Lab2. NAT、port-forwarding

TA 陳芷桓 (czh)、周廷威 (twchou)
Credit to 施羿廷 (ytshih)

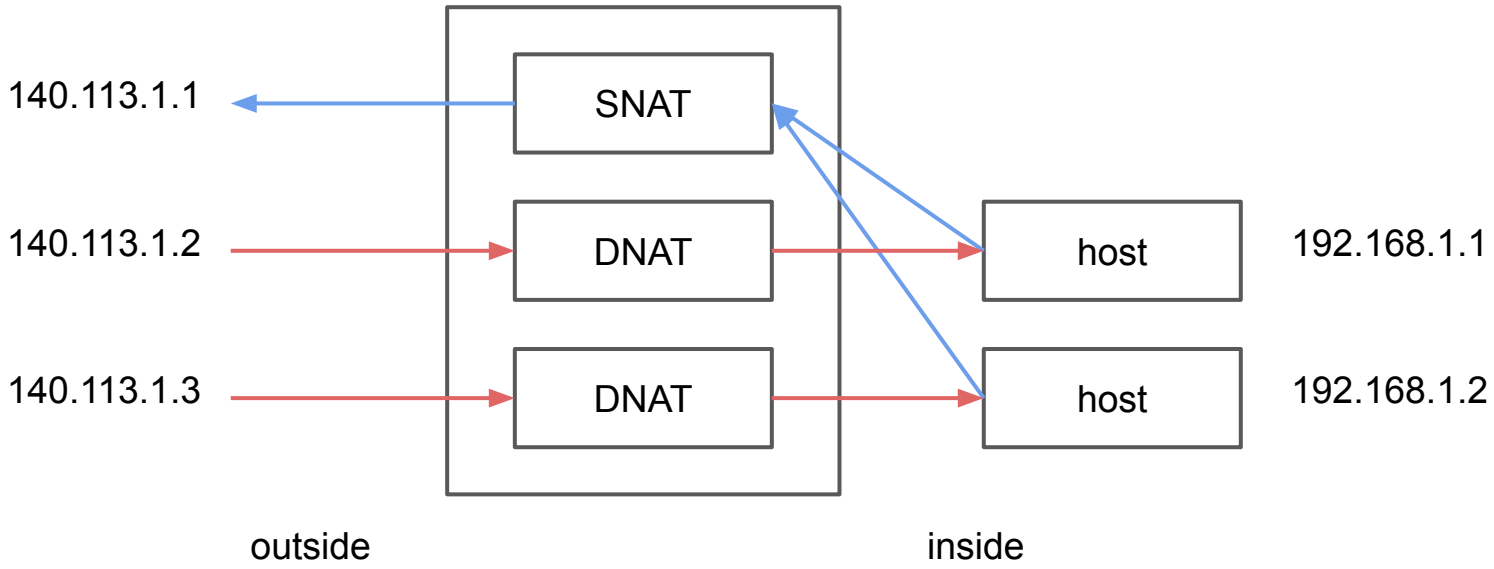
This persenration is for your refernce only, you still need to complete the Project 1 topology.

Purpose

- Basic knowledge of VM (Virtual Box)
  - Virtaul Box hardware setting
  - Virtual Box network interface modes
- Port forwarding concept
- SSH tunnel

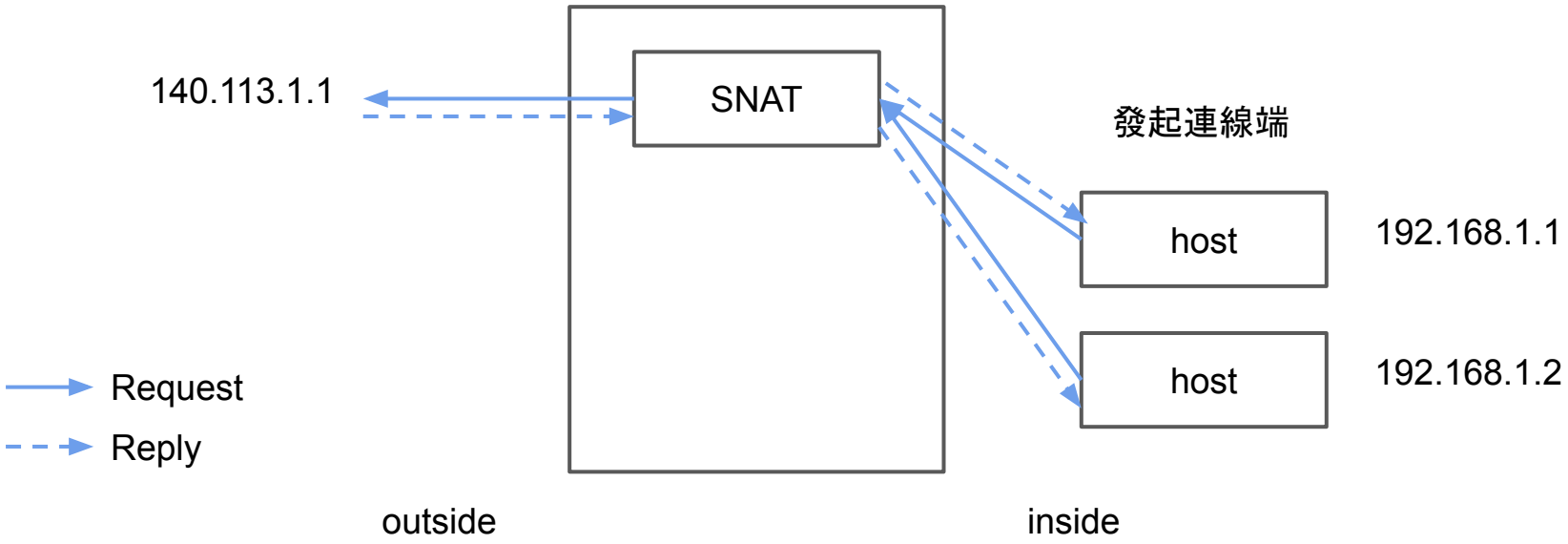
NAT(Network Address Translation) Use Cases (1/3)

- Perform SNAT when the inside packets pass through firewall or router
- Perform DNAT when the outside packets pass through firewall or router



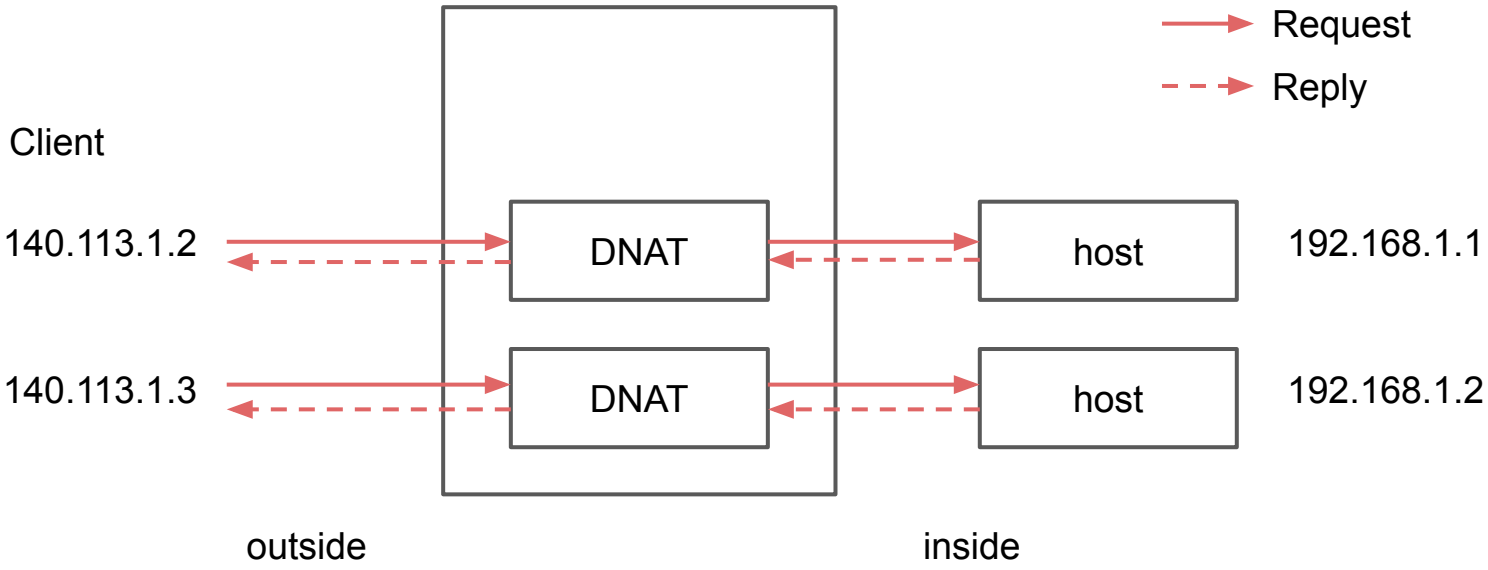
NAT Use Cases (2/3)

- Multiple private IPs treat the same public IP as an entrance

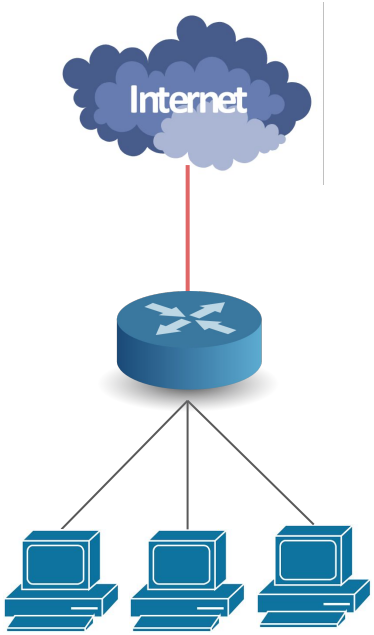


NAT Use Cases (3/3)

- Outside network try to access inside network



NAT Common Topology



NAT Server

Hosts

Translate source address

Outgoing traffic

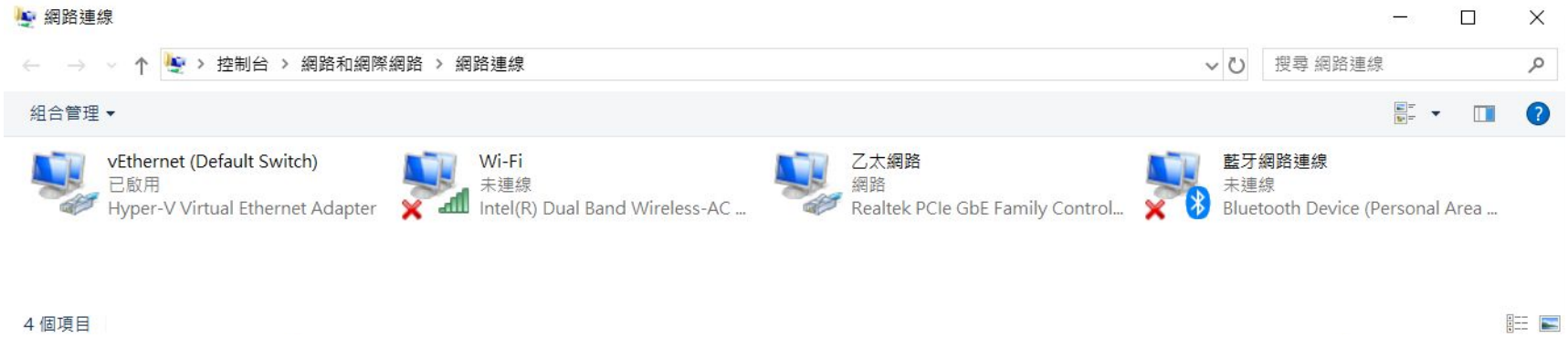
What is Virtual Machine (VM)?

- A software computer that, like a physical computer, runs an operating system and applications.
- Has virtual devices that provide the same functionality as physical hardware
- Additional benefits
  - Portability
  - Manageability
  - Security

https://www.vmware.com/support/pubs/

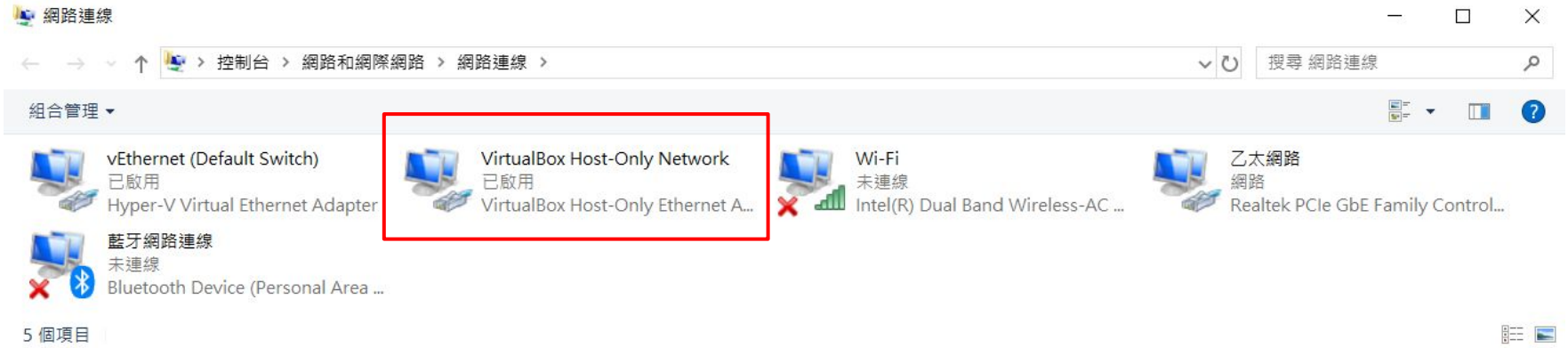
Virtual Networking

- Before install VirtualBox...



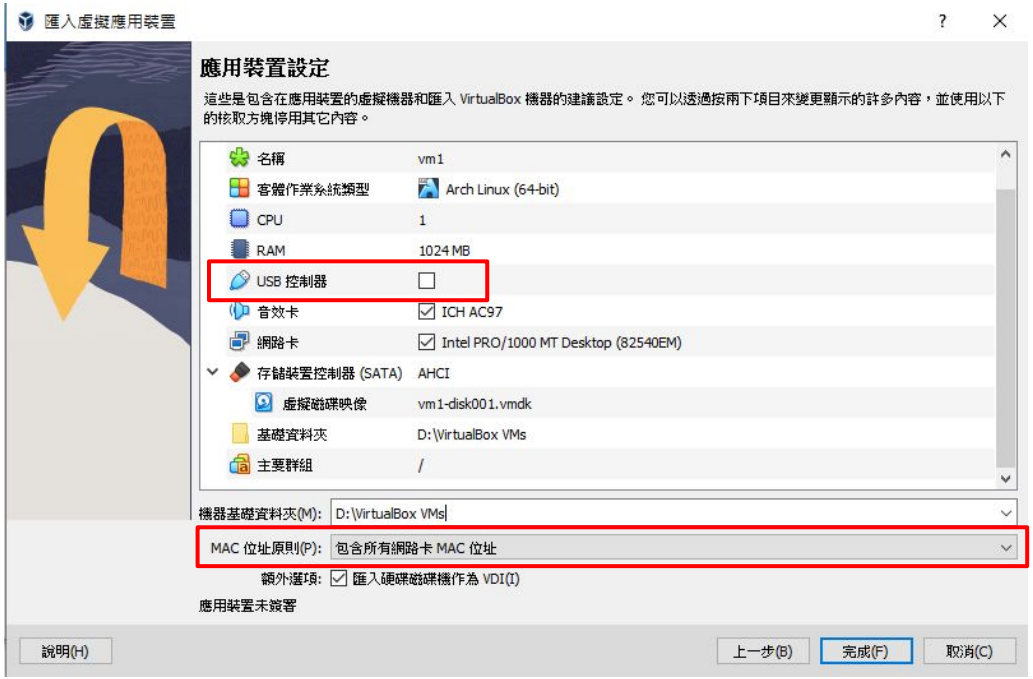
Virtual Networking

- After install VirtualBox…
  - What is the meaning of “Host-only”?



