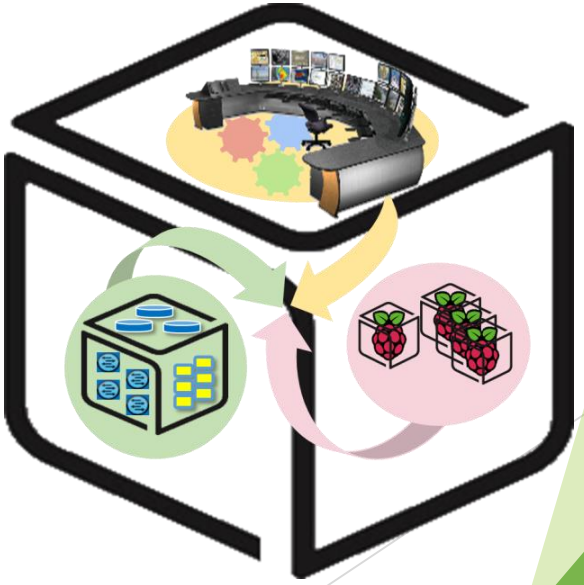
SmartX Labs
for Computer Systems

Box Lab v1.3

(2018, Spring)



NetCS Lab

History and Contributor of Box Lab

2016

Version	Updated Date	Updated Contents	Contributor
V0.1	2015/09	OvS, Docker 자료 초안 작성	배 정 주
V0.2	2015/09	KVM 자료 초안 작성	윤 희 범
V0.3	2015/10	자료 취합 및 Functions Lab 초안 제작	배 정 주
V0.4	2015/10	Functions Lab 자료 편집 및 수정	윤 희 범
V0.5	2016/03	Function Lab → Box Lab 이전 작업, 세부자료 수정	남 택 호
V0.6	2016/04	추가 설명자료 및 단어 교정, History 작성	남 택 호
V0.7	2016/04	Outline 추가, 검수자 피드백 반영	남 택 호
V0.8	2016/04	실습 중 발생한 이슈에 대한 개선사항 추가 (docker container command, if configuration 충돌)	남 택 호
V0.9	2016/06	불필요한 Prerequisite 제거, Appendix 추가 (mirror site 변경코드 제거)	남 택 호 김 병 돈

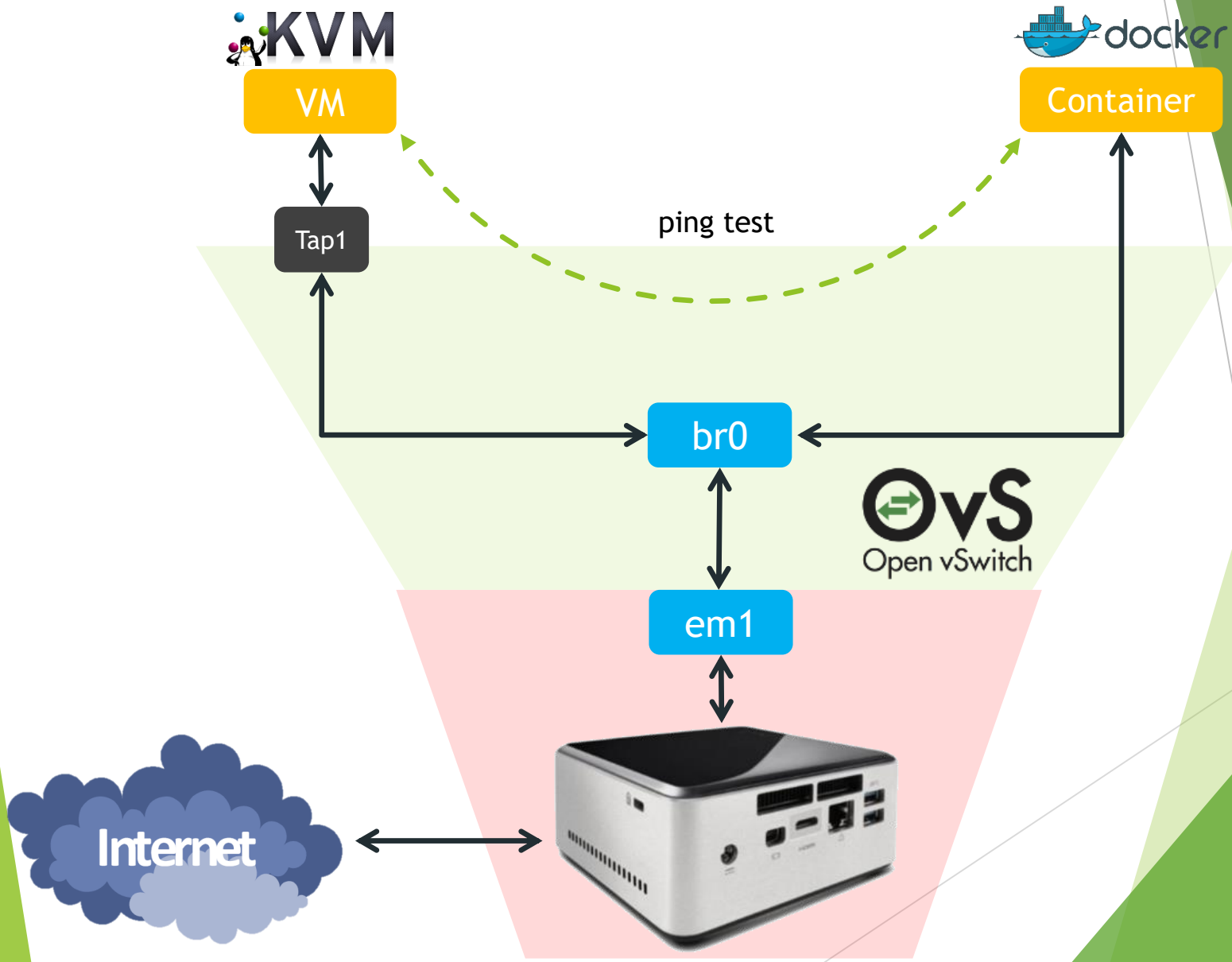
2017

Version	Updated Date	Updated Contents	Contributor
V1.0	2017/01	KVM을 활용한 VM 생성시 사용하는 이미지 주소 변경	남 택 호
V1.1	2017/04	OS version update(16.04), Network setting issue 정리	남 택 호
V1.2	2017/04	KVM VM 생성 후 부팅 문제 해결, 검수 및 테스트 검증	남 택 호

2018

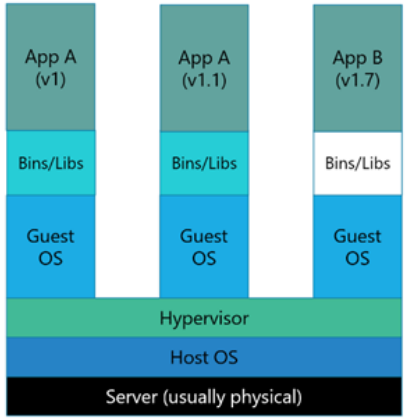
Version	Updated Date	Updated Contents	Contributor
V1.3	2017/04	VM, Container 개념 자료 보강, Ubuntu 16.04 이미지 다운로드 경로 수정, VM OS 설치 과정 수정(Ubuntu image cdrom eject)	권 진 철
V1.4	2017/04	일부 명령어 최신화 및 최적화	강 문 중

Box Lab: Outline

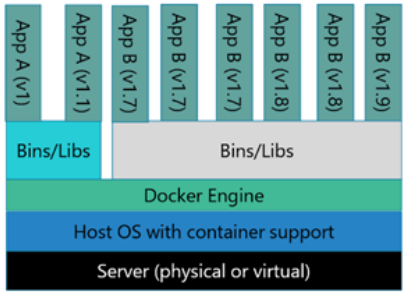


Virtualization: Virtual Machine, Container

Server Virtualisation: Each app and each version of an app has dedicated OS



Containers: All containers share host OS kernel and appropriate bins/libraries



Physical

Applications traditionally built and deployed onto physical systems with 1:1 relationship

New applications often required new physical systems for isolation of resources

Virtual

Higher consolidation ratios and better utilization

Faster app deployment than in a traditional, physical environment

Apps deployed into VMs with high compatibility success

Apps benefited from key VM features i.e., live migration, HA

Physical/virtual

Package and run apps within containers

Key benefits

Further accelerate app deployment

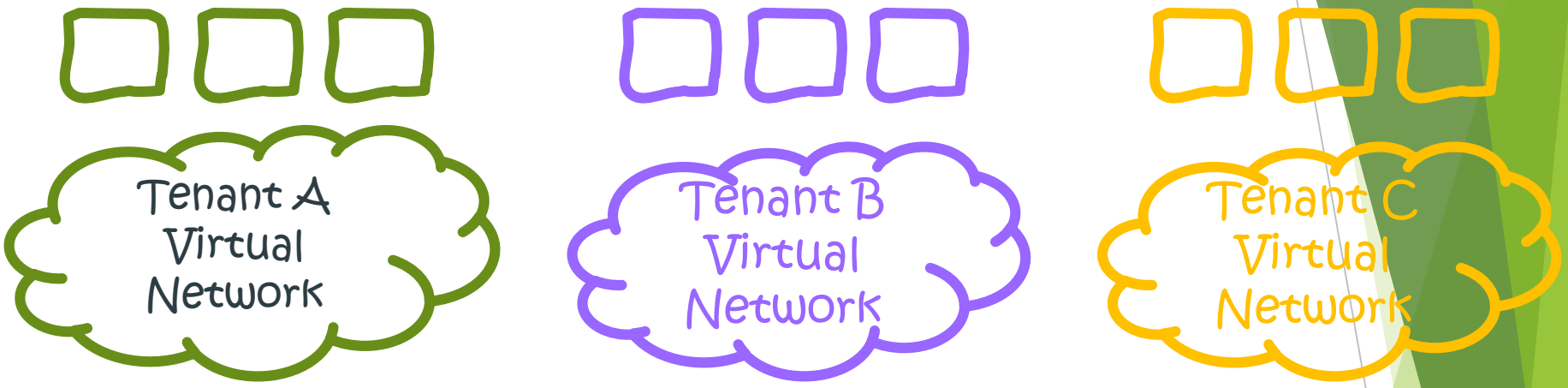
Streamline development and testing

Reduces the "it works on my machine" problem

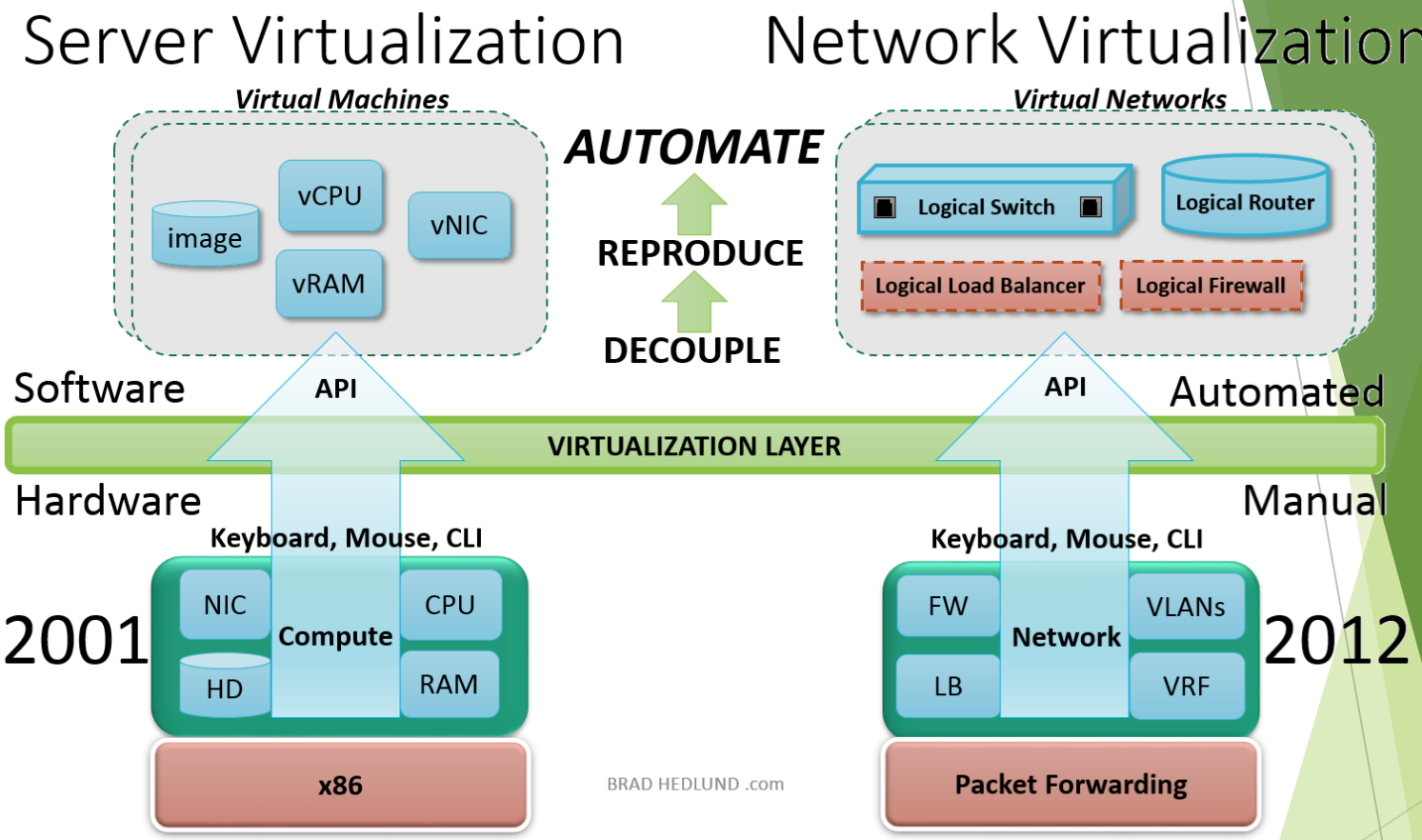
Lower costs associated with app deployment

