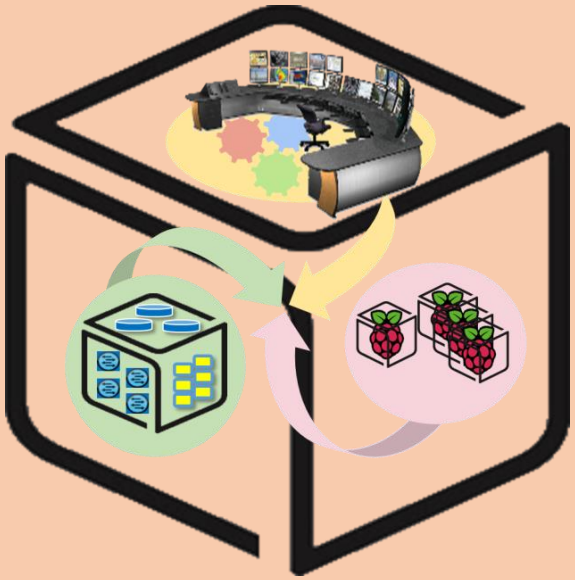
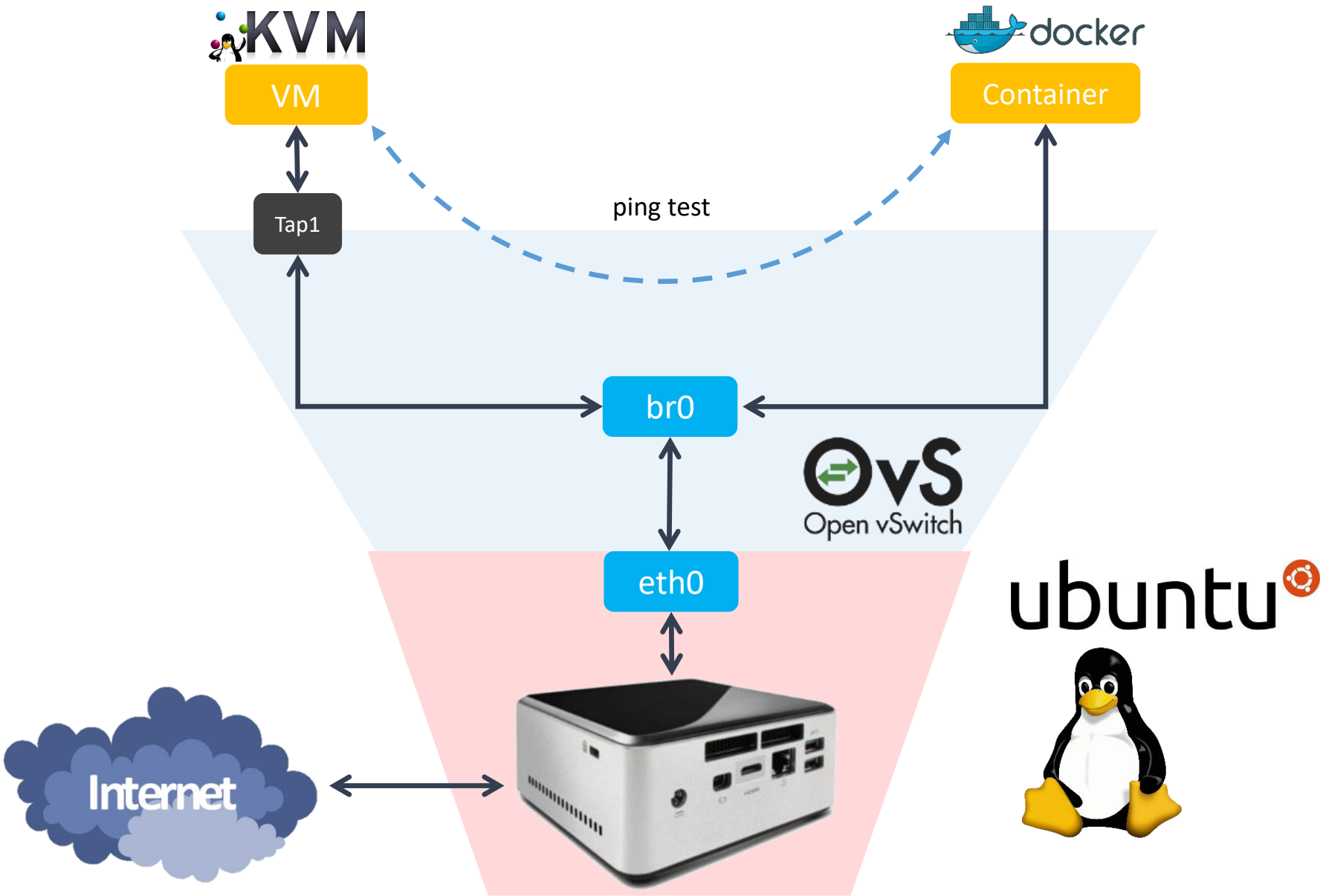
Computer Systems For AI-inspired Cloud Theory & Lab.

Lab #1: Box

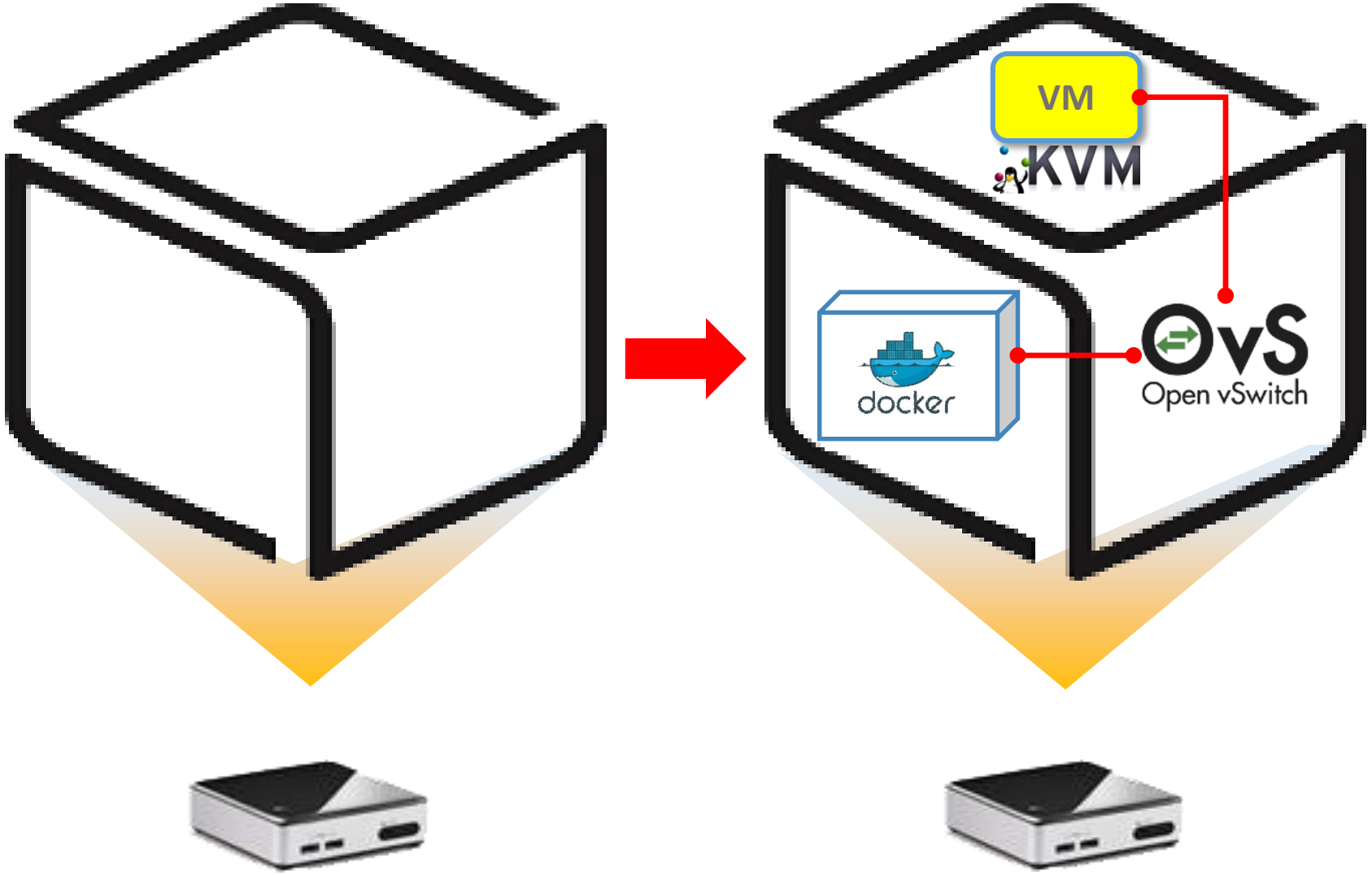


https://github.com/SmartX-Labs/SmartX-Mini-MOOC

Box Lab: Outline



Box Lab: Final Goal



Before you start

- Things you need to know

Lab Theory

Lab Practice

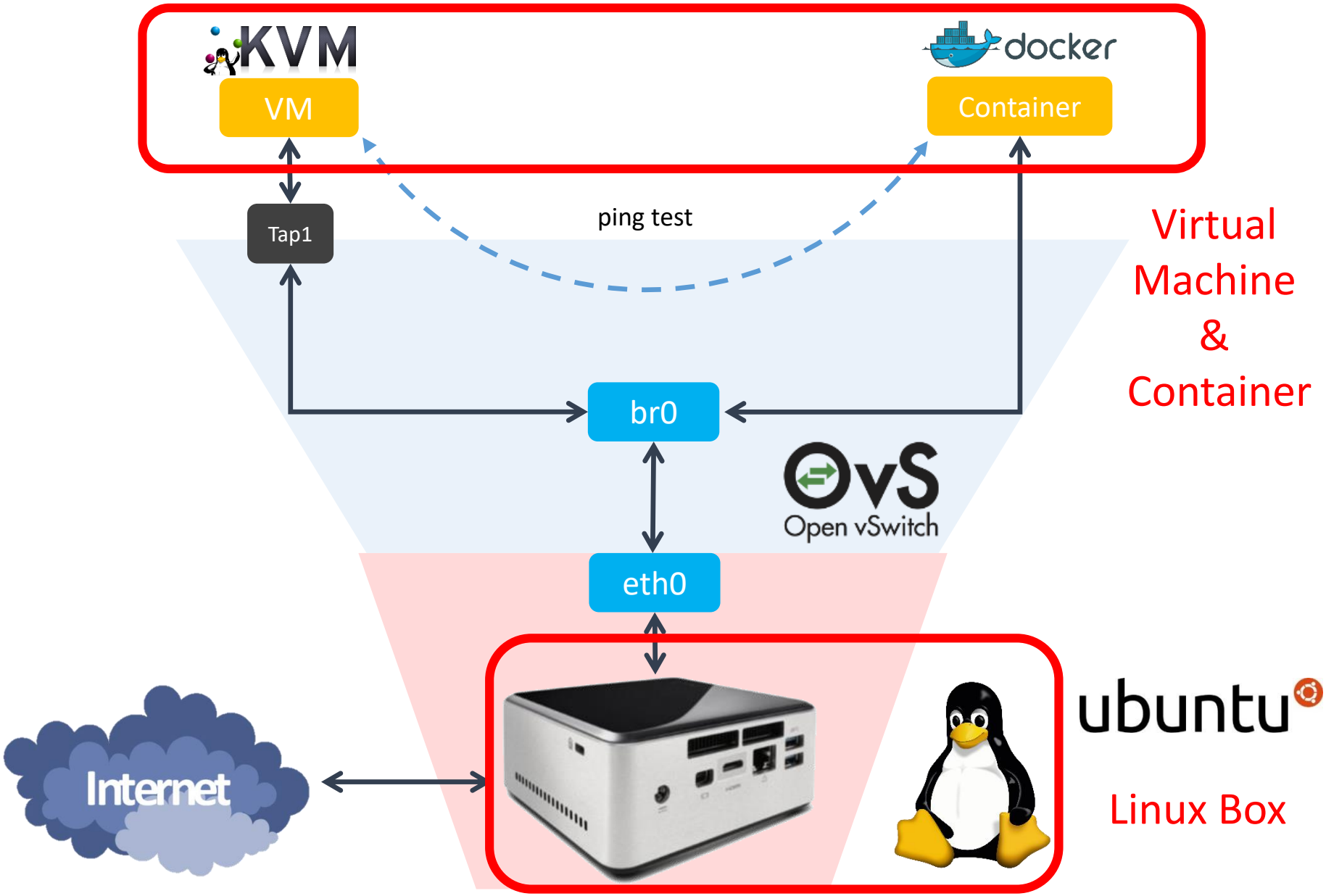
Lab Review

Lectures are divided into Lab Theory, Lab Practice and Lab Review parts.

The keyboard means that you should execute instructions by following the guidance.

Lab Theory

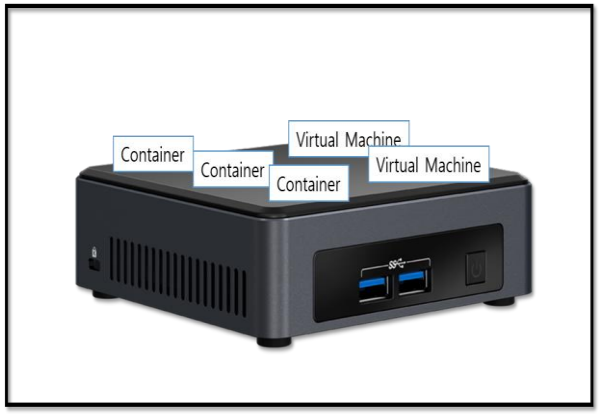
Linux Box with Virtualization/Containers



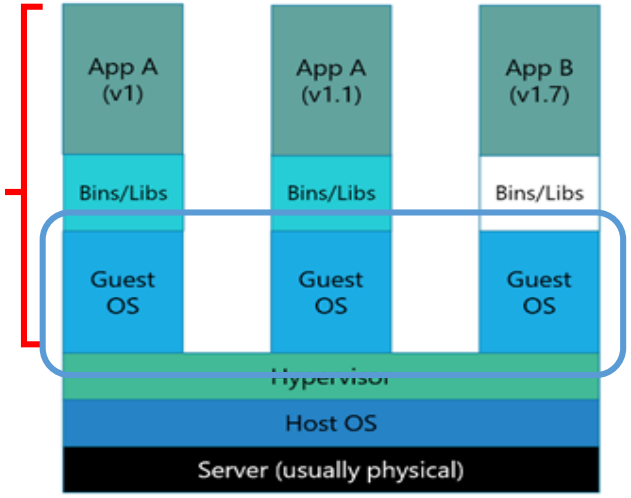
VMs and Containers on a Linux Box

On a Linux Box (e.g., NUC)

- Multiple fully-isolated (with dedicated resources), but heavy VM instances (i.e., VMs).
- Multiple partially-isolated, lightweight container instances (i.e., containers)



Virtual Machines (VMs)

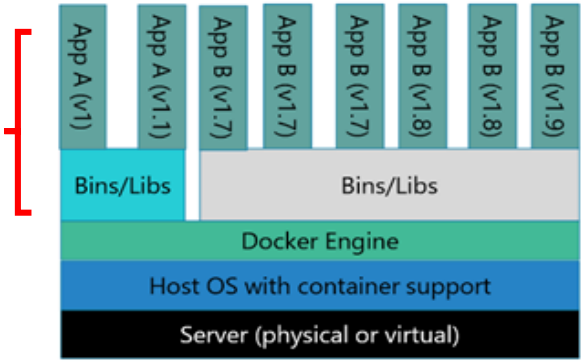


KVM

Heavier than Container!

