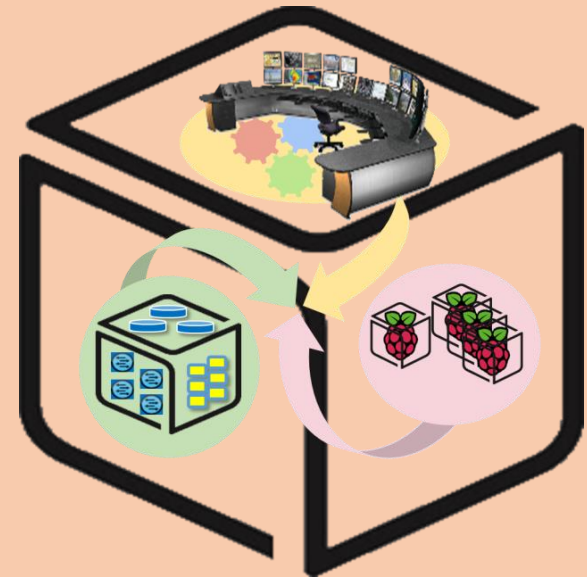


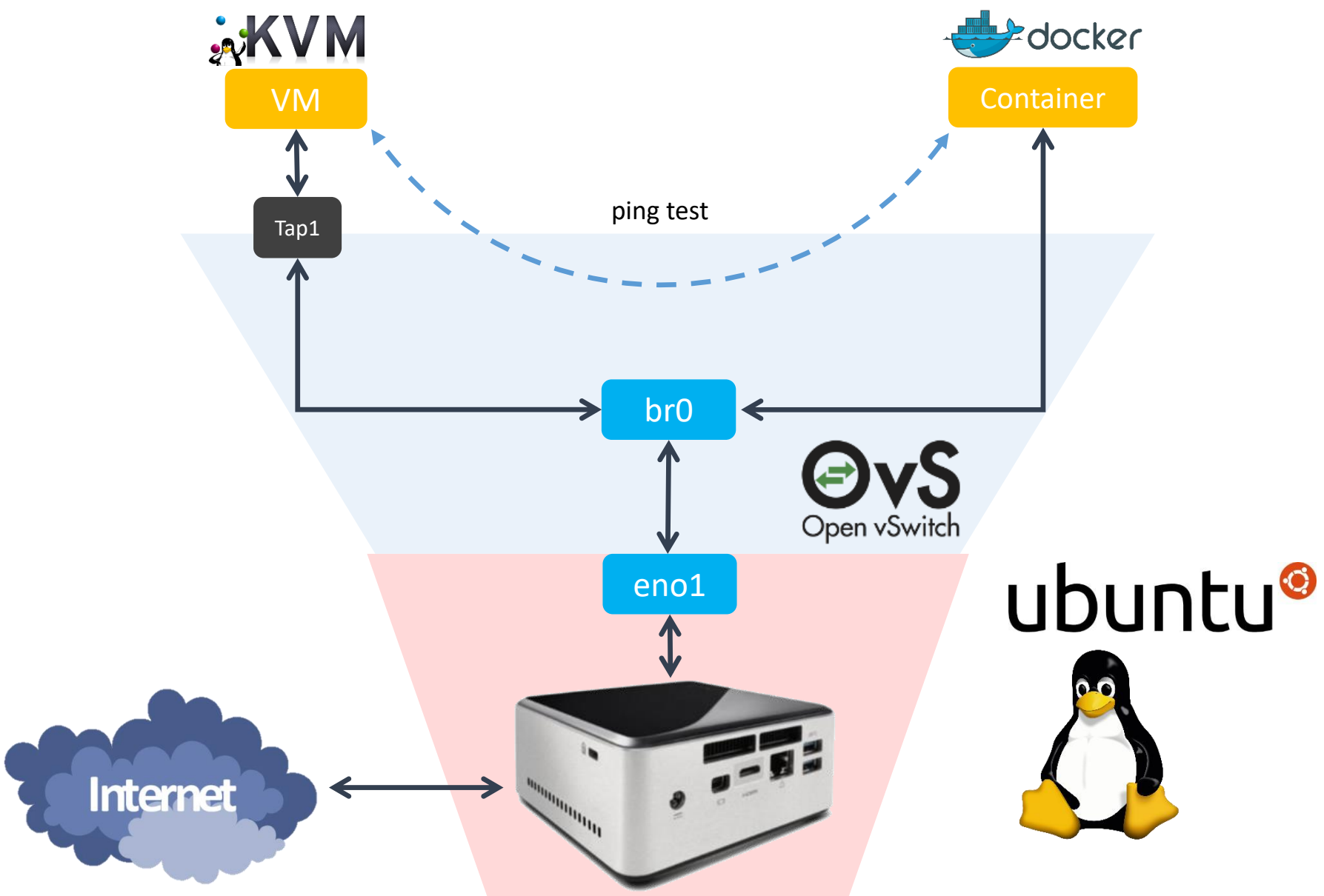
Computer Systems For AI-inspired Cloud Theory & Lab.

Lab #1: Box



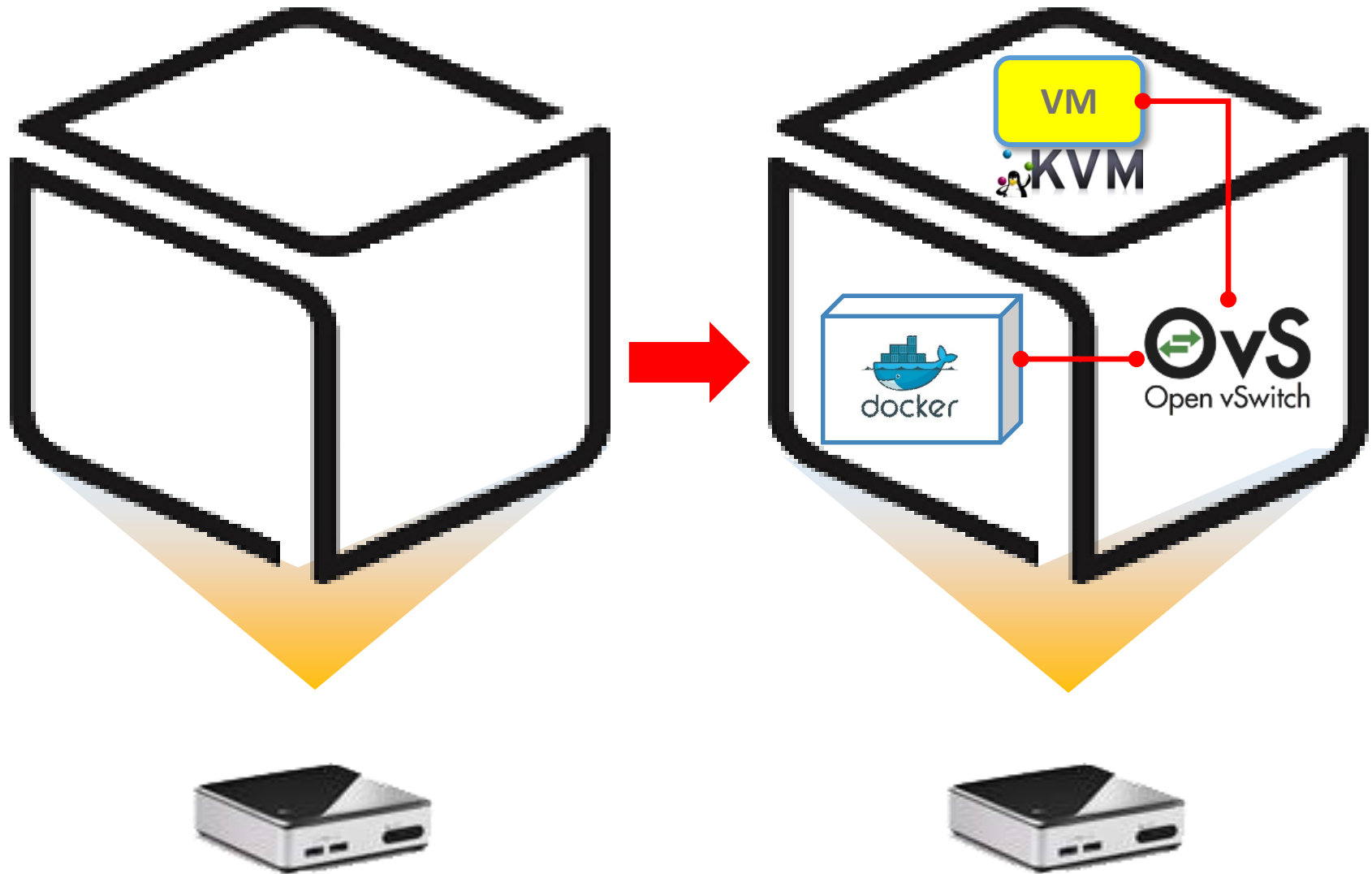
<https://github.com/SmartX-Labs/SmartX-Mini-MOOC>

Box Lab: Outline



Box Lab: Final Goal

Lab #1: Box 3



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.

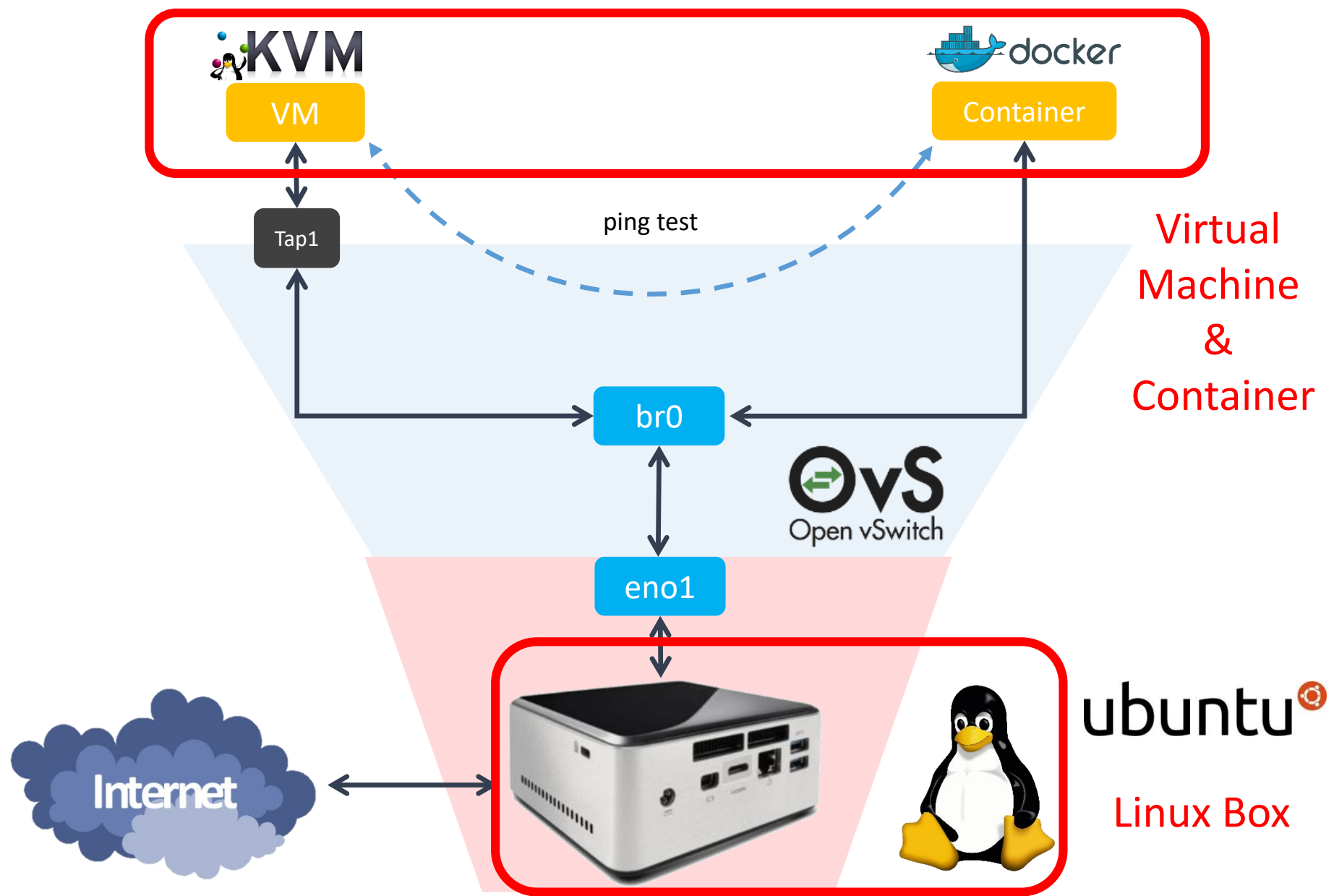


The keyboard means that you should execute instructions by following the guidance.

