

CHƯƠNG 1

BUỔI THỰC HÀNH SỐ 1

Ch! "ngngi (i bi) u * ogc + q , nt - m. ngb! / ngdgtag
h) O' bnhLi n(phi n b , n U b d % 04) wgc + h1 it 2 m
ph 3ng m . ng Kba tn n * n t , ng , d d b e Tr 4ng t a 5a
b! "ngs 6 b b \$ bnhO "ngi , nO! 2b! (ngd 7nbi t 8t 9ng
b (ch : gi psi nhiv n v (i v) c ; d+ngKba t ogm
ph 3ngm . ng g h .

1.1!CÔNG CỤ QUẢN TRỊ MẠNG TRÊN LINUX

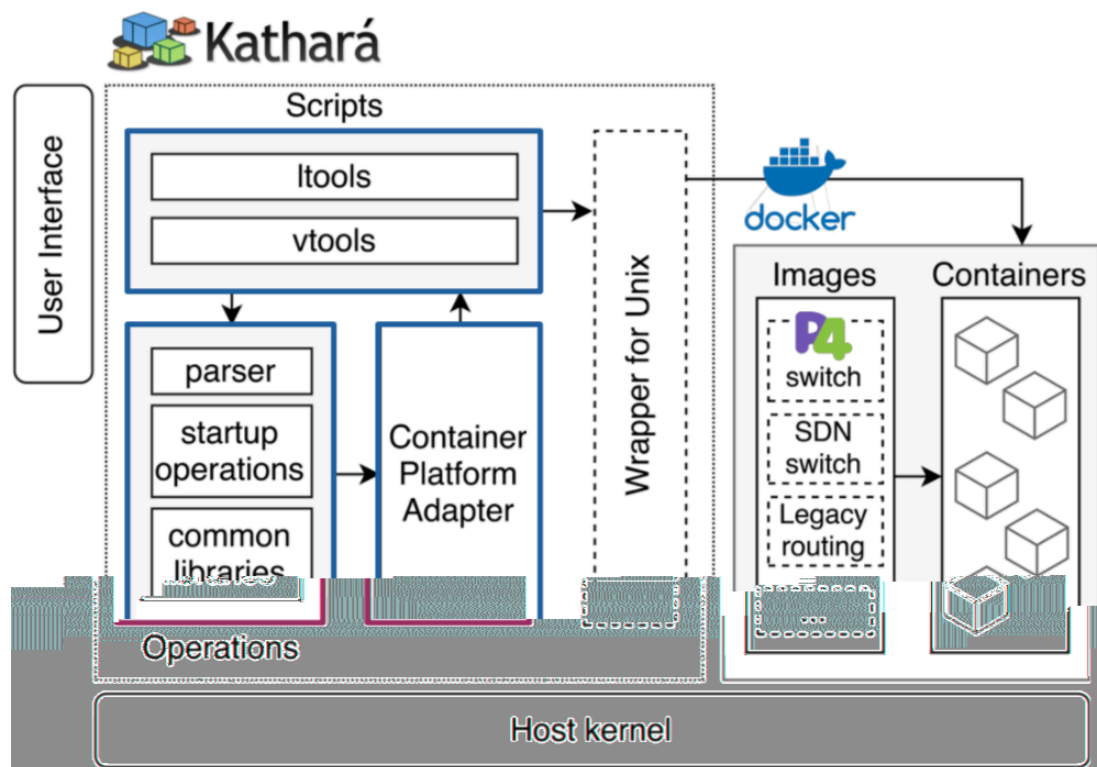
- ! ☐ ping: công cụ cho phép gửi 1 gói tin đến từ địa chỉ IP máy nguồn đến địa chỉ IP máy đích. Nếu như địa chỉ IP máy đích là tồn tại, ping trên máy đích sẽ tự động hồi đáp bằng 1 gói tin ngược lại máy nguồn. Cả 2 gói tin ping này đều chứa thông điệp ICMP - Internet Control Message Protocol.
- ! ☐ ifconfig: công cụ cho phép cấu hình giao diện mạng (network interface) của máy, ví dụ: đặt địa chỉ IP và netmask, tắt hoặc mở giao diện mạng.
- ! ☐ tcpdump: công cụ cho phép bắt các gói tin luân chuyển qua một hoặc nhiều giao diện mạng. Công cụ này cung cấp 2 chức năng 1 ớn, là packet sniffing và packet analyze với thư viện lệnh phong phú.
- ! ☐ route: công cụ cho phép xem bảng dẫn đường hiện tại của host.
- ! ☐ traceroute: công cụ cho phép lần vết của dữ liệu luân chuyển qua host.

1.2!CÔNG CỤ MÔ PHỎNG MẠNG KATHARÁ

Kba là một công cụ mã nguồn mở có chi phí thấp, hiệu suất cao được triển khai trên nền tảng ảo hóa Docker nhằm thực hiện giả lập (emulate) một hoặc nhiều hệ thống mạng từ đư gi ản cho đến phức tạp. Thực chất Kba là phiên bản nâng cao của Nk i t , một công cụ mô phỏng mạng cũng đã đ ạt được những thành công trong hỗ trợ dạy và học Mạng máy tính tại các Trường Đại học có đào tạo chuyên ngành Công nghệ thông tin.

Đ ể m nổi trội của Kba so với người tiền nhiệm của nó là nhờ vào nền tảng ảo hóa Dba đ ể mô phỏng các thiết bị thường gặp trong mạng như Router, Switch, Web Server, DNS Server...dưới dạng các máy ảo Linux chỉ có giao diện dòng lệnh. Các thiết bị mạng (máy ảo) này hoạt động dưới hình thức là các bộ chứa (Container) được quản lý ởi Dba . Một số ưu điểm nổi bật

của môi trường ảo hóa dựa trên các bộ chứa mà Docker đem lại đó là: linh động, nhanh, nhẹ, đồng nhất và đóng gói. Các máy ảo được tạo ra có thể dễ dàng được tùy chỉnh (customize) lại theo ý đồ của người quản trị mạng. Chẳng hạn, triển khai Apache Spark cho các hệ thống ứng dụng tính toán dữ liệu lớn (Big Data). Mô hình tổng quan của môi trường mạng ảo được mô phỏng bởi K8s trên nền tảng Docker được miêu tả trong hình 1.1.