Hypervisor Tool for creating VMs

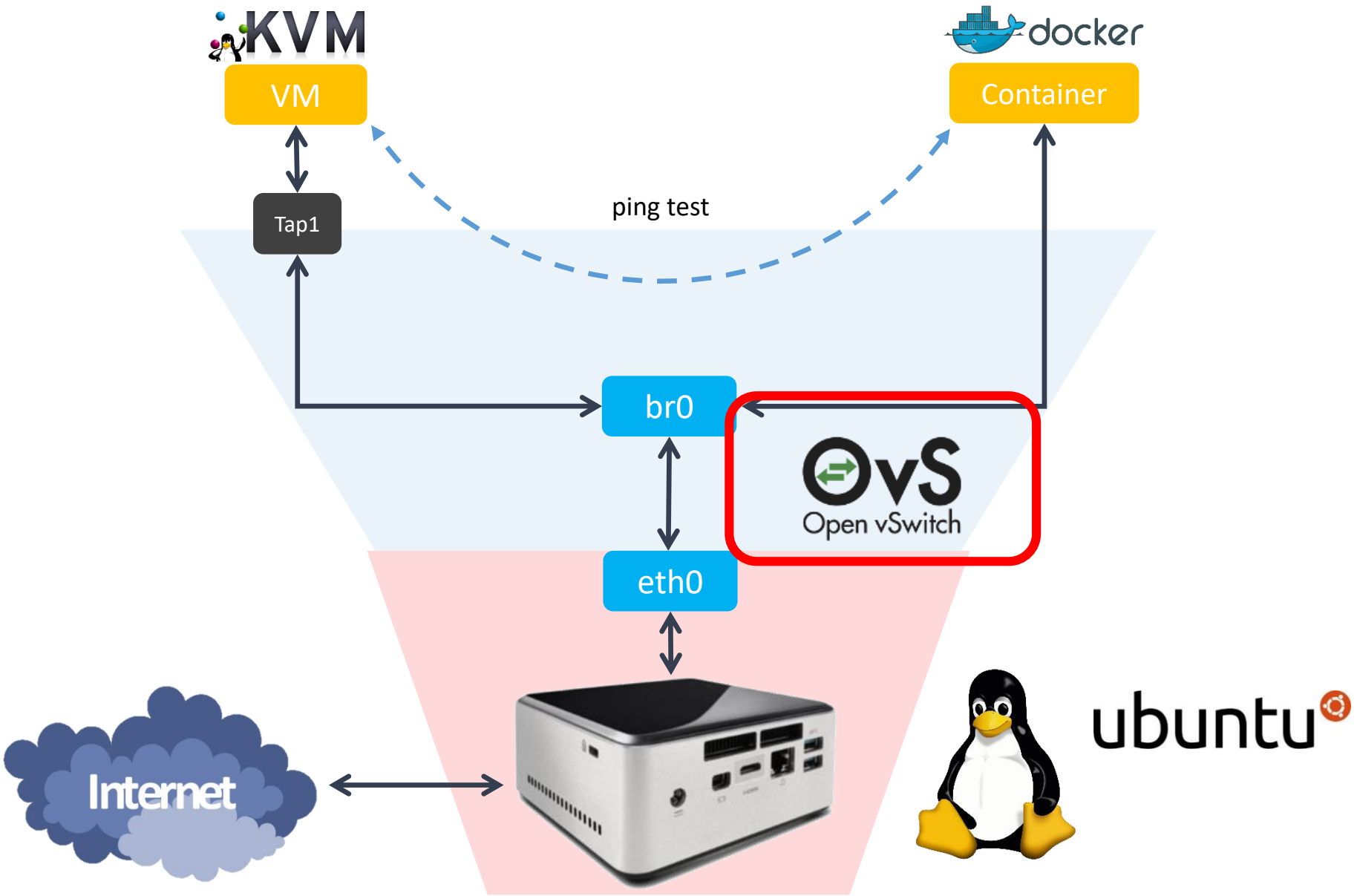
Containers



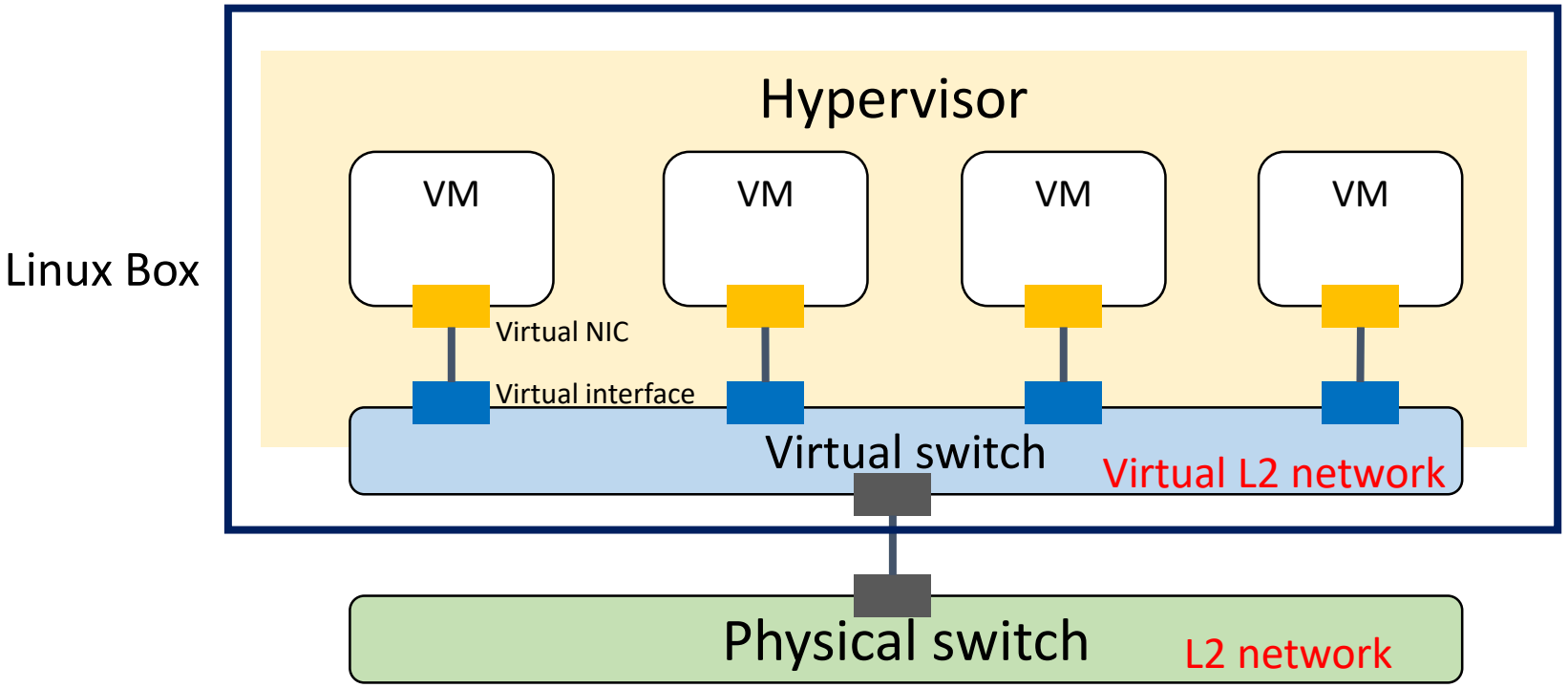
docker

Container Runtime Tool for running containers

A Switch inside Linux Box: Open vSwitch



Virtual Switch in a Box to connect VMs

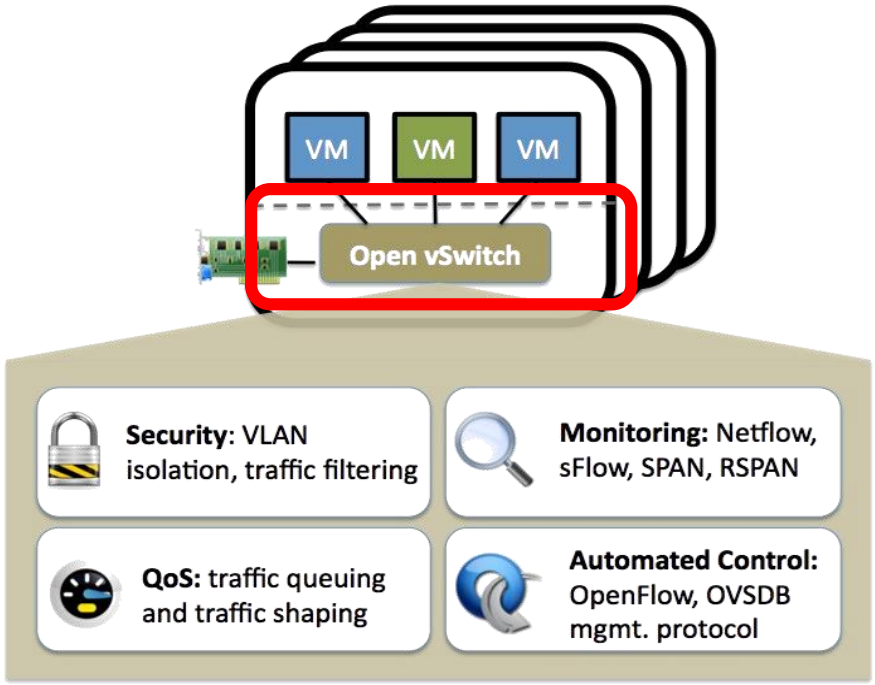


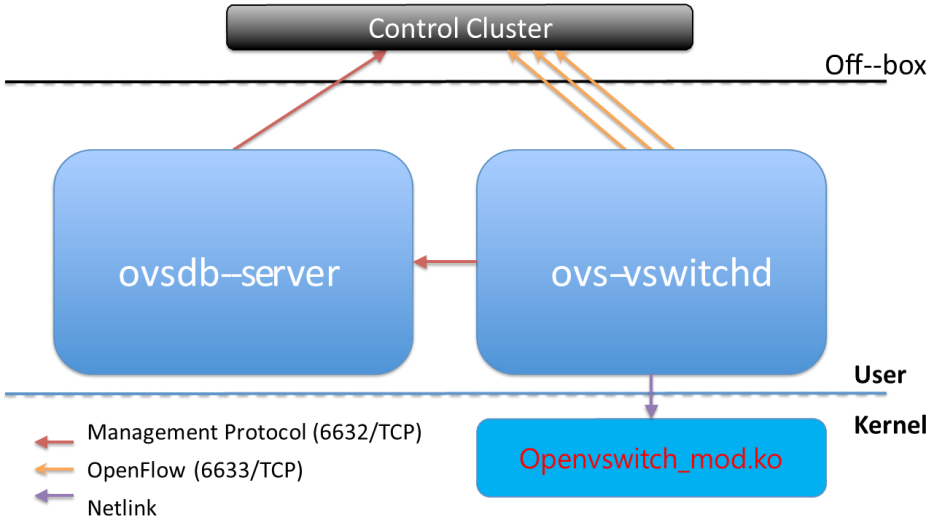
- A software-based virtual switch allows one VM to communicate with neighbor VMs as well as to connect to Internet (via physical switch).
- Software-based switches (running with the power of CPUs) are known to be more flexible/upgradable and benefited of virtualization (memory overcommit, page sharing, …)
- VMs (similarly containers) have logical (virtual) NIC with virtual Ethernet ports so that they can be plugged into the virtual interface (port) of virtual switches.

Linux-adopted virtual switch: Open vSwitch

http://openvswitch.org/

Open vSwitch is an open-source virtual switch software designed for virtual servers.





OVS Main components

Lab Practice

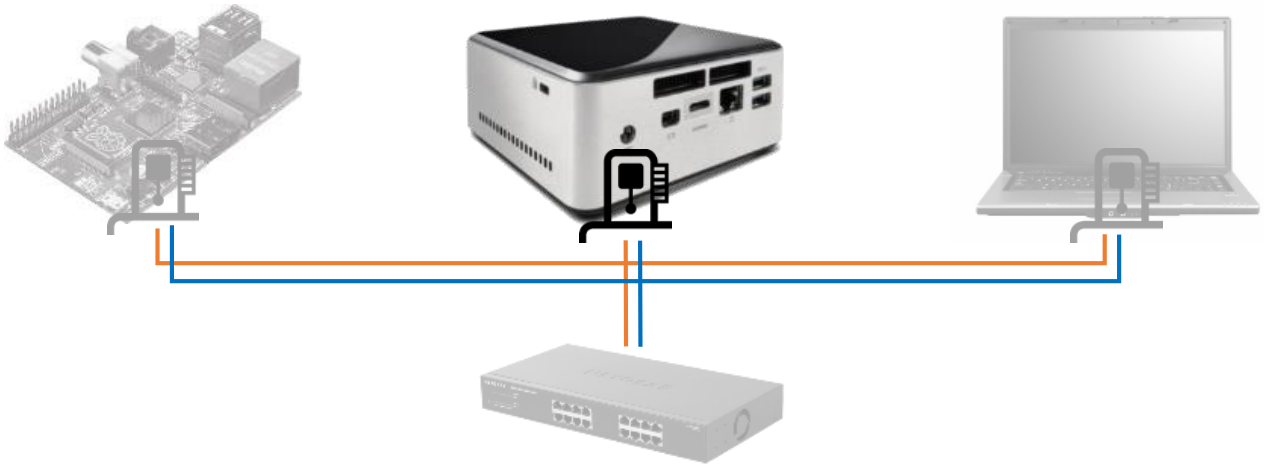
#0 - Lab Preparation

Wired connection

NAME: Raspberry Pi Model B (Pi)
CPU: ARM Cortex A7 @900MHz
CORE: 4
Memory: 1GB
SD Card: 32GB

NAME: NUC5i5MYHE (NUC PC)
CPU: i5-5300U @2.30GHz
CORE: 4
Memory: 16GB DDR3
HDD: 94GB

NAME: NT900X3A
CPU: i5-2537U @1.40GHz
CORE: 2
Memory: 4GB DDR3
HDD: 128GB



NAME: netgear prosafe 16 port gigabit switch(Switch)
Network Ports: 16 auto-sensing 10/100/1000 Mbps Ethernet ports

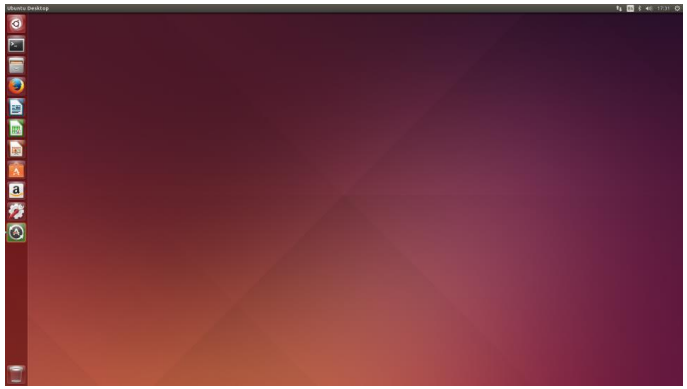
#1 - NUC: OS Installation

- OS : Ubuntu Desktop 16.04.4 LTS(64bit)
  - Download Site : http://old-releases.ubuntu.com/releases/16.04.4/

Index of /releases/16.04.4

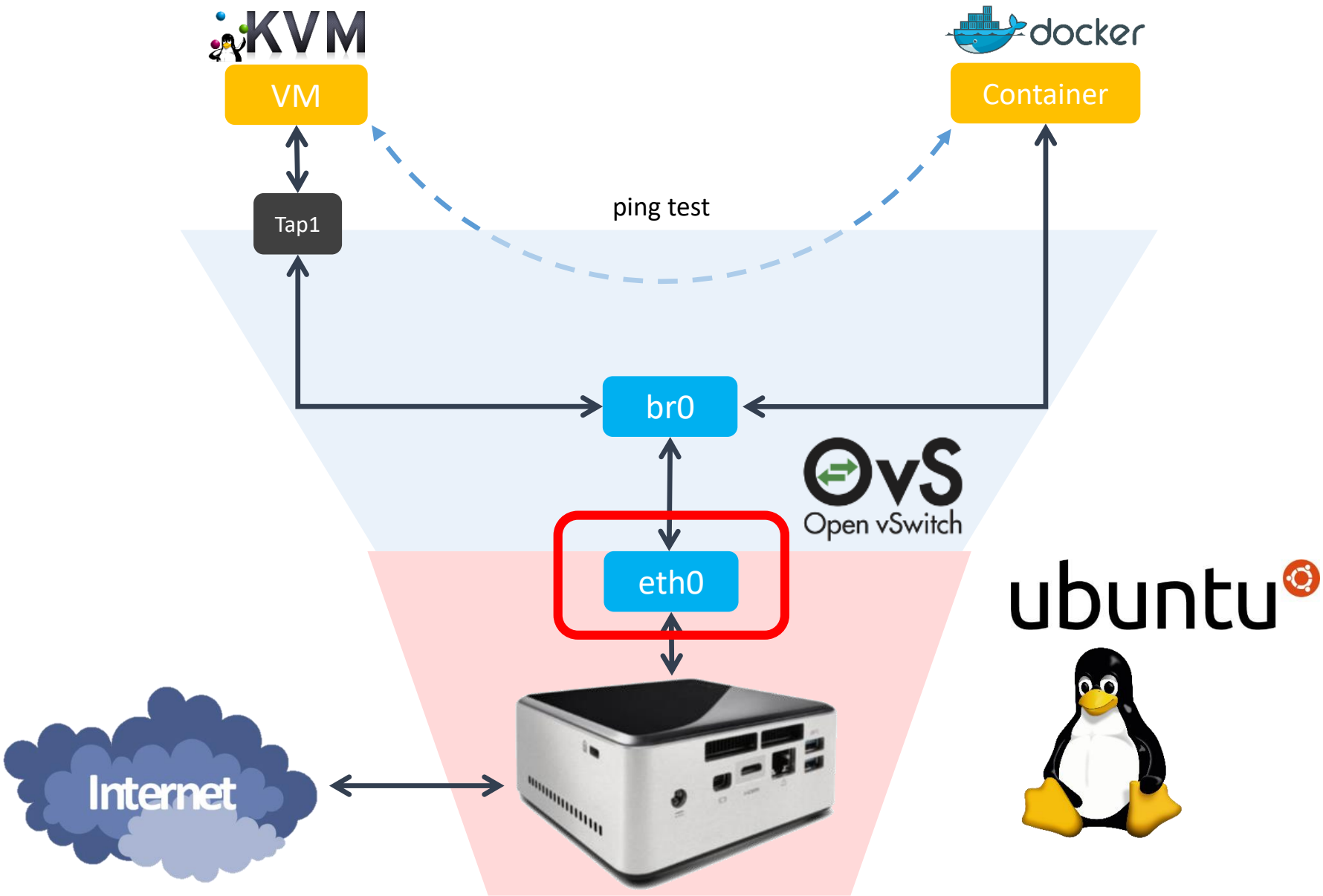
ubuntu-16.04.3-server-s390x.iso.torrent	2017-08-03 13:13	24K
ubuntu-16.04.3-server-s390x.iso.zsync	2017-08-03 13:13	1.2M
ubuntu-16.04.3-server-s390x.jigdo	2017-08-03 13:12	128K
ubuntu-16.04.3-server-s390x.list	2017-08-01 11:37	91K
ubuntu-16.04.3-server-s390x.metalink	2018-03-01 20:20	1.0K
ubuntu-16.04.3-server-s390x.template	2017-08-01 11:37	115M
ubuntu-16.04.4-desktop-amd64.iso	2018-02-28 19:15	1.5G

