BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KĨ THUẬT TP.HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KÌ

MÔN HỌC: CLOUD COMPUTING

TOPIC:

SỬ DỤNG DOCKER ĐỂ ẢO HÓA SERVER UBUNTU

NHÓM SINH VIÊN THỰC HIỆN:

1.Lê Ngọc Đoan 17133014

2.Nguyễn Đức Thuận 18130043

GIẢNG VIÊN HƯỚNG DẪN: HUỲNH XUÂN PHỤNG

TP.HỒ CHÍ MINH, tháng 05 năm 2021

LỜI CẢM ƠN

Sau quá trình học tập trong suốt học kì chúng em đã học được rất nhiều kiến thức từ thầy Huỳnh Xuân Phụng và các bạn cùng lớp. Để hoàn thành bài báo cáo cuối kỳ môn điện toán đám mây, nhờ những kiến thức mình đã học được từ thầy và các bạn cho nên chúng em chân thành cảm ơn giảng viên và các bạn cùng lớp đã nỗ lực ân cần dạy cho chúng em những kiến thức căn bản và cần thiết.

Và em củng xin cảm ơn tất cả thầy cô trong Khoa Công Nghệ Thông Tin, cùng thầy, cô trường Đại Học Sư Phạm kỹ Thuật Thành Phố Hồ Chí Minh đã tạo điều kiện cho cô và trò chúng em học tập được những kiến thức cần thiết để chúng em phát triển cho mai sau.

Vì năng lực của chúng em có hạn nên trong quá trình làm báo cáo khó tránh khỏi những sai sót, em mong nhận được những nhận xét quý giá từ giảng viên để em ngày càng hoàn thiện hơn.

Lời cuối em xin chúc thầy và quý thầy cô trong khoa củng như tất cả thầy cô trường Đại Học SPKT TP.HCM luôn dồi dào sức khỏe và thành công trên sự nghiệp cao quý là con đường giảng dạy để truyền tải những kiến thức bổ ích đến tất cả sinh viên.

TP.Hồ Chí Minh, tháng 05 năm 2021

MỤC LỤC

CHƯƠNG 1: TÌM HIỂU VỀ DOCKER VÀ SSH

1.1. Docker

1.1.1. Khái niệm

Docker là nền tảng phần mềm cho phép bạn dựng, kiểm thử và triển khai ứng dụng một cách nhanh chóng. Docker đóng gói phần mềm vào các đơn vị tiêu chuẩn hóa được gọi là container có mọi thứ mà phần mềm cần để chạy, trong đó có thư viện, công cụ hệ thống, mã và thời gian chạy. Bằng cách sử dụng Docker, bạn có thể nhanh chóng triển khai và thay đổi quy mô ứng dụng vào bất kỳ môi trường nào và biết chắc rằng mã của bạn sẽ chạy được.

Docker hoạt động bằng cách cung cấp phương thức tiêu chuẩn để chạy mã của bạn. Docker là hệ điều hành dành cho container. Cũng tương tự như cách máy ảo ảo hóa (loại bỏ nhu cầu quản lý trực tiếp) phần cứng máy chủ, các container sẽ ảo hóa hệ điều hành của máy chủ. Docker được cài đặt trên từng máy chủ và cung cấp các lệnh đơn giản mà bạn có thể sử dụng để dựng, khởi động hoặc dừng container.

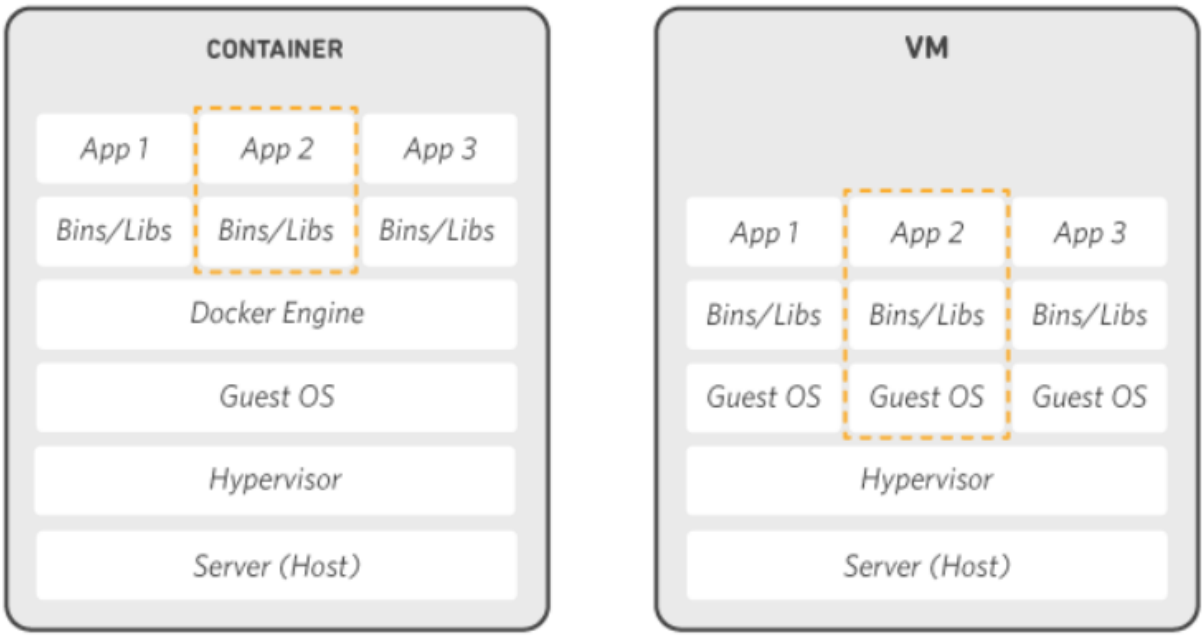
Các thành phần của Docker:

- docker engine: công cụ đóng gói ứng dụng, là thành phần chính của docker
- docker images: là 1 khuôn mẫu để tạo container.
- Container: là 1 instane của 1 image. có thể create, start, stop, move hoặc delete container dựa trên docker API hoặc CLI.
- docker hub (github cho docker images): nơi đây có hàng ngàn public images được tạo bởi cộng đồng cho phép bạn tìm và sử dụng các images mà bạn muốn.
- docker client: công cụ giúp người dùng giao tiếp với docker host
- docker daemon: tiếp nhận các yêu cầu từ docker client để quản lý các đối tượng như container, images, network, volumes thông qua REST API.
- Volumes: phần dữ liệu được tạo ra khi container được khởi tạo.
- dockerfile: là 1 tập tin bao gồm các chỉ dẫn để build 1 image.

1.1.2. Cấu trúc của Docker

Chúng ta đã từng biết đến các công nghệ máy ảo như các phần mềm ảo hóa VMWare hay Virtualbox, Docker cũng có cấu trúc cơ bản giống với chúng nhưng

điều đặc biệt hơn từ Docker đó là không phải tốn bất kỳ 1 chút bộ nhớ nào của máy để build lên các OS cho môi trường ảo mà bạn cần dùng:



Không giống các máy áo, Docker container không phải build 1 hệ điều hành hoàn chỉnh, chỉ cần có các thư viện và các thiết lập cần thiết để tạo ra môi trường làm việc. Điệu này làm cho hiệu quả, nhẹ, khép kín hệ thống và đảm bảo rằng phần mềm sẽ luôn luôn chạy như nhau, bất kể nó được triển khai ở đâu. Docker cung cấp thêm một lớp trừu tượng và tự động hóa việc ảo hóa cấp hệ điều hành trên Linux, chính vì vậy, dù bạn có sài Docker trên Windows hay MacOs, thì nó vẫn có nhân là Linux.