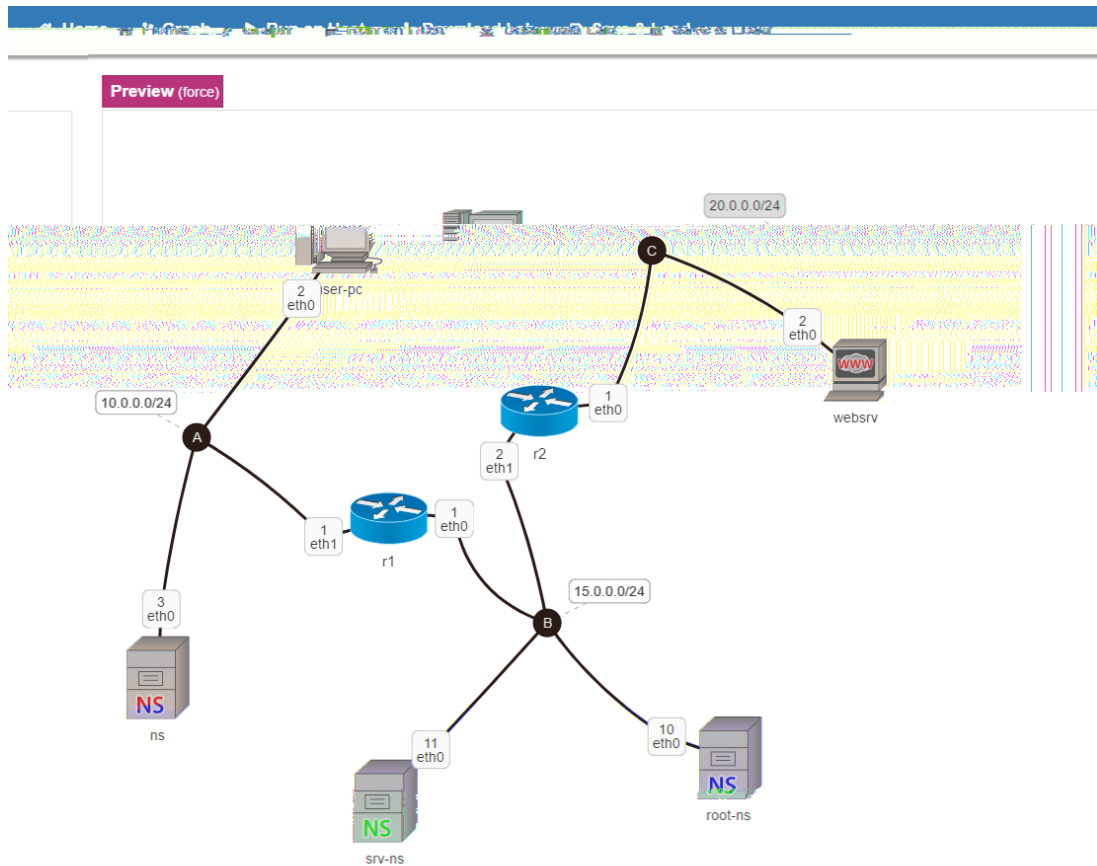
Hình 1.1 Mô hình tổng quan của môi trường mạng ảo dựa trên Docker. Nguồn: K8s

Trong các đánh giá về hiệu năng hoạt động của các phần mềm mô phỏng mạng ảo (Network Emulator), K8s đạt được những đánh giá tốt theo nhiều tiêu chí khác nhau (về hiệu suất sử dụng CPU, tiêu hao bộ nhớ, thời gian khởi động và thực thi...). Chính vì vậy, trong tài liệu hướng dẫn thực hành Mạng máy tính CT112, chúng tôi lựa chọn giới thiệu và sử dụng K8s mô phỏng các hệ thống mạng minh họa cho kiến thức lý thuyết về Mạng máy tính đã giảng dạy.

Việc sử dụng K8s để mô phỏng mạng có thể được hỗ trợ thêm bởi công cụ NetLab. Đây là công cụ được phát triển kèm theo

¹ <http://www.k8s.org/vn/ng/>.

Katacoda với mục đích cung cấp môi trường đồ họa làm giảm tính phức tạp trong việc xây dựng mô hình mạng dưới dạng các máy ảo. Hình 1.2 dưới đây mô tả một mạng ảo được xây dựng bởi công cụ Netlab.

