SDN/NFV 실습
(v0.9.6)

(8GB RAM 윈도우 PC용)

JS Lab

안종석
james@jslab.kr

2019년 1월

목차

1. 실습 환경
2. Host
3. Open vSwitch
4. SDN Controller (Docker)
5. mininet (w/ONOS)
6. Rancher 설치
7. Kubernetes 설치

❖ 부록: Docker

1. 실습 환경

❖ 실습에 사용하는 소프트웨어

① **VirtualBox 6.x**

② **VirtualBox VM image 6개**

- UbuntuServer16.04 Fresh with 2 ports.ova (1개)
- UbuntuServer16.04 Docker and OVS with 2 ports.ova (1개)
- UbuntuServer16.04 ONOS and Rancher with 2 ports.ova (1개)
- onos-tutorial-1.14.0.ova (1개)
- CentOS 7 Worker/Master for Rancher and K8s (2개)

③ **Docker Toolbox**

④ **Putty / Super Putty**

⑤ **WinSCP**

⑥ **Xming**

⑦ **Advanced IP Scanner**

CentOS 7 Rancher and K8s Worker01.ova	755,108,864	01/22/2019 16:47
CentOS 7 Rancher and K8s Master.ova	1,206,004,224	01/22/2019 16:44
UbuntuServer16.04 ONOS and Rancher with 2 ports.ova	2,512,464,384	01/21/2019 20:25
UbuntuServer16.04 Docker and OVS with 2 ports.ova	810,596,352	01/21/2019 20:20
UbuntuServer16.04 Fresh with 2 ports.ova	691,409,920	01/21/2019 20:17
SuperPuttySetup-1.4.0.9.msi	1,875,968	01/21/2019 13:28
VirtualBox-6.0.2-128162-Win.exe	219,538,432	01/21/2019 13:20
onos-tutorial-1.14.0.ova	3,483,987,968	01/15/2019 12:50
WinSCP-5.13.7-Setup.exe	9,585,880	01/15/2019 12:32
DockerToolbox.exe	221,771,936	11/02/2018 01:58
Advanced_IP_Scanner_2.5.3646.exe	20,210,200	11/02/2018 01:55
putty-64bit-0.70-installer.msi	3,048,960	11/02/2018 01:14
Xming-6-9-0-31-setup.exe	2,204,914	02/15/2016 19:28

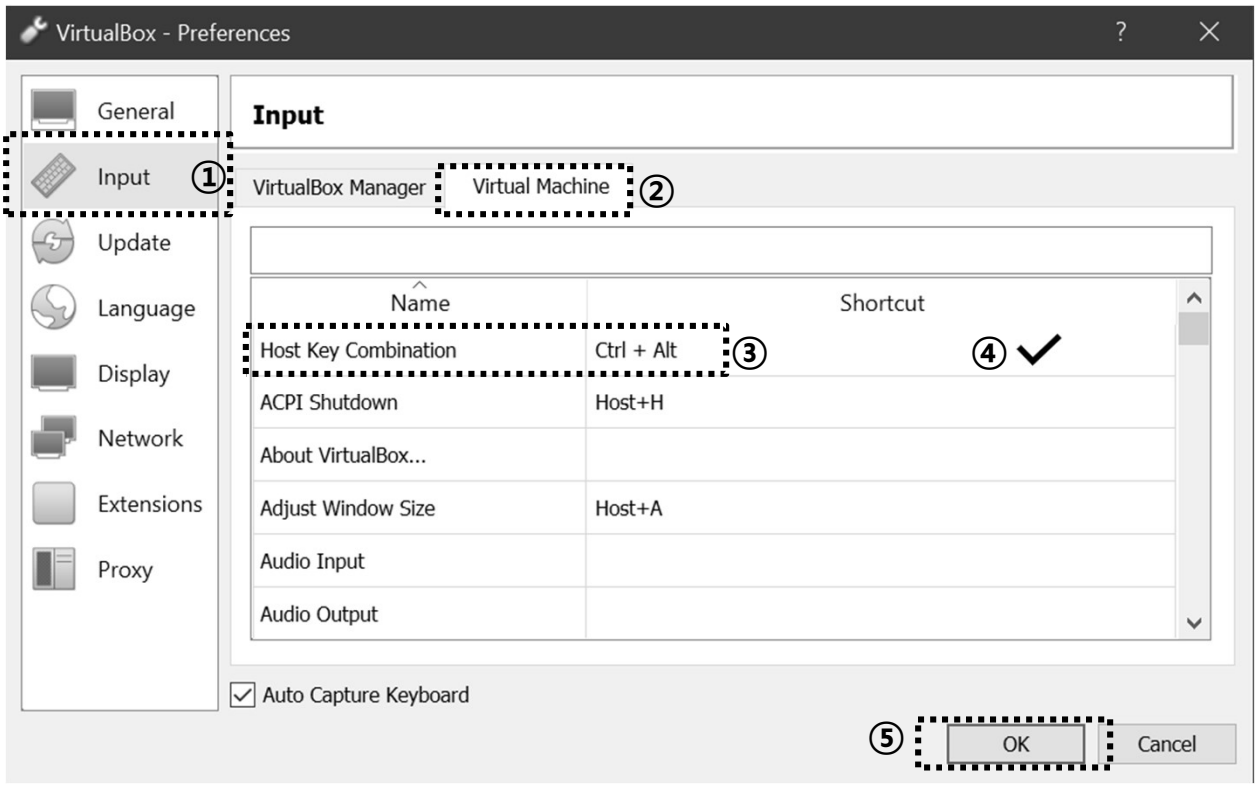
메모:

- 8GB RAM Windows OS 기반 실습 환경 고려
- 필요 Docker Container 버전 고려한 OS 선택 (Ubuntu Server 16.04, CentOS 7)
- CPU 성능과 인터넷 속도를 고려한 소프트웨어 설치
- 기 설치 된 VirtualBox 5.x 호환 실습 가능

1. 실습 환경

❖ VirtualBox 설치

① **VirtualBox-6.0.2-128162-Win.exe 설치** (5.X 실습 가능)
② **설치 후 Virtual Machine input을 키보드 고려 수정** (Ctrl+Alt)
③ **Host Key Combination 확인**
④ **Enter 키로 설정 확인**
⑤ **OK**



메모:

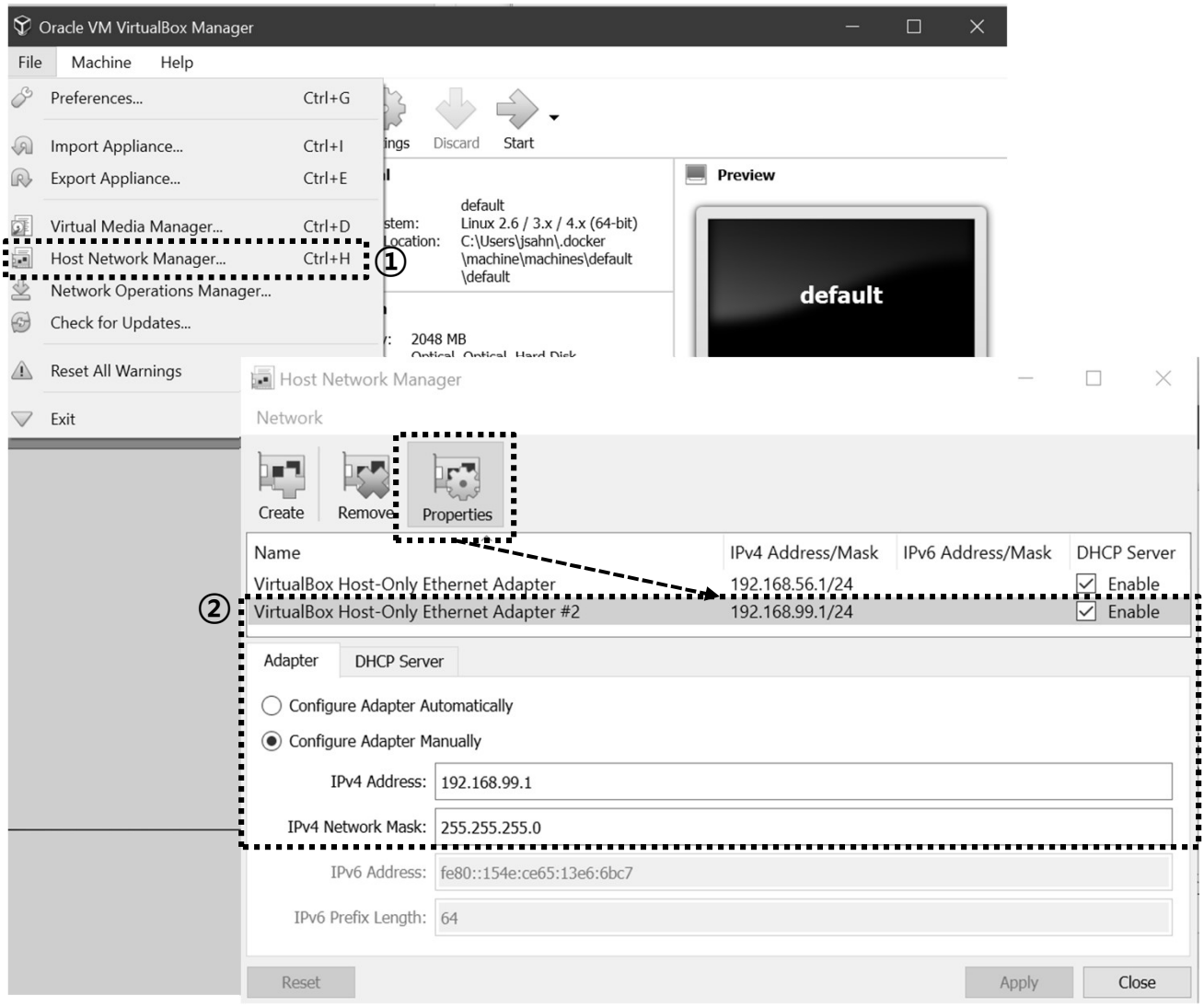
- VirtualBox Download: https://www.virtualbox.org/wiki/Downloads
- 노트북등에서 사용 가능한 키를 설정

1. 실습 환경

❖ VirtualBox Host Network Manager (영문)

① **호스트 네트워크 관리자** (Host Network Manager)

② **사용 어댑터 선택 후 확인** (Enable DHCP Server)



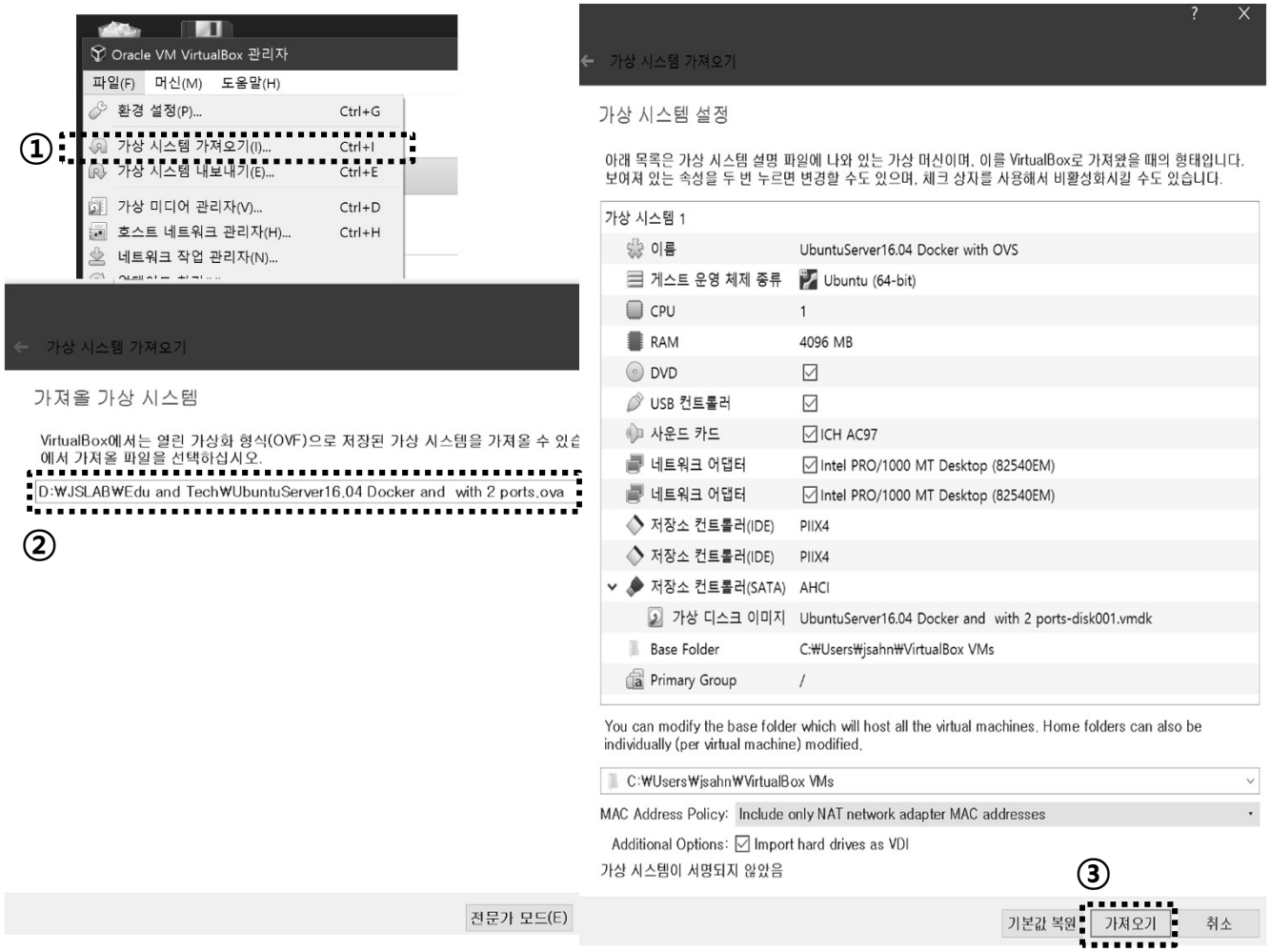
메모:

- 2개 이상의 VM을 Clone하여 사용시 동일 MAC 주소 확인

1. 실습 환경

❖ 가상 시스템 가져오기 (한글)

① 가상 시스템 가져오기
② 가져올 가상시스템 선택 후 확인
③ 가져오기



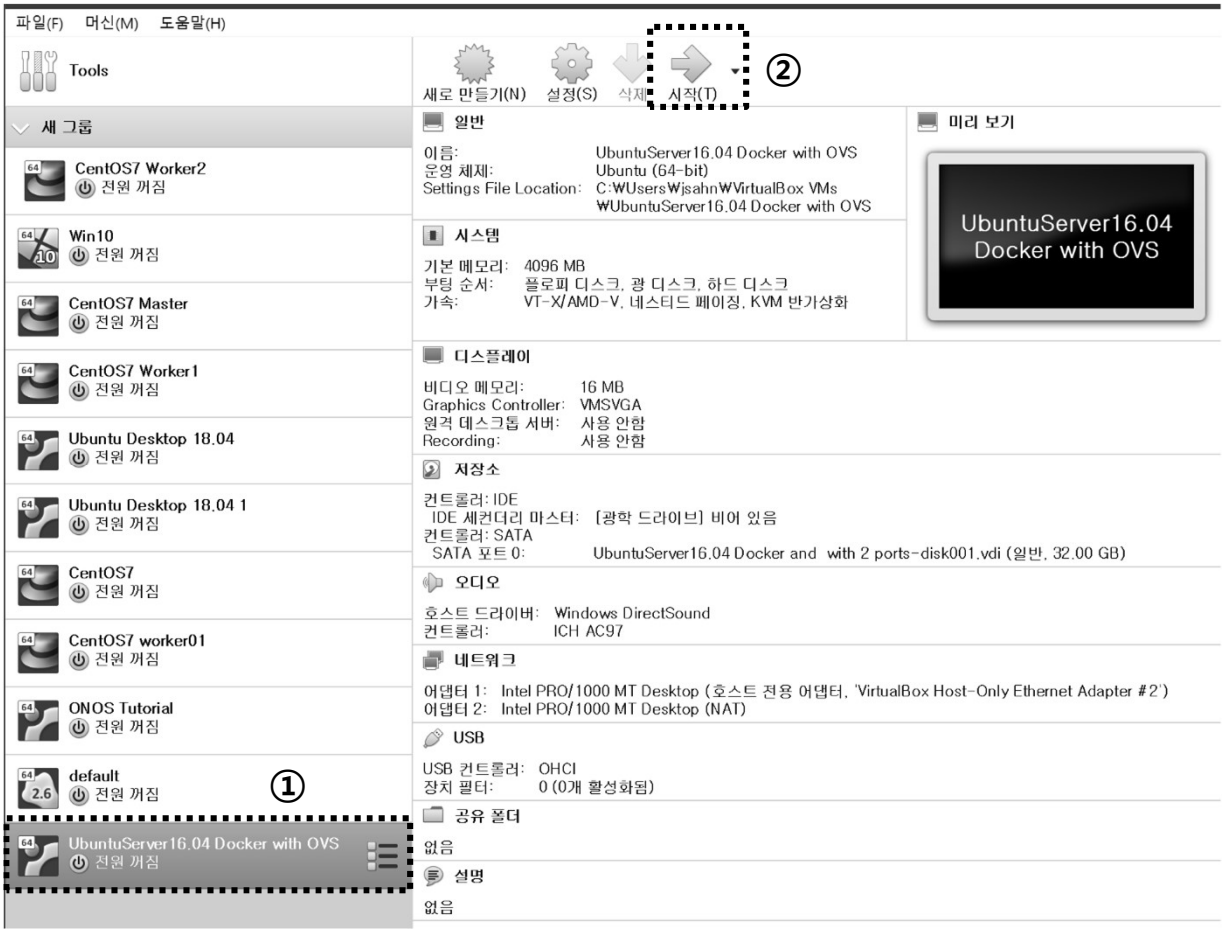
메모:

- **UbuntuServer16.04 Fresh with 2 ports.ova** (인터넷과 CPU/RAM등의 자원이 여유있는 환경에서 사용)
- **UbuntuServer16.04 Docker and OVS with 2 ports.ova** (RAM 8GB이하 환경에서 사용)
- **UbuntuServer16.04 ONOS and Rancher with 2 ports.ova** (OpenFaaS나 Hyperledger 등에 사용)
- **onos-tutorial-1.14.0.ova** (mininet 등에 사용)
- **CentOS 7 Worker/Master for Rancher and K8s** (2개, 8GB RAM 환경에서 설치 실습)

1. 실습 환경

❖ 가상 시스템 시작 (한글)

① 가상시스템 선택
② 시작



메모:

- 관리 제어용 '호스트 전용 어댑터
- 외부 통신용 NAT 어댑터
- SDN 제어기를 위한 동일 VM 가져오기 반복

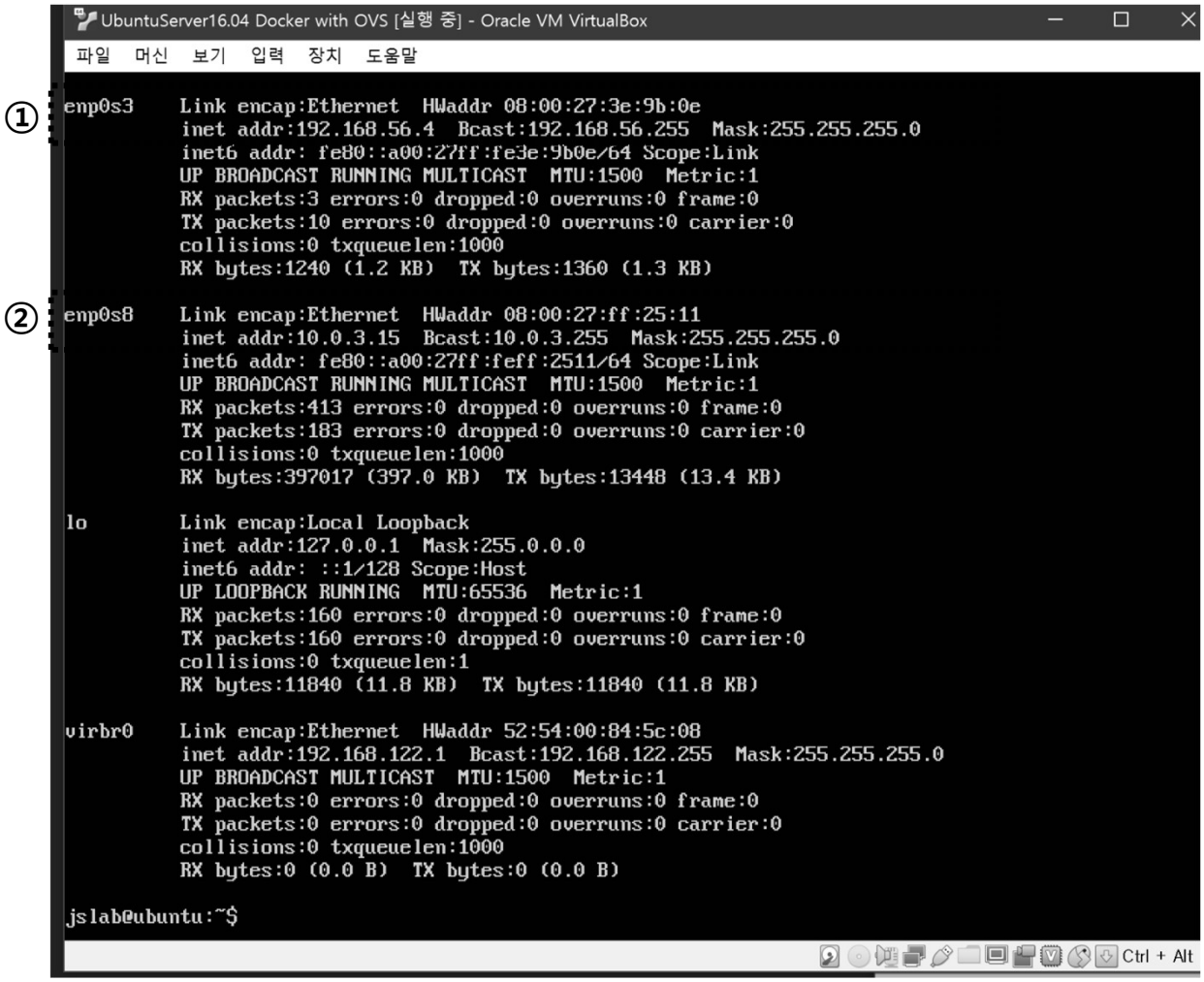
1. 실습 환경

❖ 가상 시스템 IP 주소 확인

① **계정 사용 로그인** (ID/Password: jslab/jslab123)

② **ifconfig**

③ **IP 주소 확인** (호스트 어댑터/NAT)



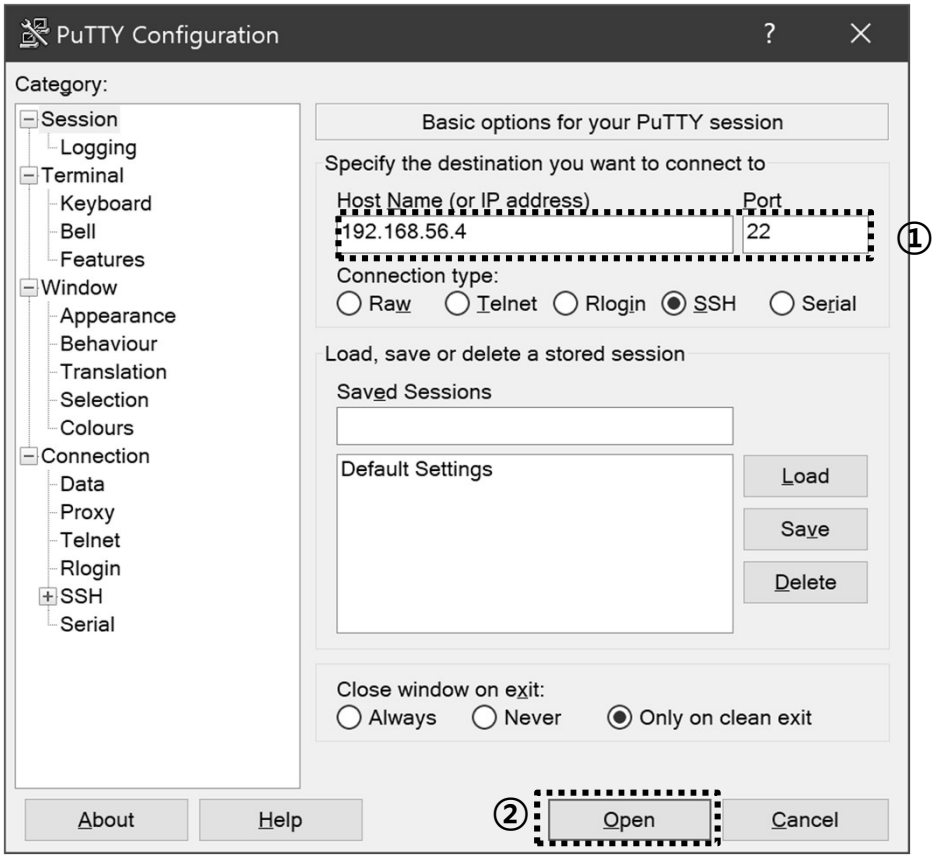
메모:

- Ubuntu Server 16.04의 enp0s3 (Host Adapter 사용 192.168.xx.xx)
- Ubuntu Server 16.04의 enp0s8 NAT 사용 (10.0.3.xx)

1. 실습 환경

❖ SSH 접속

① **putty-64bit-0.70-installer.msi 설치**
② **SuperPuttySetup-1.4.0.9.msi 설치** (선택)
③ **IP 주소 설정 후 Open** (호스트 어댑터 접속)



메모:

- Saved Sessions에 이름을 지정 후 Save 하여 필요시 Load 하여 사용 가능
- 윈도우 설치 Putty는 윈도우 OS 화면과 Copy(복사)/Paste(붙여넣기) 가능

james@jslab.kr

1. 실습 환경

❖ SSH 접속 IP 주소 확인

① Ifconfig

```
Using username "jslab".
jslab@192.168.56.4's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-131-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

122 packages can be updated.
80 updates are security updates.

Last login: Mon Jan 21 08:58:25 2019
/usr/bin/xauth:  file /home/jslab/.Xauthority does not exist
jslab@ubuntu:~$ ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:23:fb:30:91
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.25
          UP BROADCAST MULTICAST  MTU:1500  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)

enp0s3    Link encap:Ethernet  HWaddr 08:00:27:3e:9b:0e
          inet addr:192.168.56.4  Bcast:192.168.56.255  Mask:255.
          inet6 addr: fe80::a00:27ff:fe3e:9b0e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:78 errors:0 dropped:0 overruns:0 frame:0
          TX packets:83 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12292 (12.2 KB)  TX bytes:13763 (13.7 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:ff:25:11
          inet addr:10.0.3.15  Bcast:10.0.3.255  Mask:255.255.255
          inet6 addr: fe80::a00:27ff:feff:2511/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:420 errors:0 dropped:0 overruns:0 frame:0
          TX packets:195 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:397527 (397.5 KB)  TX bytes:14408 (14.4 KB)
```


```
Using username "jslab".
jslab@192.168.56.4's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-131-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

122 packages can be updated.
80 updates are security updates.


Last login: Mon Jan 21 08:58:25 2019
/usr/bin/xauth:  file /home/jslab/.Xauthority does not exist
jslab@ubuntu:~$ ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:23:fb:30:91
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  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)

enp0s3    Link encap:Ethernet  HWaddr 08:00:27:3e:9b:0e
          inet addr:192.168.56.4  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe3e:9b0e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:78 errors:0 dropped:0 overruns:0 frame:0
          TX packets:83 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12292 (12.2 KB)  TX bytes:13763 (13.7 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:ff:25:11
          inet addr:10.0.3.15  Bcast:10.0.3.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feff:2511/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:420 errors:0 dropped:0 overruns:0 frame:0
          TX packets:195 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:397527 (397.5 KB)  TX bytes:14408 (14.4 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:160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:11840 (11.8 KB)  TX bytes:11840 (11.8 KB)

virbr0    Link encap:Ethernet  HWaddr 52:54:00:84:5c:08
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```


메모:

1. 실습 환경

❖ SSH 접속 Open vSwitch 확인

① **sudo apt install -y openvswitch-switch**

② **sudo su** # 암호 필요 jslab123

③ **ovs-vsctl show** # sudo ovs-vsctl show

④ **ps -el | grep ovs**

```
root@ubuntu:/home/jslab# ovs-vsctl show
4ab4737e-b206-4308-9630-f150d5c77e17
    ovs_version: "2.5.5"
root@ubuntu:/home/jslab# ps -el | grep ovs
1 S     0  1563  1562  0  70 -10 -  5059 poll_s ?        00:00:00 ovsdb-server
5 S     0  1594  1593  0  70 -10 -  6201 poll_s ?        00:00:00 ovs-vswitchd
root@ubuntu:/home/jslab#
```

메모:

- sudo ovs-vsctl add-br ovs1
- sudo ovs-vsctl add-br ovs2

1. 실습 환경

❖ SSH 접속 Docker 확인

① **sudo su** # 암호 필요 'jslab123'

② **apt install docker.io** # docker 설치 sudo apt install docker.io

③ **cd /usr/bin** # Install ovs-docker utility.

④ **sudo wget https://raw.githubusercontent.com/openvswitch/ovs/master/utilities/ovs-docker**

⑤ **docker version** # sudo docker version

⑥ **docker info** # sudo docker info

```
jslab@ubuntu:~$ sudo su
[sudo] password for jslab:
root@ubuntu:/home/jslab# docker version
Client:
 Version:           18.06.1-ce
 API version:       1.38
 Go version:        go1.10.4
 Git commit:        e68fc7a
 Built:             Thu Nov 15 21:12:47 2018
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.06.1-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.4
  Git commit:       e68fc7a
  Built:            Sun Nov 11 21:53:22 2018
  OS/Arch:          linux/amd64
  Experimental:     false
root@ubuntu:/home/jslab#
```

메모:

- Docker 와 ovs-docker 설치

1. 실습 환경
2. Host
3. Open vSwitch
4. SDN Controller (Docker)
5. mininet (w/ONOS)
6. Rancher 설치
7. Kubernetes 설치

❖ 부록: Docker

2. Host

❖ Ubuntu Server 16.04 설치 (고정 IP 사용시 선택)

① **ip link show** # Check Interfaces

② **Static IP Address Setting**

③ **Host Name Setting** (ovs, controller)

- SSH Well-known Port 변경 -

```
sudo vi /etc/ssh/sshd_config

# What ports, IPs and protocols we listen for
Port 33322
```

- 계정 암호 변경 -

```
To change the root password:
sudo passwd
To change your user password:
passwd
To change other users password:
sudo passwd USERNAME
```

- 호스트 이름 변경 -

```
/etc/hostname
/etc/hosts
sudo nano /etc/hostname
sudo nano /etc/hosts
** reboot 권장 **
```

- 고정 IP 주소 설정-

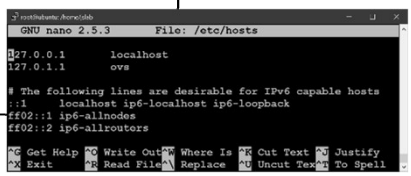
```
sudo vi /etc/network/interfaces

# Iface ens160 inet dhcp
iface ens160 inet static
        address 192.168.0.xx
        netmask 255.255.255.0
        gateway 192.168.0.1
        dns-nameservers 8.8.8.8

cntl+o → enter →  cntl+x
sudo /etc/init.d/networking restart (or reboot)
```

- Root 계정 생성 -

```
sudo -l
passwd
sudo passwd root
```



- Remote for sshd @ Putty -

```
192.168.1.xxx @ Putty for VyOS
ssh jslab@192.168.0.yy
```

- Root 활성화 -

```
sudo su - root
(return with ctrl-d)
```

메모:

- nano 수정 후 저장: Cntl+O → enter → Cntl+X
- Ubuntu Server 루트계정 활성화: sudo passwd root
- VM 이미지 Import 시 네트워크 인터페이스 확인 위한 명령어 'ip link show'
- Root 계정으로 실행 필요시 (sudo 사용 일반 계정은 실행하지 못함)
  루트계정 활성화: sudo passwd root

2. Host

❖ Ubuntu Desktop을 위한 VNC 설치 (선택)

① **sudo su - root**

② **apt-get install gnome-panel gnome-settings-daemon metacity vnc4server** # @root

③ **reboot**

④ **vncserver**

⑤ **vncserver -kill :1**

⑥ **sudo apt install gedit**

⑦ **vi ~/.vnc/xstartup** # /root/.vnc/xstartup @ root user

- gnome-panel &
- gnome-settings-daemon &
- metacity &
- nautilus &

⑧ **vncserver :1**

⑨ **ufw allow 5901/tcp**

⑩ **https://bintray.com/tigervnc/stable/tigervnc** @ Windows

- https://bintray.com/tigervnc/stable/download_file?file_path=vncviewer64-1.9.0.exe @ Windows
- Server IP:Display # @ Windows
- password @ Windows

메모:

- sudo su - root (return with ctrl-d)

2. Host

❖ CentOS 7 (고정 IP 사용시 선택)

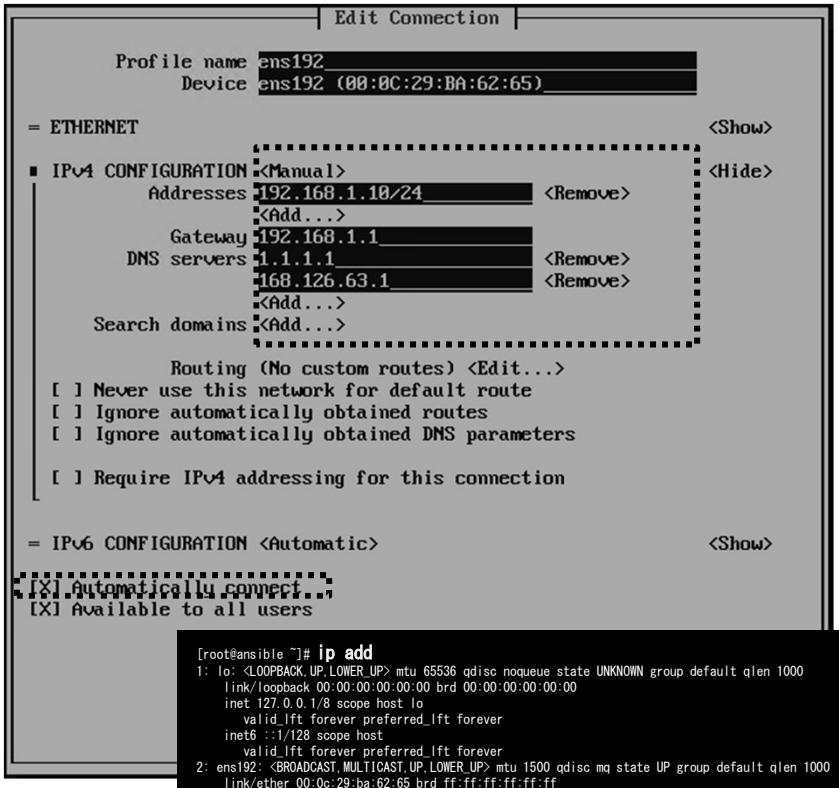
① **nmtui** # IP 주소 설정 192.168.1.10 (Tab 키 사용 이동)

② **ip add** # 설정한 IP 주소 확인 @ Terminal

③ **echo “nameserver 1.1.1.1”>> /etc/resolv.conf** # 선택

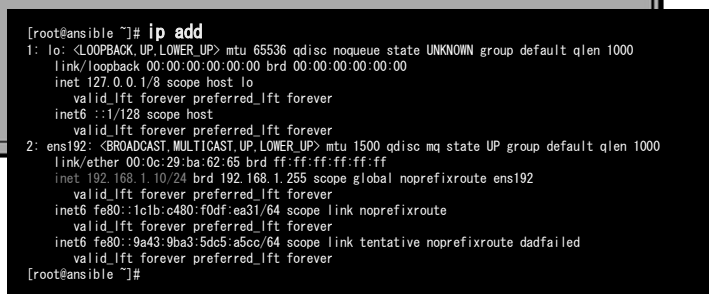
④ **vi /etc/resolv.conf** # dns 주소 1.1.1.1 추가 확인

nmtui 명령어 수행 화면



Activate a connection

IP 주소 설정



메모:

- CentOS 7 Master / Worker01 VM 이미지 복사 후 MAC reset 고려
- 접속 후 계정 사용하여 로그인 계정 (예) : root / jslab123

2. Host

❖ **CentOS 7** (for Rancher / K8s)

① **hostnamectl set-hostname master** # @ master
② **hostnamectl set-hostname worker01** # @ worker01
③ **hostnamectl set-hostname worker02** # @ worker02
④ **hostnamectl set-hostname worker03** # @ worker03
⑤ **su -** # 각 호스트에서 확인

⑥ **nmtui** # IP 주소 설정 192.168.1.1x (Tab 키 사용 이동)
⑦ **IP 주소 변경 후 Deactivate - Activate a Connection**
⑧ **ip add** # 설정한 IP 주소 확인 @ Terminal
⑨ **echo "nameserver 1.1.1.1">> /etc/resolv.conf**
⑩ **cvi /etc/resolv.conf** # dns 주소 1.1.1.1 추가 확인

VM Name	Host Name	IP Address	Interface Name	
Master	master	192.168.56.x0	enp0s3	
Worker01	worker01	192.168.56.x1	enp0s3	
Worker02	worker02	192.168.56.x2	enp0s3	
Worker03	worker03	192.168.56.x3	enp0s3	

메모:
- 다운로드 주소: https://www.centos.org/download/
- 사용 ISO 파일 위치: http://ftp.kaist.ac.kr/CentOS/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso
- SuperPutty 사용 가능

1. 실습 환경
2. Host
3. Open vSwitch
4. SDN Controller (Docker)
5. mininet (w/ONOS)
6. Rancher 설치
7. Kubernetes 설치

❖ 부록: Docker

3. Open vSwitch

❖ Open vSwitch Installation (스위치 2개 예)

① **UbuntuServer16.04 Docker and OVS with 2 ports.ova 사용**
② **ID / Password** # jslab / jslab123
③ **sudo su** # 암호 필요 jslab123
④ **ovs-vsctl show** # sudo ovs-vsctl show
⑤ **ovs-vsctl add-br ovs1** # ovs1
⑥ **ovs-vsctl show**
⑦ **ovs-vsctl add-br ovs2** # ovs2
⑧ **ovs-vsctl show**

```
jslab@ubuntu:~$ sudo su
[sudo] password for jslab:
root@ubuntu:/home/jslab# ovs-vsctl show
4ab4737e-b206-4308-9630-f150d5c77e17
    ovs_version: "2.5.5"
root@ubuntu:/home/jslab# ovs-vsctl add-br ovs1
root@ubuntu:/home/jslab# ovs-vsctl add-br ovs2
root@ubuntu:/home/jslab# ovs-vsctl show
4ab4737e-b206-4308-9630-f150d5c77e17
    Bridge "ovs2"
        Port "ovs2"
            Interface "ovs2"
                type: internal
    Bridge "ovs1"
        Port "ovs1"
            Interface "ovs1"
                type: internal
    ovs_version: "2.5.5"
root@ubuntu:/home/jslab#
```

메모:
- 실습 환경 고려 (실습 장비 RAM 16 GB 이상 시 상위 OVS 버전 사용 가능)
- Open vSwitch 설치: sudo apt install -y openvswitch-switch
- 포트 추가: sudo ovs-vsctl add-port ovs1 patch-ovs1
- 포트 추가: sudo ovs-vsctl add-port ovs2 patch-ovs2
- ps -ef | grep onos

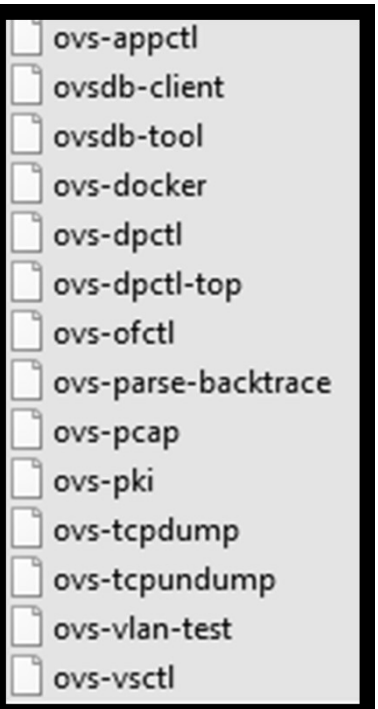
3. Open vSwitch

❖ Open vSwitch Installation

① **ovs-dpctl show** # sudo ovs-dpctl show

② **ovs-ofctl show ovs1** # sudo ovs-ofctl show ovs1