  - Bootable USB configuration (no bootable CD, no CD-Rom in NUC) using the downloaded file (ubuntu-16.04.4-desktop-amd64.iso, 1.5Gb)
  - Installed on NUC



Ubuntu Home Screen After Installation

#2 - NUC: Network Configuration (1/4)

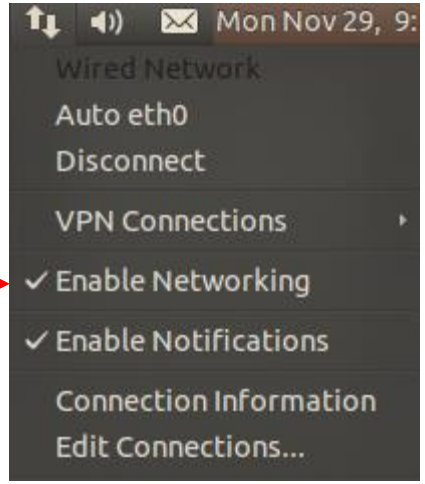


#2 - NUC: Network Configuration (1/4)

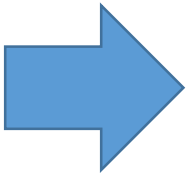
- eth0 interface

Do not check 'Enable Networking'

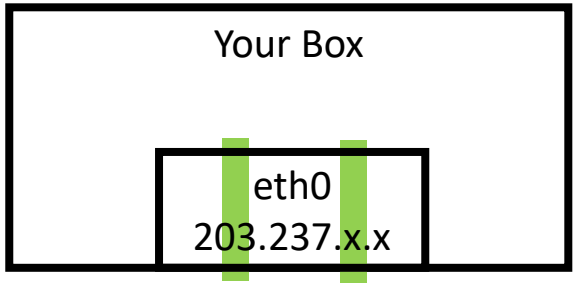
```
$sudo vi /etc/network/interfaces
----------------- /etc/network/interfaces -----------------
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address    [ip address]
        netmask    [subnet mask]
        gateway    [gateway]
        dns-nameservers [nameserver 1] [nameserver 2]
-----------------------------------------------------------------------
```



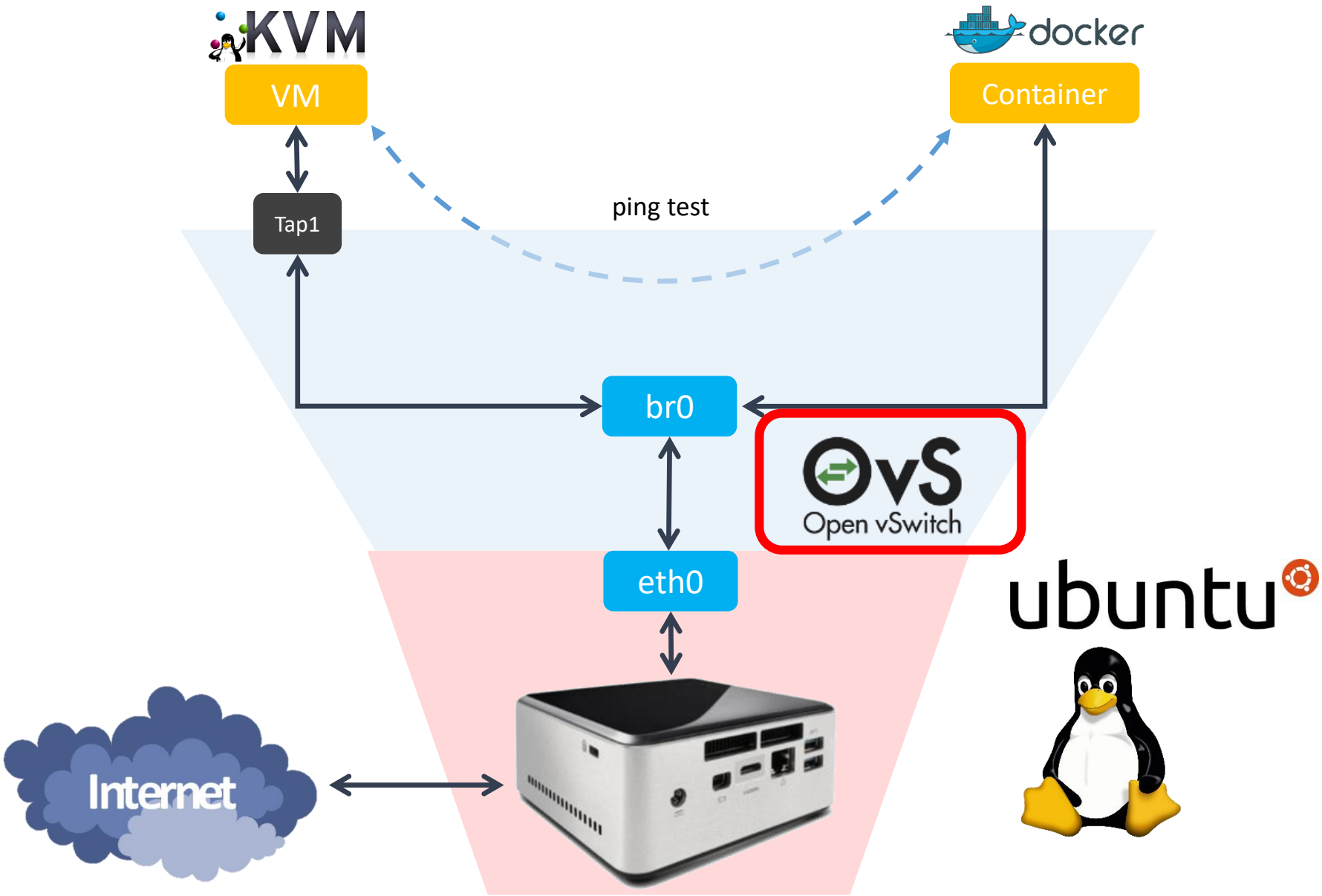

```
$sudo ifdown eth0
$sudo ifup eth0
```



NUC internet works!



#3 - NUC: OVS installation



#3 - NUC: OVS installation

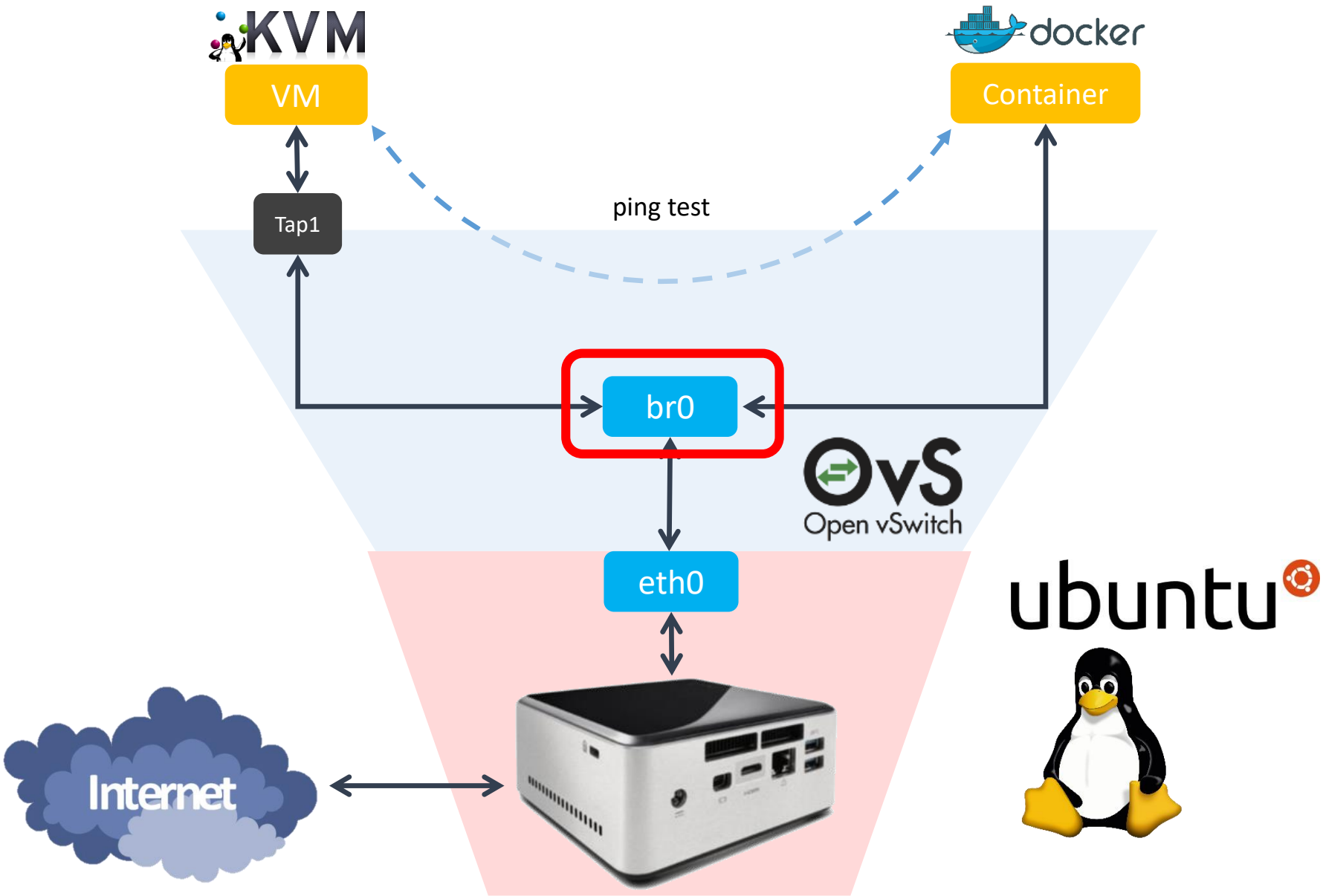
- Update installation of OVS package

Update index information of Open vSwitch package.
Install a Open vSwitch package, openvswitch-switch.
Other dependencies are automatically installed.

$sudo apt-get update
$sudo apt-get install openvswitch-switch

```
tein@SmartXCIServer:~$ sudo apt-get install openvswitch-
openvswitch-common            openvswitch-ipsec
openvswitch-controller        openvswitch-pki
openvswitch-datapath-dkms     openvswitch-switch
openvswitch-datapath-source   openvswitch-test
openvswitch-dbg
```

#4 - NUC: Network Configuration (2/4)



#4 - NUC: Network Configuration (2/4)

- Register OVS br0 in NUC

Modify network interface configuration.

$sudo vi /etc/network/interfaces

---------- /etc/network/interfaces ----------

```
# The primary network interface
auto eth0                    # Append this line
iface eth0 inet manual  # Append this line

auto (eth0->)br0
iface (eth0->)br0 inet static
…
```

Some NUC have different Interface name.
So you need to check your NUC's interface name using 'ifconfig' command.

Don't reboot or issue "ifup" or "ifdown" command until the bridge operation is completed.

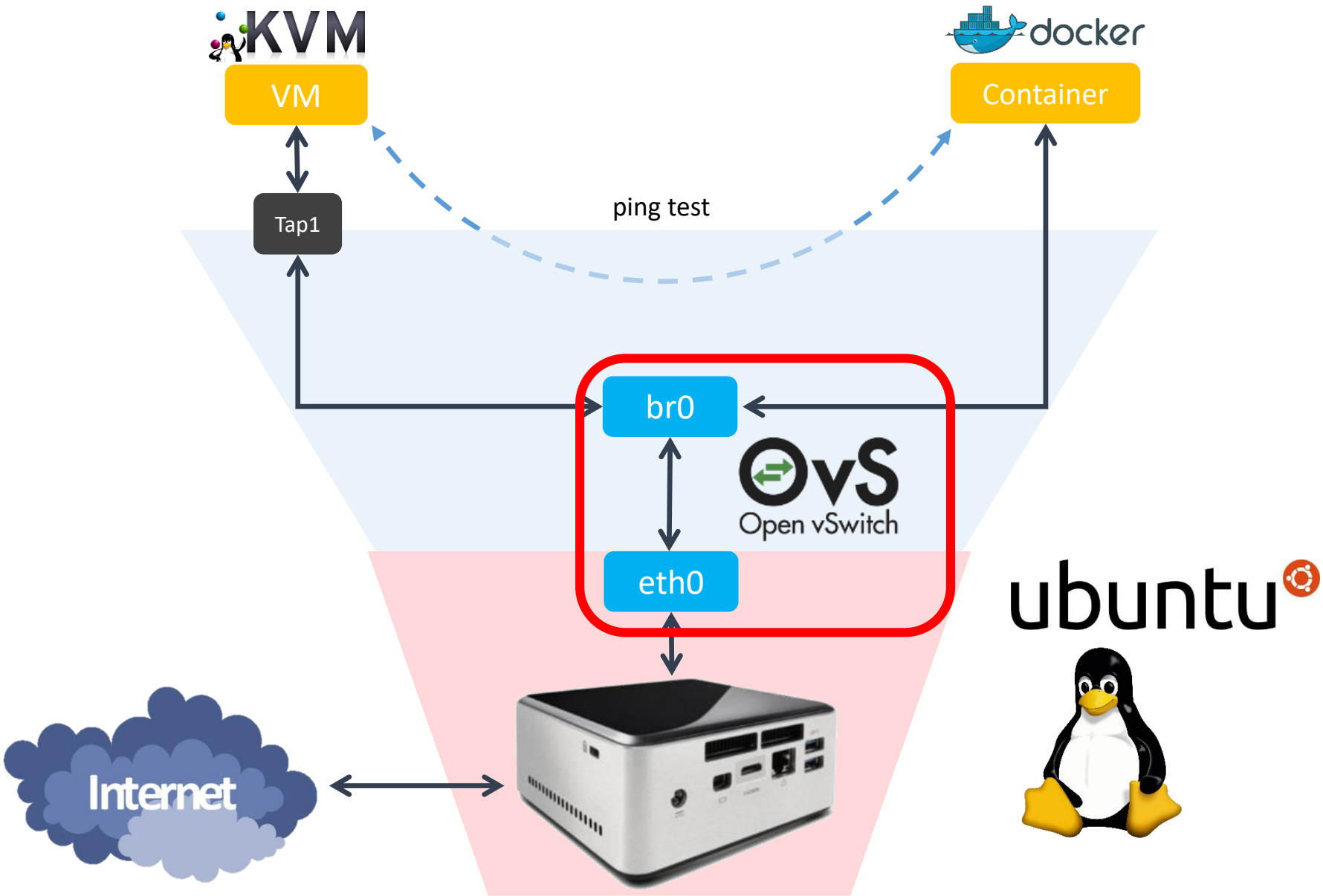
before

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
Iface eth0 inet static
        address 123.45.67.89
        netmask 255.255.255.0
        network 123.45.67.0
        broadcast 123.45.67.255
        gateway 123.45.67.1
        dns-nameservers 8.8.8.8
```

After

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto br0
Iface br0 inet static
        address 123.45.67.89
        netmask 255.255.255.0
        network 123.45.67.0
        broadcast 123.45.67.255
        gateway 123.45.67.1
        dns-nameservers 8.8.8.8
```

#5 - NUC: Network Configuration (3/4)



#5 - NUC: Network Configuration (3/4)

- Connect OVS br0 and NUC eth0 via OVS

$sudo OVS-vsctl add-br br0

```
nuc@nuc:~$
nuc@nuc:~$ sudo su -
[sudo] password for nuc:
root@nuc:~# ovs-vsctl show
3bb93923-3eac-420a-9da9-9143aff14209
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.0.2"
```

#5 - NUC: Network Configuration (3/4)

