

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO TÓM TẮT
TRIỂN KHAI ẢO HÓA & CONTAINER

Học phần: Mạng máy tính nâng cao

Lớp: CQ2019/4

Họ và tên các thành viên:

- 1. Bùi Lê Tuấn Anh – 19120163**
- 2. Ngô Nhật Tân – 19120128**

Thành phố Hồ Chí Minh, tháng 12 năm 2021

Nội dung

1. GIỚI THIỆU VỀ CÁC CÔNG NGHỆ & ỨNG DỤNG.....	1
1.1. Công nghệ Ảo hóa	1
1.2. Container & Docker.....	1
2. TRIỂN KHAI CÀI ĐẶT THỰC TẾ	3
2.1. Ứng dụng mạng trên nền ảo hóa.....	3
2.2. Ứng dụng mạng trên Docker	14
TÀI LIỆU THAM KHẢO.....	21

1. GIỚI THIỆU VỀ CÁC CÔNG NGHỆ & ỨNG DỤNG

1.1. Công nghệ Ảo hóa

Ảo hóa là một công nghệ được thiết kế để tạo ra tầng trung gian giữa phần cứng máy tính và phần mềm chạy trên nó. Công nghệ ảo hóa bao gồm một số loại sau:

- **Ảo hóa hệ thống lưu trữ:** Mô phỏng, giả lập việc lưu trữ của các thiết bị vật lý.
- **Ảo hóa hệ thống mạng:** Tiến trình hợp nhất tài nguyên, thiết bị mạng cả phần cứng lẫn phần mềm thành hệ thống mạng ảo, sau đó được phân chia cho các thiết bị.
- **Ảo hóa bộ nhớ:** Cho phép tách rời bộ nhớ và hệ điều hành, không cần lưu trữ vẫn có thể truy cập dữ liệu trên máy bình thường.
- **Ảo hóa hệ thống máy chủ:** Cho phép chạy nhiều máy ảo trên một máy chủ vật lý, có thể giúp quản lý, chia sẻ tài nguyên tốt hơn.

Hiện nay, có rất nhiều công nghệ khác nhau, trong đó công nghệ ảo hóa của VMWare được khá nhiều người tin dùng, và VMWare vSphere chính là một trong số đó. Với vSphere, người quản trị có thể thiết lập môi trường ảo hóa phù hợp cho từng điều kiện khác nhau mà không chịu ảnh hưởng bởi yếu tố cấu hình hay vận hành của hệ thống.

1.2. Container & Docker

Trong khi đó, Docker là một nền tảng giúp phát triển, phân phối và triển khai các ứng dụng một cách nhanh chóng sử dụng công nghệ ảo hóa container.

Nói một cách đơn giản hơn, Docker chính là hệ điều hành dành cho container, container ảo hóa hệ điều hành của máy chủ, cung cấp môi trường tương tự như một máy ảo thực thụ. Việc sử dụng Docker giúp tiết kiệm thời gian, tài nguyên cũng như cho phép quyền kiểm soát đối với mã nguồn, cho phép ứng dụng có thể chạy ổn định ở bất kỳ một máy nào mà không chịu ảnh hưởng bởi yếu tố hệ điều hành.

Trong báo cáo này, nhóm sẽ triển khai ứng dụng mạng đã thực hiện từ môn Mạng máy tính – Máy chủ Proxy cài đặt bằng socket trên hai nền tảng, gồm nền tảng ảo hóa sử dụng VMWare vSphere và đóng gói sử dụng Docker

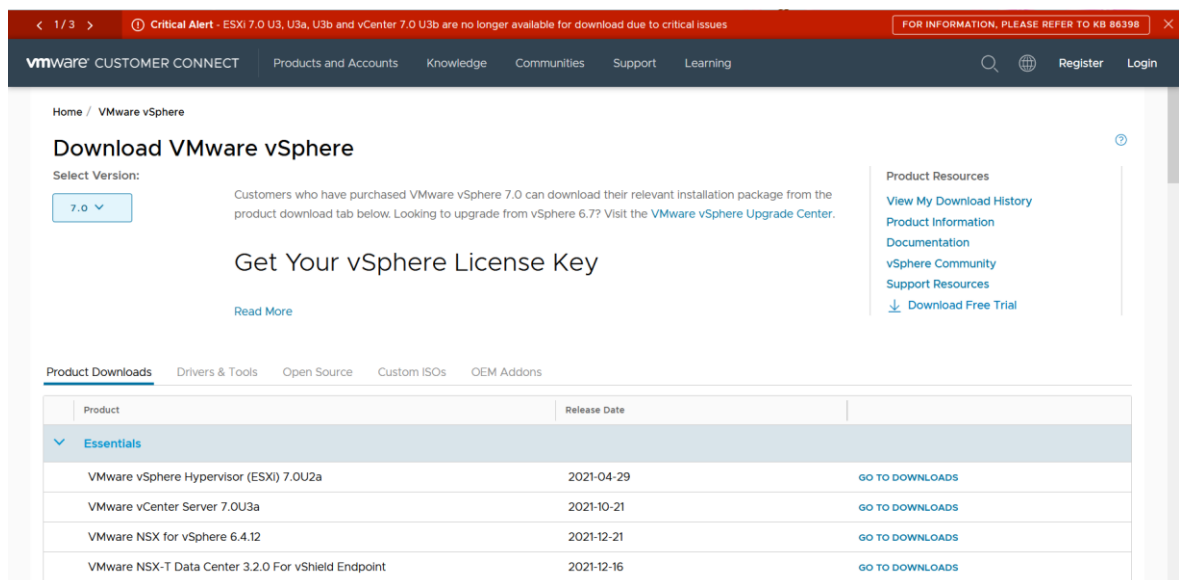
2. TRIỂN KHAI CÀI ĐẶT THỰC TẾ

2.1. Ứng dụng mạng trên nền ảo hóa

Trong phần này, nhóm tiến hành cài đặt ứng dụng sử dụng VMWare VSphere. Các bước thực hiện như sau:

a. *Cài đặt trình điều khiển VMWare ESXi*

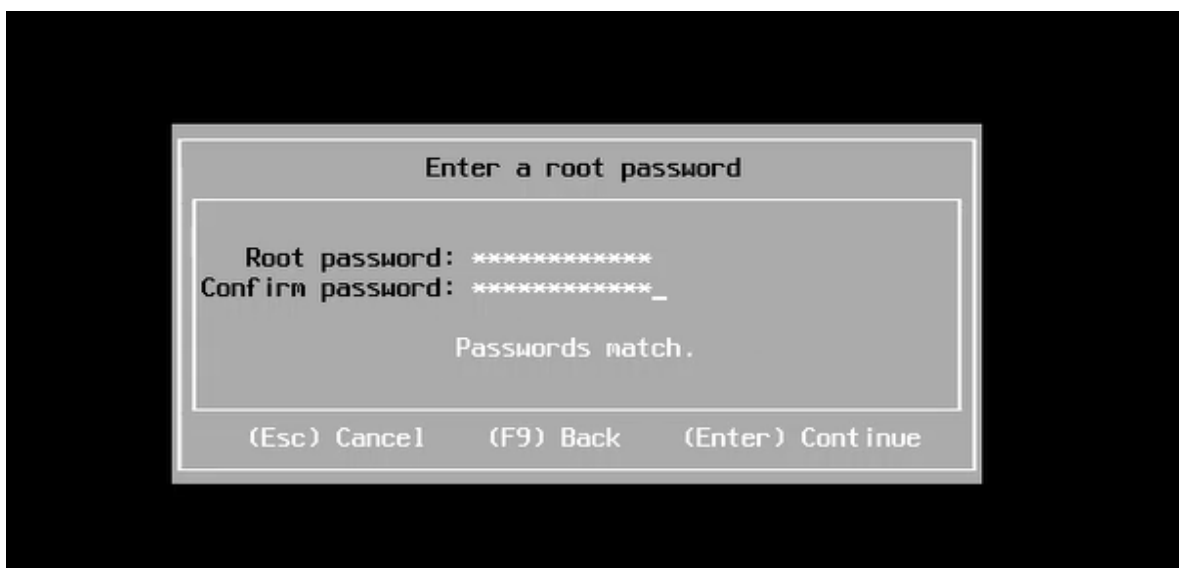
- **Bước 1:** Tiến hành cài đặt VMWare ESXi phiên bản mới nhất từ website chính thức của VMWare. Lưu ý, phải đăng ký tài khoản để được tải phiên bản chính thức. Ngoài ra chỉ cần tải tập tin ISO tương ứng là đủ, bởi lẽ việc cấu hình sẽ diễn ra trên nền của một máy ảo. Nhóm sử dụng phiên bản 6.5, trên màn hình là phiên bản 7.0.



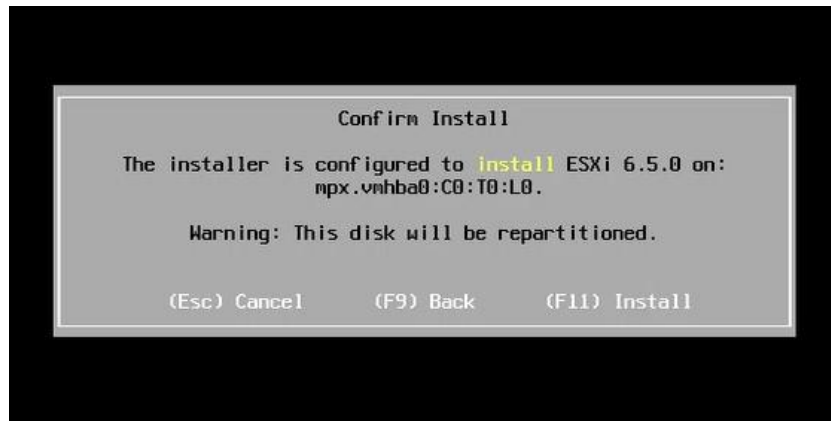
- **Bước 2 (tùy chọn):** Nếu có điều kiện, có thể cài đặt thêm trình quản lý là VMWare vCenter Server. Trong phạm vi giới hạn của báo cáo, nhóm sẽ bỏ qua bước này.
- **Bước 3:** Mở trình quản lý máy ảo (*có thể là VMWare Workstation*). Tạo máy ảo với tập tin ISO đã tải ở bước 1. Cấu hình theo yêu cầu và điều kiện cụ thể. **Lưu ý, cần có ổ cứng trên 100GB mới có thể tiến hành cài đặt được.**
- **Bước 4:** Chạy máy ảo chứa ISO ESXi để tiến hành cài đặt. Hệ thống sẽ chạy các tập tin cài đặt, sau đó xuất hiện ô thoại như hình tiếp theo. Bấm Enter để tiếp tục.



- **Bước 5:** Nhập mật khẩu cho tài khoản root. Lưu ý, mật khẩu này sẽ dùng để đăng nhập vào hệ thống nên cần chú ý kỹ.



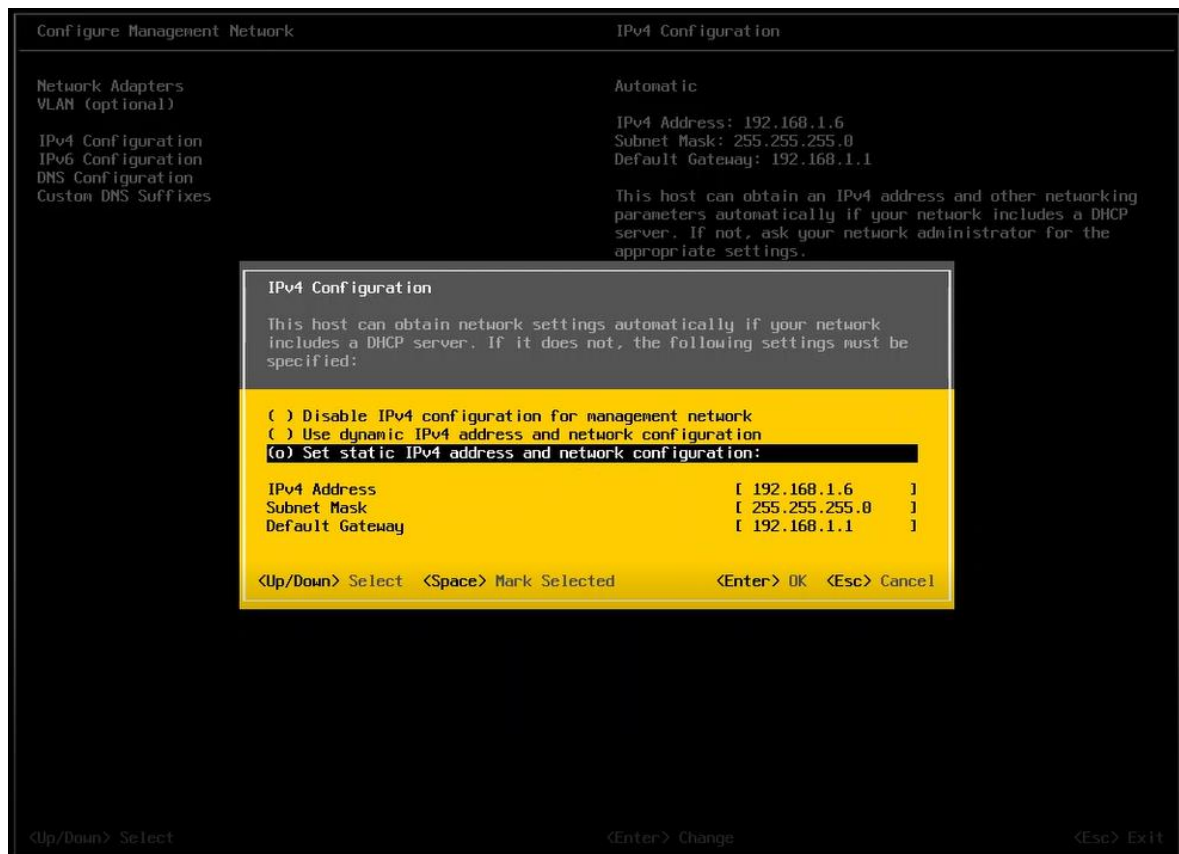
- **Bước 6:** Xác nhận cài đặt, bấm phím F11. Hệ thống tự cài đặt và sau đó sẽ khởi động lại.



- **Bước 7:** Sau khi khởi động lại, màn hình yêu cầu đăng nhập. Nhập mật khẩu ở bước 5, sau đó bấm Enter. Tiến đến màn hình chính, chọn F2 để chuyển sang cấu hình bổ sung.