Lab Theory



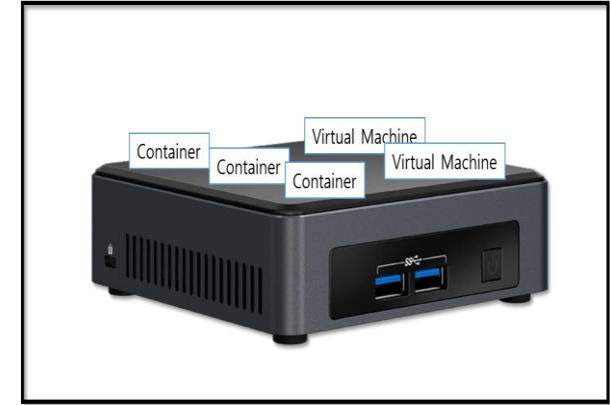
Linux Box with Virtualization/Containers



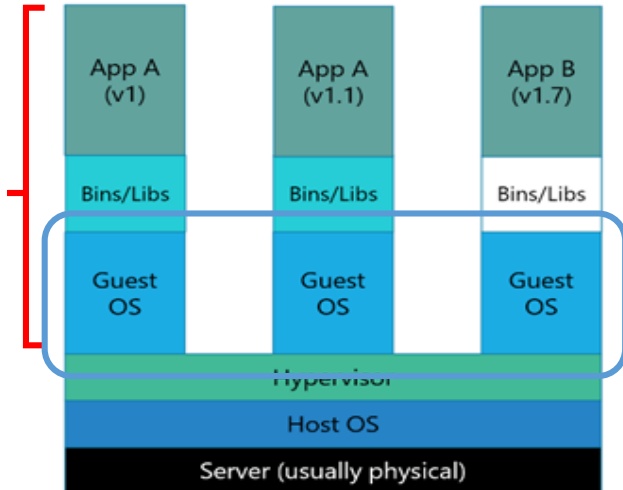
VMs and Containers on a Linux Box

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)



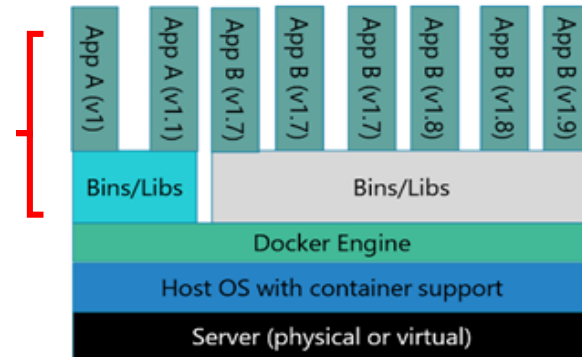
Virtual Machines (VMs)



Hypervisor Tool for creating VMs

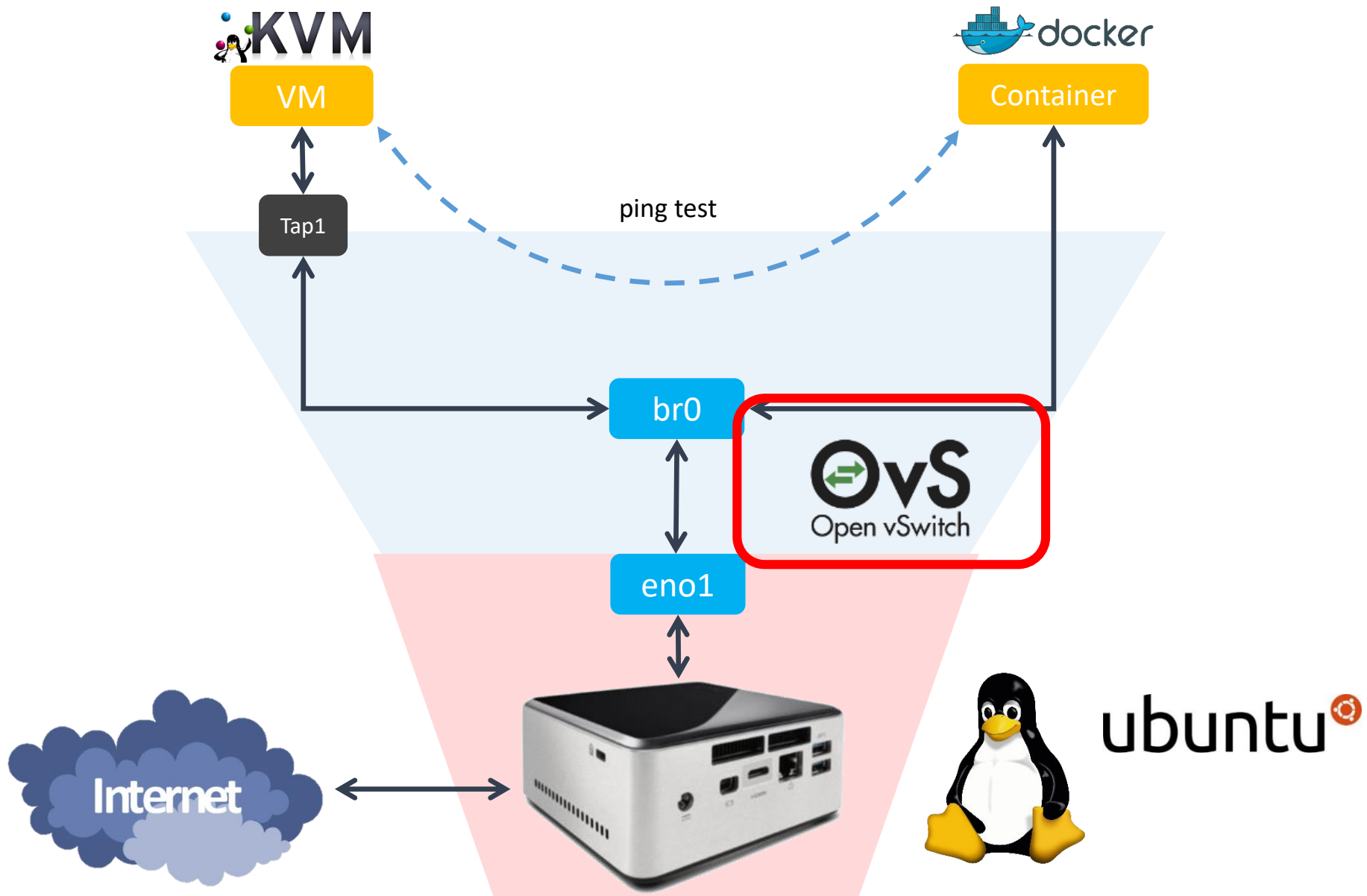
Heavier than
Container!

Containers

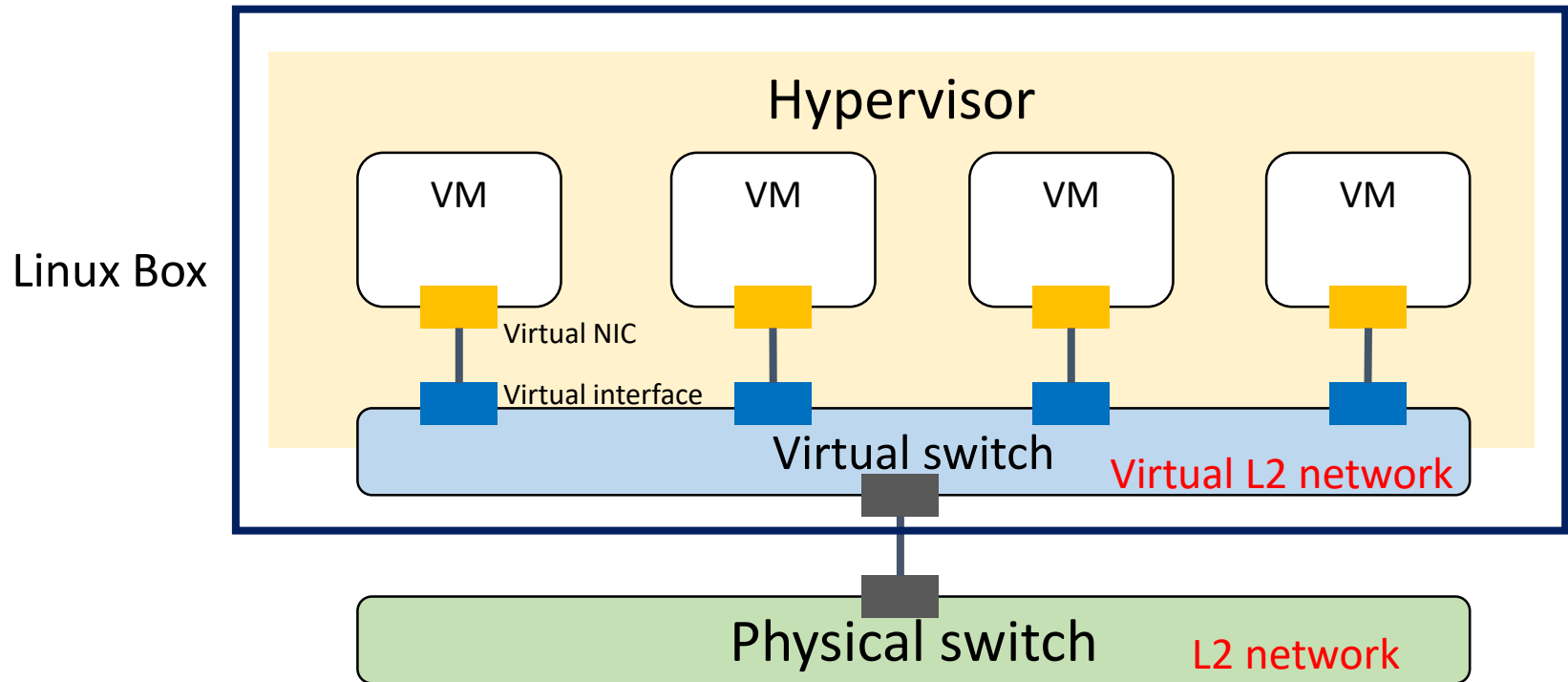


Container Runtime Tool for running containers

A Switch inside Linux Box: Open vSwitch

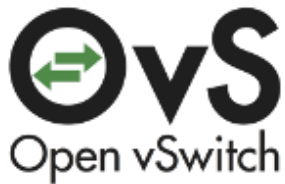


Virtual Switch in a Box to connect VMs



- 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.

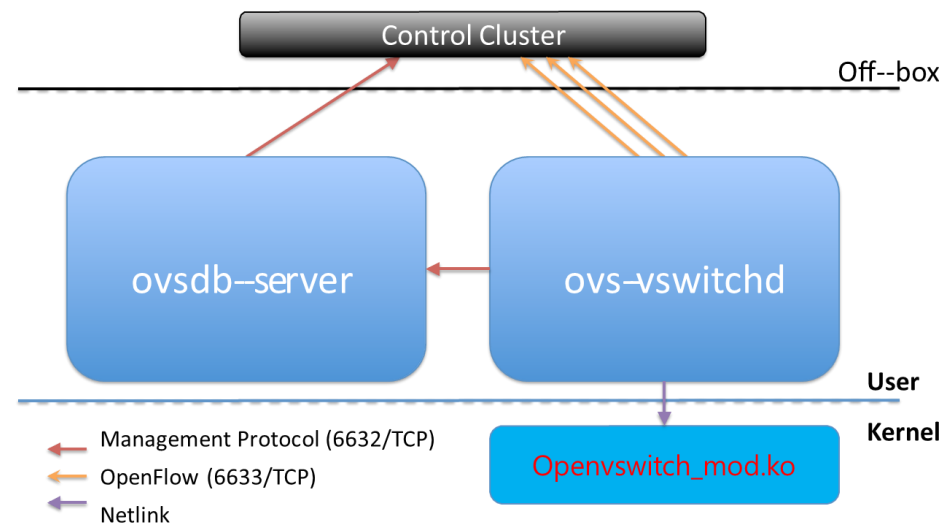
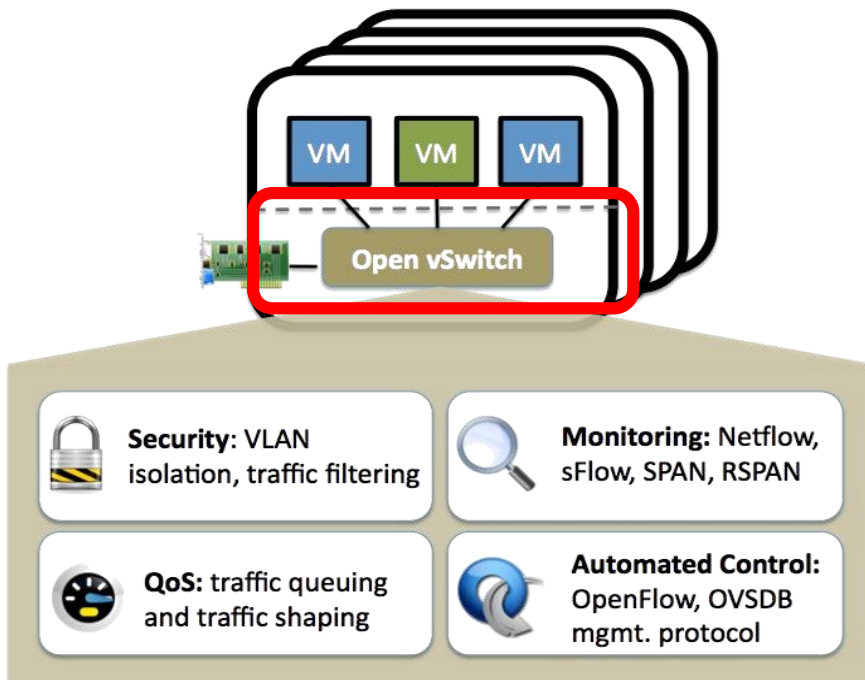
Linux-adopted virtual switch: Open vSwitch



<http://openvswitch.org/>

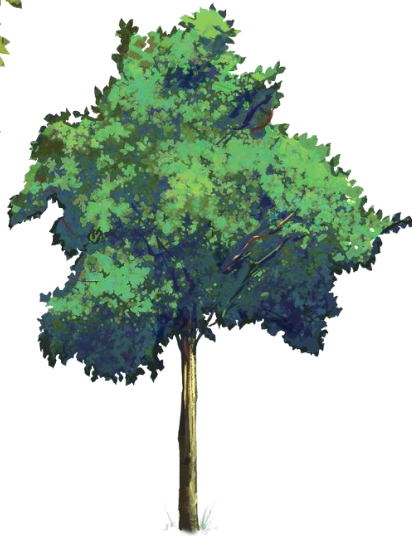


Open vSwitch is an open-source virtual switch software designed for virtual servers.