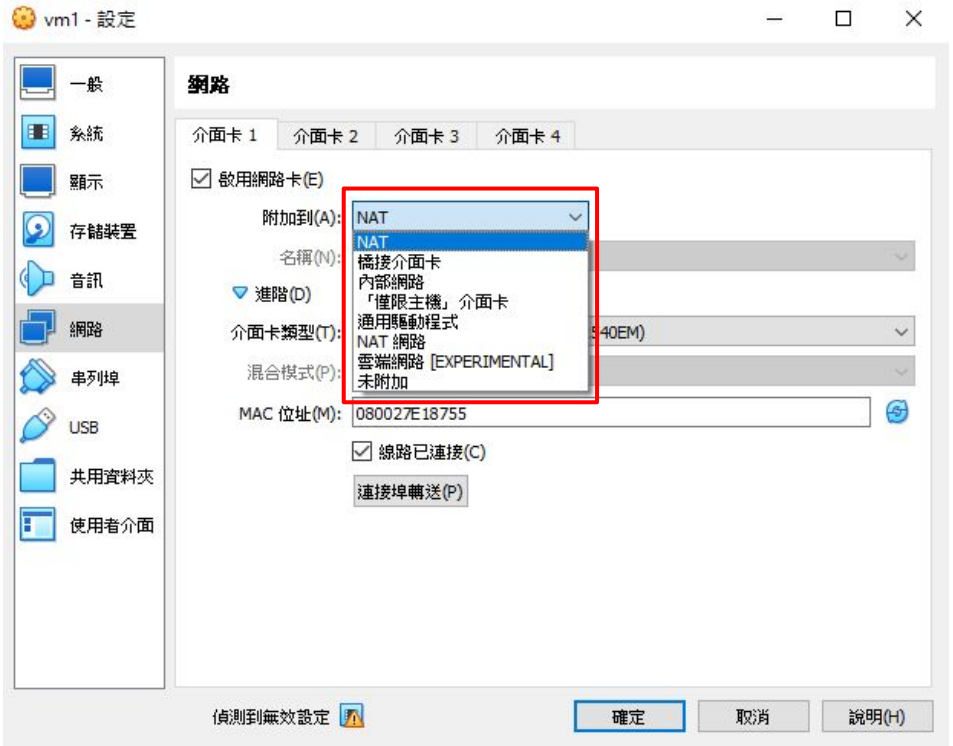
Virtualbox Import settings

- For **MAC 位址原則**, choose「**包含所有網路卡 MAC 位址**」.
- **Uncheck「USB 控制器」.**



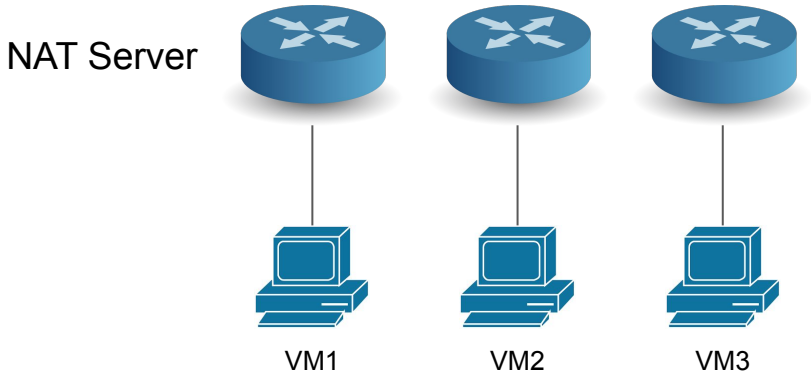
Virtual machine settings

- What is the meaning of these options?

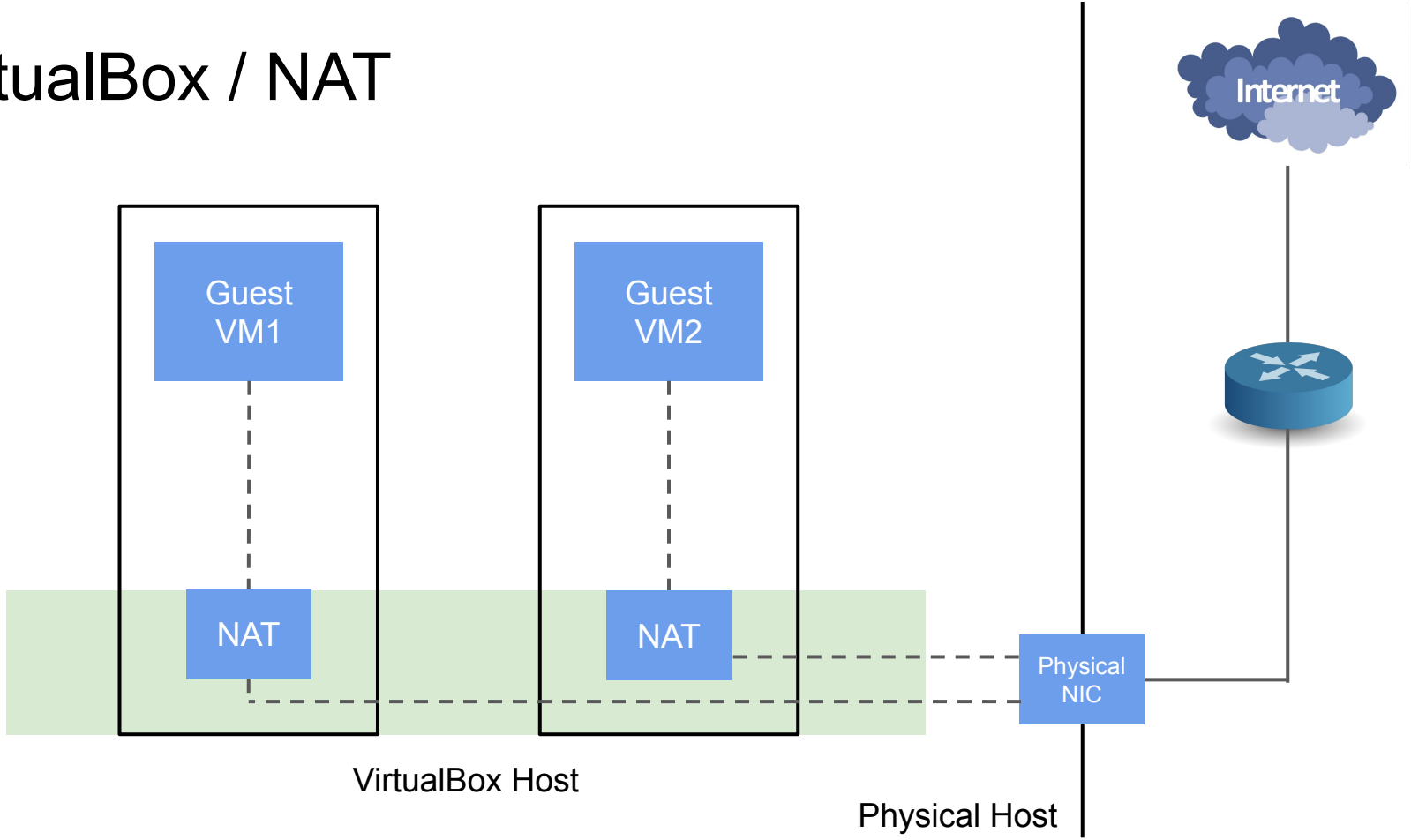


VirtualBox / NAT

- Default mode
- A virtual machine with NAT enabled acts much like a real computer that connects to the Internet through a router
- This router is placed between **each** virtual machine and the host
- This separation maximizes security since by default virtual machines **cannot talk to each other**



VirtualBox / NAT



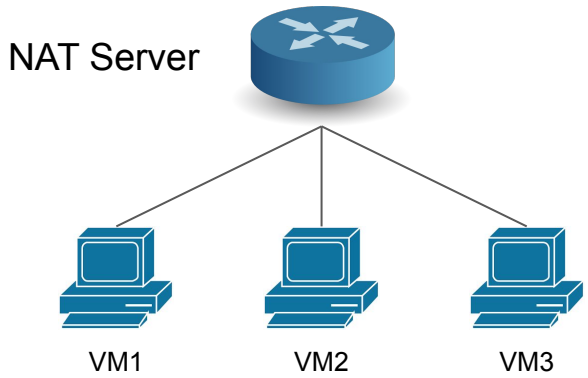
VirtualBox / NAT

- Check the IP addresses of the vms.
  - So we can easily tell that they **can’t ping each other** directly.
- Ping 1.1.1.1
  - Every vm should be able to reach the Internet.
- Find the default gateway (*ip --color route*), which is **host** in this case.
- Ping the default gateway.
  - It should work if your host OS would reply **ICMP**.

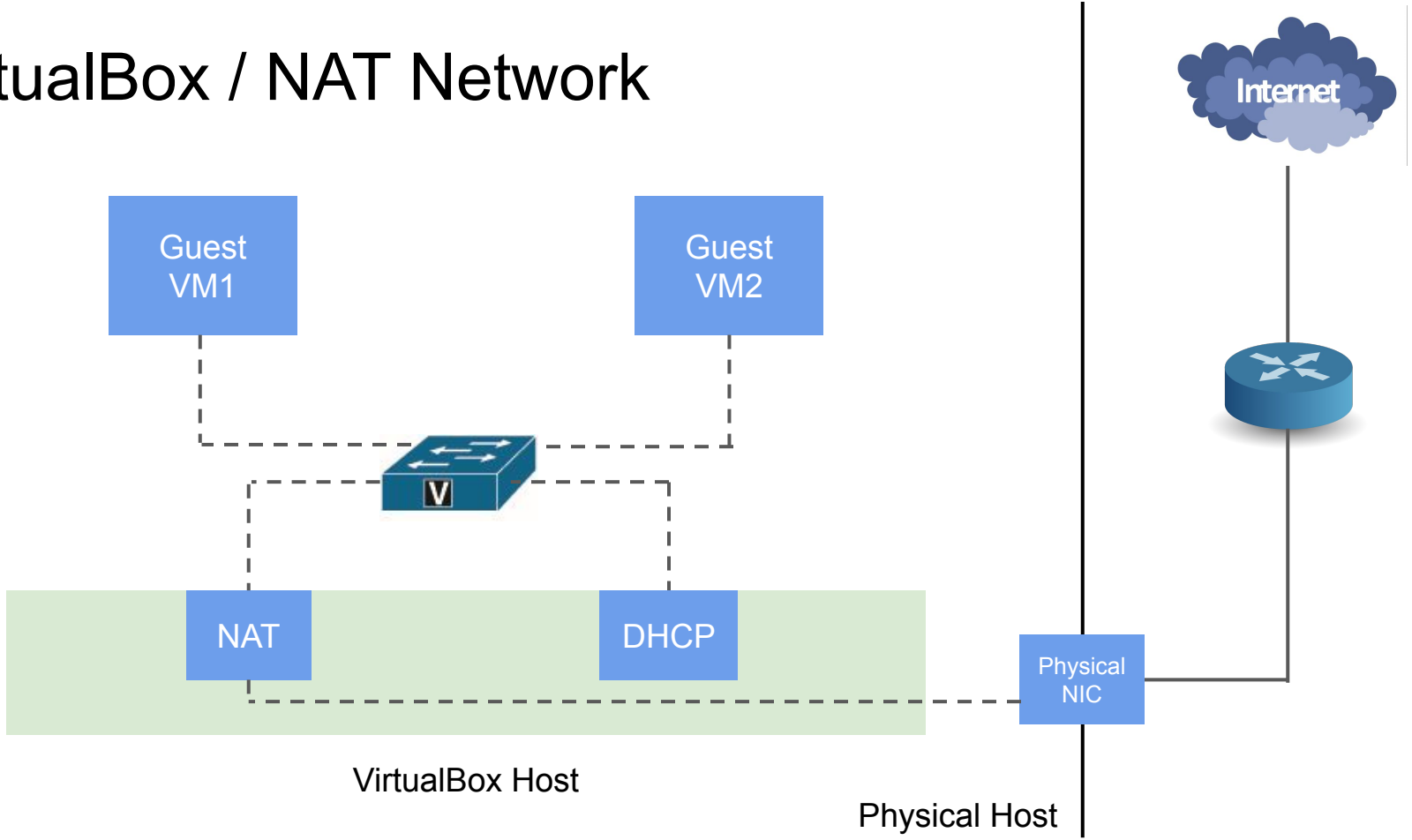
user/password: ccna/ccna

VirtualBox / NAT Network

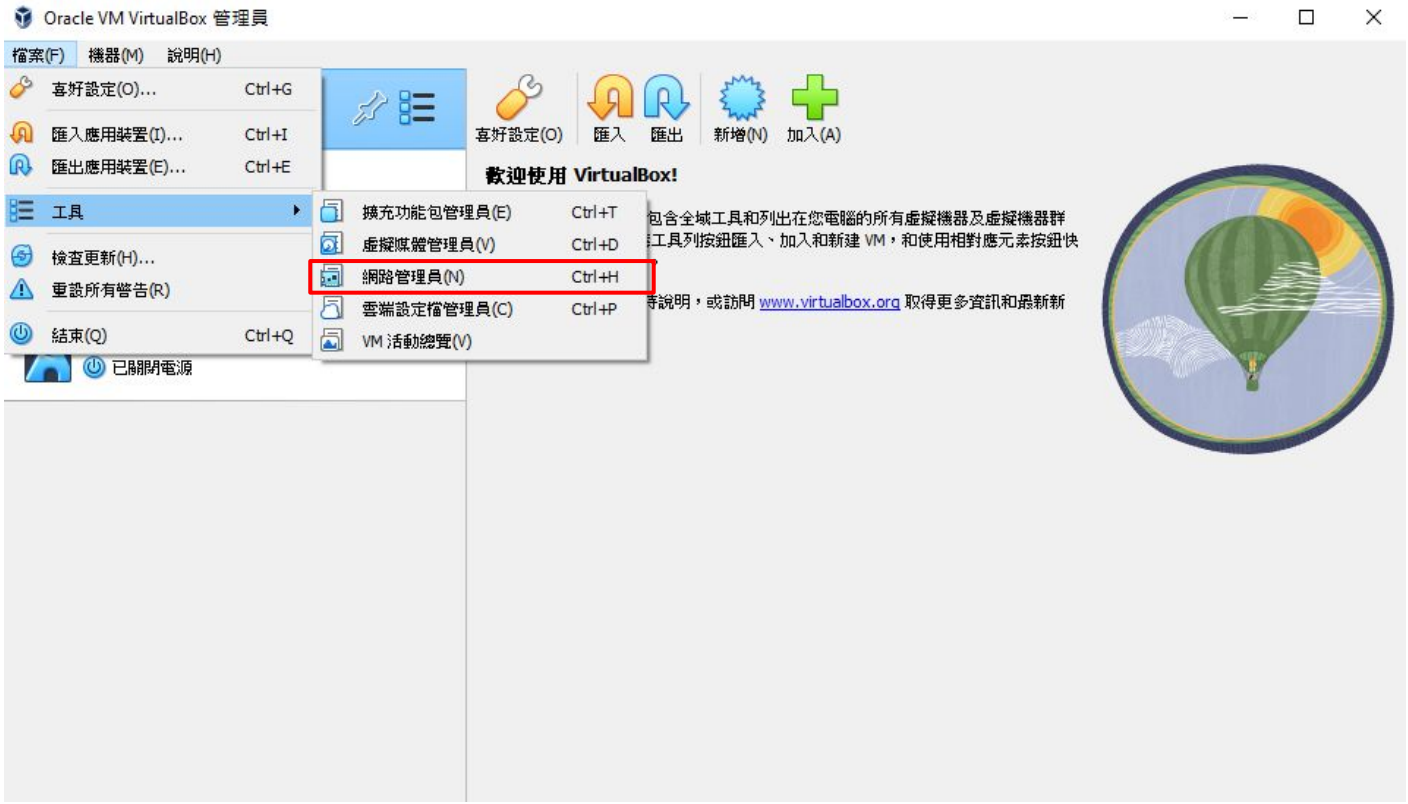
- Works in a similar way to a home router
- Not like previous NAT mode, NAT Network letting systems inside **communicate with each other** and with systems outside using TCP and UDP over IPv4 and IPv6