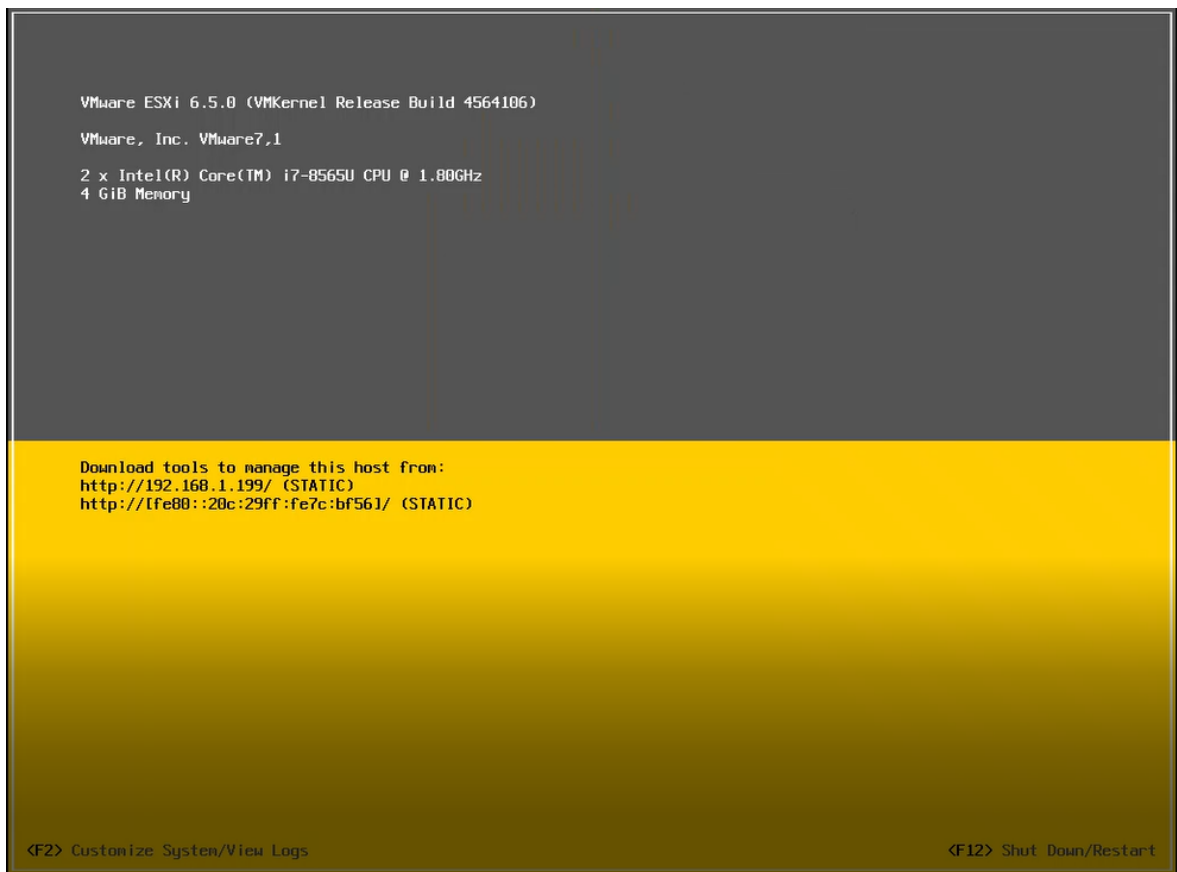
- **Bước 8:** Cấu hình địa chỉ IP để truy cập vào hệ thống ở mục **Configure Management Network → IPv4 Configuration** hoặc **IPv6 Configuration**. Lúc này việc cài đặt ở giai đoạn đầu hoàn tất.



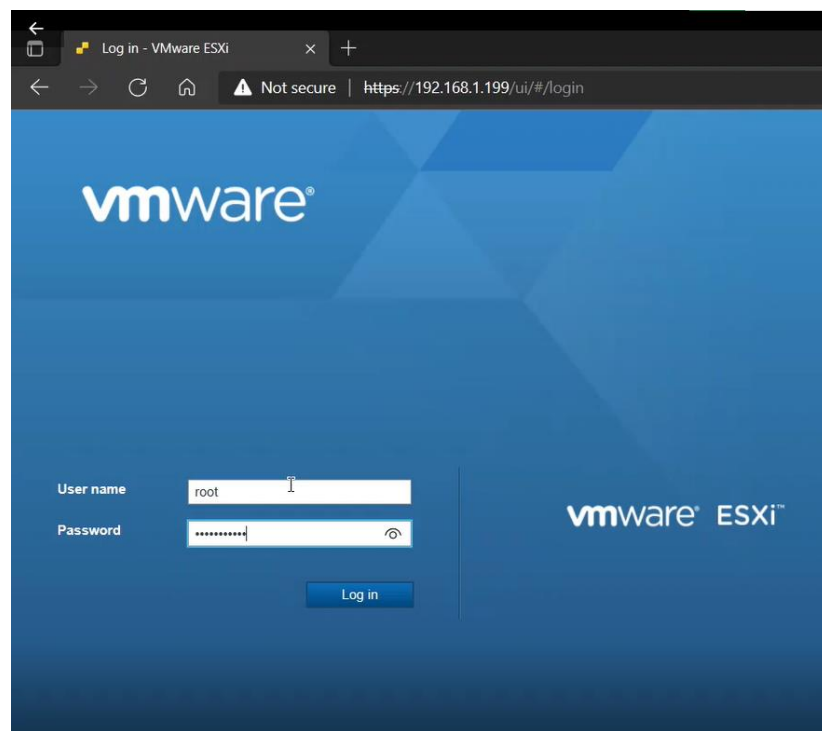
b. Cài đặt và cấu hình các máy ảo

Theo đề bài, ta sẽ cấu hình 3 máy ảo, gồm **hai máy ảo tương ứng máy khách (Client) và một máy ảo tương ứng máy chủ (Server)**. Sau khi thực hiện giai đoạn 2 của quá trình là tạo các máy ảo, ta tiến hành cấu hình và chạy chương trình. Ta thực hiện theo các bước sau:

- **Bước 1:** Mở trình duyệt bất kỳ, gõ địa chỉ được ghi ở các dòng http phía dưới. Ta sẽ thêm /ui vào phía sau địa chỉ để truy nhập vào hệ thống.



- **Bước 2:** Đăng nhập vào hệ thống sử dụng tài khoản đã được tạo ở phần trên.



- **Bước 3:** Tiến hành tạo 3 máy ảo, sử dụng hệ điều hành Ubuntu, gồm 1 máy chủ và 2 máy khách. Đối với điều kiện của nhóm, nhóm sẽ sử dụng phiên bản 32 bit thay vì 64 bit.