1.1.3. Các ưu điểm của Docker

- **Vận chuyển phần mềm nhiều hơn và nhanh hơn:** Người dùng sử dụng Docker vận chuyển phần mềm nhanh hơn trung bình 7 lần so với người dùng không sử dụng Docker. Docker đem đến cho bạn khả năng vận chuyển dịch vụ được tách riêng với tần suất mong muốn.
- **Tiêu chuẩn hóa quá trình vận hành:** Ứng dụng được đóng gói vào container nhỏ sẽ khiến cho việc triển khai, xác định vấn đề và đảo ngược để khắc phục trở nên dễ dàng.
- **Di chuyển trơn tru:** Ứng dụng trên nền tảng Docker có thể được di chuyển trơn tru từ các máy phát triển cục bộ đến đơn vị triển khai sản xuất trên AWS.
- **Tiết kiệm tiền bạc:** Container Docker giúp cho việc chạy nhiều mã hơn trên từng máy chủ trở nên dễ dàng hơn, cải thiện khả năng tận dụng và tiết kiệm tiền bạc cho bạn.

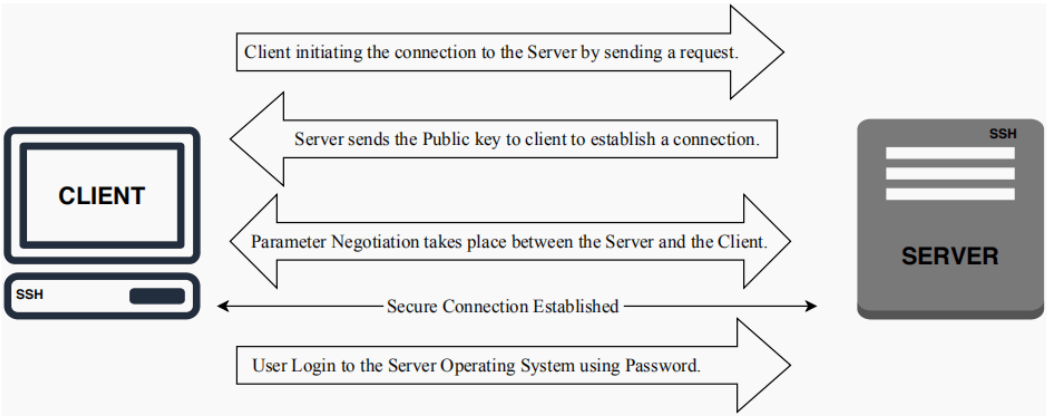
1.2. SSH

1.2.1. Khái niệm

SSH (Secure Socket Shell) là một giao thức mạng cung cấp cho quản trị viên một cách an toàn để truy cập máy tính từ xa. SSH cũng đề cập đến bộ tiện ích thực hiện giao thức. Secure Shell cung cấp khả năng xác thực mạnh và bảo mật thông tin liên lạc giữa hai máy tính kết nối qua mạng không an toàn như Internet.

SSH là một chương trình tương tác giữa máy chủ và máy khách có sử dụng cơ chế mã hoá đủ mạnh nhằm ngăn chặn các hiện tượng nghe trộm, đánh cắp thông tin trên đường truyền. Các chương trình trước đây: telnet, rlogin không sử dụng phương pháp mã hoá. Vì thế bất cứ ai cũng có thể nghe trộm thậm chí đọc được toàn bộ nội dung của phiên làm việc bằng cách sử dụng một số công cụ đơn giản. Sử dụng SSH là biện pháp hữu hiệu bảo mật dữ liệu trên đường truyền từ hệ thống này đến hệ thống khác.

Hình bên dưới trình bày một luồng thiết lập đơn giản của kết nối shell an toàn.



1.2.2. Những công dụng tiêu biểu của giao thức SSH

Giao thức được sử dụng trong các mạng công ty để:

- Cung cấp quyền truy cập an toàn cho người dùng và quy trình tự động
- Chuyển các file tương tác và tự động
- Phát lệnh từ xa
- Quản lý cơ sở hạ tầng mạng và các thành phần hệ thống giữ nhiệm vụ quan trọng khác.

1.2.3. Cách thức làm việc của SSH

SSH làm việc thông qua 3 bước:

Định danh host

Việc định danh host được thực hiện qua việc trao đổi khoá. Mỗi máy tính có hỗ trợ kiểu truyền thông SSH có một khoá định danh duy nhất. Khoá này gồm hai thành phần: khoá riêng và khoá công cộng. Khoá công cộng được sử dụng khi cần trao đổi giữa các máy chủ với nhau trong phiên làm việc SSH, dữ liệu sẽ được mã hoá bằng khoá riêng và chỉ có thể giải mã bằng khoá công khai. Khi có sự thay đổi về cấu hình trên máy chủ: thay đổi chương trình SSH, thay đổi cơ bản trong hệ điều hành, khoá định danh cũng sẽ thay đổi. Khi đó mọi người sử dụng SSH để đăng nhập vào máy chủ này đều được cảnh báo về sự thay đổi này. Khi hai hệ thống bắt đầu một phiên làm việc SSH, máy chủ sẽ gửi khoá công cộng của nó cho máy khách. Máy khách sinh ra một khoá phiên ngẫu nhiên và mã hoá khoá này bằng khoá công cộng của máy chủ, sau đó gửi lại cho máy chủ. Máy chủ sẽ giải mã khoá phiên này bằng khoá riêng của mình và nhận được khoá phiên. Khoá phiên này sẽ là khoá sử dụng để trao đổi dữ liệu giữa hai máy. Quá trình này được xem như các bước nhận diện máy chủ và máy khách.

Mã hoá

Sau khi hoàn tất việc thiết lập phiên làm việc bảo mật (trao đổi khoá, định danh), quá trình trao đổi dữ liệu diễn ra thông qua một bước trung gian đó là mã hoá/giải mã. Điều đó có nghĩa là dữ liệu gửi/nhận trên đường truyền đều được mã hoá và giải mã theo cơ chế đã thoả thuận trước giữa máy chủ và máy khách. Việc lựa chọn cơ chế mã hoá thường do máy khách quyết định. Các cơ chế mã hoá thường được chọn bao gồm: 3DES, IDEA, và Blowfish. Khi cơ chế mã hoá được lựa chọn, máy chủ và máy khách trao đổi khoá mã hoá cho nhau. Việc trao đổi này cũng được bảo mật dựa trên đinh danh bí mật của các máy. Kẻ tấn công khó có thể nghe trộm thông tin trao đổi trên đường truyền vì không biết được khoá mã hoá. Các thuật toán mã hoá khác nhau và các ưu, nhược điểm của từng loại:

- 3DES (cũng được biết như Triple-DES) -- phương pháp mã hoá mặc định cho SSH.
- IDEA—Nhanh hơn 3DES, nhưng chậm hơn Arcfour và Blowfish.
- Arcfour—Nhanh, nhưng các vấn đề bảo mật đã được phát hiện.
- Blowfish—Nhanh và bảo mật, nhưng các phương pháp mã hoá đang được cải tiến.