Increase server consolidation

Virtualization: Basic Concept



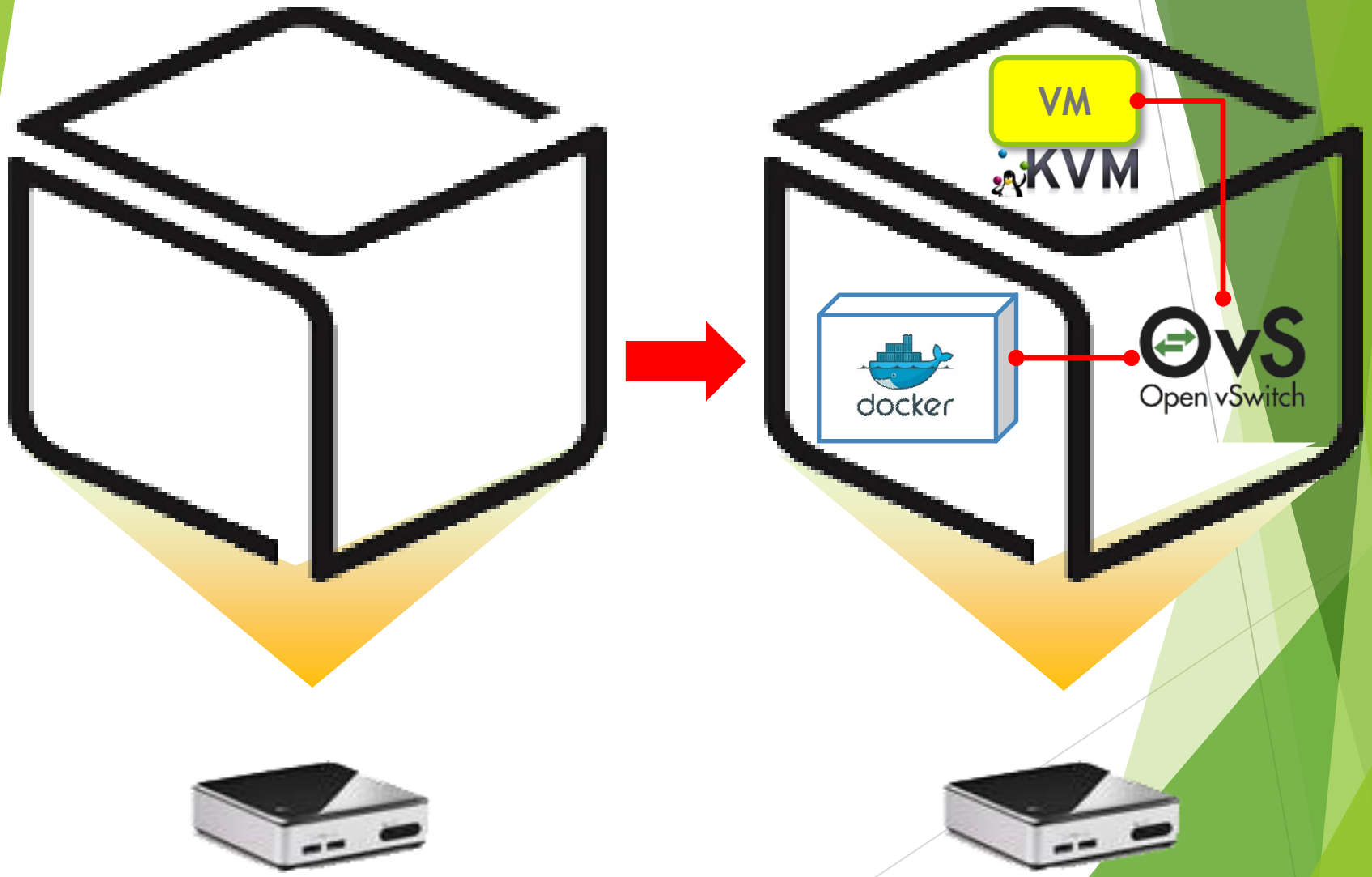
Virtualization: Basic Concept



- Virtualization is the basic act of **decoupling** an infrastructure service from the physical assets on which that service operates. With the virtual network fully decoupled, the physical network configuration is simplified to provide packet forwarding service from one hypervisor to the next

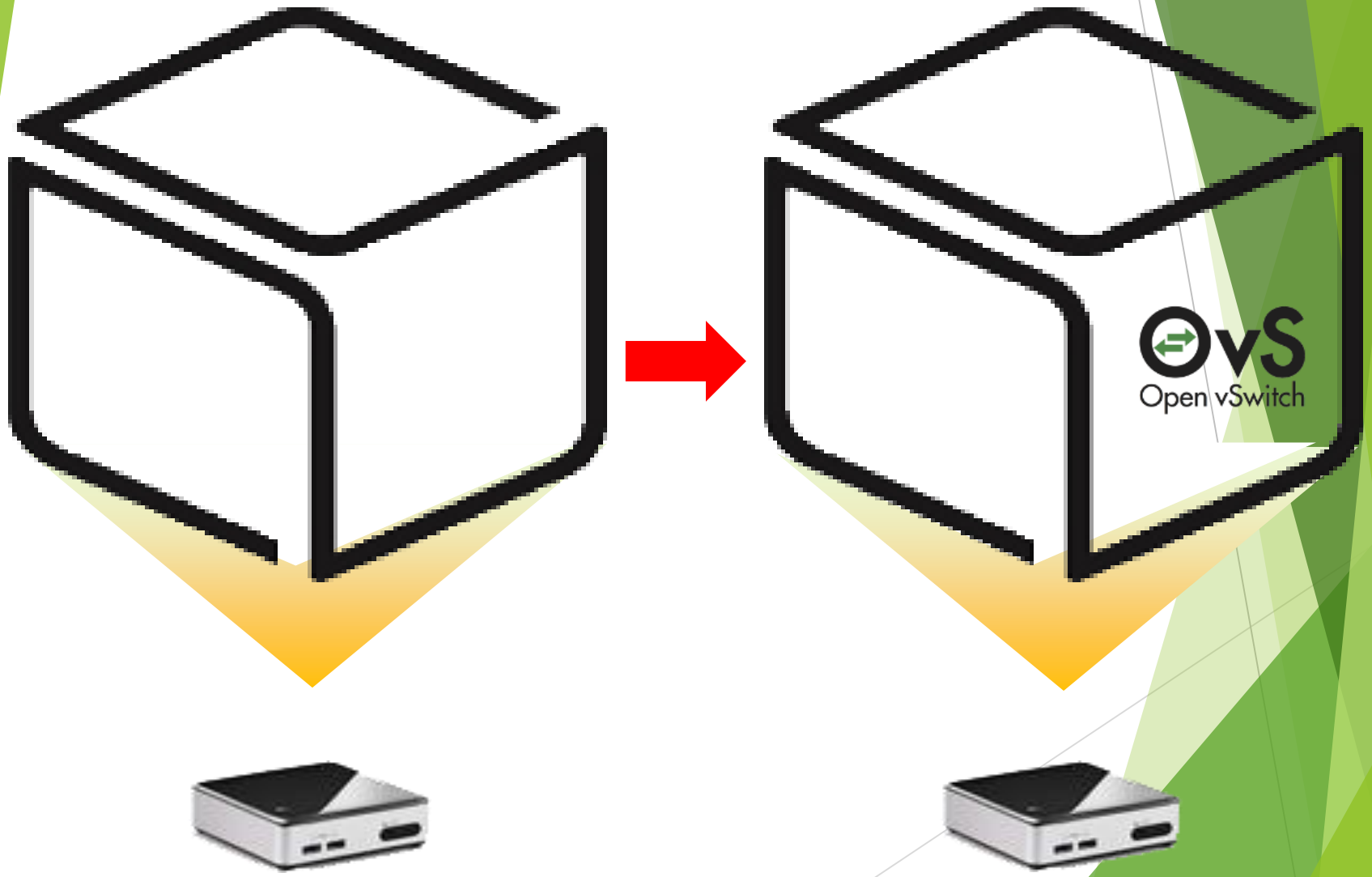
http://bradhedlund.com/2013/05/28/what-is-network-virtualization/

Box Lab: Final Goal



Open vSwitch: Goal of this part

- Update and Configuration



vSwitches: Virtual(soft) Switches

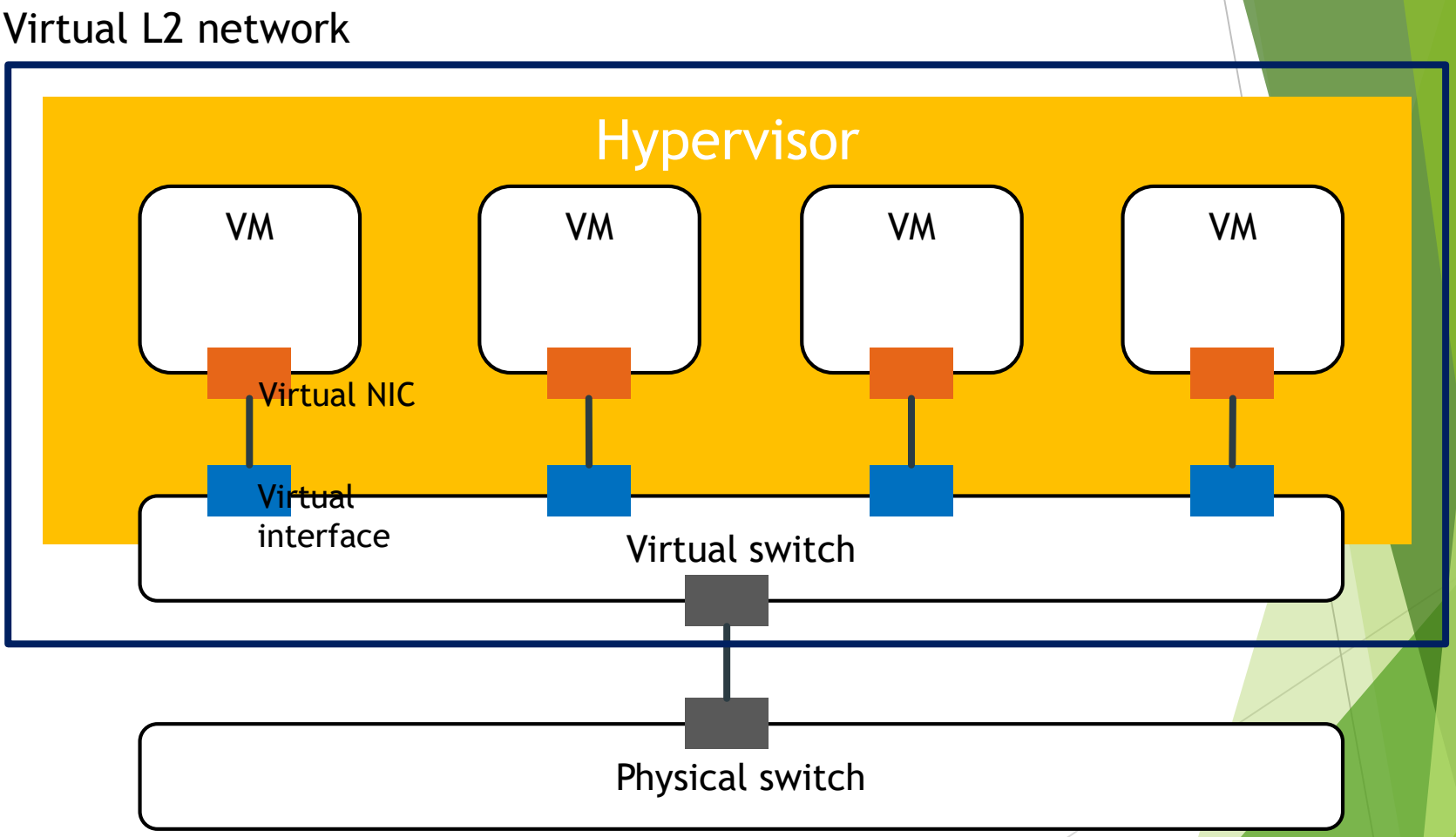
- A switch for every server means a 50:1 (virtual : physical) ratio; Approaching feature-parity with hardware switches: Visibility, ACLs, Quality of Service; Optionally leveraging hardware off-loading
- Tight integration with hypervisor ➔ Good for virtual edge (e.g., inter-VM) networking, network overlay: the leading initial NV use case
- Centralized management

- VMWare, Nicira (open & proprietary), Cisco, IBM, Microsoft
  - VMware vSwitch (vDS), Cisco Nexus 1000V, **Open vSwitch**

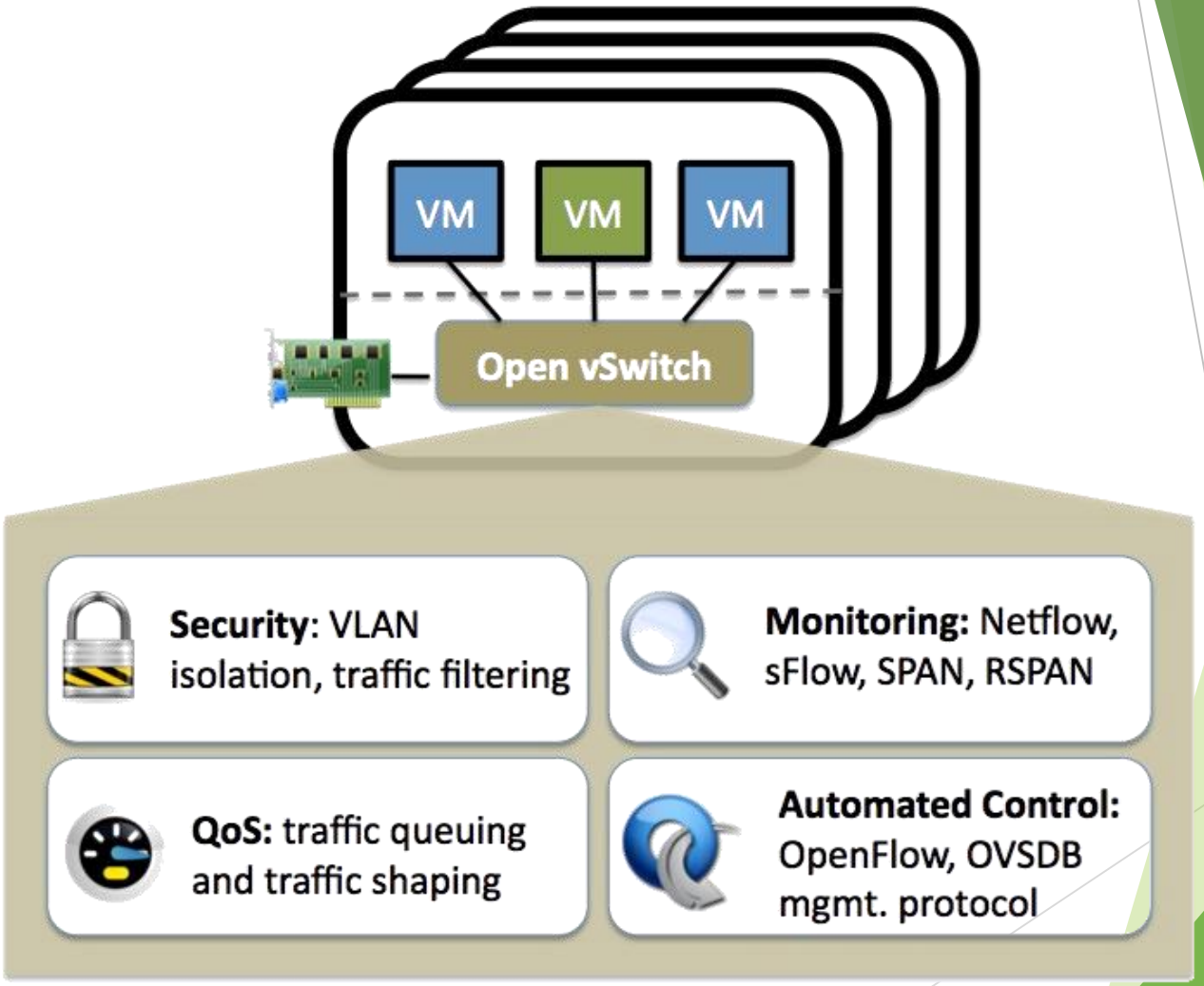
Virtual Switches: Soft vs Hardware

- H/W Switches: With an approach like pass through + tagging + switching in the NIC (with enforcement in the first hop switch); Latency is reduced to the wire speed; Packet classification noticeably outperforms x86-based software switches.
- S/W Switches: Flexibility and upgrade cycle of software + Benefits of virtualization (memory overcommit, page sharing, etc.); Simple resource efficiency
  - Can saturate a 10G link from a guest to the wire with less than a x86 CPU core (assuming MTU size packets), Can saturate a 1G link with less than 20% of a core. In the case of Open vSwitch, these numbers include full packet lookup over L2, L3 and L4 headers.