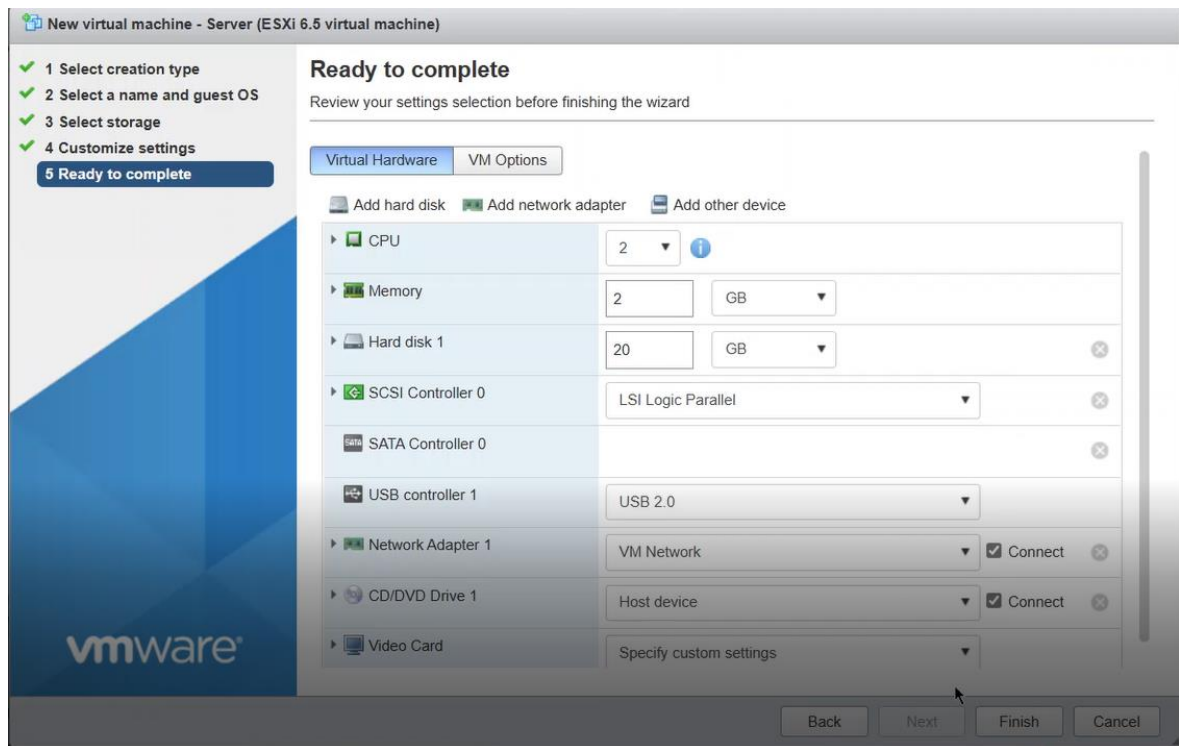
The screenshot shows the 'New virtual machine - Server (ESXi 6.5 virtual machine)' wizard. On the left, a progress bar indicates the steps: 1. Select creation type (checked), 2. Select a name and guest OS (current step), 3. Select storage, 4. Customize settings, and 5. Ready to complete. The main area is titled 'Select a name and guest OS' and asks to 'Specify a unique name and OS'. The 'Name' field contains 'Server'. Below this, a note states: 'Virtual machine names can contain up to 80 characters and they must be unique within each ESXi instance.' Another note says: 'Identifying the guest operating system here allows the wizard to provide the appropriate defaults for the operating system installation.' There are three dropdown menus: 'Compatibility' set to 'ESXi 6.5 virtual machine', 'Guest OS family' set to 'Linux', and 'Guest OS version' set to 'Ubuntu Linux (32-bit)'. At the bottom right, there are buttons for 'Back', 'Next', 'Finish', and 'Cancel'. A mouse cursor is pointing at the 'Next' button.

- **Bước 4:** Chọn bộ nhớ lưu trữ

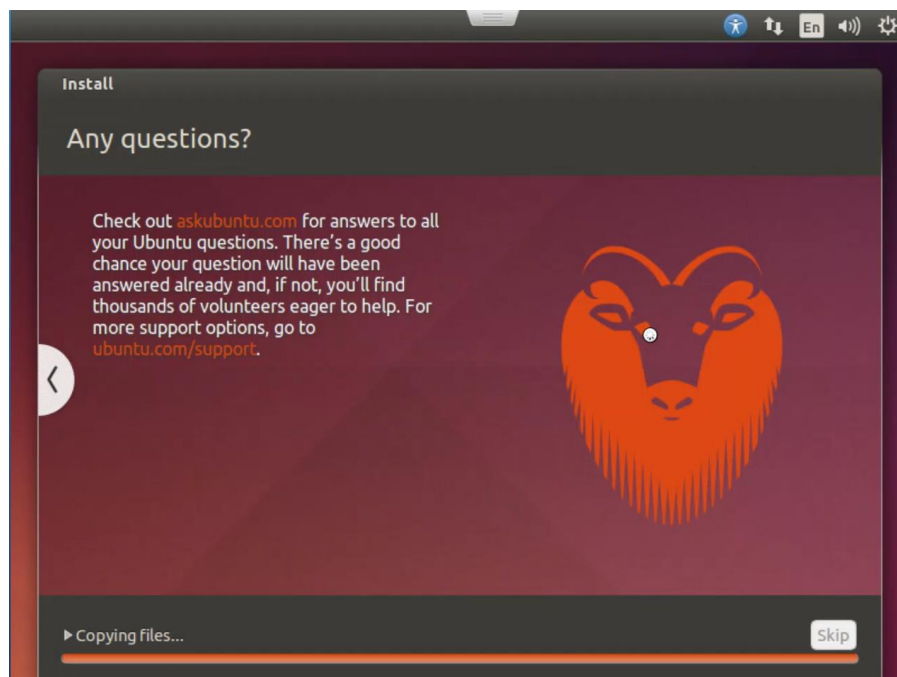
The screenshot shows the 'New virtual machine - Server (ESXi 6.5 virtual machine)' wizard at Step 3: 'Select storage'. The progress bar on the left shows steps 1, 2, and 3 as completed. The main area is titled 'Select storage' and asks to 'Select the datastore in which to store the configuration and disk files.' A note states: 'The following datastores are accessible from the destination resource that you selected. Select the destination datastore for the virtual machine configuration files and all of the virtual disks.' Below this is a table with columns: Name, Capacity, Free, Type, Thin pro..., and Access. The table contains one row for 'datastore1' with values: 92.5 GB, 91.55 GB, VMFS5, Supported, and Single. A '1 items' label is at the bottom right of the table. At the bottom right, there are buttons for 'Back', 'Next', 'Finish', and 'Cancel'. A mouse cursor is pointing at the 'Next' button.

Name	Capacity	Free	Type	Thin pro...	Access
datastore1	92.5 GB	91.55 GB	VMFS5	Supported	Single

