

# Computer Systems Lab.

Computer  
Systems Lab  
@ Spring 2016

김종원 교수님

실험 조교 :

배정주 (NetCS Lab) 석사 과정

이준기 (NetCS Lab) 석사 과정

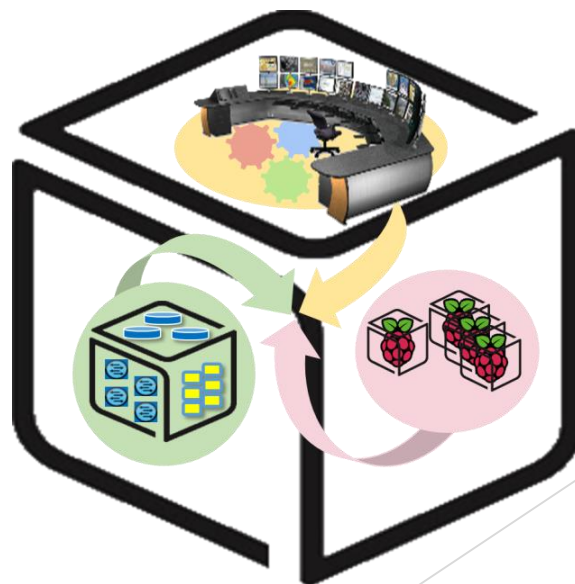
윤희범 (NetCS Lab) 석사 과정

김철원 (NetCS Lab) 석사 과정

김승룡 (NetCS Lab) 석사 과정

송지원 (NetCS Lab) 석사 과정

남택호 (NetCS Lab) 석사 과정



# CSLab: Materials Table

No	Lab Name	Env Setting	Hypriot OS	Docker	Kafka	Flume	Net SNMP	Open vSwitch	KVM	VNC	ONOS	Mesos	Spark	HDFS	Zeppelin
1	Playground	○	○												
2	Box			○ ●	○	○	○	○ ●	○ ●	○ ●					
3	InterConnect		●	●	●	●	●	●							
3X	SDN										○ ●				
4	Functions & Orchestration		●	●											
5	Cluster											○	○	○	
6	Analytics											●	●	●	○ ●
7X	Services														

○ Install / Download    ● Use

# OpenVswitch

## - Installation



### Install OpenVswitch

There are many OVS related package, but install openvswitch-switch is enough.  
Because 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

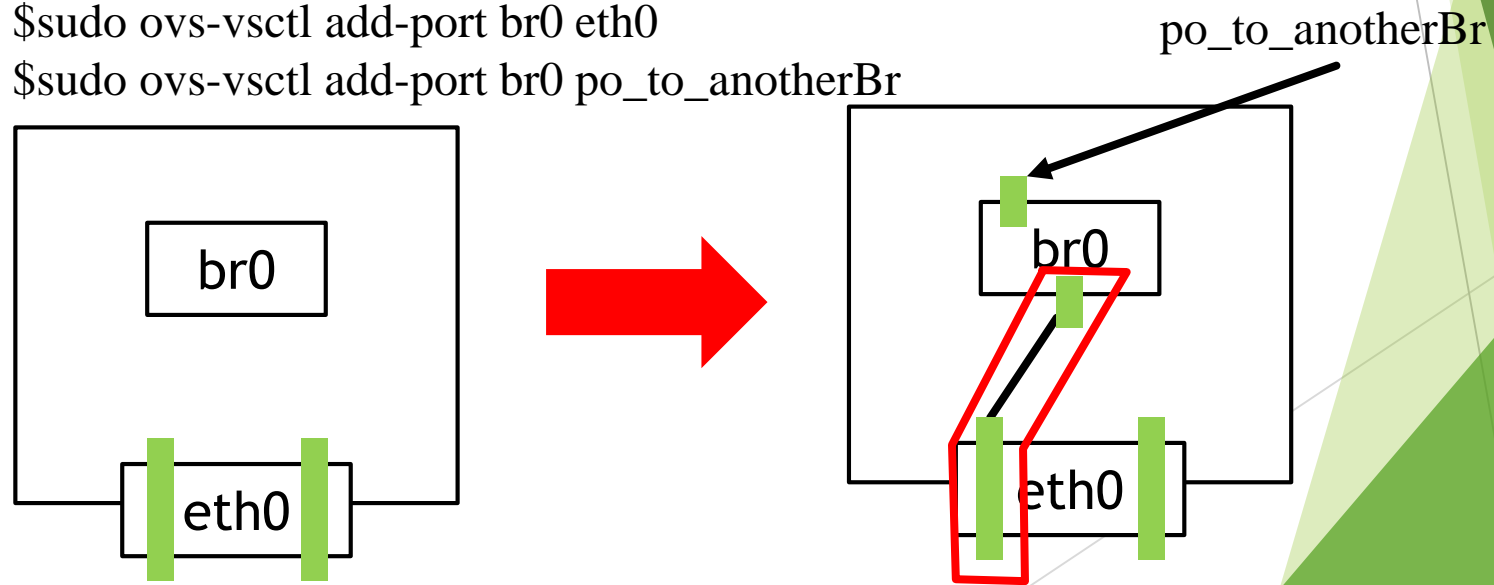
## - Basic command

```
$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

## - Basic command

**Example)**

**✗You don't need to follow below command.**