Virtual Switch: Basic Architecture

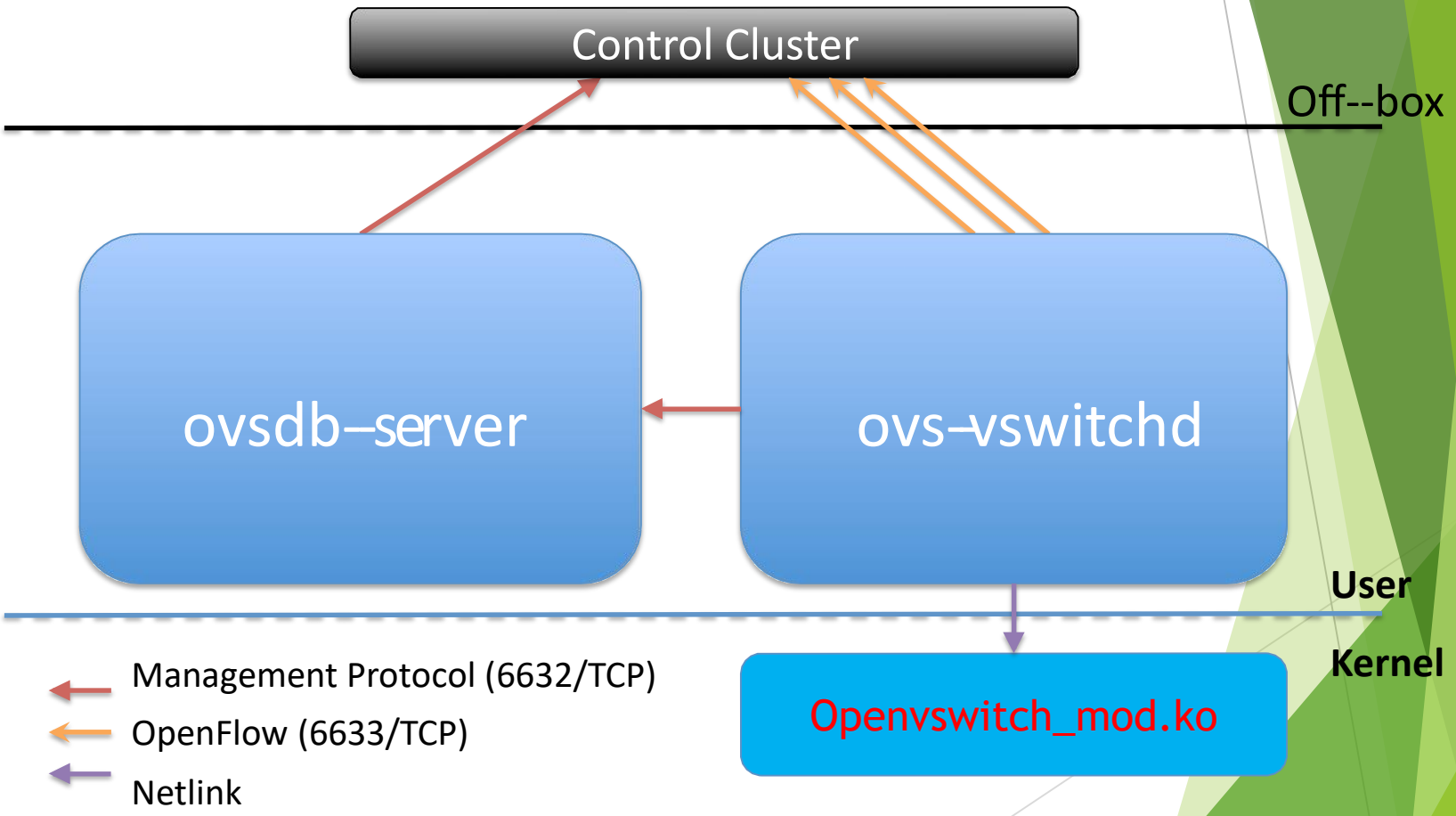


Open vSwitch: Introduction

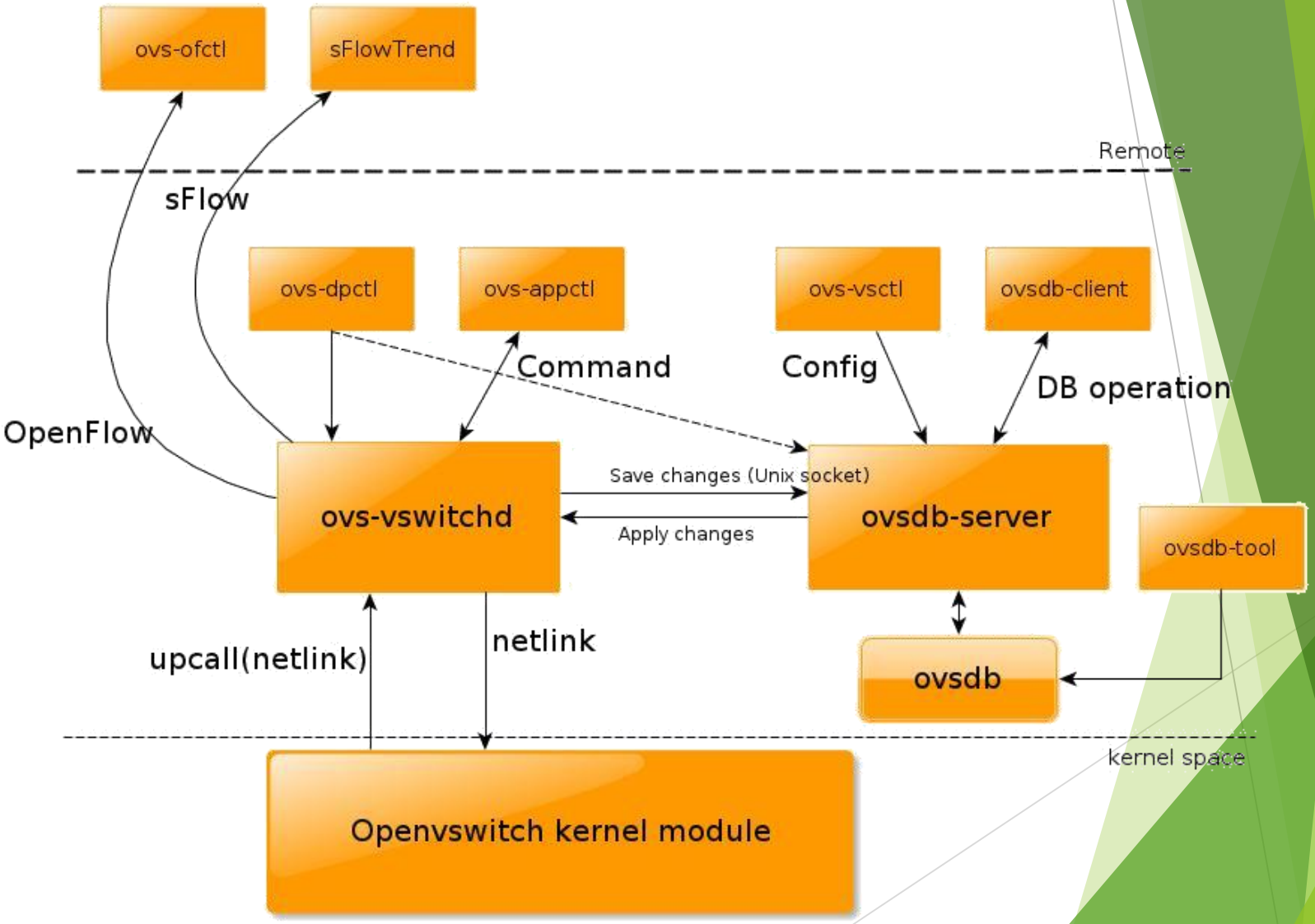


http://openvswitch.org/

Open vSwitch: Main Components



Open vSwitch: All Components



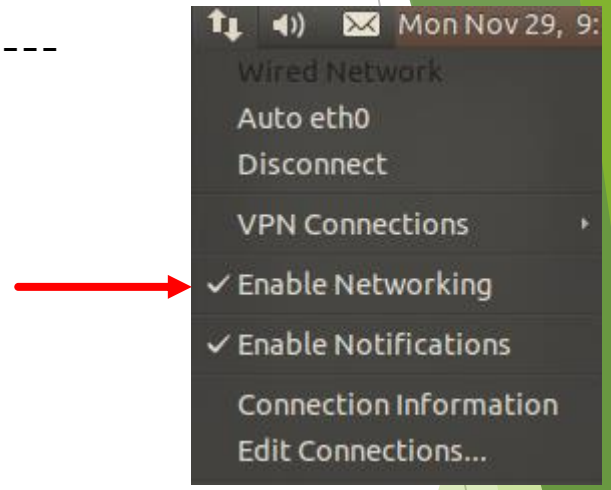
Prerequisite for Box Lab

Installed 64bit Ubuntu 16.04 LTS in Intel NUC
(We recommend to use desktop version)

Prerequisite for Box Lab

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
```

OpenVswitch
- Update & Install

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
```

Open vSwitch
- Basic command (ovs-vsctl)

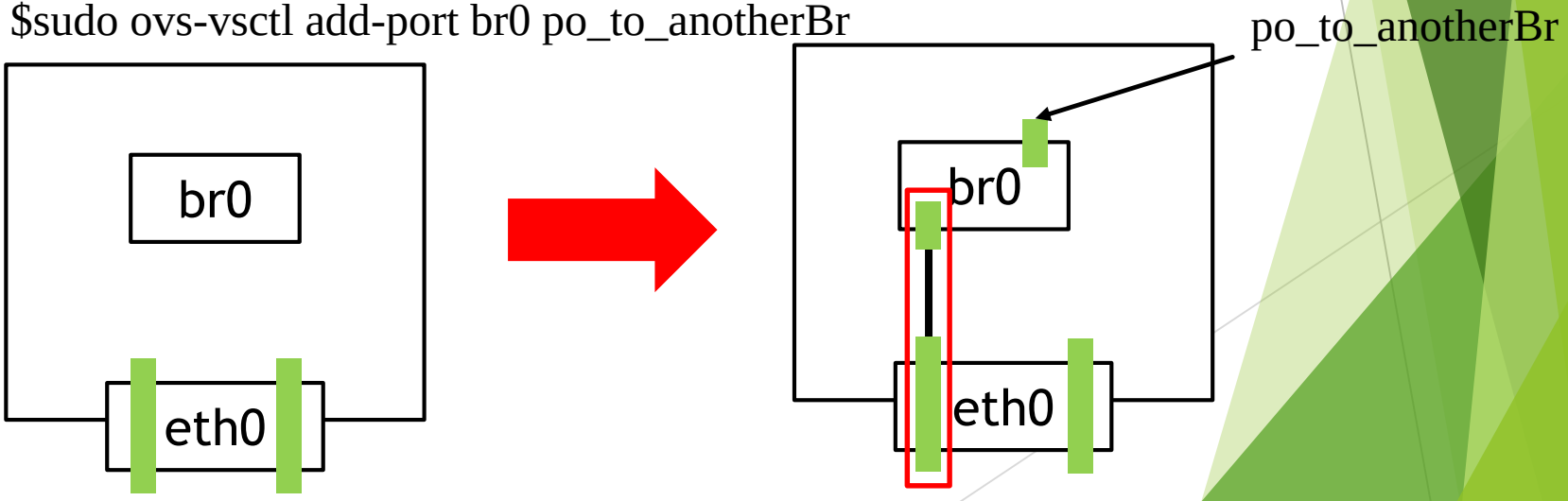
※You don't need to follow below command.

$sudo ovs-vsctl add-br <bridge_name>
$sudo ovs-vsctl add-port <bridge_name> <NIC>
$sudo ovs-vsctl add-port <bridge_name> <port_name>
$sudo ovs-vsctl del-br <bridge_name>
$sudo ovs-vsctl del-port <port_name>

Example)

※You don't need to follow below command.

$sudo ovs-vsctl add-br br0
$sudo ovs-vsctl add-port br0 eth0
$sudo ovs-vsctl add-port br0 po_to_anotherBr

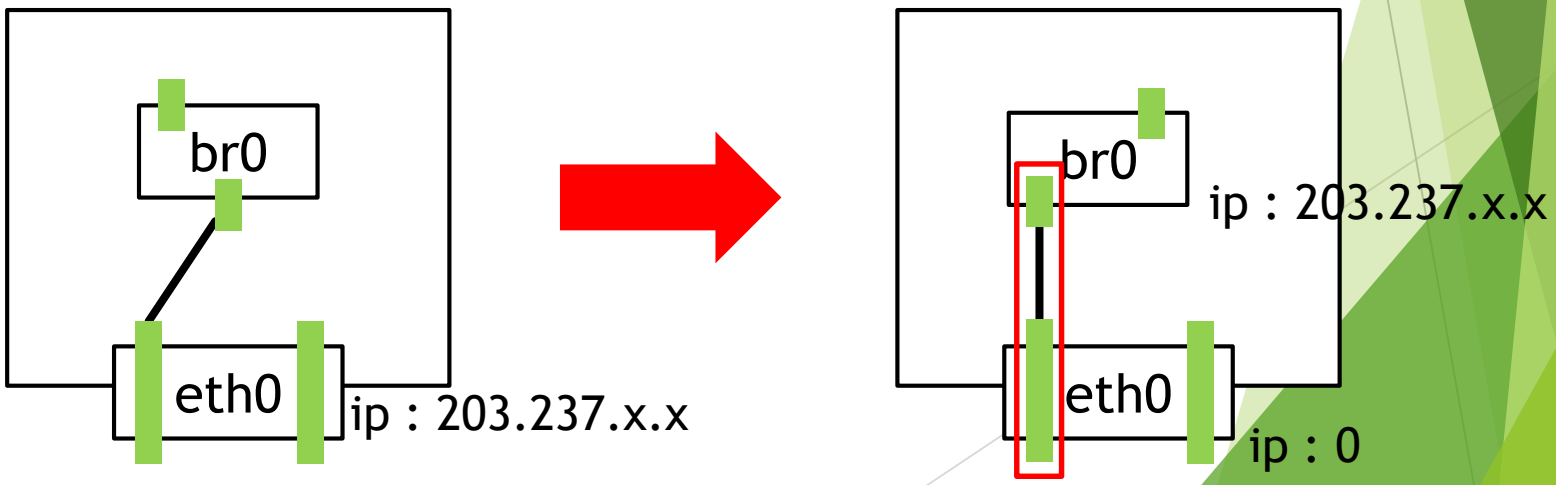


Open vSwitch
- Interface configuration setting

Example)

※You don't need to follow below command.