- **Bước 5:** Điều chỉnh thông số cài đặt phù hợp

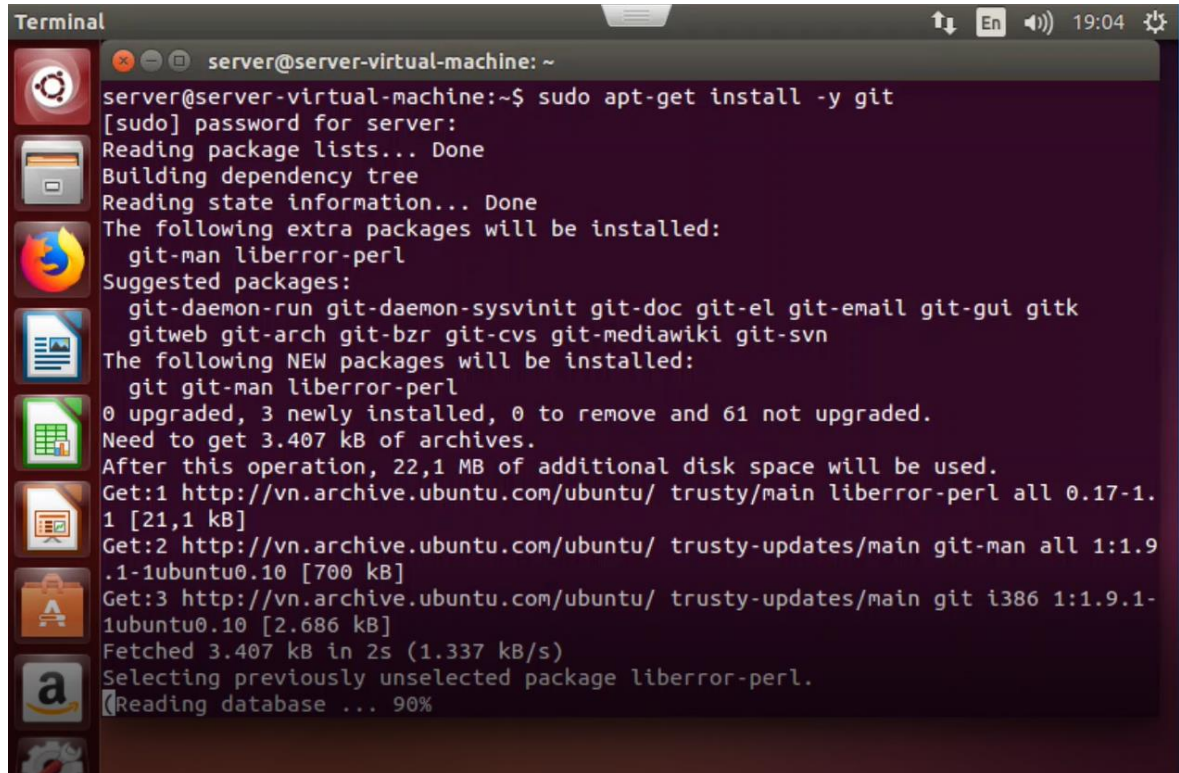


- **Bước 6:** Tiến hành khởi động các máy và cài đặt hệ điều hành cho các máy. Việc cài đặt được tiến hành giống nhau trên các máy, không có nhiều khác biệt.

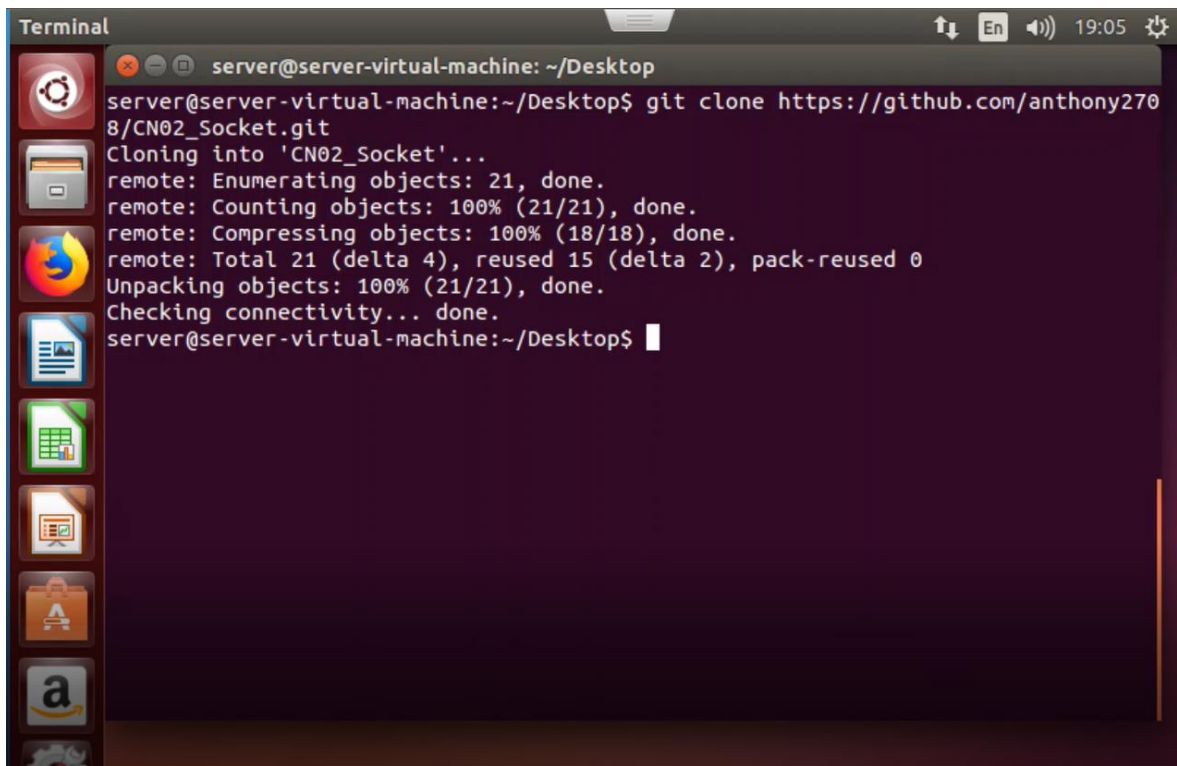


c. Cấu hình ứng dụng và chạy chương trình

- **Bước 1:** Tại máy chủ, tiến hành mở Terminal, gõ các lệnh sau:
 - **Sudo apt-get install -y git** để cài đặt Git
 - **Git clone** https://github.com/anthony2708/CN02_Socket.git để đưa mã nguồn về máy