OVS Main components

Lab Practice

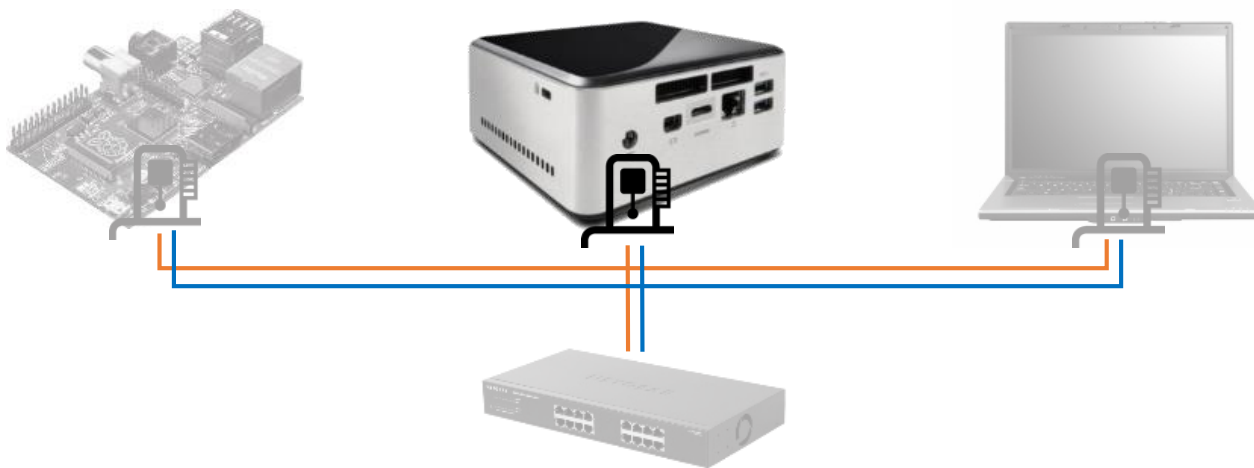


Wired connection

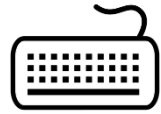
NAME: Raspberry Pi Model B (Pi)
CPU: ARM Cortex A7 @900MHz
CORE: 4
Memory: 1GB
SD Card: 32GB

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



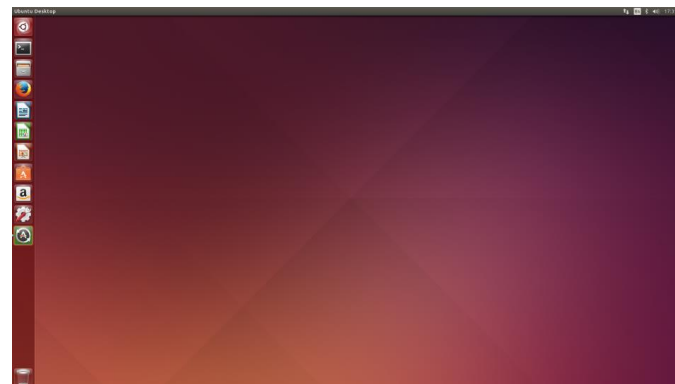
NAME: netgear prosafe 16 port gigabit switch(Switch)
Network Ports: 16 auto-sensing 10/100/1000 Mbps Ethernet ports



- OS : Ubuntu Desktop 18.04 LTS(64bit)
 - Download Site : <http://old-releases.ubuntu.com/releases/bionic/>

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

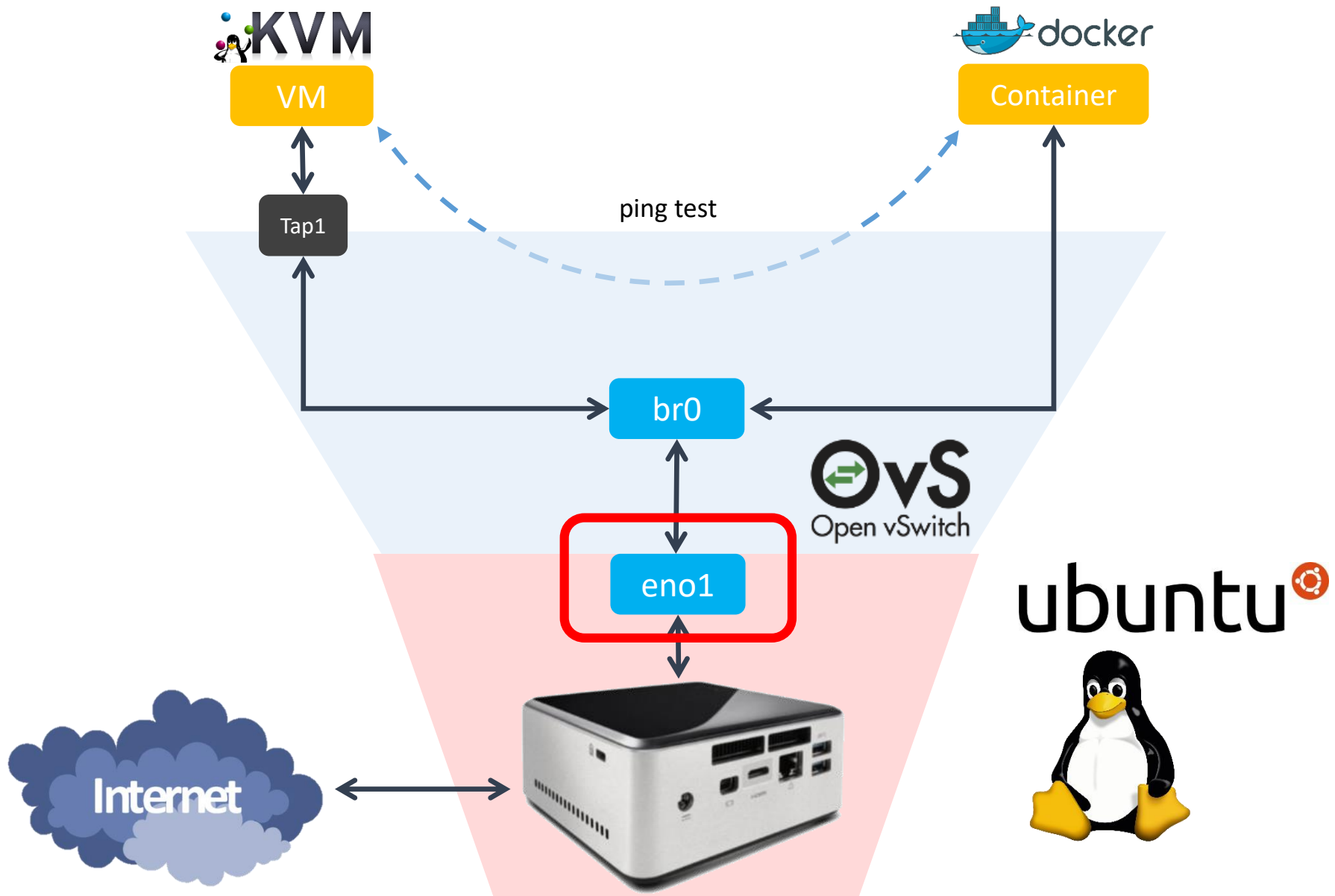
- ~~Bootable USB configuration (no bootable CD, no CD-Rom in NUC) using the downloaded file (ubuntu-16.04.4-desktop-amd64.iso, 1.5Gb)~~
- Installed on NUC



**Ubuntu Home
Screen After
Installation**

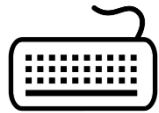
#2 - NUC: Network Configuration (1/3)

Lab #1: Box 14



#2 - NUC: Network Configuration (1/3)

- Network interface



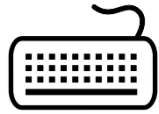
```
$sudo apt update
$sudo apt upgrade
$sudo apt install net-tools
$ifconfig -a
```

```
netcs@netcs-desktop:~$ ifconfig -a
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.199 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::82bb:956:11d6:e640 prefixlen 64 scopeid 0x20<link>
    ether ec:a8:6b:fb:a2:09 txqueuelen 1000 (Ethernet)
    RX packets 255746 bytes 382878532 (382.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 40500 bytes 3146671 (3.1 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 20 memory 0xf7c00000-f7c20000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 2771 bytes 237705 (237.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2771 bytes 237705 (237.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

#2 - NUC: Network Configuration (1/3)

- Network interface (for people using the iptime)



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

auto eno1
iface eno1 inet static
    address 192.168.0.6
    netmask 255.255.255.0
    gateway 192.168.0.1
    dns-nameservers [nameserver 1] [nameserver 2]
-----
```

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

#2 - NUC: Network Configuration (1/3)

- Network interface (for dormitory students)



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

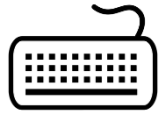
auto eno1
iface eno1 inet static
    address 210.XXX.XXX.XXX <- Input your dorm. room's IP address
    netmask 255.255.255.0
    gateway 210.xxx.xxx.xxx <- Input your dorm. room's gateway
    dns-nameservers 8.8.8.8 <- Input your dorm. room's dns-nameservers
-----
```

Some NUC have different Interface name.

So you need to check your NUC's interface name using 'ip a' command.

#2 - NUC: Network Configuration (1/3)

- Network interface (for lab students)



```
$sudo apt install ifupdown
```

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

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

```
auto lo
```

```
iface lo inet loopback
```

```
auto eno1
```

```
iface eno1 inet static
```

```
address 172.XXX.XXX.XXX <- Input your given IP address
```

```
netmask 255.255.255.0
```

```
gateway 172.xxx.x.x <- Input your given gateway
```

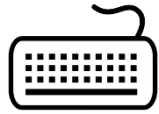
```
dns-nameservers 8.8.8.8 <- Input your given dns-nameservers
```

Some NUC have different Interface name.

So you need to check your NUC's interface name using 'ip a' command.

#2 - NUC: Network Configuration (1/3)

- Network interface



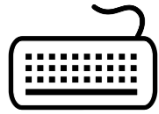
```
$ sudo su
#ifdown --force eno1 lo && ifup -a
#systemctl unmask networking
#systemctl enable networking
#systemctl restart networking
```

Disable netplan

```
#systemctl stop systemd-networkd.socket systemd-networkd \
networkd-dispatcher systemd-networkd-wait-online
#systemctl disable systemd-networkd.socket systemd-networkd \
networkd-dispatcher systemd-networkd-wait-online
#systemctl mask systemd-networkd.socket systemd-networkd \
networkd-dispatcher systemd-networkd-wait-online
#apt-get --assume-yes purge nplan netplan.io
#exit
```

#2 - NUC: Network Configuration (1/3)

- eno1 interface



```
$ sudo vi /etc/systemd/resolved.conf
```

```
----- /etc/systemd/resolved.conf -----
```

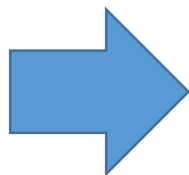
```
...
```

```
DNS = 8.8.8.8 8.8.4.4
```

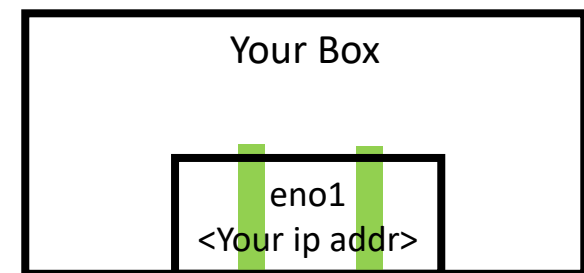
```
...
```