```
root@ubuntu:/home/jslab# ovs-dpctl show
system@ovs-system:
        lookups: hit:0 missed:0 lost:0
        flows: 0
        masks: hit:0 total:1 hit/pkt:0.00
        port 0: ovs-system (internal)
        port 1: ovs1 (internal)
        port 2: ovs2 (internal)
root@ubuntu:/home/jslab# ovs-vsctl show
4ab4737e-b206-4308-9630-f150d5c77e17
    Bridge "ovs2"
        Port "ovs2"
            Interface "ovs2"
                type: internal
    Bridge "ovs1"
        Port "ovs1"
            Interface "ovs1"
                type: internal
    ovs_version: "2.5.5"
root@ubuntu:/home/jslab# ovs-ofctl show ovs1
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000aee2397c3f43
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_nw_src
mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 LOCAL(ovs1): addr:ae:e2:39:7c:3f:43
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
root@ubuntu:/home/jslab#
```



메모:

- sudo ovs-vsctl add-port ovs1 patch-ovs1
- sudo ovs-vsctl add-port ovs2 patch-ovs2
- sudo ovs-vsctl -- set interface patch-ovs1 type=patch options:peer=patch-ovs2
- sudo ovs-vsctl -- set interface patch-ovs2 type=patch options:peer=patch-ovs1

3. Open vSwitch

❖ Using OVS bridge for docker networking

① **ifconfig**

② **cd /usr/bin** # Install ovs-docker utility.

③ **sudo wget https://raw.githubusercontent.com/openvswitch/ovs/master/utilities/ovs-docker**

④ **ovs-vsctl add-br ovs1** # Create an OVS bridge.

⑤ **ifconfig ovs1 173.16.1.1 netmask 255.255.255.0 up**

⑥ **Ifconfig**

```
root@ubuntu:/home/jslab# sudo ifconfig ovs1 173.16.1.1 netmask 255.255.255.0 up
root@ubuntu:/home/jslab# ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:23:fb:30:01
          inet addr:172.17.0.1  Bcast:172.17.255.255  Ma
          UP BROADCAST MULTICAST  MTU:1500  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)

enp0s3    Link encap:Ethernet  HWaddr 08:00:27:3e:9b
          inet addr:192.168.56.4  Bcast:192.168.56.2
          inet6 addr: fe80::a00:27ff:fe3e:9b0e/64 Sc
          UP BROADCAST RUNNING MULTICAST  MTU:1500
          RX packets:1232 errors:0 dropped:0 overrun
          TX packets:998 errors:0 dropped:0 overruns
          collisions:0 txqueuelen:1000
          RX bytes:114040 (114.0 KB)  TX bytes:14209

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:ff:25
          inet addr:10.0.3.15  Bcast:10.0.3.255  Mas
          inet6 addr: fe80::a00:27ff:feff:2511/64 Sc
          UP BROADCAST RUNNING MULTICAST  MTU:1500
          RX packets:797 errors:0 dropped:0 overruns
          TX packets:396 errors:0 dropped:0 overruns
          collisions:0 txqueuelen:1000
          RX bytes:744812 (744.8 KB)  TX bytes:29115
```

```
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:160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:11840 (11.8 KB)  TX bytes:11840 (11.8 KB)

ovs1      Link encap:Ethernet  HWaddr ae:e2:39:7c:3f:43
          inet addr:173.16.1.1  Bcast:173.16.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ace2:39ff:fe7c:3f43/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:578 (578.0 B)

virbr0    Link encap:Ethernet  HWaddr 52:54:00:84:5c:08
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

메모:

- http://containertutorials.com/network/ovs_docker.html

3. Open vSwitch

❖ OVS bridge for docker networking (선택)

① **docker run -t -i --name container1 alpine**

② **/ # ifconfig** # at container1

③ **sudo docker run -t -i -d --name container2 alpine** # New Term

④ **sudo docker ps** # Check container ID

⑤ **sudo ovs-docker add-port ovs1 eth1 container1 --ipaddress=173.16.1.2/24** # Connect the container to OVS bridge

⑥ **sudo ovs-docker add-port ovs1 eth1 container2 --ipaddress=173.16.1.3/24** # Connect the container to OVS bridge

⑦ **sudo docker exec container2 ifconfig**

⑧ **sudo docker exec container2 ping 192.168.0.1**

⑨ **ovs-vsctl add-port ovs1 ethx** # Check for Internet physical port

```
root@ubuntu:/home/jslab# sudo docker run -t -i --name container1 alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
cd784148e348: Pull complete
Digest: sha256:46e71df1e5191ab8b8034c5189e325258ec44ea739bba1e5645cff83c9048ff1
Status: Downloaded newer image for alpine:latest
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16 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:1296 (1.2 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          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:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
/ # ping 1.1.1.1
```

ovs-docker 버그 있음

```
sdn@sdn:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
16ddc135de29        alpine              "/bin/sh"           7
seconds ago         Up 6 seconds
container2
564a21911e7f        alpine              "/bin/sh"           5
minutes ago         Up 5 minutes
container1
```

메모:

- alpine은 리눅스 최소화 도커 이미지 (아마존 클라우드 내 도커허브 접속 가능해야 함)
- alpine 도커 이미지를 사용 2개의 컨테이너를 생성 실행 (container1 , container2)
- **-d** Option 사용/미사용 Putty 사용 2개의 Terminal 접속
- It will be back after docker container
- 미사용 컨테이너 삭제: sudo docker system prune

3. Open vSwitch

❖ Using OVS bridge for docker networking

① **sudo ovs-docker add-port ovs1 eth1 container1 --ipaddress=173.16.1.2/24** # Connect the container to OVS bridge

② **sudo ovs-docker add-port ovs1 eth1 container2 --ipaddress=173.16.1.3/24** # Connect the container to OVS bridge

③ **sudo docker exec container2 ifconfig** # check for Internet

④ **sudo docker exec container2 ping 173.16.1.2**

```
sdn@sdn:/usr/bin$ sudo docker run -t -i --name container1 alpine
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10 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:828 (828.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 6E:58:43:53:F5:D8
          inet addr:173.16.1.2  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:648 (648.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          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:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ #
```

```
/ # ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=118 time=34.857 ms
64 bytes from 8.8.8.8: seq=1 ttl=118 time=241.197 ms
64 bytes from 8.8.8.8: seq=2 ttl=118 time=206.229 ms
```

메모:

- 포트 삭제: sudo ovs-docker del-port ovs2 eth1 container2 --ipaddress=173.16.1.3/24
- It will be back after docker container
- 미사용 컨테이너 삭제: sudo docker system prune
- http://containertutorials.com/network/ovs_docker.html

3. Open vSwitch

❖ Open vSwitch Installation (SDN 제어기 설치 후)

① **sudo ovs-vsctl set-controller ovs1 tcp:192.168.99.xxx:6653**

② **sudo ovs-vsctl set-controller ovs2 tcp:192.168.99.xxx:6653**

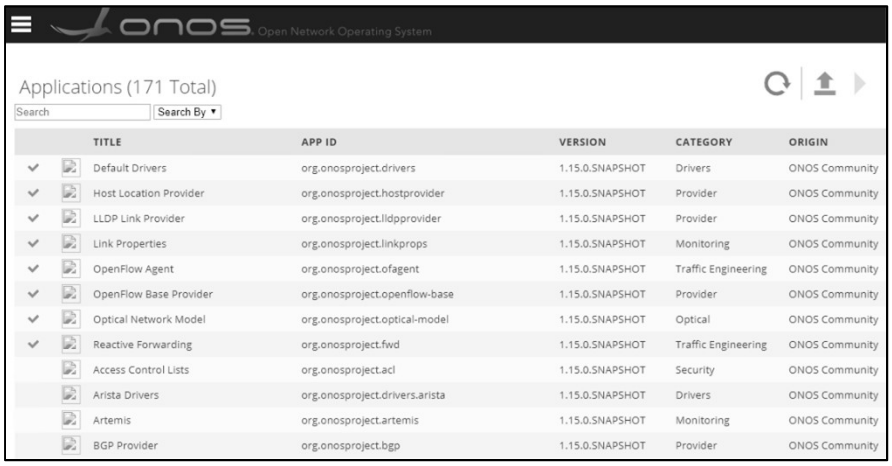
③ **sudo ovs-vsctl show**

④ **http://192.168.99.100:8181/onos/ui** # onos / rocks

⑤ **ssh james@192.168.99.100:8101**

⑥ **Check ONOS App**

```
sdn@sdn:~$ sudo ovs-vsctl set-controller ovs1 tcp:192.168.99.100:6653
sdn@sdn:~$ sudo ovs-vsctl show
9f6387ed-d253-4434-91ed-611082a9c52d
    Bridge "ovs1"
        Controller "tcp:192.168.99.100:6653"
        Port "ovs1"
            Interface "ovs1"
                type: internal
        Port "78d47c80d9c54_l"
            Interface "78d47c80d9c54_l"
    ovs_version: "2.8.1"
```



TITLE	APP ID	VERSION	CATEGORY	ORIGIN
Default Drivers	org.onosproject.drivers	1.15.0.SNAPSHOT	Drivers	ONOS Community
Host Location Provider	org.onosproject.hostprovider	1.15.0.SNAPSHOT	Provider	ONOS Community
LLDP Link Provider	org.onosproject.lldpprovider	1.15.0.SNAPSHOT	Provider	ONOS Community
Link Properties	org.onosproject.linkprops	1.15.0.SNAPSHOT	Monitoring	ONOS Community
OpenFlow Agent	org.onosproject.ofagent	1.15.0.SNAPSHOT	Traffic Engineering	ONOS Community
OpenFlow Base Provider	org.onosproject.openflow-base	1.15.0.SNAPSHOT	Provider	ONOS Community
Optical Network Model	org.onosproject.optical-model	1.15.0.SNAPSHOT	Optical	ONOS Community
Reactive Forwarding	org.onosproject.fwd	1.15.0.SNAPSHOT	Traffic Engineering	ONOS Community
Access Control Lists	org.onosproject.acl	1.15.0.SNAPSHOT	Security	ONOS Community
Arista Drivers	org.onosproject.drivers.arista	1.15.0.SNAPSHOT	Drivers	ONOS Community
Artemis	org.onosproject.artemis	1.15.0.SNAPSHOT	Monitoring	ONOS Community
BGP Provider	org.onosproject.bgp	1.15.0.SNAPSHOT	Provider	ONOS Community

메모:

- sudo docker run -t -d -p 8181:8181 -p 8101:8101 -p 6653:6653 --name onos1 onosproject/onos
- sudo docker run -t -d -p 2181:8181 -p 2101:8101 -p 2653:6653 --name onos2 onosproject/onos # the second ONOS
- Check ONOS App

❖ OVS 스위치간 연결 (선택)

① **sudo ovs-vsctl add-port ovs1 patch-ovs1**

② **sudo ovs-vsctl add-port ovs2 patch-ovs2**

③ **sudo ovs-vsctl -- set interface patch-ovs1 type=patch options:peer=patch-ovs2**

④ **sudo ovs-vsctl -- set interface patch-ovs2 type=patch options:peer=patch-ovs1**

```
sdn@sdn:~$ sudo ovs-vsctl show
9f6387ed-d253-4434-91ed-611082a9c52d
    Bridge "ovs1"
        Controller "tcp:192.168.99.100:6653"
            is_connected: true
        Port "patch-ovs1"
            Interface "patch-ovs1"
                type: patch
                options: {peer="patch-ovs2"}
        Port "ovs1"
            Interface "ovs1"
                type: patch
                options: {peer="ovs2"}
        Port "78d47c80d9c54_l"
            Interface "78d47c80d9c54_l"
    Bridge "ovs2"
        Controller "tcp:192.168.99.100:6653"
            is_connected: true
        Port "patch-ovs2"
            Interface "patch-ovs2"
                type: patch
                options: {peer="patch-ovs1"}
        Port "ce66ad158d7e4_l"
            Interface "ce66ad158d7e4_l"
        Port "ovs2"
            Interface "ovs2"
                type: patch
                options: {peer="ovs1"}
    ovs_version: "2.8.1"
```

메모:

- Port 서로 연결 전에는 오류 발생 (정상)
- ONOS 에서 연결 확인

1. 실습 환경
2. Host
3. Open vSwitch
4. SDN Controller (Docker)
5. mininet (w/ONOS)
6. Rancher 설치
7. Kubernetes 설치