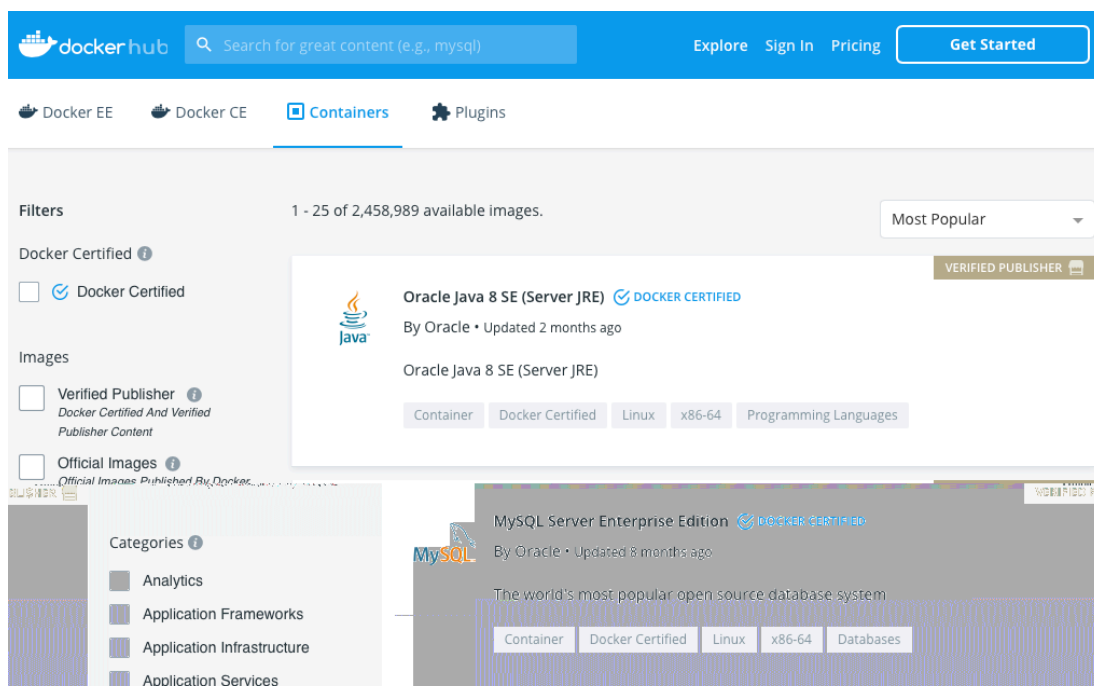


Hình 1.2 Hình minh họa mạng ảo & nhúng trong môi trường Netlab

1.2.1! Các đặc điểm chính của Kathará

- ! ☐ Hoạt động trên nền tảng ảo hóa Docker.
- ! ☐ Máy thực (host machine) triển khai Kathará có thể chạy hệ điều hành Linux, Window hoặc MacOS. Tuy nhiên, môi trường phù hợp nhất và được khuyến khích triển khai Kathará đó là Linux.
- ! ☐ Các máy ảo (virtual machine) được xem là 1 nút (node) trong mô hình mạng được mô phỏng.

- ! ☐ Các máy ảo được tạo ra dưới dạng UML (User Mode Linux) và được quản lý như là các Containerized Applications² trên nền tảng Docker của máy thực. Các máy ảo này chiếm rất ít tài nguyên máy thực (CPU, Mem...)
- ! ☐ Các máy ảo UML mặc nhiên được tạo ra từ cùng 1 ảnh (image) có sẵn hoặc từ các ảnh mà người dùng xây dựng lại theo nhu cầu sử dụng. Docker cung cấp thư viện ảnh tại Docker Hub³ rất phong phú và sẵn dùng. Hình 1.3 là danh sách các ảnh thông dụng mà Docker Hub cung cấp:



Hình 1.3 Danh sách các ảnh thông dụng mà Docker Hub cung cấp. Nguồn: Docker Hub

- ! ☐ Các máy ảo UML liên thông, kết nối với nhau dễ dàng do cùng hoạt động trong một môi trường ảo hóa là Docker. Ngoài ra, các máy ảo UML cũng có thể kết nối đến máy thực cục bộ (local machine) hoặc đến các máy thực ở xa (remote machine).
- ! ☐ Tại thời điểm hiện tại, Kathara chỉ cho phép mô phỏng mạng ảo với hình thái trục (bus topology) sử dụng chuẩn Ethernet II.

² <https://www.docker.com/resources/what-container>

³ <https://hub.docker.com/u/kathara/>

1.2.2! Hệ thống tập lệnh trong Kathará

Kathará cung cấp 2 tập lệnh với phần tiếp đầu ngữ (prefix) là: ! " # \$ % & ' () * và + " # \$ % & ' () . 2 tập lệnh này được sử dụng trên màn hình điều khiển (terminal) của máy thực. Trong đó:

! [] ! " # \$ % & ' () được sử dụng để tương tác với một máy ảo đơn lẻ và tất cả các liên kết mạng tới nó. Tập lệnh này chủ yếu bao gồm các lệnh như: chạy (run), dừng (stop), xóa bỏ (remove), truy vấn (info).

! [] + " # \$ % & ' () được sử dụng để tương tác với nhiều hoặc tất cả các máy ảo cùng một lúc trong một mô hình mạng ảo. Tập lệnh này cho phép tự động hóa nhiều thao tác quản lý trên các máy ảo qua hệ thống thư mục chứa các file thiết lập cấu hình (configuration files) mà Kathará quy định. Hầu hết các lệnh được cung cấp bởi v-commands đều có trong l-commands.

Một số lệnh thông dụng trong tập lệnh ! " # \$ % & ' () và + " # \$ % & ' () được trình bày trong Bảng 1.1

	v-commands	l-commands
),&-, *	Khởi động 1 máy ảo. Ví dụ: !),&-, * "" ./ *012*3#4	Khởi động một hoặc nhiều máy ảo trong 1 mạng ảo. Ví dụ: +),&-,
#-&)/ *	Dừng (forcefully stopped) 1 máy ảo. Máy ảo đó sẽ dừng mà không quan tâm đến các hoạt động bên trong máy ảo đã hoàn thành hay chưa. Tài nguyên cấp phát cho máy ảo vẫn duy trì để tái khởi động. Ví dụ: !#-&)/ *3#4	Dừng (forcefully) một hoặc nhiều máy ảo trong mạng. Ví dụ: +#-&)/ *3#4 *
/&+, *	Dừng (gracefully stopped) 1 máy ảo. Máy ảo đó sẽ dừng khi tất cả các hoạt động bên trong đã hoàn thành. Tài nguyên cấp phát cho máy ảo vẫn duy trì để tái khởi động. Ví dụ: !/&+, *3#4	Dừng (gracefully stopped) một hoặc nhiều máy ảo. Ví dụ: +/&+,

#+.&' *	Dừng hoạt động của máy ảo và các kết nối liên quan đến máy ảo đó. Tài nguyên cấp phát cho máy ảo bị thu hồi. Ví dụ: !#+.& '*3#4	Dừng hoạt động của các máy ảo và các kết nối liên quan đến các máy ảo đó. Tài nguyên cấp phát cho các máy ảo bị thu hồi. Ví dụ: ++.&'
#\$'567 *	Giúp người dùng tạo ra các liên kết mới trên máy ảo (post creation) sau khi máy ảo đã khởi động. Ví dụ: !#\$'567* "" .//*819*3#4	Không có
+6), *	Giúp người dùng quan sát được các thông số hữu ích liên quan đến hoạt động của máy ảo trên máy thực (% CPU, %Mem...) Ví dụ: !+6),*3#4	Không có
:63. *	Không có	Lệnh sử dụng tương tự như lệnh ++.&' . Ví dụ: +:63.
-.),&-, *	Không có	Khởi động lại một hoặc nhiều máy ảo trong mạng. Ví dụ: +-.),&-,*3#4