```
$sudo ifconfig eth0 0
```

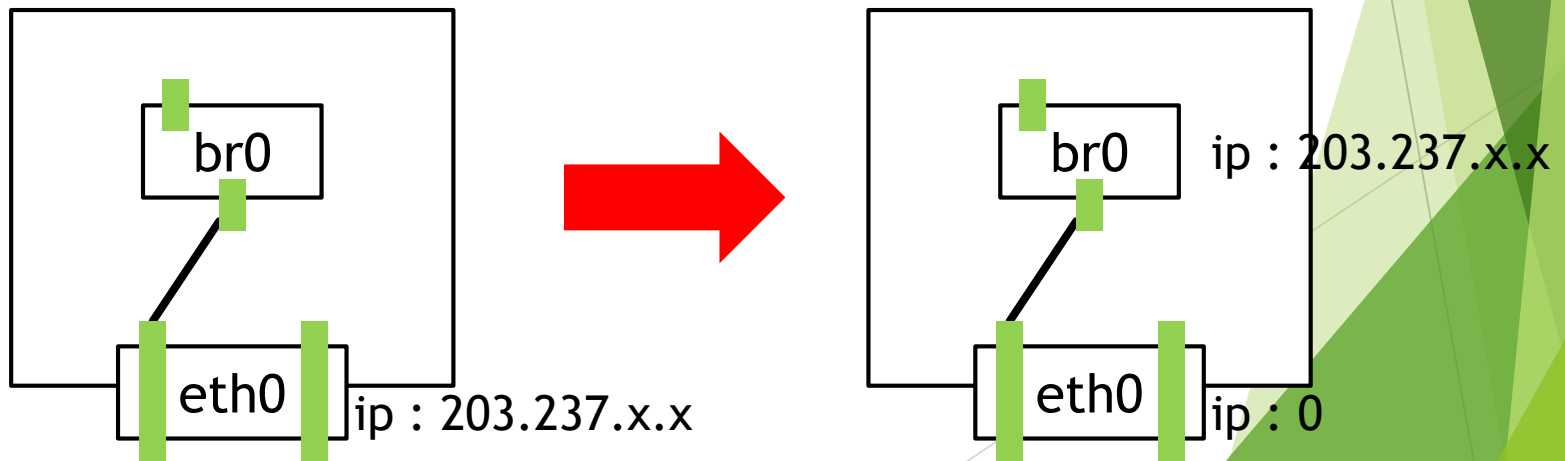
```
$sudo ifconfig br0 0
```

```
$sudo ifup br0
```

```
$sudo ifconfig br0 up
```

```
$sudo ifconfig br0 203.237.x.x
```

```
$sudo ifconfig eth0 up
```



# OpenVswitch

## - Configure bridged networking



Modify network interface configuration.

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

```
----- /etc/network/interfaces -----  
# The primary network interface  
auto (eth0->)br0  
iface (eth0->)br0 inet static  
...
```

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
```

# OpenVswitch

## - 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"
```

# OpenVswitch

## - 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:210.125.84.255  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:210.125.84.255  Bcast:210.125.84.255  Mask:255.255.255.0
        inet6 addr: fe80::eea8:6bff:fe23:4e1c/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



# OpenVswitch

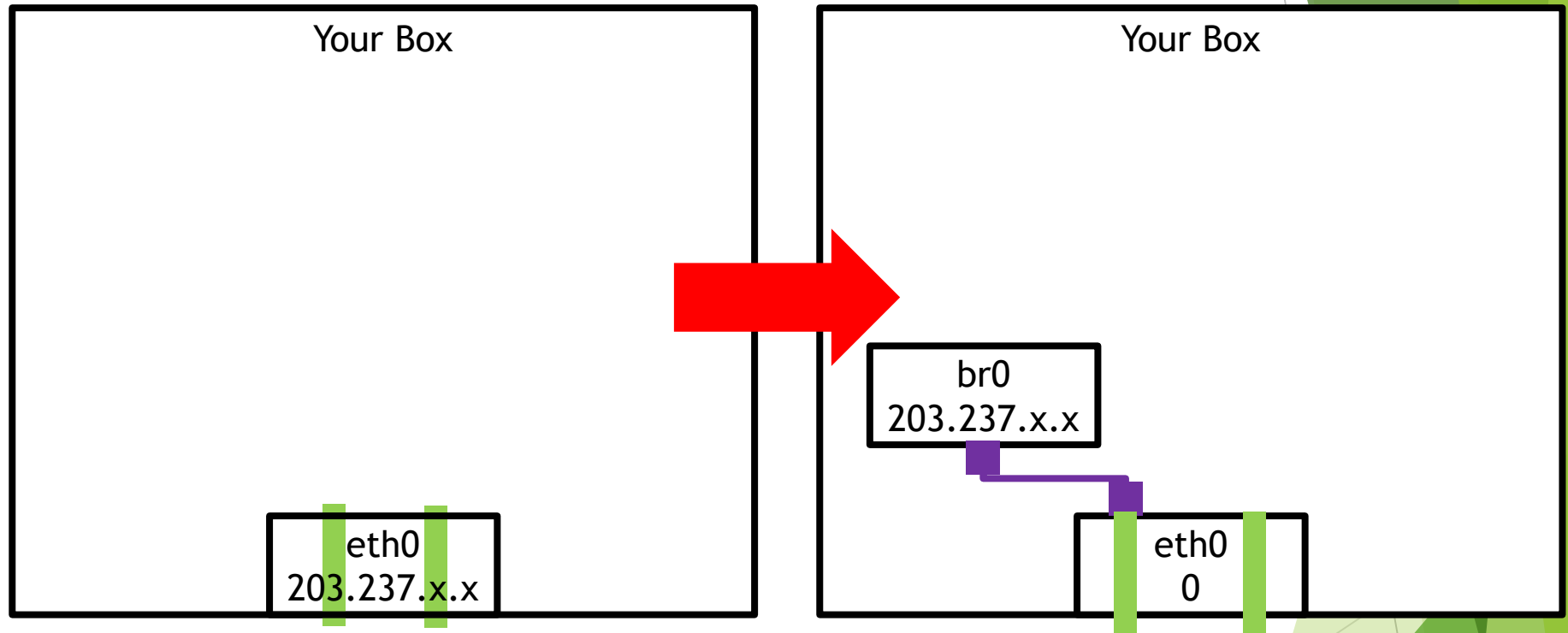
## - Configure bridged networking