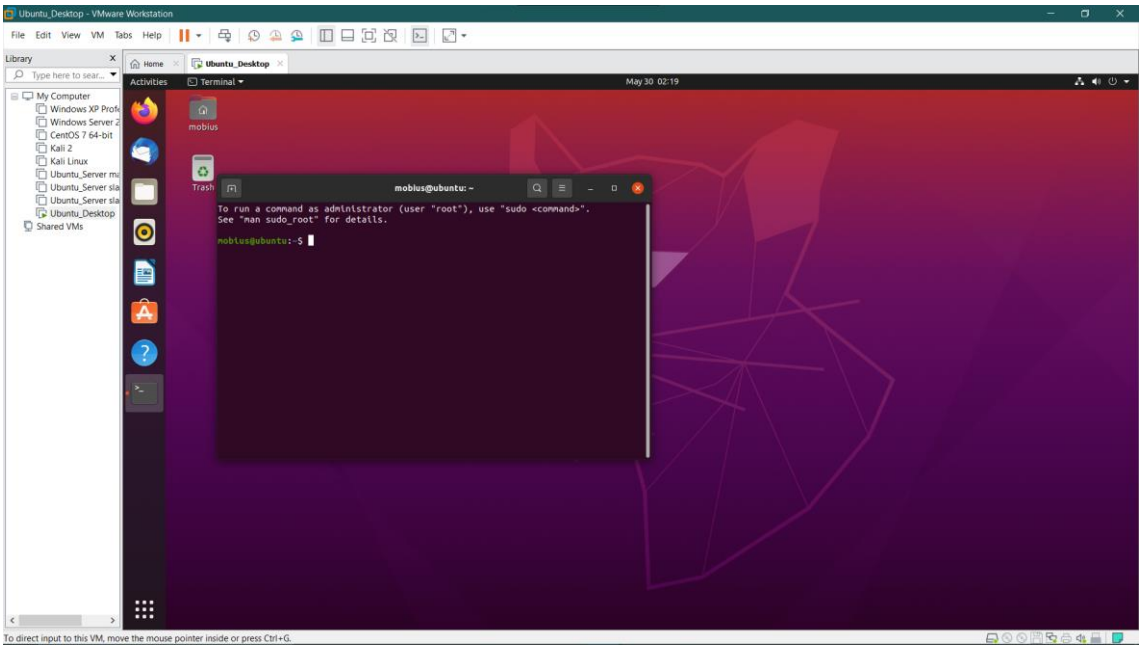
Chứng thực

Việc chứng thực là bước cuối cùng trong ba bước, và là bước đa dạng nhất. Tại thời điểm này, kênh trao đổi bản thân nó đã được bảo mật. Mỗi định danh và truy nhập của người sử dụng có thể được cung cấp theo rất nhiều cách khác nhau. Chẳng hạn, kiểu chứng thực rhosts có thể được sử dụng, nhưng không phải là mặc định; nó đơn giản chỉ kiểm tra định danh của máy khách được liệt kê trong file rhost (theo DNS và địa chỉ IP). Việc chứng thực mật khẩu là một cách rất thông dụng để định danh người sử dụng, nhưng ngoài ra cũng có các cách khác: chứng thực RSA, sử dụng ssh-keygen và ssh-agent để chứng thực các cặp khoá.

CHƯƠNG 2: CÀI ĐẶT DOCKER TRÊN UBUNTU

2.1. Cài đặt Ubuntu

Vì tài nguyên máy tính có hạn nên ở đây chúng em triển khai cài đặt docker trên 1 máy ảo Ubuntu 20.04.2 được ảo hóa bằng phần mềm VMware.



Cấu hình máy:

Device	Summary
Memory	2 GB
Processors	2
Hard Disk (SCSI)	20 GB
CD/DVD (SATA)	Auto detect
Network Adapter	Bridged (Automatic)
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

2.2. Cài đặt Docker trên Ubuntu

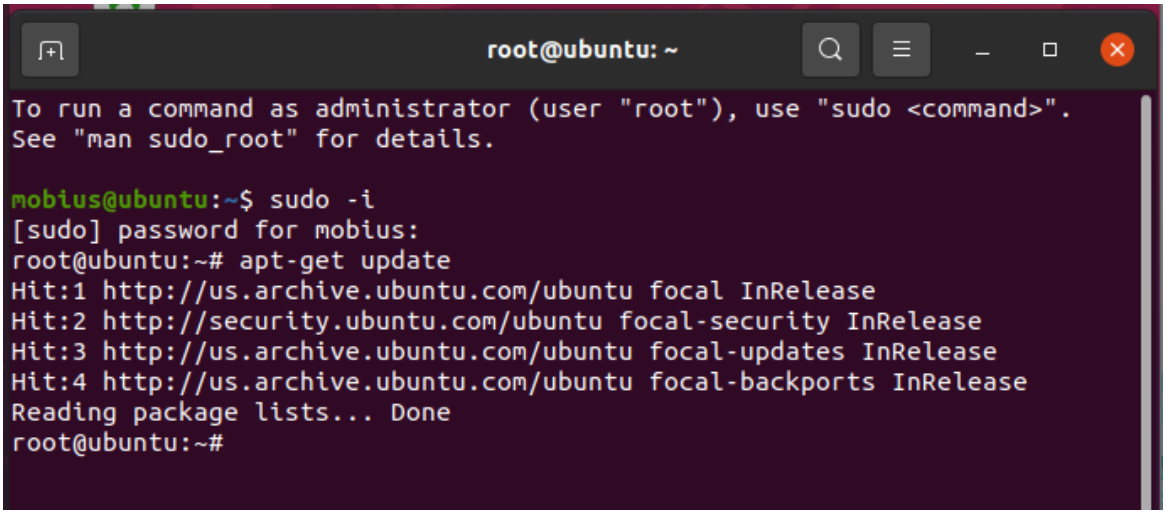
Bước 1: Đăng nhập tài khoản Root

```
mobius@ubuntu:~$ sudo -i
[sudo] password for mobius:
root@ubuntu:~#
```

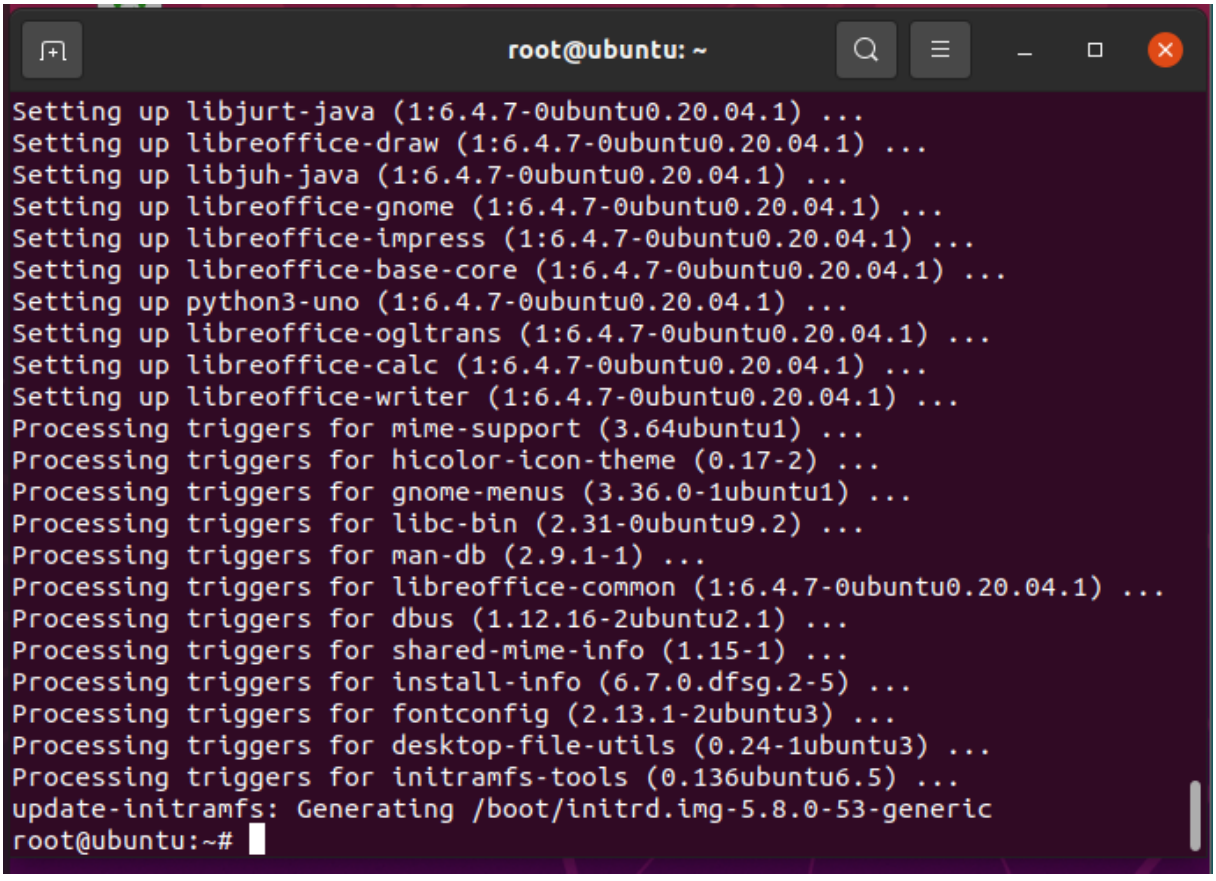
Bước 2: Cập nhật APT

cập nhật và nâng cấp các APT của Ubuntu.