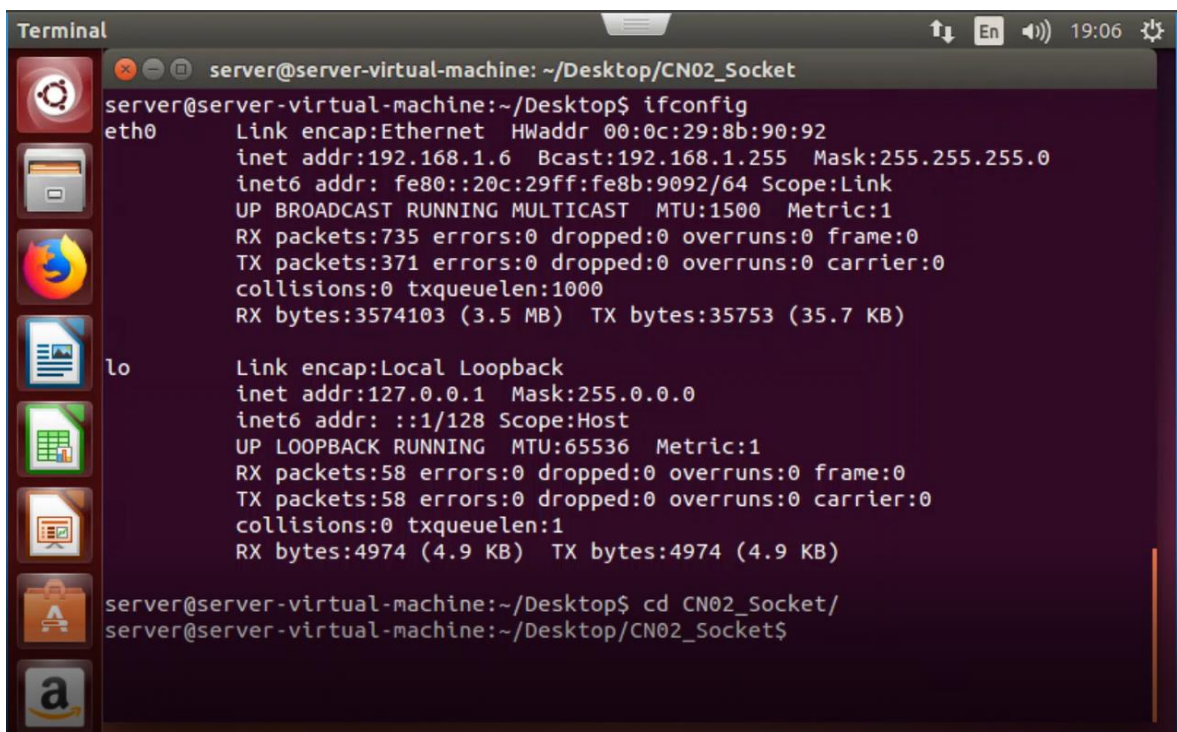


```
Terminal
server@server-virtual-machine: ~
server@server-virtual-machine:~$ sudo apt-get install -y git
[sudo] password for server:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-bzr git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 61 not upgraded.
Need to get 3.407 kB of archives.
After this operation, 22,1 MB of additional disk space will be used.
Get:1 http://vn.archive.ubuntu.com/ubuntu/ trusty/main liberror-perl all 0.17-1.1 [21,1 kB]
Get:2 http://vn.archive.ubuntu.com/ubuntu/ trusty-updates/main git-man all 1:1.9.1-1ubuntu0.10 [700 kB]
Get:3 http://vn.archive.ubuntu.com/ubuntu/ trusty-updates/main git i386 1:1.9.1-1ubuntu0.10 [2.686 kB]
Fetched 3.407 kB in 2s (1.337 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 90%
```

```
Terminal
server@server-virtual-machine: ~/Desktop
server@server-virtual-machine:~/Desktop$ git clone https://github.com/anthony2708/CN02_Socket.git
Cloning into 'CN02_Socket'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 21 (delta 4), reused 15 (delta 2), pack-reused 0
Unpacking objects: 100% (21/21), done.
Checking connectivity... done.
server@server-virtual-machine:~/Desktop$
```

- **Ifconfig** để kiểm tra địa chỉ IP của máy
- **cd CN02_Socket** để vào thư mục chính



```
Terminal
server@server-virtual-machine: ~/Desktop/CN02_Socket
server@server-virtual-machine:~/Desktop$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:8b:90:92
          inet addr:192.168.1.6  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe8b:9092/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:735 errors:0 dropped:0 overruns:0 frame:0
          TX packets:371 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3574103 (3.5 MB)  TX bytes:35753 (35.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:58 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:4974 (4.9 KB)  TX bytes:4974 (4.9 KB)

server@server-virtual-machine:~/Desktop$ cd CN02_Socket/
server@server-virtual-machine:~/Desktop/CN02_Socket$
```

- **python3 start.py** để chạy chương trình. Lúc này người dùng sẽ được yêu cầu nhập IP và cổng, nhập IP của máy chủ và cổng tùy ý.

```
server@server-virtual-machine:~/Desktop$ cd CN02_Socket/  
server@server-virtual-machine:~/Desktop/CN02_Socket$ python3 start.py  
Enter host proxy: 192.168.1.6  
Enter port proxy: 8888  
Proxy server started on port: 8888
```

- **Bước 2:** Tại hai máy khách, tiến hành gõ các lệnh tương tự bước 1, tuy nhiên ở lệnh cuối cùng sẽ là **python3 client.py** (*do máy thực hiện không đủ cấu hình*).
- **Bước 3:** Nhập thông tin và đường dẫn, kiểm tra kết quả vận hành.

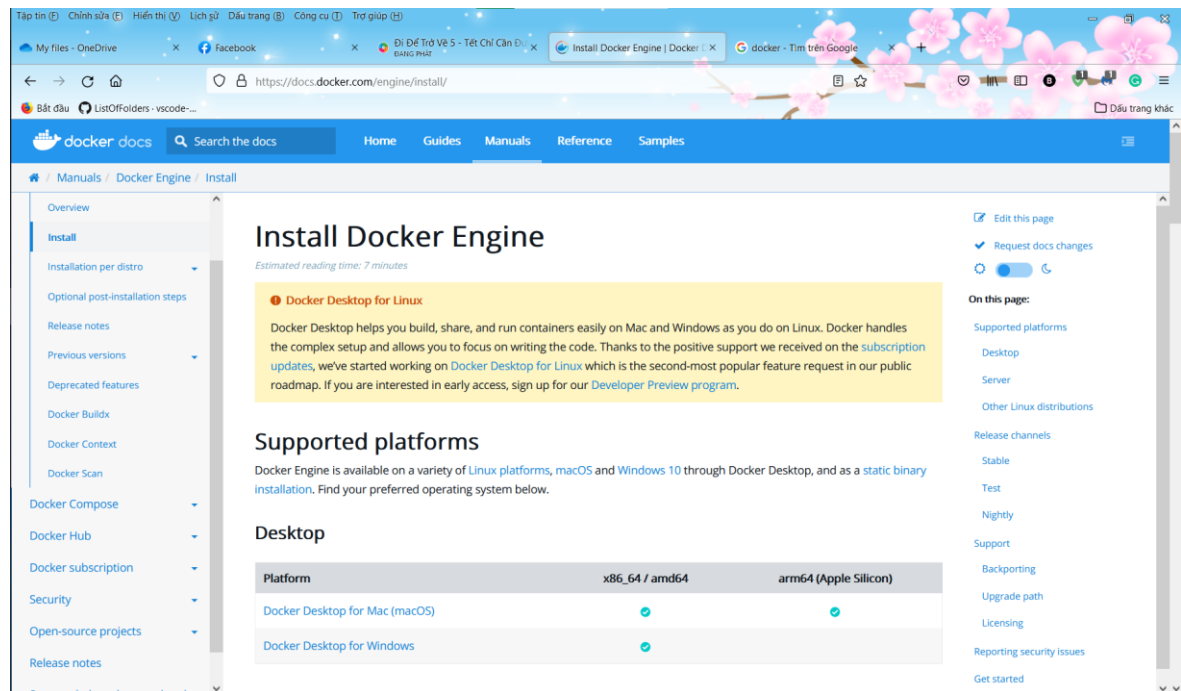
Đến đây, ứng dụng đã có thể vận hành bình thường trên nền ảo hóa.

2.2. Ứng dụng mạng trên Docker

Trong phần này, nhóm tiến hành cài đặt ứng dụng lập trình Socket sử dụng Docker. Các bước thực hiện như sau:

a. *Cài đặt Docker phiên bản mới nhất*

- **Bước 1:** Tải phiên bản Docker mới nhất từ website chính thức, tùy thuộc vào phiên bản hệ điều hành của máy chính.



