

網路模擬與分析-筆記

請記得先下載 ubuntu16.04

安裝流程:



安裝客戶機作業系統

虛擬機如同物理機，需要作業系統。您將如何安裝客戶機作業系統？

安裝來源：


☐ 安裝程序光盤(D):

無可用磁碟機

☒ 安裝程序光盤映像文件(iso)(M):

D:\國立金門大學\網路模擬與分析\ubuntu-16.04.7-de: ▾

瀏覽(R)...

 已檢測到 Ubuntu 64 位元 16.04.7。
該作業系統將使用簡易安裝。 [\(這是什麼?\)](#)

☐ 稍後安裝作業系統(S)。

创建的虚拟机将包含一个空白硬盘。

說明

< 上一步 (B)

下一步 (N) >

取消

簡易安裝資訊

這用於安裝 Ubuntu 64 位元。

個性化 Linux

全名(F):

user

用戶名(U):

user

密碼(P):

••••

確認(C):

••••

說明

< 上一步 (B)

下一步 (N) >

取消

命名虛擬機

您要為此虛擬機使用什麼名稱？

虚拟机名称(V):

mininet

位置(L):

C:\Users\chen yong xin\Documents\Virtual Machines\mininet

浏览(R)...

在“编辑”>“首选项”中可更改默认位置。

< 上一步 (B)

下一步 (N) >

取消

指定磁碟容量

磁碟大小為多少？

虛擬機的硬碟作為一個或多個檔案儲存在主機的物理磁碟中。這些檔案最初很小，隨著您向虛擬機中加入應用程式、檔案和資料而逐漸變大。

最大磁碟大小 (GB) (S) : 50

針對 Ubuntu 64 位元 的建議大小: 20 GB

☐ 將虛擬磁碟儲存為單個檔案 (O)☒ 將虛擬磁碟拆分成多個檔案 (M)

拆分磁碟後，可以更輕鬆地在電腦之間移動虛擬機，但可能會降低大容量磁碟的效能。

說明

< 上一步 (B)

下一步 (N) >

取消

已準備好建立虛擬機

按一下「完成」建立虛擬機，並開始安裝 Ubuntu 64 位元 和 VMware Tools。

將使用下列設置創建虚拟机：

| | |
|--------|---|
| 名稱： | mininet |
| 位置： | C:\Users\chen yong xin\Documents\Virtual Machines\mi... |
| 版本： | Workstation 16.2.x |
| 作業系統： | Ubuntu 64 位元 |
| 硬碟： | 50 GB, 拆分 |
| 記憶體： | 8192 MB |
| 網路介面卡： | NAT |
| 其他裝置： | 2 個 CPU 核心, CD/DVD, USB 控制器, 印表機, 音效卡 |

自定义硬件(C)...

☒ 创建后开启此虚拟机(P)

< 上一步 (B)

完成

取消

| 裝置 | 摘要 |
|-----------------|--------------------|
| 記憶體 | 8 GB |
| 處理器 | 2 |
| 新 CD/DVD (SATA) | 正在使用檔案 D:\國立金門大... |
| 網路介面卡 | NAT |
| USB 控制器 | 存在 |
| 音效卡 | 自動檢測 |
| 印表機 | 存在 |
| 顯示器 | 自動檢測 |

加入 (A) ... 移除 (R)

記憶體

指定指定給此虛擬機的記憶體量。記憶體大小必需為 4 MB 的倍數。

此虛擬機的記憶體 (M) : MB

128 GB -
64 GB -
32 GB -
16 GB -
8 GB -
4 GB -
2 GB -
1 GB -
512 MB -
256 MB -
128 MB -
64 MB -
32 MB -
16 MB -
8 MB -
4 MB -

