ce `docker version`
  - Kết quả như sau: `Client:`

```
 Version:      17.07.0-ce
 API version:  1.31
 Go version:   go1.8.3
 Git commit:   8784753
 Built:        Tue Aug 29 17:42:01 2017
 OS/Arch:      linux/amd64
```

```
Server:
 Version:      17.07.0-ce
 API version:  1.31 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   8784753
 Built:        Tue Aug 29 17:43:23 2017
 OS/Arch:      linux/amd64
 Experimental: false
```

- Kiểm tra xem docker đã hoạt động hay chưa `docker pull hello-world`
  - Kết quả như sau: `sh Using default tag: latest latest: Pulling from library/hello-world b04784fba78d: Pull complete Digest: sha256:f3b3b28a45160805bb16542c9531888519430e9e6d6ffc09d72261b0d26ff74f Status: Downloaded newer image for hello-world:latest`
- Thử tạo container đầu tiên `docker run hello-world`
  - Kết quả như sau:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
        executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
        to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

- Kết thúc việc cài đặt, nếu kết quả ổn thì bạn đã có thể bắt đầu tìm hiểu về docker.

Một số thao tác để trải nghiệm docker sau khi cài đặt

Sau khi cài đặt docker xong, bạn nên trải nghiệm cách sử dụng docker với các thao tác cơ bản trước khi đi vào chi tiết của từng thành phần.

Chạy một container

Chạy một container tức là khởi chạy một ứng dụng nào đó trong container. Hãy thực hiện các lệnh dưới và cùng đối chiếu kết kết quả.

```
sh docker run busybox echo 'Xin chao'
```

- Kết quả: Màn hình sẽ trả về dòng thông báo ở trên. `Xin chao`
  `docker run busybox whoami`
- Kết quả: Lệnh `whoami` sẽ được thực hiện trong container và đưa thông báo ra bên ngoài. `root`
  `docker run busybox route`
- Kết quả: `Kernel IP routing table`
  `Destination Gateway Genmask Flags Metric Ref Use Iface`
  `default 172.17.0.1 0.0.0.0 UG 0 0 0 eth0`
  `172.17.0.0 * 255.255.0.0 U 0 0 0 eth0`
  `root@devstack01:~#`
  - Chúng ta có thể thấy cú pháp của 03 lệnh trên là `docker run <ten_image> <cau_lenh_duoc_thuc_hien>`. Trong đó:
- `docker`: là tên lệnh để máy cài docker thực hiện thao tác với docker bằng CLI.
- `run`: là tùy chọn để thực hiện, ngoài `run` còn nhất nhiều các tùy chọn khác như `images, pull, rmi ...`, chúng ta sẽ khám phá sau.
- `busybox`: là tên images dùng để tạo các container.
- `echo, whoami, route`: là các lệnh sẽ được truyền vào trong container thực hiện.
  - Ta cũng để ý, khi thực hiện lệnh `docker run` thì máy sẽ tiến hành tìm kiếm images được chỉ định trong localhost, nếu không có thì mặc nó sẽ thực hiện `pulled` từ registry Docker Hub về máy cài docker. Registry Docker Hub là một `kho` lưu trữ các images. Ta cũng có thể sử dụng một registry local - tức là một registry offline trong nội bộ mạng LAN.

Thao tác với một container với chế độ tương tác, sử dụng tùy chọn `-it`

- Trong các ví dụ trước ta mới thao tác để thực thi nhanh với các container, trong phần này ta sẽ sử dụng cách tương tác với một container. Có nghĩa là tạo ra các container và thao tác trực tiếp với chúng. Hãy chạy lệnh dưới. `docker run -it busybox`

Trong lệnh trên ta sử dụng tùy chọn `-it` - đây chính là tùy chọn cho phép tương tác trực tiếp trong container, kết quả ta sẽ nhận được cửa sổ thao tác trong container như sau.

```
sh root@devstack01:~# docker run -it busybox / # echo "Chao cac ban" Chao cac ban / # -
```

Trong mục trên các bạn có thể thấy ta đã thực hiện lệnh `echo "Chao cac ban"` và kết quả hiển thị ra màn hình. Để thoát khỏi chế độ tương tác với container ta dùng lệnh `exit`/ `sh`

```
root@devstack01:~# docker run -it busybox / # echo "Chao cac ban" Chao cac ban / # exit
```

Trong phần trên ta đã sử dụng tùy chọn `-it`, trong đó `-i` là tùy chọn sử dụng để tạo container với chế độ tương tác, tùy chọn `-t` là tùy chọn mở ra một phiên làm việc. Nếu chỉ sử dụng tùy chọn `-i` thì chúng ta sẽ mở ra một section và đóng lại luôn. Nếu sử dụng chỉ tùy chọn `-t` thì sẽ mở ra một section và không thao tác được.

Tạo một container với chế độ deamon, sử dụng tùy chọn `-d`

Thông thường, khi tạo một container với các tùy chọn trước thì sau khi tạo xong hoặc thoát container thì ngay lập tức container đó sẽ dừng hoạt động. Trong một số trường hợp ta sẽ cần các container chạy ngầm, trong trường hợp này ta sử dụng tùy chọn `-d`.

```
sh docker run -d httpd
```

Sau khi chạy lệnh trên xong, ta có thể sử dụng lệnh `docker ps` để xem container này còn hoạt động hay không (nếu để quan sát cả container đã dừng thì dùng lệnh `docker ps -a`).