```
$sudo ifconfig eth0 0           // IP of eth0 → None
$sudo ifconfig br0 0           // IP of br0 → None
$sudo ifup br0                 // interface br0 turn on
$sudo ifconfig br0 up          // interface br0 turn on
$sudo ifconfig eth0 up
```

# OpenVswitch

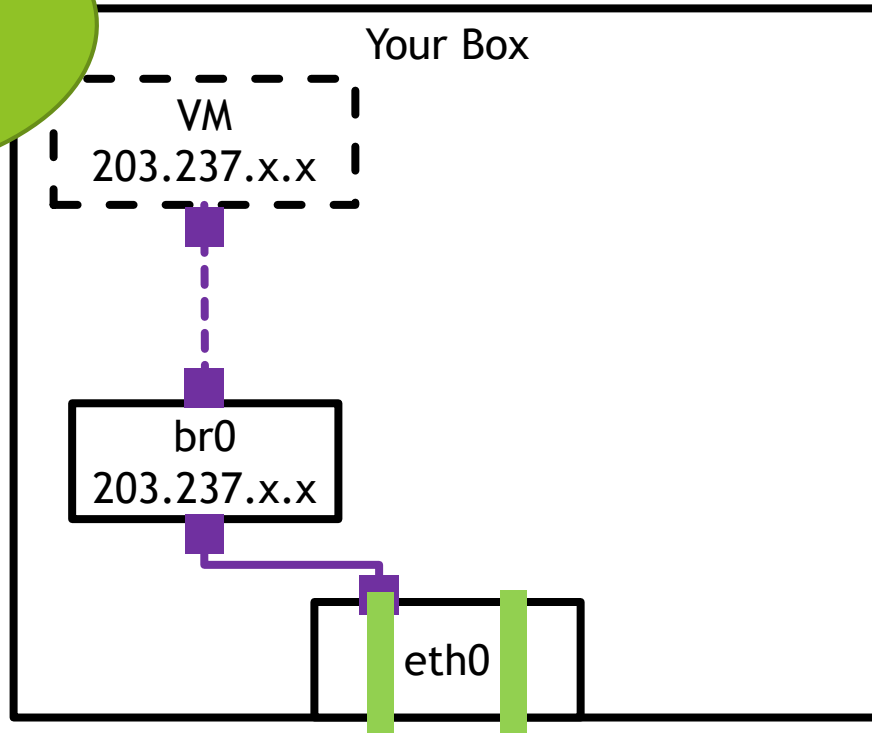
## - Situation



# OpenVswitch

## - Goal of this section

Stop OVS  
setting  
Now need to  
build KVM



# Modify

- /etc/rc.local



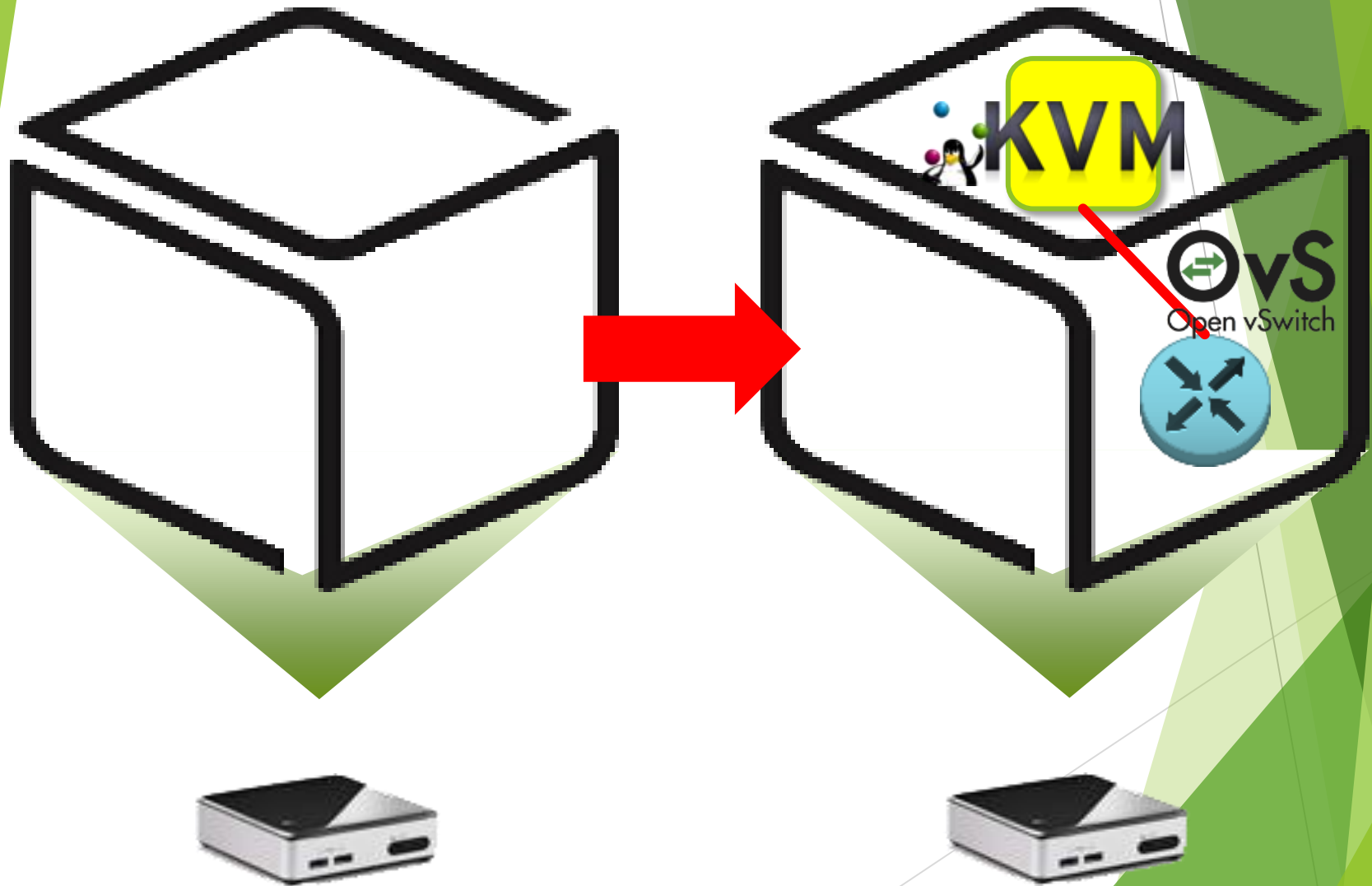
Modify /etc/rc.local

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

```
ifconfig eth0 up
```

# 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 (library, etc.)



Install dependency & download Ubuntu 14.04.3 64bit server image.