$sudo ifconfig eth0 0
$sudo ifconfig br0 0
$sudo ifup br0
$sudo ifconfig br0 up
$sudo ifconfig br0 203.237.x.x
$sudo ifconfig eth0 up



Open Vswitch
- Configure bridged networking

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
```

Open Vswitch
- Configure bridged networking

$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"
```

Open Vswitch
- Configure bridged networking

$sudo ovs-vsctl add-port br0 eth0

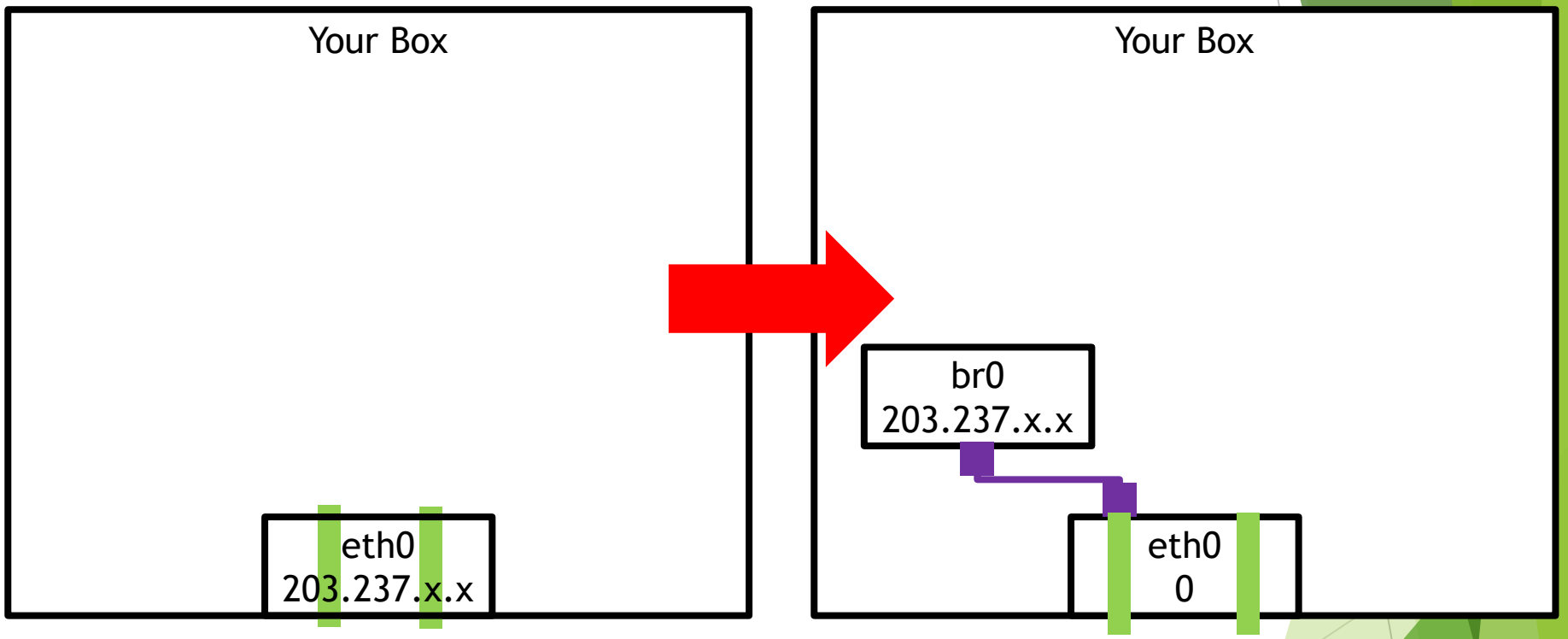
```
root@nuc:~# ifconfig
br0       Link encap:Ethernet  HWaddr 86:f9:ed:3c:74:42
          inet addr:210.125          Bcast:210.125.84.255  Mask:255.255.255.0
          inet6 addr: fe80::fccc:4fff:fe23:4e1c/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 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

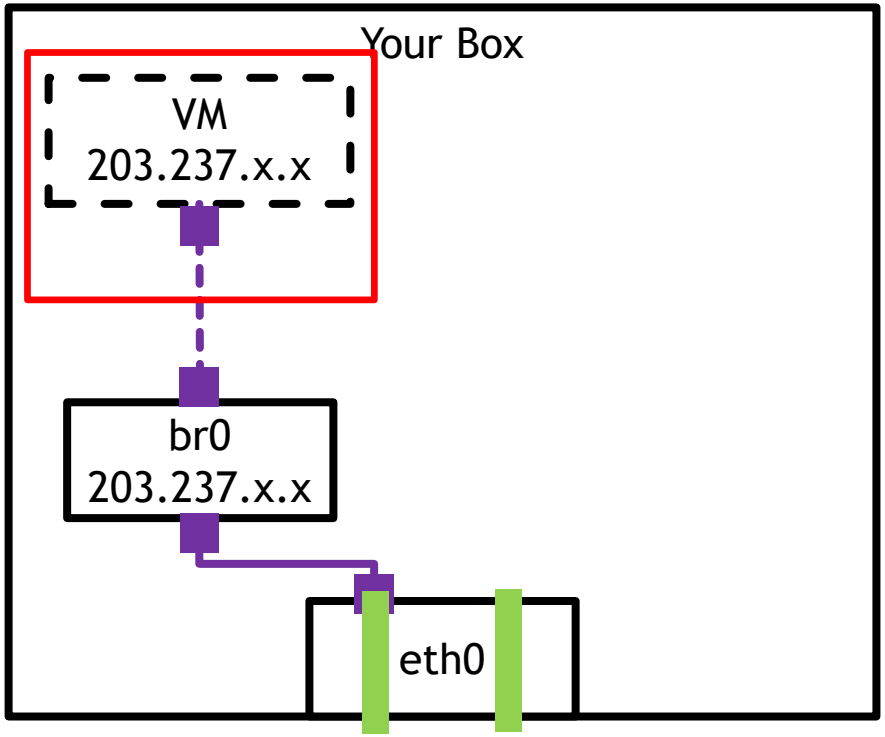
Open Vswitch
- Configure bridged networking

```
$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
```

Open Vswitch
- Situation

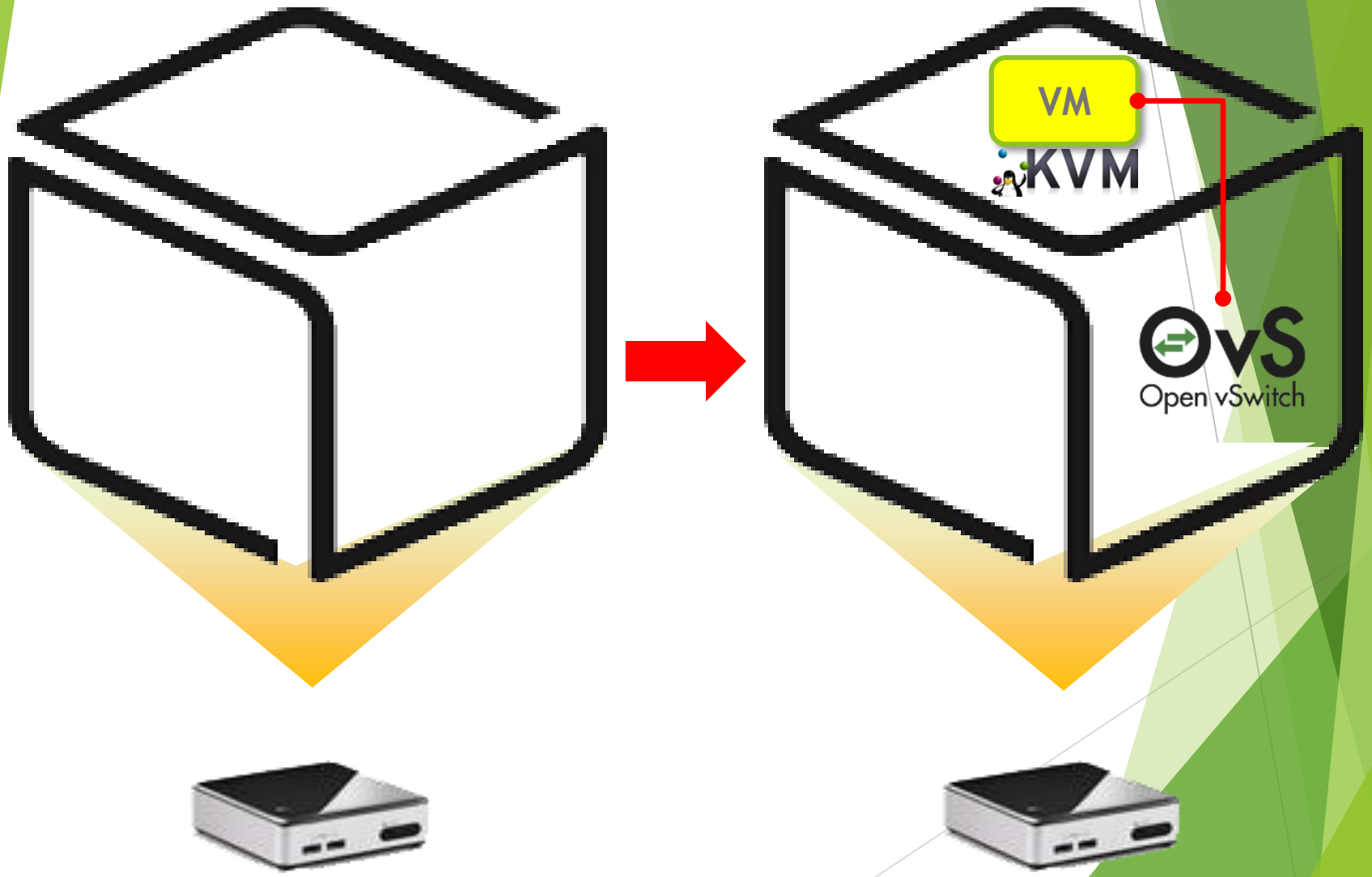


Open Vswitch
- Goal of this section



Stop OvS setting, Now need to build KVM

KVM: Goal of this part



What is KVM?

- KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko.
- Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc.
- KVM is open source software. The kernel component of KVM is included in mainline Linux, as of 2.6.20. The userspace component of KVM is included in mainline QEMU, as of 1.3.

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://kr.archive.ubuntu.com/ubuntu-releases/16.04/ubuntu-16.04.4-server-amd64.iso
```

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

Open Vswitch
- Configure bridged networking

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)

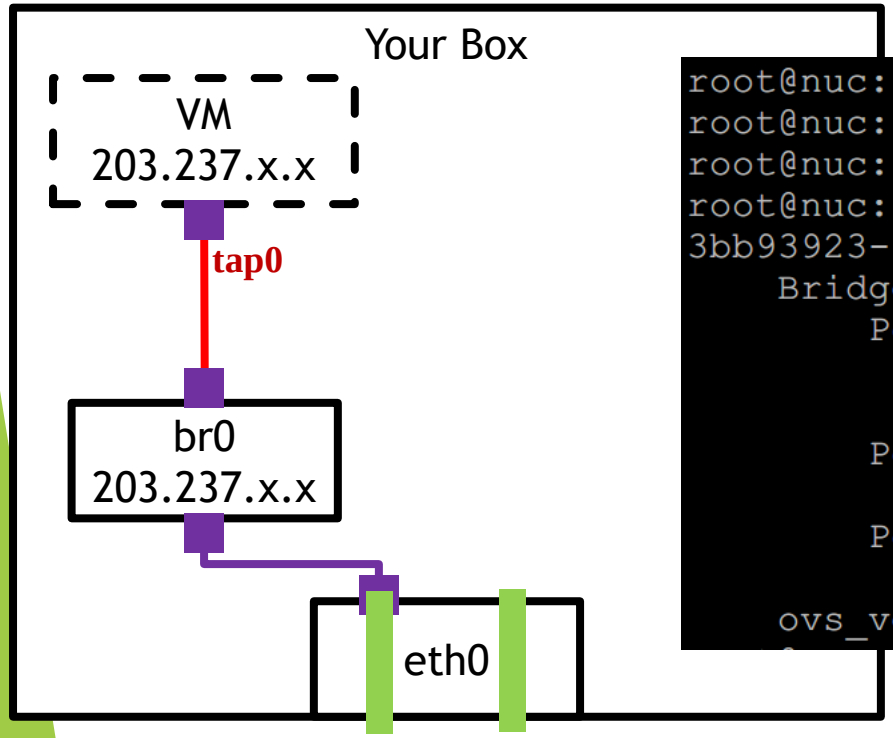
```
auto tap0
iface tap0 inet manual
        pre-up ip tuntap add tap0 mode tap
        up ip link set dev tap0 up
        post-down ip link del dev tap0
```

Open Vswitch
- Configure bridged networking

Turn on the tap interface and attach it to your VM.