- Kết quả lệnh `docker ps`. Chú ý quan sát cột `STATUS` `CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 4522587672e0 httpd "httpd-foreground" 4 seconds ago Up 3 seconds 80/tcp modest_perlman root@devstack01:~#`
- Kết quả lệnh `docker ps -a`.Chú ý quan sát cột `STATUS`. Ta thấy trước đó có các container đã được tạo mà không có tùy chọn `-d` đang có status là `Exited` `root@devstack01:~# docker ps -a CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 4522587672e0 httpd "httpd-foreground" 58 seconds ago Up 57 seconds 80/tcp modest_perlman 30b7e0f132af busybox "sh" 33 minutes ago Exited (0) 32 minutes ago cocky_leakey a0075342c64f busybox "route" 41 minutes ago Exited (0) 41 minutes ago youthful_lovelace ab55d78fa2c9 httpd "httpd-foreground" About an hour ago`

```
Exited (0) About a minute ago                       elated_brahmagupta
6bf88da1f473        httpd               "httpd-foreground"      About an hour ago
Exited (0) About a minute ago                       loving_curie
46fe544a7d20        httpd               "httpd-foreground"      About an hour ago
Exited (0) About a minute ago                       compassionate_hugle
```

Tạo một container với port chỉ định, sử dụng tùy chọn `-p`

Nếu không chỉ định tùy chọn `-p` cho container thì thường container sinh ra sẽ có một port mặc định nào đó, lúc đó ta muốn sử dụng container đó thì phải đứng ở máy chứa container và thao tác, ví dụ như ta có một container chạy ứng dụng web, lúc đó ta có thể truy cập tới ứng dụng web trên máy cài đặt container với port của container được sinh ra mặc định.

Do vậy, để `phơi` một port của container ra bên ngoài - giúp các máy ngoài container có thể sử dụng được thì ta cần dùng tùy chọn `-p`. Ở ví dụ dưới ta sẽ tạo ra một container chạy web và ánh xạ port 4000 của máy host tới port 80 của container được sinh ra.

```
sh docker run -d -p 4000:80 httpd
```

- Sử dụng lệnh `docker ps`, ta có thể quan sát cột `PORTS` để thấy thông số ánh xạ port giữa host (Máy cái docker) và container `CONTAINER ID   IMAGE   COMMAND
CREATED   STATUS   PORTS   NAMES
4594542ec71b   httpd   "httpd-foreground"   40 seconds ago   Up
39 seconds   0.0.0.0:4000->80/tcp   goofy_euler
4522587672e0   httpd   "httpd-foreground"   8 minutes ago   Up 8
minutes   80/tcp   modest_perlman`

Khi đó ta có thể đứng ở các máy bên ngoài và truy cập web với địa chỉ `http://ip_may_cai_docker:4000`, kết quả là ta sẽ thấy nội dung của web. Hoặc ta có thể đứng trên máy cài docker và sử dụng lệnh `ss -lan | grep 4000`, kết quả ta sẽ thấy port 4000 trên host cài docker.

```
```sh
root@devstack01:~# ss -lan | grep 4000
tcp    LISTEN     0      128      :::4000                 :::*
root@devstack01:~#
```
```

Ngoài cách mở trình duyệt vào địa chỉ, ta có thể sử dụng lệnh `curl localhost:4000` trên ngay máy cài docker, ta sẽ có kết quả trả về như bên dưới.

```
sh root@devstack01:~# curl localhost:4000 <html><body><h1>It works!</h1></body></html>
```

- Trong trường hợp một container có nhiều port, ta cần sử dụng nhiều port thì có thể sử dùng tùy chọn `-p 4000:80 8081:3306`.
- Trong trường hợp ta muốn ánh xạ port ngẫu nhiên cho toàn bộ các port có sẵn trong container, ta sử dụng tùy chọn `-P`, ví dụ

```
docker run -d -P nginx
```

- Sử dụng lệnh `docker ps` để xem container đang được ánh xạ port nào. `sh root@devstack01:~# docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 8be19991391d nginx "nginx -g 'daemon of…" 29 seconds ago Up 27 seconds 0.0.0.0:32769->80/tcp suspicious_bhaskara`

Thao tác với một container đã tồn tại

- Khi một container đang ở trạng thái `UP` thì ta có thể truy cập vào để thao thác, giả sử như ta có container với ID là `8be19991391d`, để truy cập vào trong container đó ta dùng lệnh `docker exec -it <ID_Container> /bin/bashdocker exec -it 8be19991391d /bin/bash`
  - Kết quả: lưu ý ta có thể quan sát cửa sổ nhắc lệnh `root@8be19991391d`
- `root@8be19991391d:/#`
  Để thoát khỏi container ta sử dụng lệnh `exit`

Chỉ định RAM và CPU cho một container

- Một container có thể chỉ định lượng RAM và CPU khi tạo. `sh docker run -d -p 4000:80 --name webserver --memory 400m --cpus 0.5 httpd`

Trong ví dụ trên: - `--memory`: là giá trị RAM gán cho container. - `--cpus`: là giá trị CPU gán cho container. - `--name`: tên của container.

Lệnh xem nội dung của một container, xem log và kiểm tra port

- Container có thông tin có thể xem được, trong đó chứa rất nhiều tham số có thể khai thác được, để xem nội dung của container ta kiểm tra bằng lệnh `docker inspect <ten_hoac_ID_Container>`, ví dụ `docker inspect 459c2c5f8d32`
- Để xem log của một container ta sử dụng lệnh `docker logs -f <ten_hoac_ID_Container>` `docker`

`logs -f 459c2c5f8d32`

- Kiểm tra port của một container `docker port 459c2c5f8d32`

Các lệnh xóa các container.

- Để xóa các container ta sử dụng lệnh `docker rm <ten_hoac_ID_Container>` hoặc `docker rm -f <ten_hoac_ID_Container>` `docker rm 459c2c5f8d32`
  hoặc `docker rm 459c2c5f8d32`
- Có thể sử dụng thủ thuật dưới để xóa toàn bộ các container `docker rm -f `docker ps -aq``