```
$sudo systemctl restart systemd-resolved.service
```

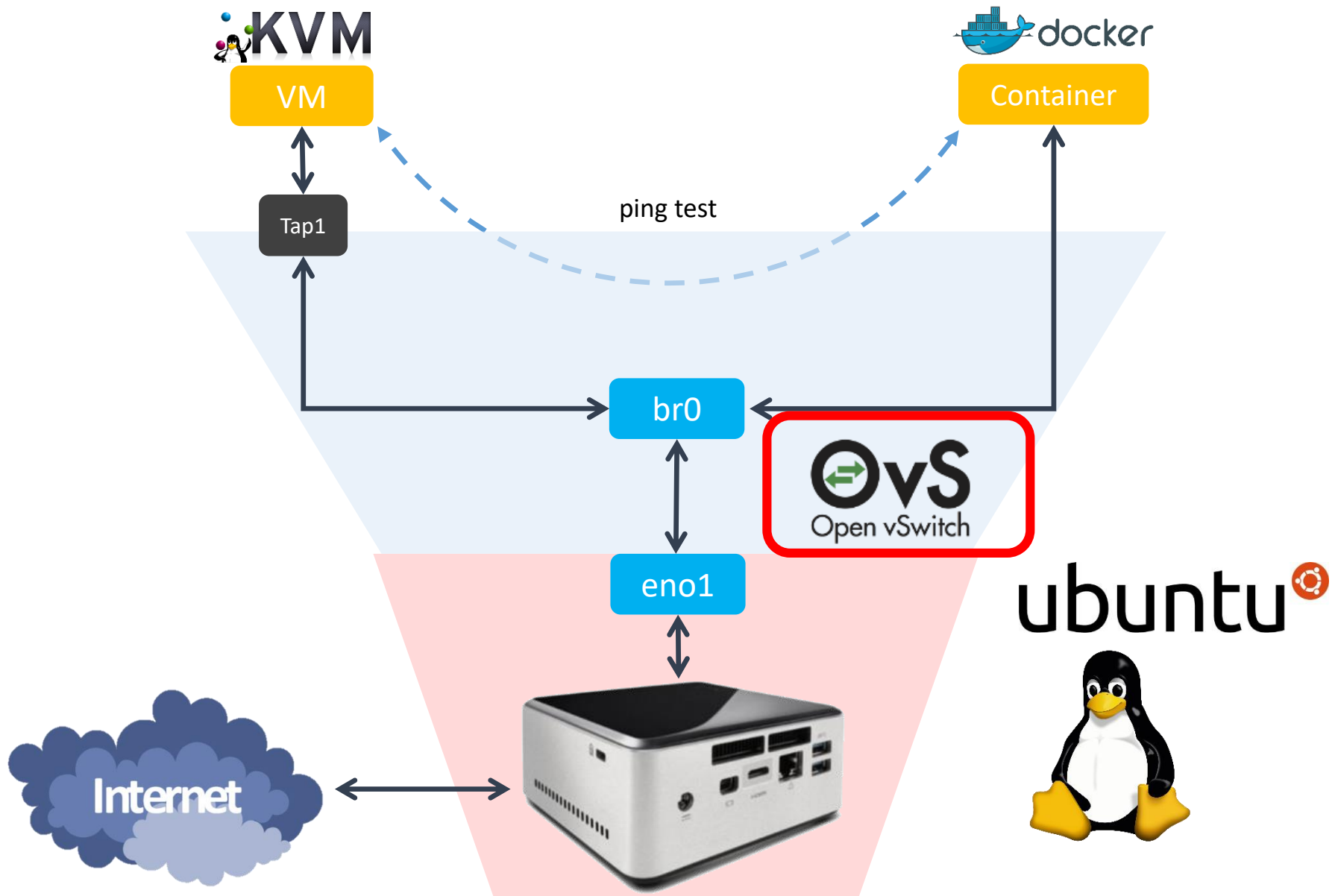
```
$sudo ifup eno1
```



NUC internet works!



#3 - NUC: OVS installation



#3 - NUC: OVS installation

- Update installation of OVS package

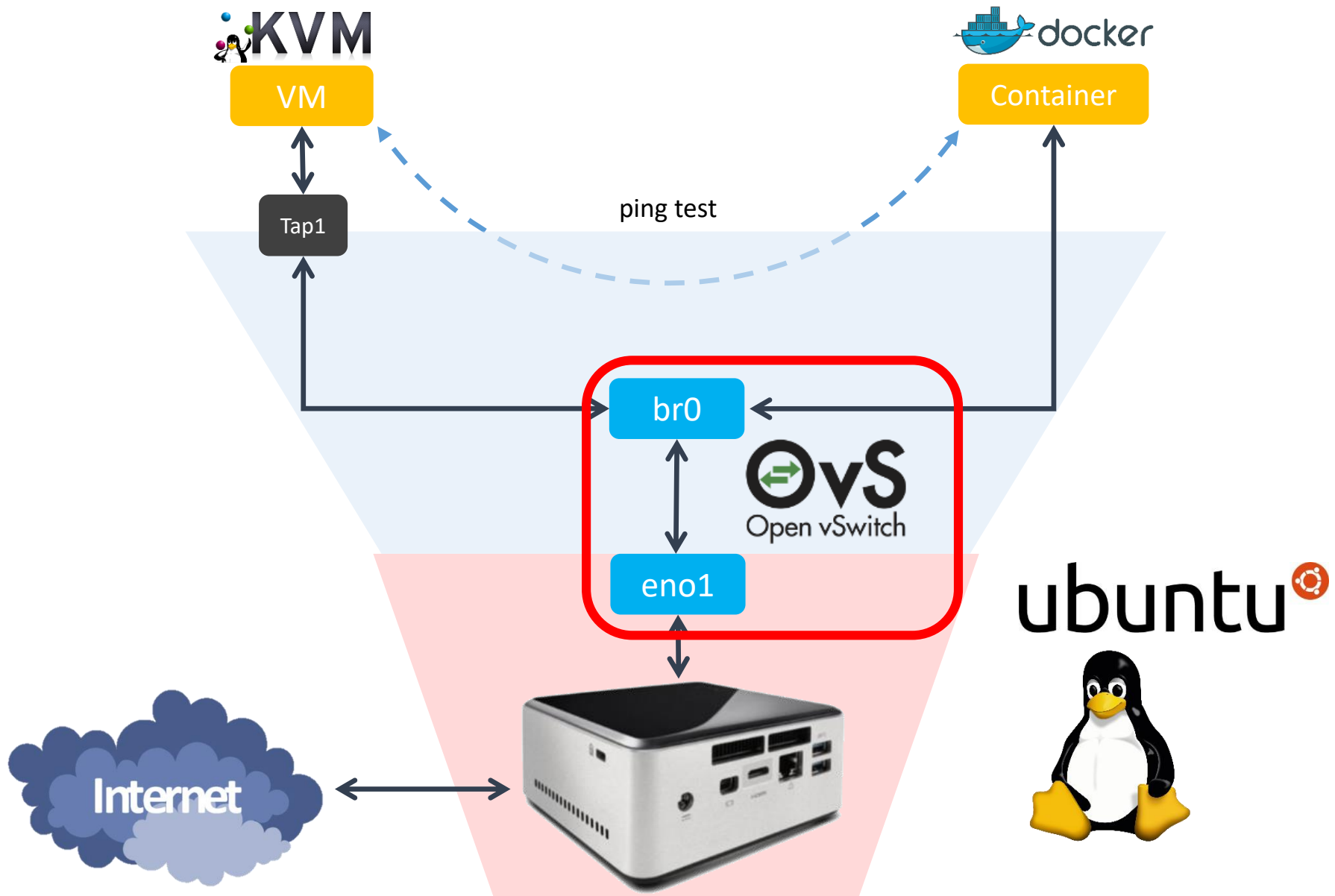


Update index information of Open vSwitch package.
Install a Open vSwitch package, **openvswitch-switch**.
Other dependencies are automatically installed.

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

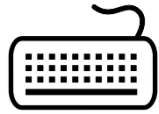
#4 - NUC: Network Configuration (2/3)

Lab #1: Box 23



#4 - NUC: Network Configuration (2/3)

- Connect OVS br0 and NUC eno1 via OVS

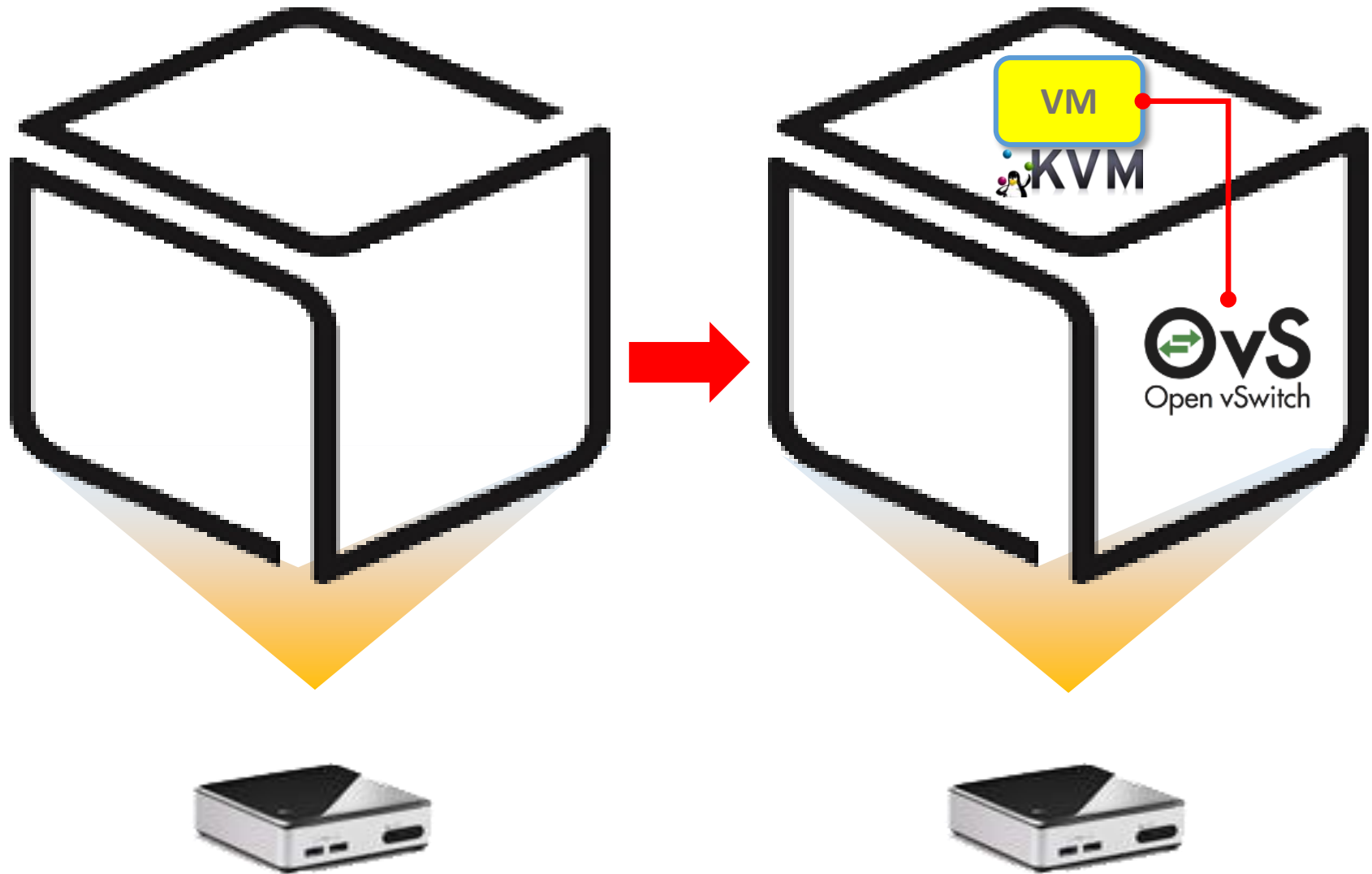


\$sudo ovs-vsctl add-br br0

```
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"
```

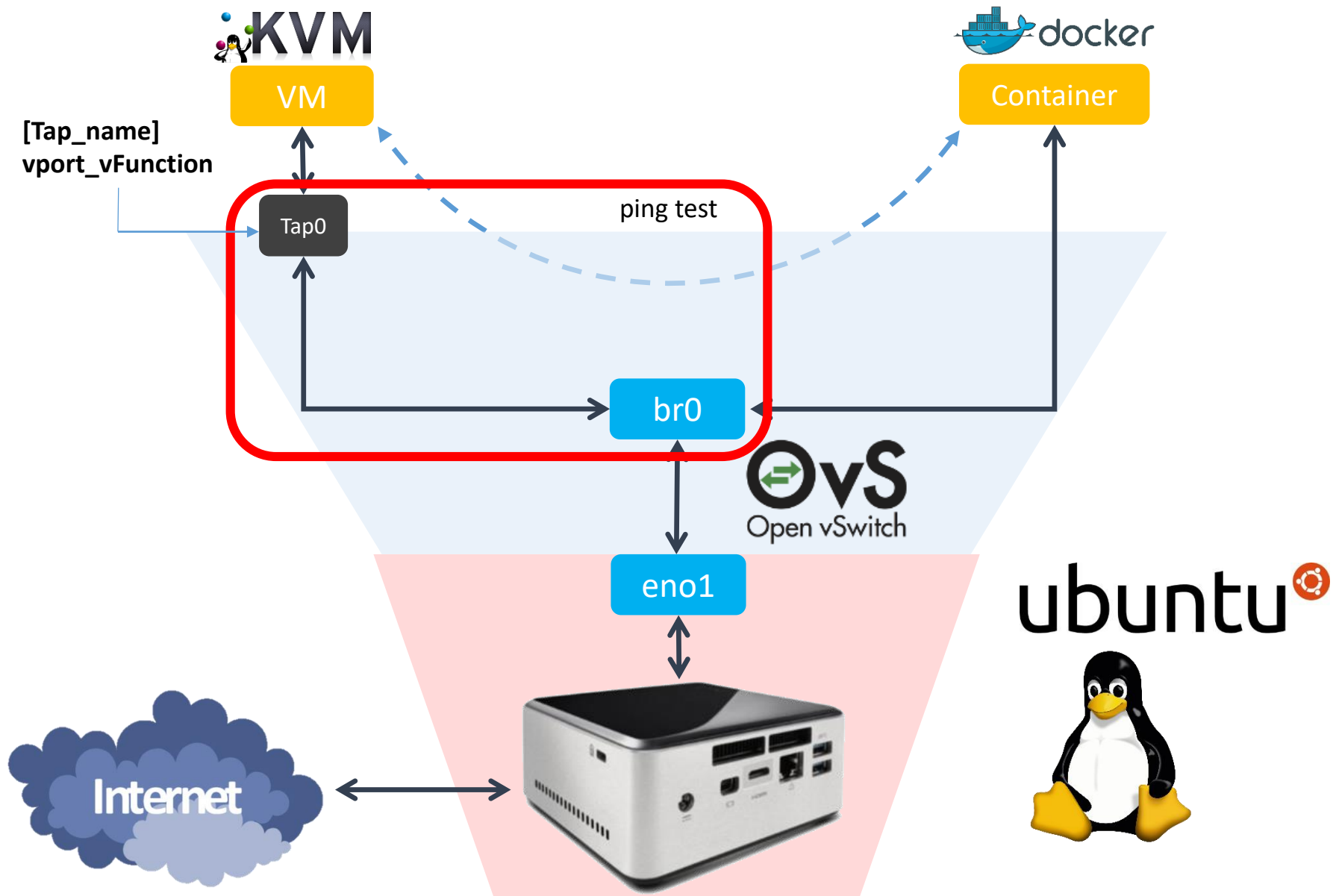

KVM-based VM connected via OVS

- Goal of this section



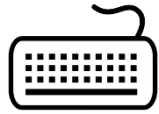
#5 - NUC: Network Configuration (3/3)

Lab #1: Box 26



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

- 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 -----
```

```
...
```

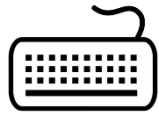
(Append the lines below to the config file)

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

auto br0
iface br0 inet manual
    address 192.168.100.1
    netmask 255.255.255.0
```

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

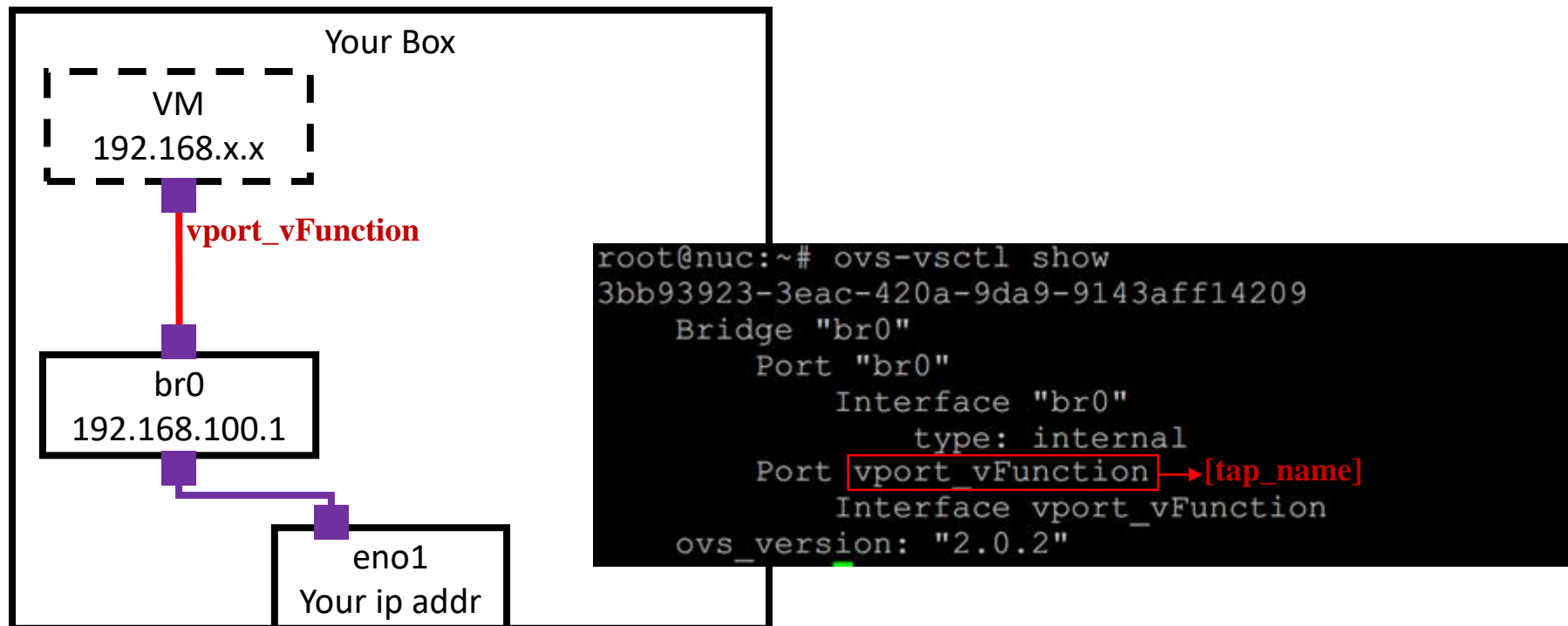
-Connect vport_vFunction, br0 through OVS



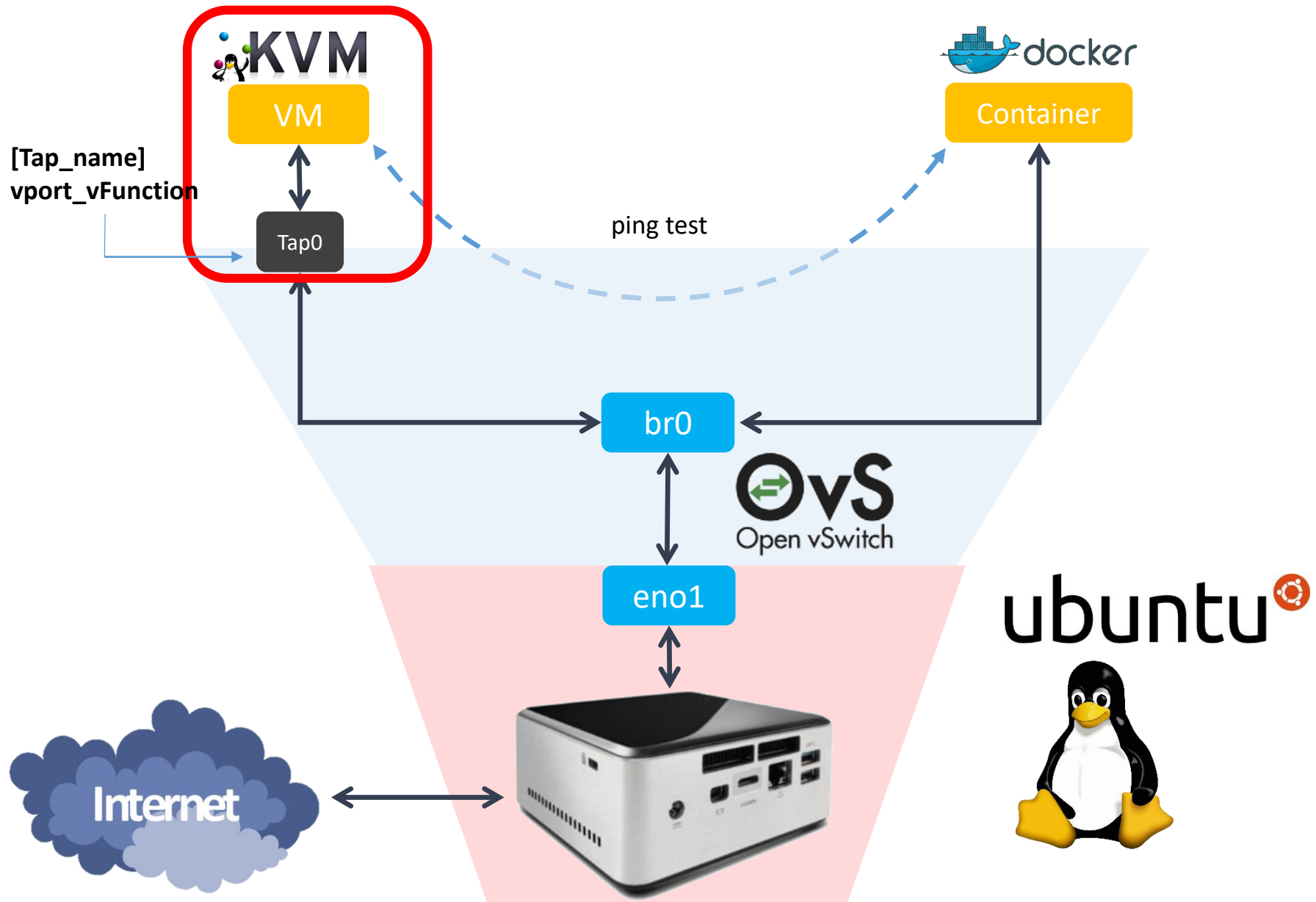
Turn on the tap interface and attach it to br0.