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

This tap device will be attached VM. 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"
```

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

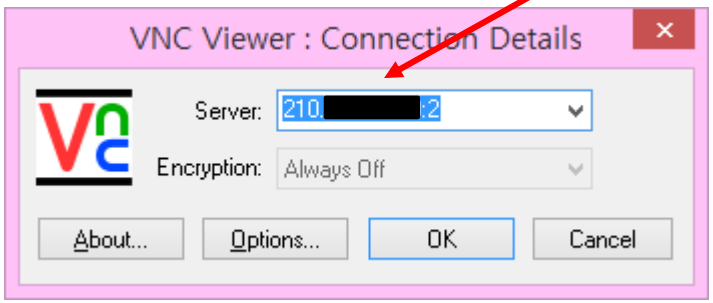
$ sudo kvm -m 512 -name tt -smp cpus=2,maxcpus=2 -device virtio-net-pci,netdev=net0 -netdev tap,id=net0,ifname=vport_vFunction,script=no -boot d vFunction20.img -cdrom ubuntu-16.04.4-server-amd64.iso -vnc :5 -daemonize -monitor telnet:127.0.0.1:3010,server,nowait,ipv4

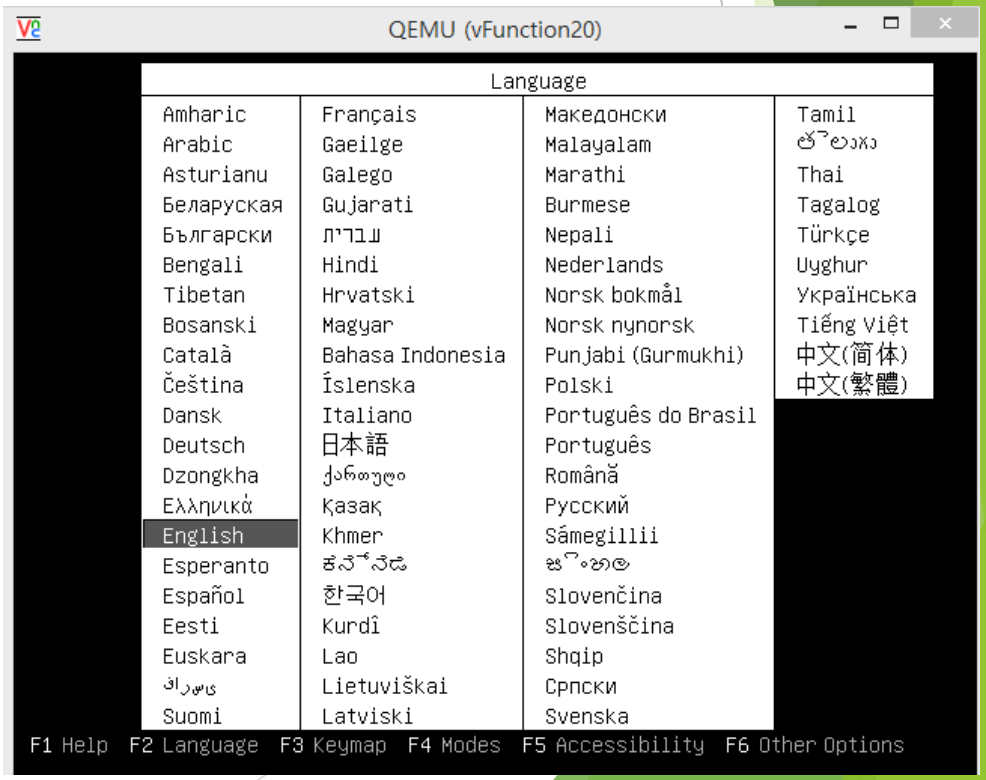
Install VNC viewer and see inside of VM

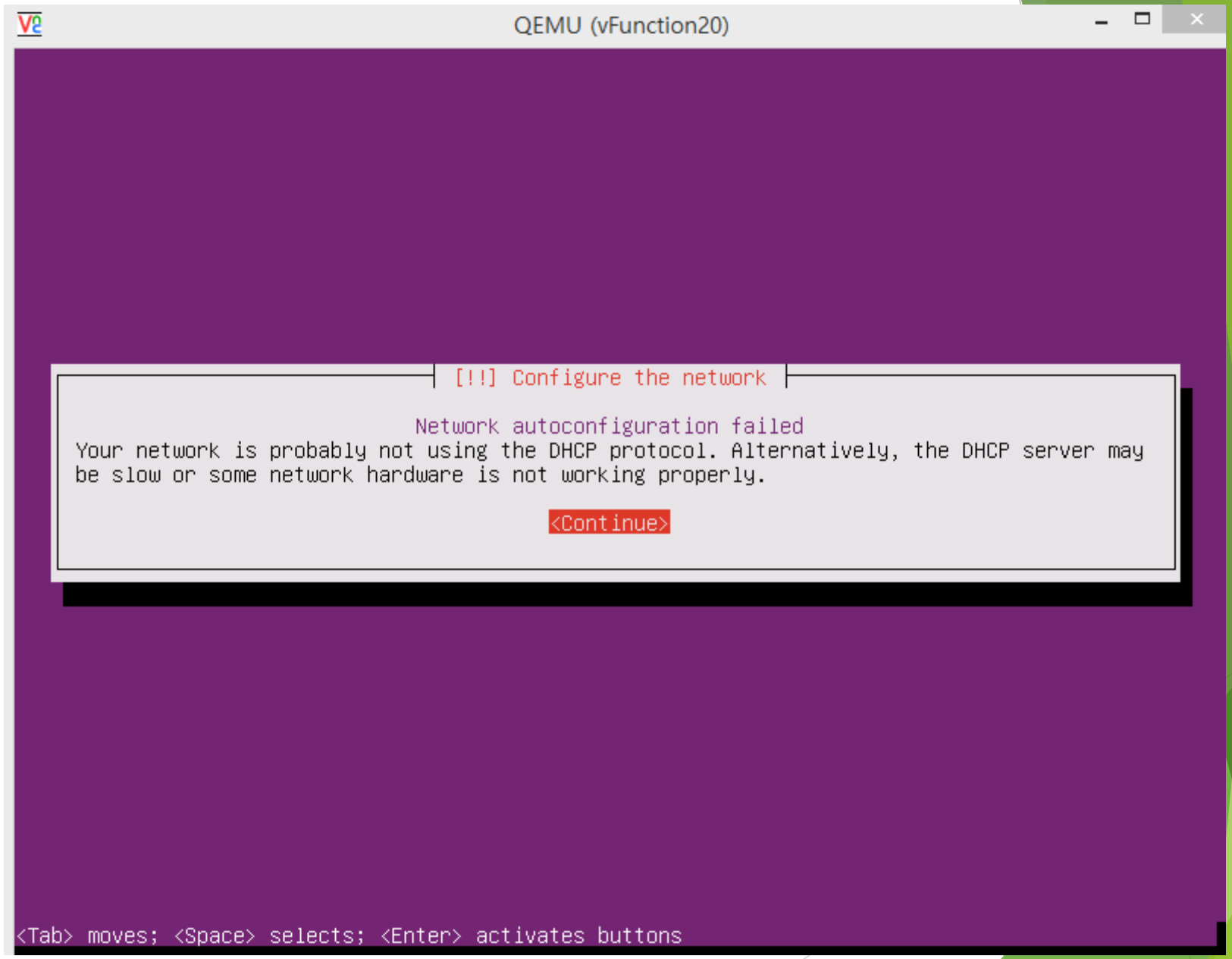
$sudo apt-get install xvnc4viewer
$xvnc4viewer localhost :5

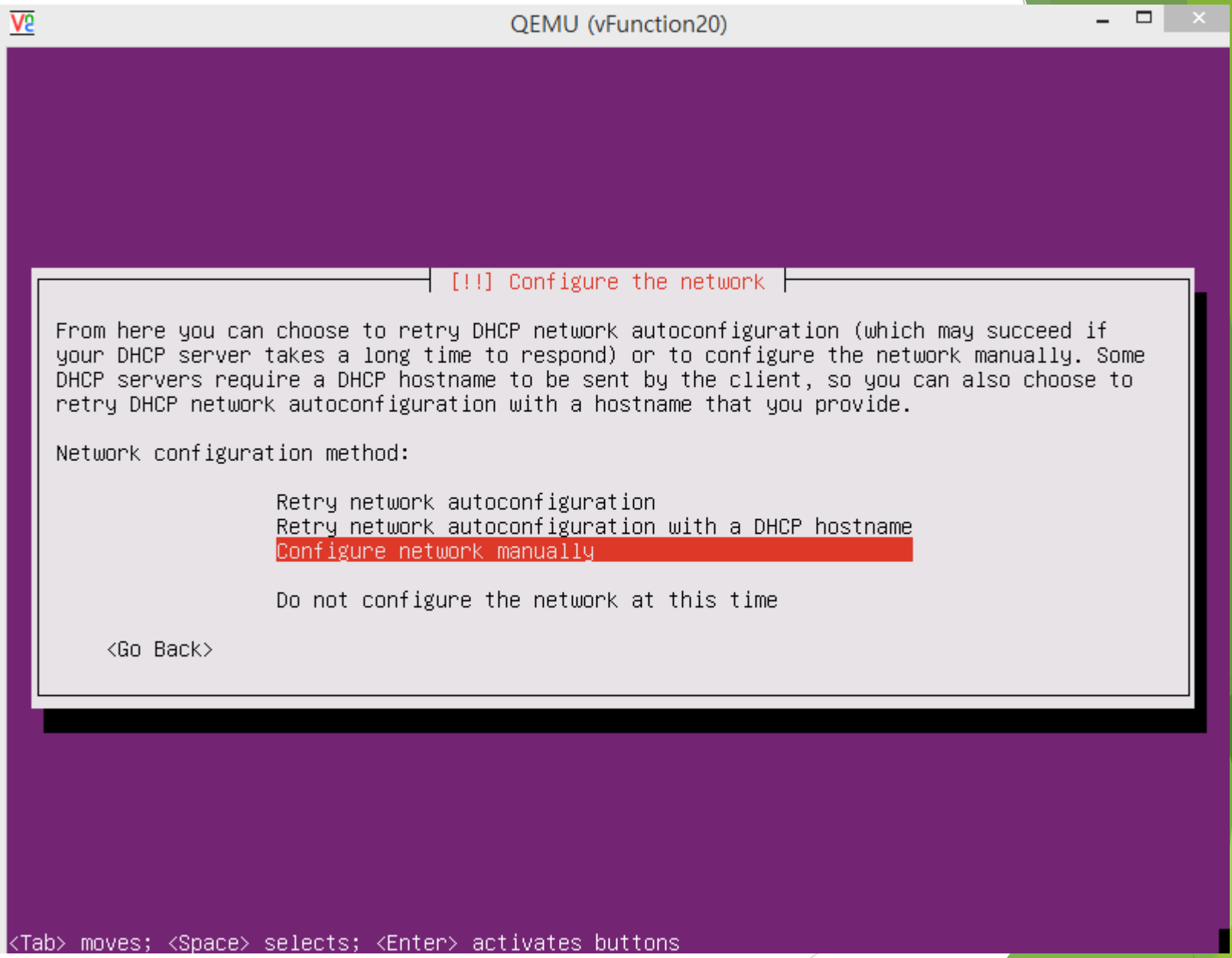
KVM
- Install Ubuntu to VM

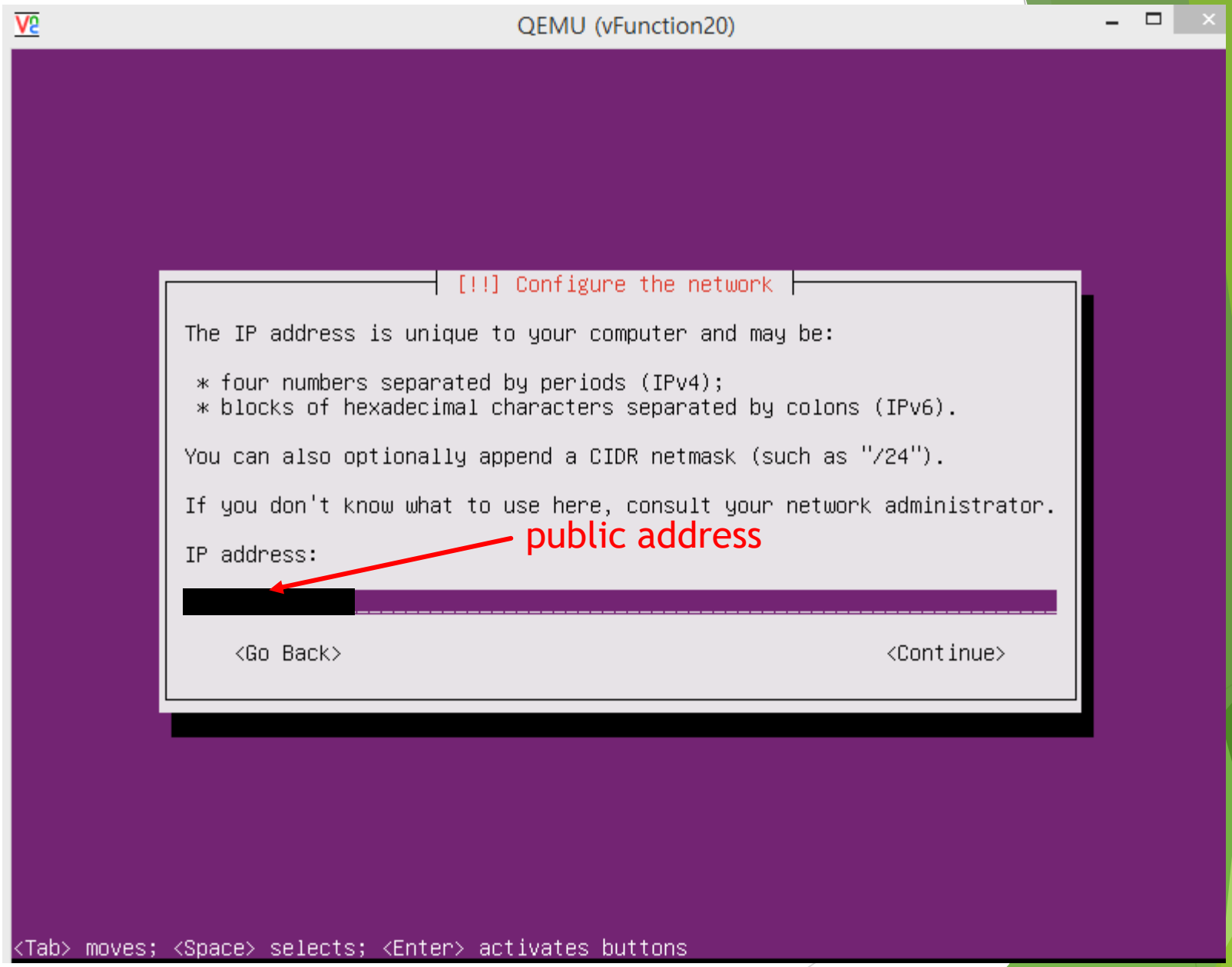
IP address:vnc number
ex) 210.203.x.x:5

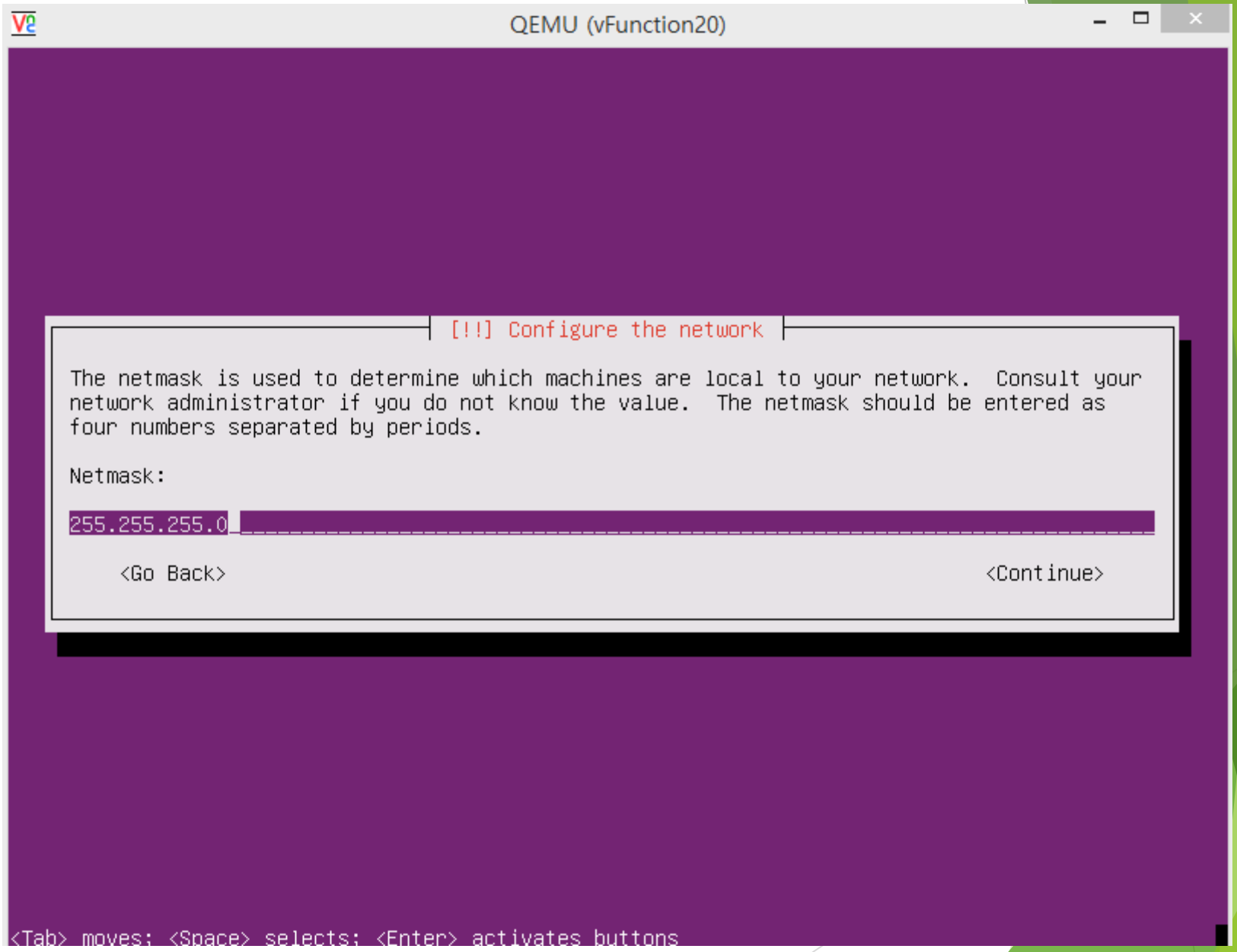




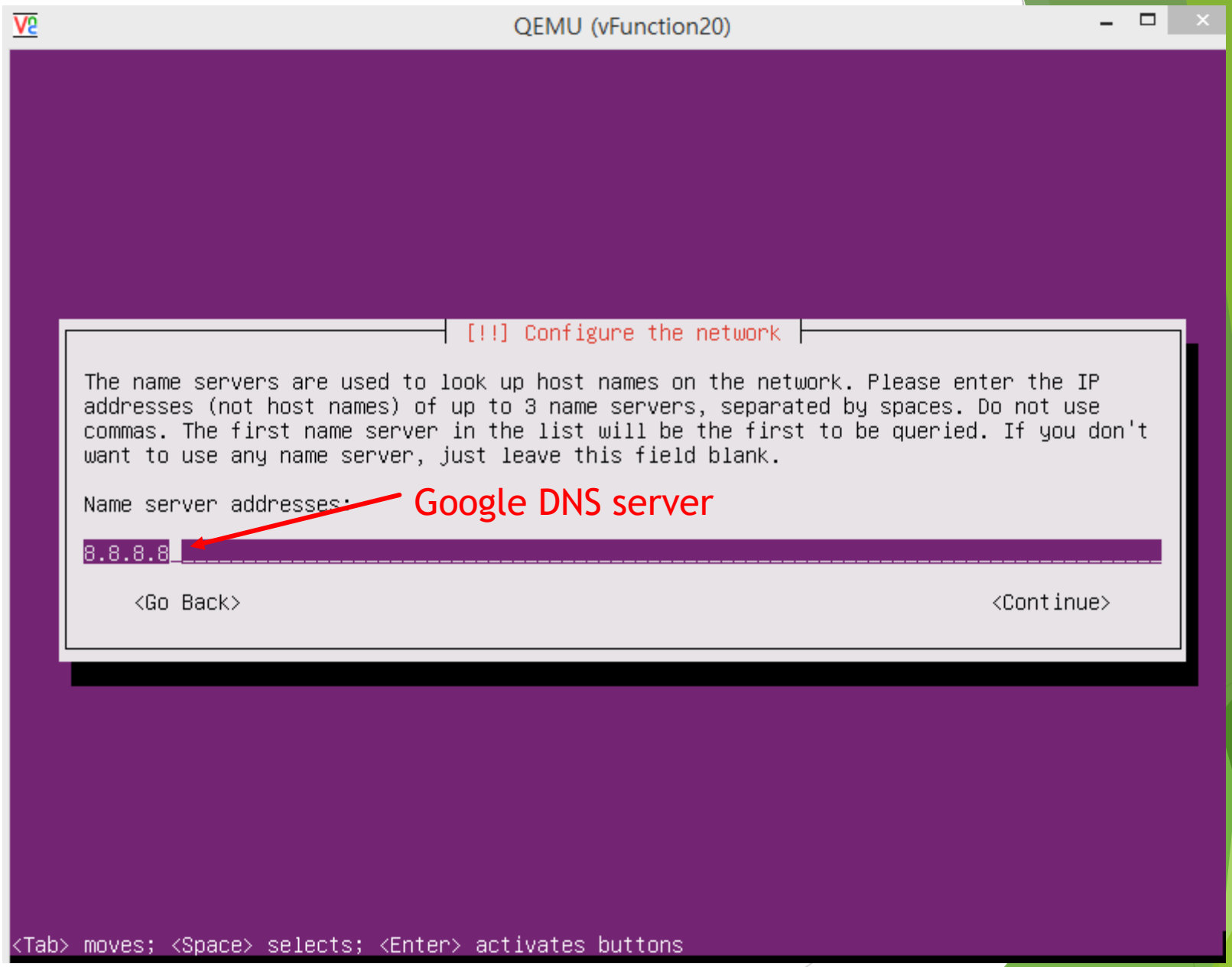

QEMU (vFunction20)
[!!] Configure the network
Network autoconfiguration failed
Your network is probably not using the DHCP protocol. Alternatively, the DHCP server may be slow or some network hardware is not working properly.
<Continue>
<Tab> moves; <Space> selects; <Enter> activates buttons


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.
IP address:
public address
<Go Back>
<Continue>
<Tab> moves; <Space> selects; <Enter> activates buttons