apt-get update



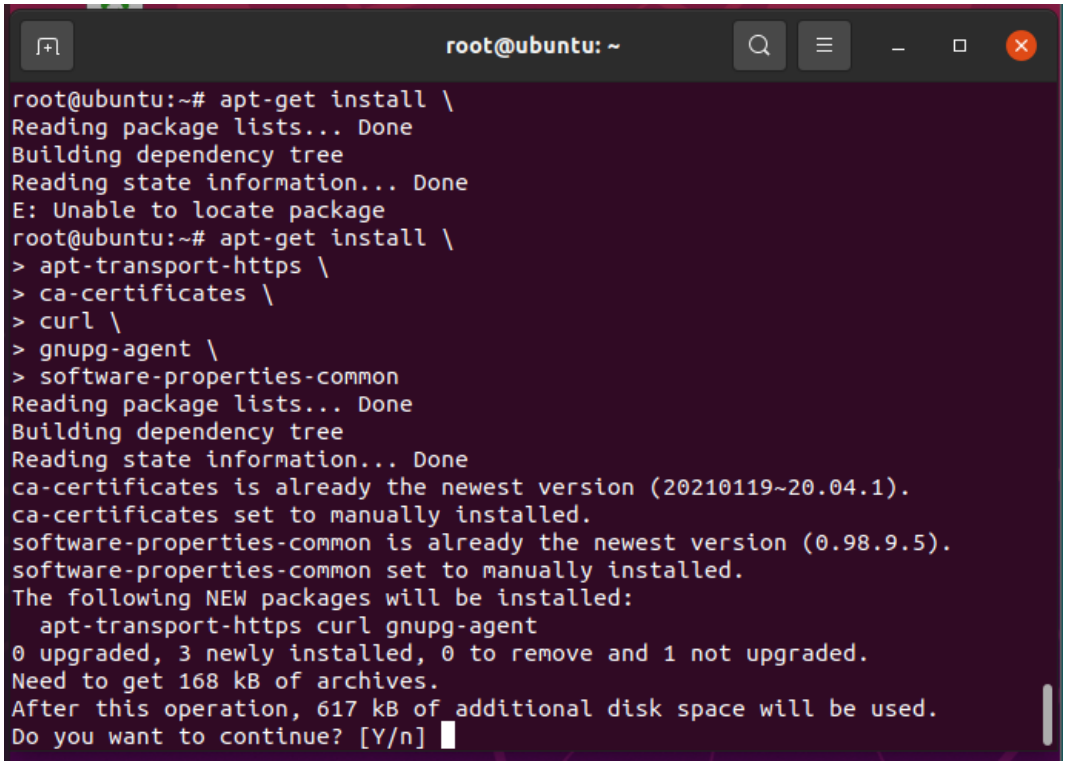
apt-get upgrade



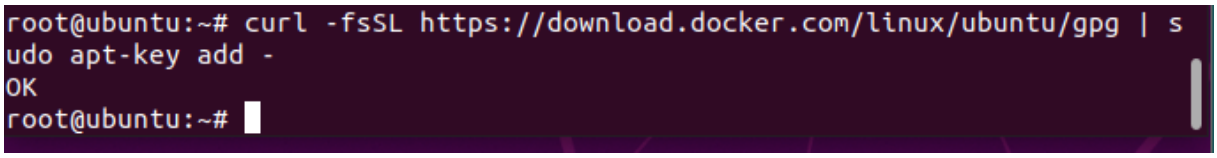
Bước 3: Tải và cài đặt Docker sử dụng phương thức repository

- Cài đặt repository:

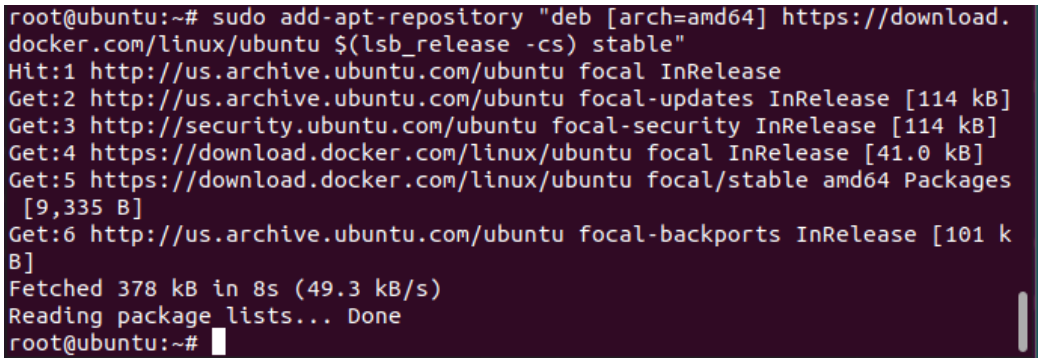
- Cài đặt các package để cho phép apt sử dụng hệ thống lưu trữ qua HTTPS



- Thêm khóa GPG chính thức của Docker:



- Thêm Docker APT repository:



- Cài đặt Docker Engine

- cài đặt phiên bản mới nhất của Docker Engine và containerd

```
root@ubuntu:~# apt-get install docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras docker-scan-plugin git git-man liberror-perl
  pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run
  | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-scan-plugin git git-man liberror-perl pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 113 MB of archives.
After this operation, 504 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

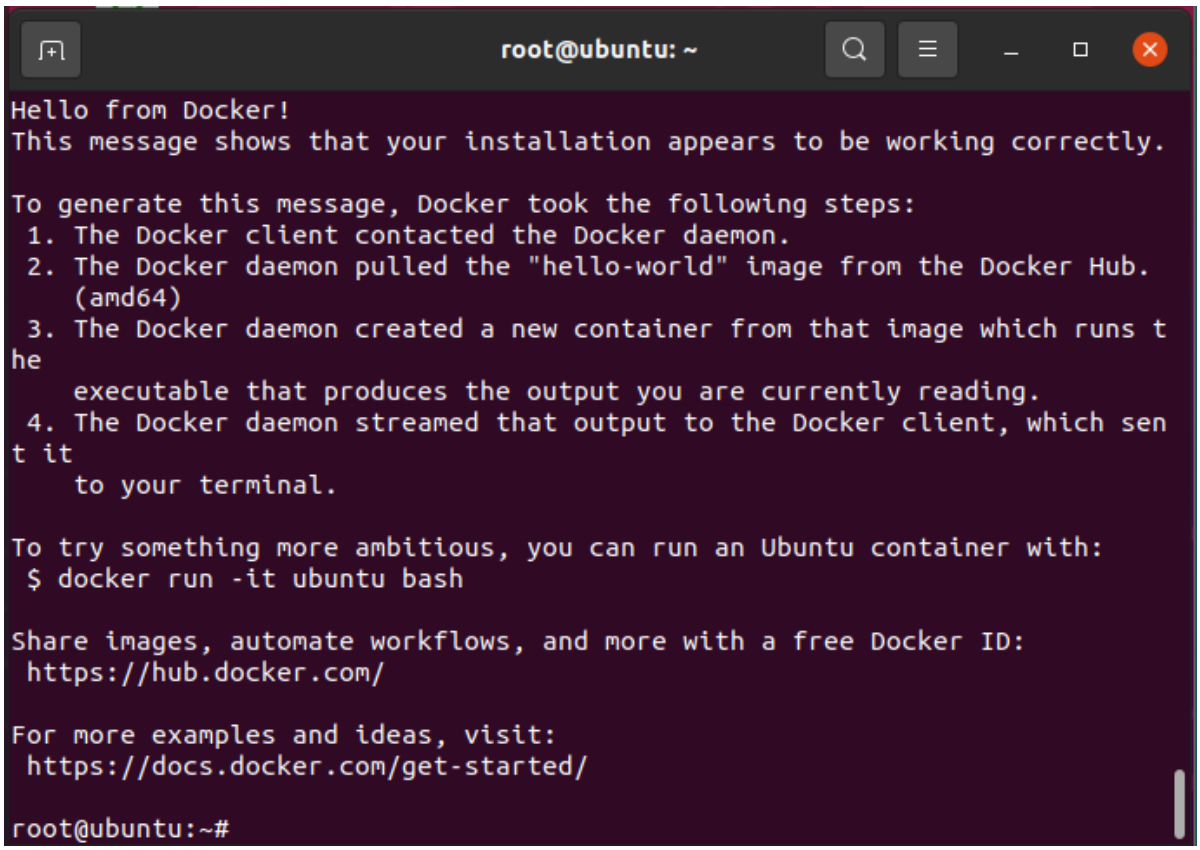
- Kiểm tra xem docker đã được cài đặt chưa

$ systemctl status docker

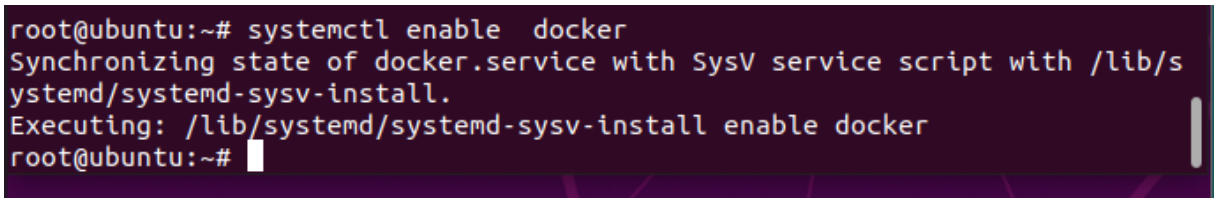
```
root@ubuntu: ~
Processing triggers for systemd (245.4-4ubuntu3.6) ...
root@ubuntu:~# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor >
     Active: active (running) since Sun 2021-05-30 02:44:15 PDT; 3min 51s>
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 33777 (dockerd)
      Tasks: 10
     Memory: 42.9M
     CGroup: /system.slice/docker.service
             └─33777 /usr/bin/dockerd -H fd:// --containerd=/run/containe>