```
$sudo ifup vport_vFunction
$sudo ifup br0
$sudo ovs-vsctl add-port br0 vport_vFunction // Turn on and attach to br0
```

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

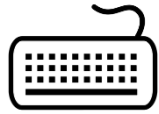


#6 - NUC: Making VM with KVM



#6 - NUC: Making VM with KVM

-Install dependency to upgrade KVM



Install dependency & download Ubuntu 16.04.4 64bit server image.

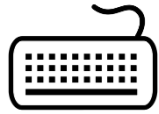
```
$sudo apt-get install qemu-kvm libvirt-bin           //upgrade KVM
                                                    //qemu is open-source emulator

$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.

#6 - 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
```

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
```

#6 - NUC: Making VM with KVM

-Prepare for Ubuntu VM

Configure SNAT with iptables for VM network

```
$sudo iptables -A FORWARD -i eno1 -j ACCEPT
$sudo iptables -A FORWARD -o eno1 -j ACCEPT
$sudo iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o eno1 -j SNAT --to <Your ip address>
```

```
$vi /etc/sysctl.conf
```

```
#net.ipv4.ip_forward=1
```



```
net.ipv4.ip_forward=1
```

```
$sysctl -p
```

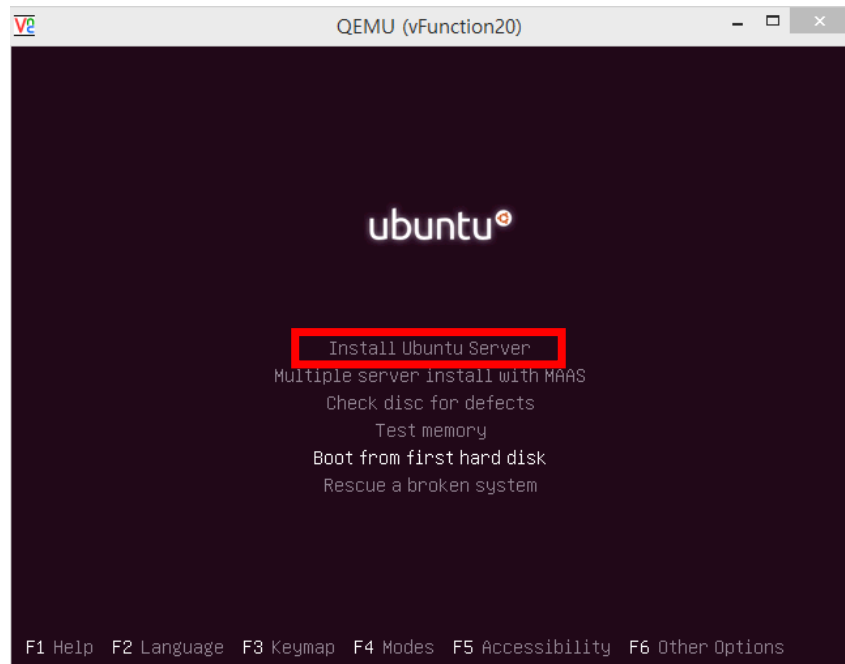
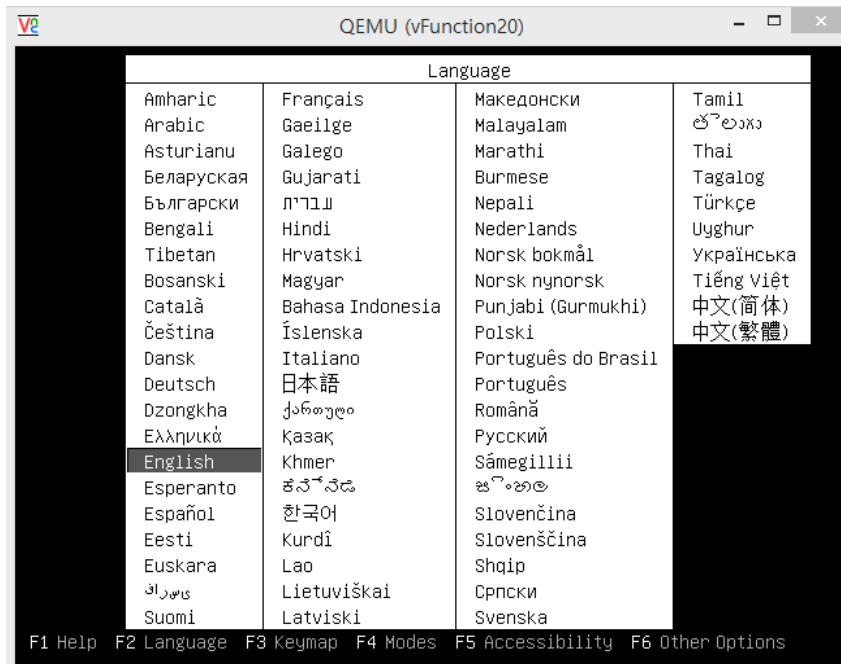
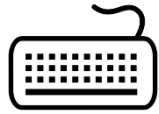
```
→ net.ipv4.ip_forward = 1
```

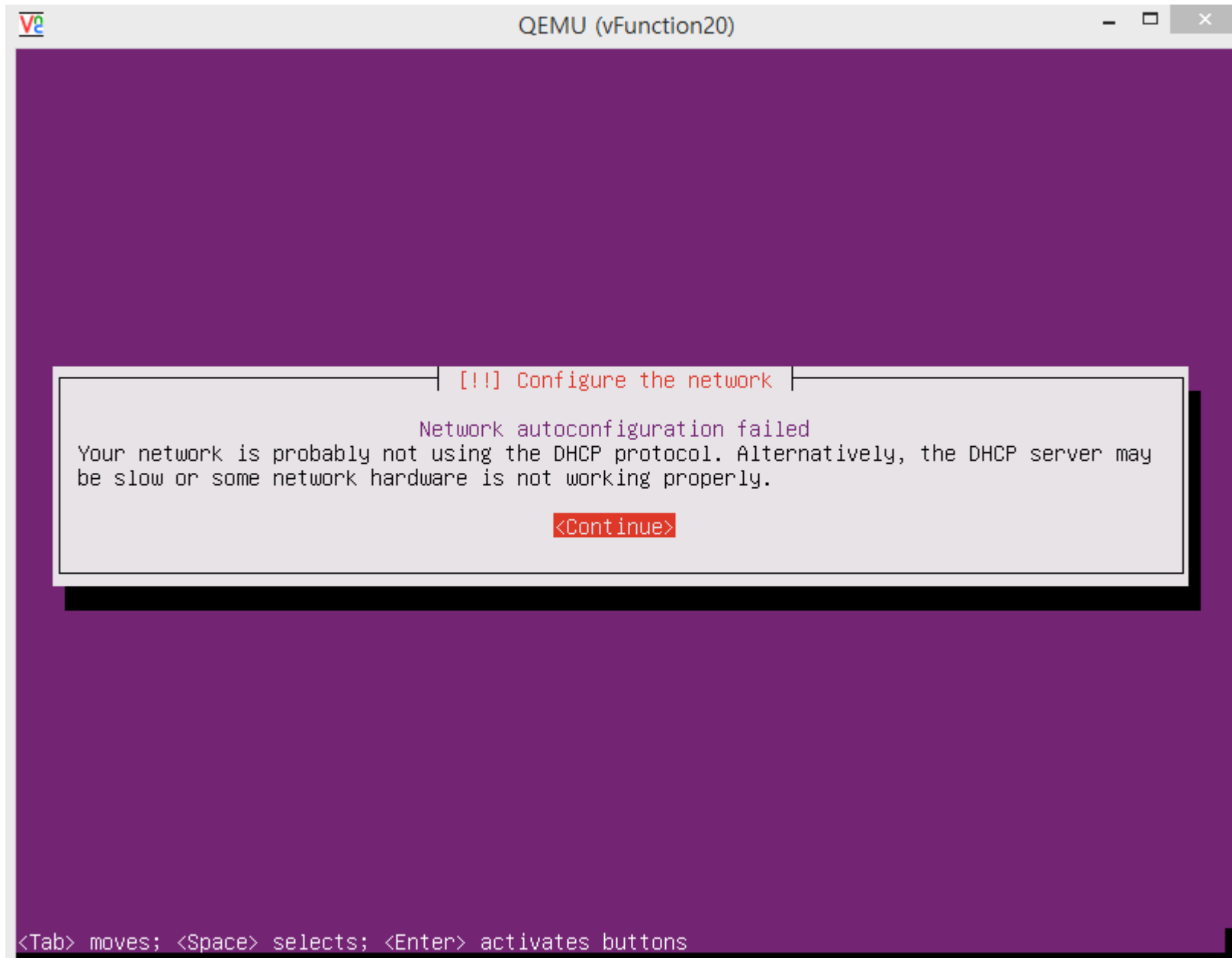
Configuration complete

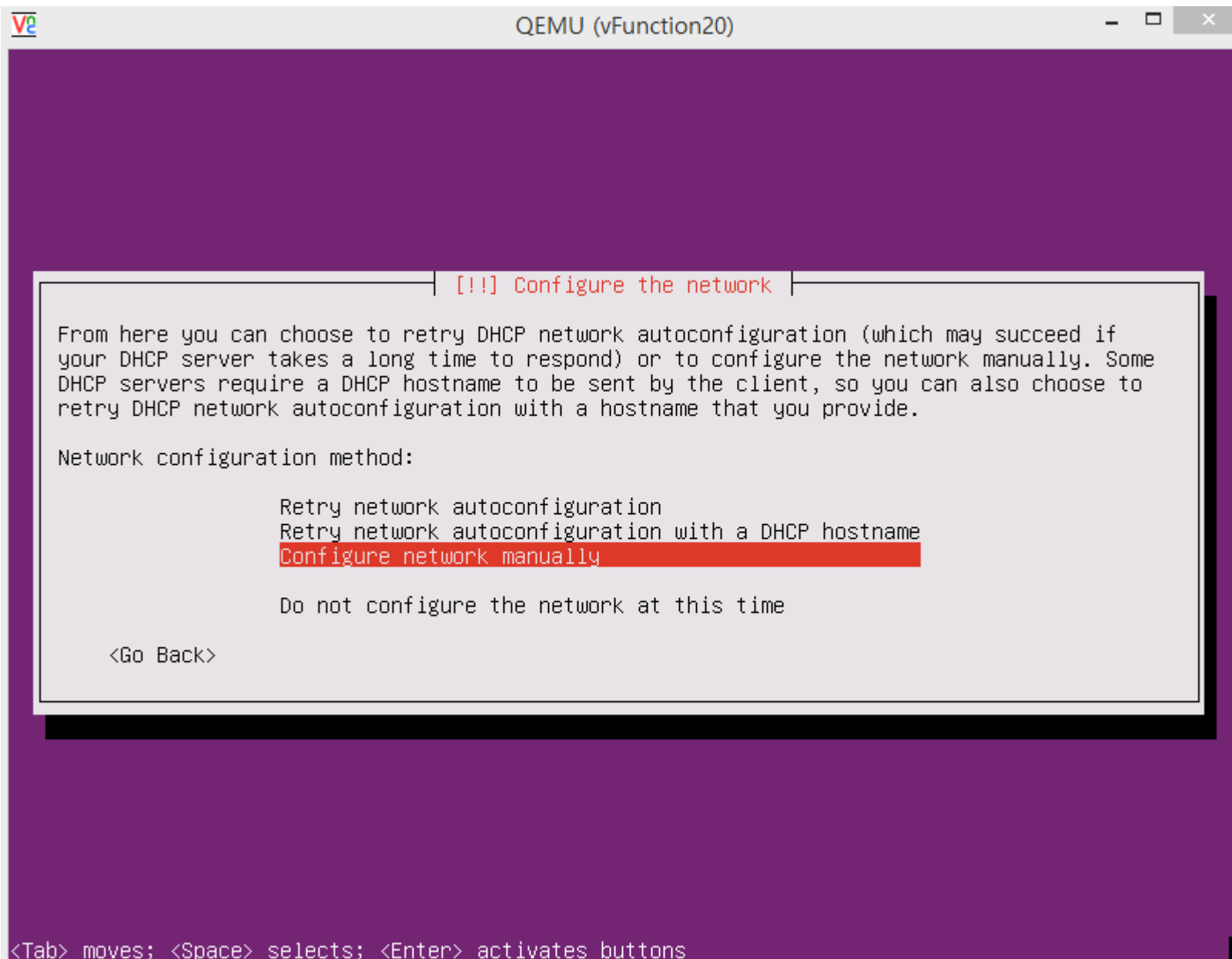
```
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#
#kernel.domainname = example.com
#
# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
#
#####3
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
#
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1
#
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
#
# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
#
```