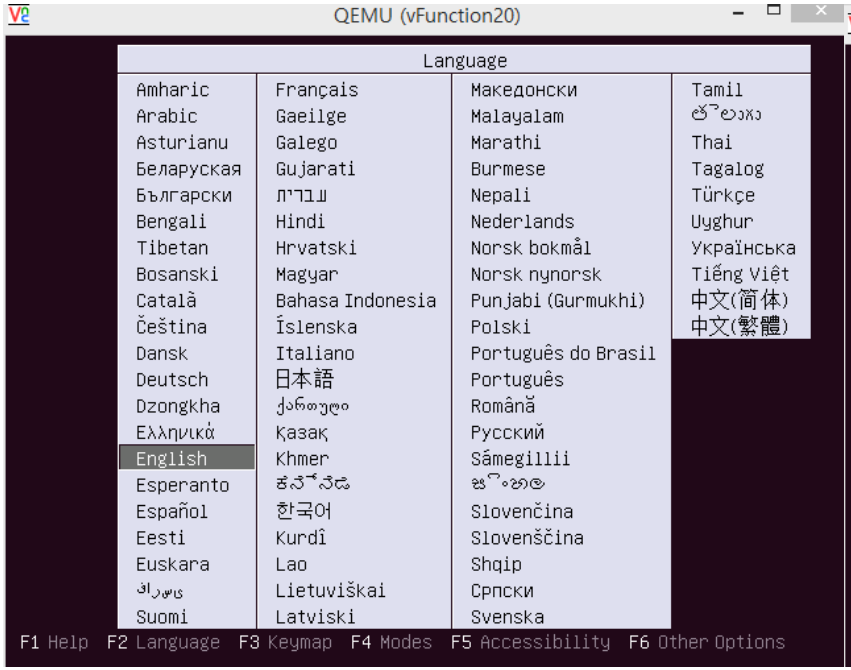

QEMU (vFunction20)
[!!] Configure the network
The netmask is used to determine which machines are local to your network. Consult your network administrator if you do not know the value. The netmask should be entered as four numbers separated by periods.
Netmask:
255.255.255.0
<Go Back>
<Continue>
<Tab>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

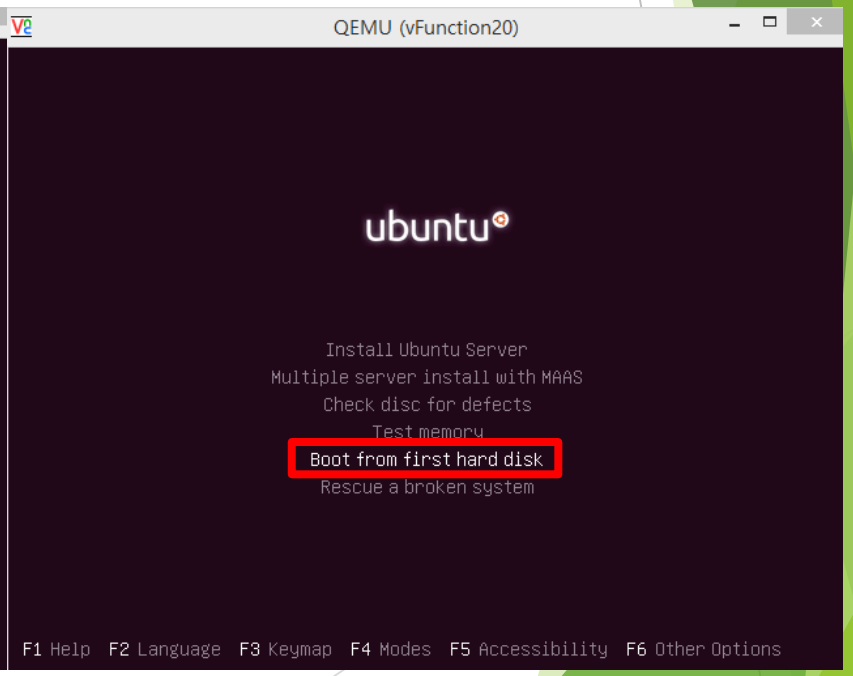

QEMU (vFunction20)
[!!] Configure the network
The gateway is an IP address (four numbers separated by periods) that indicates the gateway router, also known as the default router. All traffic that goes outside your LAN (for instance, to the Internet) is sent through this router. In rare circumstances, 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

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

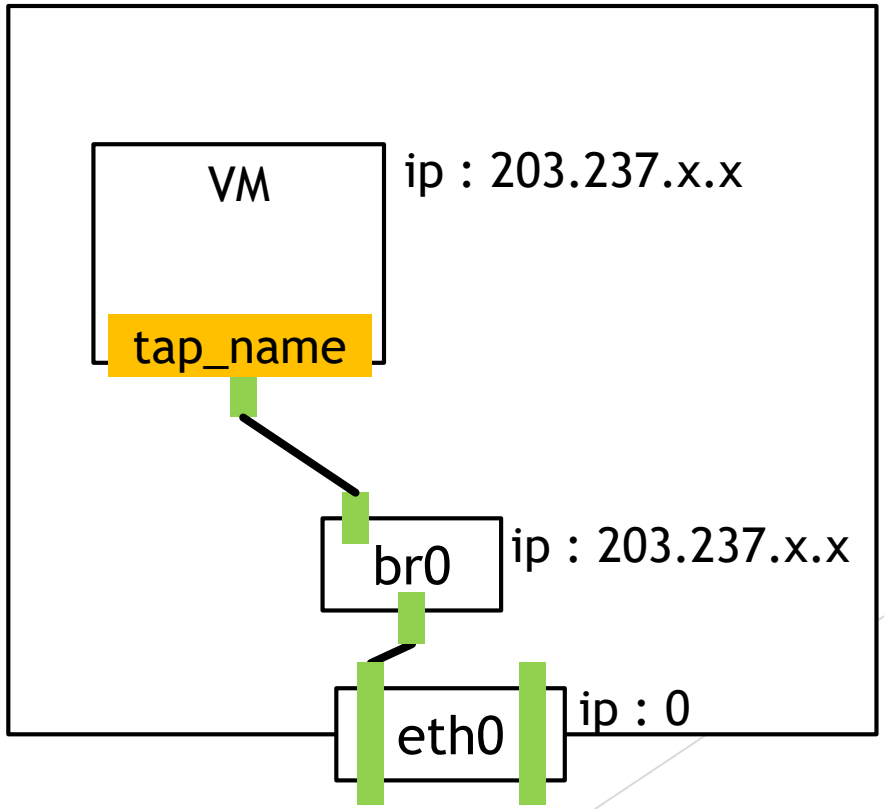
KVM
- VM boot command

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

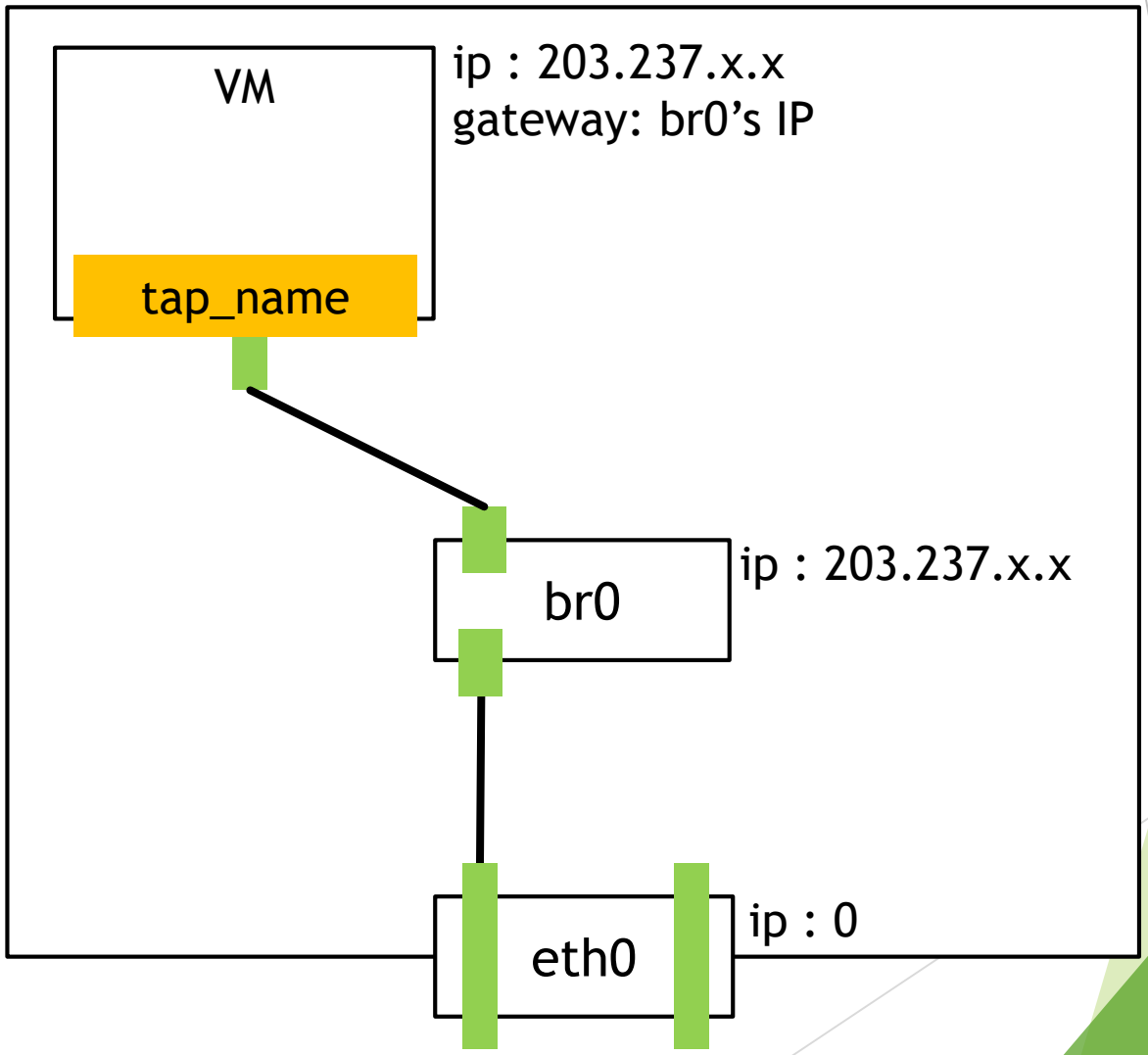
$sudo kvm -m [memory capacity] -name [name] -smp cpus=[#cpu],maxcpus= [#maxcpu] -device virtio-net-pci,netdev=net0,mac= [EE:EE:EE:EE:EE:EE] –netdev tap,id=net0,ifname= [tap_name],script=no -boot d [name].img -vnc : [#] -daemonize

Open Vswitch connects with KVM
- 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"
```