B, ng 1.1 Nhm)nhv ~~ad~~ ~~ad~~ ~~ad~~

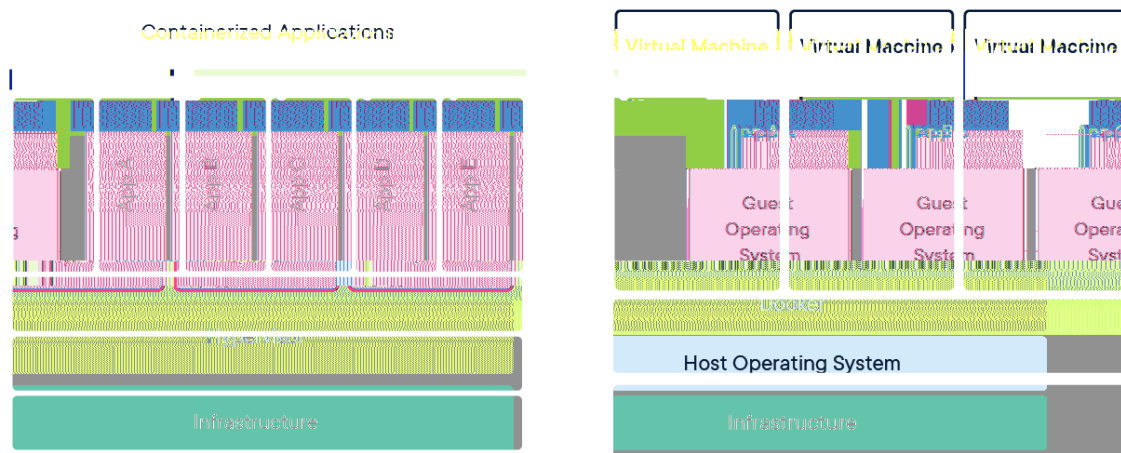
1.2.3! Cài đặt Docker và Kathará

Trong các đánh giá về hiệu năng hoạt động của các phần mềm mô phỏng mạng ảo, ~~Kha~~ đạt được những đánh giá tốt theo nhiều tiêu chí khác nhau (về hiệu suất sử dụng CPU, tiêu hao bộ nhớ, thời gian khởi động và thực thi...). Chính vì vậy, trong tài liệu hướng dẫn thực hành Mạng máy tính CT112, chúng tôi lựa chọn giới thiệu và sử dụng ~~Kha~~ mô phỏng các hệ thống mạng minh họa cho kiến thức lý thuyết về Mạng máy tính đã giảng dạy.

! ☐ Đối với hệ điều hành Ubuntu 18.04, các thao tác cài đặt phải được thực hiện bằng tài khoản người dùng có thẩm quyền.

! ☐ Các lệnh cài đặt được thực hiện trên terminal của Ubuntu 18.04.

- ! □ Docker, Platform cũng như các thiết lập cần thiết cho môi trường thực hành Mạng máy tính CT112 đã được hoàn tất tại các phòng thực hành chuyên sâu về Mạng máy tính của khoa CNTT&TT, Đại học Cần Thơ. Vì vậy sinh viên không cần phải thực hiện lại các công việc dưới đây.
- ! □ Tài khoản người dùng mà sinh viên được phép thao tác trên Ubuntu 18.04 là `ubuntu`. Lưu ý đây không phải là tài khoản có quyền cài đặt các gói phần mềm lên hệ thống.
- 1) □ Cài đặt Docker trên Ubuntu 18.04 (hoặc mới hơn) có thể tham khảo trong tài liệu này⁴. Hình 1.4 dưới đây thể hiện rõ sự khác biệt trong cách thức quản lý các ứng dụng kiểu Containerized Application trên Docker với ứng dụng trên máy ảo truyền thống.



Hình 1.4 ? ngđ +ng tđ @khả tđ Docker, tđ *nh %g. Nguồn: Docker

Một số lưu ý đối với việc cài đặt và thực thi Docker:

- ! □ Khởi động Docker là bắt buộc để chạy Kubernetes. Để khởi động Docker bằng tay (manually) có thể dùng lệnh `sudo systemctl start docker` hoặc `sudo service docker start`. Ngoài ra Docker có thể khởi động tự động cùng hệ điều hành (automatically)⁵.
- ! □ Mặc nhiên chỉ có người dùng gốc (root) mới có quyền thao tác trên Docker. Tuy nhiên, sử dụng tài khoản root là không được khuyến khích mà thay vào đó sẽ sử dụng các tài khoản đã được thêm vào nhóm người dùng có quyền

⁴ <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

⁵ <https://docs.docker.com/config/daemon/systemd/>

thao tác trên Docker. Lệnh thêm 1 người dùng (user) vào nhóm có quyền thao tác trên Docker như sau:

```
);($*);-.%$( " &=*( $#>.-*<$;- " );.- !
```

! □ Có thể kiểm tra thử hoạt động của Docker bằng cách khởi động một Containerized Application từ ảnh là `hello-world`. Ảnh này không có sẵn trên máy cài đặt Docker, chính vì vậy Docker sẽ kết nối đến Docker Hub để tải ảnh về và sử dụng để khởi động Containerized Application. Lệnh sử dụng để thực thi công việc này là:

```
($#>.-*-'*/.++$ " :$-+( *
```

```
[vanlongs-MBP:~ vanlong$ docker run hello-world
```

```
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. (amd64)
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://hub.docker.com/
```

```
For more examples and ideas, visit:
```

```
https://docs.docker.com/get-started/
```

```
vanlongs-MBP:~ vanlong$ █
```

Hình 1.5K 8q , h \$bi l)nhdeanh -wcd

! □ Nếu người dùng có nhu cầu thực thi một ứng dụng hay dịch vụ nào đó, chẳng hạn hệ điều hành Ubuntu trên một Containerized Application, thì có thể tìm kiếm phiên bản phù hợp của ứng dụng và thực thi giống như hướng dẫn trên.

```
($#>.-*)& -#/*;'?';; *
```