May 30 02:44:14 ubuntu dockerd[33777]: time="2021-05-30T02:44:14.22940479>
May 30 02:44:14 ubuntu dockerd[33777]: time="2021-05-30T02:44:14.22943415>
May 30 02:44:14 ubuntu dockerd[33777]: time="2021-05-30T02:44:14.22944196>
May 30 02:44:14 ubuntu dockerd[33777]: time="2021-05-30T02:44:14.22971851>
May 30 02:44:15 ubuntu dockerd[33777]: time="2021-05-30T02:44:15.09864035>
May 30 02:44:15 ubuntu dockerd[33777]: time="2021-05-30T02:44:15.45994533>
May 30 02:44:15 ubuntu dockerd[33777]: time="2021-05-30T02:44:15.53582899>
May 30 02:44:15 ubuntu dockerd[33777]: time="2021-05-30T02:44:15.53605020>
May 30 02:44:15 ubuntu systemd[1]: Started Docker Application Container E>
May 30 02:44:15 ubuntu dockerd[33777]: time="2021-05-30T02:44:15.60050535>
lines 1-21/21 (END)
```

- Kiểm tra xem docker đã hoạt động chưa:

$ docker run hello-world

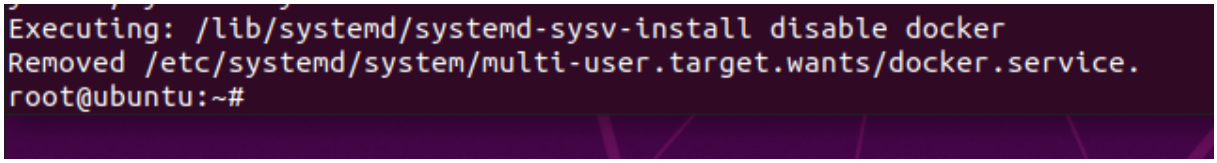


Bước 4: Chạy Docker



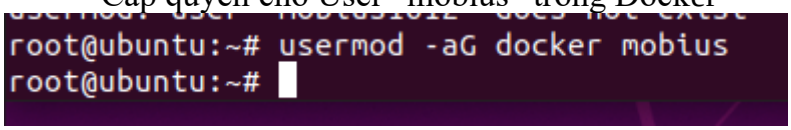
Để disable Docker, đơn giản nhập câu lệnh dưới đây.

```
systemctl disable  docker
```



Bước 5: Thiết lập quyền User

Cấp quyền cho User “mobius” trong Docker



Bước 6: Pull Image từ docker (ở đây dùng ubuntu:18.04)

- Cài đặt ubuntu:18.04 cho trên docker

```
root@ubuntu:~# docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
4bbfd2c87b75: Pull complete
d2e110be24e1: Pull complete
889a7173dcfe: Pull complete
Digest: sha256:67b730ece0d34429b455c08124ffd444f021b81e06fa2d9cd0adaf0d0b8
75182
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04
root@ubuntu:~#
```

- Khởi tạo container ubuntu trên docker

```
root@ubuntu:~# docker run -it ubuntu:18.04
root@e46597c78f5e:/# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB
]
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB
]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 P
ackages [423 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.
3 MB]
Get:7 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 P
ackages [24.7 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Pac
kages [1413 kB]
Get:9 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Package
s [2152 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 k
B]
Get:11 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [
186 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [
13.5 kB]
```

- Cài đặt java cho container

```
root@e46597c78f5e:/# apt install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  at-spi2-core ca-certificates ca-certificates-java dbus
  default-jdk-headless default-jre default-jre-headless
  fontconfig-config fonts-dejavu-core fonts-dejavu-extra java-common
  krb5-locales libapparmor1 libasound2 libasound2-data
  libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni
  libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3
  libavahi-common-data libavahi-common3 libbsd0 libcups2 libdbus-1-3
  libdrm-amdgpu1 libdrm-common libdrm-intel1 libdrm-nouveau2
  libdrm-radeon1 libdrm2 libedit2 libelf1 libexpat1 libfontconfig1
  libfontenc1 libfreetype6 libgif7 libgl1 libgl1-mesa-dri libglapi-mesa
  libglib2.0-0 libglib2.0-data libglvnd0 libglx-mesa0 libglx0
  libgraphite2-3 libgssapi-krb5-2 libharfbuzz0b libice-dev libice6
  libicu60 libjpeg-turbo8 libjpeg8 libk5crypto3 libkeyutils1 libkrb5-3
  libkrb5support0 liblcms2-2 libllvm10 libnspr4 libnss3 libpciaccess0
  libpcsclite1 libpng16-16 libpthread-stubs0-dev libsensors4 libsm-dev
  libsm6 libsqlite3-0 libssl1.1 libx11-6 libx11-data libx11-dev
  libx11-doc libx11-xcb1 libxau-dev libxau6 libxaw7 libxcb-dri2-0
  libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-shape0 libxcb-sync1
  libxcb1 libxcb1-dev libxcomposite1 libxdamage1 libxdmcp-dev libxdmcp6
```

- Kiểm tra phiên bản java đã cài đặt:

```
done.
root@e46597c78f5e:/# java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.18.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.18.04, mixed mod
e, sharing)
root@e46597c78f5e:/#
```

- Commit thay đổi lên image:

```
root@ubuntu:~# docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS
   PORTS     NAMES
a10c11d579c8   ubuntu:18.04   "/bin/bash"   11 minutes ago   Up 11 minutes
             gracious_poitras
root@ubuntu:~# docker commit a10c11d579c8 ubuntu:18.04
sha256:e4cd6038555fd106d2d8f6ba89907d1fee607b5741b69051a6ca396c92226e44
root@ubuntu:~#
```

2.3. Cài đặt SSH trên Ubuntu

Bước 1: Cập nhật APT, đăng nhập tài khoản root

Trước hết, cần cập nhật và nâng cấp APT và đăng nhập vào tài khoản root giống như các bước cài đặt docker phía trên

Bước 2: Cài đặt SSH

```
root@ubuntu:~# apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 1 not upgraded.
Need to get 688 kB of archives.
After this operation, 6,010 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