❖ 부록: Docker

4. SDN Controller

❖ 실습 개요 – 도커 컨테이너 네트워킹 구성

① **생성한 ovs1에 Ping 가능한 리눅스 OS 구동 컨테이너 접속**

sudo ovs-docker add-port ovs1 eth1 container1 --ipaddress=173.16.1.2/24

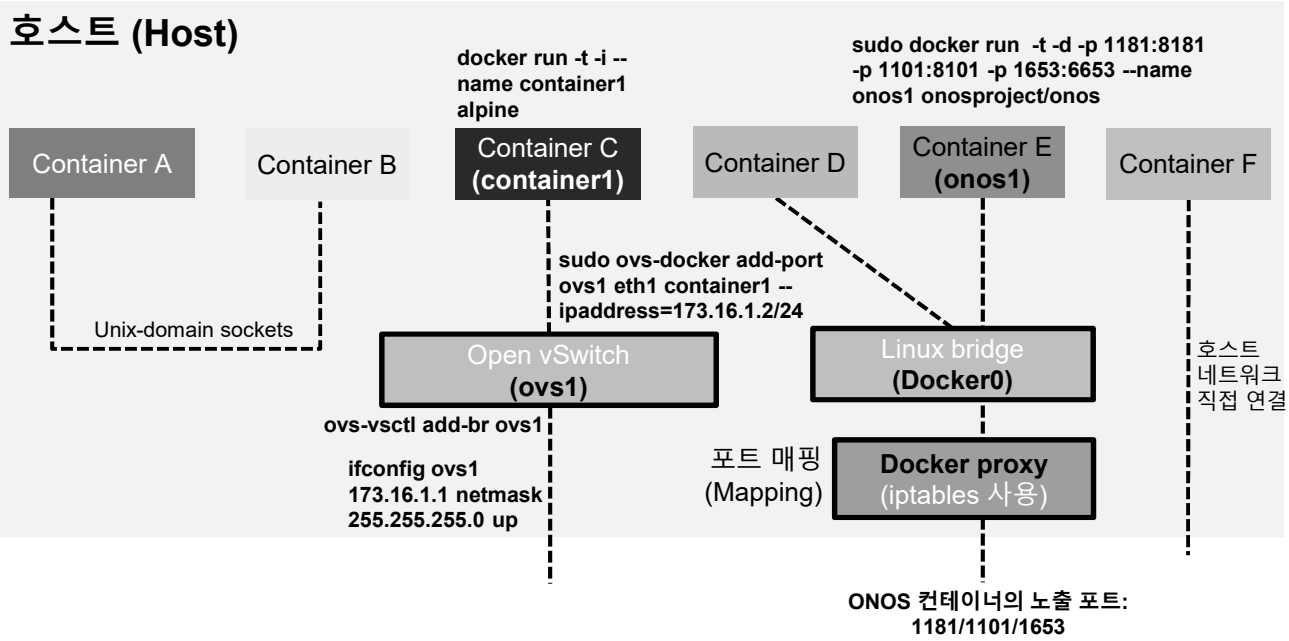
② **ovs1 스위치에 호스트 외부 접속 인터페이스 생성**

ifconfig ovs1 173.16.1.1 netmask 255.255.255.0 up

③ **onos 컨테이너 생성시 노출포트 지정 생성 (8181, 8101, 6653)**

sudo docker run -t -d -p 1181:8181 -p 1101:8101 -p 1653:6653 --name onos1 onosproject/onos

Docker network option과 구성

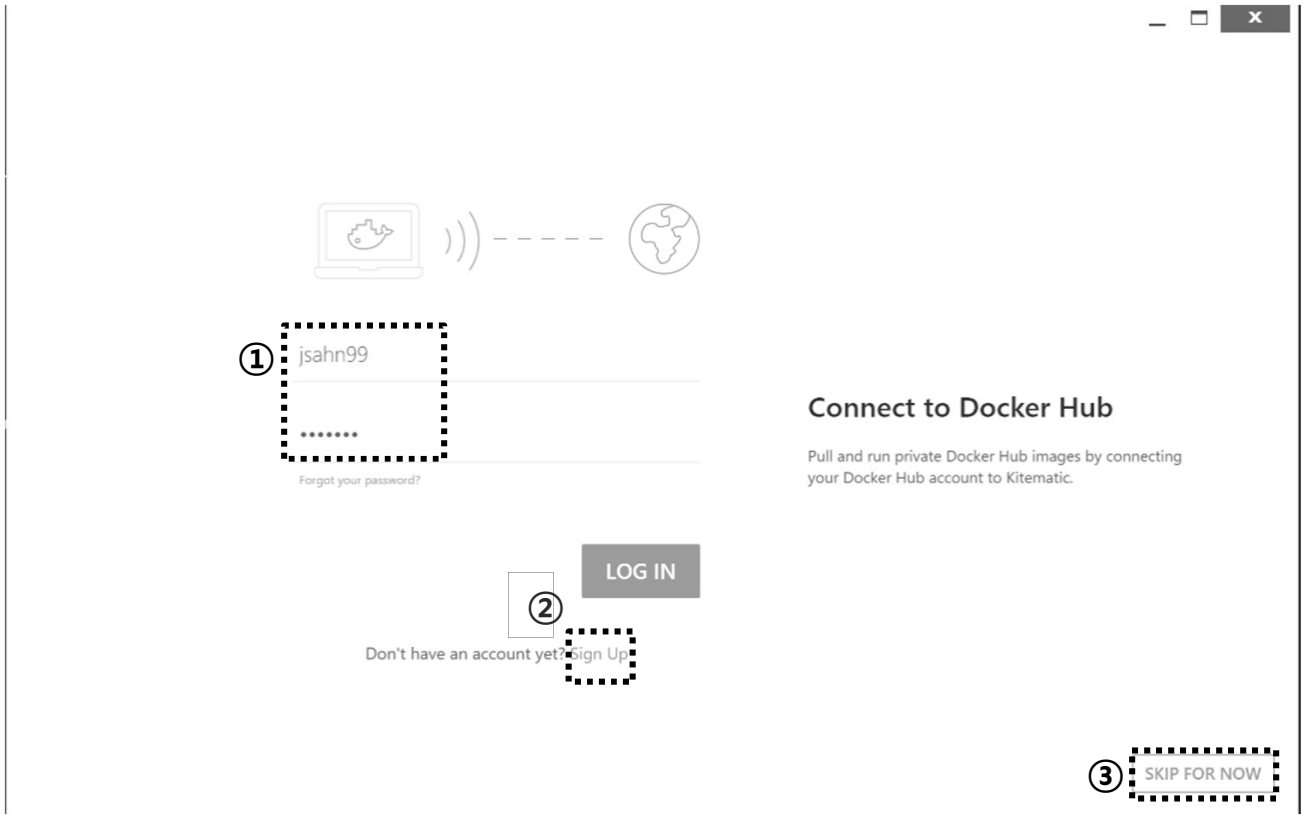


메모:

4. SDN Controller

❖ Docker Toolbox 사용 (Docker Hub 접속)

① **사용자 계정 'ID/Password'**
② **Sign Up 가능**
③ **Skip 가능 'SKIP FOR NOW'**



메모:

- Docker Toolbox 설치 (윈도우/맥 설치 가능) : DockerToolbox.exe 실행
- 기설치 DockerToolbox 사용 가능

james@jslab.kr

4. SDN Controller

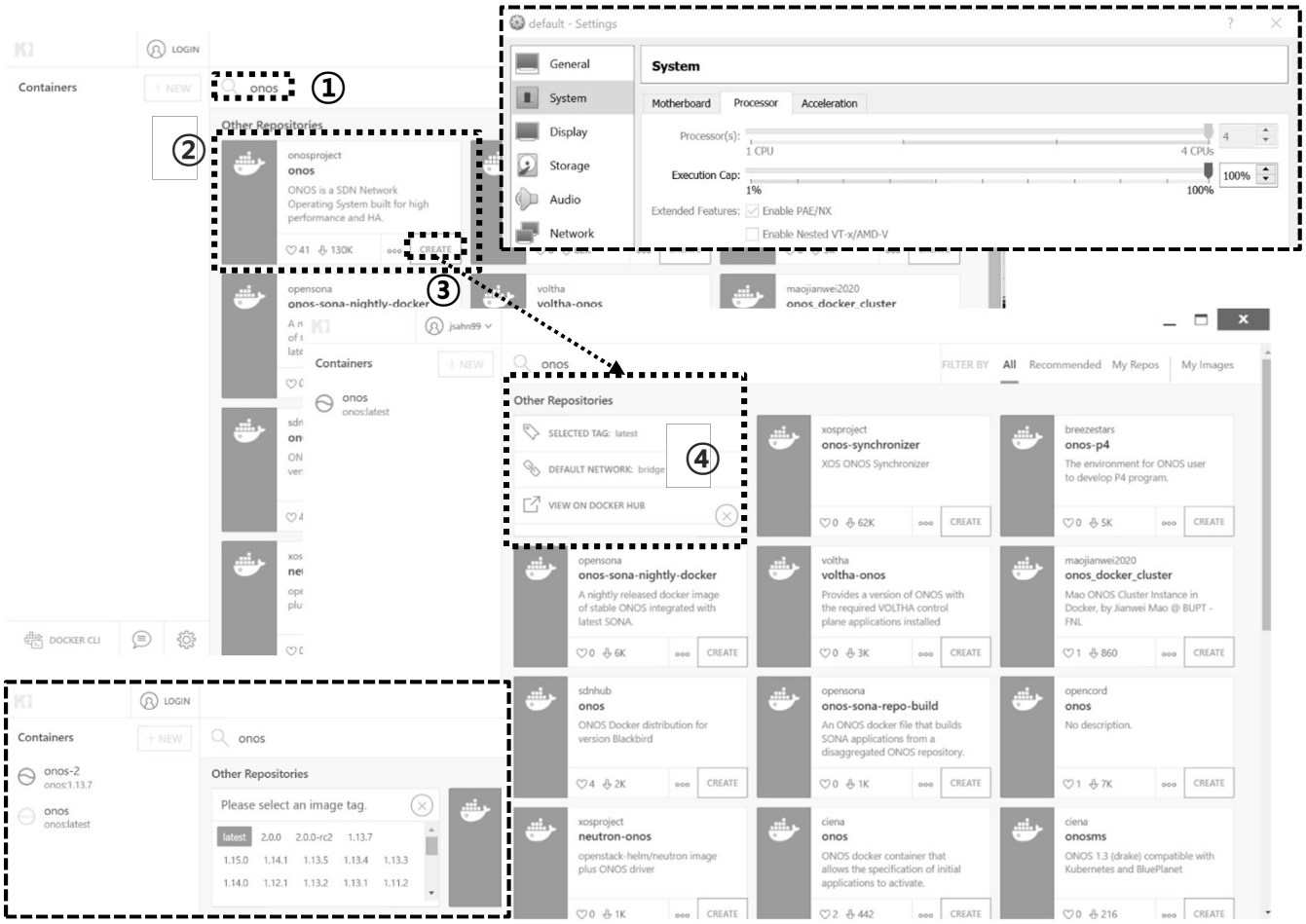
❖ Docker Toolbox 사용 SDN Controller 검색 (ONOS/ODL)

① **Search 'onos'**

② **Check 'onosproject/onos'**

③ **Check '∘∘∘'**

④ **Check 'Default Network'**



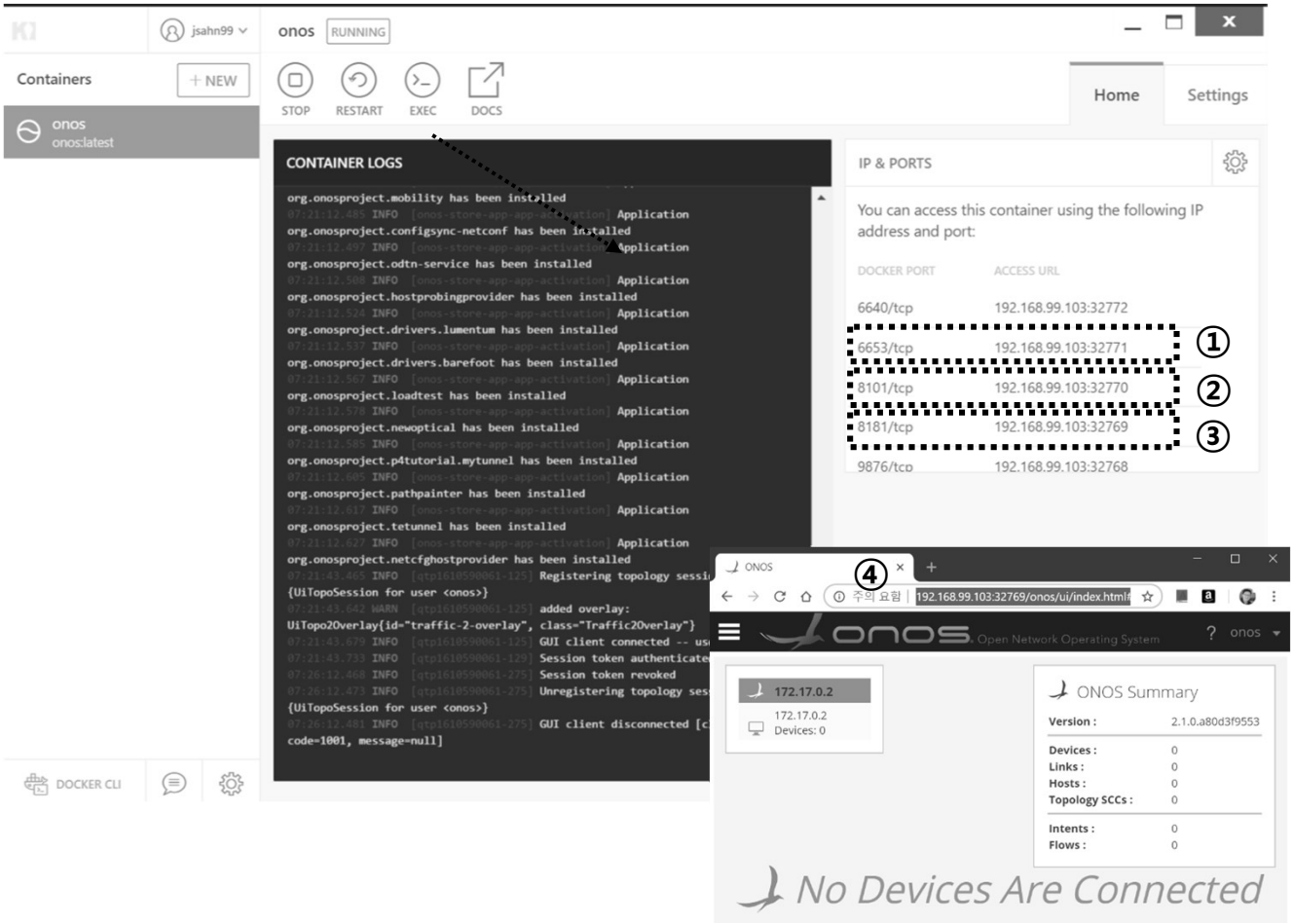
메모:

- 도커 ONOS 검색(Secrch) CLI: sudo docker search onos
- 도커 ODL 검색(Secrch) CLI: sudo docker search odl
- ONOS 도커 실행 CLI: sudo docker run -t -d -p 1181:8181 -p 1101:8101 -p 1653:6653 --name onos1 onosproject/onos
- 대형 망 연결 시 리눅스 커널 제공 VM의 vCPU 개수 증가 및 안정 버전 선택 권장

4. SDN Controller

❖ Docker Hub 접속 ONOS 도커 컨테이너

① **SDN 컨트롤러와 스위치 연결 6653/tcp 변환 확인 32771/tcp**
② **CLI 연결 포트 8101/tcp 확인 32770/tcp**
③ **WEB 연결 8181/tcp 확인 32769/tcp**
④ **http://192.168.99.103:32769/onos/ui** # 계정 onos / rocks



메모:

- http://IP Address:8181/onos/ui (예): http://192.168.99.103:32769/onos/ui
- OVS1 스위치의 ONOS 연결: sudo ovs-vsctl set-controller ovs1 tcp:192.168.99.103:32771
- OVS2 스위치의 ONOS 연결: sudo ovs-vsctl set-controller ovs2 tcp:192.168.99.103:32771

❖ Linux Host 의 ONOS 컨테이너 실행

① **UbuntuServer16.04 Docker and OVS with 2 ports.ova 사용**

② **sudo docker run -t -d -p 8181:8181 -p 8101:8101 -p 6653:6653 --name onos1 onosproject/onos**

③ **sudo docker run -t -d -p 1181:8181 -p 1101:8101 -p 1653:6653 --name onos1 onosproject/onos** # student 1

④ **sudo docker run -t -d -p 2181:8181 -p 2101:8101 -p 2653:6653 --name onos2 onosproject/onos** # student 2

⑤ **sudo docker run -t -d -p 3181:8181 -p 3101:8101 -p 3653:6653 --name onos3 onosproject/onos** # student 3

⑥ **sudo docker run -t -d -p 4181:8181 -p 4101:8101 -p 4653:6653 --name onos4 onosproject/onos** # student 4

⑦ **sudo docker run -t -d -p 5181:8181 -p 5101:8101 -p 5653:6653 --name onos5 onosproject/onos** # student 5

⑧ **sudo docker run -t -d -p 6181:8181 -p 6101:8101 -p 6653:6653 --name onos6 onosproject/onos** # student 6

⑨ **sudo docker run -t -d -p 7181:8181 -p 7101:8101 -p 7653:6653 --name onos7 onosproject/onos** # student 7

⑩ **sudo docker run -t -d -p 9181:8181 -p 9101:8101 -p 9653:6653 --name onos9 onosproject/onos** # student 9

⑪ **sudo docker run -t -d -p 10181:8181 -p 10101:8101 -p 10653:6653 --name onos10 onosproject/onos** # student 10

메모:

- **도커허브 연결 불가능 시** 'UbuntuServer16.04 ONOS and Rancher with 2 ports.ova' 사용
- 별도의 VM 사용 (아마존 클라우드 AWS의 Docker Hub 연결)
- 한 개의 호스트에서 ONOS 복수 제공 가능
- RUN: **run** 실행 시 image가 없는 경우 **pull** (Docker Hub 접속 이미지 다운로드) → **create** (컨테이너 생성) → **start** (컨테이너 실행)

4. SDN Controller

❖ ONOS ← → OVS 연결 확인

① **sudo ovs-vsctl set-controller ovs1 tcp:192.168.99.xxx:6653**

② **sudo ovs-vsctl set-controller ovs2 tcp:192.168.99.xxx:6653**

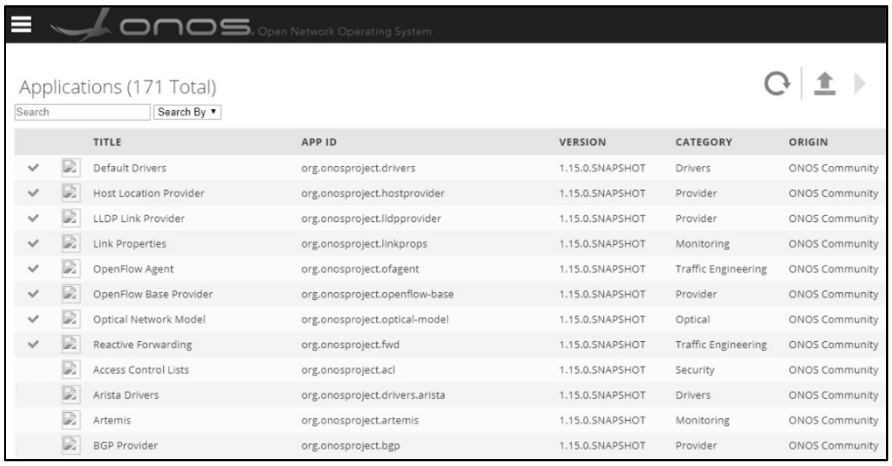
③ **sudo ovs-vsctl show**

④ **http://192.168.99.xxx:8181/onos/ui** # onos / rocks

⑤ **ssh james@192.168.99.xxx:8101** # putty 사용 가능

⑥ **Check ONOS App**

```
sdn@sdn:~$ sudo ovs-vsctl set-controller ovs1 tcp:192.168.99.100:6653
sdn@sdn:~$ sudo ovs-vsctl show
9f6387ed-d253-4434-91ed-611082a9c52d
    Bridge "ovs1"
        Controller "tcp:192.168.99.100:6653"
        Port "ovs1"
            Interface "ovs1"
                type: internal
        Port "78d47c80d9c54_l"
            Interface "78d47c80d9c54_l"
    ovs_version: "2.8.1"
```



메모:

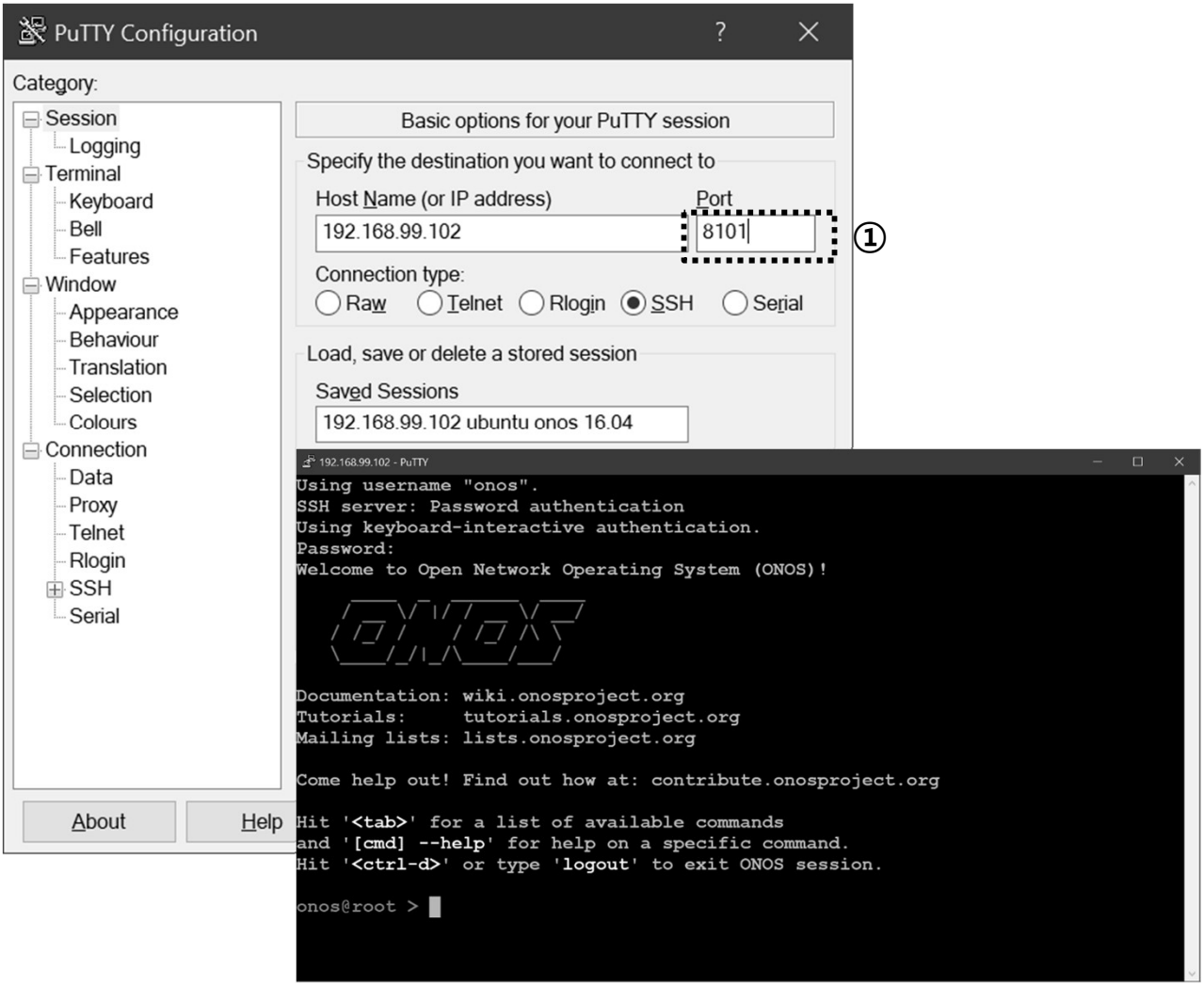
- sudo docker run -t -d -p 8181:8181 -p 8101:8101 -p 6653:6653 --name onos1 onosproject/onos
- sudo docker run -t -d -p 2181:8181 -p 2101:8101 -p 2653:6653 --name onos2 onosproject/onos # the second ONOS
- Check ONOS App

4. SDN Controller

❖ ONOS ← → OVS 연결 확인

① **ssh <u>james@192.168.99.xxx:8101</u>** # putty 사용 가능

② **ID / Password** (onos / rocks)



메모:

- sudo docker run -t -d -p 8181:8181 -p 8101:8101 -p 6653:6653 --name onos1 onosproject/onos
- sudo docker run -t -d -p 2181:8181 -p 2101:8101 -p 2653:6653 --name onos2 onosproject/onos # the second ONOS
- Check ONOS App

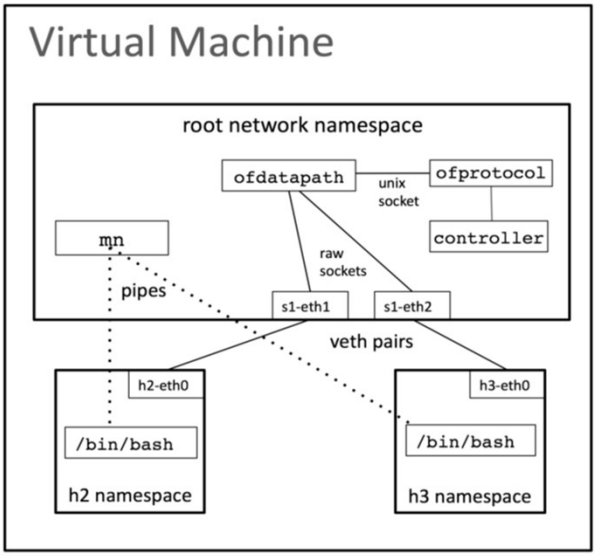
1. 실습 환경
2. Host
3. Open vSwitch
4. SDN Controller (Docker)
5. mininet (w/ONOS)
6. Rancher 설치
7. Kubernetes 설치

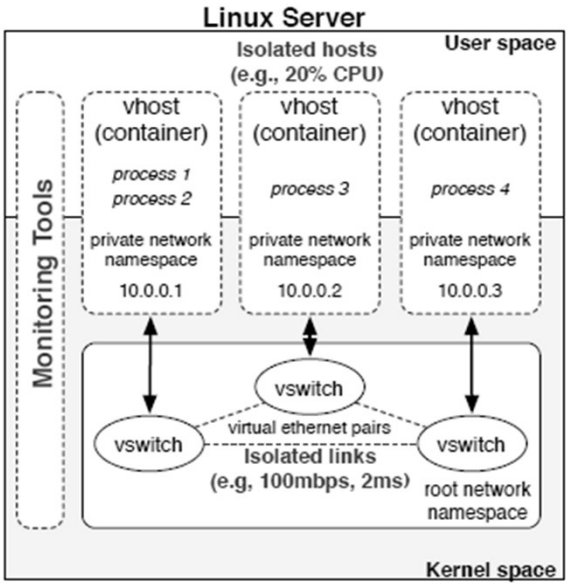
❖ 부록: Docker

5. mininet (w/ONOS)