2) □ Cài đặt Python3 (trường hợp sử dụng phiên bản thấp hơn 18.04) bằng lệnh:

```
);($*&3, " 6'),& ++*3<./,$'@
```

3) □ Đặt Python3 thay thế cho Python2. 7 có sẵn bằng lệnh sau:

```
&+6&)*3<./,$'A3<./,$'@
```


- 4) □ Trường hợp **Phân phối** bị thiếu thư viện 63&((-)) thì có thể tiến hành cài đặt trực tiếp như sau:

363*6'),&++*63&((-)) *

- 5) □ Cài đặt **Wrapper** để đảm bảo tín an ninh (security) và riêng tư (privilege) trong vận hành giữa các mạng ảo trên **Docker**. Ngoài ra **Wrapper** còn rất cần thiết trong việc mở rộng sử dụng **Kubernetes** giữa các máy tính thực chia sẻ tài nguyên với nhau (shared computer).

);(\$*&3,*6'),&++*?;6+(")).',6&+ *

- 6) □ Cài đặt **Xen** để cho phép truy cập vào các máy ảo và điều khiển trực tiếp chúng. **Xen** thường đã cài đặt sẵn trên các phiên bản của Ubuntu.

);(\$*&3,*6'),&++*B,-% *

- 7) □ Truy cập: <https://github.com/KatharaFramework/Kathara/releases>. Chọn tải về Source code (zip). Giải nén file .zip đã tải về, được thư mục C&/&-&"OD@ED4

- 8) □ Sao chép thư mục C&/&-&"OD@ED4 vào trong thư mục F/\$%.* của máy thực. Đặt lại tên cho thư mục vừa sao chép được là C&/&-&GGHF

- 9) □ Mở file bashrc của người dùng hiện hành trên máy thực bằng lệnh: '&'\$*IFD?&)/-# . Di chuyển đến cuối file D?&)/-# và thêm vào nội dung sau đây để đặt biến môi trường cho **Kubernetes**:

.B3\$-,*JKHCLHMNOGKAIFC&/&-&GGHF?&'
 .B3\$-,*P2HNAQP2HN1QJKHCLHMNOGK
 .B3\$-,*G2JP2HNA1QJKHCLHMNOGKF%&'

- 10) □ Thực thi việc cài đặt **Kubernetes** lên **Docker** bằng lệnh sau đây:

QJKHCLHMNOGKF6'),&?+

- 11) □ Sau khi hoàn tất quá trình cài đặt, **Docker** trên máy cục bộ đã có thông tin của ảnh mà **Kubernetes** dùng để tạo ra các máy ảo. Kiểm tra thông tin ảnh đang được quản lý ở **Docker** bằng lệnh:

(\$#>.-*6%&7.) *

```
[vanlongs-MBP:~ vanlong$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
k8s/etcd              latest             30d3e3030303       6 months ago      624M
k8s/pause             latest             1a0e34436363       6 months ago      728K
k8s/kube-proxy         latest             1a0e34436363       6 months ago      728K
k8s/kube-controller-manager latest             1a0e34436363       6 months ago      728K
```

Hình 1.6 K8s, Docker và Kubernetes

12) Khởi động 1 máy ảo để thử nghiệm việc cài đặt. Nếu máy ảo được khởi động thành công nghĩa là quá trình cài đặt đã hoàn tất chính xác. Lệnh để khởi động 1 máy ảo (đặt tên là 3#4) như sau:

!,&-,* "" ,/* 012*3#4

```
[vanlongs-MBP:~ vanlong$ vstart --eth 0:A pc1
===== Starting lab =====
#4223f1f552253fb43f22a2e83dfc824a17ae5a843845f2a
atch to link d7b644b5ed37/e1e
Applying brctl p
8f=8d38e9764ad/w
net.ipv4.conf.all
net.ipv4.conf.de
net.ipv4.conf.et
net.ipv4.conf.lo
netkit_b21_pc1
vanlongs-MBP:~ v
anlong$
```

Hình 1.7K 8q , h \$bi l)nhstt - hO: Apđ

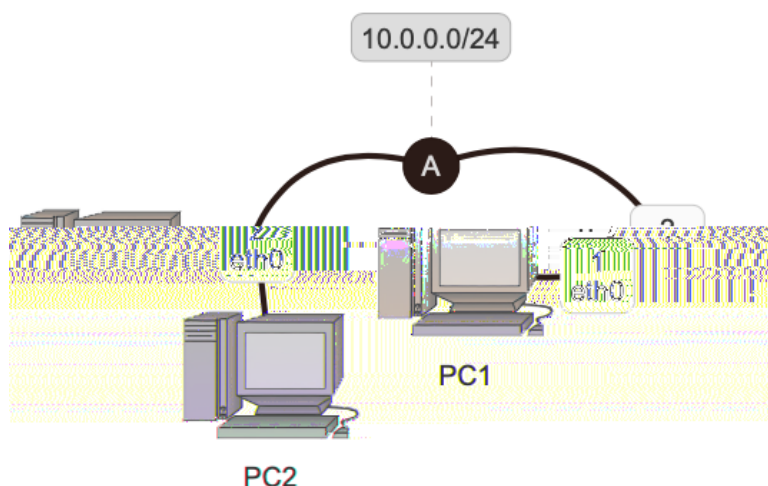
13) Trên máy thực, tắt và xóa máy ảo bằng lệnh:

!#+.&'*3#4 *

1.3! BÀI TẬP THỰC HÀNH

Các thao tác trong phần hướng dẫn thực hành được diễn giải theo từng bước một. Cần chú ý vào công việc cụ thể trong nội dung thực hành, sinh viên có thể thực hiện 1 trong 2 cách sau: 1) s; d+ngl)nhq tnhb hoặc 2) s; d+ngl b GUI C5đ nh ; hoặc kết hợp cả 2 cách. Các phần hướng dẫn thực hành được trình bày dưới đây sẽ sử dụng cách 1).