- 最大建議記憶體
(超出此大小可能發生記憶體交換。)
20.9 GB
- 建議記憶體
4 GB
- 建議的最小客戶機作業系統記憶體
2 GB

新增虛擬機精靈



已準備好建立虛擬機

按一下「完成」建立虛擬機，並開始安裝 Ubuntu 64 位元 和 VMware Tools。

將使用下列設置創建虚拟机：

| | |
|--------|---|
| 名稱： | mininet |
| 位置： | C:\Users\chen yong xin\Documents\Virtual Machines\mi... |
| 版本： | Workstation 16.2.x |
| 作業系統： | Ubuntu 64 位元 |
| 硬碟： | 50 GB, 拆分 |
| 記憶體： | 8192 MB |
| 網路介面卡： | NAT |
| 其他裝置： | 2 個 CPU 核心, CD/DVD, USB 控制器, 印表機, 音效卡 |

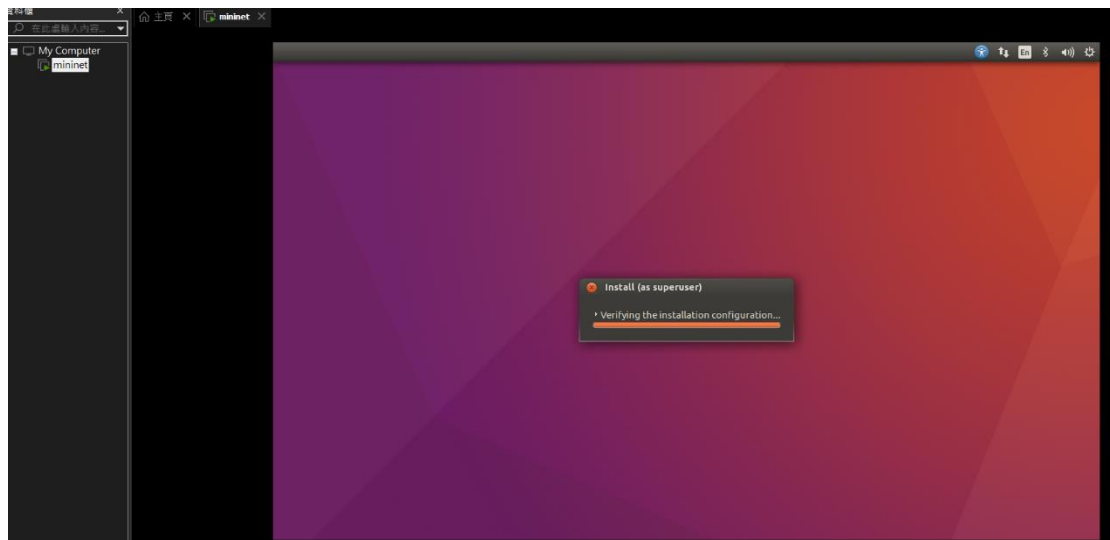
自定义硬件(C)...