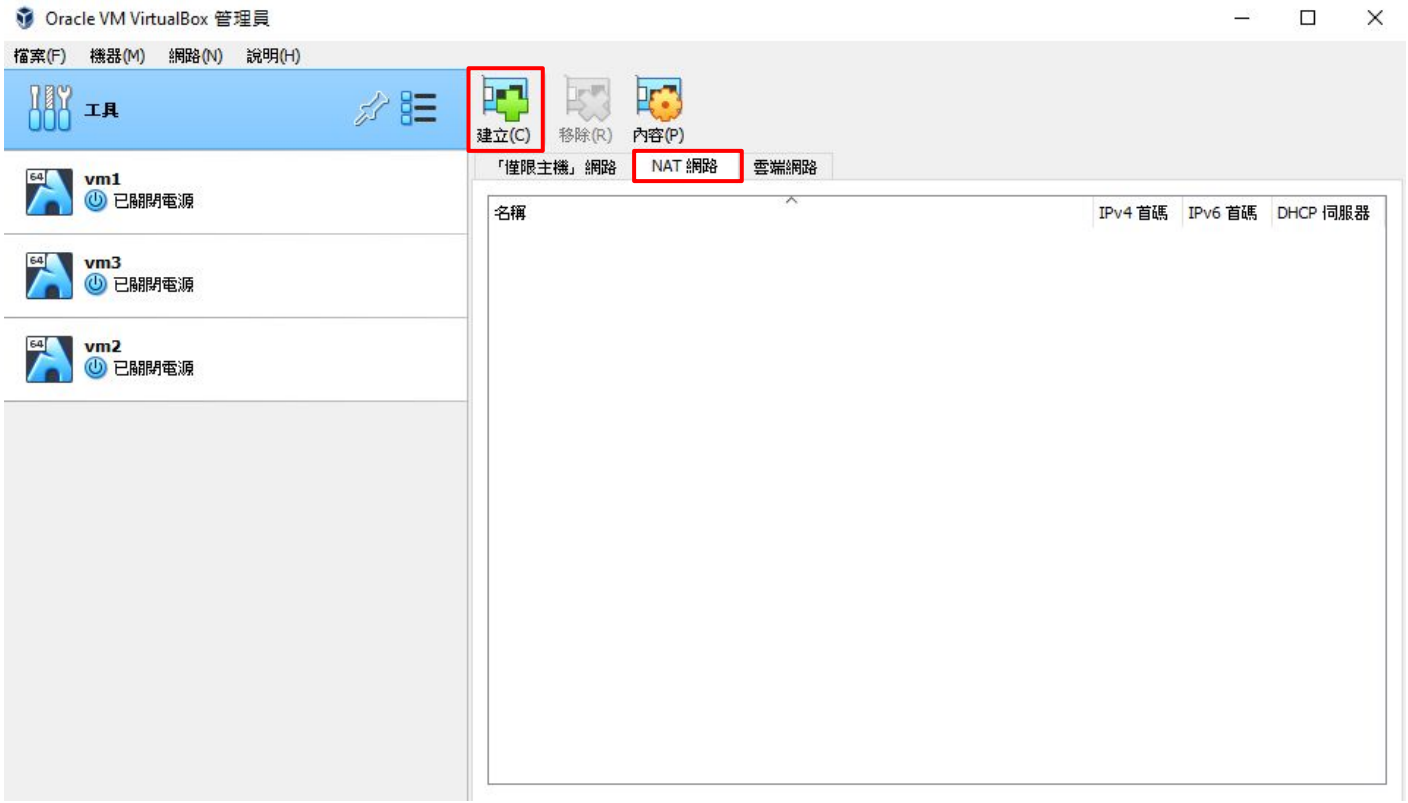
VirtualBox / NAT Network



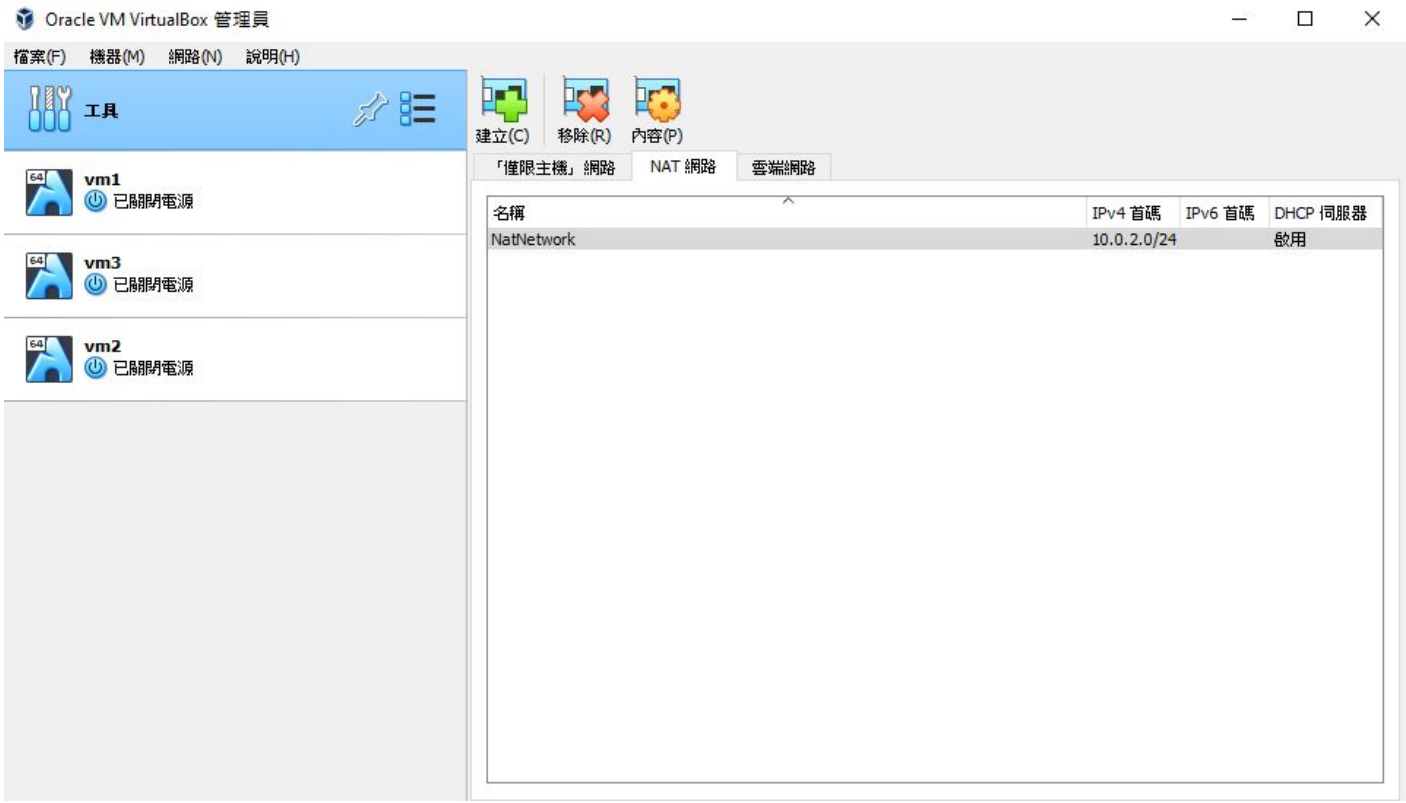
VirtualBox / NAT Network



VirtualBox / NAT Network (Service)



VirtualBox / NAT Network (Service)

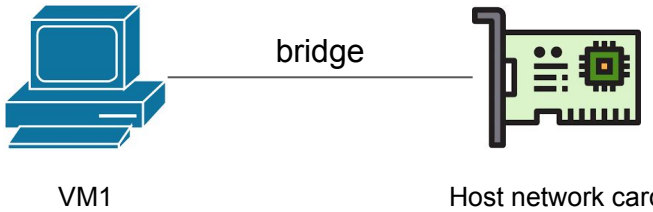


VirtualBox / NAT Network (Service)

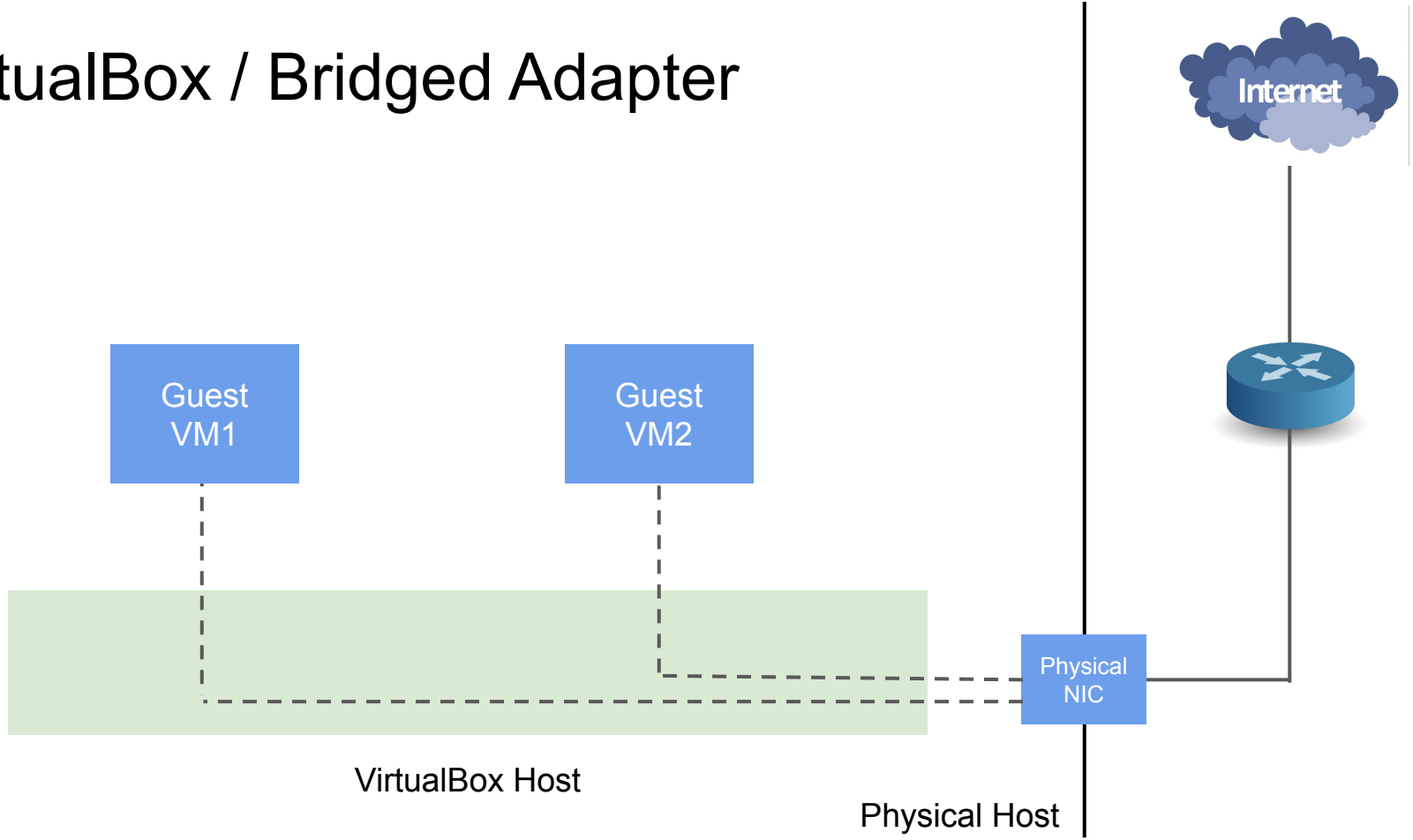
- Check their IP address.
  - The IP address should be **different**, and in the **same subnet**.
- Ping 1.1.1.1
  - Every vm should be able to reach the Internet.
- Ping each other
- Find the default gateway (*ip --color route*), which is host in this case.
- Ping the default gateway.
  - It should work if your host OS would reply ICMP.

VirtualBox / Bridged Adapter

- Connect to one of your installed network cards and exchanges network packets **directly**
- Uses a device driver on your host system that filters data from your physical network adapter
  - called a *net filter* driver

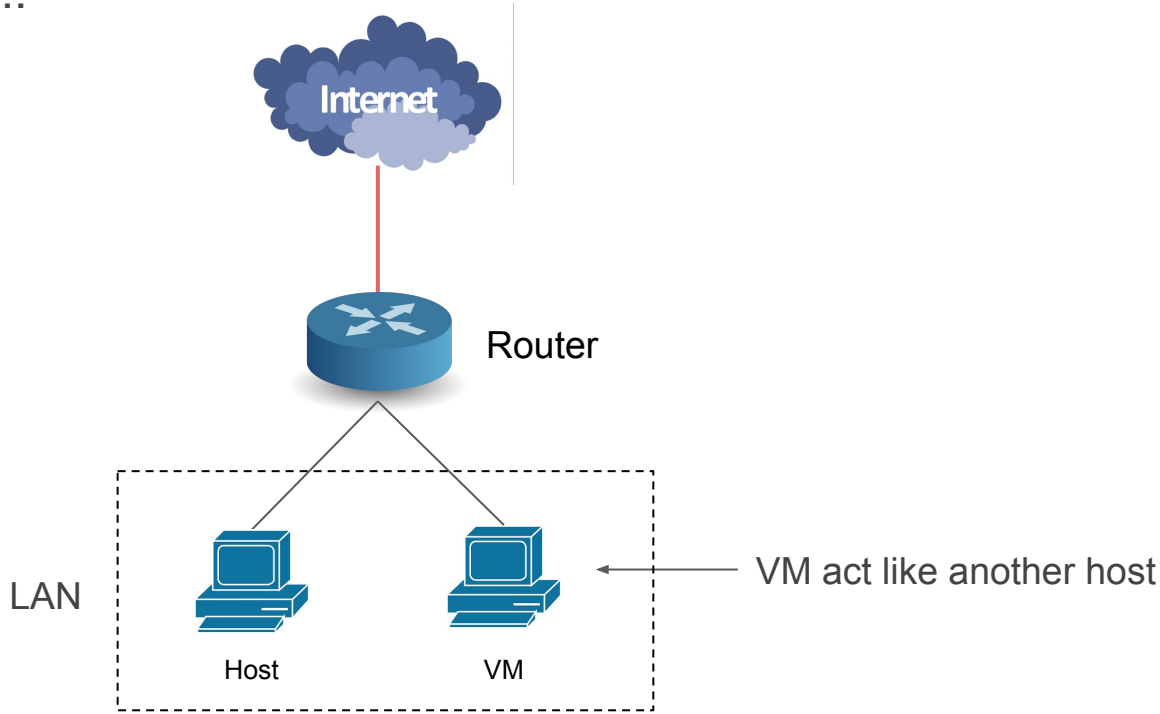


VirtualBox / Bridged Adapter



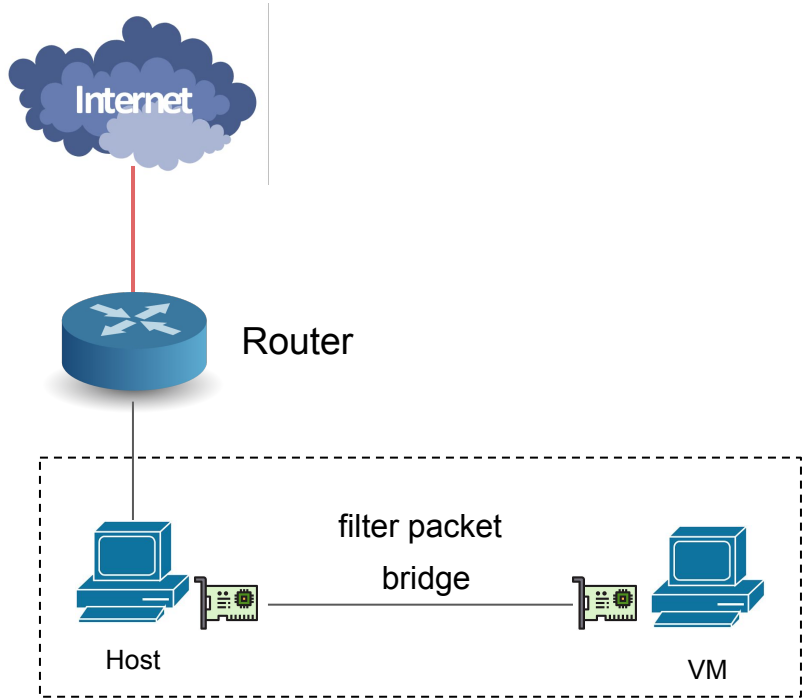
VirtualBox / Bridged Adapter Use Case

- What you try to do...