```
$sudo apt-get install qemu-kvm libvirt-bin // KVM dependency
```

```
$wget http://releases.ubuntu.com/14.04.3/ubuntu-14.04.3-server-amd64.iso  
(official)
```

//If above URL is can't be downloaded, try below URL(dropbox link)

```
$wget https://www.dropbox.com/s/v6pwrksl9xdnvux/ubuntu-14.04.3-server-amd64.iso  
(dropbox link, recommended)
```

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

# OpenVswitch

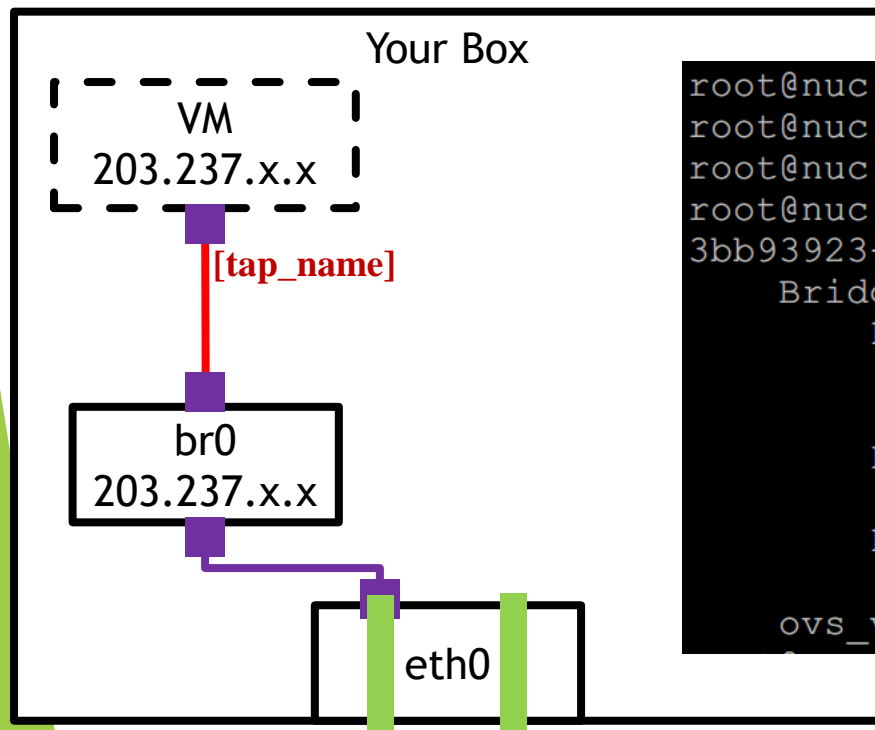
## - Configure bridged networking



Make a tap and attach to VM.

```
$sudo ip tuntap add mode tap [tap_name]
$sudo ifconfig [tap_name] up
$sudo ovs-vsctl add-port br0 [tap_name] // 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]
```

```
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-14.04.3-server-amd64.iso -vnc :[#]
-daemonize
```