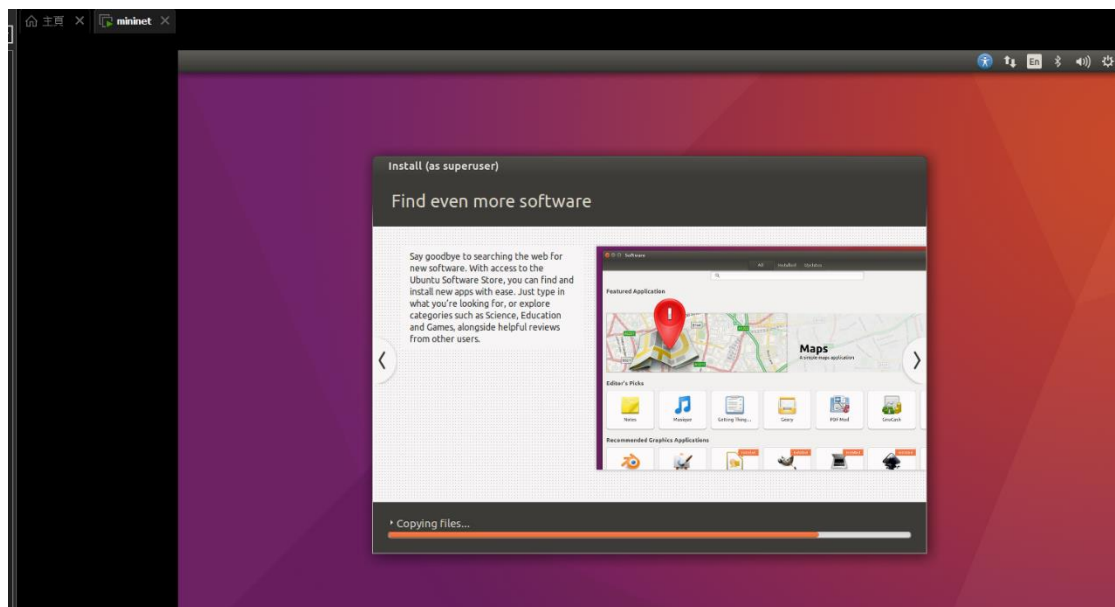
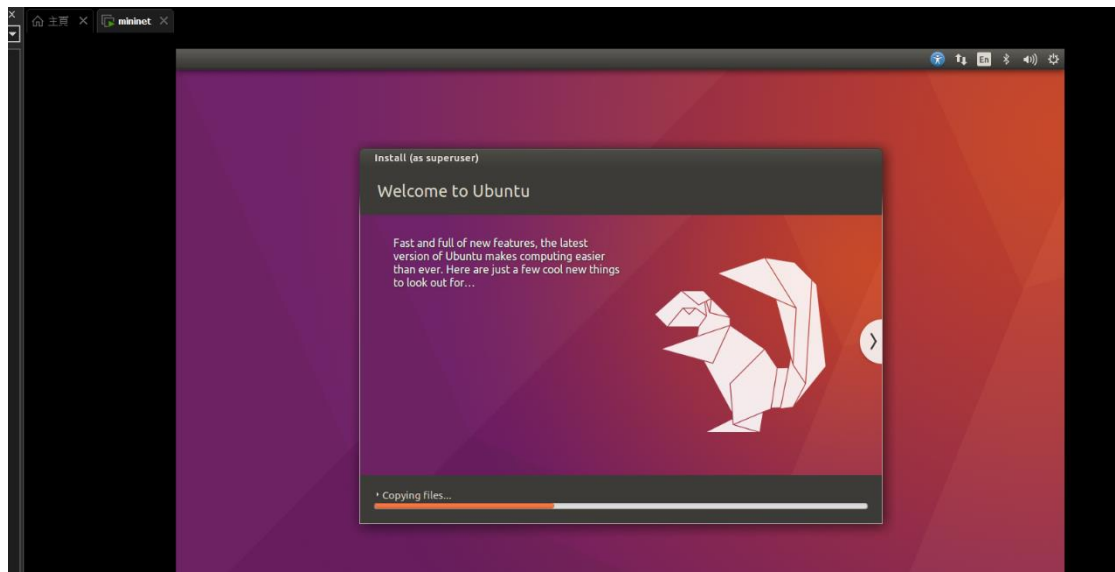
☒ 创建后开启此虚拟机(P)