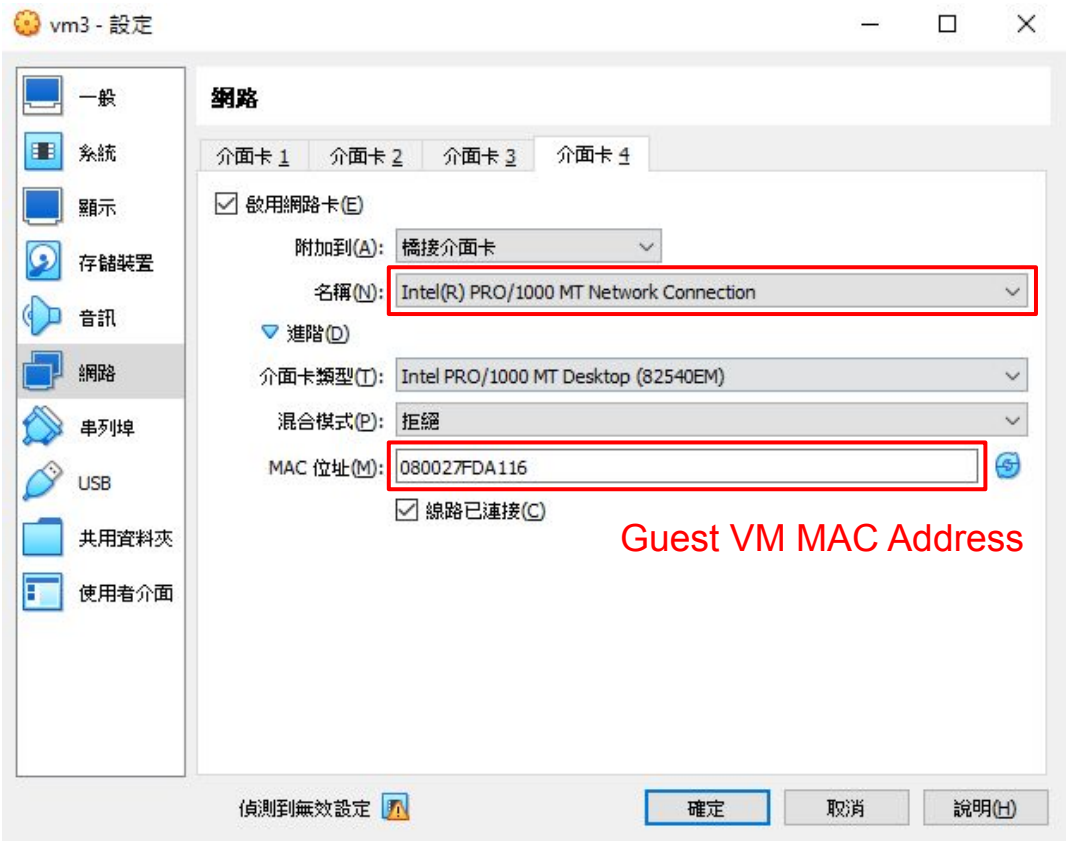
VirtualBox / Bridged Adapter Use Case

- What VirtualBox actually does...



In same machine

VirtualBox / Bridged Adapter Setting



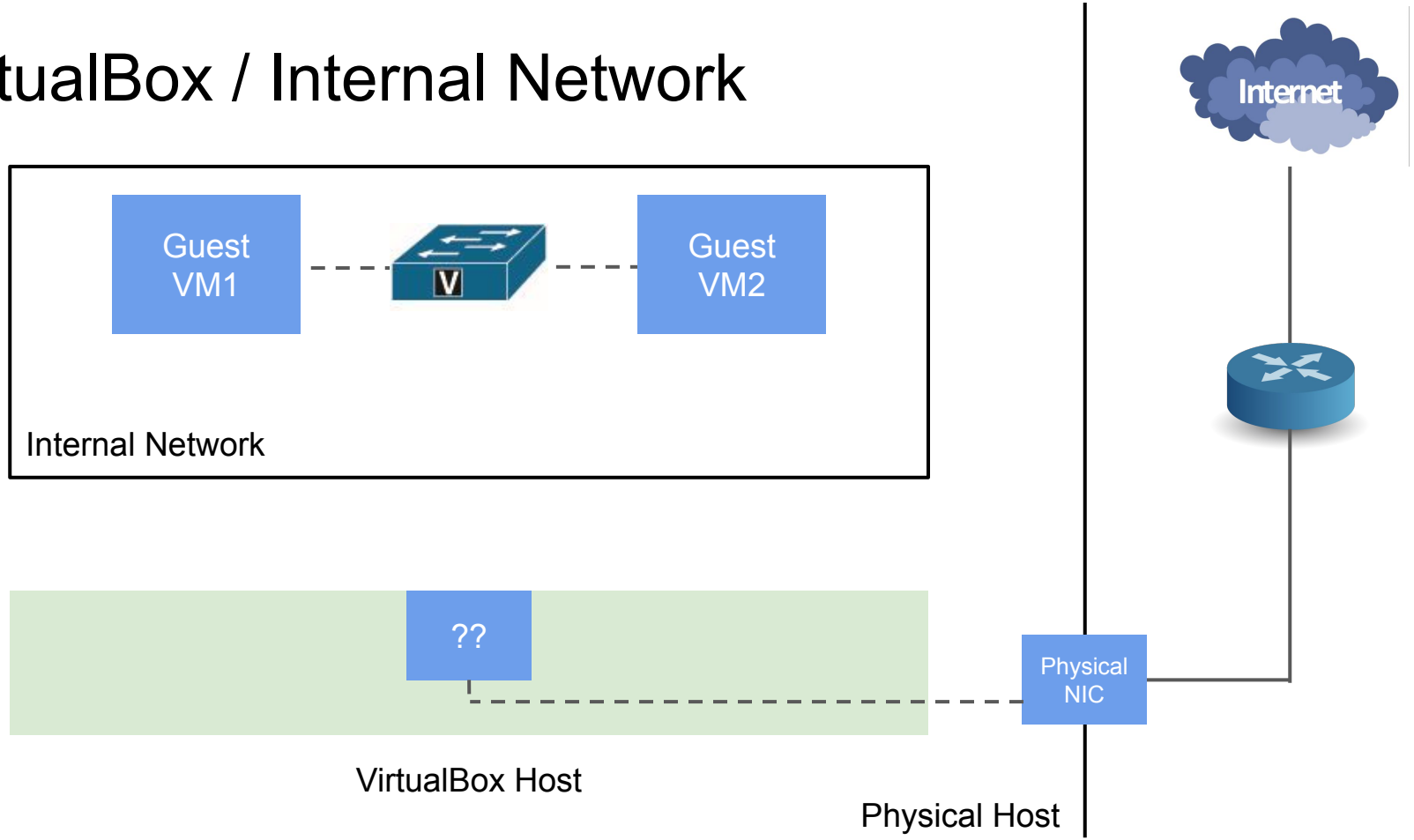
VirtualBox / Bridged Adapter

- Check their IP address.
  - **In our PCroom,** there should be **no IP address** since the DHCP rule is different.
  - **In your own PC**, the IP address should be **different**, and in the same subnet **as host**.
- Ping 1.1.1.1
  - Every vm should **not** be able to reach the Internet in our PCroom.
- Find the default gateway in Windows (`ipconfig`), which is **the same as host**.
- Ping the default gateway in vm.
  - This should **not** work unless your set the static IP.

VirtualBox / Internal Network

- Guest VM connected to an **isolated** virtual network
- VMs connected to same internal network can communicate with each other
- VMs cannot communicate with VirtualBox host
- VMs cannot communicate with any other hosts in external networks
- Internal network can be used for **modelling real networks**

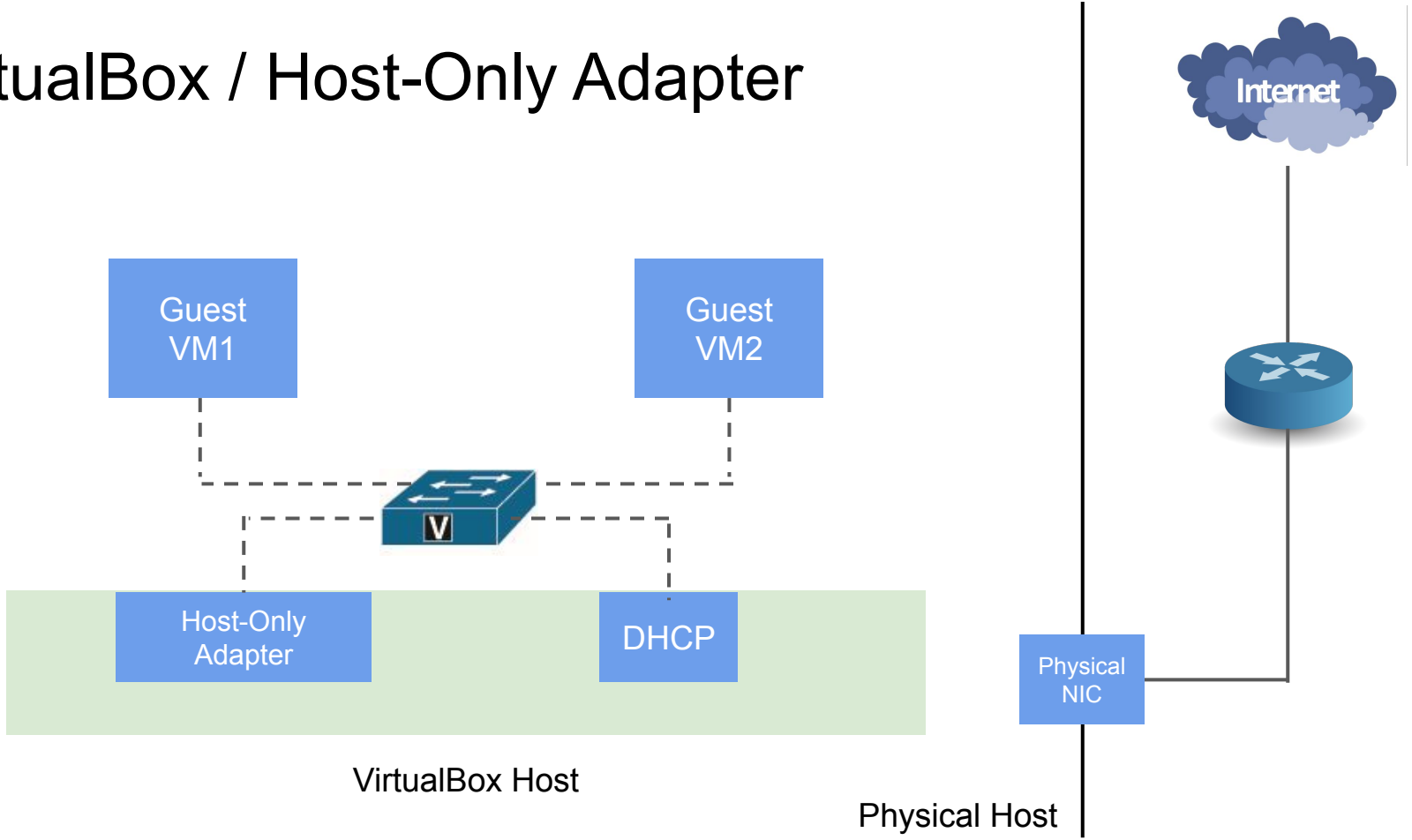
VirtualBox / Internal Network



VirtualBox / Internal Network

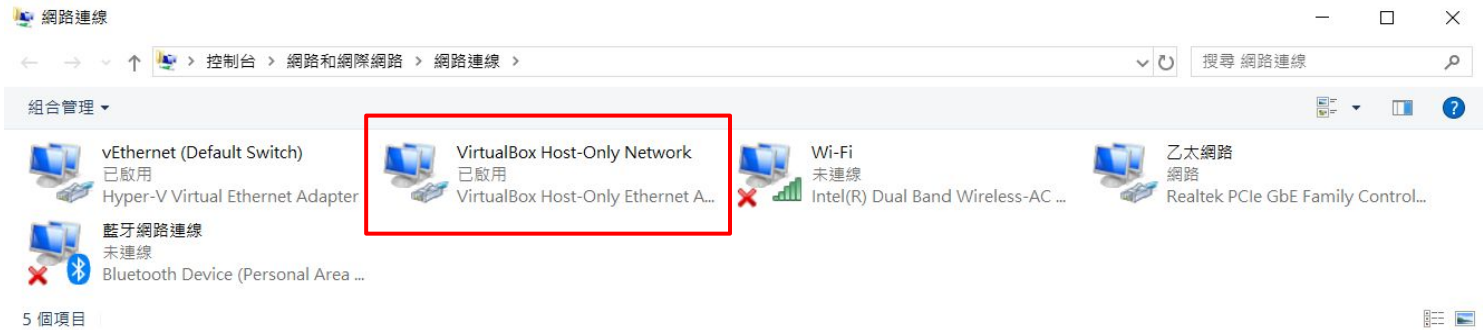
- Check their IP address.
  - There are **no IP addresses** assigned to the vms (no DHCP).
- Ping 1.1.1.1
  - Vms should **not** be able to access the internet.
- Ping each other.
  - Vms should **not** be able to ping each other since they have **no IP address**.
- Find the default gateway (*ip --color route*)
  - There is **no** default gateway.