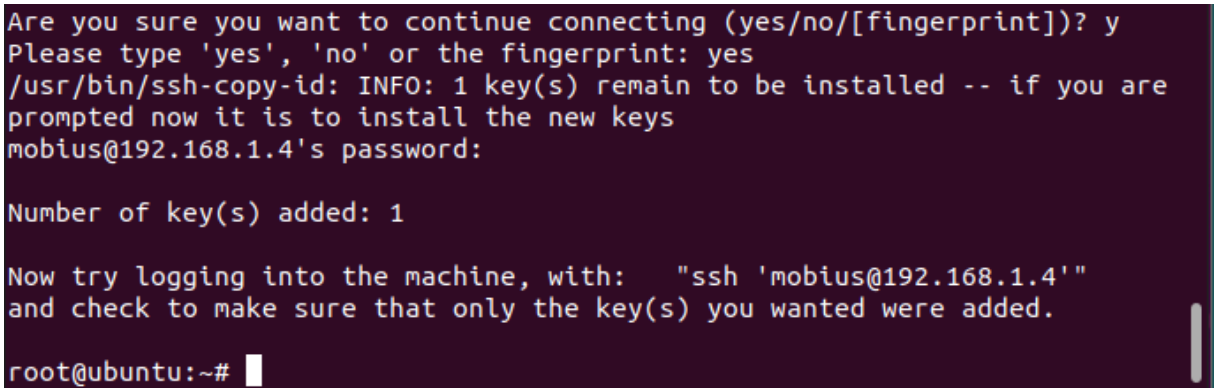
Bước 3: Khởi động service SSH:

```
root@ubuntu:~# service ssh start
root@ubuntu:~# sudo ssh-keygen
```

Bước 4: Tạo cặp key

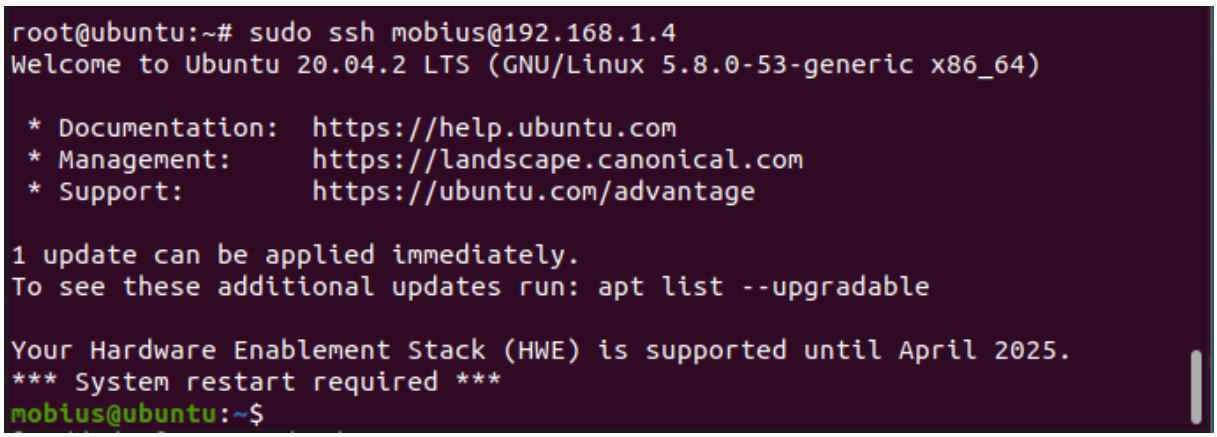
Bước 5: Copy key lên ubuntu 20.04

$ ssh-copy-id mobius@192.168.1.4



Bước 6: Kiểm tra lại SSH có thể kết nối hay chưa

$ sudo ssh mobius@192.168.1.4


root@ubuntu:~# sudo ssh mobius@192.168.1.4
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-53-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
1 update can be applied immediately.
To see these additional updates run: apt list --upgradable
Your Hardware Enablement Stack (HWE) is supported until April 2025.
*** System restart required ***
mobius@ubuntu:~$

CHƯƠNG 3: XÂY DỰNG WEBSITE QUẢN LÝ COTAINERS

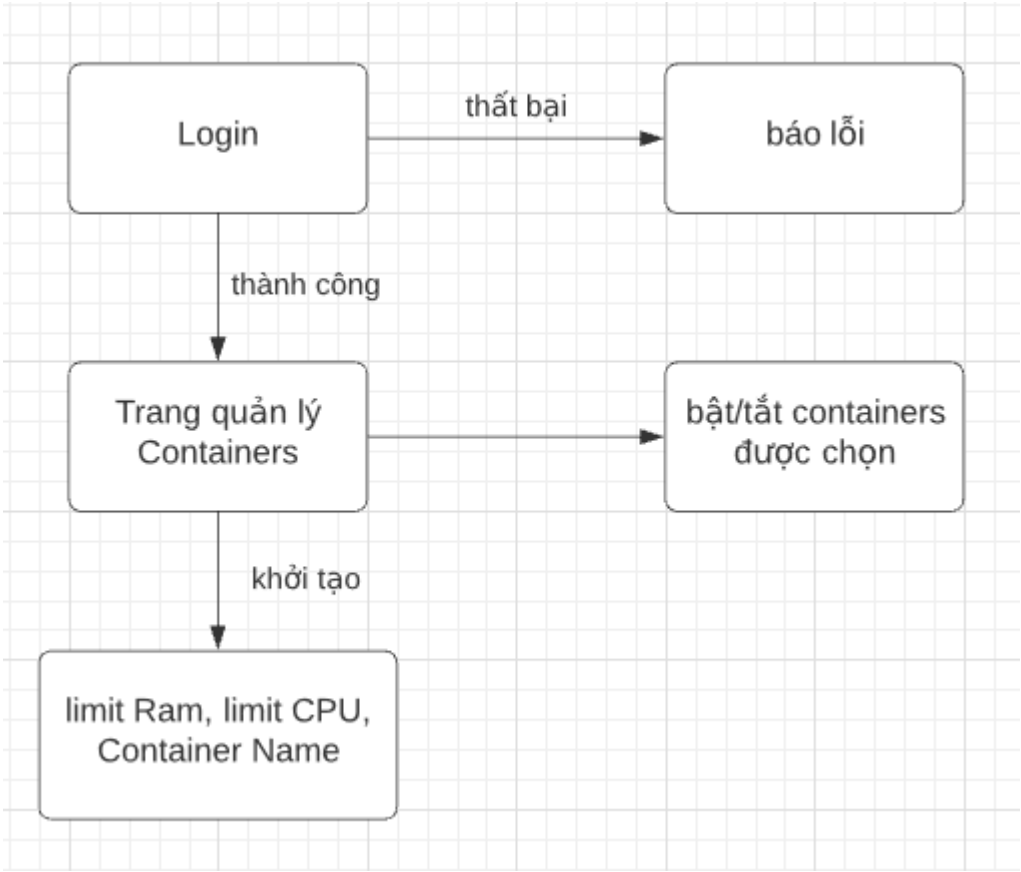
3.1. Yêu cầu

Xây dựng một website quản lý container với các tính năng sau:

- Đăng nhập (username, passwork).
- Khởi tạo Container với các thiết lập cho phép chọn giới hạn CPU và RAM cho mỗi Containers được khởi tạo.
- Quản lý Containers theo Users.
- User dùng SSH để kết nối đế container
- Bật/tắt những Containers đã tạo.

Sử dụng ASP.Net Core của Visual Studio 2019 để phát triển website.

3.2. Mô hình hệ thống



Đầu tiên, user sẽ tiến hành đăng nhập vào website. Thông tin của User được lưu trong table User của Database

	Column Name	Data Type	Allow Nulls
🔑	UserID	int	☐
	UserName	varchar(50)	☑
	PassWord	varchar(50)	☑
▶			☐

	UserID	UserName	PassWord
	1	admin	admin
	2	mobius	mobius
▶*	*NULL*	*NULL*	*NULL*

Việc đăng nhập sẽ giúp người dùng xem được những container mà họ đã tạo, được hiển thị ở *trang quản lý containers*. Thông tin Containers được lưu trong table Containers của database.