1.3.1! Bài tập 1



Hình 1.8 Mnhm . ng s; d+ngbo Bít Ap1 Bít Ap2

M+tu : Xây dựng một mạng LAN đơ gi ản theo Phương pháp 1. Các bu ớc thực hiện Bì t Ập1 đượ trình bày chi tiết như sau:

1)□ Quan sát mô hình mạng cần xây dựng và nhận diện các thiết bị, giao diện với các địa chỉ IP đượ gán. Mô hình mạng này đượ thực hiện bởi công cụ ~~Như hình 1.9~~.

2)□ Tạo thư mục Bài Ập1 nằm trong workspace của sinh viên. Trên terminal của máy thực, di chuyển đến thư mục Bài Ập1 bằng lệnh:

```
#(*F/$%.F),;('F<$;-M:$->)3&#.FR&6H&34
```

3)□ Khởi tạo máy ảo 3#4 bằng lệnh:

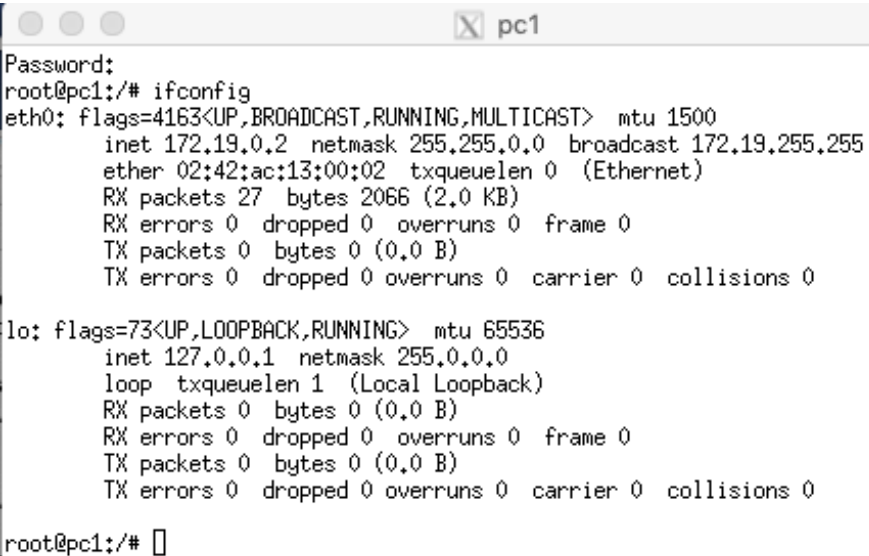
```
!),&-,* "" ./*O12*3#4 .
```

Lệnh này tạo ra 1 máy ảo tên là 3#4 với 1 giao diện mạng eth0. Giao diện eth0 giúp kết nối 3#4 vào 1 nhánh mạng LAN (LAN segment) có tên là A.

Lặp lại công việc tương t ự để tạo ra tiếp 1 máy ảo nữa là 3#8

```
!),&-,* "" ./*O12*3#8
```

4)□ Trên giao diện Xterm của máy ảo 3#4 hoặc 3#8, gõ lệnh 65#\$'567 để kiểm tra cấu hình mạng như hình 1.9



```

Password:
root@pc1:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.19.0.2 netmask 255.255.0.0 broadcast 172.19.255.255
    ether 02:42:ac:13:00:02 txqueuelen 0 (Ethernet)
    RX packets 27 bytes 2066 (2.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@pc1:/# 

```

Hình 1.9K Ập , h \$bì l)nhi ợg mìn , pđ

Câu hỏi:

- ! □ Có những giao diện mạng nào đã đượ tạo ra trong máy ảo?
- ! □ Địa chỉ IP của các giao diện mạng đó là bao nhiêu? Có đúng với địa chỉ IP cần gán mà Bài Ập đã miêu tả hay không?

- 5) ☐ Đặt lại địa chỉ IP cho giao diện eth0 của 3#4 bằng lệnh sau (sử dụng trên máy ảo pc1):

```
65#$'567*./O*4ODODOD4*'.,%&)>*8SSD8SSD8SSDO*?-$&(#&),*4ODODOD8SS
```

Gợi ý sử dụng một lệnh đợ gi ản hơ (vẫn trên máy ảo 3#4)

```
65#$'567*./O*4ODODOD4F8T*;3
```

- 6) ☐ Thực hiện đặt lại địa chỉ IP cho giao diện eth0 của 3#8 tương t ự như đã thực hiện với 3#4. Kiểm tra lại một lần nữa bằng lệnh 65#\$'567 trên 2 máy để đảm bảo việc gán địa chỉ IP mới đã thành công.

- 7) ☐ Trên 3#4 thực hiện gửi gói tin ICMP đến 3#8 bằng lệnh:

```
36'7*4ODODOD8
```

Câu hỏi: Kết quả hiển thị trên màn hình của 3#4 là gì?

- 8) ☐ Lần lượt thực hiện các thao tác sau:

! ☐ Sử dụng lệnh ,-&#.-\$;,. để kiểm tra thông tin đường đi của gói tin từ 3#4 đến 3#8. Kết quả hiển thị cho biết gì?

! ☐ Sử dụng lệnh -\$;,. để hiển thị thông tin bảng vạch đường của 3#4 hoặc 3#8 trong mạng LAN A. Kết quả hiển thị cho biết gì?

- 9) ☐ Trên máy thực, sử dụng lần lượt

```
!#+.&'*3#4
```

```
!#+.&'*3#8
```

để hủy 2 máy ảo vừa tạo và kết thúc Bi t Ap 1.

1.3.2! Bài tập 2

M+&u : Xây dựng một mạng LAN đợ gi ản theo Phương pháp 2. Các bước thực hiện Bi t Ap 2 được trình bày chi tiết như sau:

- 1) ☐ Sử dụng lại mô hình mạng đã cho ở Bi t Ap 1.