VirtualBox / Host-Only Adapter

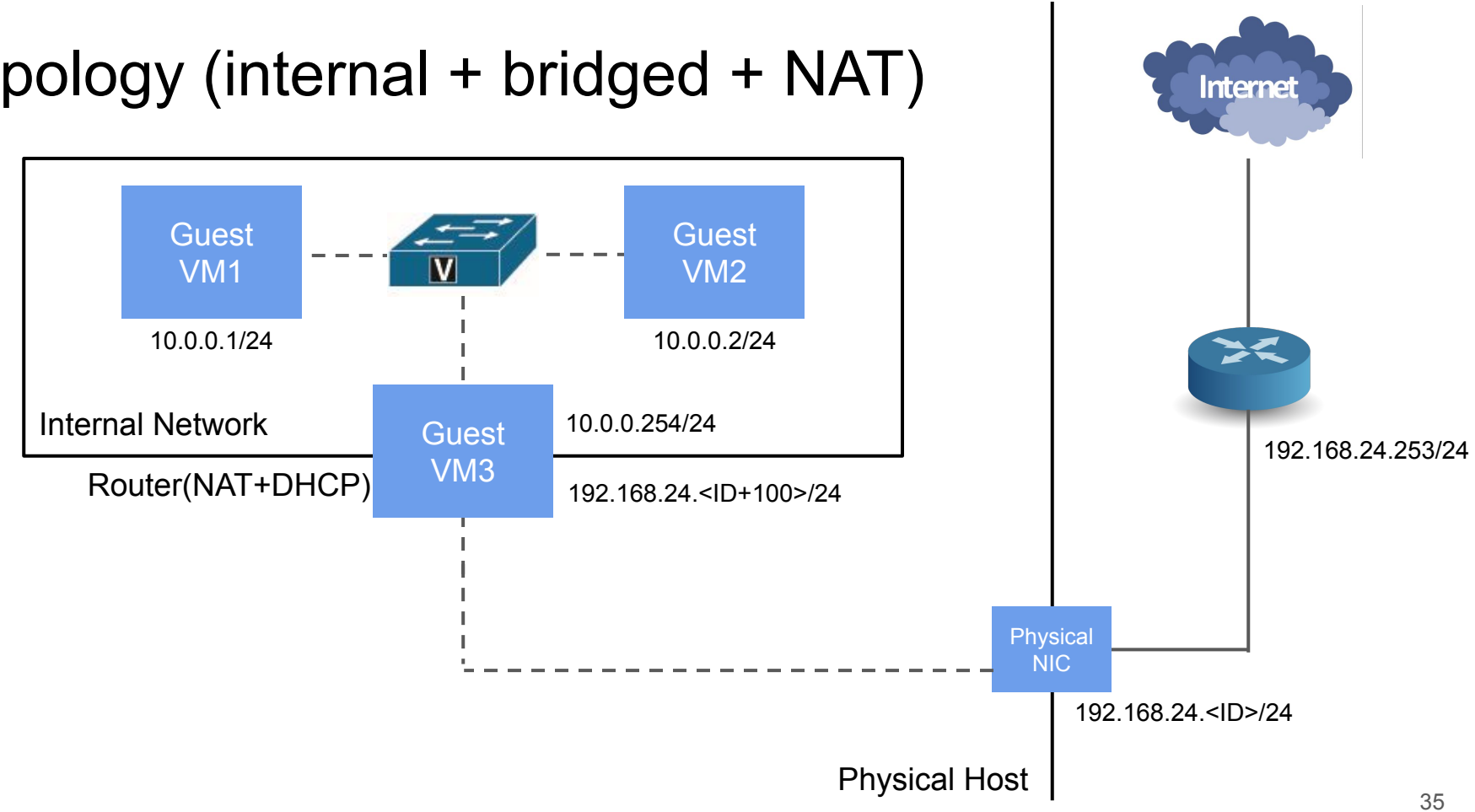


VirtualBox / Host-Only Adapter

- Used for communicating between a host and guests
- A VM can communicate with other VMs connected to the host-only network, and **with the host machine**
- Instead, a virtual network interface, similar to a loopback interface, is created on the host, **providing connectivity among virtual machines and the host**



Topology (internal + bridged + NAT)



VM3 setup

- configure IPs

```
sudo ip link set enp0s10 up
sudo ip addr add 192.168.24.<ID+100>/24 dev enp0s10
```

- configure default gateway

```
sudo ip route add default via 192.168.24.253 dev enp0s10
```

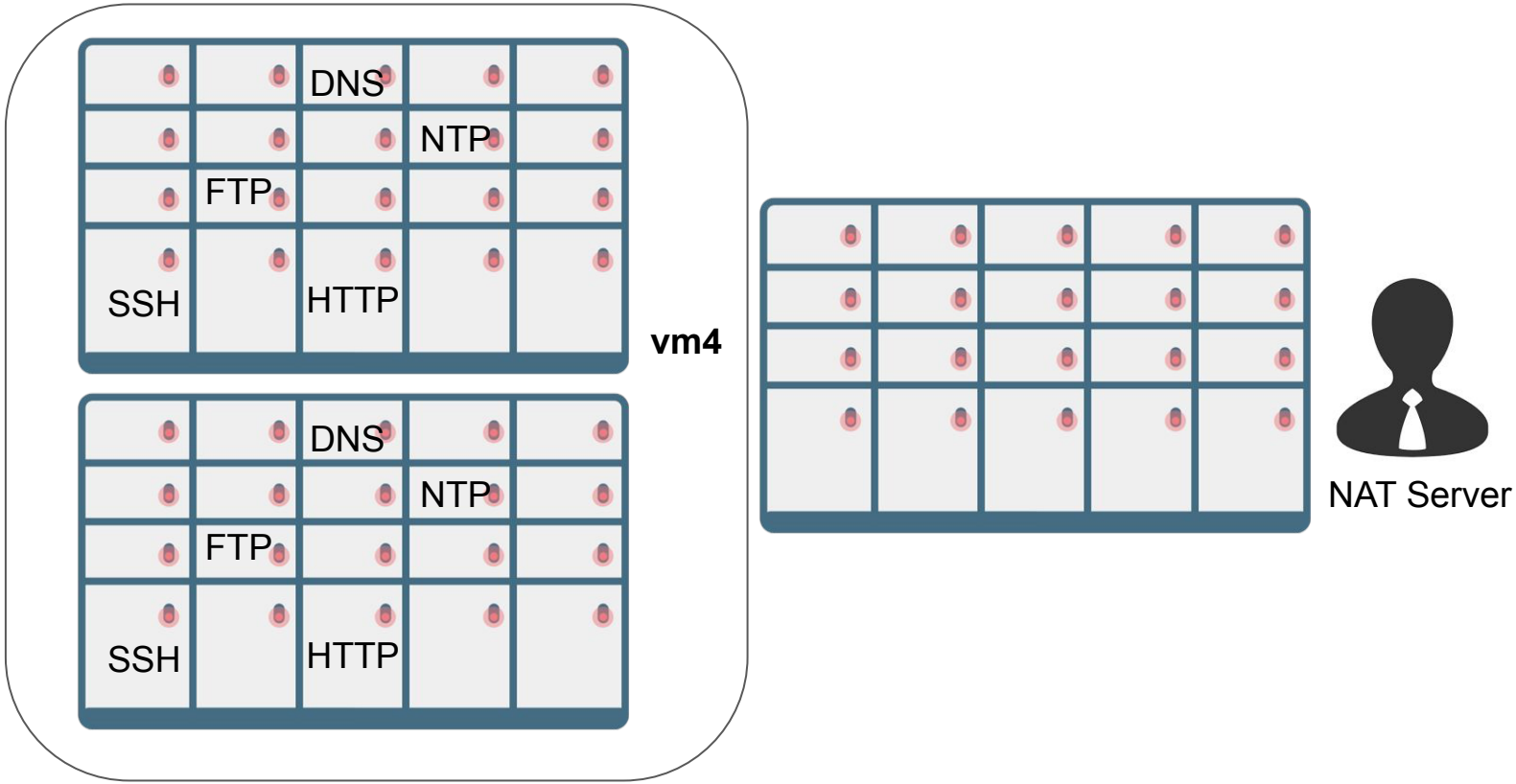
Check the correctness of topology

- Check their IP address.
  - The IP address on vms should be **different**, and in the same subnet.
- Ping 1.1.1.1
  - Every vm should be able to reach the Internet.
- Ping each other.
- Find the default gateway of vm1 & vm2 (*ip --color route*), which is **vm3**.
- Ping the default gateway.
  - The default gateway, vm3, will reply ICMP.

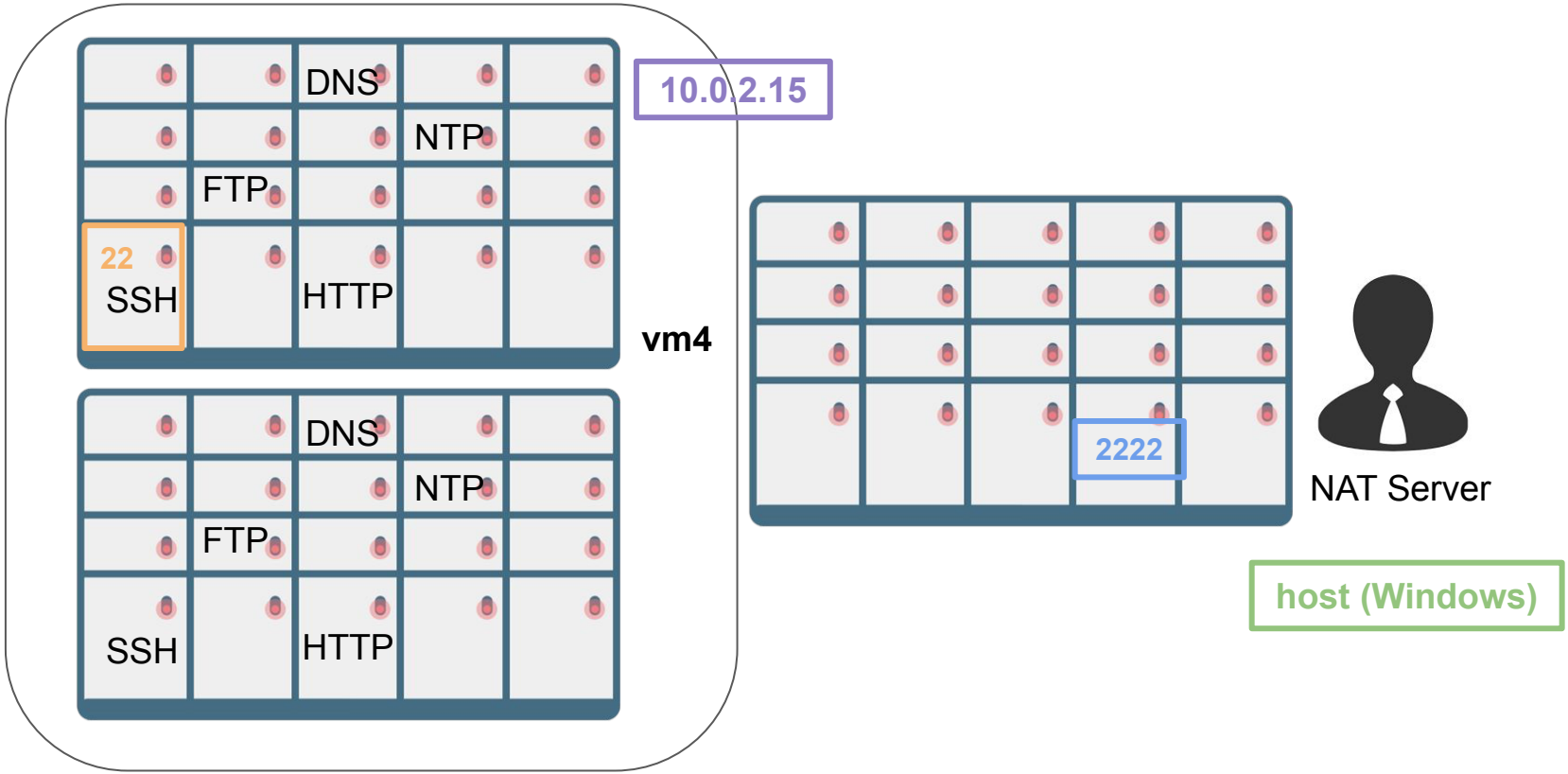
VM connection

Mode	**VM → Host**	**VM ← Host**	**VM1 ↔ VM2**	**VM → Net/LAN**	**VM ← Net/LAN**
Host-only	**+**	**+**	**+**	**–**	**–**
Internal	**–**	**–**	**+**	**–**	**–**
Bridged	**+**	**+**	**+**	**+**	**+**
NAT	**+**	Port forwarding	**–**	**+**	Port forwarding
NAT network	**+**	Port forwarding	**+**	**+**	Port forwarding

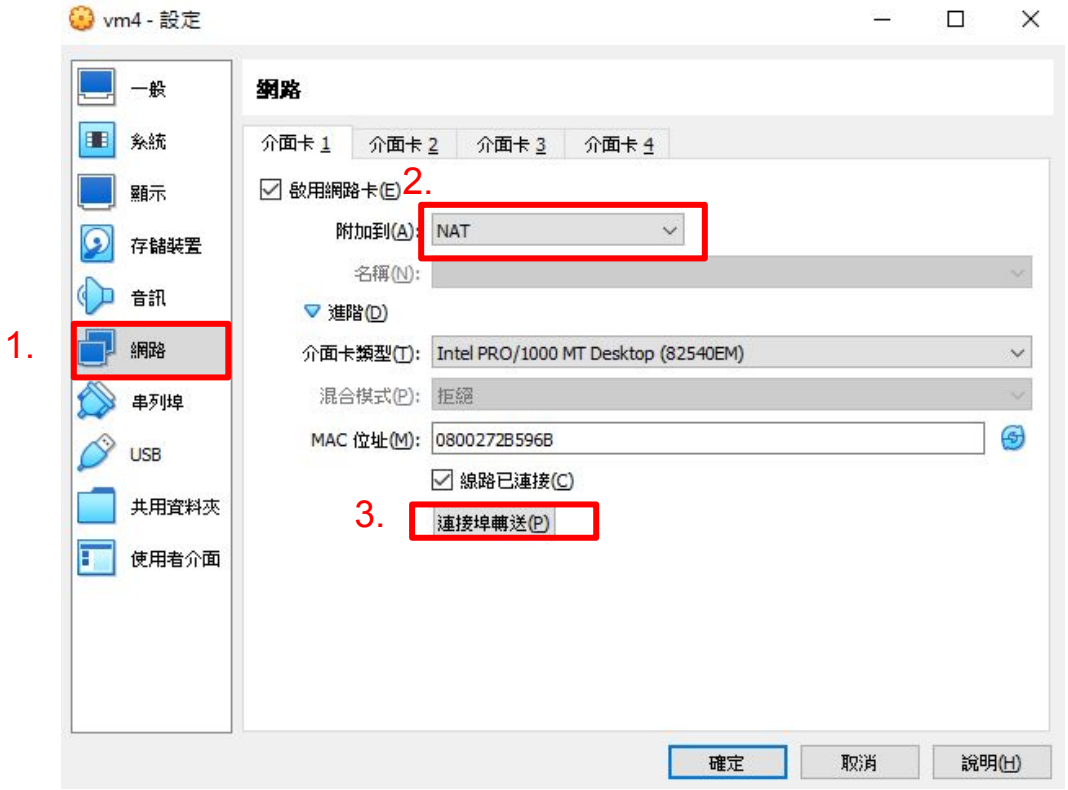
Port Forwarding



Port Forwarding

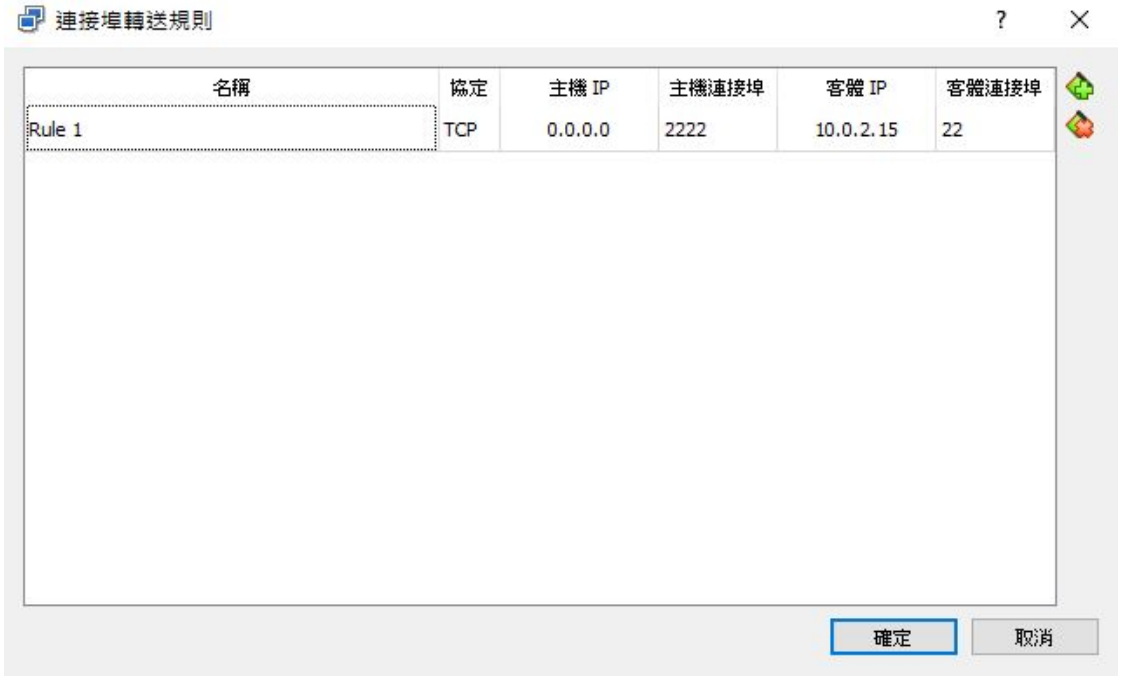


Virtual Box Port Forwarding Setting



Virtual Box Port Forwarding Setting

- 0.0.0.0: any IP address on host (Windows)