```
sudo kvm -m 512 -name tt -smp cpus=1,maxcpus=1 -device virtio-net-pci,netdev=net0,mac='EE:EE:EE:EE:EE:77'
-netdev tap,id=net0,ifname=vport_vFunction,script=no -boot d tt.img -cdrom ubuntu-14.04.3-server-amd64.iso
-vnc :5 -daemonize
```

Download VNC viewer to see inside of VM

**Windows :** <https://www.dropbox.com/s/h2mr0g8obm5mgpe/vncviewer.exe?dl=0>

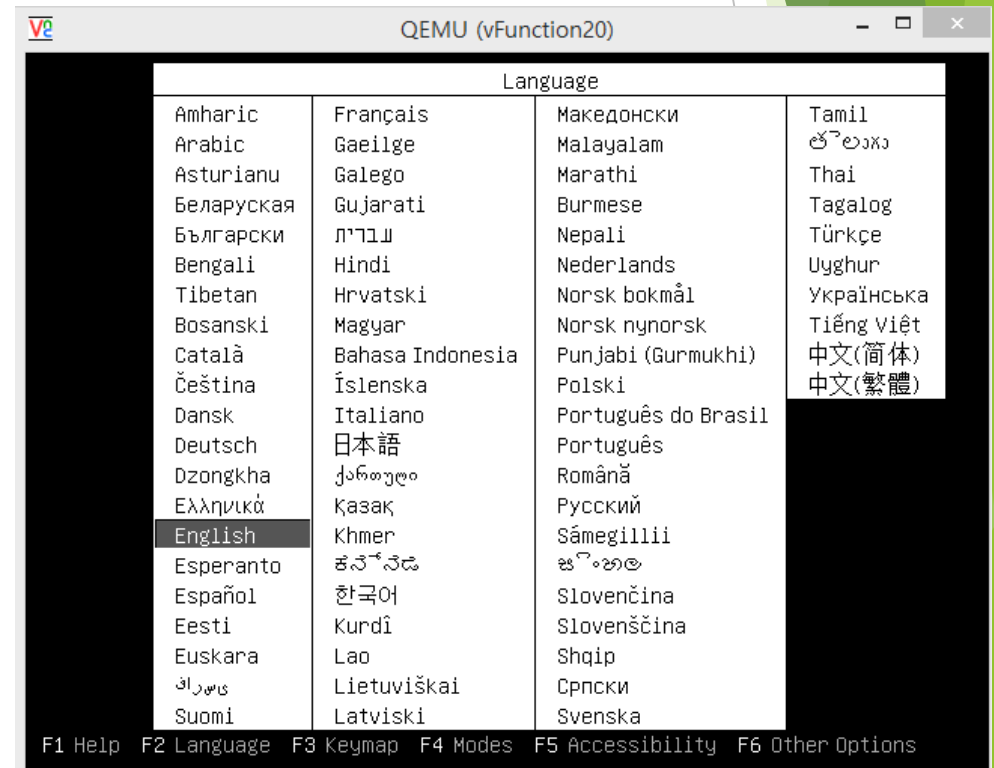
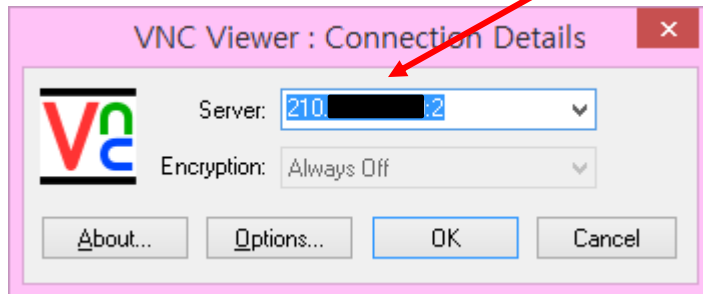
**Ubuntu :** `sudo apt-get install xvnc4viewer`

# KVM

## - Install Ubuntu to VM



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



[!!] 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

### [!!] 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.

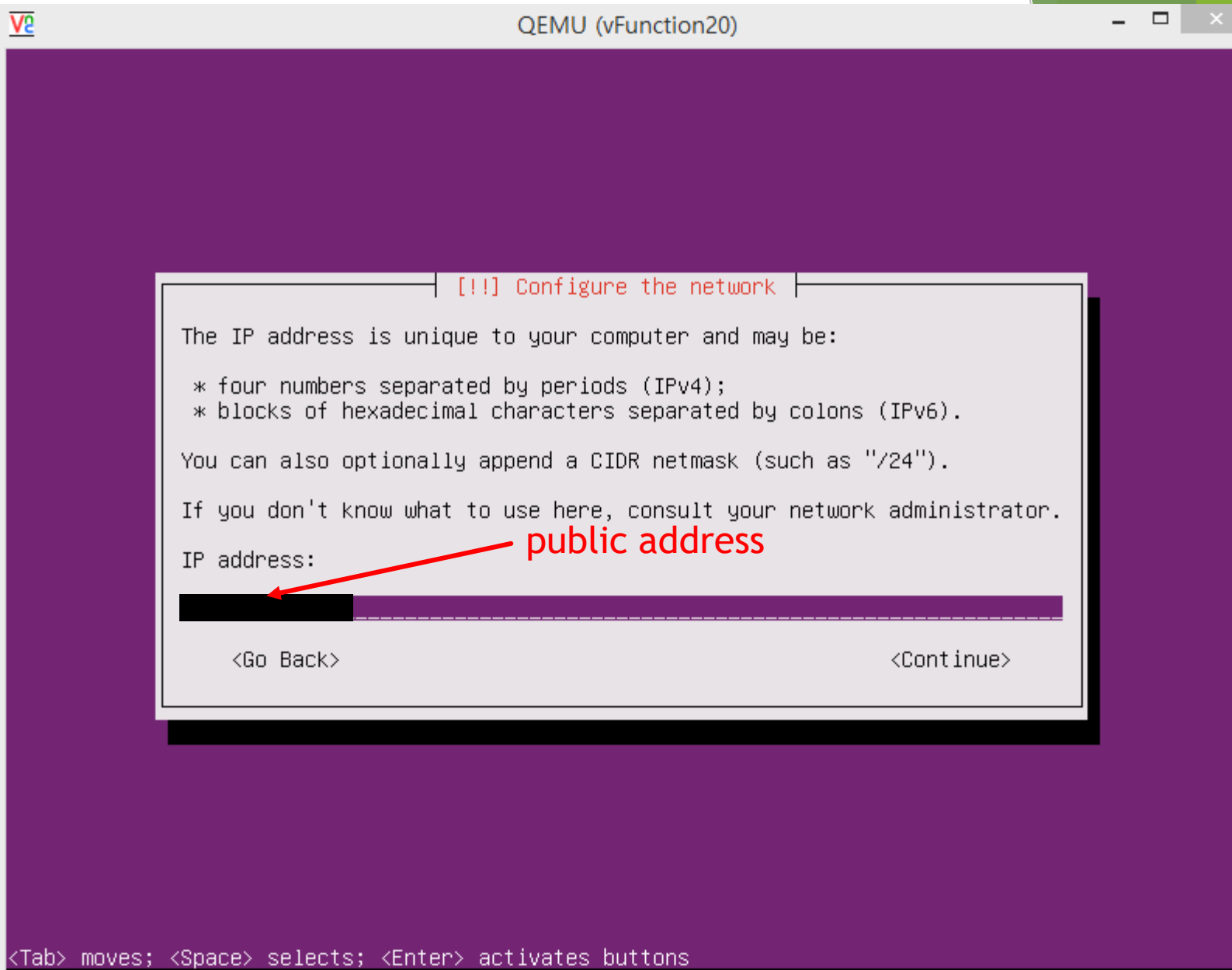
Network configuration method:

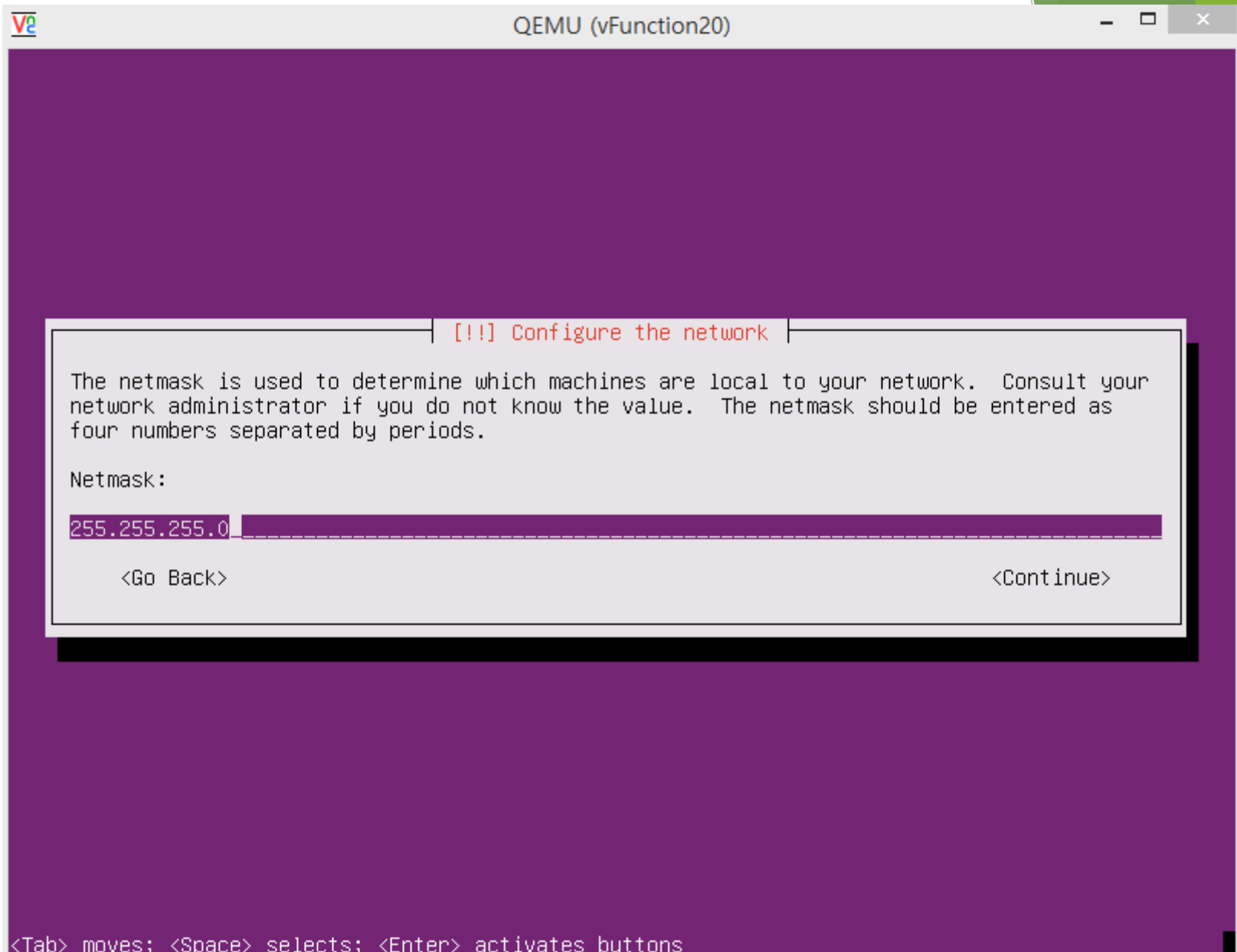
Retry network autoconfiguration  
Retry network autoconfiguration with a DHCP hostname  
**Configure network manually**

Do not configure the network at this time

<Go Back>

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