- 2) ☐ Tạo thư mục Bàlp2 trong workspace của sinh viên. Thư mục sẽ này chứa các thư mục con và các file cấu hình (.),&-;3U*+&?D#\$'5) theo cấu trúc quy định của K&h&. Trên máy thực, di chuyển đến thư mục Bàlp2 bằng

```
#(*F/$%.F);;('F<$;-M:$->)3&#.FR&6H&38
```

Cấu trúc thư mục Bàlp2 được miêu tả như hình 1.10:



Hình 1.1: Hình ảnh minh họa cấu trúc thư mục và tệp tin

- ! □ File `topology` chứa miêu tả về hình thái (topology) của một mạng ảo
- ! □ Thư mục `pc1` và `pc2` đại diện cho 2 máy ảo của mạng muốn đưa vào hoạt động. Máy ảo sẽ có tên đại diện là tên của thư mục, chẳng hạn: máy ảo số 1 có tên là `pc1`, máy ảo số 2 có tên là `pc2`. Lưu ý Thư mục đại diện cho máy ảo không được chứa ký tự viết hoa (theo quy tắc khởi tạo Containerized Application của Docker).
- ! □ File `pc1.conf` và `pc2.conf` (gọi chung là các file `pc.conf`) là nơi chứa các cấu hình muốn áp dụng cho một máy ảo (`pc1` hoặc `pc2`) ngay khi máy ảo được khởi động cùng mạng ảo. Lưu ý để đảm bảo rằng 1 máy ảo sẽ nhận được cấu hình mong muốn nhờ vào nội dung trong file `pc.conf` thì file `pc.conf` đó phải có tên trùng với tên của thư mục đại diện cho máy ảo. Ví dụ: máy ảo tên là `pc1` được thể hiện qua thư mục tên `pc1` và file tên `pc1.conf`.

Để thực hiện tạo thư mục, file cũng như soạn thảo nội dung cho file, ngoài chế độ đồ họa GUI trên máy thực, người dùng có thể lựa chọn chế độ dòng lệnh để thực hiện

- ! □ Tạo mới thư mục: `mkdir -p pc1 pc2`
 - ! □ Tạo mới file: `touch pc1.conf pc2.conf`
 - ! □ Soạn thảo nội dung file: `vim pc1.conf`
- 3) □ Trên file `topology`, soạn thảo nội dung mô tả hình thái mạng theo thiết kế
- ```

graph LR
 S((Switch)) --- PC1[pc1]
 S --- PC2[pc2]

```
- 4) □ Trên file `pc1.conf`, cấu hình của `pc1` được miêu tả như sau:
- ```

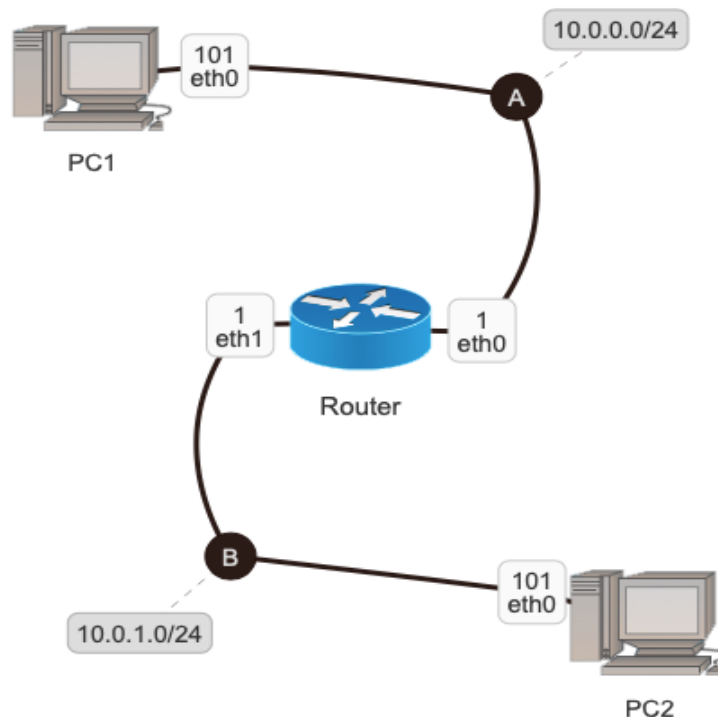
pc1 {
  name pc1
  ip 10.0.0.1
  gateway 10.0.0.254
  dns 8.8.8.8
}
  
```
- Trên file `pc2.conf`, cấu hình của `pc2` được miêu tả như sau:
- ```

pc2 {
 name pc2
 ip 10.0.0.2
 gateway 10.0.0.254
 dns 8.8.8.8
}

```
- 5) □ Trên máy thực, tại thư mục `lab` sử dụng lệnh `docker-compose up` để khởi động mạng ảo `lab` đã tạo.

- ! □ Có thể sử dụng +),&-, để khởi động từng máy ảo riêng lẻ trong trường hợp muốn kiểm tra từng máy ảo. Ví dụ: +),&-, \* 3#4 hoặc +),&-, \* 3#8D
- ! □ Việc khởi động từng máy ảo riêng lẻ trong một mạng ảo sẽ giúp sinh viên kiểm tra được tính đúng đắn của từng nút trong mô hình mạng trước khi khởi động toàn bộ mô hình mạng đó.
- 6) □ Trên 3#4 lần lượt dùng các lệnh 36'7, ,-&#.-\$;,, và -\$;,, để kiểm tra tính liên thông giữa 3#4 và 3#8 trong nhánh mạng LAN A giống như 7) và 8) của Bì t Āp 1
- 7) □ Trên máy thực, sử dụng lệnh +:63. để hủy 2 máy ảo vừa tạo. Kết thúc Bì t Āp 2.

### 1.3.3! Bài tập 3



Hnh 1.1 1Mnhm . ng s; d+ngtag Bì t Āp 3

M+đt : Xây dựng 2 nhánh mạng thuộc cùng một LAN được kết nối bởi 1 router. Các bước thực hiện Bì t Āp 3 được trình bày chi tiết như sau:

- 1) □ Quan sát mô hình mạng cần xây dựng và nhận diện các thiết bị, giao diện với các địa chỉ IP được gán trên các máy ảo.
- 2) □ Tạo thư mục Bì t Āp 3 trong workspace của sinh viên. Thư mục sẽ này chứa các thư mục con và các file cấu hình ZD),&-,;3U\*+&?D#\$'5 ) theo cấu trúc quy định của Kba .