- Connect OVS br0 and NUC eth0 via OVS

$sudo OVS-vsctl add-port br0 eth0

```
root@nuc:~# ifconfig
br0       Link encap:Ethernet  HWaddr 86:f9:ed:3c:74:42
          inet addr:210.125.      Bcast:210.125.84.255  Mask:255.255.255.0
          inet6 addr: fe80::fccc:4fff:fe23:4e1c/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:648 (648.0 B)

em1       Link encap:Ethernet  HWaddr ec:a8:6b:fb:a2:09
          inet addr:210.125.      Bcast:210.125.84.255  Mask:255.255.255.0
          inet6 addr: fe80::eea8:6bff:fefb:a209/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10899 errors:0 dropped:0 overruns:0 frame:0
          TX packets:566 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3485825 (3.4 MB)  TX bytes:78389 (78.3 KB)
          Interrupt:20 Memory:f7c00000-f7c20000

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:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:366 (366.0 B)  TX bytes:366 (366.0 B)
```

eth0 인터페이스가 global 영역에서 bridge 영역으로 이동하기 때문에 연결이 끊김

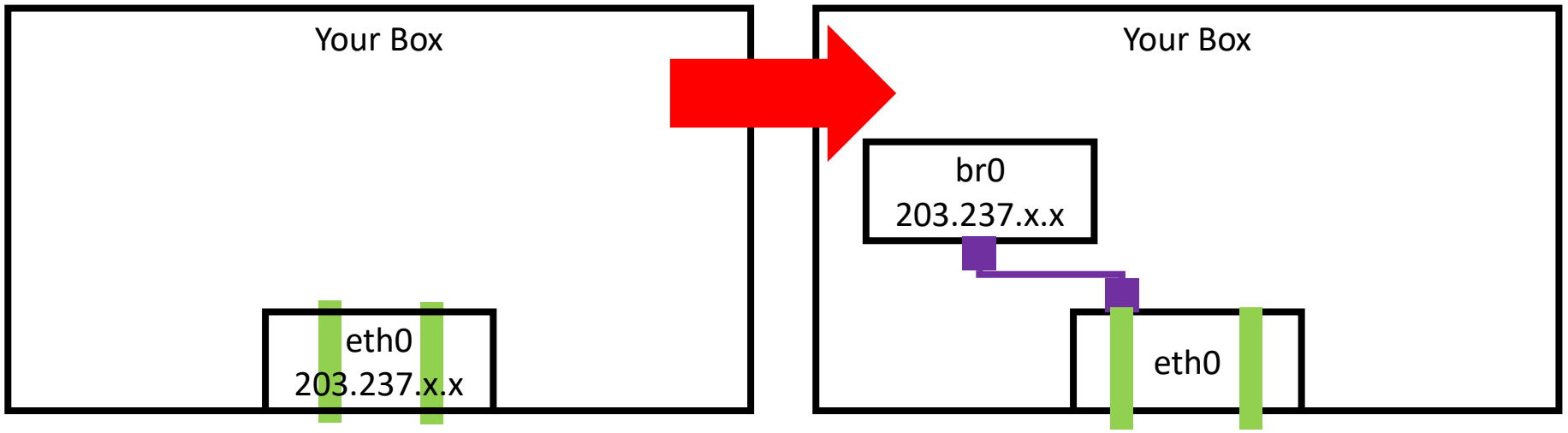
#5 - NUC: Network Configuration (3/4)

- Connect OVS br0 and NUC eth0 via OVS

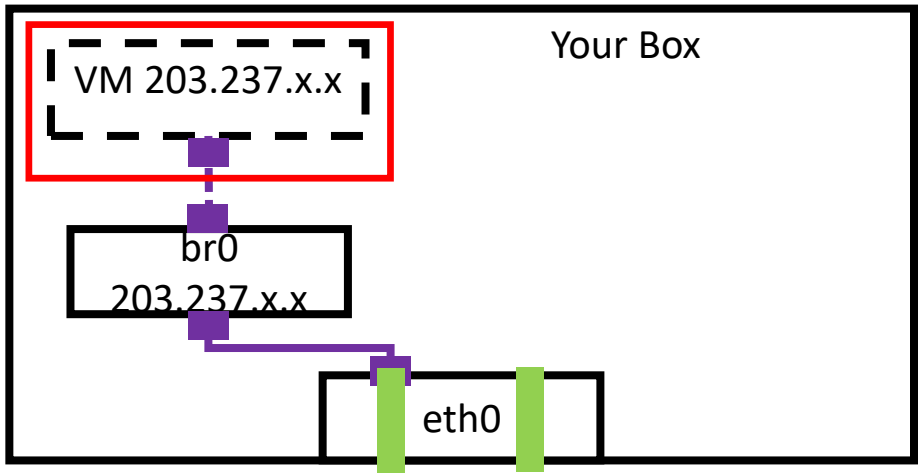
```
$sudo ip route flush table main          // Removing routing table of OS
$sudo ip addr flush dev eth0             // Removing the IP address of eth0
$sudo ip addr flush dev br0              // Removing the IP address of br0
$sudo ifup br0                           // Turning on “br0” interface
$sudo ip link set eth0 up                // Turning on “eth0” interface
```

#5 - NUC: Network Configuration (3/4)

- Connect OVS br0 and NUC eth0 via OVS

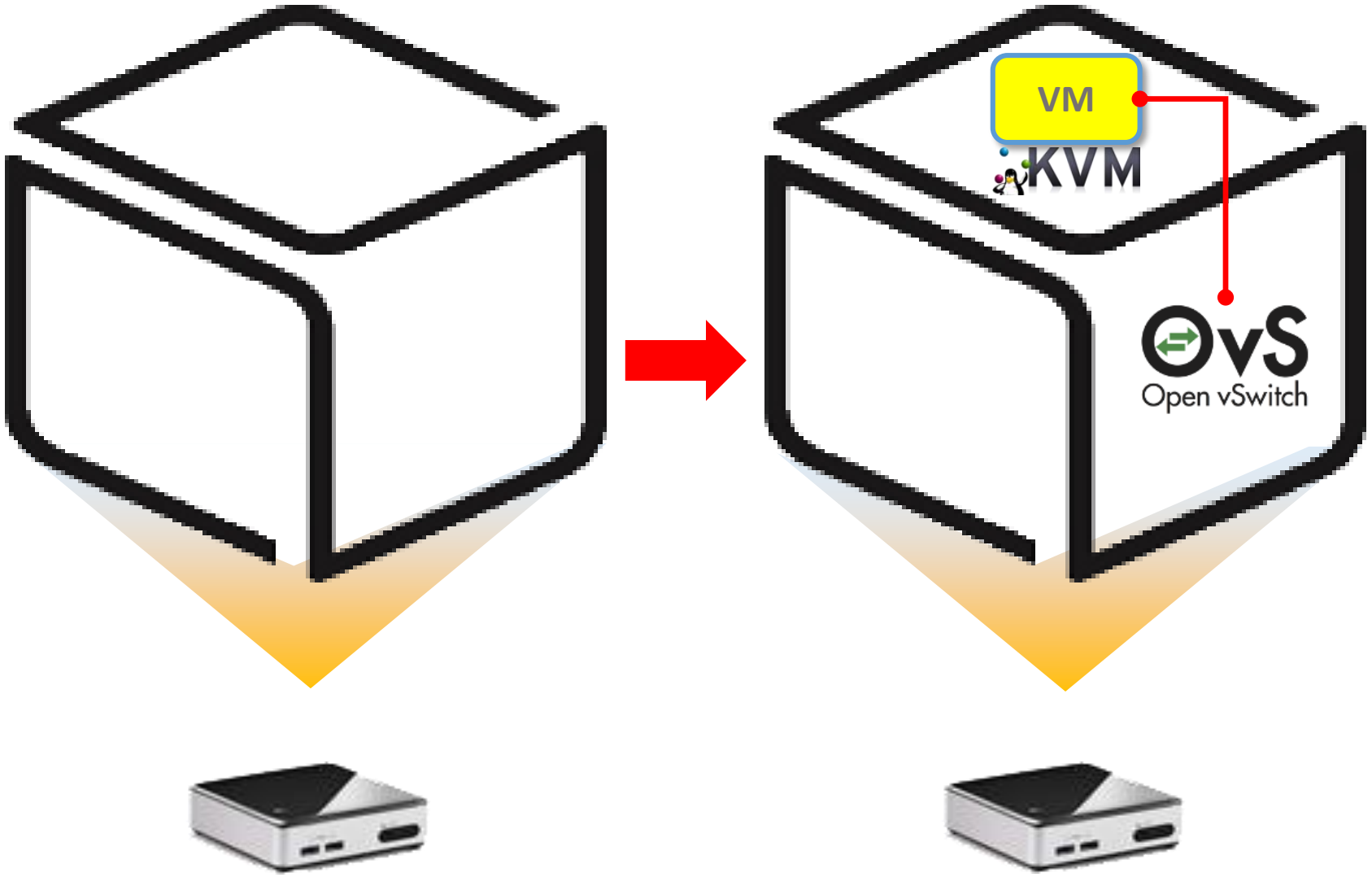


Note: After OVS setting,
later we will create VM
and connect it.

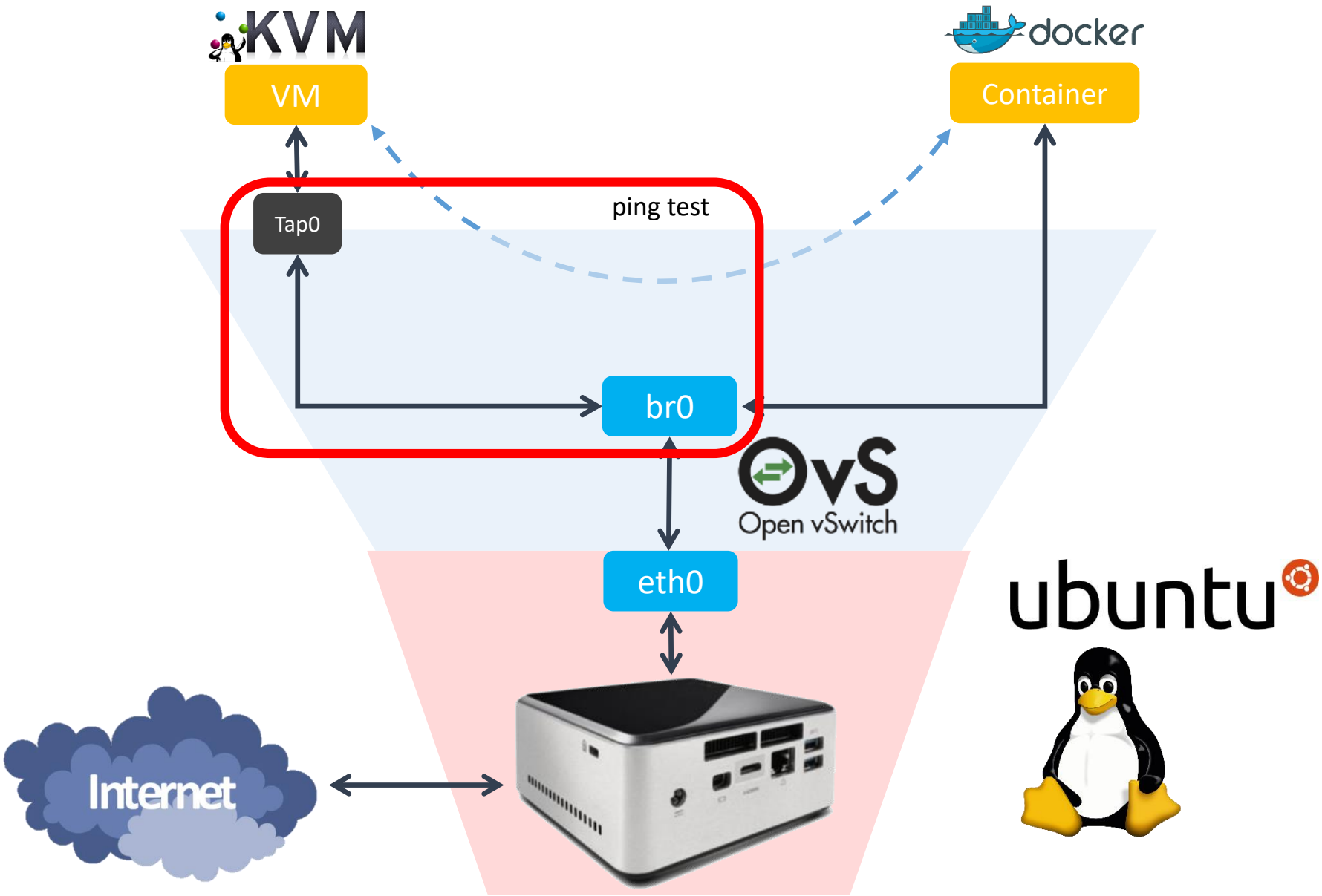


KVM-based VM connected via OVS

- Goal of this section



#6 - NUC: Network Configuration (4/4)



#6 - NUC: Network Configuration (4/4)

- Connect OVS tap0 & br0 through OVS

Let's make a tap interface and attach it to your VM.

```
$sudo vi /etc/network/interfaces

----------------- /etc/network/interfaces -----------------
…
(Append the lines below to the config file)

auto tap0
iface tap0 inet manual
        pre-up ip tuntap add tap0 mode tap
        up ip link set dev tap0 up
        post-down ip link del dev tap0
```

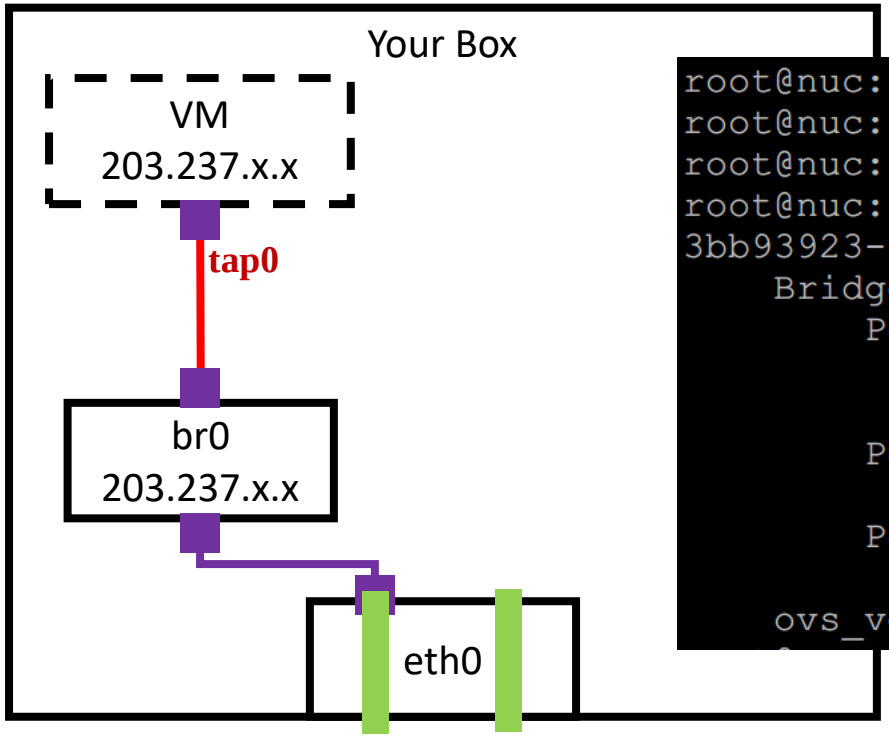
#6 - NUC: Network Configuration (4/4)

-Connect tap0, br0 through OVS

Turn on the tap interface and attach it to br0.