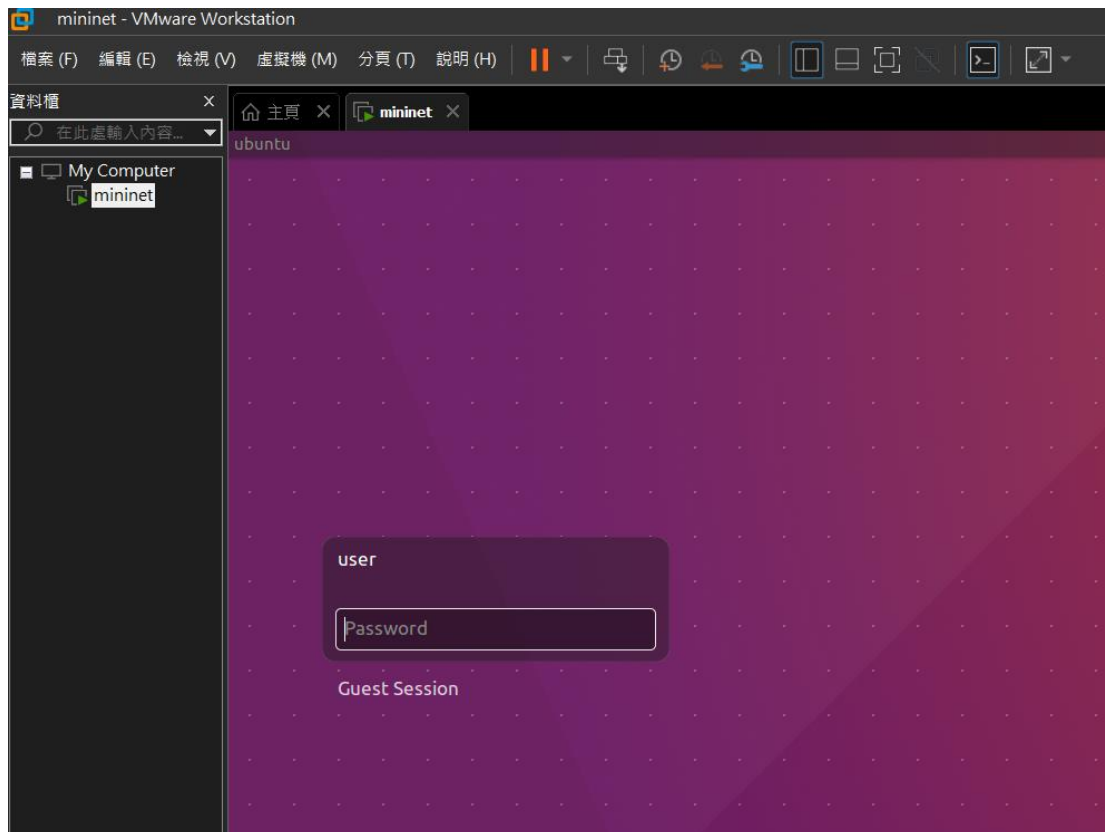
< 上一步 (B)