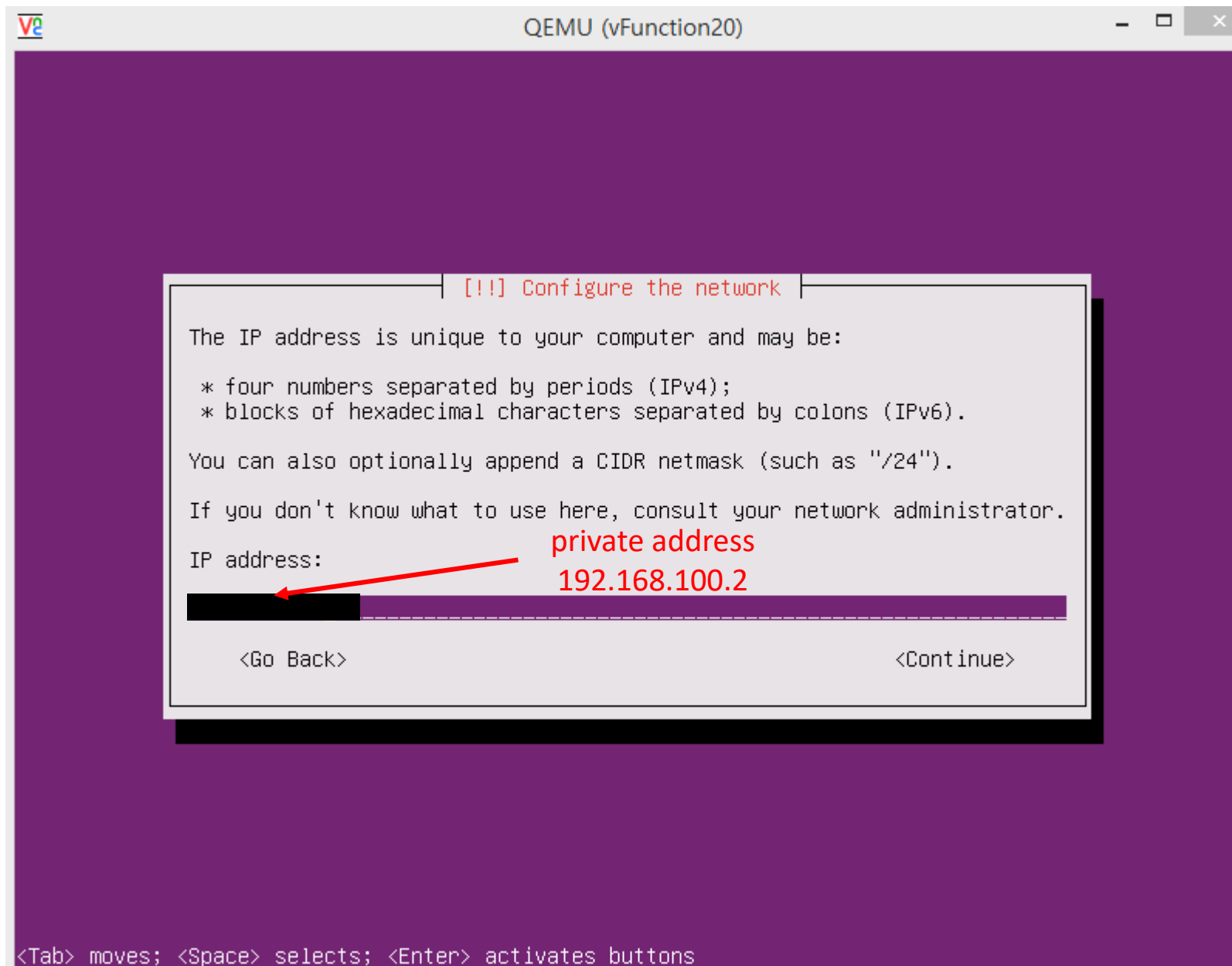

#6 - NUC: Making VM with KVM

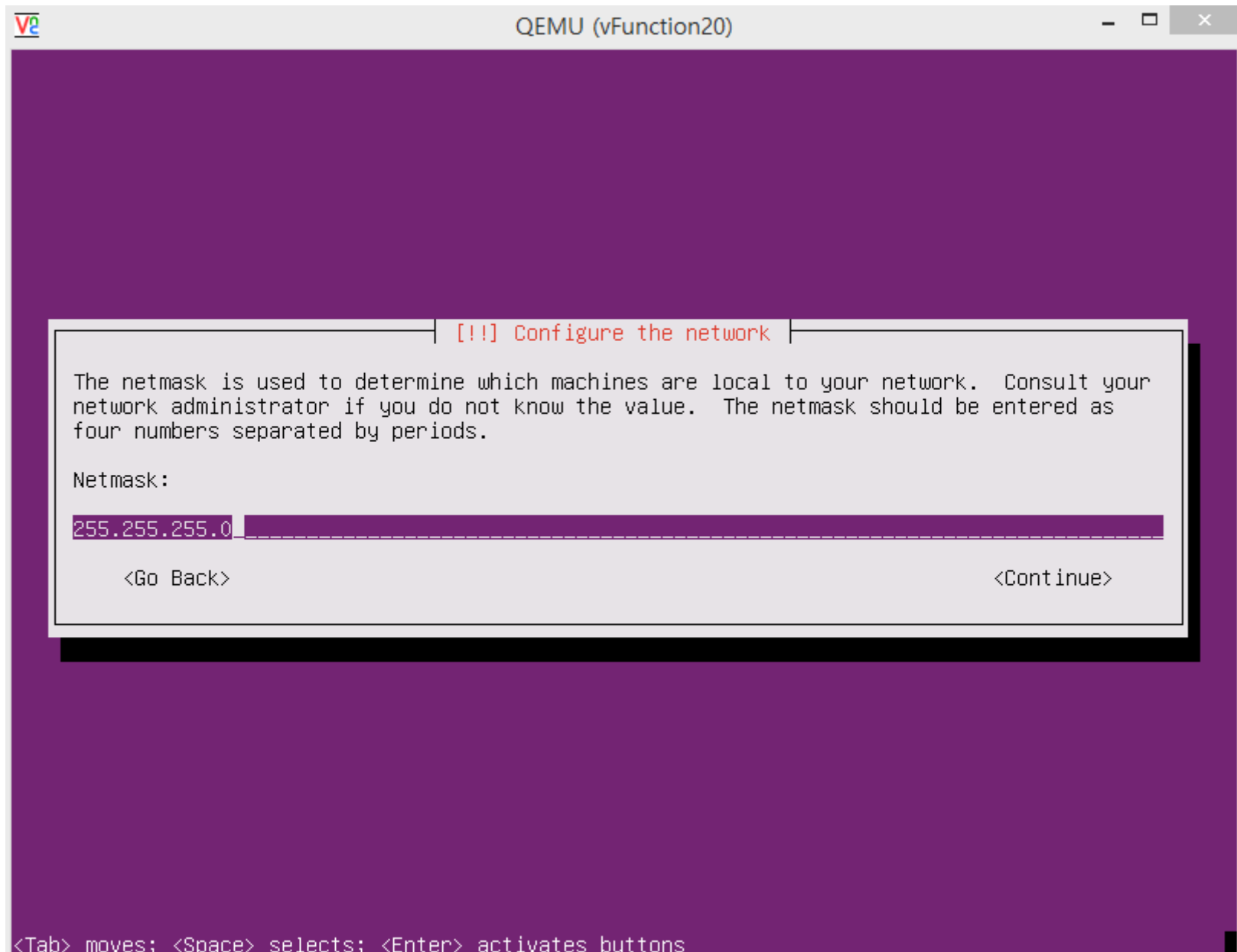
-Install Ubuntu VM












QEMU (vFunction20)

!!! Configure the network

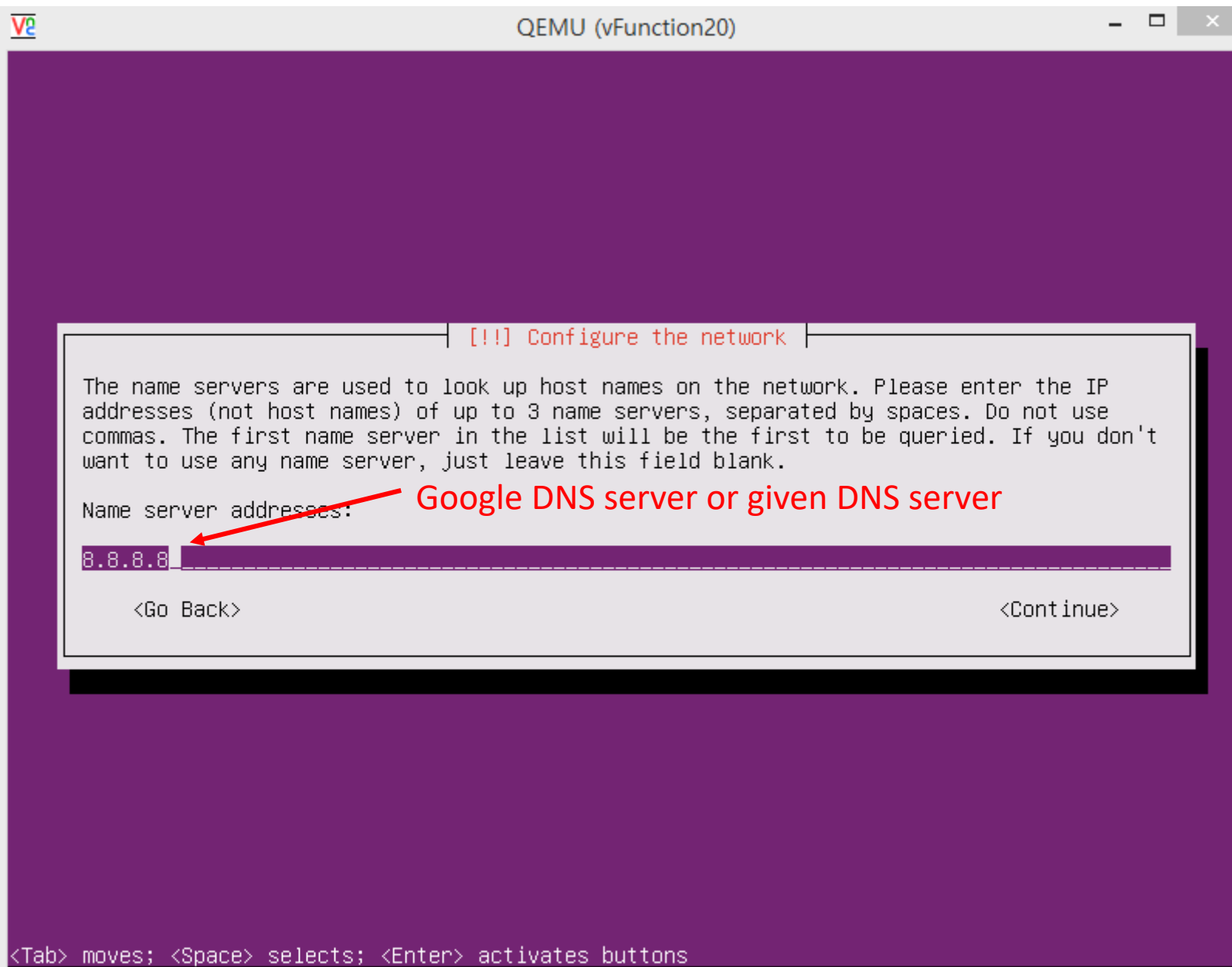
The gateway is an IP address (four numbers separated by periods) that indicates the gateway router, also known as the default router. All traffic that goes outside your LAN (for instance, to the Internet) is sent through this router. In rare circumstances, you may have no router; in that case, you can leave this blank. If you don't know the proper answer to this question, consult your network administrator.