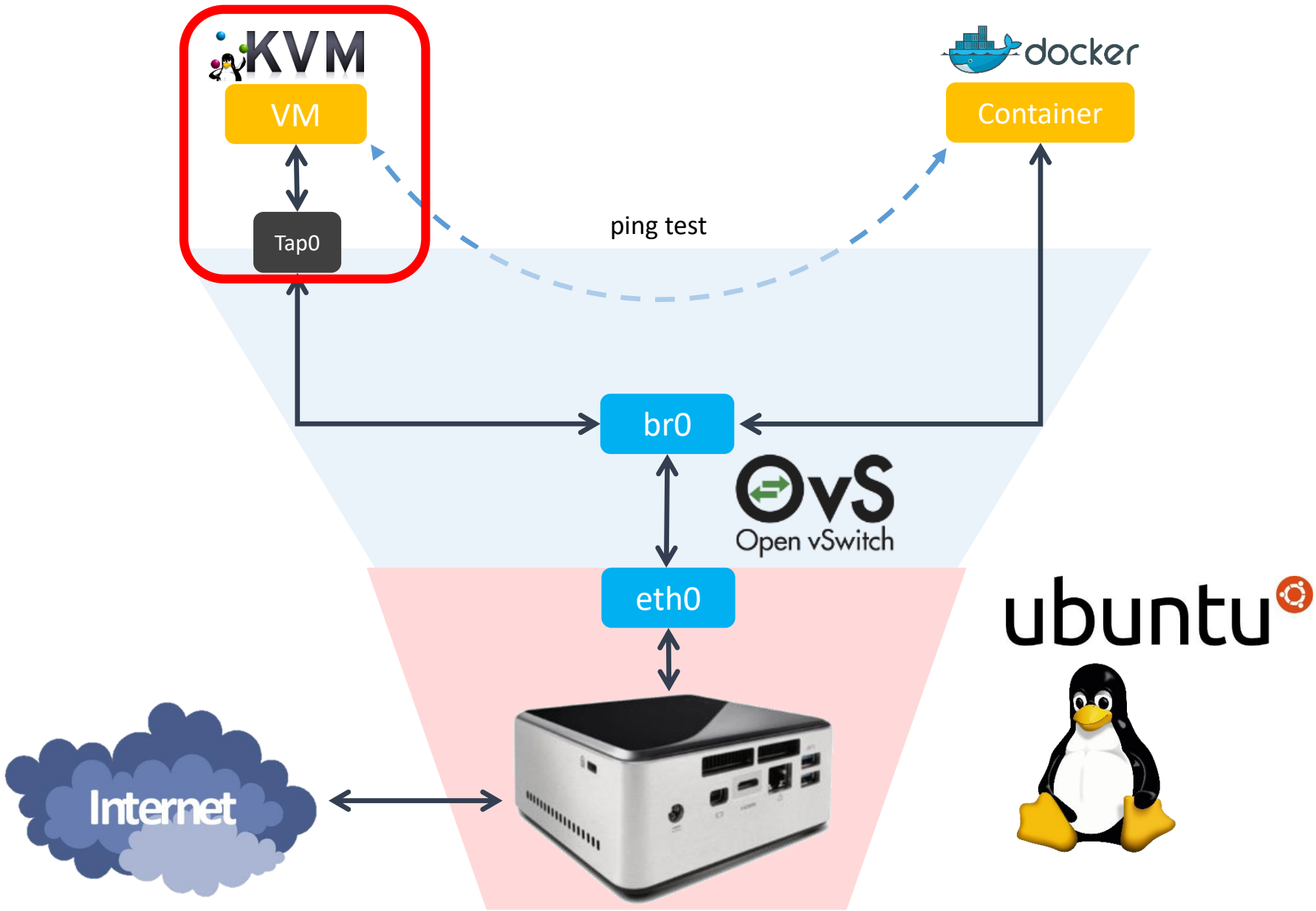
```
$sudo ifup tap0
$sudo OVS-vsctl add-port br0 tap0      // Turn on and attach to br0
```

We should make VM attaching tap0. You can think this tap as a NIC of VM.




```
root@nuc:~# ip tuntap add mode tap vport_vFunction
root@nuc:~# ifconfig vport_vFunction up
root@nuc:~# ovs-vsctl add-port br0 vport_vFunction
root@nuc:~# ovs-vsctl show
3bb93923-3eac-420a-9da9-9143aff14209
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "em1"
            Interface "em1"
        Port vport_vFunction  → [tap_name]
            Interface vport_vFunction
    ovs_version: "2.0.2"
```

#7 - NUC: Making VM with KVM



#7 - NUC: Making VM with KVM

-Install dependency to upgrade KVM

Install dependency & download Ubuntu 16.04.4 64bit server image.

```
$sudo apt-get install qemu-kvm libvirt-bin          //upgrade KVM
                                                    //qemu is open-source emulator

$wget http://old-releases.ubuntu.com/releases/16.04.4/ubuntu-16.04.4-server-amd64.iso
```

Now we are ready to make VM. So continue the setting.

#7 - NUC: Making VM with KVM

-Prepare for Ubuntu VM

Make a VM image.

$sudo qemu-img create [img_name].img -f qcow2 [storage_capacity]

$sudo qemu-img create vFunction20.img –f qcow2 10G

Result..

```
nuc@nuc:~/VMs$ sudo qemu-img create vFunction20.img -f qcow2 10G
Formatting 'vFunction20.img', fmt=qcow2 size=10737418240 encryption=off cluster_size=65536 lazy_refcounts=off
```

Boot VM image from Ubuntu iso file (mac should be different from others).

$sudo kvm -m [memory_capacity] -name [vm_name] -smp cpus=[#cpu],maxcpus= [#maxcpu] -device virtio-net-pci,netdev=net0,mac= [EE:EE:EE:EE:EE:EE] -netdev tap,id=net0,ifname=[tap_name],script=no -boot d [img_name].img -cdrom ubuntu-16.04.4-server-amd64.iso -vnc :[#] –daemonize -monitor telnet:127.0.0.1:3010,server,nowait,ipv4

$ sudo kvm -m 512 -name tt -smp cpus=2,maxcpus=2 -device virtio-net-pci,netdev=net0 -netdev tap,id=net0,ifname=vport_vFunction,script=no -boot d vFunction20.img -cdrom ubuntu-16.04.4-server-amd64.iso -vnc :5 -daemonize -monitor telnet:127.0.0.1:3010,server,nowait,ipv4

Install VNC viewer and see inside of VM

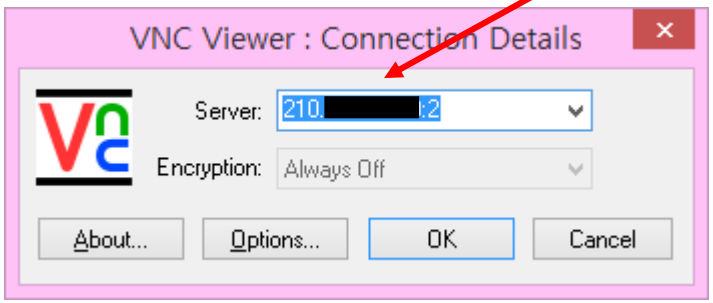
$sudo apt-get install xvnc4viewer
$xvnc4viewer localhost :5