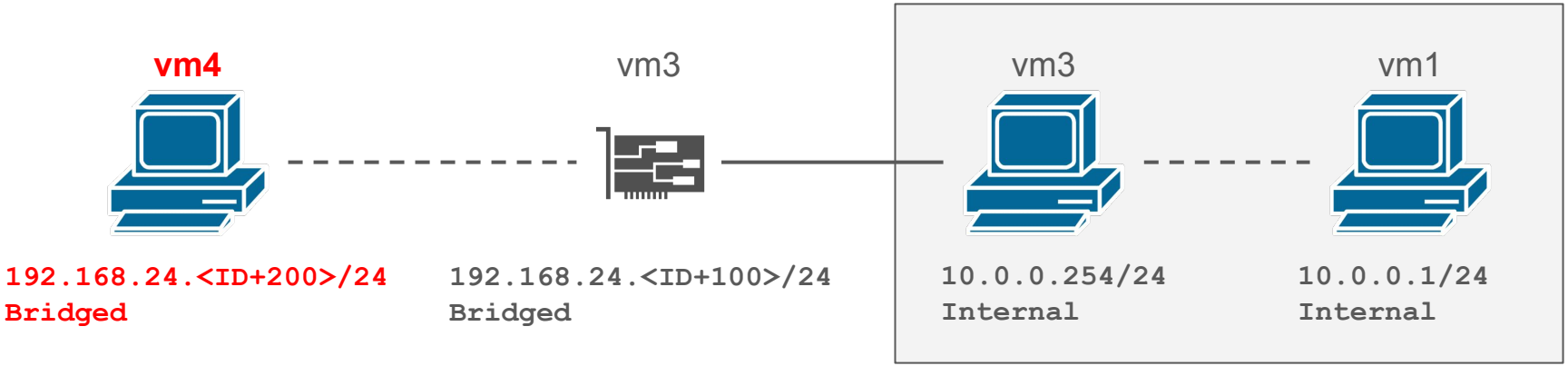


Virtual Box Port Forwarding Setting

- Try ssh from host (Windows).
- Open the terminal on the Windows and initiate an SSH connection.

```
C:\Users\twchou> ssh localhost -p 2222 -l ccna
ccna@localhost's password:
Last login: Sun Oct 27 15:48:35 2024
[ccna@vm4 ~]$
```

Topology


vm4
vm3
vm3
vm1
192.168.24.<ID+200>/24
Bridged
192.168.24.<ID+100>/24
Bridged
10.0.0.254/24
Internal
10.0.0.1/24
Internal

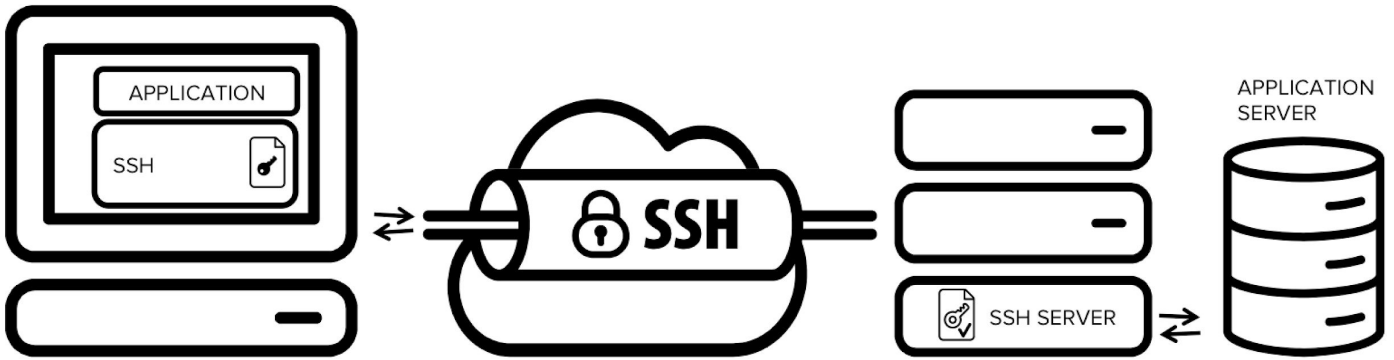
SSH Tunnel

- A method of transporting arbitrary networking data over an encrypted SSH connection.
- Provides a way to secure the data traffic of any given application using port forwarding, basically tunneling any TCP / IP port over SSH.

https://www.ssh.com/ssh/tunneling/

SSH Tunnel Procedure

1. Application contacts to a port that the SSH listens on.
2. SSH forwards the application over its encrypted tunnel to the other side.
3. SSH then connects to the actual application server.

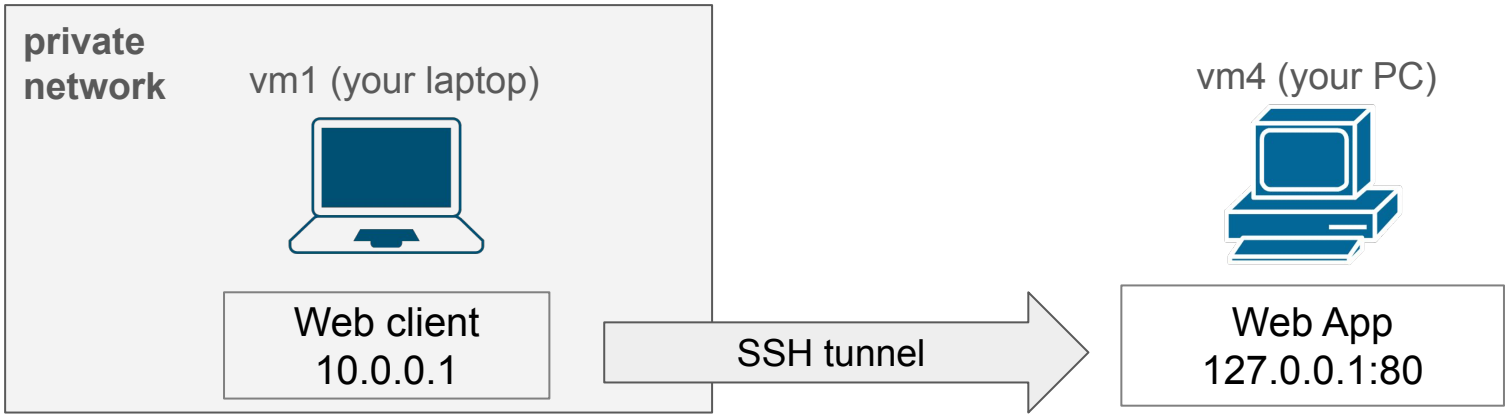


SSH Local Port Forwarding - Scenario

- You're doing your Web App homework with vm1 (your laptop) at Starbucks.
- Your Web App is running on vm4 (your PC) in your dorm.
  - The Web App server will only accecpt connections from localhost.
- vm4 (your PC) has an SSH server running with a public IP.
- You want to access your Web App from vm1 (your laptop).

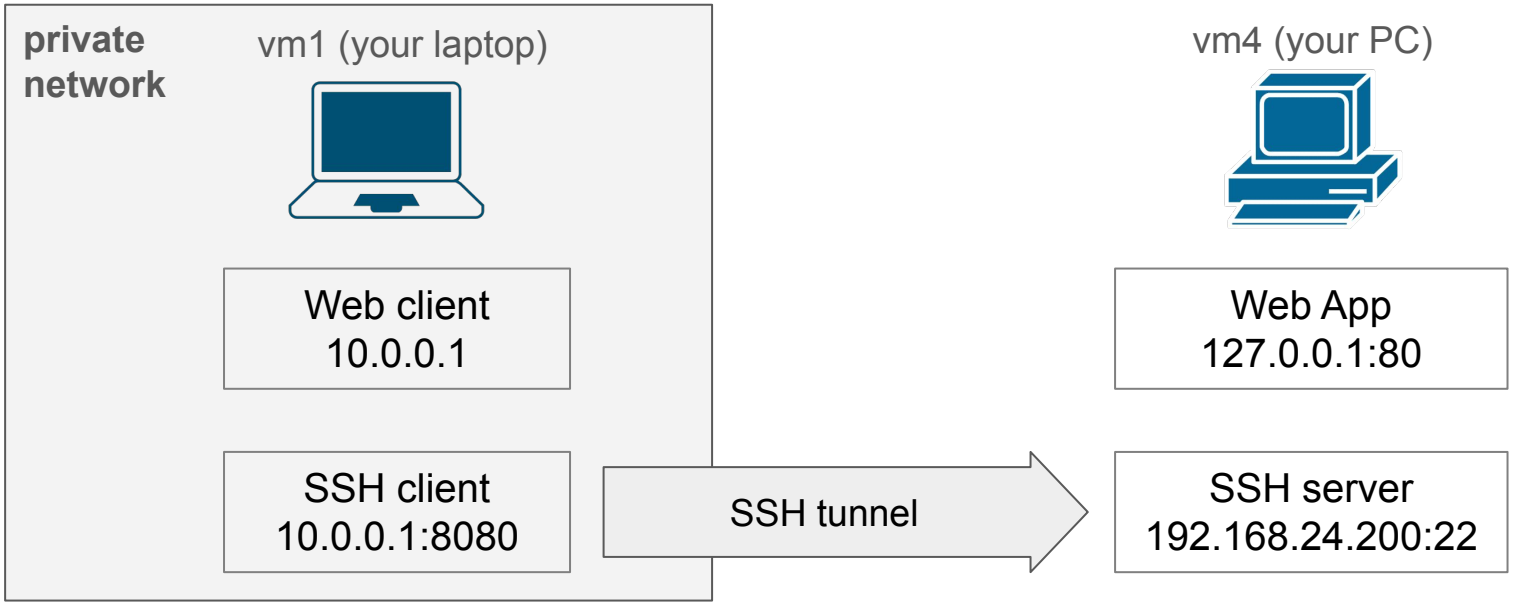
SSH Local Port Forwarding - Solution

- **`-L`** `[`*`bind_address`*`:]`*`port`*`:`*`host`*`:`*`hostport`*
- Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side.



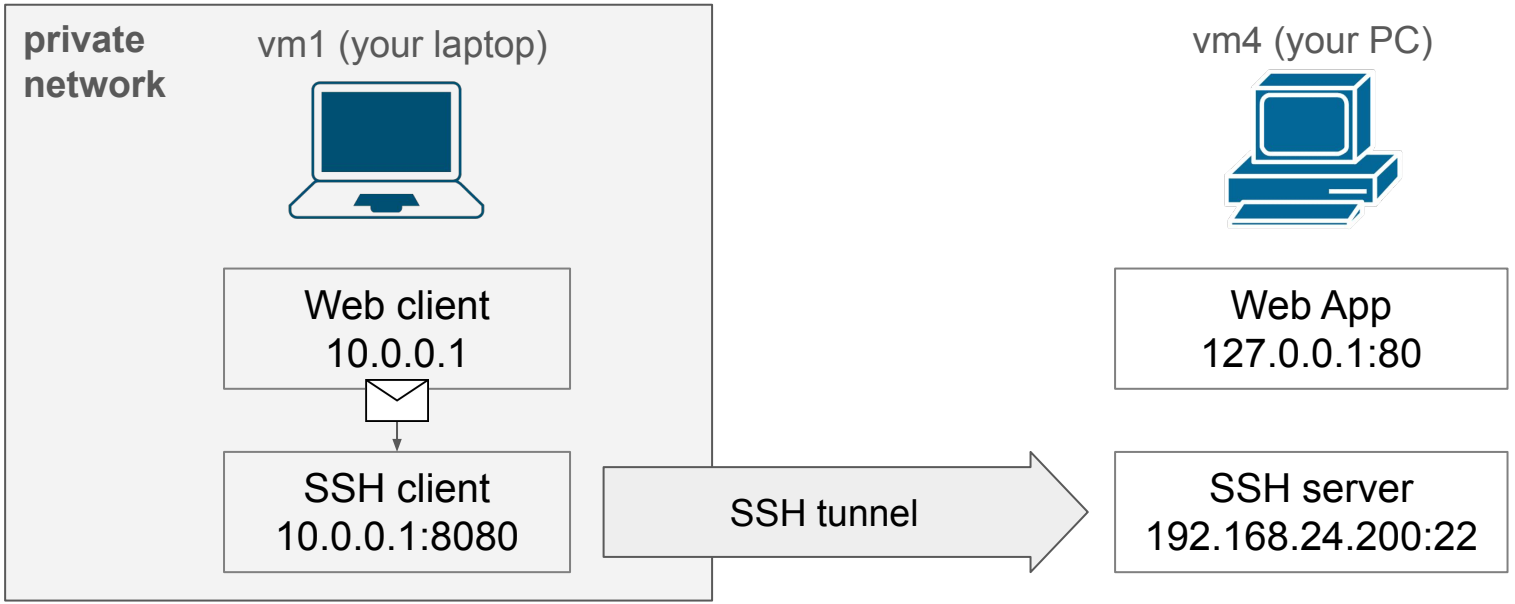
SSH Local Port Forwarding - Explanation

- Initialize an SSH connection from vm1 (your laptop) to vm4 (your PC)
  - Create an SSH tunnel



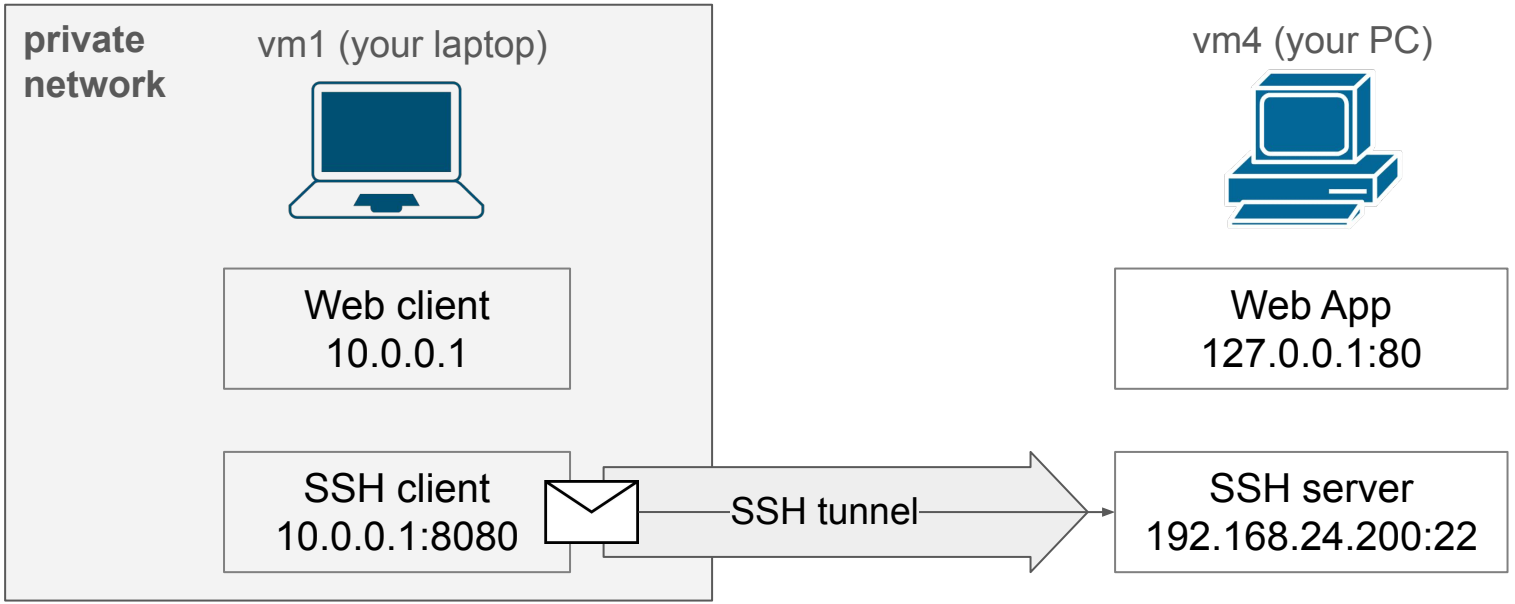
SSH Local Port Forwarding - Explanation

- Web client sends a packet to 10.0.0.1:8080 (localhost:8080 in vm1)