Gateway:

 Gateway ip of your private network
192.168.100.1

<Go Back> <Continue>

<Tab> moves; <Space> selects; <Enter> activates buttons

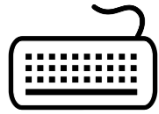


#6 - NUC: Making VM with KVM

- 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...
```

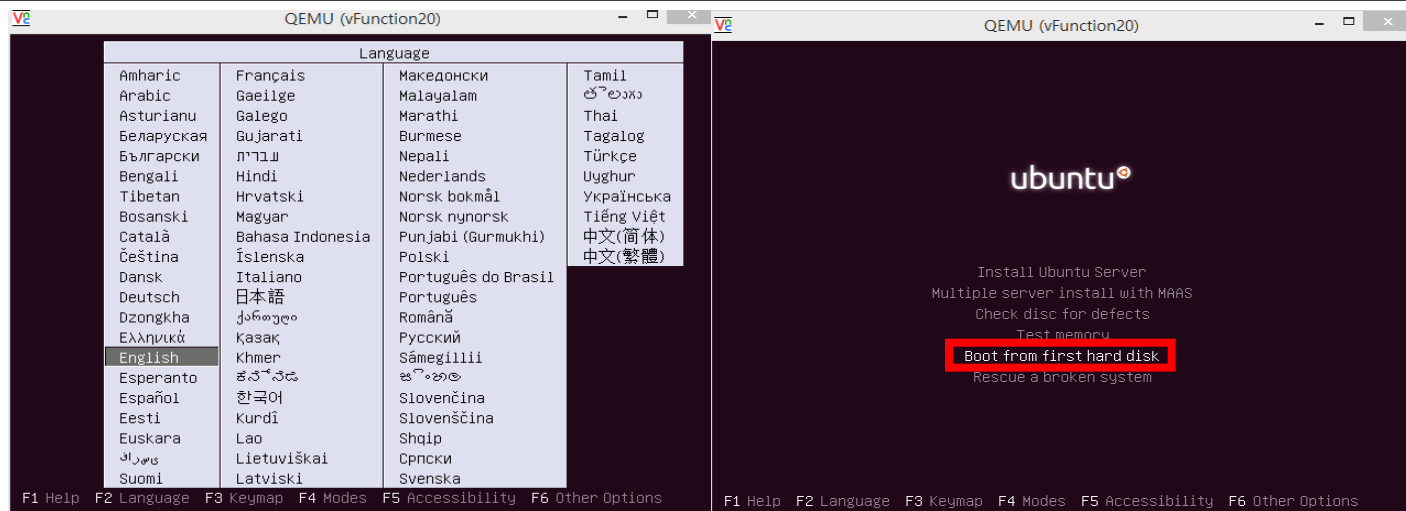
```
Connected to localhost.Escape character is '^]' (Ctrl+)].
```

```
QEMU 0.11.0 monitor - type 'help' for more information
```

```
(qemu) eject ide1-cd0
```

```
(qemu) Ctrl+]
```

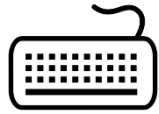
```
$ xvnc4viewer localhost :5
```



Push Esc

#7 - NUC: Booting VM

- VM boot command

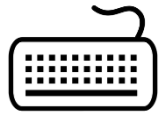


If you want boot VM again (mac should be different from others).

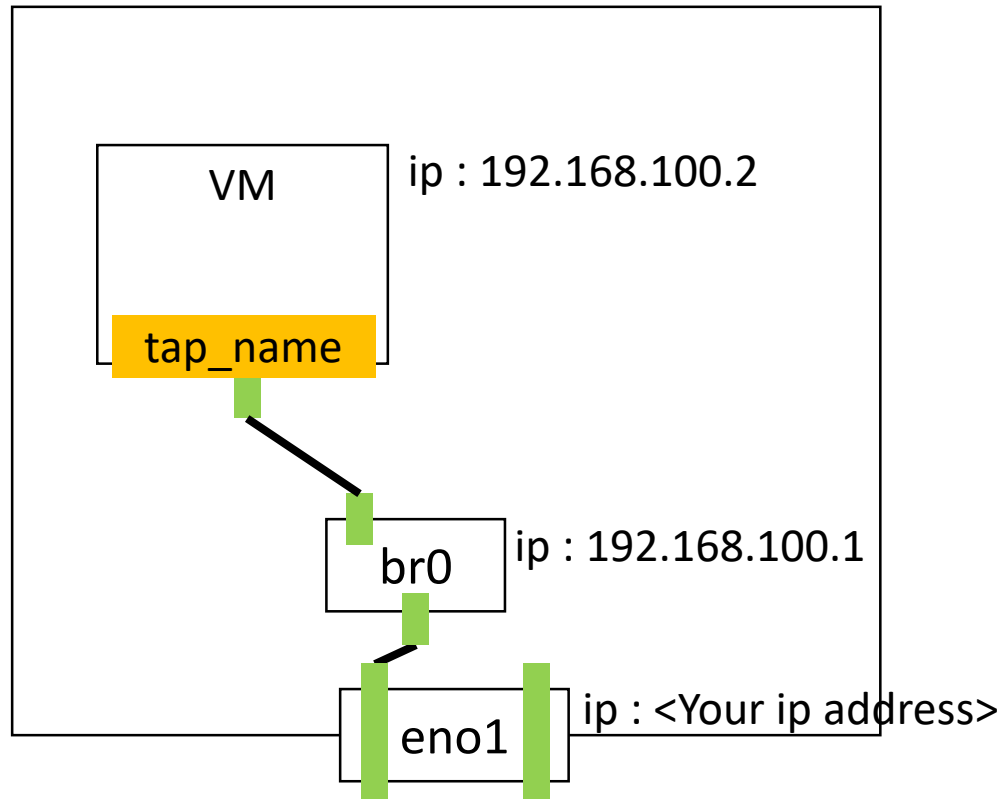
```
$sudo kvm -m [memory capacity] -name [name] -smp cpus=[#cpu],maxcpus= [#maxcpu] -  
device virtio-net-pci,netdev=net0,mac= [EE:EE:EE:EE:EE:EE] -netdev tap,id=net0,ifname=  
[tap_name],script=no -boot d [name].img -vnc : [#] -daemonize
```

#8 - OVS connects with KVM

- Check situation

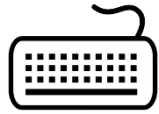


```
root@nuc:~# ovs-vsctl show
3bb93923-3eac-420a-9da9-9143aff14209
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port vport_vFunction
            Interface vport_vFunction
    ovs_version: "2.0.2"
```



#9 - NUC: Installing ssh in VM

- Don't forget to install ssh in VM



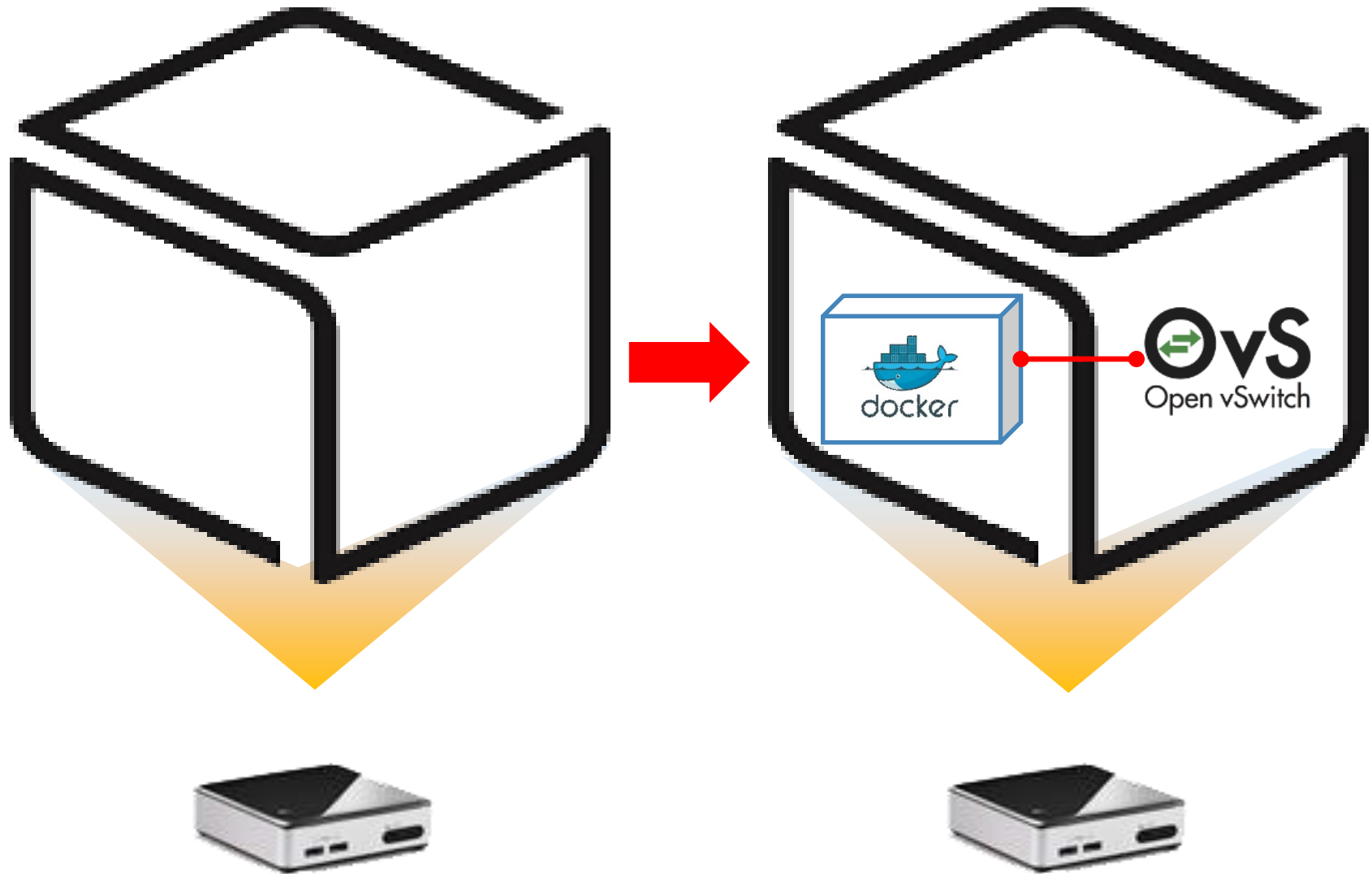
In VMs,

```
$sudo apt-get update  
$sudo apt-get install net-tools ssh -y
```

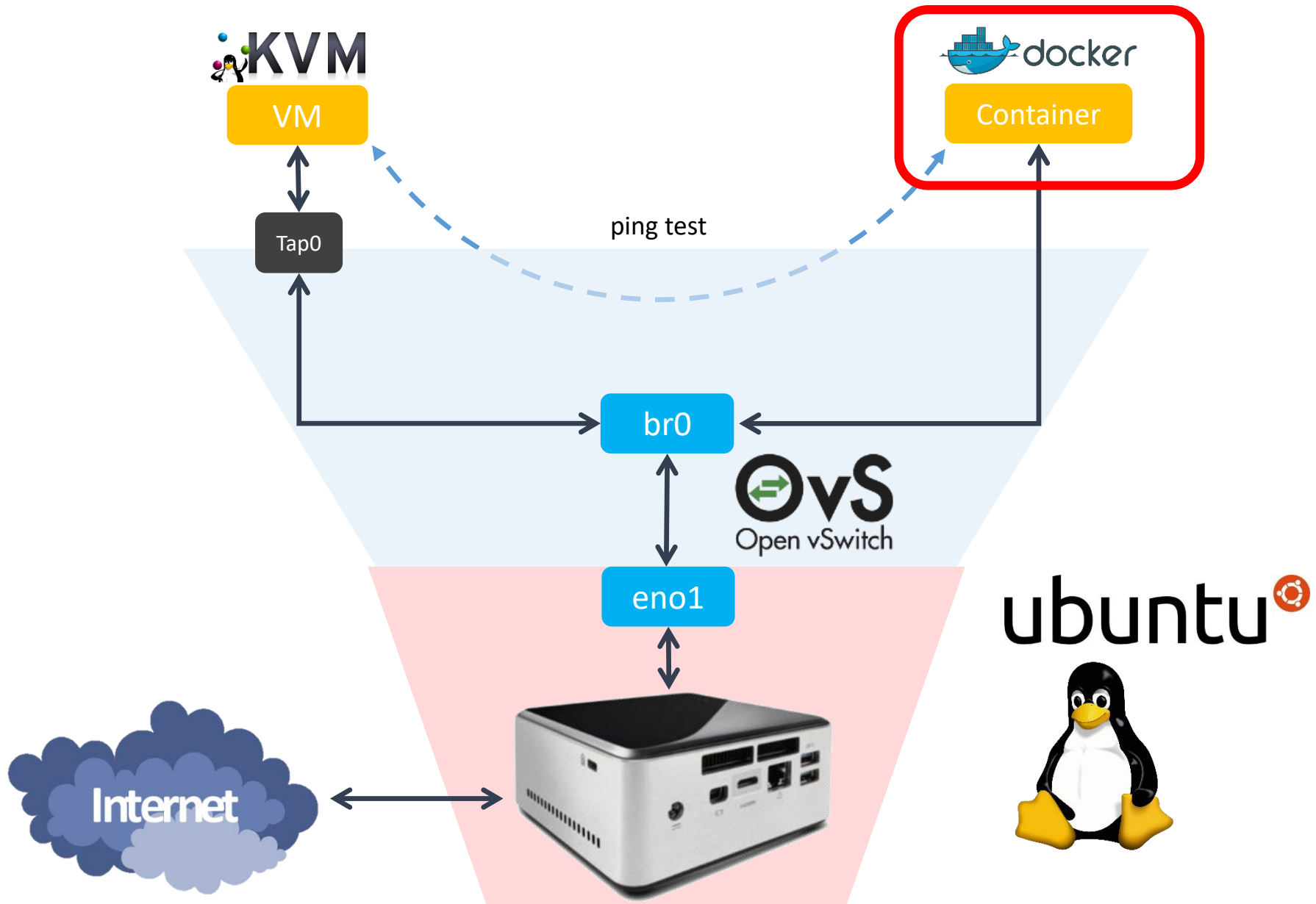
```
nuc@nuc:~$ ssh vbox@192.168.0.3  
The authenticity of host '192.168.0.3 (192.168.0.3)' can't be established.  
ECDSA key fingerprint is da:c5:2c:53:5a:6f:b4:3c:03:02:04:f3:6a:17:ca:ab.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.0.3' (ECDSA)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

Docker Container connected via OVS

- Goal of this section



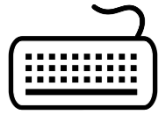
#10 - Making a Docker Container



#10 - Making a Docker Container

- Docker installation

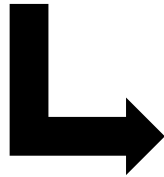
In NUC,
Docker installation.



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

(Session restart)

```
$sudo docker run hello-world
```

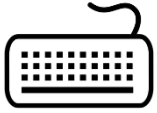


```
Hello from Docker.  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
3. The Docker 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/
```

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

#10 - Making a Docker Container

- Make container



Run docker container.