- **Bước 2:** Sau khi tải bộ cài, tiến hành kích hoạt bộ cài và thực hiện theo hướng dẫn. Đối với hệ điều hành Microsoft Windows, ta có thể lựa chọn việc Docker có thể chạy trên nền Windows Subsystem for Linux trong quá trình cài đặt. Sau khi cài đặt xong, có thể kiểm tra phiên bản mới nhất bằng cách mở Windows Terminal, gõ lệnh **docker --version**. Kết quả như ở hình dưới nghĩa là đã cài đặt thành công.

```
Command Prompt
Microsoft Windows [Version 10.0.19042.1415]
(c) Microsoft Corporation. All rights reserved.

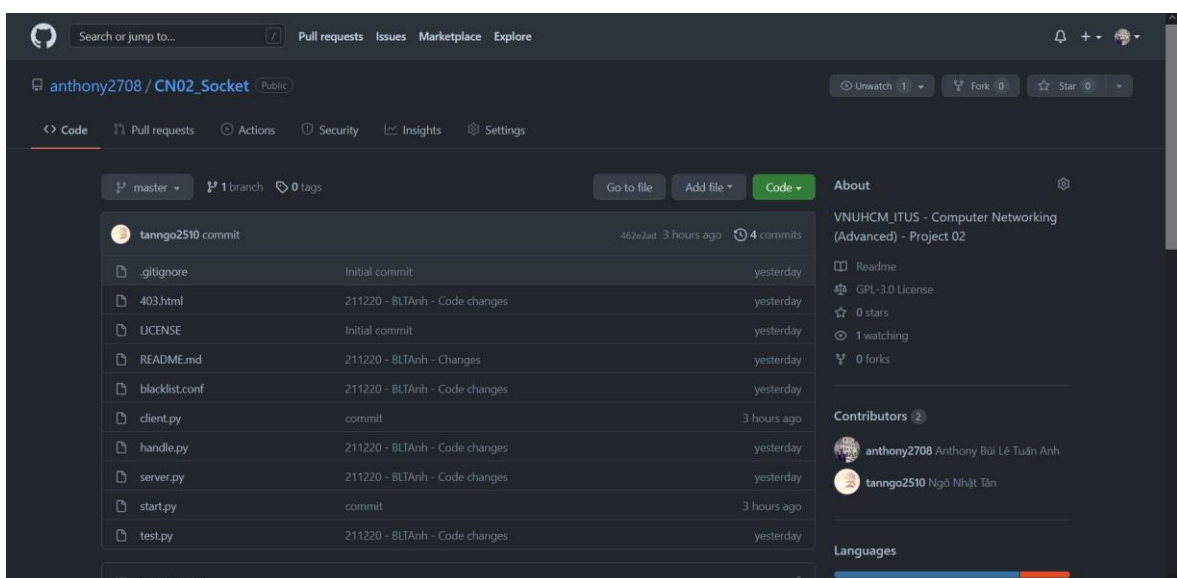
C:\Users\ADMIN>docker --version
Docker version 20.10.11, build dea9396

C:\Users\ADMIN>|
```

- **Bước 3:** Tiến hành đăng nhập vào hệ thống Docker, bằng cách gõ lệnh **docker login**. Nếu chưa có tài khoản, tiến hành đăng ký ở website chính thức của Docker (Docker Hub). Nhập tên đăng nhập và mật khẩu, nếu gặp được câu lệnh **Login successfully** tức là đã đăng nhập thành công.

b. Cài đặt và cấu hình các container

- **Bước 1:** Đối với Docker, người ta sẽ tải lên mã nguồn của chương trình lên một nền tảng quản lý. Cụ thể nhóm sử dụng Github. Tiến hành **cài đặt Git, tạo tài khoản Github**, đồng thời **tạo một bộ quản lý (repository)** trên Github để tải mã nguồn. Như ở hình dưới là mã nguồn của chương trình đã được đưa lên Github.



- **Bước 2:** Tiến hành viết **Dockerfile**. Đây chính là tập tin có tính chất cực kỳ quan trọng để điều khiển toàn bộ chương trình. Tập tin này cụ thể sẽ được chạy để **tạo 1 image** (gọi đơn giản là bản mẫu). Các container sử dụng bản mẫu này sẽ sử dụng và **chạy toàn bộ thông số kỹ thuật** được định nghĩa trong Dockerfile. **Dockerfile** có nội dung như sau:

```
Dockerfile > ...
    Ngô Nhật Tân, 3 hours ago | 1 author (Ngô Nhật Tân)
1  # Xây dựng image mới từ image ubuntu:latest
2  FROM ubuntu:latest
3
4  # Cập nhật apt và cài đặt git
5  RUN apt-get update && apt-get install -y git
6  # Cài đặt python3
7  RUN apt-get install -y python3
8  # Thiết lập thành thư mục /home/github
9  WORKDIR /home/github
10 # Clone repo từ github
11 RUN git clone https://github.com/anthony2708/CN02_Socket.git
12 # Thay đổi thư mục
13 WORKDIR /home/github/CN02_Socket
14
```

○ *Giải thích chương trình:*

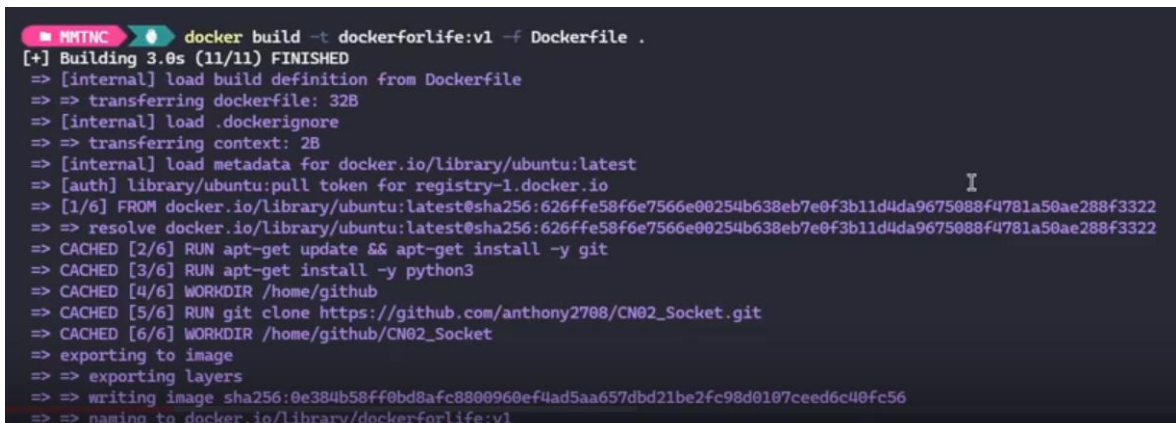
- **Dòng 1:** Xác định hệ điều hành trên image. Cụ thể ở đây, hệ điều hành trên image là **Ubuntu, phiên bản mới nhất**.
- **Dòng 3:** Thực hiện hai công việc
 - Chạy lệnh **update (cập nhật)** toàn bộ các gói của image
 - Chạy lệnh **install (cài đặt)** git vào image
- **Dòng 7:** Chạy lệnh **install (cài đặt)** Python 3 vào image
- **Dòng 9:** Thiết lập thư mục hoạt động của image, cụ thể ở đây là **/home/github**
- **Dòng 11:** Chạy lệnh **clone** (tức tải toàn bộ mã nguồn từ Github, nơi lưu trữ mã nguồn như đã xác định ở bước 1)