❖ Mininet 구성

- 1 대의 PC에서 가상 네트워크 환경을 제공
- 실제 커널을 컨테이너 기술 기반으로 스위치 애플리케이션 코드를 사용
- 명령어, UI, 파이손(Python) 인터페이스 제공
- 오픈플로우(OpenFlow) 기능 포함



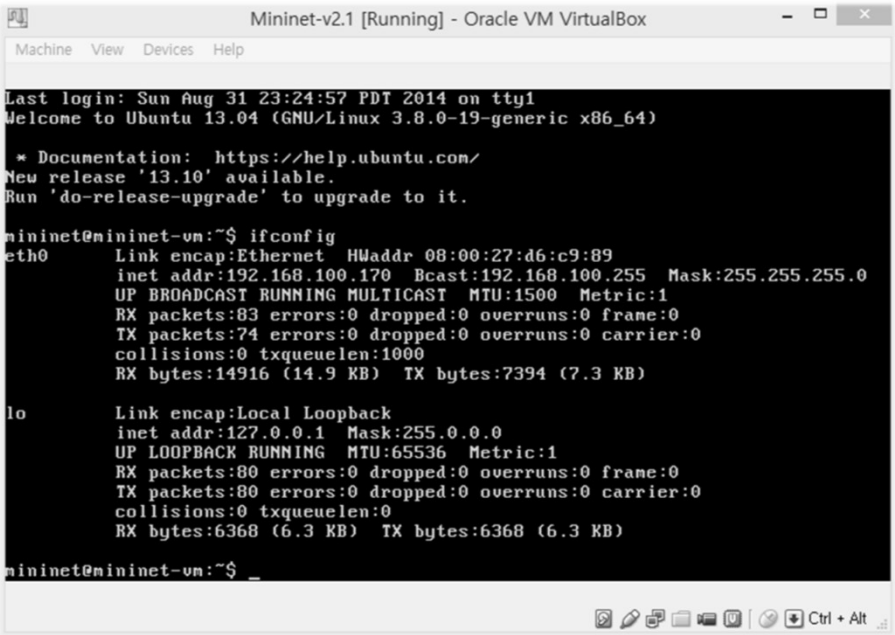


메모:

5. mininet (w/ONOS)

❖ Mininet 설치

① **Run 'Oracle VM VirtualBox Manager'**
② **Start VM ‘Mininet-v2.2.2**
③ **User ID/Password: mininet/mininet**
④ **ifconfig (@ VM ‘Mininet-v2.2.2’)**
⑤ **$ sudo mn --topo single,3 --mac –controller=remote, ip=x.x.x.x, port=6633**
⑥ **h1 ping h2**
⑦ **mininet> exit**
⑧ **$ sudo mn –c**



메모:

- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip
- 실습에서는 mininet이 설치 된 VM **‘onos-tutorial-1.14.0.ova’** 사용
- ‘가져오기’ 시에 Network 확인 필요

5. mininet (w/ONOS)

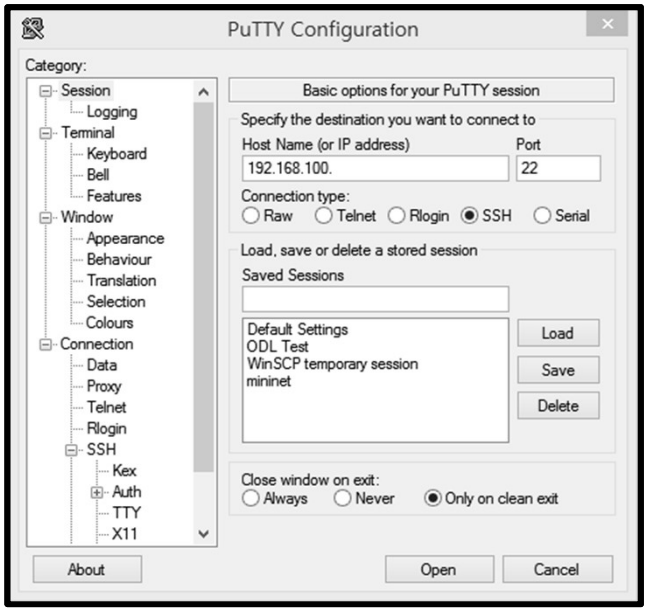
❖ Mininet GUI 실행 (선택)

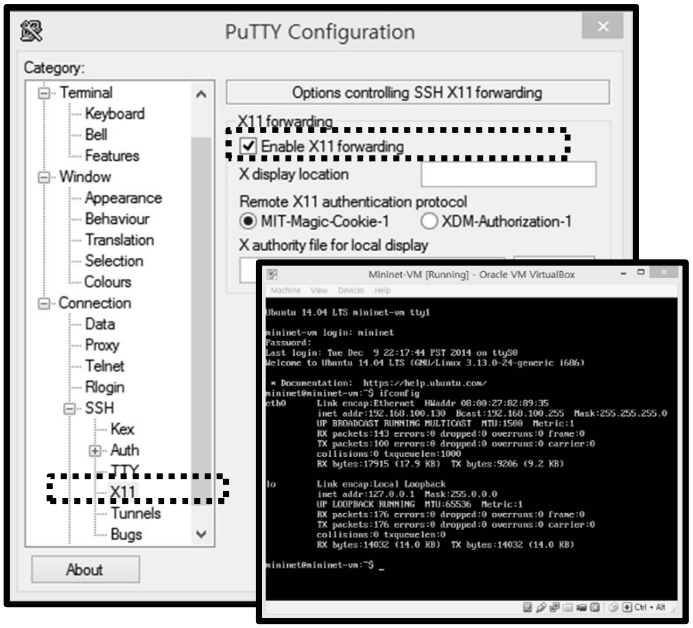
① **Xming Installation**

② **PuTTY Installation**

③ **PuTTY Configuration** (w/X11)

(GUI 기반 WireShark이나 호스트 터미널 창들의 구동 가능하며, root 권한 'sudo' 명령어가 필요 할 수 있음)





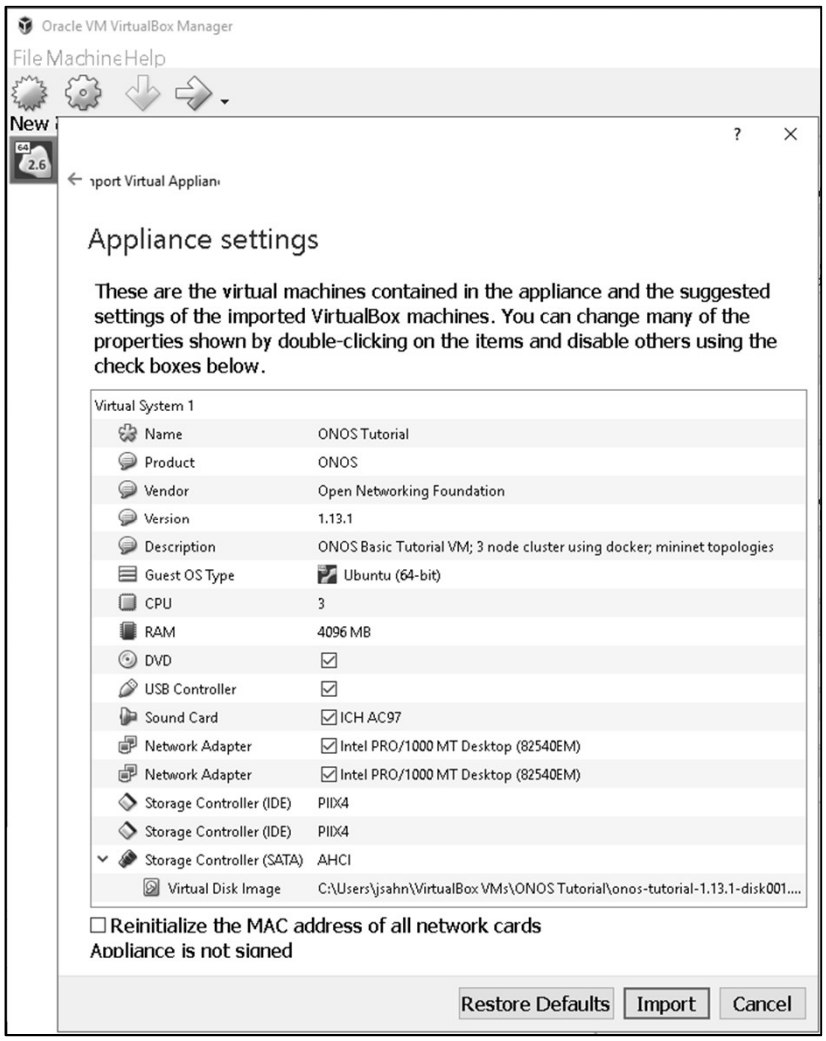
메모:

- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip

5. mininet (w/ONOS)

❖ Import ONOS Tutorial OVA @ VirtualBox

① **Windows, Max OS X, Linux 에서 설치 가능**
② **RAM 8GB 이상, HDD 20GB 이상**
③ **VirtualBox 설치 후 ONOS OVA 파일 import**



메모:

- Basic ONOS Tutorial: https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial
- ONOS Tutorial OVA: https://downloads.onosproject.org/vm/onos-tutorial-1.13.1.ova
- VirtualBox Download: https://www.virtualbox.org/wiki/Downloads

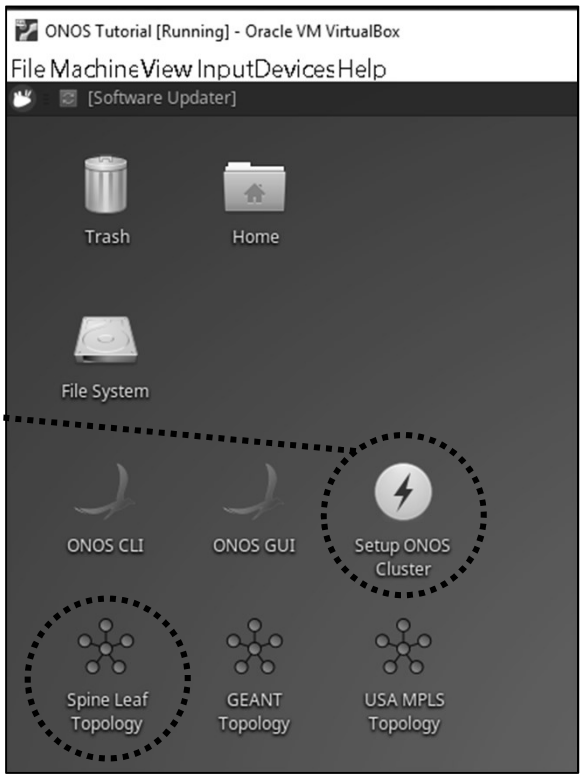
5. mininet (w/ONOS)

❖ Launch Mininet

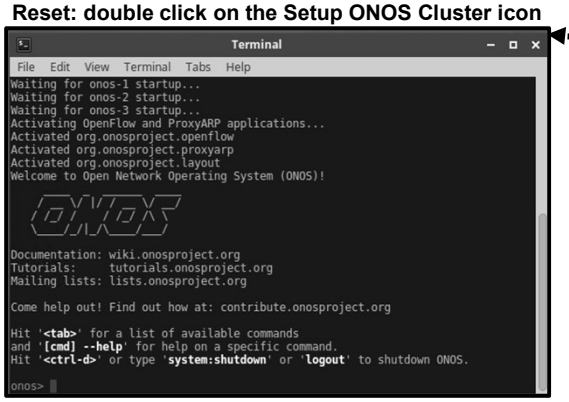
① **User ID / Password:** SDN User (sdn) / rocks
② **Reset:** double click on the **Setup ONOS Cluster** icon
③ **Start Mininet:** double click on the Spine Leaf Topology icon
④ **Start ONOS GUI:** double click on the ONOS GUI icon

Login: User ID / Password

데스크탑 화면



Reset: double click on the Setup ONOS Cluster icon



메모:

- **Setup ONOS Cluster icon**: Simply, click on the Setup ONOS Cluster icon on your desktop and this will reset ONOS cluster to its initial state. Double click the Setup ONOS Cluster icon now and wait for ONOS to start-up.
- basic features of ONOS version 1.14 (Owl release)

5. mininet (w/ONOS)