Open Vswitch connects with KVM
- Situation



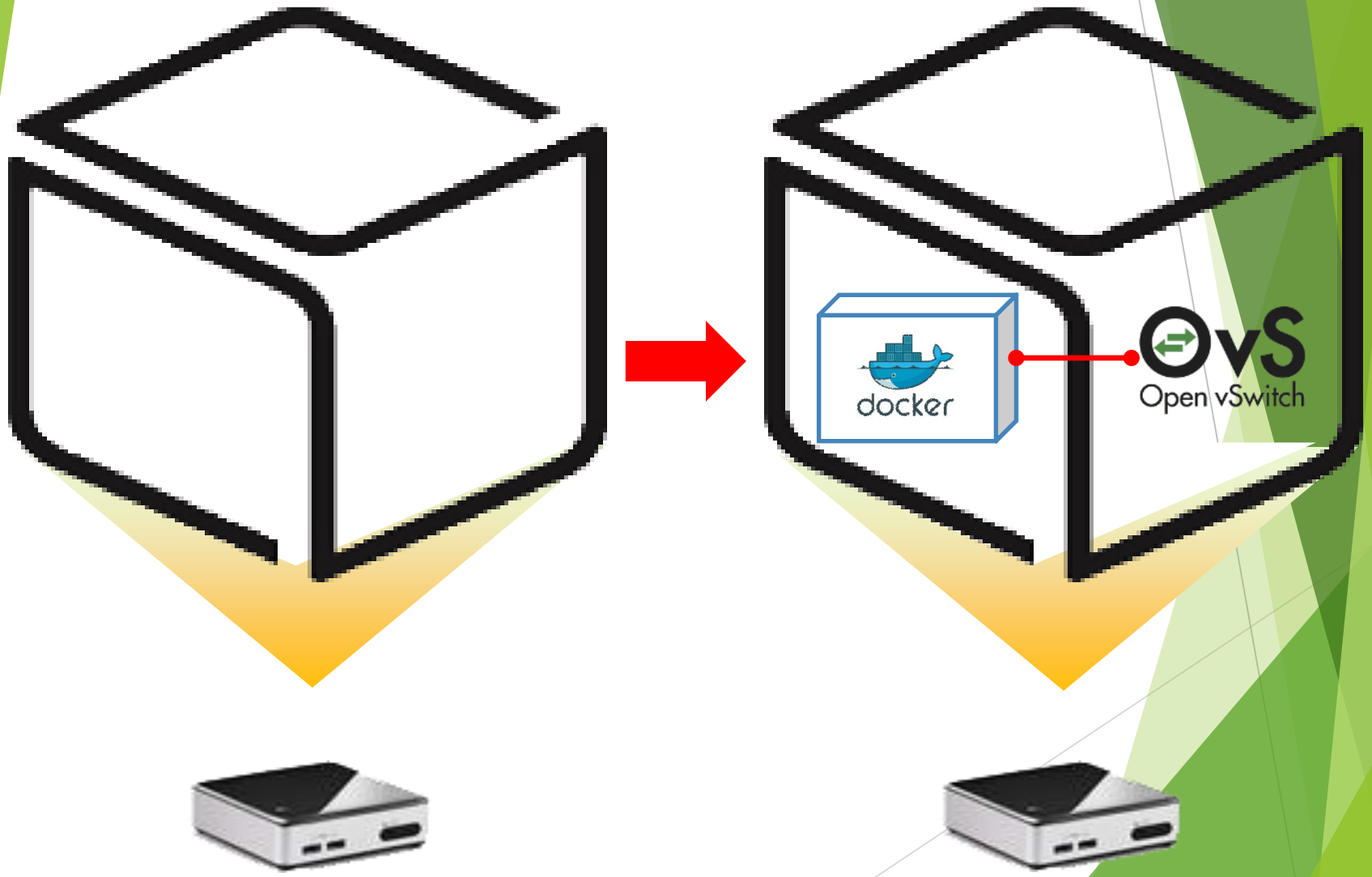
KVM
- Don’t forget to install ssh in VM!

In VMs,

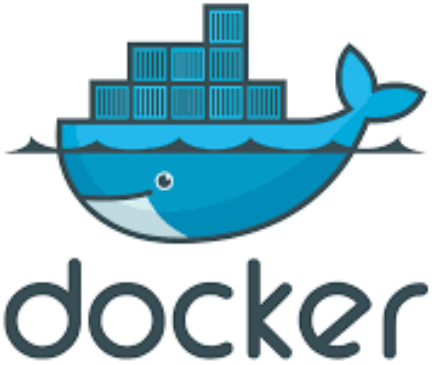
$sudo apt-get update
$sudo apt-get install ssh

```
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: Goal of this part



What is Docker ?



- Docker is an open platform for building, shipping and running distributed applications. It gives programmers, development teams and operations engineers the common toolbox they need to take advantage of the distributed and networked nature of modern applications.

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/

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/
```

Docker
- Make containers

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

Docker
- About [--net host] option

- Pros
  - Easy to use
- Cons
  - Security problem (Violate docker's strong point:isolated)
- Solution?
  - Establish L2 tunneling
    (Can easily achieved by using ovs.
    However, ovs doesn't support raspberry pi officially.
    That's why we are using this option.)
    Related keyword : GRE, vlan, vxlan

Docker
- Connect docker container to 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

Modify
- /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

Docker
- Check connectivity

```
root@nuc:/usr/bin# ovs-docker add-port br0 eth0 docker1 --ipaddress=210.125.84.70/24 --gateway=210.125.84.1
root@nuc:/usr/bin# docker attach docker1

root@b8c3bab8204b:/# ifconfig
eth0      Link encap:Ethernet  HWaddr ae:e5:9c:cc:88:b7
          inet addr:210.125.84.70  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
```

Docker connect with KVM - Check connectivity

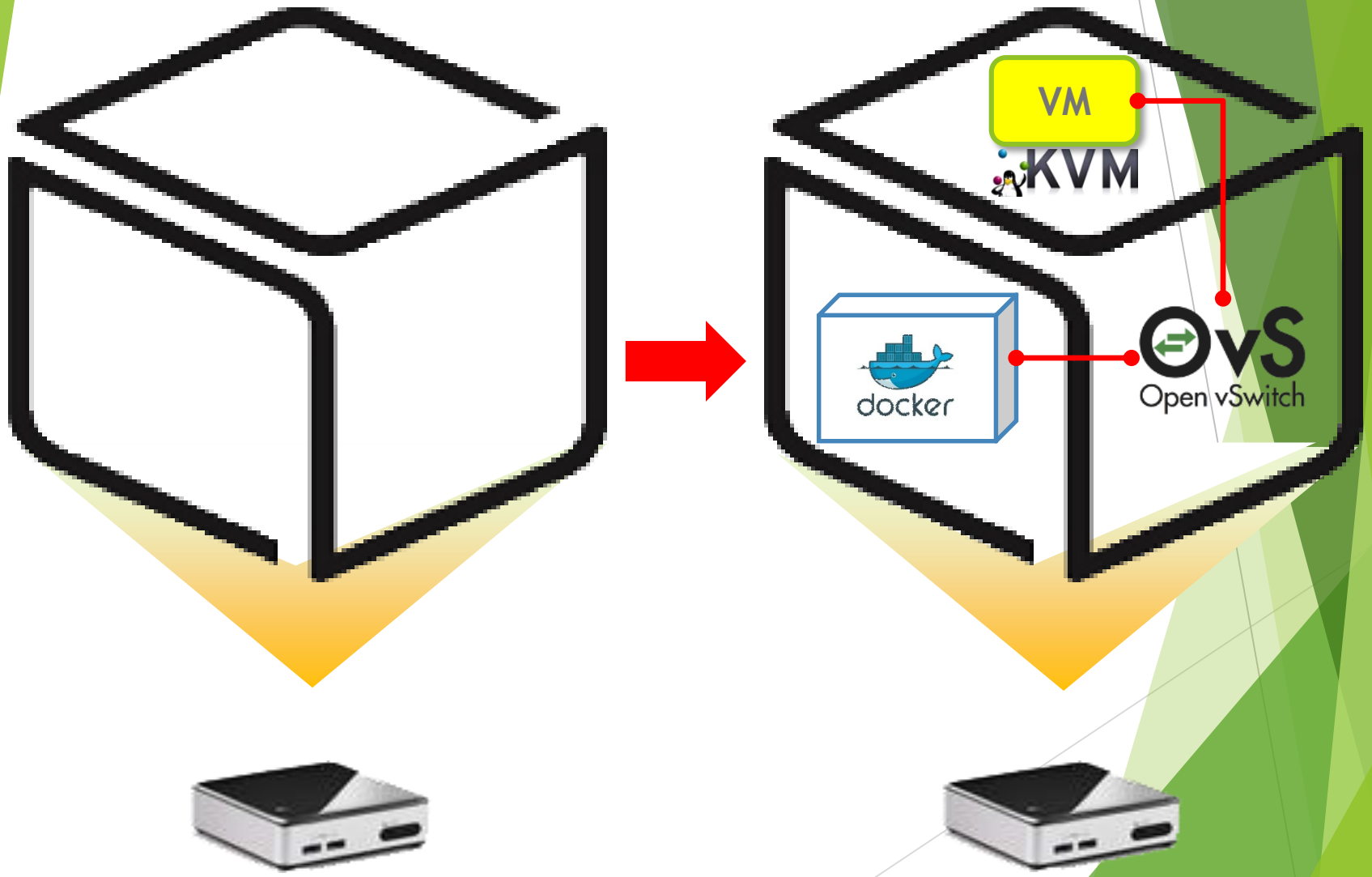
```
root@b8c3bab8204b:/# ifconfig
eth0      Link encap:Ethernet  HWaddr a2:86:d9:c2:33
          inet addr:192.168.0.3  Bcast:0.0.0.0  Mask
          inet6 addr: fe80::a086:d9ff:fec2:337b/64 S
          UP BROADCAST RUNNING MULTICAST  MTU:1500
          RX packets:136 errors:0 dropped:0 overruns
          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 192.168.0.2
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:~$
vbox@vFunction:~$
vbox@vFunction:~$
vbox@vFunction:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr ee:ee:ee:ee:ee:01
          inet addr:192.168.0.2  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 192.168.0.3
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

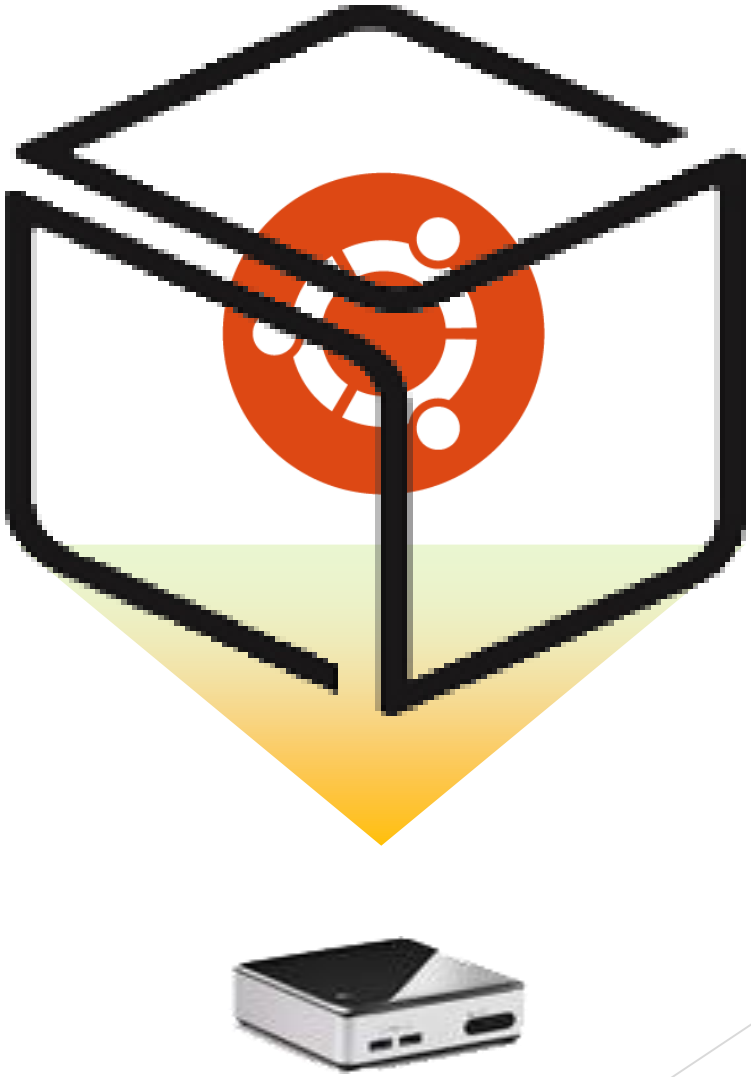
KVM VM

Box Lab: Final Goal



Appendix

NUC OS Setting

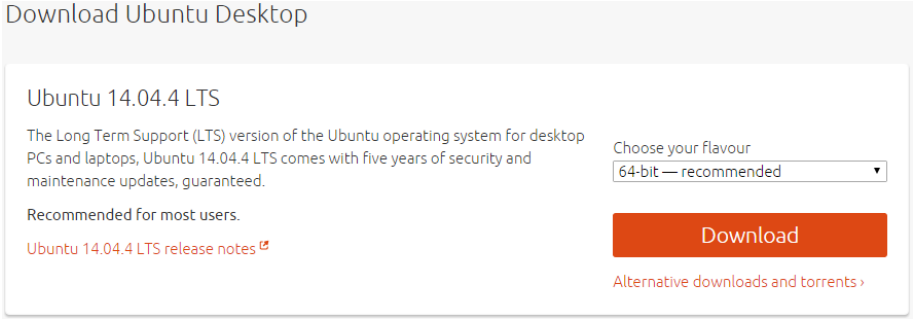


NUC OS Setting

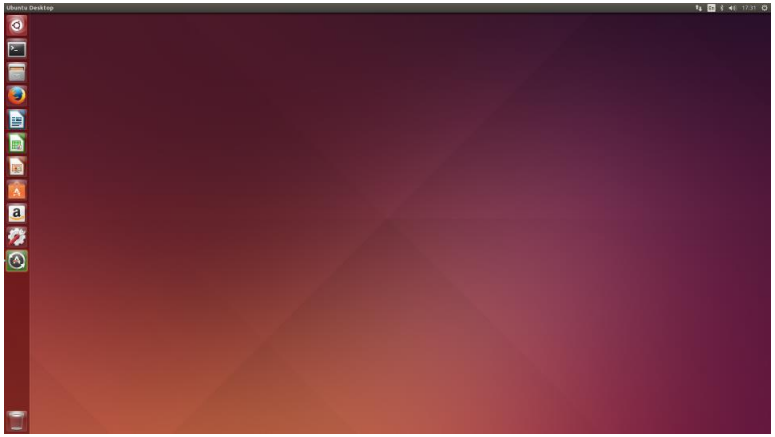
- BIOS Setting (64bit OS 설치시 필요)
  - Booting 후 F2 버튼을 통해 BIOS Setting 화면으로 접속
  - UEFI boot 체크 해제, (Legacy 방식으로 부팅)

NUC OS Setting

- OS : Ubuntu Desktop 14.04.4 LTS(64bit)
  - Download Site : http://www.ubuntu.com/download/desktop



  - OS 설치를 위해 download 된 파일(ubuntu-14.04.4-desktop-amd64.iso, 0.99Gb)을 이용하여 bootable USB 구성(bootable CD 는 불가능, NUC에 CD-Rom이 없음)
  - NUC에 설치



설치 완료 후
Ubuntu 초기화면

NUC & Pi2 IP address Setting

- Pi2 IP address 설정을 위해 필요한 파일(편집은 root 만 가능)
  - /etc/network/interfaces

$cd /etc/network
$sudo vi interfaces