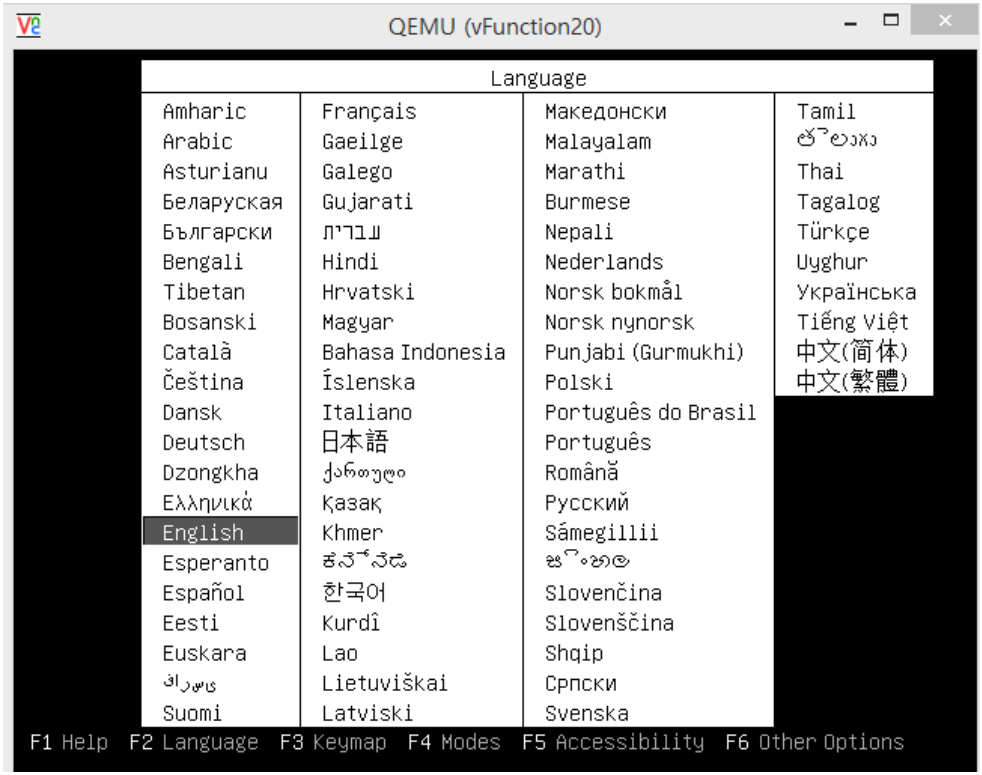
#7 - NUC: Making VM with KVM

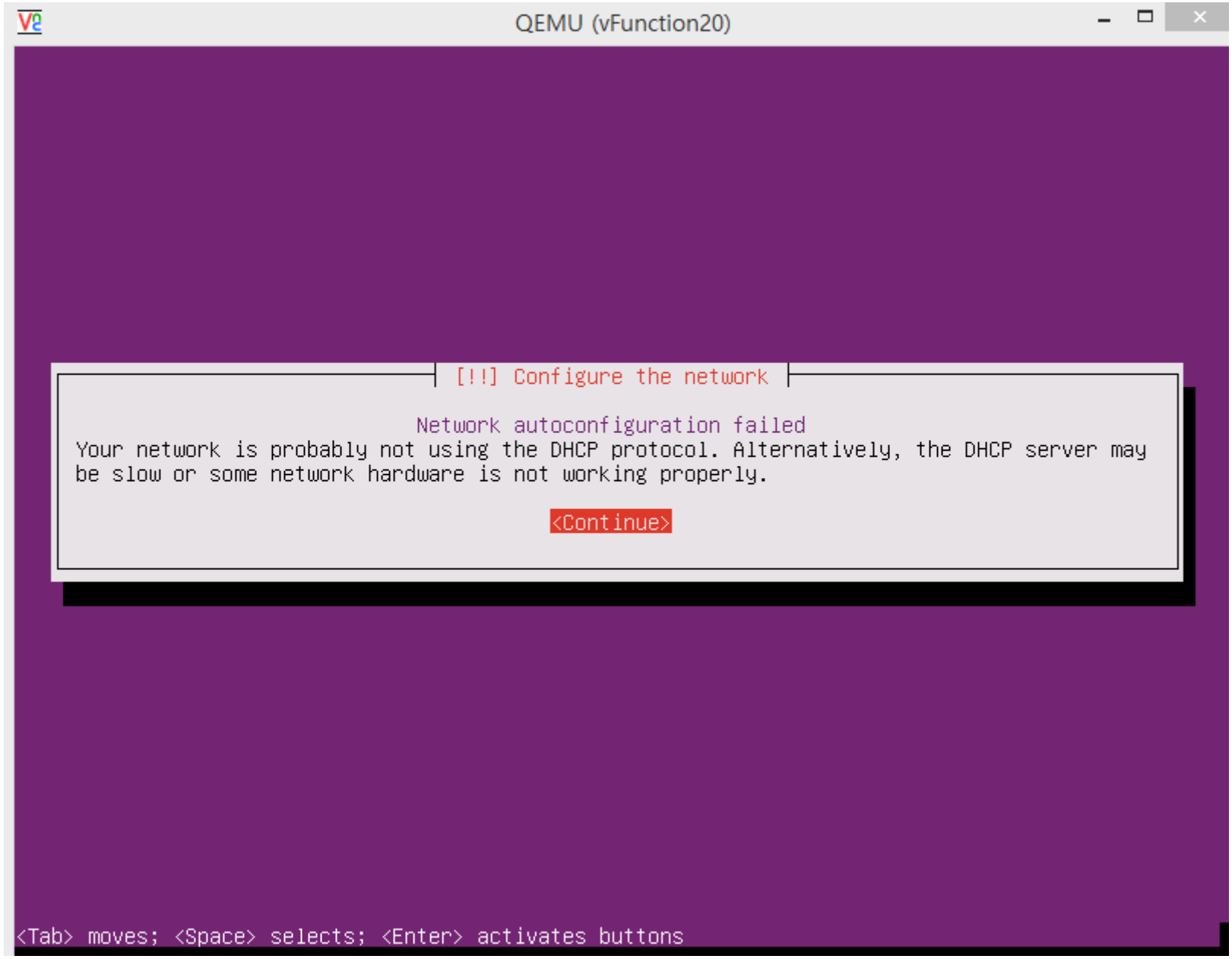
-Install Ubuntu VM

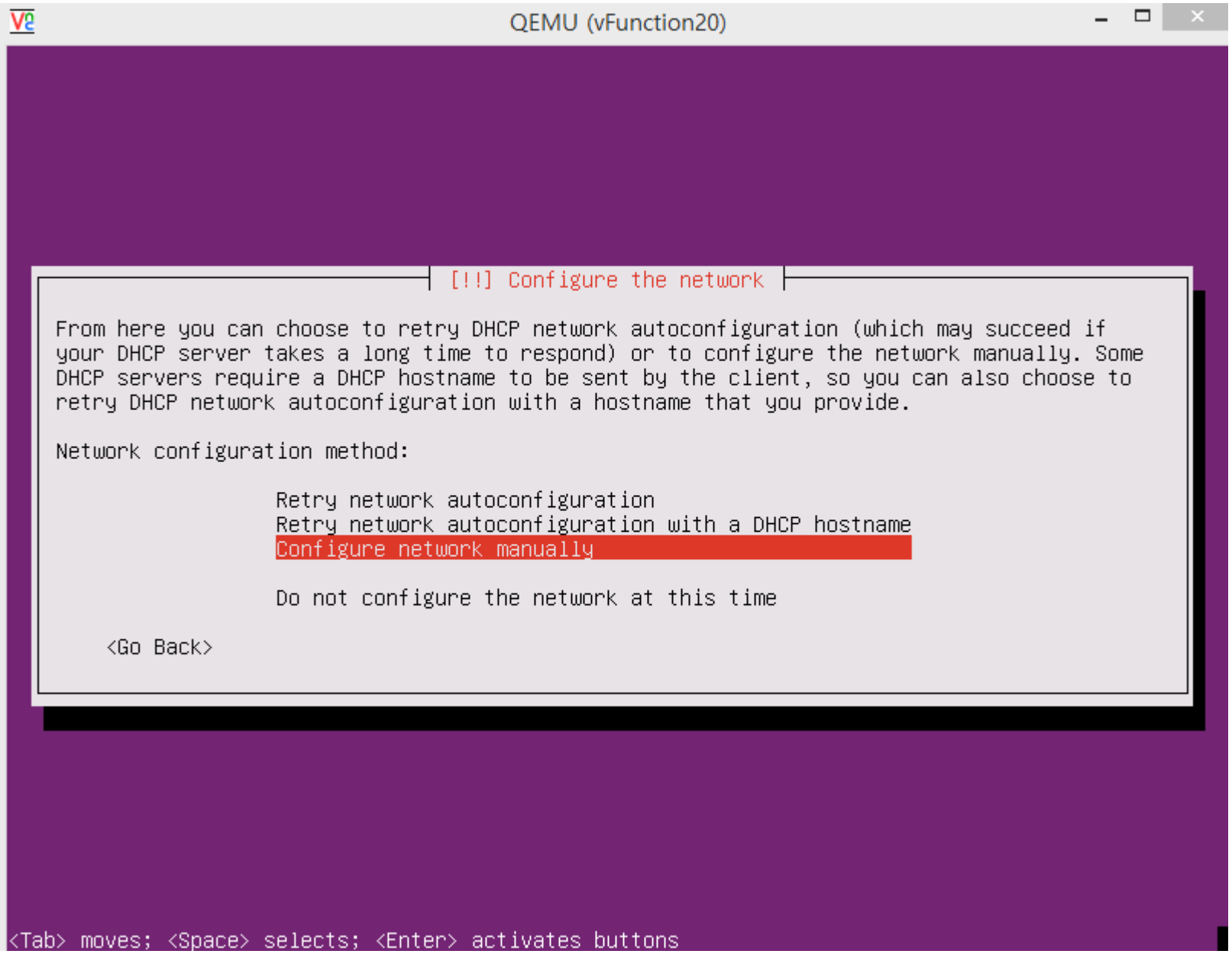
IP address:vnc number

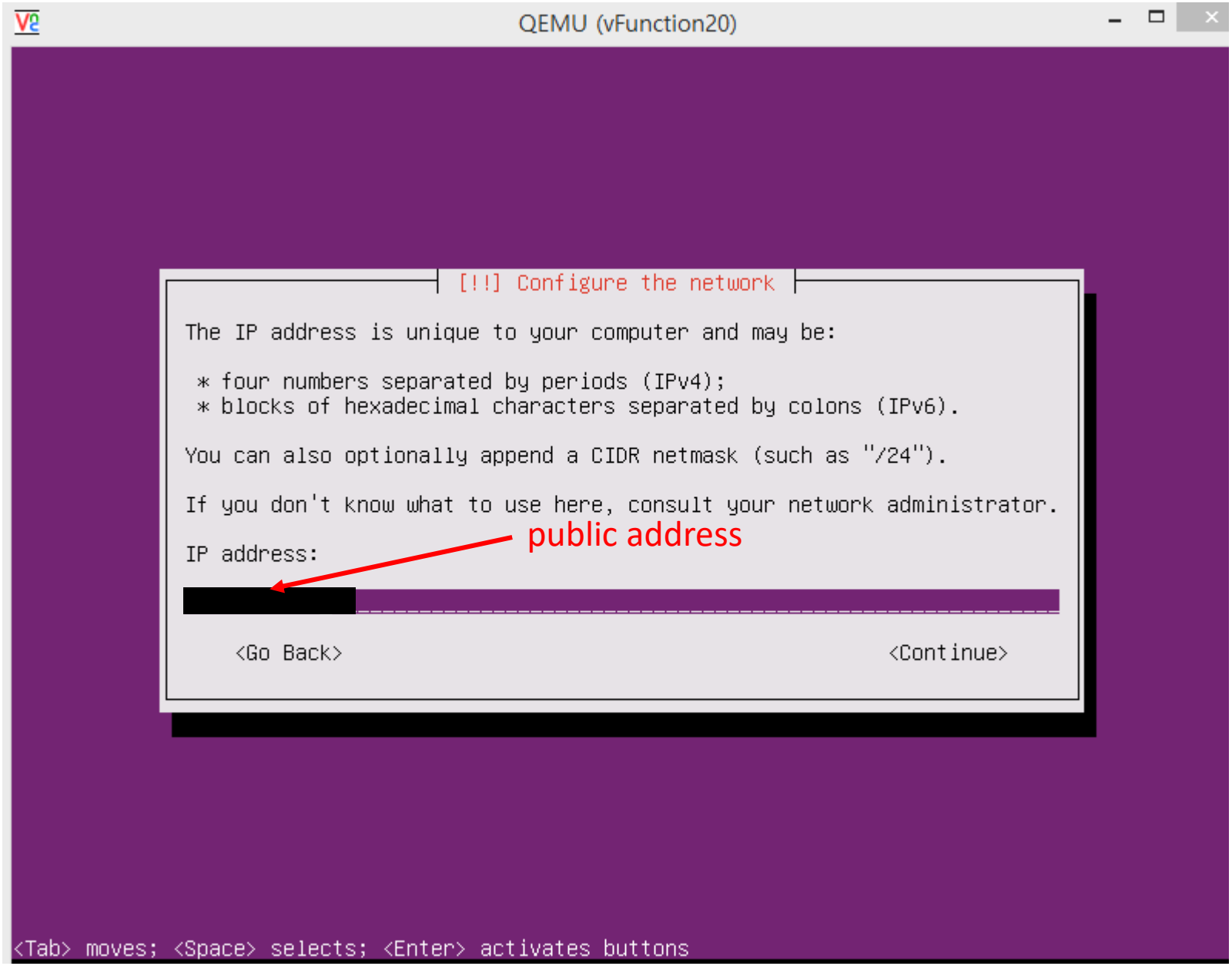
ex) 210.203.x.x:5

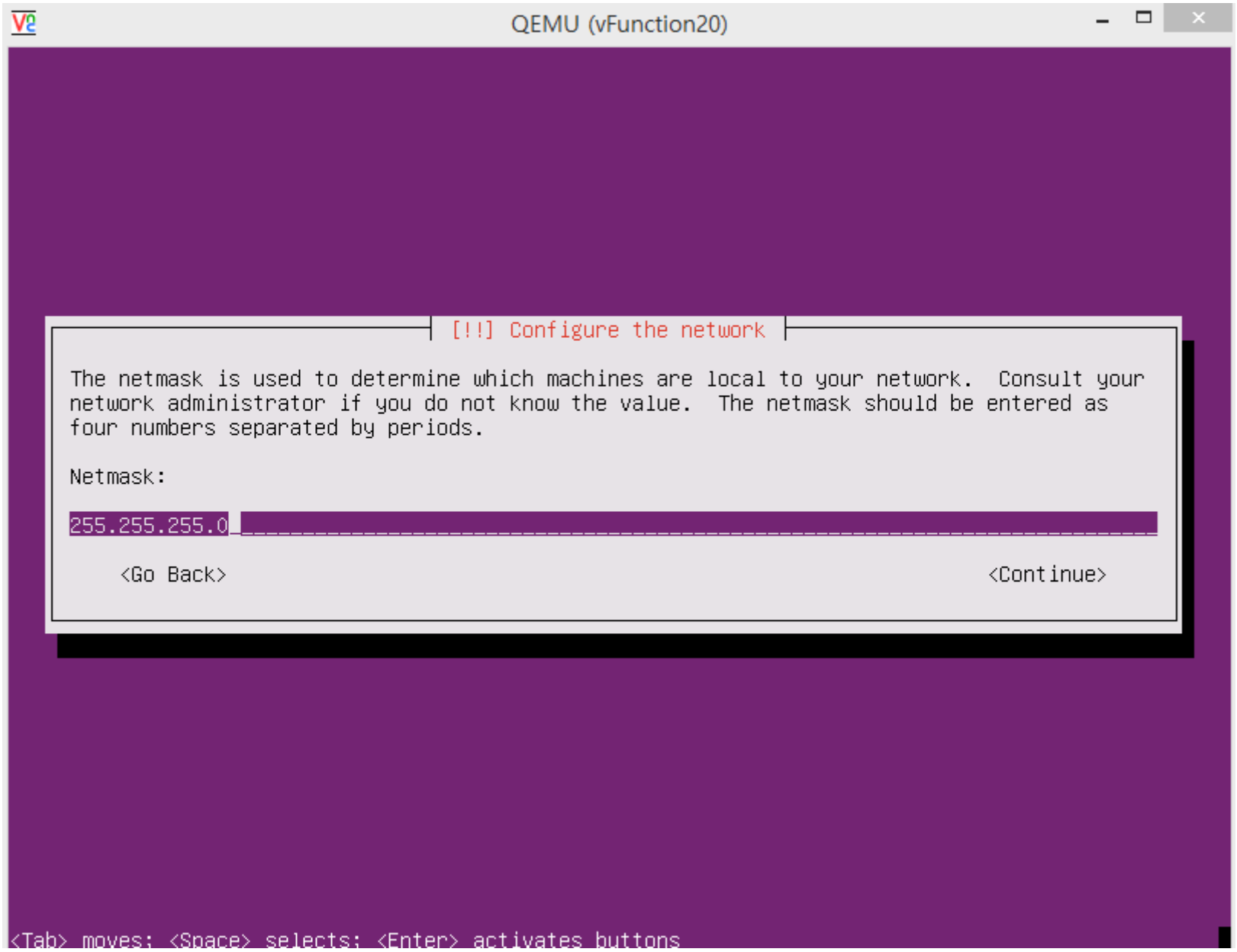


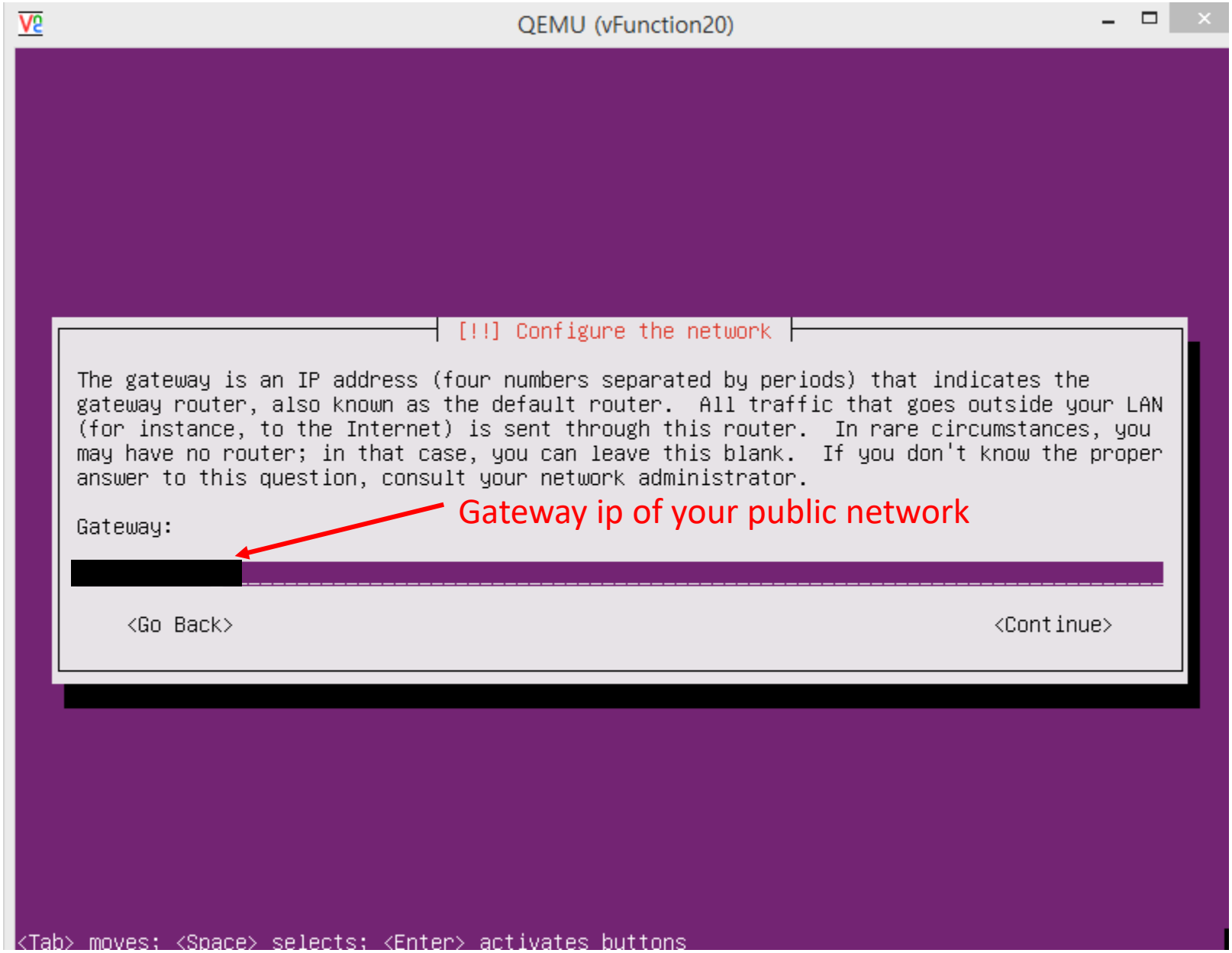


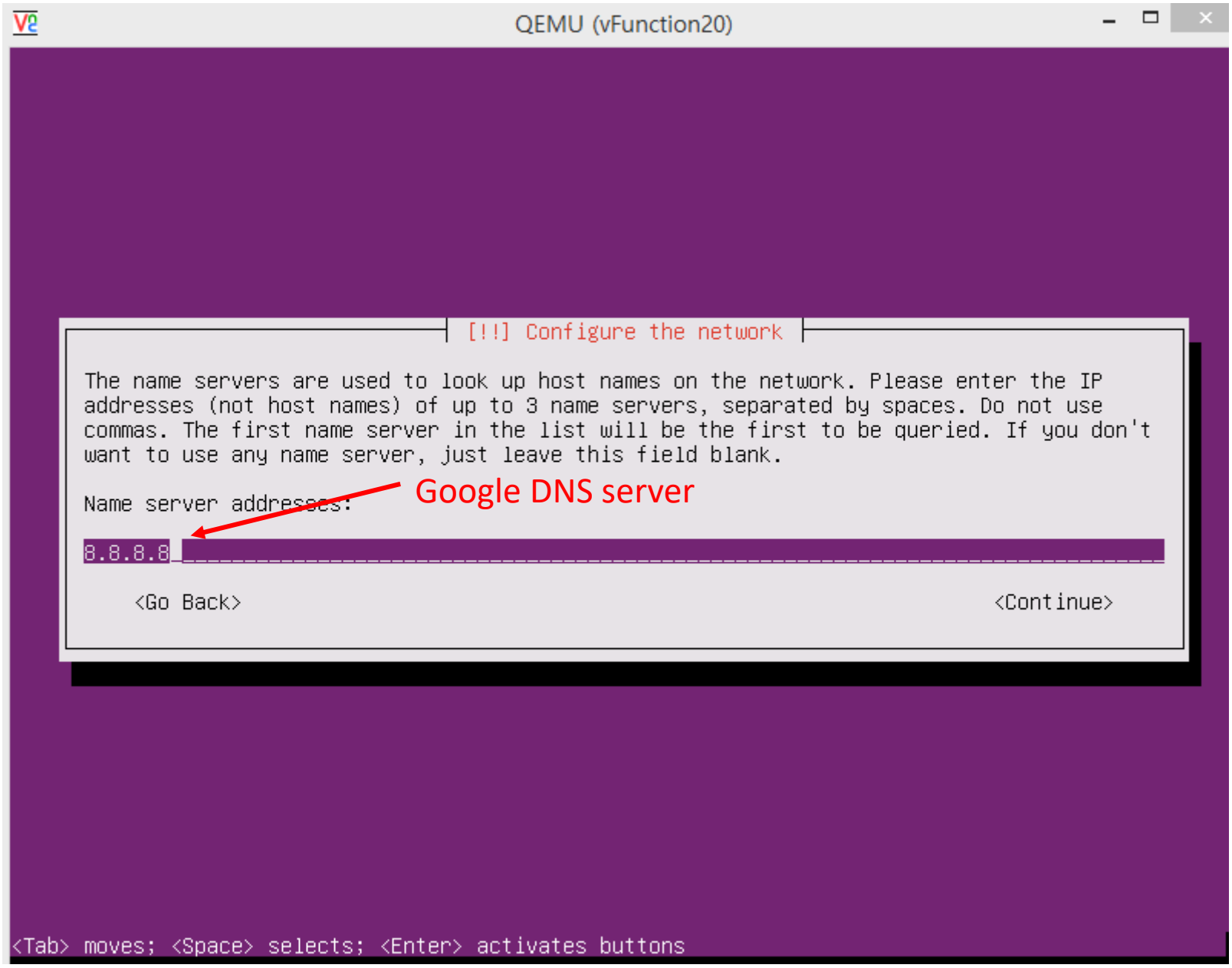

QEMU (vFunction20)
[!!] Configure the network
Network autoconfiguration failed
Your network is probably not using the DHCP protocol. Alternatively, the DHCP server may be slow or some network hardware is not working properly.
<Continue>
<Tab> moves; <Space> selects; <Enter> activates buttons


QEMU (vFunction20)
[!!] Configure the network
From here you can choose to retry DHCP network autoconfiguration (which may succeed if your DHCP server takes a long time to respond) or to configure the network manually. Some DHCP servers require a DHCP hostname to be sent by the client, so you can also choose to retry DHCP network autoconfiguration with a hostname that you provide.
Network configuration method:
Retry network autoconfiguration
Retry network autoconfiguration with a DHCP hostname
Configure network manually
Do not configure the network at this time
<Go Back>
<Tab> moves; <Space> selects; <Enter> activates buttons


QEMU (vFunction20)
[!!] Configure the network
The IP address is unique to your computer and may be:
* four numbers separated by periods (IPv4);
* blocks of hexadecimal characters separated by colons (IPv6).
You can also optionally append a CIDR netmask (such as "/24").
If you don't know what to use here, consult your network administrator.
public address
IP address:
<Go Back>
<Continue>
<Tab> moves; <Space> selects; <Enter> activates buttons


QEMU (vFunction20)
[!!] Configure the network
The netmask is used to determine which machines are local to your network. Consult your network administrator if you do not know the value. The netmask should be entered as four numbers separated by periods.
Netmask:
255.255.255.0
<Go Back>
<Continue>
<Tab> moves; <Space> selects; <Enter> activates buttons


QEMU (vFunction20)
[!!] Configure the network
The gateway is an IP address (four numbers separated by periods) that indicates the gateway router, also known as the default router. All traffic that goes outside your LAN (for instance, to the Internet) is sent through this router. In rare circumstances, you may have no router; in that case, you can leave this blank. If you don't know the proper answer to this question, consult your network administrator.
Gateway:
Gateway ip of your public network
<Go Back>
<Continue>
<Tab> moves; <Space> selects; <Enter> activates buttons


QEMU (vFunction20)
[!!] Configure the network
The name servers are used to look up host names on the network. Please enter the IP addresses (not host names) of up to 3 name servers, separated by spaces. Do not use commas. The first name server in the list will be the first to be queried. If you don't want to use any name server, just leave this field blank.
Name server addresses:
Google DNS server
8.8.8.8
<Go Back>
<Continue>
<Tab> moves; <Space> selects; <Enter> activates buttons