完成

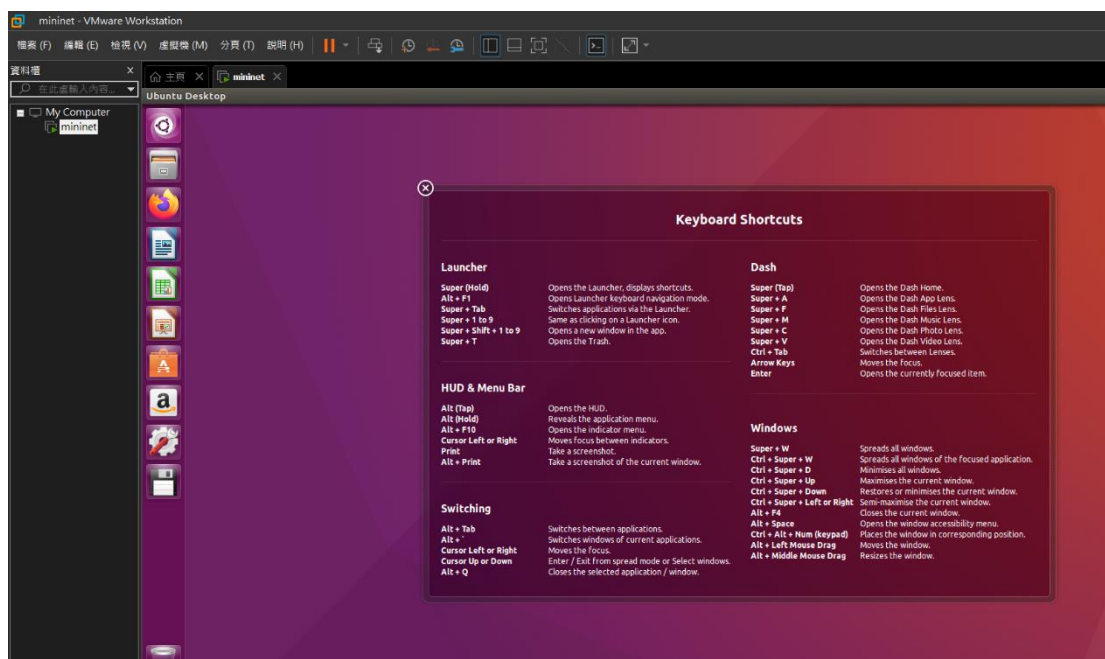
取消





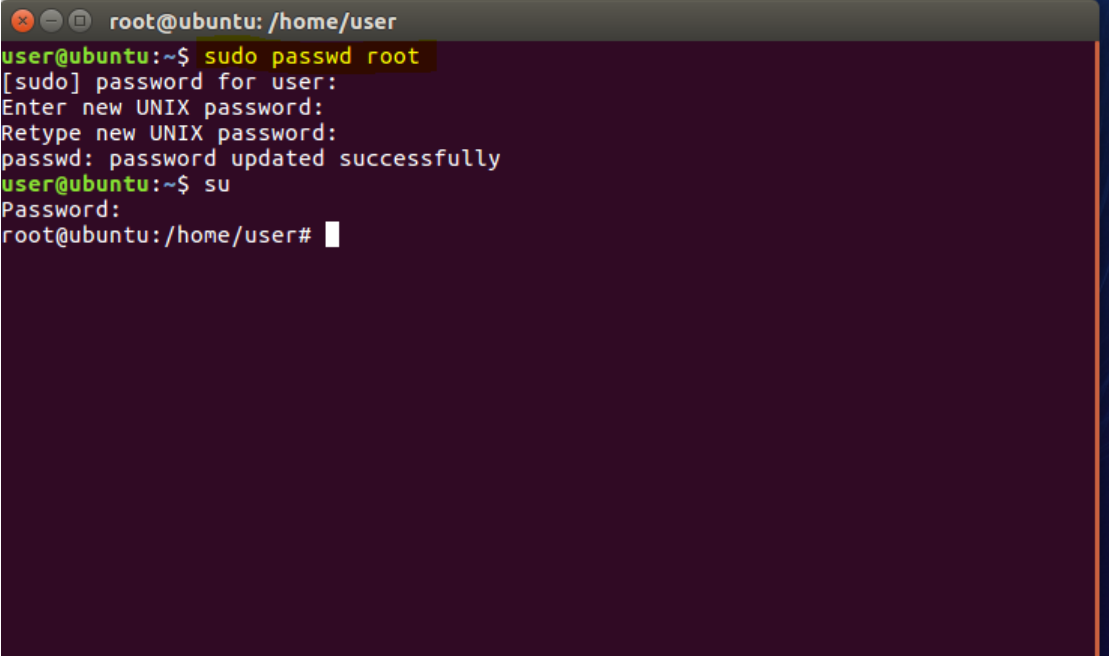


輸入登入密碼



完成安裝登入首頁

1. 設定超級使用者密碼

A terminal window with a dark purple background. The title bar shows 'root@ubuntu: /home/user'. The prompt is 'user@ubuntu:~\$'. The user enters 'sudo passwd root'. The prompt changes to '[sudo] password for user:'. The user enters a password (not visible). The prompt changes to 'Enter new UNIX password:'. The user enters a new password (not visible). The prompt changes to 'Retype new UNIX password:'. The user enters the same password (not visible). The prompt changes to 'passwd: password updated successfully'. The user enters 'su'. The prompt changes to 'Password:'. The user enters the password (not visible). The prompt changes to 'root@ubuntu:/home/user#'.

```
root@ubuntu: /home/user
user@ubuntu:~$ sudo passwd root
[sudo] password for user:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
user@ubuntu:~$ su
Password:
root@ubuntu:/home/user#
```

2.