	Column Name	Data Type	Allow Nulls
🔑	ContainerID	int	☐
	ContainerName	varchar(50)	☑
	LimitCPU	float	☑
	LimitRAM	float	☑
	CreatedDate	date	☑
	UserID	int	☑
▶			☐

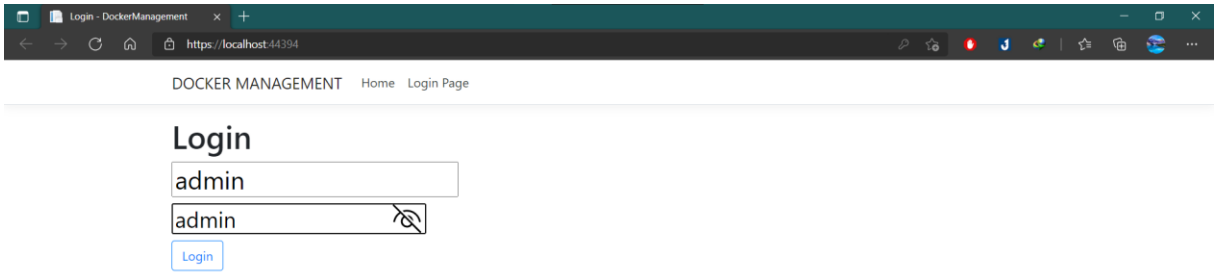
	ContainerID	ContainerN...	LimitCPU	LimitRAM	CreatedDate	UserID
	3011	Container_a...	10	128	2021-05-30	1
	3014	Container_a...	10	128	2021-05-30	1
▶*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Ở trang quản lý Containers, người dùng được phép *khởi tạo* và *bật/tắt containers*. Ngoài ra, trang quản lý sẽ cho chúng ta quan sát những container đang chạy, hay những container đang tắt.

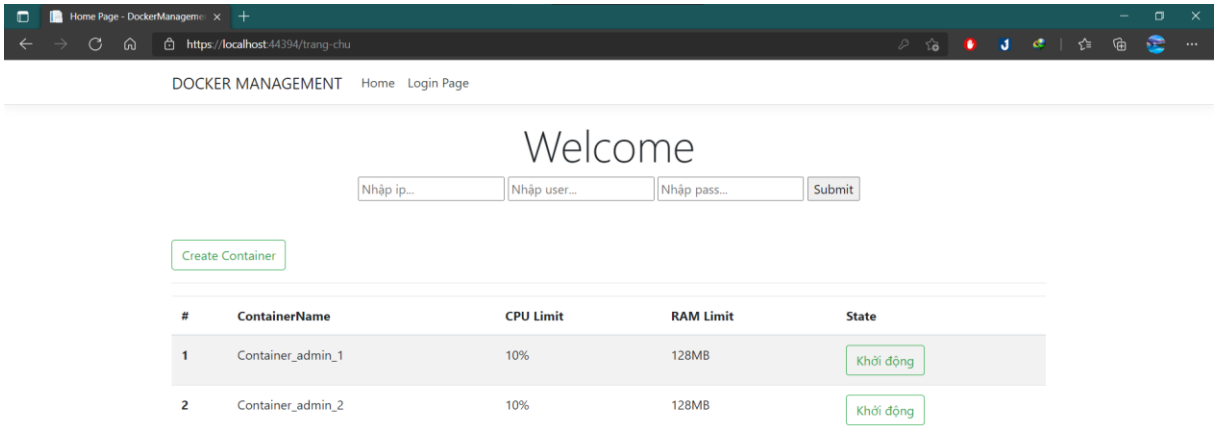
3.3. Hướng dẫn sử dụng

3.3.1. Đăng nhập và kết nối SSH đến server Ubuntu

Bước 1: Ở trang Login, nhập username và password



Bước 2: Nếu đăng nhập thành công chuyển sang trang quản lý containers, thất bại thì thông báo lỗi.



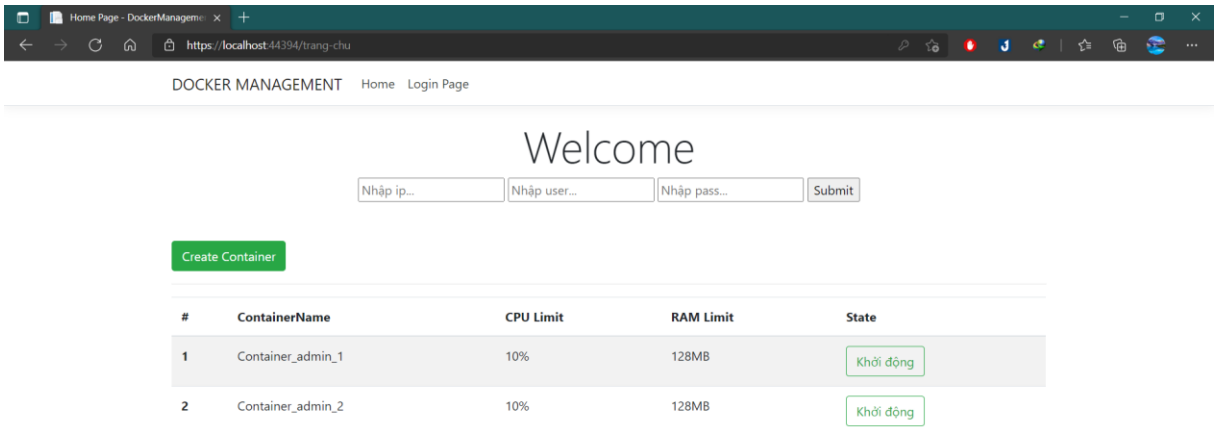
Trong quá trình login, website tự động thực hiện kết nối SSH đến server ubuntu

```
SshClient sshclient = new SshClient("ip", "user", "password");
     //ip: địa chỉ ip của máy ubuntu
     //user: username của máy ubuntu
```

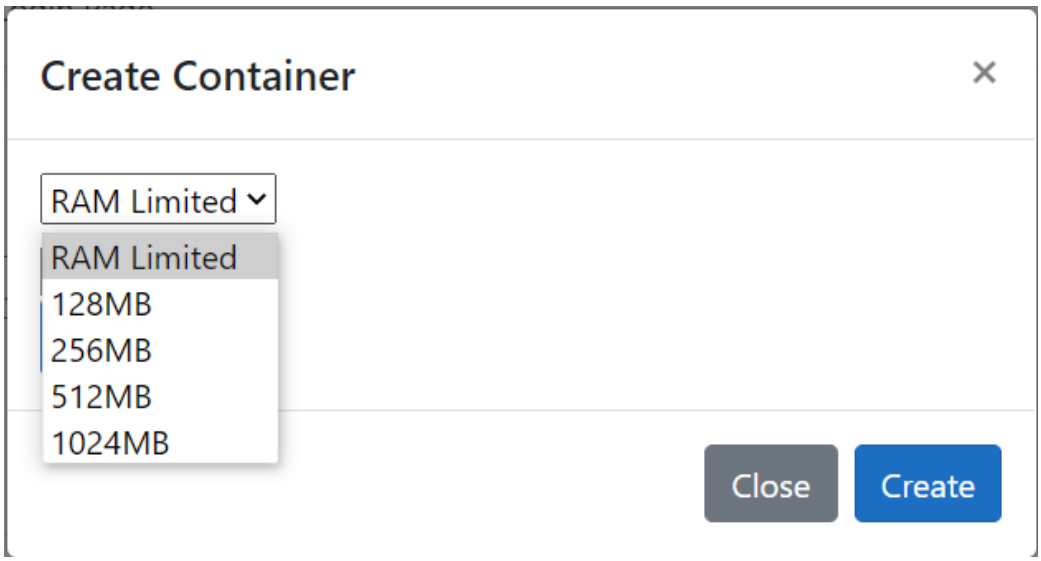
```
//password: mật khẩu của máy ubuntu

sshclient.Connect();
```