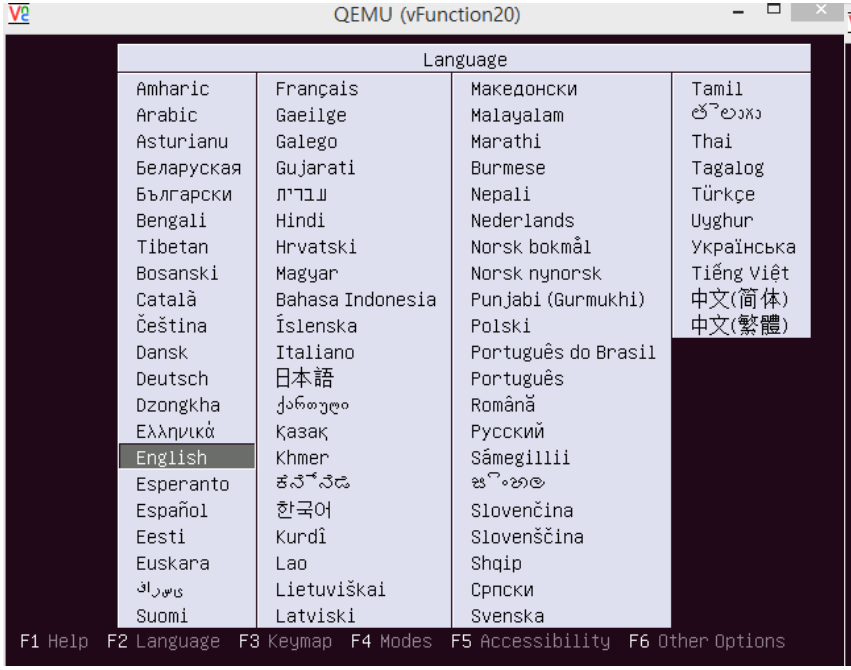
#7 - NUC: Making VM with KVM

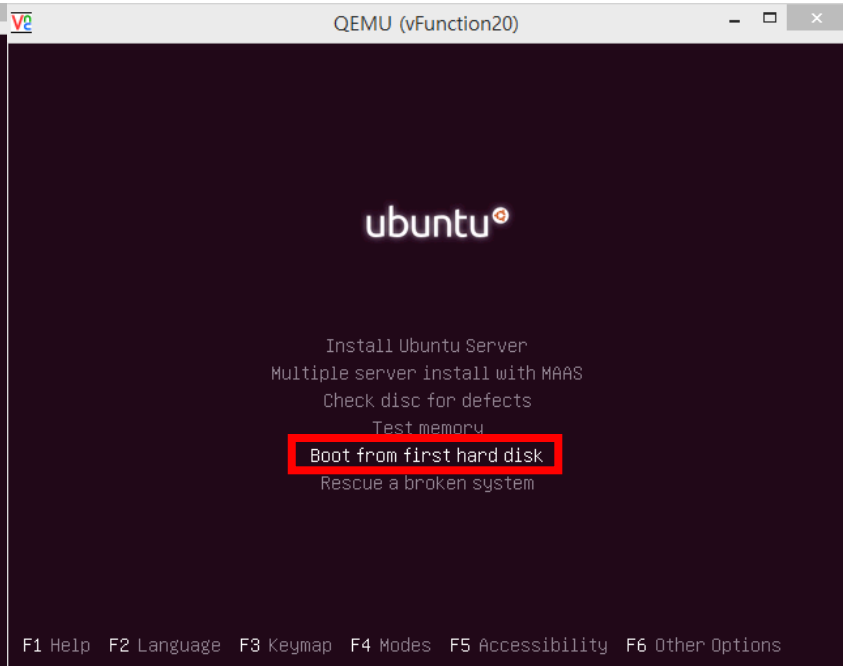
- Eject Ubuntu install image

After installing Ubuntu Linux on the VM….

You need to eject Ubuntu install image before booting to the installed OS

```
$telnet localhost 3010
Trying 127.0.0.1...
Connected to localhost.Escape character is '^]'.

QEMU 0.11.0 monitor - type 'help' for more information
(qemu) eject ide1-cd0
```





Push Esc

#8 - NUC: Booting VM

- VM boot command

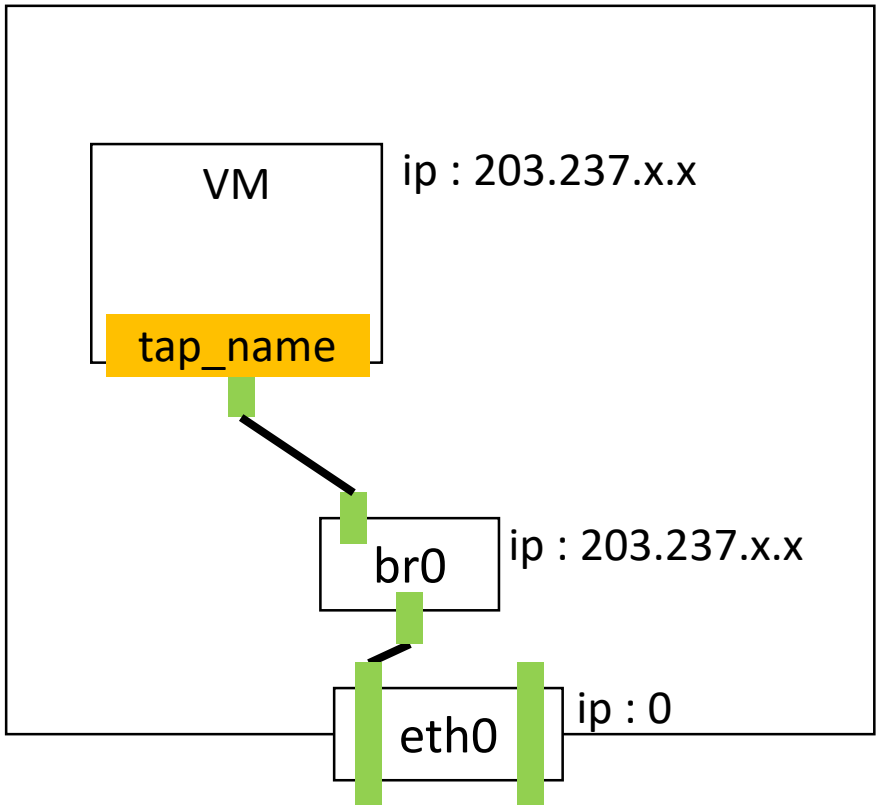
If you want boot VM again (mac should be different from others).

```
$sudo kvm -m [memory capacity] -name [name] -smp cpus=[#cpu],maxcpus= [#maxcpu] -device virtio-net-pci,netdev=net0,mac= [EE:EE:EE:EE:EE:EE] –netdev tap,id=net0,ifname= [tap_name],script=no -boot d [name].img -vnc : [#] -daemonize
```

#9 - OVS connects with KVM

- Check situation

```
root@nuc:~# ovs-vsctl show
3bb93923-3eac-420a-9da9-9143aff14209
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "em1"
            Interface "em1"
        Port vport_vFunction
            Interface vport_vFunction
    ovs_version: "2.0.2"
```



#10 - NUC: Installing ssh in VM

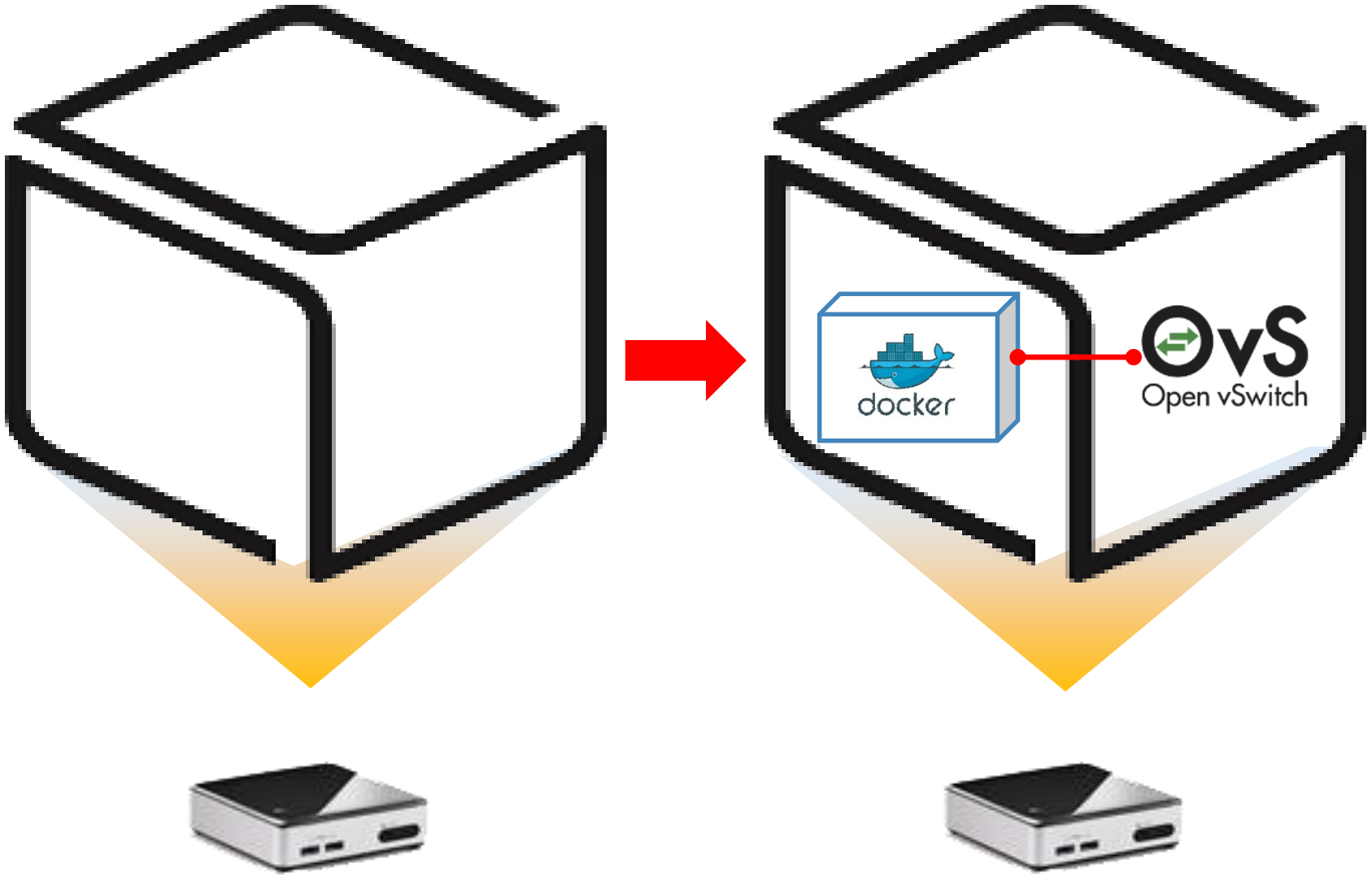
- Don't forget to install ssh in VM

In VMs,

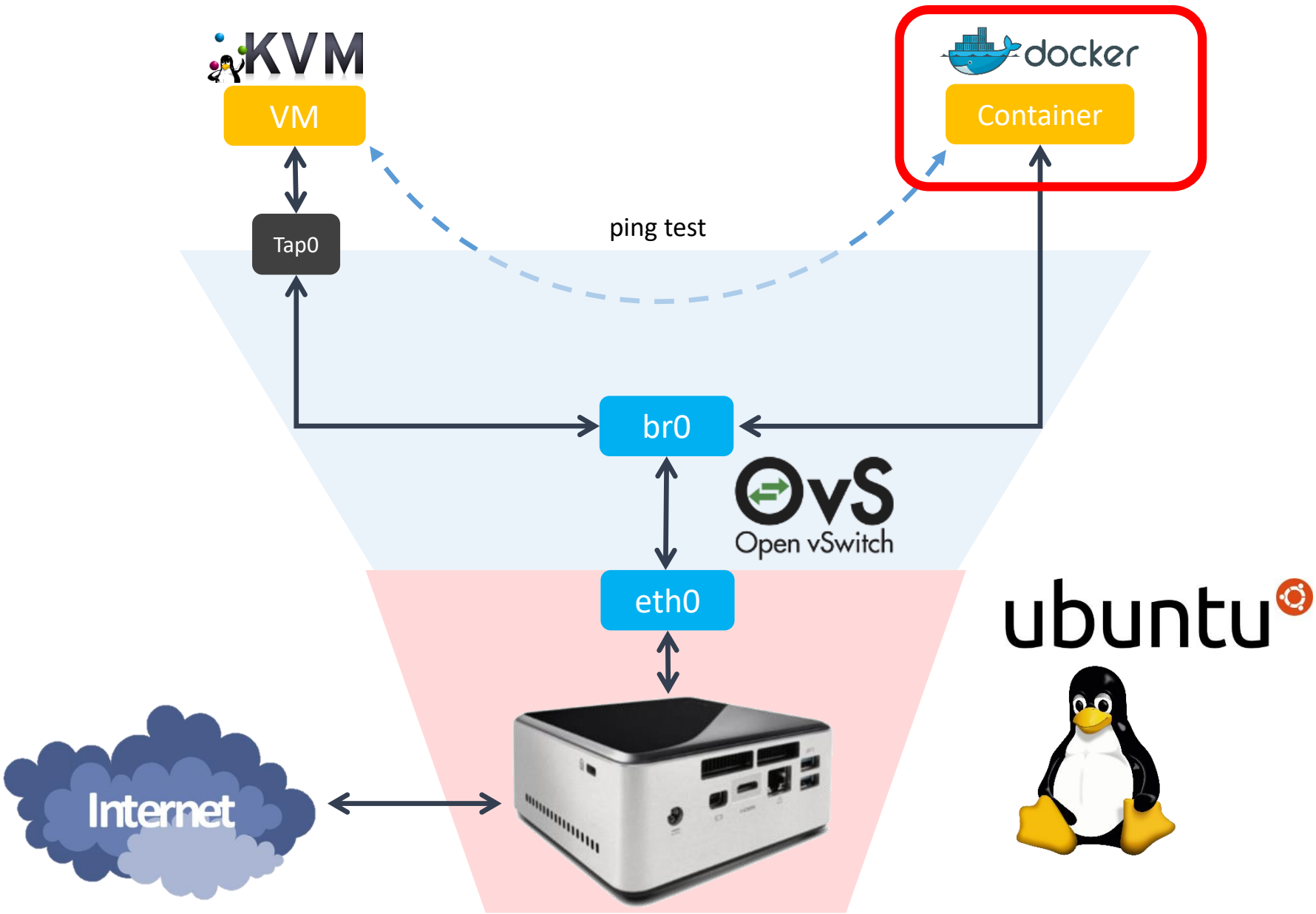
```
$sudo apt-get update
$sudo apt-get install ssh
```

```
nuc@nuc:~$ ssh vbox@192.168.0.3
The authenticity of host '192.168.0.3 (192.168.0.3)' can't be established.
ECDSA key fingerprint is da:c5:2c:53:5a:6f:b4:3c:03:02:04:f3:6a:17:ca:ab.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.3' (ECDSA) to the list of known hosts.
vbox@192.168.0.3's password:
```

Docker Container connected via OVS
- Goal of this section



#11 - Making a Docker Container



#11 - Making a Docker Container

- Docker installation

Docker installation.

```
$sudo wget -qO- https://get.docker.com/ | sh
$sudo systemctl start docker
$sudo adduser [Your_account] docker

(Session restart)

$sudo docker run hello-world
```

reference: http://docs.docker.com/linux/step_one/

#11 - Making a Docker Container

- Docker installation

```
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.
 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 Hub account:
 https://hub.docker.com

For more examples and ideas, visit:
 https://docs.docker.com/userguide/
```

#11 - Making a Docker Container

- Make container

Run docker container.