Trên máy thực, di chuyển đến thư mục **Bài3** bằng lệnh:

```
#(*F/$%.F),;(.',F<$;-M:$->)3&#.FR&6H&3 @*
```

Cấu trúc thư mục **Bài3** được như hình 1.12:



Hình 1.12: Cấu trúc thư mục **Bài3**

3) □ Trên file **lab.conf**, soạn thảo nội dung mô tả hình thái mạng theo thiết kế

```
ip address 10.1.1.1 255.255.255.0
ip address 10.1.2.1 255.255.255.0
ip address 10.1.3.1 255.255.255.0
ip address 10.1.4.1 255.255.255.0
```

4) □ Trên file **pc1.startup**, chứa nội dung được miêu tả như sau

```
ip address 10.1.1.1 255.255.255.0
ip address 10.1.2.1 255.255.255.0
```

! □ Lệnh **ip address 10.1.3.1 255.255.255.0**: thêm thông tin vạch đường mặc nhiên (default route) vào bảng vạch đường của một thiết bị.

à □ Tham số **10.1.3.1** đại diện cho Gateway là hướng mà thiết bị sẽ gửi gói tin đến để các gói tin có thể đi ra bên ngoài mạng.

à □ Thông tin vạch đường mặc nhiên sẽ được sử dụng khi thiết bị (Router, PC) không tìm thấy bất kỳ thông tin vạch đường cụ thể nào đến đích trong bảng vạch đường.

5) □ Trên file **pc2.startup** chứa nội dung được miêu tả như sau

```
ip address 10.1.1.1 255.255.255.0
ip address 10.1.2.1 255.255.255.0
```

6) □ Trên file **router.startup**, cấu hình của eth0 và eth1 được miêu tả như sau

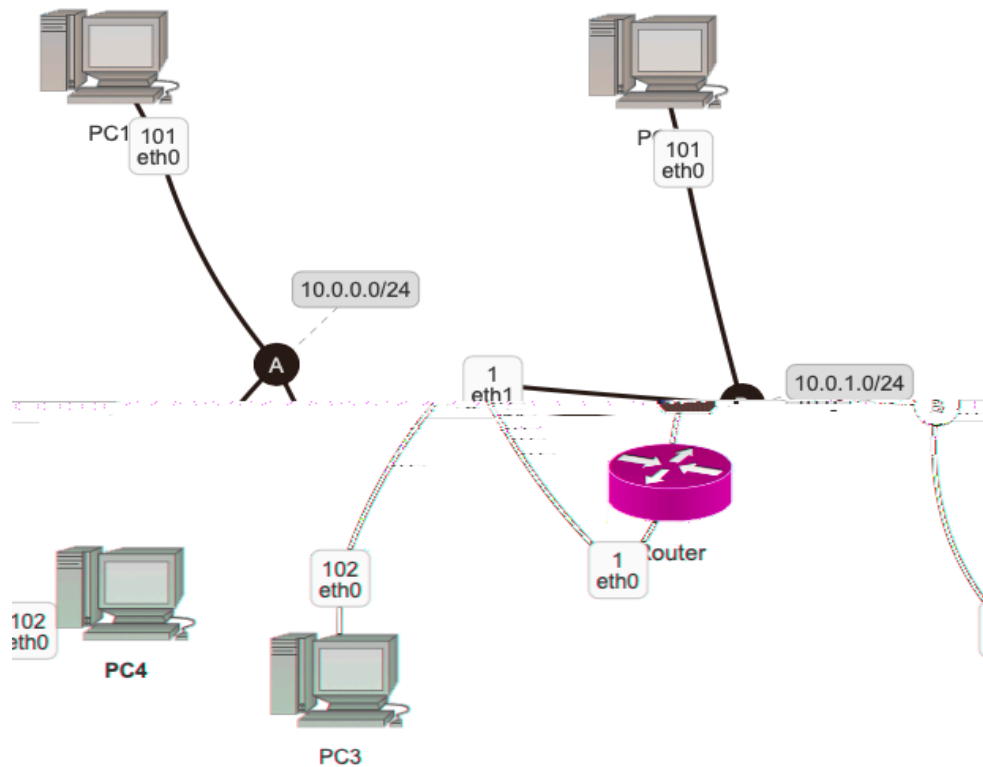
```
ip address 10.1.1.1 255.255.255.0
ip address 10.1.2.1 255.255.255.0
```

7) □ Trên máy thực, tại thư mục **Bài3** sử dụng lệnh **./start.sh** để khởi động mạng ảo **Bài3** đã tạo.

8) □ Trên pc1 lần lượt dùng các lệnh **ifconfig**, **ip netns exec** và **ping** để kiểm tra tính liên thông tới **10.1.3.1** và **10.1.4.1**.

- 9) Trên máy thực, sử dụng lệnh `ifconfig` để hủy mạng ảo `Bài3` vừa tạo. Kết thúc Bài tập 3.

#### 1.3.4! Bài tập 4



Hình 1.13Mhnh . ng s; d+ngngog Bài t Ap4

M+đư Xây dựng 2 nhánh mạng khác LAN, mỗi nhánh mạng có hai máy tính. Hai nhánh mạng khác LAN này được kết nối với nhau thông qua 1 router. Các bước thực hiện Bài t Ap 4 được trình bày chi tiết như sau:

- 7) Quan sát mô hình mạng cần xây dựng và nhận diện các thiết bị, giao diện với các địa chỉ IP được gán.
- 8) Tạo thư mục `Bài4` trong workspace của sinh viên. Thư mục sẽ này chứa các thư mục con và các file cấu hình (.75"354)?O!"#\$%&'( ) theo cấu trúc quy định của `Kha`.

Trên máy thực, di chuyển đến thư mục `Bài4` bằng lệnh

```
%=0;:&@6;754=6'5;A&43B>&3C7)"%6;2"8D")< 0
```

Gợi ýVề cơ bản, nội dung Bài t Ap4 là phân mở rộng của Bài t Ap3, vì vậy sinh viên có thể sao chép nội dung thư mục `Bài3` sang cho `Bài4` và bổ sung thêm phần khai báo, cấu hình cho máy 3#@và 3#T

- 9) Thực hiện giống như hướng dẫn trong Bài t Ap3 đã làm.