## [!!] 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

## [!!] 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>

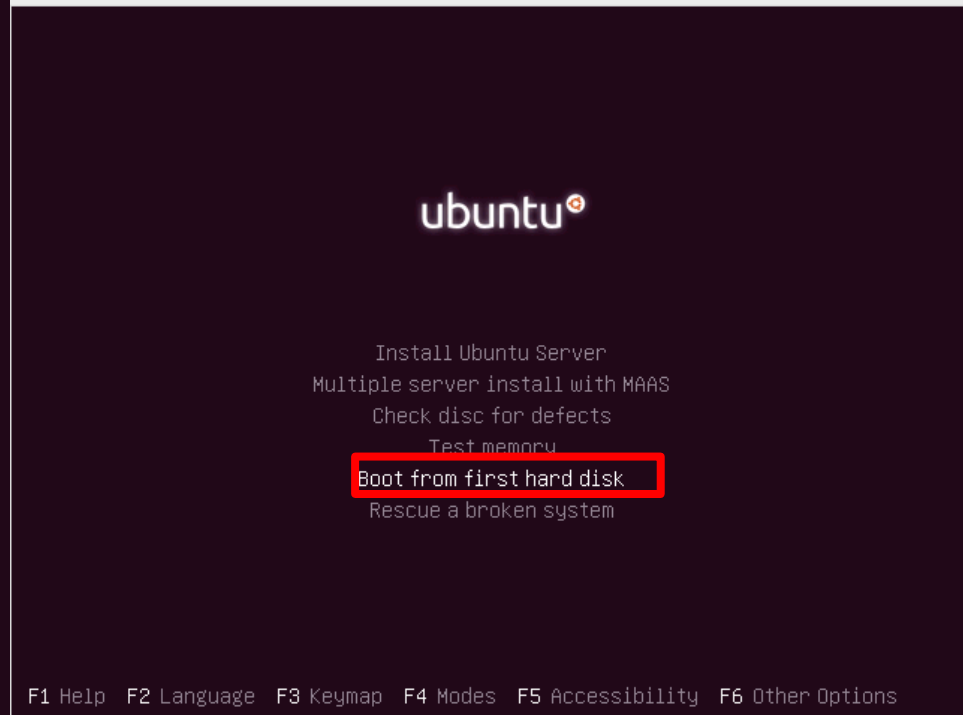
<Continue>

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



Language			
Amharic	Français	Македонски	Tamil
Arabic	Gaeilge	Malayalam	தமிழ்
Asturiano	Galego	Marathi	Thai
Беларуская	Gujarati	Burmese	Tagalog
Български	ᱚᱱᱟᱹᱨᱤᱧ	Nepali	Türkçe
Bengali	Hindi	Nederlands	Uyghur
Tibetan	Hrvatski	Norsk bokmål	Українська
Bosanski	Magyar	Norsk nynorsk	Tiếng Việt
Català	Bahasa Indonesia	Punjabi (Gurmukhi)	中文(简体)
Čeština	Íslenska	Polski	中文(繁體)
Dansk	Italiano	Português do Brasil	
Deutsch	日本語	Português	
Dzongkha	தமிழ்	Română	
Ελληνικά	Қазақ	Русский	
English	Khmer	Sámegillii	
Esperanto	ಕನ್ನಡ	සිංහල	
Español	한국어	Slovenčina	
Eesti	Kurdî	Slovenščina	
Euskara	Lao	Shqip	
فارسی	Lietuviškai	Српски	
Suomi	Latviski	Svenska	

F1 Help F2 Language F3 Keymap F4 Modes F5 Accessibility F6 Other Options



F1 Help F2 Language F3 Keymap F4 Modes F5 Accessibility F6 Other Options

Push esc

# KVM

## - VM boot command



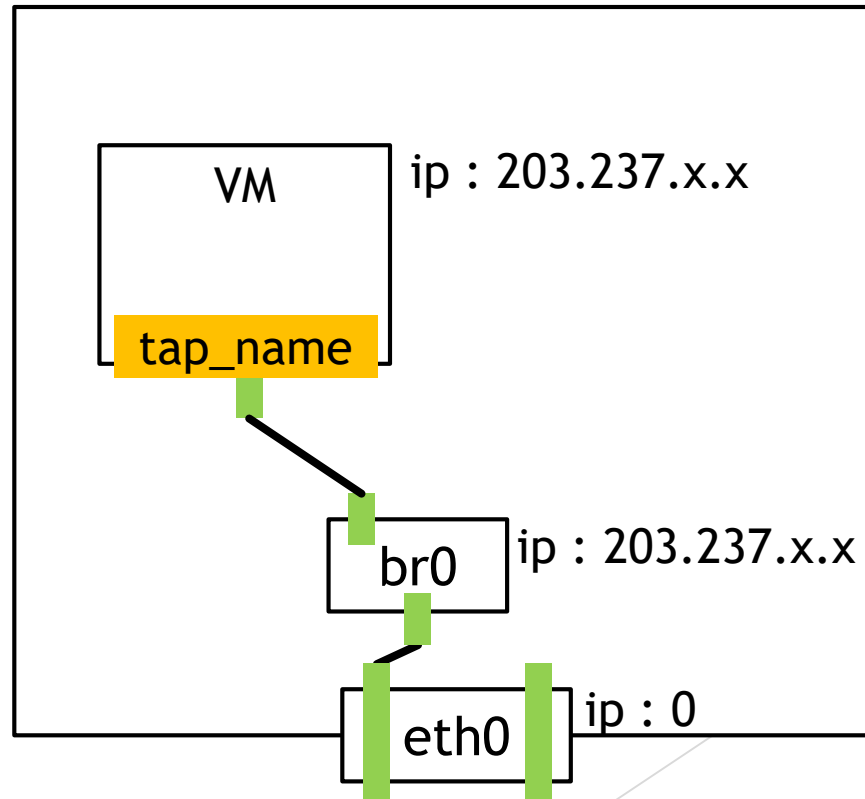
If you want boot VM again (mac should be different from others).