$sudo docker run -it --net=none --name [container_name] ubuntu /bin/bash

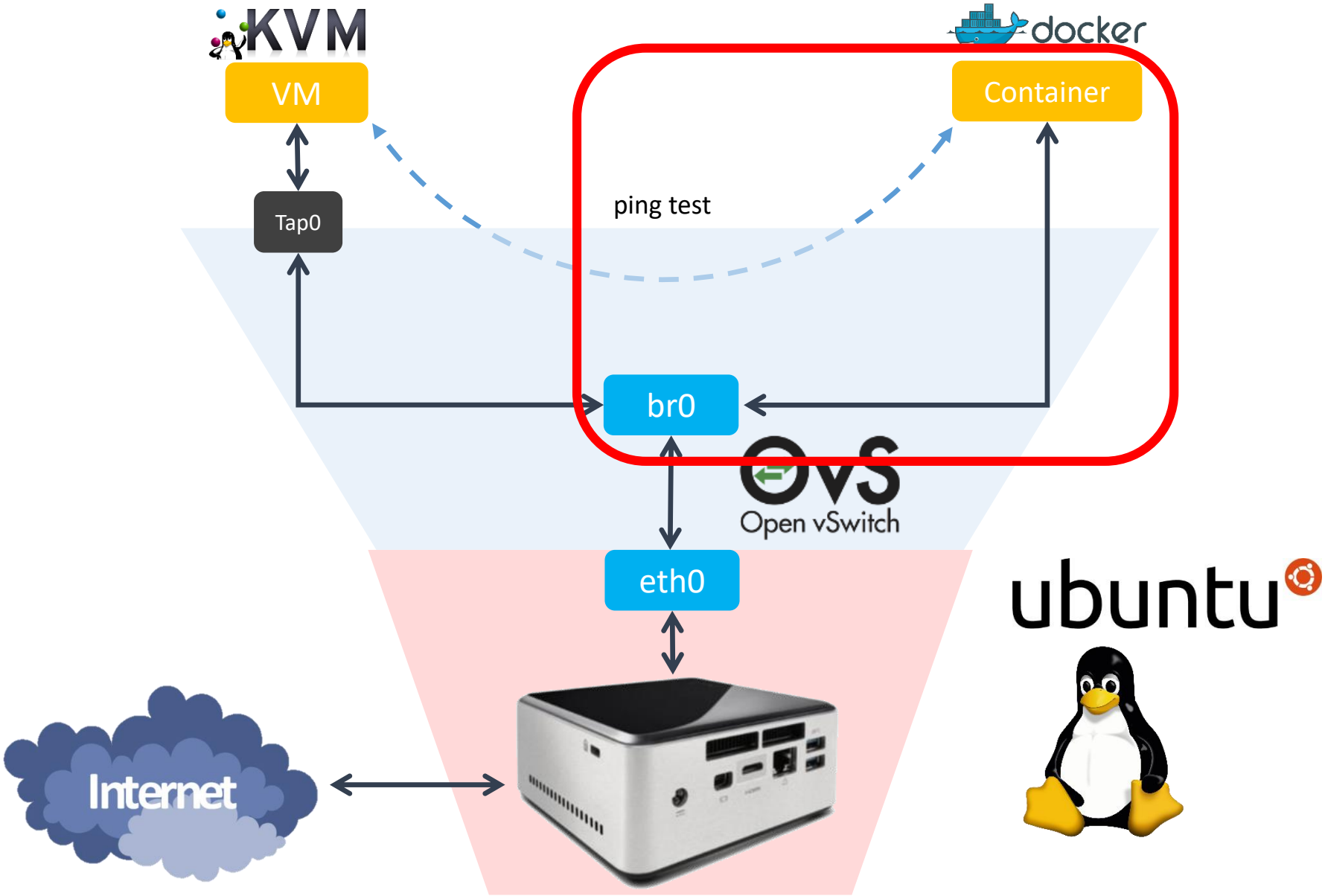
```
nuc@nuc:~$ docker run -it --net=none --name c1 ubuntu /bin/bash
root@8346684676d8:/#
```

I want to make interface that has 203.237.x.x IP address.

※ctrl + p, q → detach docker container
※docker attach [container_name] → get into docker container console

#12 - Connect Docker Container

- Connect with OVS bridge



#12 - Connect docker Container

- Connect with OVS bridge

Install OVS-docker utility in host machine. (Not in inside of Docker container.)

```
$sudo OVS-docker add-port br0 eth0 [containerName] --ipaddress=[IP_address]/24 --gateway=[Gateway_address]


$sudo docker attach [containerName]
#apt-get update
#apt-get install net-tools
#apt-get install iputils-ping
```

#13 – Keep Docker network configuration

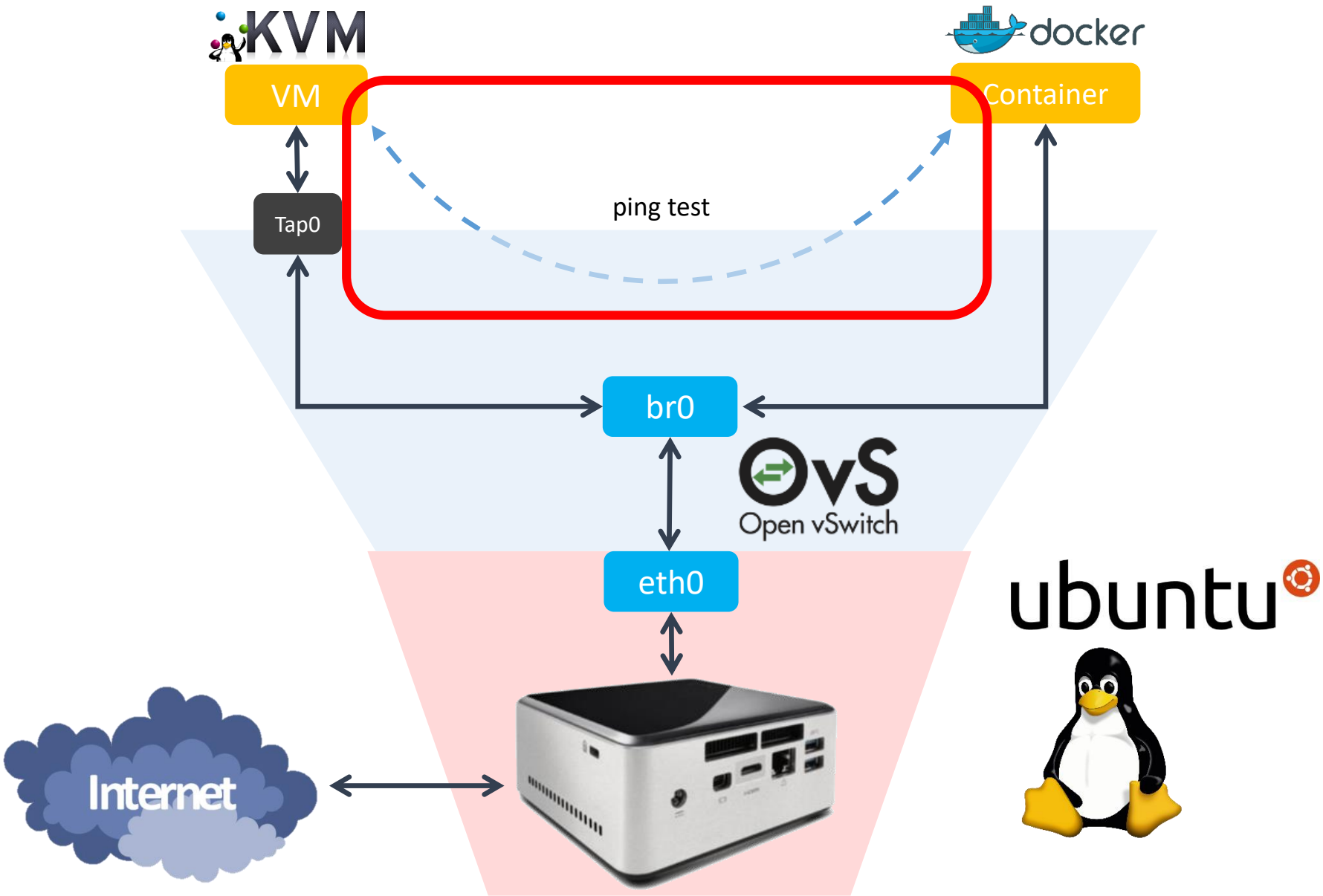
- /etc/rc.local

Modify /etc/rc.local

```
$sudo vi /etc/rc.local

docker start [container_name]
OVS-docker del-port br0 eth0 [containerName]
OVS-docker add-port br0 eth0 [containerName] --ipaddress=[IP_address/24] --gateway=[Gateway_address]
```

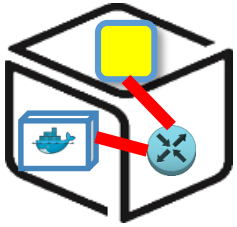
Whenever NUC is rebooted,
network configuration of Docker container is initialized
by executing commands in rc.local

#14 – Check connectivity: VM & Container



#14 - Check connectivity: VM & Container

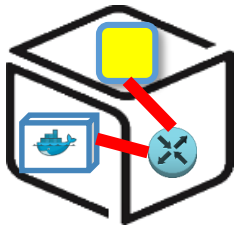
-Check connectivity with ping command



```
root@nuc:/usr/bin# ovs-docker add-port br0 eth0 docker1 --ipaddress=210.125.     /24 --gateway=210.125
root@nuc:/usr/bin# docker attach docker1

root@b8c3bab8204b:/# ifconfig
eth0      Link encap:Ethernet  HWaddr ae:e5:9c:cc:88:b7
          inet addr:210.125        Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::ace5:9cff:fecc:88b7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:120 errors:0 ___pped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8842 (8.8 KB)  TX bytes:648 (648.0 B)

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:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@b8c3bab8204b:/# ping google.com
PING google.com (216.58.221.238) 56(84) bytes of data.
64 bytes from hkg07s21-in-f14.1e100.net (216.58.221.238): icmp_seq=1 ttl=52 time=41.3 ms
64 bytes from hkg07s21-in-f14.1e100.net (216.58.221.238): icmp_seq=2 ttl=52 time=41.3 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.306/41.343/41.380/0.037 ms
```

#14 - Check connectivity: VM & Container

-Check connectivity with ping command



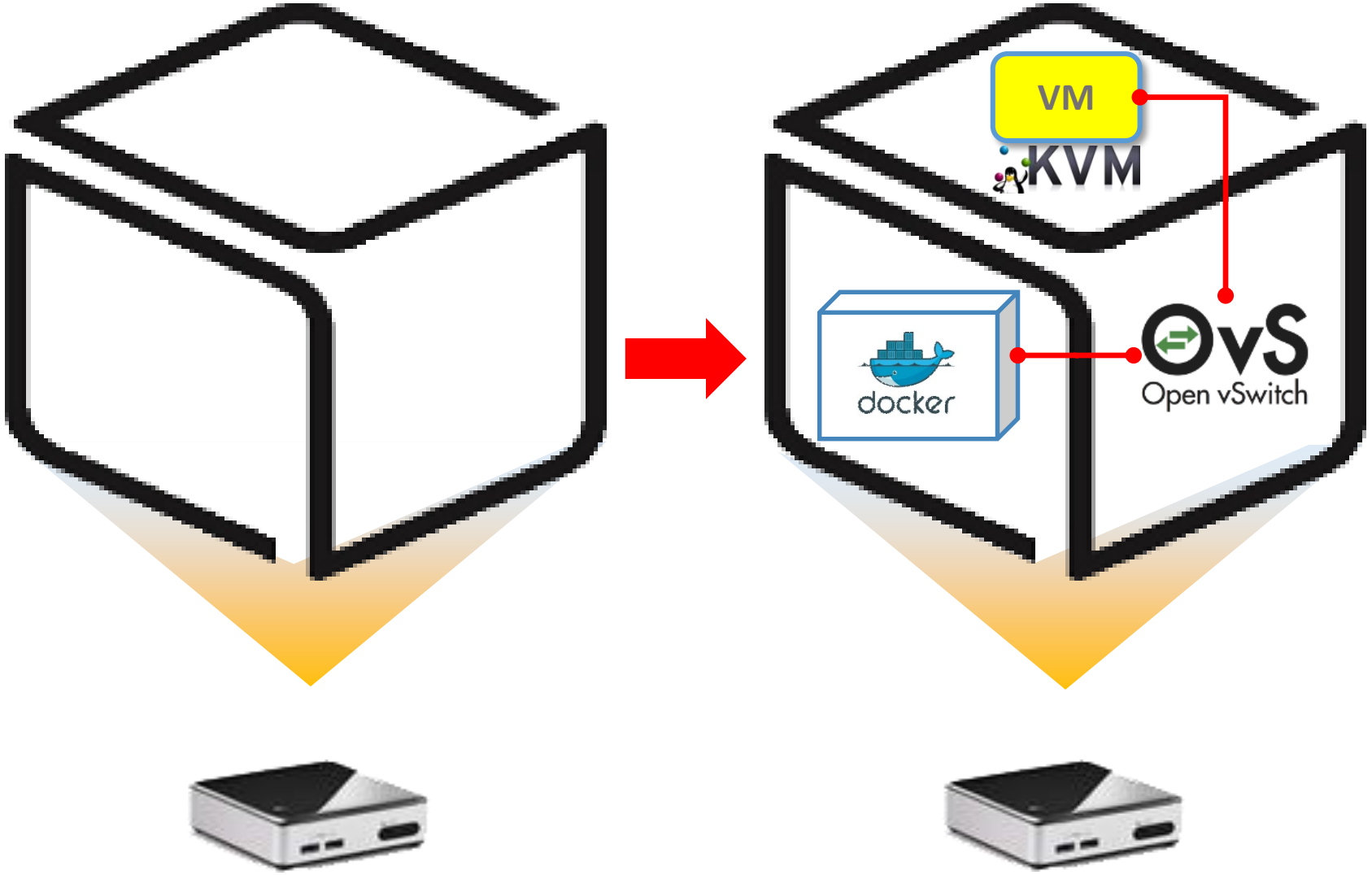
```
root@b8c3bab8204b:/# ifconfig
eth0      Link encap:Ethernet  HWaddr a2:86:d9:c2:33
          inet addr:192.168.     Bcast:0.0.0.0  Mask
          inet6 addr: fe80::a086:d9ff:fec2:337b/64 S
          UP BROADCAST RUNNING MULTICAST  MTU:1500
          RX packets:136 errors:0 dropped:0 overruns
          TX packets:13 errors:0 dropped:0 overruns:
          collisions:0 txqueuelen:1000
          RX bytes:10448 (10.4 KB)  TX bytes:1043 (1

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:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@b8c3bab8204b:/# ping google.com
PING google.com (216.58.221.238) 56(84) bytes of dat
64 bytes from hkg07s21-in-f238.1e100.net (216.58.221
64 bytes from hkg07s21-in-f238.1e100.net (216.58.221
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, t
rtt min/avg/max/mdev = 41.376/41.380/41.384/0.004 ms
root@b8c3bab8204b:/# ping 192.168.
PING 192.168.0.2 (192.168.     56(84) bytes of data.
64 bytes from 192.168.    : icmp_seq=1 ttl=64 time=1.
64 bytes from 192.168.    : icmp_seq=2 ttl=64 time=0.
64 bytes from 192.168.    : icmp_seq=3 ttl=64 time=0.
^C
--- 192.168.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, t
rtt min/avg/max/mdev = 0.651/1.028/1.519/0.365 ms
root@b8c3bab8204b:/#
```

```
vbox@vFunction:~$
vbox@vFunction:~$
vbox@vFunction:~$
vbox@vFunction:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr ee:ee:ee:ee:ee:01
          inet addr:192.168.     Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::ecee:eeff:feee:ee01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18857 errors:0 dropped:0 overruns:0 frame:0
          TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1323453 (1.3 MB)  TX bytes:3507 (3.5 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:38 errors:0 dropped:0 overruns:0 frame:0
          TX packets:38 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3512 (3.5 KB)  TX bytes:3512 (3.5 KB)

vbox@vFunction:~$ ping 192.168
PING 192.168.0.3 (192.168.     56(84) bytes of data.
64 bytes from 192.168.     icmp_seq=1 ttl=64 time=0.872 ms
64 bytes from 192.168.     icmp_seq=2 ttl=64 time=0.590 ms
64 bytes from 192.168.     icmp_seq=3 ttl=64 time=0.585 ms
64 bytes from 192.168.     icmp_seq=4 ttl=64 time=0.573 ms
^C
--- 192.168.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.573/0.655/0.872/0.125 ms
vbox@vFunction:~$
```

Docker container

KVM VM

Box Lab: Final Goal (Recap)



Lab Review

Lab Summary

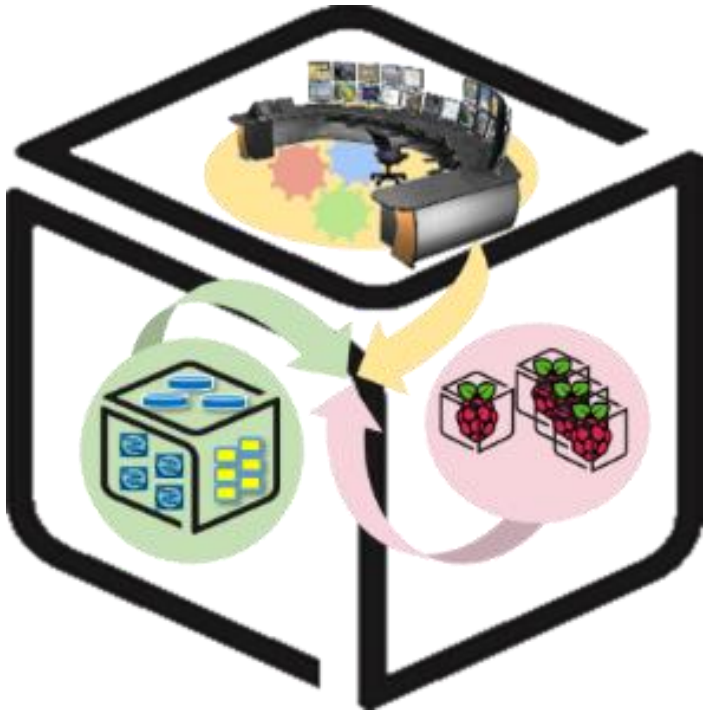
With Box Lab, you have experimented

1. How to install and configure **Linux OS** into Box (i.e., computer).

2. How to install and configure **OVS (Open vSwitch) virtual switch** inside a Linux Box and configure it.

3. How to create **VMs and Docker containers** inside a Linux Box and then **inter-connect** each of them together and to the Internet.

Thank You for Your Attention
Any Questions?



mini@smartx.kr