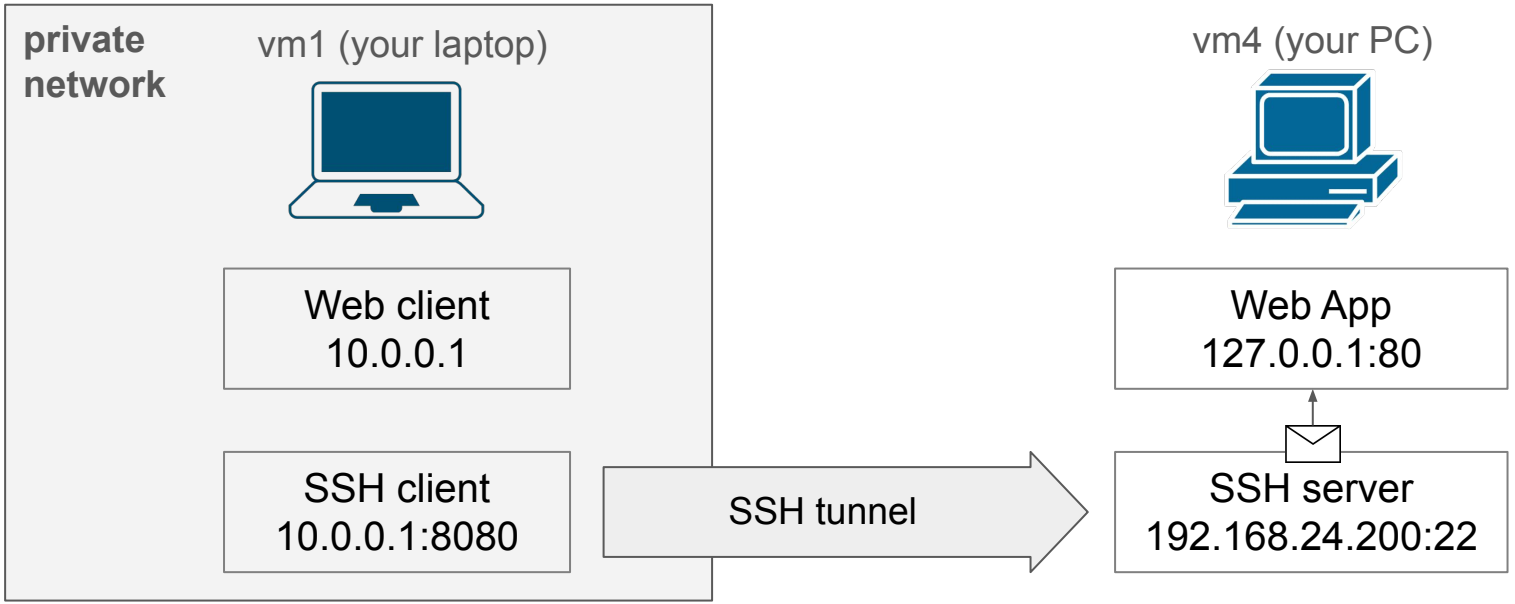
SSH Local Port Forwarding - Explanation

- SSH client forwards the packet to SSH server via SSH tunnel



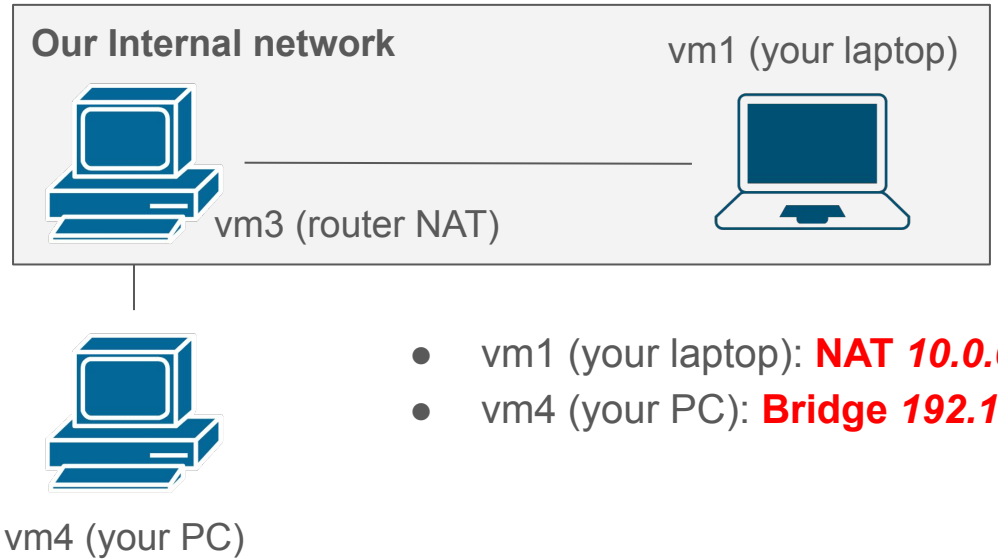
SSH Local Port Forwarding - Explanation

- SSH server then forward the packet to 127.0.0.1:80 (localhost:80 in vm4)



SSH Local Port Forwarding - Experiment Steps

1. Try to connect the Web App on vm4 (your PC) from vm1 (your laptop).
2. Use SSH local port forwarding to create tunnel.
3. Try to connect the Web App on vm4 (your PC) again.



- vm1 (your laptop): **NAT *10.0.0.1***
- vm4 (your PC): **Bridge *192.168.24.<ID+200>***

SSH Local Port Forwarding - Experiment Step 1

- Try to access the Web App on vm1(his laptop) from vm4(your PC).
  - Use the ip addresses of your own machines.

```
[ccna@vm1 ~]$ curl 192.168.24.<ID+200>
curl: (7) Failed to connect to 192.168.24.<ID+200> after 8 ms: Couldn’t
connect to server
[ccna@vm1 ~]$ curl localhost:8080
curl: (7) Failed to connect to localhost port 8080 after 0 ms: Couldn’t
connect to server
```

SSH Local Port Forwarding - Experiment Step 2

- Use SSH local port forwarding to create tunnel.
  - **-L** [*bind_address*:]*port*:*host*:*hostport*
  - **-N** *Do not execute a remote command.*
  - **-f** *Requests ssh to go to background just before command execution.*

```
[ccna@vm1 ~]$ ssh 192.168.24.<ID+200> -L 8080:localhost:80 -Nf
ccna@192.168.24.<ID+200>'s password:
[ccna@vm1 ~]$
```

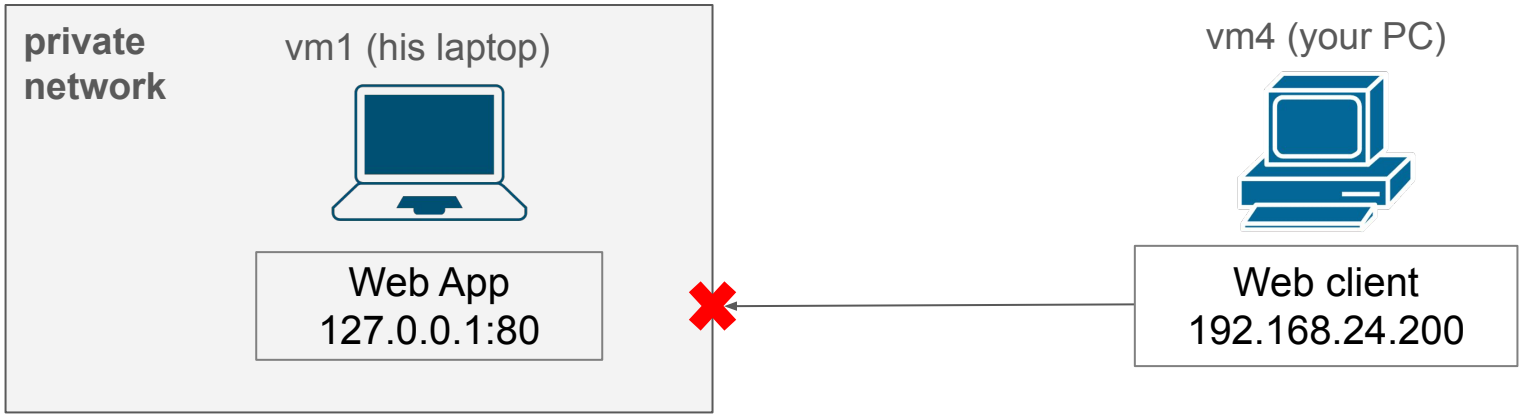
SSH Local Port Forwarding - Experiment Step 3

- Try to access the Web App again.

```
[ccna@vm1 ~]$ curl localhost:8080
hello, world from vm4
[ccna@vm1 ~]$
```

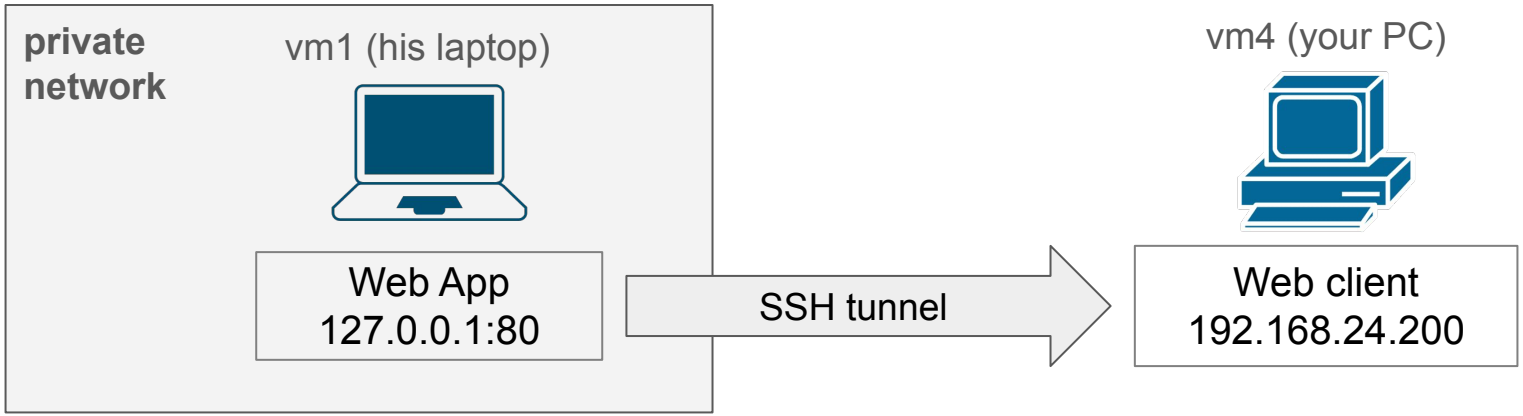
SSH Remote Port Forwarding - Scenario

- Your classmate want to show you his Web App on vm1 (his laptop) behind a private network, which is not publicliy accessible.
- On the other hand, vm4 (your PC) in your dorm is publicliy accessible.
- How can you access vm1 (his laptop) from vm4 (your PC)?



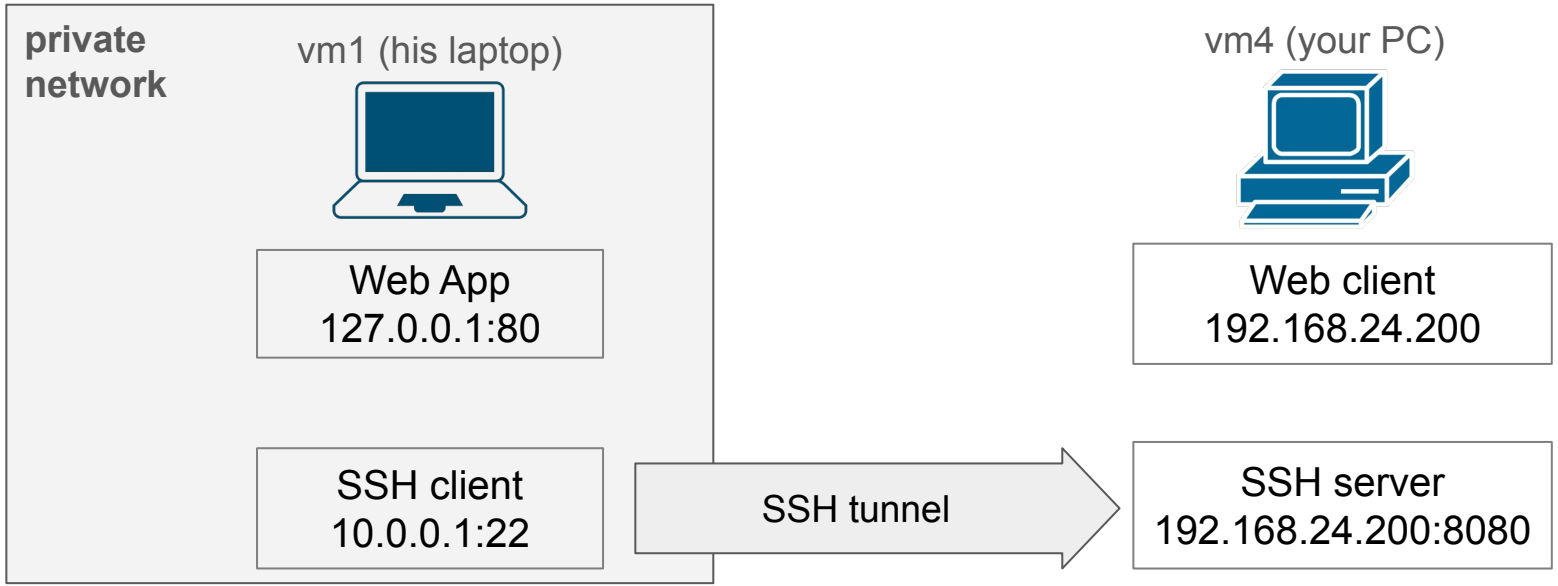
SSH Remote Port Forwarding - Solution

- **`-R`** `[`*`bind_address`*`:]`*`port`*`:`*`host`*`:`*`hostport`*
- Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side.