```
$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
```

# OpenVswitch connects with KVM

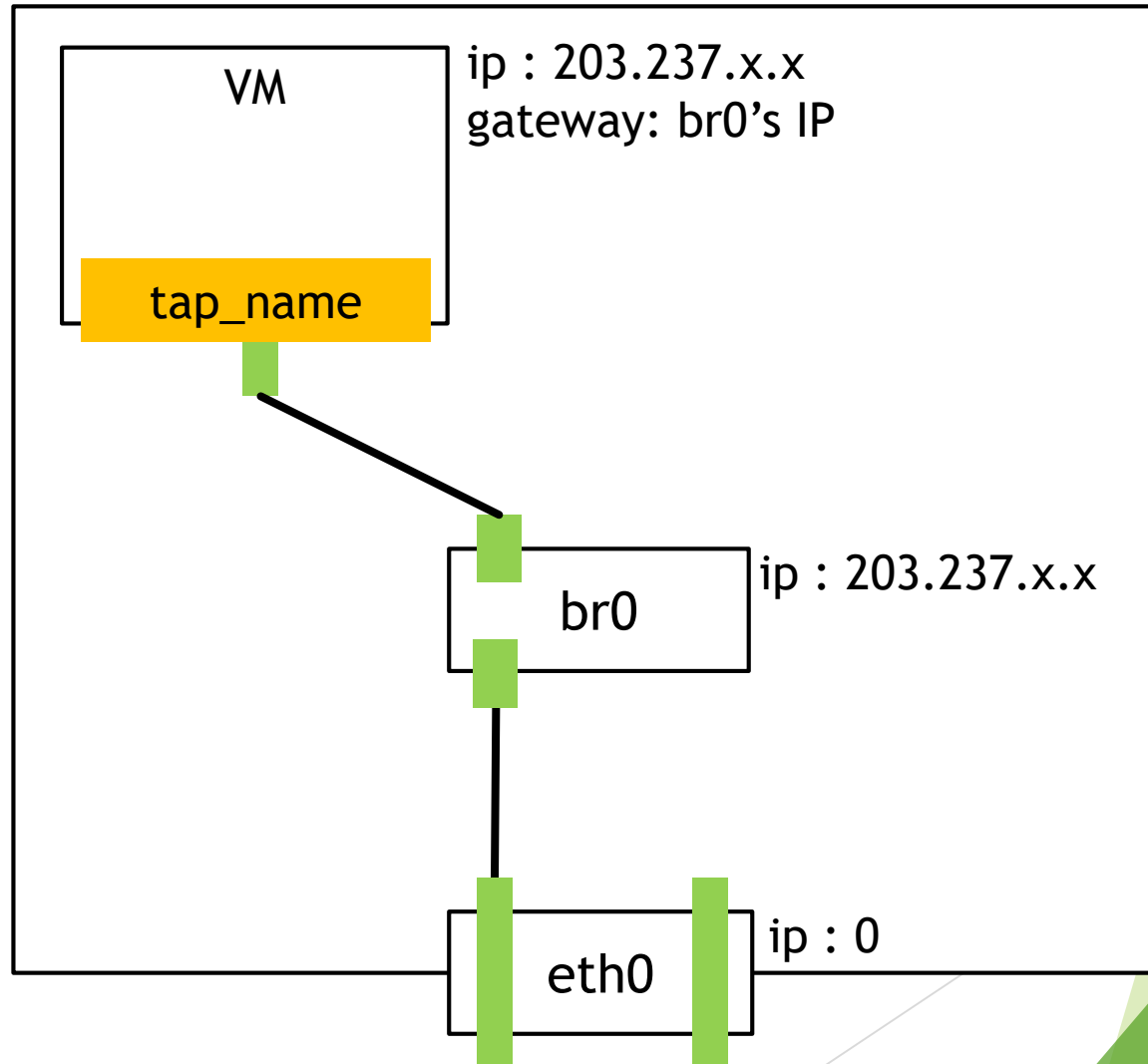
## - Situation