補充: 集縮比 (英語 : Concentration Rate)

是 ISP 是用戶端電路總頻寬至 ISP 端頻寬比例。

業者均會告知用戶下載及上傳頻寬，但該速率僅為該用戶端線路最高可能達到的瞬間速率。

實際頻寬則受限於集縮比。

集縮比也是 ISP 創造利潤的基礎，集縮比越大，越能降低 ISP 分攤至各用戶頻寬的線路成本。

補充:Linux 下容器的隔離技術

1. 命名空間(Namespace)

2. docker 使用六種不同的 Namespace，來實現資源隔離器。

3. 1. UTS namespace(Unix Time-sharing)：每台不同的主機

都會有不同的主機名稱，UTS namespace 提供了主機名與

域名的隔離，這樣每個 docker 容器就可以擁有獨立的主機

名和域名了，在網路上可以被視作一個獨立的節點，而不會

與宿主機的主機名稱互相干擾。

4. 2. IPC namespace(Inter-Process Communication)：

是 Unix/Linux 下行程之間通信的方式，例如 IPC 有共享記憶

體、訊號量、消息隊列等方法。

3. PID namespace(Process ID)：

不管在 Windows 或是 Linux 上，在同一個主機上，

上面跑的 Process 的 ID 是不可能一樣的，

例如：主機 A 上面跑了兩個行程 tomcat(PID 10000)、

mongodb(PID 10001)，這邊的 tomcat PID 與 mongodb

PID 在同一台主機下永遠是不可能一樣的

是如果在另外一台 B 主機下，PID 就可能會重複。那我們有

沒有辦法既在同一台主機下，可以模擬成多台主機下，彼此可以擁有相同的 PID 又不互相干擾？

使用劃分不同的 PID namespace，每個 PID namespace，裡面的 process 可以和另外一個 PID namespace，擁有重複的 PID 名稱，彼此不互相干擾，就像在多台主機上。

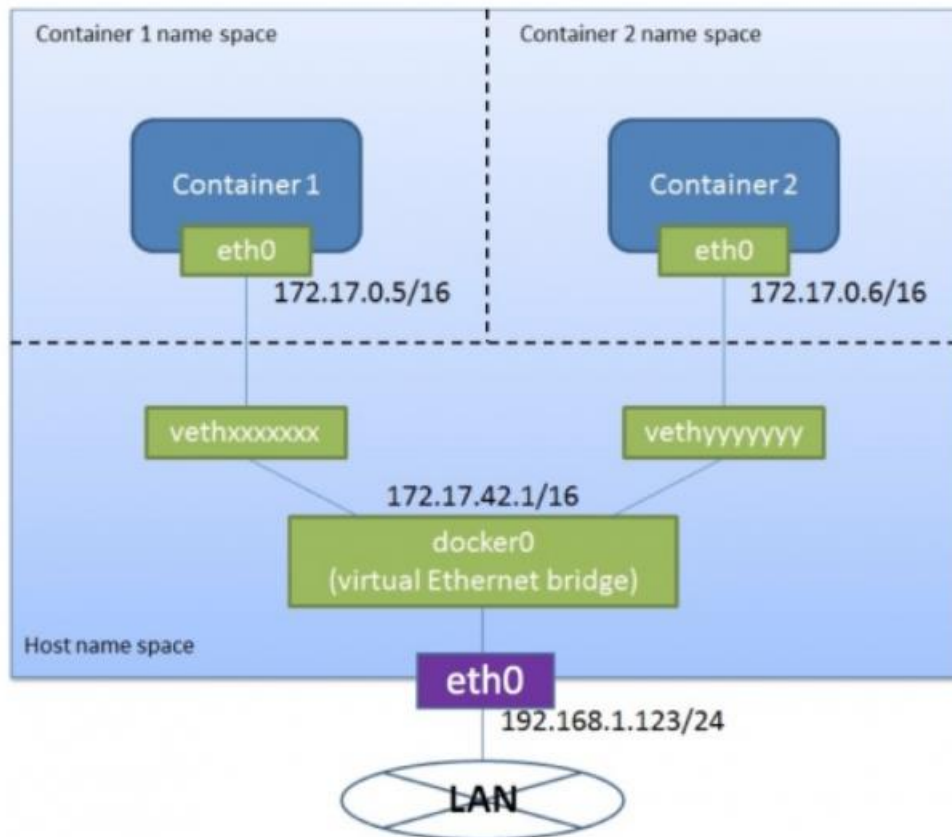
4. mount namespace：

就是用來隔離檔案系統的掛載點，使得不同的 mount namespace 擁有自己的獨立掛載點訊息，宿主機與容器內互相使用不同的 mount namespace 而不會互相干擾。