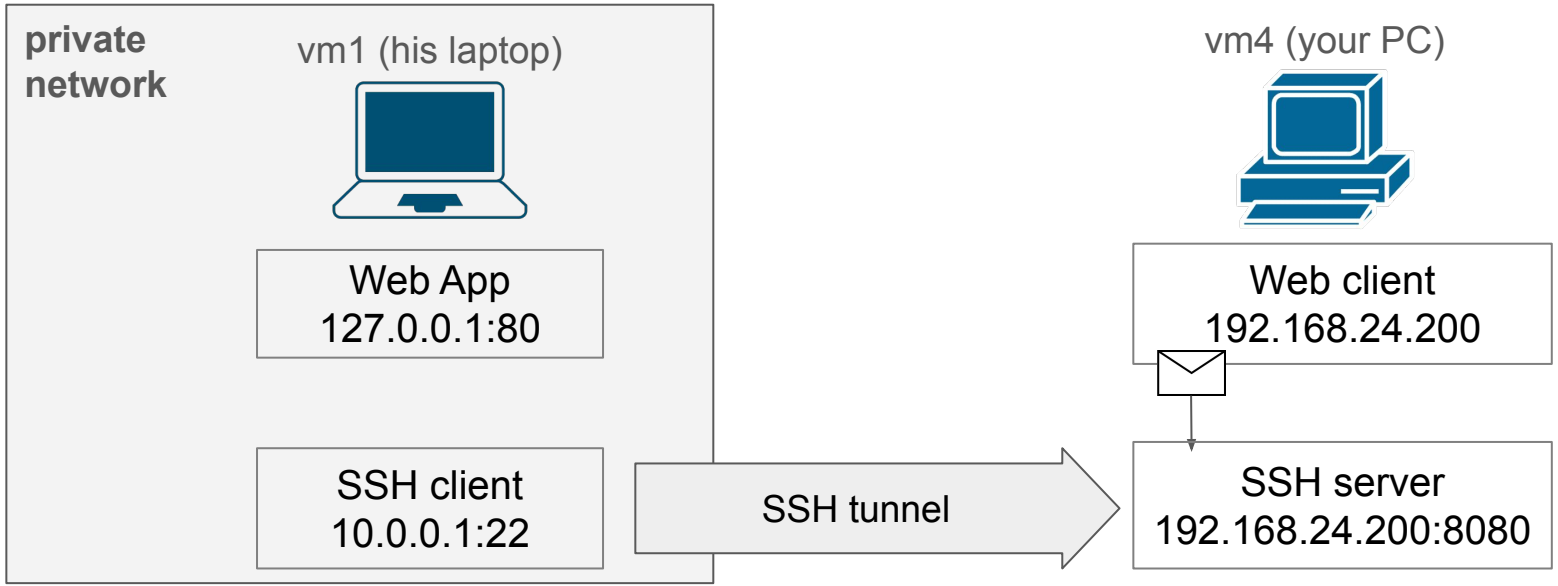
SSH Remote Port Forwarding - Explanation

- Initialize an SSH connection from his laptop to your PC
  - Create an SSH tunnel



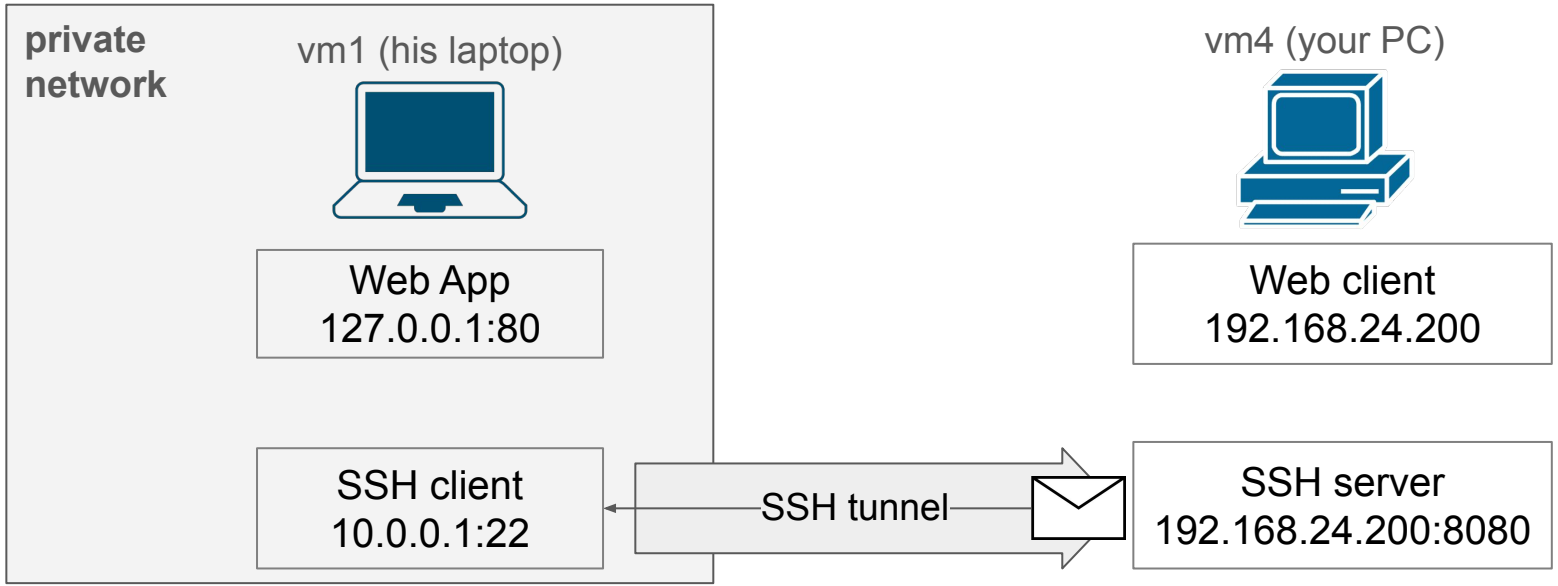
SSH Remote Port Forwarding - Explanation

- Web client sends a packet to 192.168.24.200:8080 (localhost:8080 in vm4)



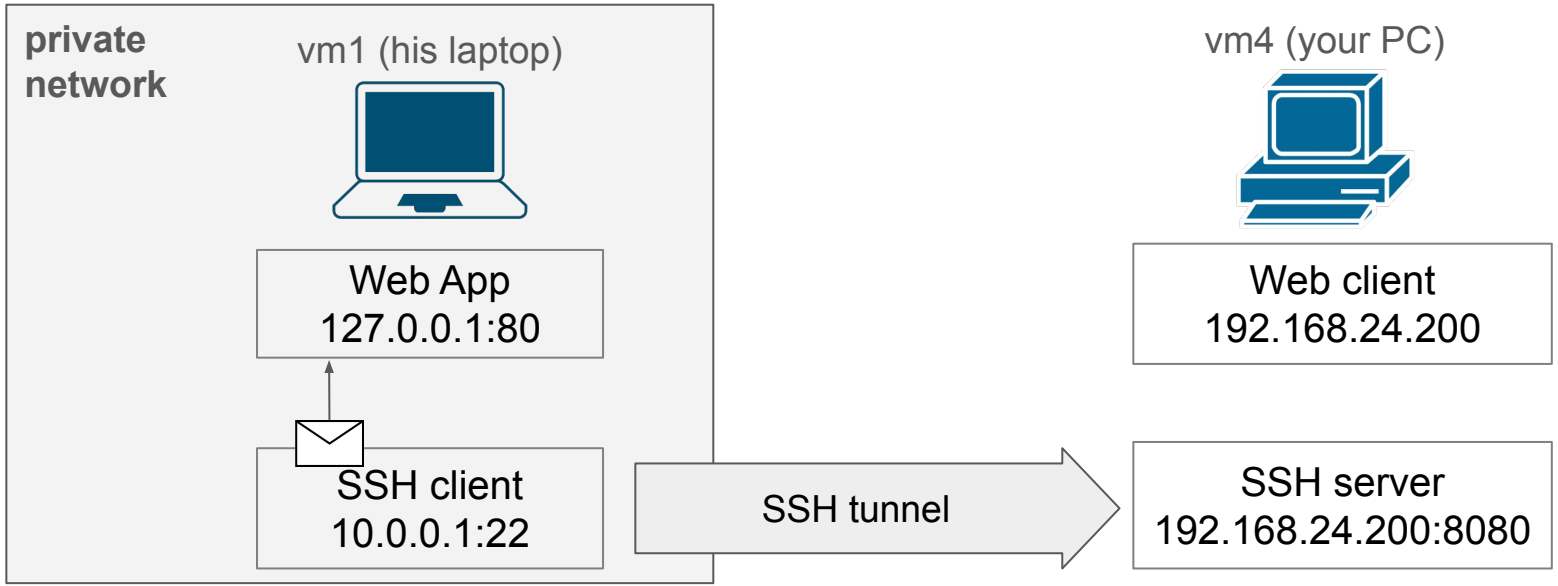
SSH Remote Port Forwarding - Explanation

- SSH server forwards the packet to SSH client via the SSH tunnel



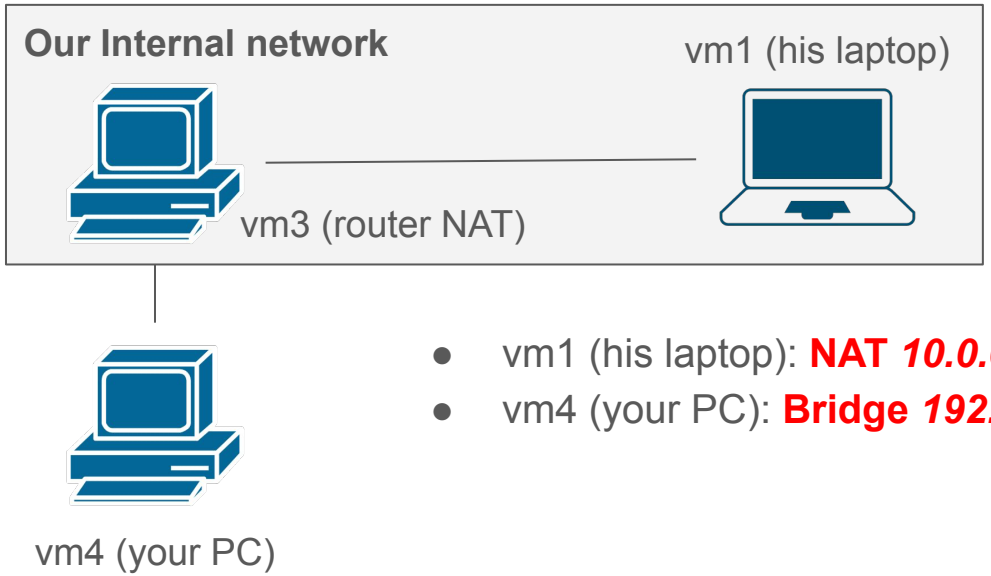
SSH Remote Port Forwarding - Explanation

- SSH client forwards the packet to the Web App on his laptop.



SSH Remote Port Forwarding - Experiment Steps

1. Try to access the Web App on vm1 (his laptop) from vm4 (your PC).
2. Use SSH remote port forwarding to create tunnel.
3. Try to access the Web App again.



- vm1 (his laptop): **NAT *10.0.0.1***
- vm4 (your PC): **Bridge *192.168.24.<ID+200>***

SSH Remote Port Forwarding - Experiment Step 1

- Try to access the Web App on vm1 (his laptop) from vm4 (your PC).
  - Use the ip addresses of your own machines.
- You can stop the previous ssh connections by running `killall ssh`.

```
[ccna@vm4 ~]$ curl 10.0.0.1
curl: (28) Connection timed out after 10002 milliseconds
[ccna@vm4 ~]$ curl localhost:8080
curl: (7) Failed to connect to localhost port 8080 after 0 ms: Couldn’t
connect to server
```

SSH Remote Port Forwarding - Experiment Step 2

- Use SSH remote port forwarding to create tunnel.
  - **-R** [*bind_address*:]*port*:*host*:*hostport*
  - **-N** *Do not execute a remote command.*
  - **-f** *Requests ssh to go to background just before command execution.*

```
[ccna@vm1 ~]$ ssh 192.168.24.<ID+200> -R 8080:localhost:80 -Nf
ccna@192.168.24.<ID+200>'s password:
[ccna@vm1 ~]$
```

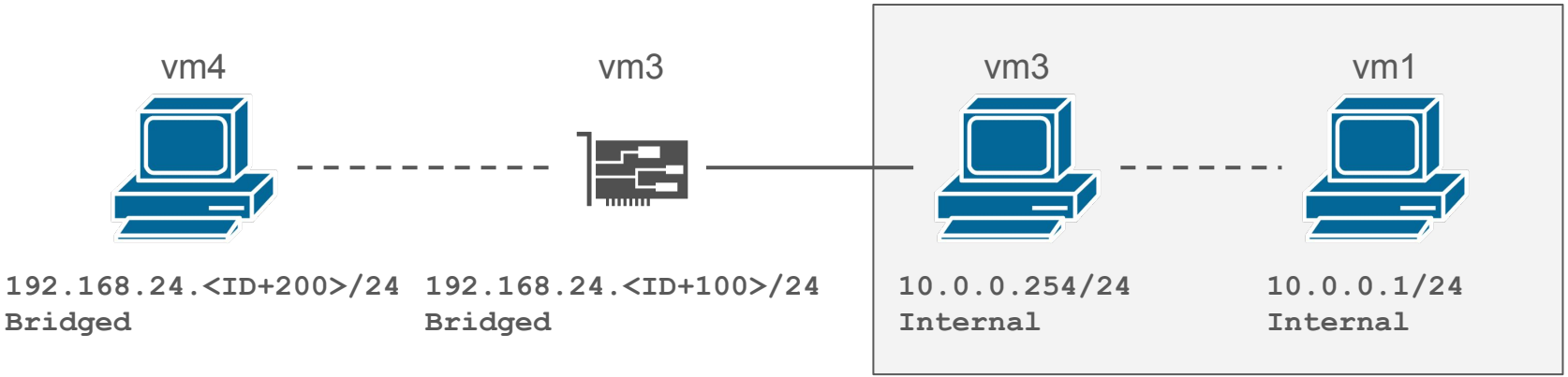
SSH Remote Port Forwarding - Experiment Step 3

- Try to access the Web App again.

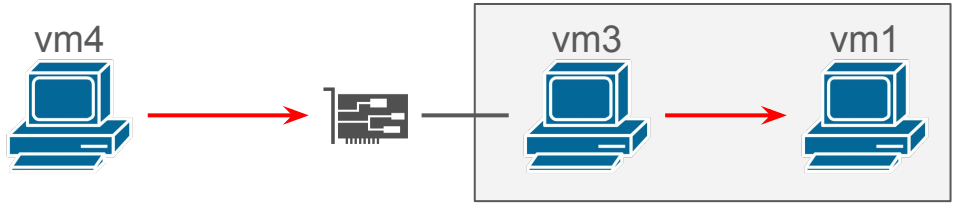
```
[ccna@vm4 ~]$ curl localhost:8080
hello, world from vm1
[ccna@vm4 ~]$
```

SSH Proxy Jump - Scenario & Spec

- You have to access *vm1*, but *vm1* is in a private network.
- However, there's another machine *vm3*, which have two NIC.
  - One is in the private network.
  - The other is in a network that *vm4* can access.



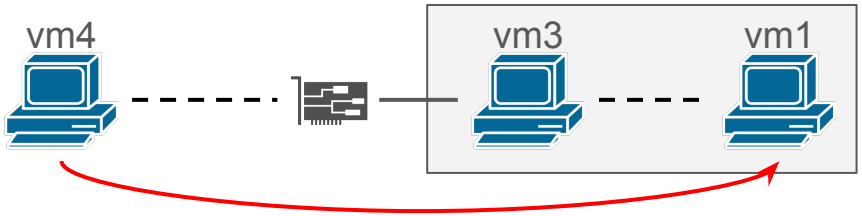
SSH Proxy Jump - Issue



- ssh to *vm3* and then ssh to *vm1* might be a good approach.
- This can be annoying if you have to constantly access *vm1*.

```
[ccna@vm4 ~]$ ssh 192.168.24.<ID+100> -l ccna
ccna@192.168.24.<ID+100>'s password:
Last login: Sun Oct 27 14:05:00 2024
[ccna@vm3 ~]$ ssh 10.0.0.1
ccna@10.0.0.1's password:
Last login: Sun Oct 27 14:05:23 2024
[ccna@vm1 ~]$
```

SSH Proxy Jump - Solution



- **-J** *[user@]destination[:port]*
- Setting this option will cause ssh to connect to the target host by first making a ssh connection to the specified ProxyJump host and then establishing a TCP forwarding to the ultimate target from there.

```
[ccna@vm4 ~]$ ssh 10.0.0.1 -l ccna -J ccna@192.168.24.<ID+100>
ccna@192.168.24.<ID+100>'s password:
ccna@10.0.0.1's password:
Last login: Sun Oct 27 14:12:05 2024
[ccna@vm1 ~]$
```

That’s all

- Practice the experiments in the slides.
- Feel free to ask any question.