```
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"
```



# OpenVswitch 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: 
```

# Modify

- /etc/rc.local

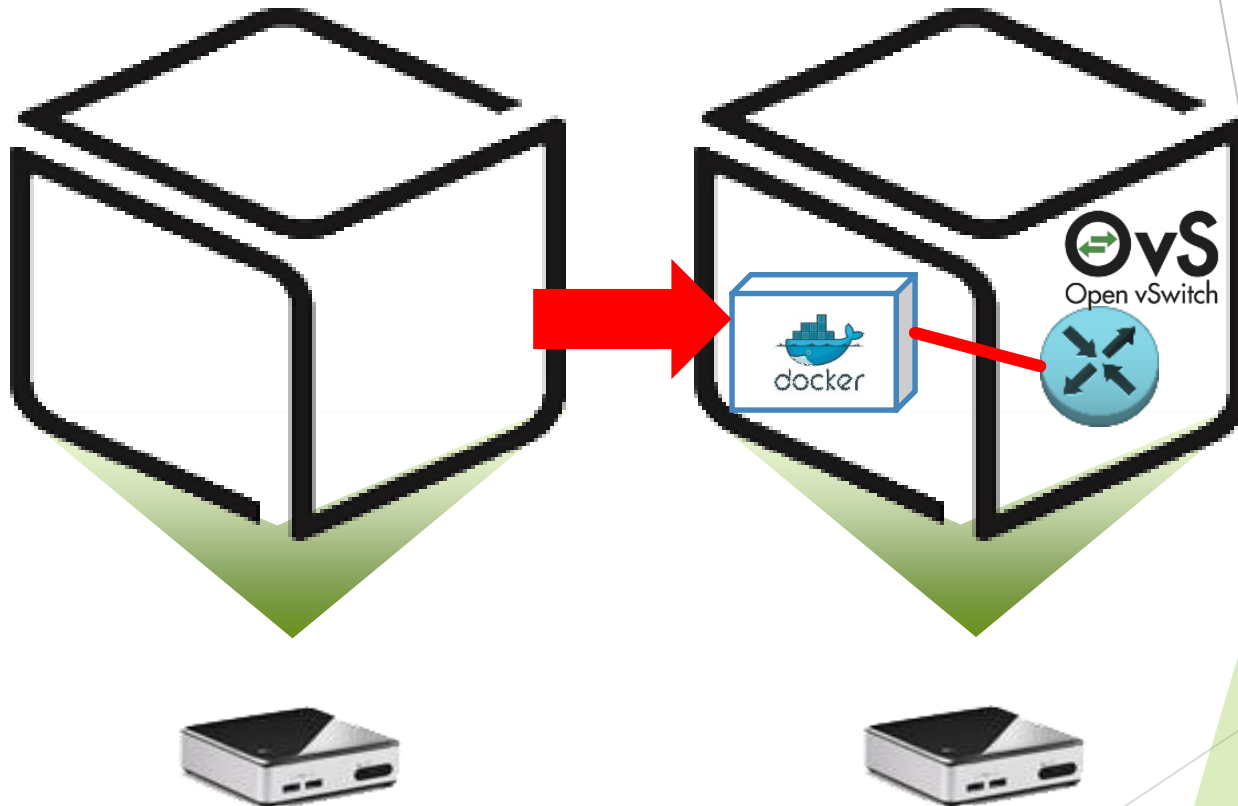


Modify /etc/rc.local

```
$sudo vi /etc/rc.local  
  
ip tuntap add mode tap [tap_name]  
ifconfig [tap_name] up  
ovs-vsctl del-port br0 [tap_name]  
ovs-vsctl add-port br0 [tap_name]
```

# 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 adduser [your_id] docker
```

(Session restart)

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

reference: [http://docs.docker.com/linux/step\\_one/](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:/#
root@8346684676d8:/# ifconfig
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)
```

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.)

```
$cd /usr/bin
```

```
$sudo wget https://raw.githubusercontent.com/openvswitch/ovs/master/utilities/ovs-docker
```

```
$sudo chmod a+rwX ovs-docker
```

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

# Modify

- /etc/rc.local



Modify /etc/rc.local

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

```
docker start [container_name]
```

```
ovs-docker del-port br0 eth0 [containerName] --ipaddress=[IP_address] --gateway=[Gateway_address]
```

```
ovs-docker add-port br0 eth0 [containerName] --ipaddress=[IP_address] --gateway=[Gateway_address]
```

# 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 data:
64 bytes from hkg07s21-in-f238.1e100.net (216.58.221.238): icmp_seq=1 ttl=64 time=0.573 ms
64 bytes from hkg07s21-in-f238.1e100.net (216.58.221.238): icmp_seq=2 ttl=64 time=0.590 ms
64 bytes from hkg07s21-in-f238.1e100.net (216.58.221.238): icmp_seq=3 ttl=64 time=0.585 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1700 ms
rtt min/avg/max/mdev = 0.573/0.584/0.587/0.004 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=0.872 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.590 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.585 ms
^C
--- 192.168.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1700 ms
rtt min/avg/max/mdev = 0.651/1.028/1.519/0.365 ms
root@b8c3bab8204b:/#

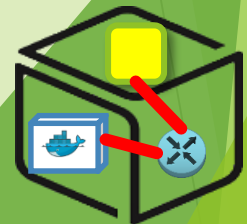
vbox@vFunction:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr ee:ee:ee:ee:01
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::ecee:eeff:feee:ee01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18857 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 3004 ms
rtt min/avg/max/mdev = 0.573/0.655/0.872/0.125 ms
vbox@vFunction:~$
```

Docker container

KVM VM





# Download Sources from Github



Install Git package in NUC machine.  
(In Hypriot OS, Git package is already installed.)

```
$sudo apt-get install git
```

Download all files from Github in Raspberry Pi and NUC machine.

```
$git clone https://github.com/SmartXBox/SmartX-mini.git
```

- Folder List

▢	raspbian-flume
▢	raspbian-kafka
▢	ubuntu-kafka
▢	ubuntu-nodejs

# Install Net-SNMP



## 1. Download Net-SNMP.

```
$sudo apt-get install -y snmp snmpd
```

## 2. Download and apply mibs.

```
$sudo apt-get install -y snmp-mibs-downloader  
$sudo download-mibs
```

## 3. Modify configuration file.

```
$sudo vi /etc/snmp/snmpd.conf  
#rocommunity public localhost → Delete #  
$sudo /etc/init.d/snmpd restart
```