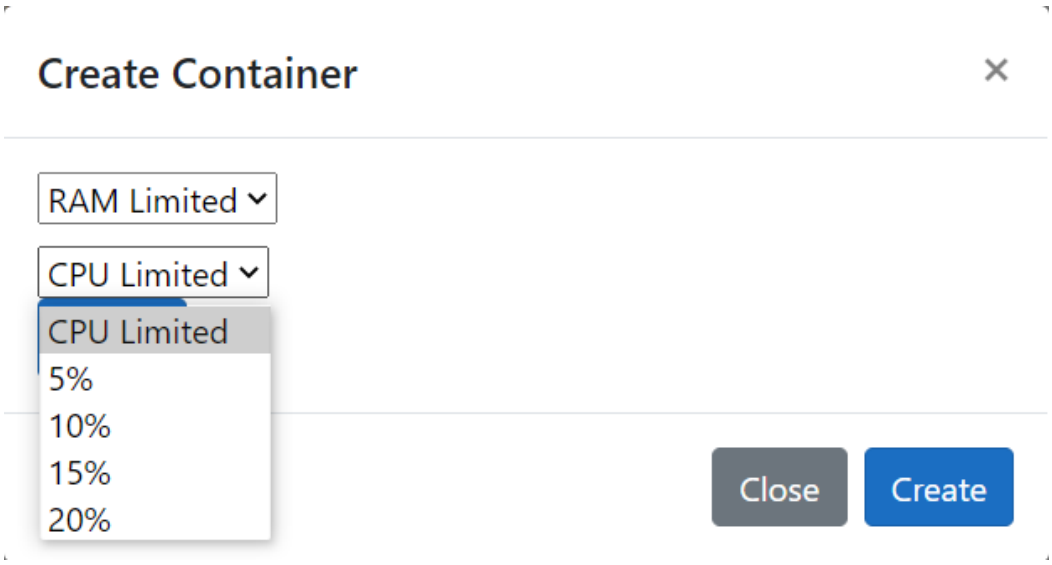
3.3.2. Khởi tạo Containers

Bước 1: Nhấn nút "Create Container" *ở trang quản lý container.*



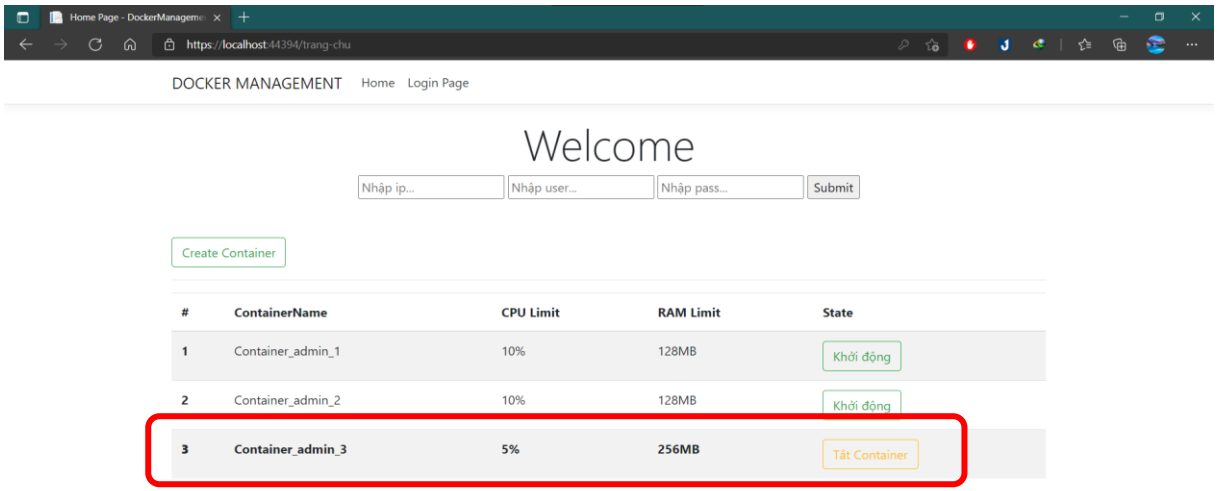
Bước 2: Chọn giới hạn RAM và CPU. Hệ thống sẽ tự động tạo ra ContainerName, theo nguyên tắc "Container_<username>_<count(containerID)byUserID + 1>".





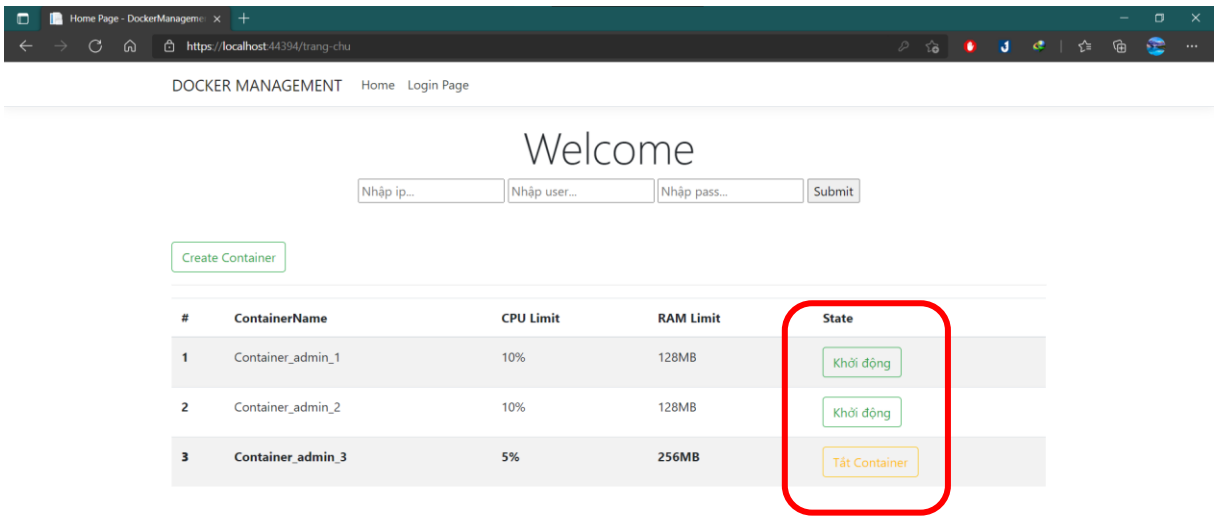
Bước 3: Click button “Create”. Chạy dòng lệnh với các input như bước 2. Nếu tạo thành công, trả về *trang quản lý container* và lưu input vào database. Nếu lỗi, báo lỗi và kết thúc.

```
Lệnh tạo container: docker run --memory=\"" + <limitRAM>+ "m\" --cpus=\"" + <limitCPU>/
100 + "\" --name " + <container_name> + " -d -it ubuntu:18.04
```



3.3.3. Bật/tắt Containers

Dễ dàng bật/tắt các containers bằng các button ở cột State. Khi các button được kích hoạt, các lệnh được thực hiện



Lệnh khởi động container: `"docker stop " + <container_name>`
Lệnh tắt container: `"docker start " + <container_name>`

CHƯƠNG 4: TỔNG KẾT

Trong suốt quá trình thực hiện đề tài, nhóm chúng em gồm 2 thành viên, cùng với nhóm thứ 2 của đề tài này đã cùng nhau nghiên cứu và tìm hiểu về các khái niệm cơ bản về docker, ssh và đã triển khai hệ thống theo yêu cầu của giảng viên. Thông qua đề tài, chúng em biết được về docker, cách triển khai docker trên server ubuntu.

Bọn em có sử dụng những thông tin có trên internet cũng như những nghiên cứu của các anh (chị) đã học môn này những học kỳ trước để thực hiện đề tài này. Tuy nhiên bọn em cam kết là không sao chép y chang mà có tìm hiểu, sửa đổi, bổ sung góp phần hoàn thiện đề tài.

Vì thời gian hạn chế và dịch bệnh đang diễn biến phức tạp, ảnh hưởng rất nhiều đến đồ án cuối kỳ của môn điện toán đám mây và nhiều môn học khác. Vì vậy đồ án còn rất nhiều sai sót, vd: khi tạo container không tự động load lên page mà phải đăng nhập lại, giao diện còn đơn giản, chưa có trang tạo user mà phải tự điền trực tiếp và database, … Tuy nhiên chúng em căn bản đã làm được các yêu cầu mà giảng viên đã đề ra.