```
HypriotOS: pi@black-pearl in ~
$ cd /etc/network/
HypriotOS: pi@black-pearl in /etc/network
$ sudo vi interfaces
```

#iface eth0 inet dhcp ← # 은 주석

auto eth0

iface eth0 inet static

address 172.29.0.X ← ip address

netmask 255.255.255.0 ← subnet mask

gateway 172.29.0.254 ← Gateway

dns-nameservers 203.237.32.100 203.237.32.101

입력 예

```
auto lo
iface lo inet loopback

#allow-hotplug eth0
#iface eth0 inet dhcp

auto eth0
iface eth0 inet static
        address 172.29.1.9
        netmask 255.255.255.0
        gateway 172.29.1.254
        dns-nameservers 203.237.32.100

iface eth0 inet6 auto

allow-hotplug wlan0
iface wlan0 inet dhcp
pre-up /usr/bin/occi
wpa-conf /etc/wpa_supplicant/wpa_suppl
iface default inet dhcp
```

  - 일반적으로 dns-nameservers 를 입력하면 9 page 는 필요 없으나, Hypriot OS 는 삽입되지 않으므로 resolv.conf 파일에 직접 nameserver를 입력해야 함!

NUC & Pi2 IP address Setting

- Pi2 IP address 설정을 위해 필요한 파일
  - /etc/resolv.conf

$cd /etc/
$sudo vi resolv.conf

```
# nameserver config
nameserver 203.237.32.100
nameserver 203.237.32.101
```

기존의 nameserver 는 #을 추가하여 주석처리

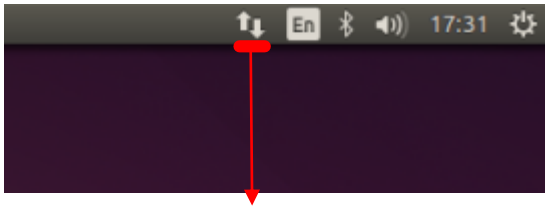
```
# nameserver config
#nameserver 213.133.98.98
#nameserver 213.133.99.99
#nameserver 213.133.100.100

nameserver 203.237.32.100
nameserver 203.237.32.101
```

$sudo /etc/init.d/networking restart 입력 또는 rebooting 후 network 확인
$sudo reboot (rebooting command)

```
$ sudo /etc/init.d/networking restart
[....] Restarting networking (via systemctl): networking.serviceWarni
ce changed on disk, 'systemctl daemon-reload' recommended.
. ok
HypriotOS: pi@black-pearl in /etc
```

NUC & Pi IP address Setting

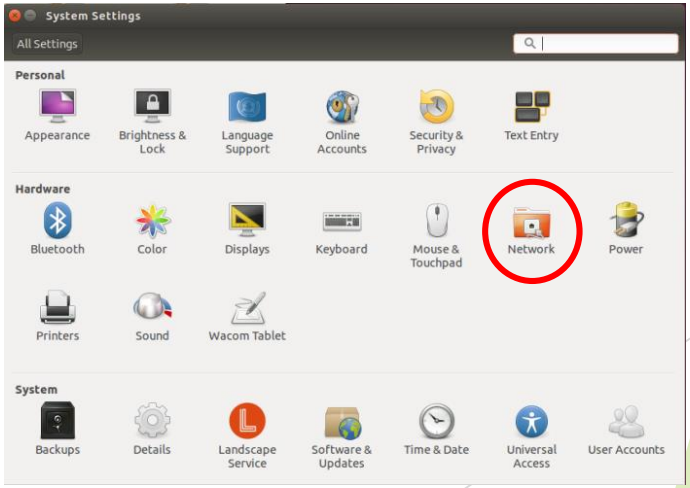
- NUC IP address Setting
  1. Pi와 동일하게 진행
  2. GUI 환경에서 setting



Network icon

Edit Connection.. 선택

System Setting icon



Linux Setting(NUC & Pi)

- Ubuntu & Hypriot OS : Debian 계열 linux
  - Package 관리 관련 명령어(apt-get : Advanced Packaging Tool)

    $sudo apt-get update
    - Update package index 갱신(/etc/apt/sources.list)

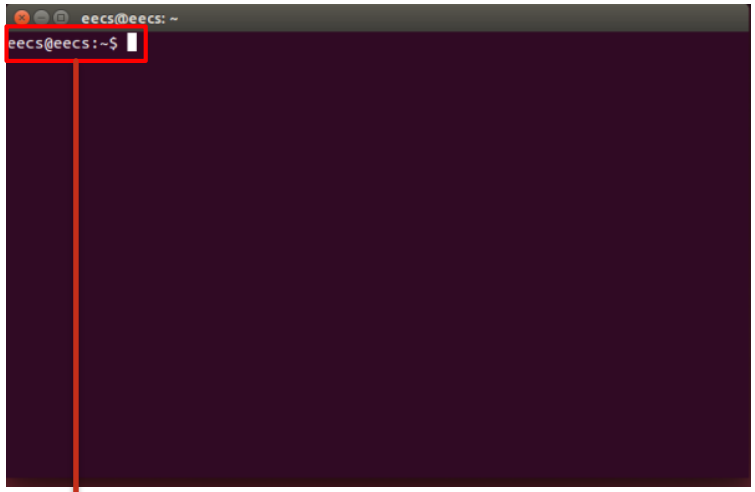
    $sudo apt-get upgrade
    - Package 목록을 비교하여 package upgrade 실행

    $sudo apt-get dist-upgrade
    - Package 간의 의존성 검사를 하며 upgrade(optional)
  - Package 설치(/var/cache/apt/archive/에 설치)

    $sudo apt-get install <package_name>
  - 자신에 맞는 기본적인 package 설치
    - Kernel Lab(Pi2)을 위한 package : gcc-4.9, make
    - Editor : vim(vi iMproved), emacs, gedit, etc.
    - SSH(Secure Shell) : openssh

Linux Beginner Guide

- Terminal 창



eecs@eecs:~$

user **hostname** **home directory**

< 간단한 명령어 모음 >

$pwd : 현재위치
$cd : Change directory
$ls : directory 보기
$mkdir : directory 생성
$su : root 계정으로 이용
$apt-get : package 설치 및 삭제
$poweroff
$shutdown –h now

Linux Beginner Guide

- gcc-4.9 설치(NUC & Pi2)
  - gcc 설치 **$sudo apt-get install gcc-4.9**
  - gcc version 확인 : **$gcc --version**

```
root@eecs:/home/eecs# clear
root@eecs:/home/eecs# gcc --version
gcc (Ubuntu 4.8.4-2ubuntu1~14.04.1) 4.8.4
Copyright (C) 2013 Free Software Foundation
This is free software; see the source for c
warranty; not even for MERCHANTABILITY or F
```

< NUC >

```
HypriotOS: pi@black-pearl in ~
$ gcc --version
gcc (Raspbian 4.9.2-10) 4.9.2
Copyright (C) 2014 Free Software
This is free software; see the s
warranty; not even for MERCHANTA
```

< pi2 >

  - NUC 에 gcc 설치를 위해 아래의 command 실행

$sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$sudo apt-get update
$sudo apt-get install gcc-4.9

```
eecs@eecs:~$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
[sudo] password for eecs:
 Toolchain test builds; see https://wiki.ubuntu.com/ToolChain

 More info: https://launchpad.net/~ubuntu-toolchain-r/+archive/ubuntu/test
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmpkwcjvts6/secring.gpg' created
gpg: keyring `/tmp/tmpkwcjvts6/pubring.gpg' created
gpg: requesting key BA9EF27F from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpkwcjvts6/trustdb.gpg: trustdb created
gpg: key BA9EF27F: public key "Launchpad Toolchain builds" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
eecs@eecs:~$
```

Linux Beginner Guide

- gcc-4.9 설치(NUC & Pi2)

```
eecs@eecs:~$ sudo apt-get install gcc-4.9
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  cpp-4.9 gcc-4.9-base gcc-5-base lib32gcc1 libasan1 libatomic1 libcilkrts5
  libgcc-4.9-dev libgcc1 libgomp1 libisl15 libitm1 liblsan0 libmpfr4
  libquadmath0 libtsan0 libubsan0 libx32gcc1
Suggested packages:
  gcc-4.9-locales gcc-4.9-multilib gcc-4.9-doc libgcc1-dbg libgomp1-dbg
  libitm1-dbg libatomic1-dbg libasan1-dbg liblsan0-dbg libtsan0-dbg
  libubsan0-dbg libcilkrts5-dbg libquadmath0-dbg
The following NEW packages will be installed:
  cpp-4.9 gcc-4.9 gcc-5-base libasan1 libcilkrts5 libgcc-4.9-dev libisl15
  liblsan0 libubsan0
The following packages will be upgraded:
  gcc-4.9-base lib32gcc1 libatomic1 libgcc1 libgomp1 libitm1 libmpfr4
  libquadmath0 libtsan0 libx32gcc1
10 upgraded, 9 newly installed, 0 to remove and 46 not upgraded.
Need to get 14.7 MB of archives.
After this operation, 49.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

  - version 확인

```
eecs@eecs:~$ gcc --version
gcc (Ubuntu 4.8.4-2ubuntu1~14.04.1) 4.8.4
Copyright (C) 2013 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

  - link 설정을 해주면 된다.

Linux Beginner Guide

- gcc-4.9 link 설정
  - 현재 gcc-4.8 version 이 link 되어 있음
  - 현재의 link 를 지우고 4.9 version으로 재설정

$sudo rm /usr/bin/gcc : **link 삭제**
$sudo ln –s /usr/bin/gcc-4.9 /usr/bin/gcc : **link 설정**

```
eecs@eecs:~$ ls -al /usr/bin/gcc
lrwxrwxrwx 1 root root 16  3월 14 16:06 /usr/bin/gcc -> /usr/bin/gcc-4.8
eecs@eecs:~$ sudo rm /usr/bin/gcc
eecs@eecs:~$
eecs@eecs:~$ sudo ln -s /usr/bin/gcc-4.9 /usr/bin/gcc
eecs@eecs:~$
eecs@eecs:~$ ls -al /usr/bin/gcc
lrwxrwxrwx 1 root root 16  3월 14 16:07 /usr/bin/gcc -> /usr/bin/gcc-4.9
eecs@eecs:~$
eecs@eecs:~$ gcc --version
gcc (Ubuntu 4.9.3-8ubuntu2~14.04) 4.9.3
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Vi/vim editor 명령어 모음

- 입력 모드 : 원하는 글자를 입력
  - **a** : 현재 위치의 다음부터 입력 시작
  - **i** : 현재 위치의 앞에서부터 입력 시작
- 명령 모드 : 문서 편집을 할 수 있으며, 입력 모드 상태에서 ESC키를 누르면 명령모드로 전환됨
  - **x** : 커서가 있는 문자 삭제
  - **dd** : 현재 줄 전체 삭제
- 라인 모드 : ESC 키를 누른 후 colon(:) prompt 에서 명령을 입력하며 저장, 편집, 검색 기능 제공
  - **:q** : 그대로 종료하기
  - **:q!** : 변경된 내용을 저장하지 않고 종료하기
  - **:wq** : 변경된 내용을 저장하고 종료하기
- 보다 다양한 명령어가 있으며, 직접 실습을 하는 것이 좋은 방법

Thank You for Your Attention Any Questions?