5. network namespace：主要提供了關於網路資源的隔離，包含網路設備、IPv4、IPv6 協定、IP 路由表、防火牆等資源的隔離，否則如果你在容器內執行一個 apache 的行程，宿主機上也執行 apache，你很可能會得到 80 端口已經被佔用的錯誤。就是靠這個 network namespace 才有辦法使得我們在容器的世界裡，宿主機與各個不同容器內，都使用不同的 network namespace，彼此才可以使用相同的行程相同的端口不互相被干擾。

6. 6. user namespace：主要隔離了 user 權限相關的 Linux 資

源，包括 user IDs and group IDs，keys，和 capabilities



```
#!/usr/bin/python
```

```
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.link import Link, TCLink, Intf

if '__main__' == __name__:
    net=Mininet(link=TCLink)
    h1=net.addHost('h1')
    h2=net.addHost('h2')
    Link(h1,h2)
    net.build()
    CLI(net)
    net.stop()
```

```
root@ubuntu:/home/user/2022mininet# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.026 ms
^C
--- 10.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.024/0.025/0.026/0.001 ms
root@ubuntu:/home/user/2022mininet#
```

```
root@ubuntu:/home/user/2022mininet# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.019 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.023 ms
^C
--- 10.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.019/0.021/0.023/0.002 ms
root@ubuntu:/home/user/2022mininet#
```

補充

smallko 柯老師

回覆

```
h1:~$ ip netns exec h1-eth0 192.168.1.2 dev h1-eth0
192.168.1.0/24 dev h1-eth0 proto kernel scope link src 192.168.1.1
root@ubuntu:/home/user/2022mininet#

#!/usr/bin/python

from mininet.cli import CLI
from mininet.net import Mininet
from mininet.link import Link, TCLink, Intf

if __name__ == '__main__':
    net = Mininet(link=TCLink)
    h1 = net.addHost('h1')
    h2 = net.addHost('h2')
    h3 = net.addHost('h3')
    Link(h1, h2)
    Link(h2, h3)
    net.build()
    h1.cmd("ifconfig h1-eth0 0")
    h1.cmd("ip addr add 192.168.1.1/24 brd + dev h1-eth0")
    h2.cmd("ifconfig h2-eth0 0")
    h2.cmd("ip addr add 192.168.1.2/24 brd + dev h2-eth0")
    h2.cmd("ifconfig h2-eth1 0")
    h2.cmd("ip addr add 192.168.2.2/24 brd + dev h2-eth1")
    h3.cmd("ifconfig h3-eth0 0")
    h3.cmd("ip addr add 192.168.2.1/24 brd + dev h3-eth0")
    h1.cmd("ip route add default via 192.168.1.2")
    h3.cmd("ip route add default via 192.168.2.2")
    h2.cmd("echo 1 > /proc/sys/net/ipv4/ip_forward")
    CLI(net)
    net.stop()
```



```
root@ubuntu:/home/ubuntu/2022mininet
1.py 2.py 3.py
root@ubuntu:/home/ubuntu/2022mininet# ./3.py
mininet> net
h1 h1-eth0:h2-eth0
h2 h2-eth0:h1-eth0 h2-eth1:h3-eth0
h3 h3-eth0:h2-eth1
mininet> h1 ping -c 5 h3
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data:
64 bytes from 192.168.2.1: icmp_seq=1 ttl=63 time=0.049 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=63 time=0.029 ms
64 bytes from 192.168.2.1: icmp_seq=3 ttl=63 time=0.029 ms
64 bytes from 192.168.2.1: icmp_seq=4 ttl=63 time=0.030 ms
64 bytes from 192.168.2.1: icmp_seq=5 ttl=63 time=0.100 ms

--- 192.168.2.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4098ms
rtt min/avg/max/mdev = 0.029/0.047/0.100/0.028 ms
mininet>
(gedit:3494): dconf-WARNING **: failed to commit changes to dconf: The connection is closed

** (gedit:3494): WARNING **: Set document metadata failed: Setting attribute metadata:gedit-position not supported
```