- **Dòng 13:** Xác định thư mục hoạt động mới của image, cụ thể ở đây là đi vào trong thư mục **CN02_Socket**

c. Cấu hình ứng dụng và chạy chương trình

Theo đề bài, ta sẽ cấu hình 3 container, gồm **hai container cho máy khách (Client)** và **một container cho máy chủ (Server)**. Sau khi thực hiện giai đoạn 2 của quá trình là viết tập tin tạo image phục vụ cho việc tạo và chạy container, ta tiến hành cấu hình và chạy các container dựa trên image có sẵn. Ta thực hiện theo các bước sau:

- **Bước 1:** Tiến hành tạo image. Chạy lệnh **docker build -t <nhãn thông tin> -f <Dockerfile> .** Khi chạy lệnh này, 1 bản mẫu sẽ được tạo ra. Như trên hình, nhãn thông tin tương ứng với tên và phiên bản là **dockerforlife:v1**. Ta có thể thêm **--force-rm** trước **-f** để có thể xóa và tạo lại image mới nếu đã có image trùng.



```

MHTNC docker build -t dockerforlife:v1 -f Dockerfile .
[+] Building 3.0s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
=> => resolve docker.io/library/ubuntu:latest@sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
=> CACHED [2/6] RUN apt-get update && apt-get install -y git
=> CACHED [3/6] RUN apt-get install -y python3
=> CACHED [4/6] WORKDIR /home/github
=> CACHED [5/6] RUN git clone https://github.com/anthony2708/CN02_Socket.git
=> CACHED [6/6] WORKDIR /home/github/CN02_Socket
=> exporting to image
=> => exporting layers
=> => writing image sha256:0e384b58ff0bd8afc8800960ef4ad5aa657dbd21be2fc98d0107ceed6c40fc56
=> => naming to docker.io/library/dockerforlife:v1
  
```

- **Bước 2:** Tiến hành tạo và chạy container số 1 – container chứa máy chủ. Chạy lệnh **docker run --name=<tên container> --hostname=<tên host> -it <nhãn thông tin>**. Ở đây tên container và tên host là **Server**,

nhãn thông tin là **dockerforlife:v1**.

```
MMTNC docker run --name=Server --hostname=Server -it dockerforlife:v1
root@Server:/home/github/CN02_Socket# awk 'END{print $1}' /etc/hosts
172.17.0.2
root@Server:/home/github/CN02_Socket# cat blacklist.conf
iuh.edu.vn
root@Server:/home/github/CN02_Socket# python3 start.py
Enter host proxy: 172.17.0.2
Enter port proxy: 8888
Proxy server started on port: 8888
```

○ *Giải thích chương trình:*

- Sau khi chạy xong dòng lệnh **docker run**, hệ thống sẽ nhảy vào bên trong thư mục tương ứng.
 - Bởi mỗi container máy chủ có một địa chỉ IP riêng (giống như một máy chủ Proxy thật sự bên ngoài), do đó phải xác định được địa chỉ IP để có thể chạy được máy chủ. **Dòng lệnh đầu tiên** cho biết địa chỉ này.
 - **Dòng lệnh thứ hai** có tác dụng cho biết nội dung của tập tin **blacklist.conf**, tức là tập tin chứa các website theo chuẩn HTTP bị chặn theo yêu cầu của Proxy.
 - **Dòng lệnh thứ ba** là dòng lệnh chạy chương trình máy chủ Proxy. Chương trình được viết bằng Python 3, do đó, lệnh chạy có phần mở đầu là **python3**
 - **Sau khi chạy dòng lệnh thứ 3**, hệ thống sẽ yêu cầu người dùng nhập địa chỉ IP và cổng giao tiếp. Đối với địa chỉ IP ta nhập địa chỉ IP của container. Cổng giao tiếp tùy ý.
- **Bước 3:** Tiến hành tạo và cấu hình container số 2 và số 3 – container chứa hai máy khách. Lệnh chạy **tương tự bước 2** ở trên, thay **tên container và tên host** là **Client1** và **Client2**. Mở 2 cửa sổ dòng lệnh để thực hiện đơn giản hơn.

```
MMTNC docker run --name=Client1 --hostname=Client1 -it dockerforlife:v1
root@Client1:/home/github/CN02_Socket# python3 client.py
Enter host proxy: 172.17.0.2
Enter port proxy: 8888
```

```
MMTNC docker run --name=Client2 --hostname=Client2 -it dockerforlife:v1
root@Client2:/home/github/CN02_Socket# python3 client.py
Enter host proxy: 172.17.0.2
Enter port proxy: 8888
```

○ **Chú ý:**

- Sau khi lệnh chạy, ta gọi lệnh: **python3 client.py**. Nhập thông tin địa chỉ IP và cổng giao tiếp của container chứa **MÁY CHỦ**, không phải máy khách.
- Ngay sau khi nhập xong dòng lệnh này, chương trình sẽ yêu cầu người dùng nhập địa chỉ trang web cần truy cập. Người dùng có thể nhập địa chỉ, **miễn rằng đây là địa chỉ HTTP**.

Sau đây là một số hình ảnh minh họa quá trình kết nối thành công giữa các container với nhau.

```
Enter domain: www.iuh.edu.vn
b'HTTP/1.1 403 Forbidden\nContent-Type: text/html\n\n\n'
```

```
MMTNC docker run --name=Server --hostname=Server -it dockerforlife:v1
root@Server:/home/github/CN02_Socket# awk 'END{print $1}' /etc/hosts
172.17.0.2
root@Server:/home/github/CN02_Socket# cat blacklist.conf
iuh.edu.vn
root@Server:/home/github/CN02_Socket# python3 start.py
Enter host proxy: 172.17.0.2
Enter port proxy: 8888
Proxy server started on port: 8888
172.17.0.3 Request: Host: www.example.com
BLOCKED: www.iuh.edu.vn
```

Một trang web bị chặn theo yêu cầu của máy chủ Proxy

```
Enter domain: www.example.com
b'HTTP/1.0 501 Not Implemented\r\nContent-Type: text/html\r\nContent-Length: 357\r\nConnection: close\r\nDate: Tue, 21 Dec 2021 09:40:06 GMT\r\nServer: ECSF (oxr/8380)\r\n\r\n<?xml version="1.0" encoding="iso-8859-1"?>\n<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"\n"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">\n<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">\n<head>\n<title>501 - Not Implemented</title>\n</head>\n<body>\n<h1>501 - Not Implemented</h1>\n</body>\n</html>\n'
```

Lỗi 501: Website kết nối sử dụng HTTPS.

Đến đây, ứng dụng lập trình Socket đã có thể vận hành bình thường trên các container sử dụng Docker.

TÀI LIỆU THAM KHẢO

1. Tìm hiểu về Docker:
 - a. <https://aws.amazon.com/vi/docker/>
2. Slides bài giảng Ảo hóa & Containers, môn Mạng máy tính nâng cao.
3. Tìm hiểu về công nghệ ảo hóa:
 - a. <https://viettelidc.com.vn/tin-tuc/cong-nghe-ao-hoa-cac-kieu-ao-hoa-co-ban>
 - b. <https://quantrimang.com/ao-hoa-la-gi-tai-sao-ban-nen-su-dung-cong-nghe-nay-157936>
 - c. <https://viettelidc.com.vn/tin-tuc/moi-dieu-ve-cong-nghe-ao-hoa-vmware>
 - d. <https://viettelidc.com.vn/tin-tuc/tim-hieu-ve-cong-nghe-ao-hoa-va-ao-hoa-vmware>