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

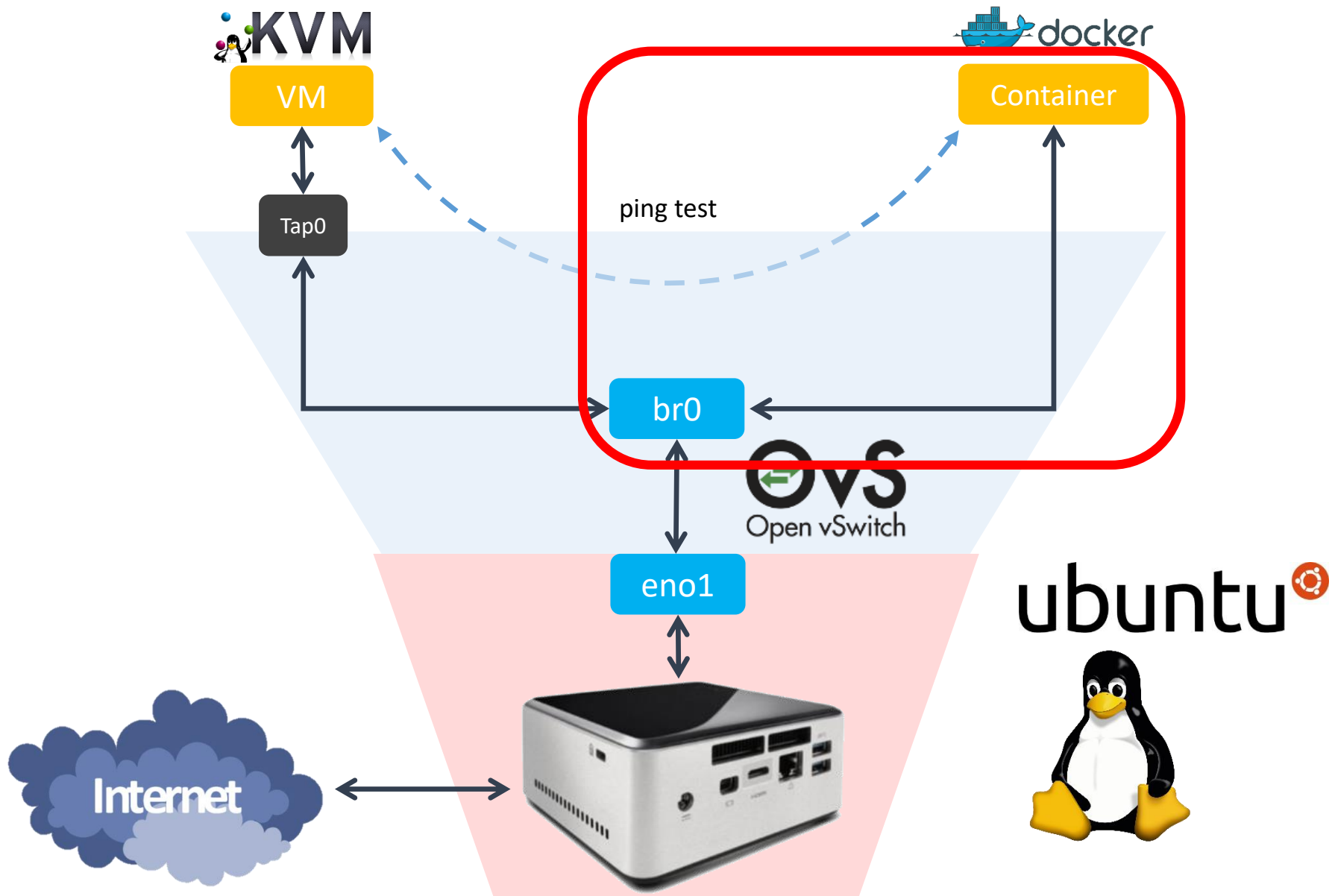
```
nuc@nuc:~$ docker run -it --net=none --name c1 ubuntu /bin/bash
root@8346684676d8:/#
```

※ctrl + p, q → detach docker container

※docker attach [container_name] → get into docker container console

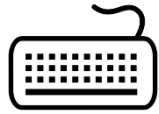
#11 - Connect Docker Container

- Connect with OVS bridge



#11 - 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 veno1 [container_name] --ipaddress=192.168.100.3/24 --gateway=192.168.100.1
```

```
$sudo docker attach [container_name]
```

```
#apt-get update
```

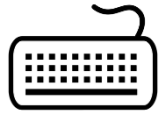
```
#apt-get install net-tools
```

```
#apt-get install iputils-ping
```

#12 – Keep Docker network configuration

Lab #1: Box 50

- /etc/rc.local



Modify /etc/rc.local

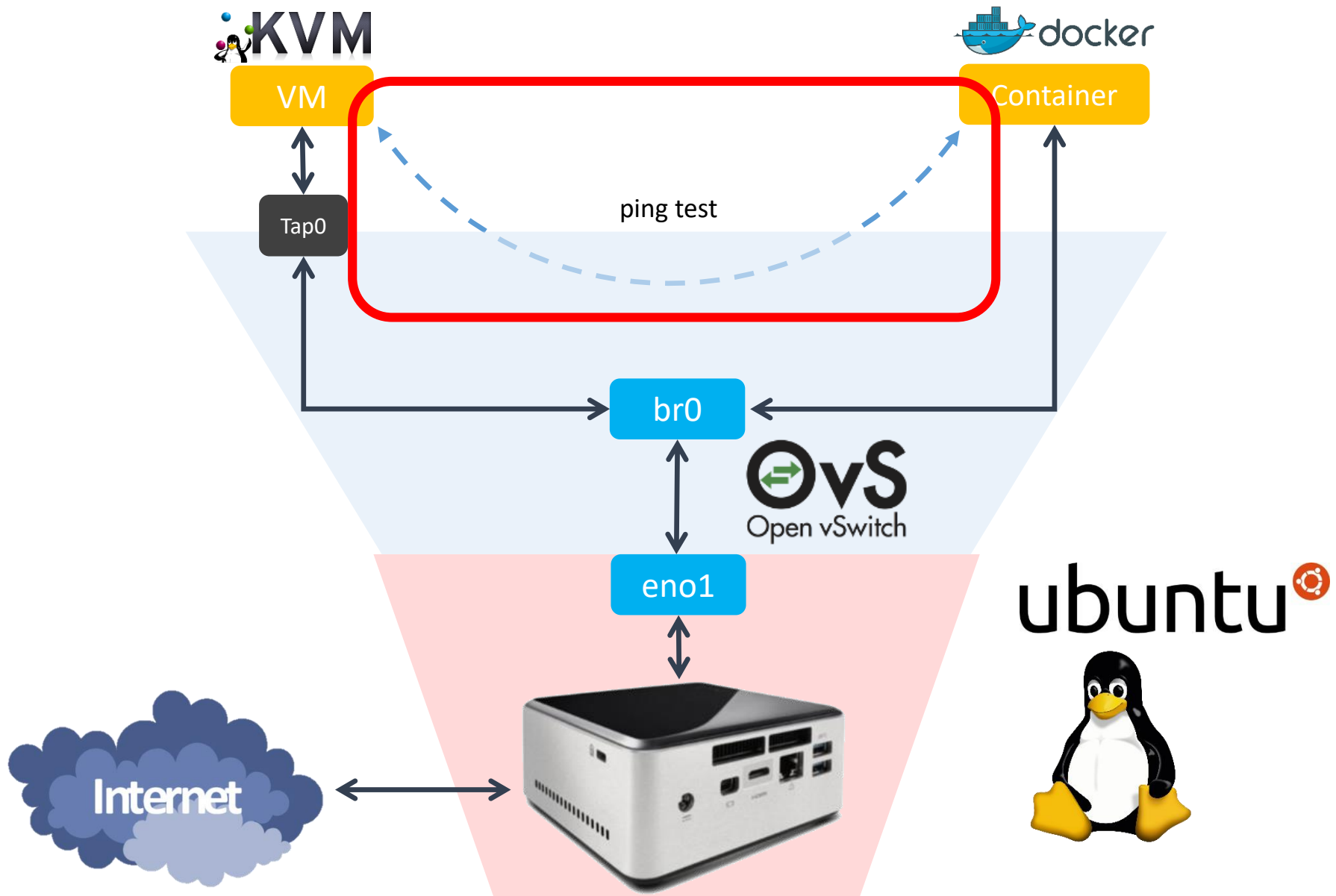
```
$sudo vi /etc/rc.local

#!/bin/bash
docker start [container_name]
ovs-docker del-port br0 veno1 [containerName]
ovs-docker add-port br0 veno1 [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**

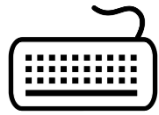
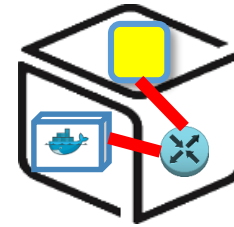
#13 – Check connectivity: VM & Container

Lab #1: Box 51



#13 - 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

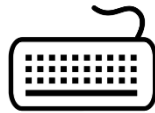
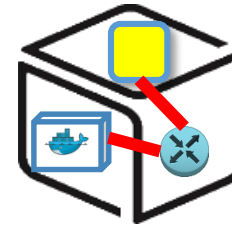
root@b8c3bab8204b:/# ifconfig
eth0      Link encap:Ethernet  HWaddr ae:e5:9c:cc:88:b7
          inet addr:210.125      Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::ace5:9cff:fecc:88b7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:120 errors:0  dropped: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
```

#13 - 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 data:
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 (192.168.0.2) 56(84) bytes of data:
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=1.
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.
64 bytes from 192.168.0.2: 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:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 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)

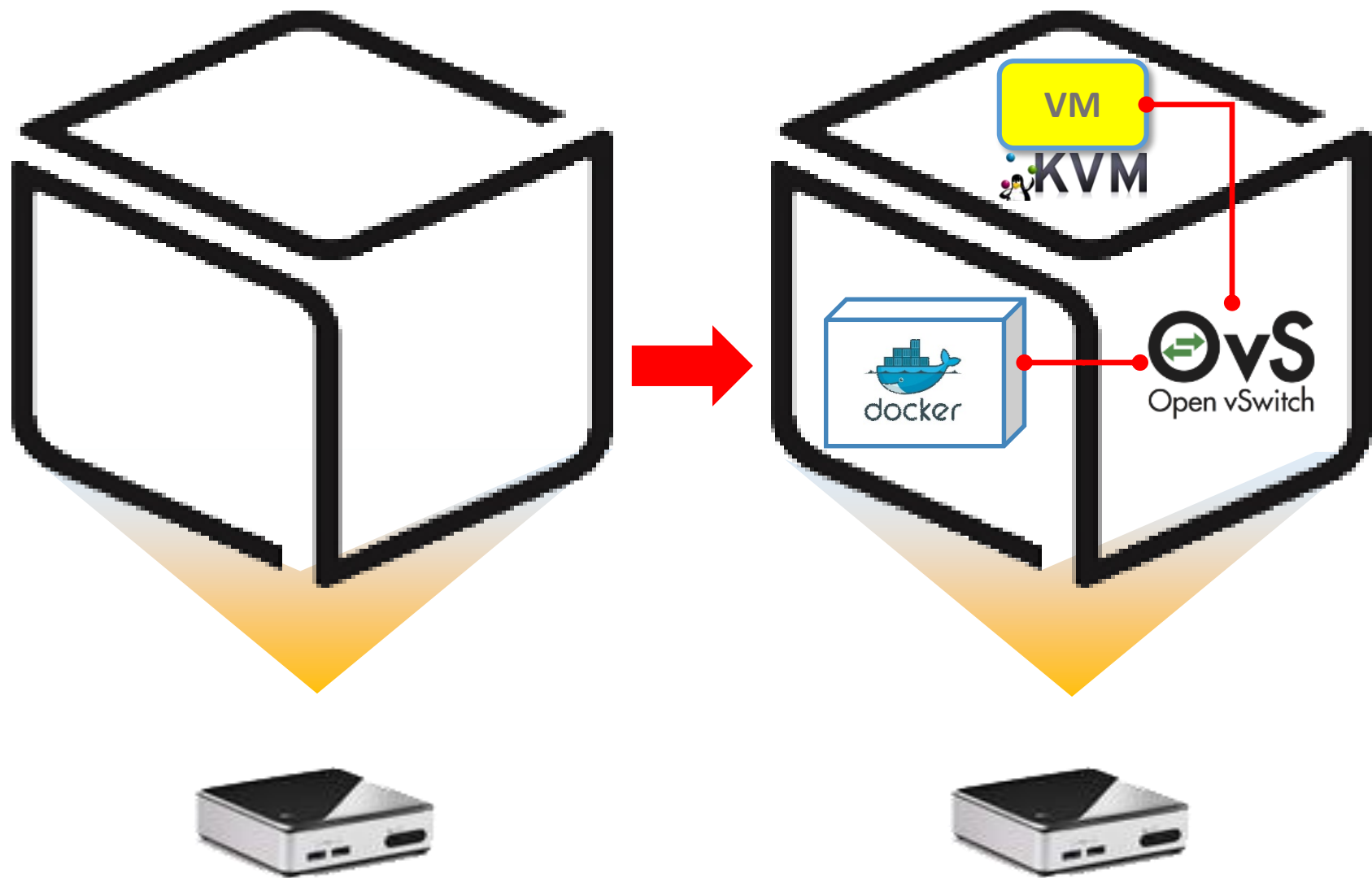
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:38 errors:0 dropped:0 overruns:0 frame:0
          TX packets:38 errors:0 dropped:0 overruns:0 carrier:0
          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 (192.168.0.3) 56(84) bytes of data:
64 bytes from 192.168.0.3: icmp_seq=1 ttl=64 time=0.872 ms
64 bytes from 192.168.0.3: icmp_seq=2 ttl=64 time=0.590 ms
64 bytes from 192.168.0.3: icmp_seq=3 ttl=64 time=0.585 ms
64 bytes from 192.168.0.3: 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:~$

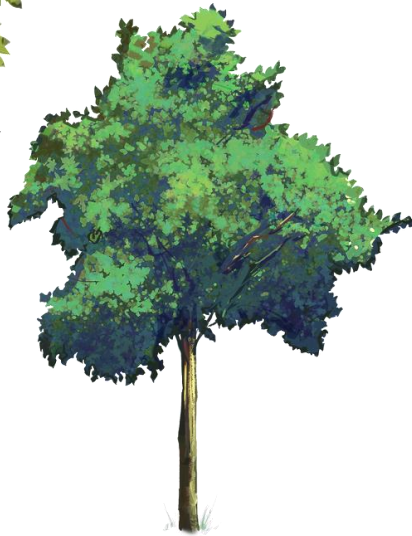
```

Docker container

KVM VM



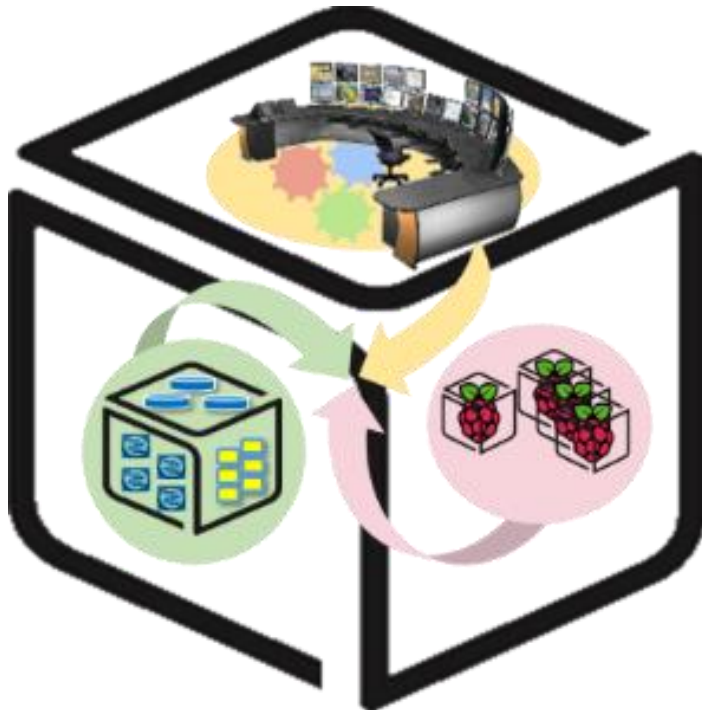
Lab Review



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