❖ Launch ONOS GUI

① **lfconfig**

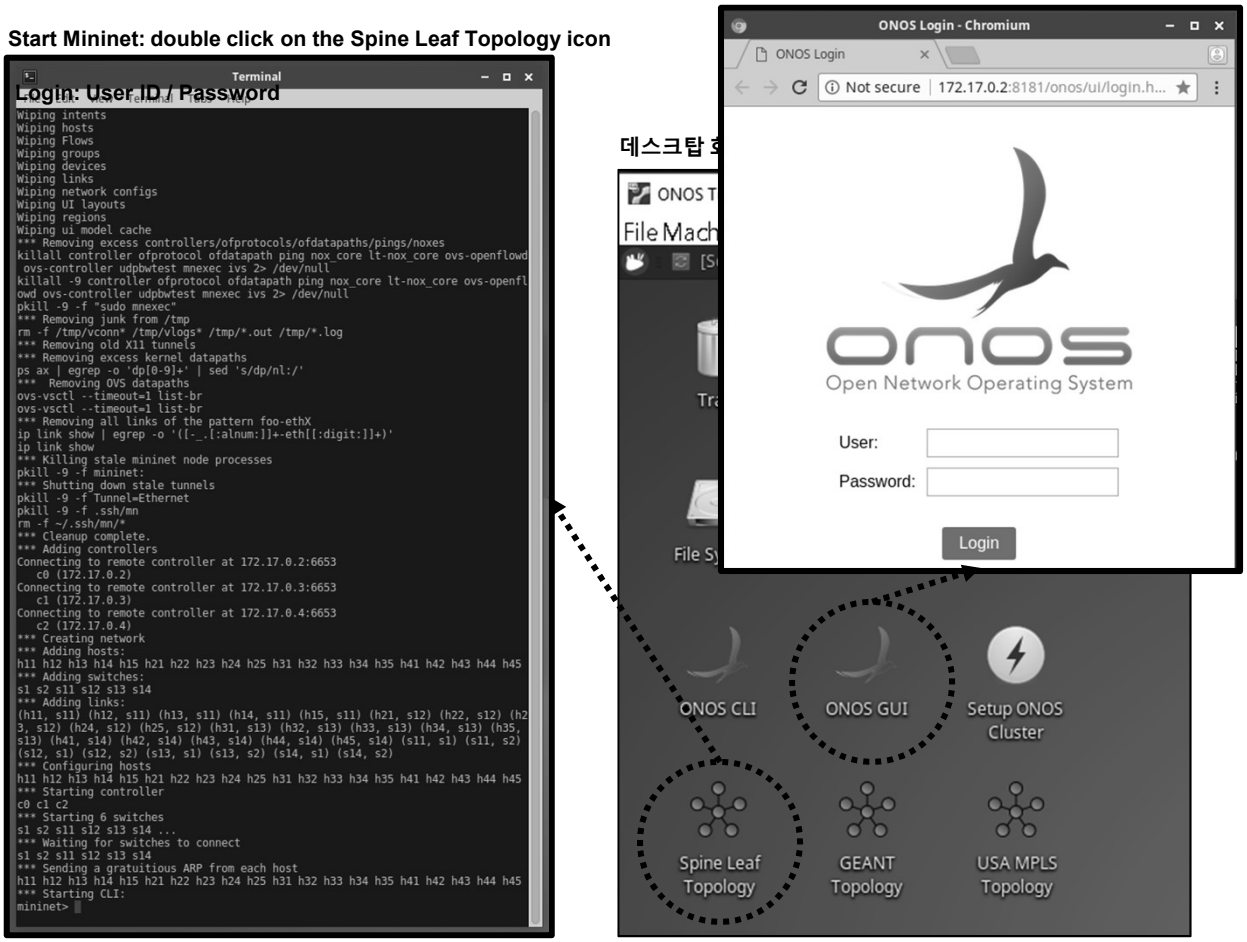
② **docker ps**

③ **Start ONOS GUI:** double click on the ONOS GUI icon

④ **User ID / Password:** onos / rocks

Start Mininet: double click on the Spine Leaf Topology icon

Login: User ID / Password



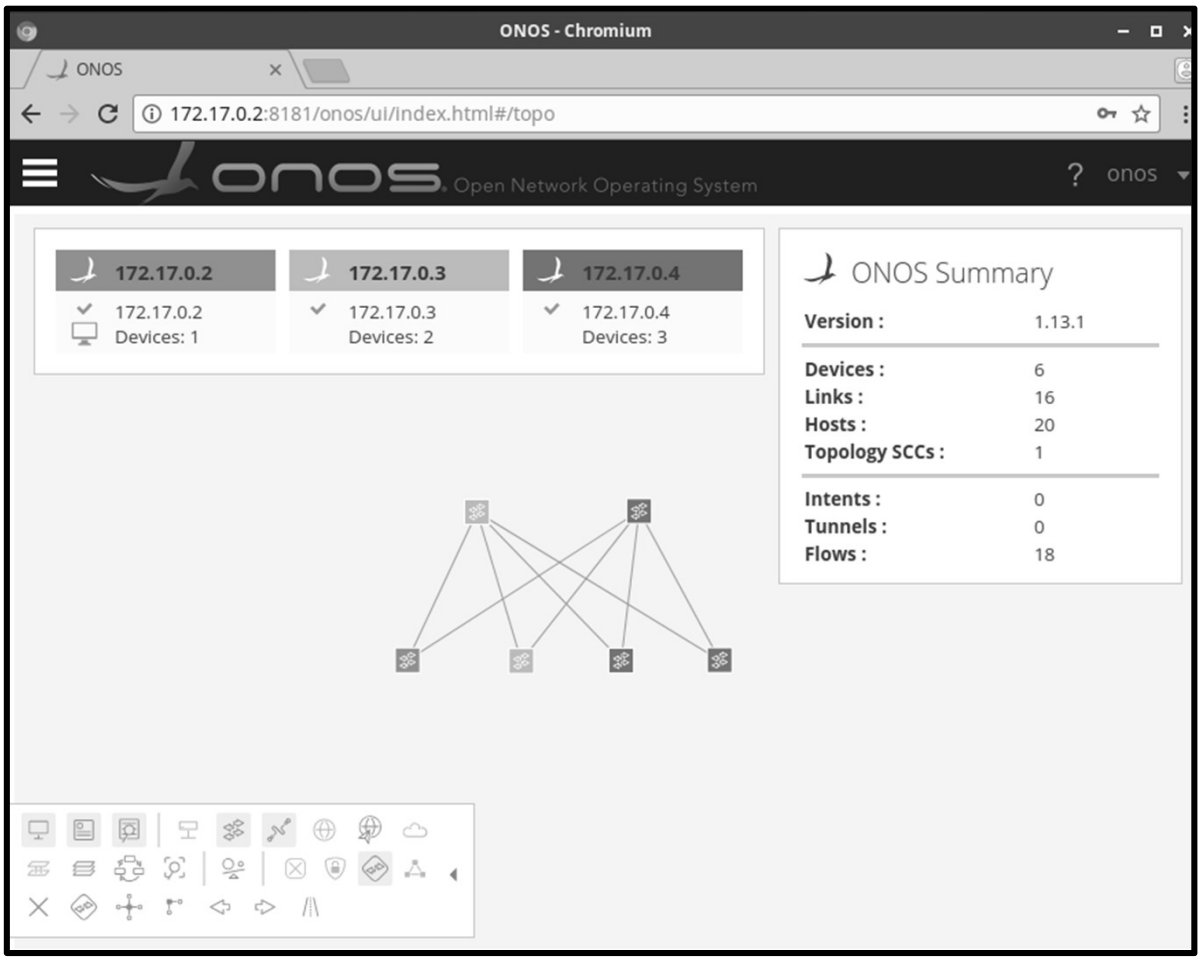
메모:

- **type Ctrl-D or exit in the mininet prompt.**

5. mininet (w/ONOS)

❖ ONOS GUI 명령어

① lfconfig
② docker ps
③ / key, L key, H Key, E key



메모:

- **Mastership re-balancing:** The network devices will be roughly equally divided between all nodes in the ONOS cluster. To do this from the GUI, press the E key.

5. mininet (w/ONOS)

❖ Reactive Forwarding app 실행

① **mininet> h11 ping -c3 h41** # xterm h11 h41

② **Check ONOS Applications**

메모:

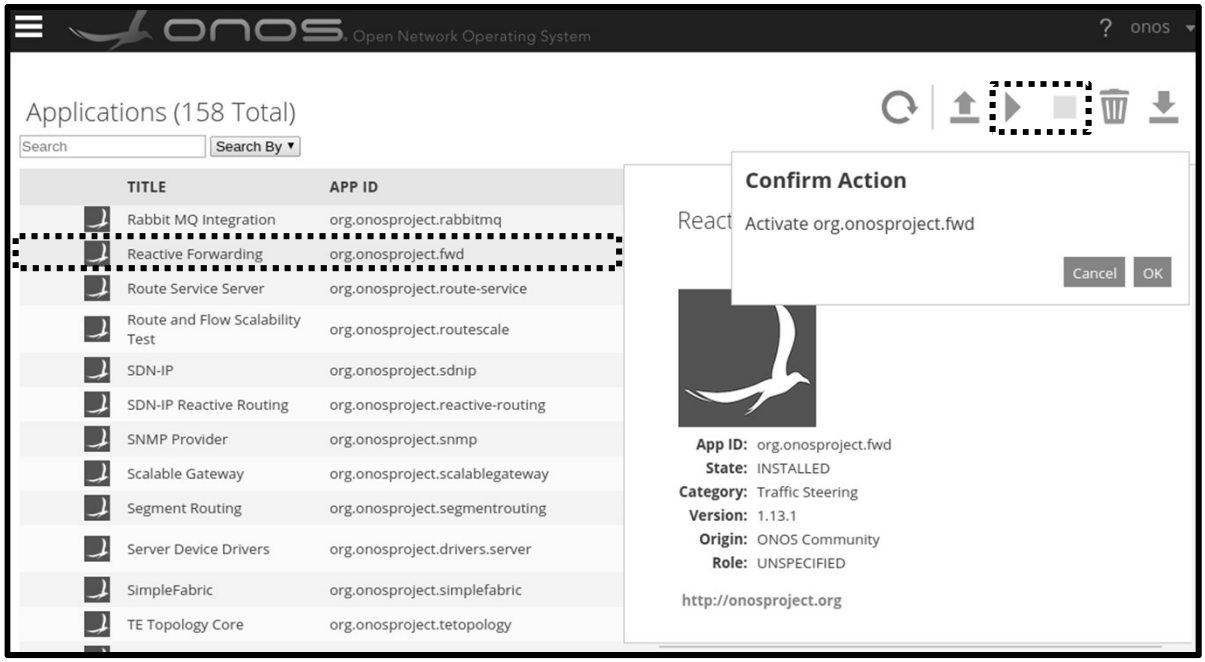
- **Check ONOS Applications:** onos> apps -a -s
- **Active the Reactive Forwarding app:** onos> app activate org.onosproject.fwd

james@jslab.kr

5. mininet (w/ONOS)

❖ Reactive Forwarding app stop/start 후 Ping 비교

① **stop app**
② **mininet> h11 ping -c3 h41**
③ **start app**
④ **mininet> h11 ping -c3 h41**



메모:

- **stop app:** onos> app deactivate fwd
- **start app:** onos> app activate fwd

james@jslab.kr

5. mininet (w/ONOS)

❖ ONOS 명령어 help, devices, links, hosts

① onos> help onos

② onos> devices

③ onos> links

④ onos> hosts

```
Terminal
File  Edit  View  Terminal  Tabs  Help
src=of:000000000000000e/2, dst=of:0000000000000002/4, type=DIRECT, state=ACTIVE, expected=false
onos> hosts^C
onos> devices
id=of:0000000000000001, available=true, local-status=connected 6m21s ago, role=STANDBY, type=SWITCH, mfr=Nicira,
 Inc., hw=Open vSwitch, sw=2.5.2, serial=None, chassis=1, driver=ovs, channelId=172.17.0.1:45410, locType=none,
managementAddress=172.17.0.1, name=Spine-1, protocol=OF_13
id=of:0000000000000002, available=true, local-status=connected 6m21s ago, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.5.2, serial=None, chassis=2, driver=ovs, channelId=172.17.0.1:41176, locType=none, m
anagementAddress=172.17.0.1, name=Spine-2, protocol=OF_13
id=of:000000000000000b, available=true, local-status=connected 6m21s ago, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.5.2, serial=None, chassis=b, driver=ovs, channelId=172.17.0.1:41150, locType=none, m
anagementAddress=172.17.0.1, name=Leaf-1, protocol=OF_13
id=of:000000000000000c, available=true, local-status=connected 6m21s ago, role=STANDBY, type=SWITCH, mfr=Nicira,
 Inc., hw=Open vSwitch, sw=2.5.2, serial=None, chassis=c, driver=ovs, channelId=172.17.0.1:41182, locType=none,
managementAddress=172.17.0.1, name=Leaf-2, protocol=OF_13
id=of:000000000000000d, available=true, local-status=connected 6m21s ago, role=STANDBY, type=SWITCH, mfr=Nicira,
 Inc., hw=Open vSwitch, sw=2.5.2, serial=None, chassis=d, driver=ovs, channelId=172.17.0.1:45420, locType=none,
managementAddress=172.17.0.1, name=Leaf-3, protocol=OF_13
id=of:000000000000000e, available=true, local-status=connected 6m21s ago, role=STANDBY, type=SWITCH, mfr=Nicira,
 Inc., hw=Open vSwitch, sw=2.5.2, serial=None, chassis=e, driver=ovs, channelId=172.17.0.1:45418, locType=none,
managementAddress=172.17.0.1, name=Leaf-4, protocol=OF_13
onos> links
src=of:0000000000000001/1, dst=of:000000000000000b/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000001/2, dst=of:000000000000000c/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000001/3, dst=of:000000000000000d/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000001/4, dst=of:000000000000000e/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000002/1, dst=of:000000000000000b/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000002/2, dst=of:000000000000000c/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000002/3, dst=of:000000000000000d/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000002/4, dst=of:000000000000000e/2, type=DIRECT, state=ACTIVE, expected=false
src=of:000000000000000b/1, dst=of:0000000000000001/1, type=DIRECT, state=ACTIVE, expected=false
src=of:000000000000000b/2, dst=of:0000000000000002/1, type=DIRECT, state=ACTIVE, expected=false
src=of:000000000000000c/1, dst=of:0000000000000001/2, type=DIRECT, state=ACTIVE, expected=false
src=of:000000000000000c/2, dst=of:0000000000000002/2, type=DIRECT, state=ACTIVE, expected=false
src=of:000000000000000d/1, dst=of:0000000000000001/3, type=DIRECT, state=ACTIVE, expected=false
src=of:000000000000000d/2, dst=of:0000000000000002/3, type=DIRECT, state=ACTIVE, expected=false
src=of:000000000000000e/1, dst=of:0000000000000001/4, type=DIRECT, state=ACTIVE, expected=false
src=of:000000000000000e/2, dst=of:0000000000000002/4, type=DIRECT, state=ACTIVE, expected=false
onos> hosts
```

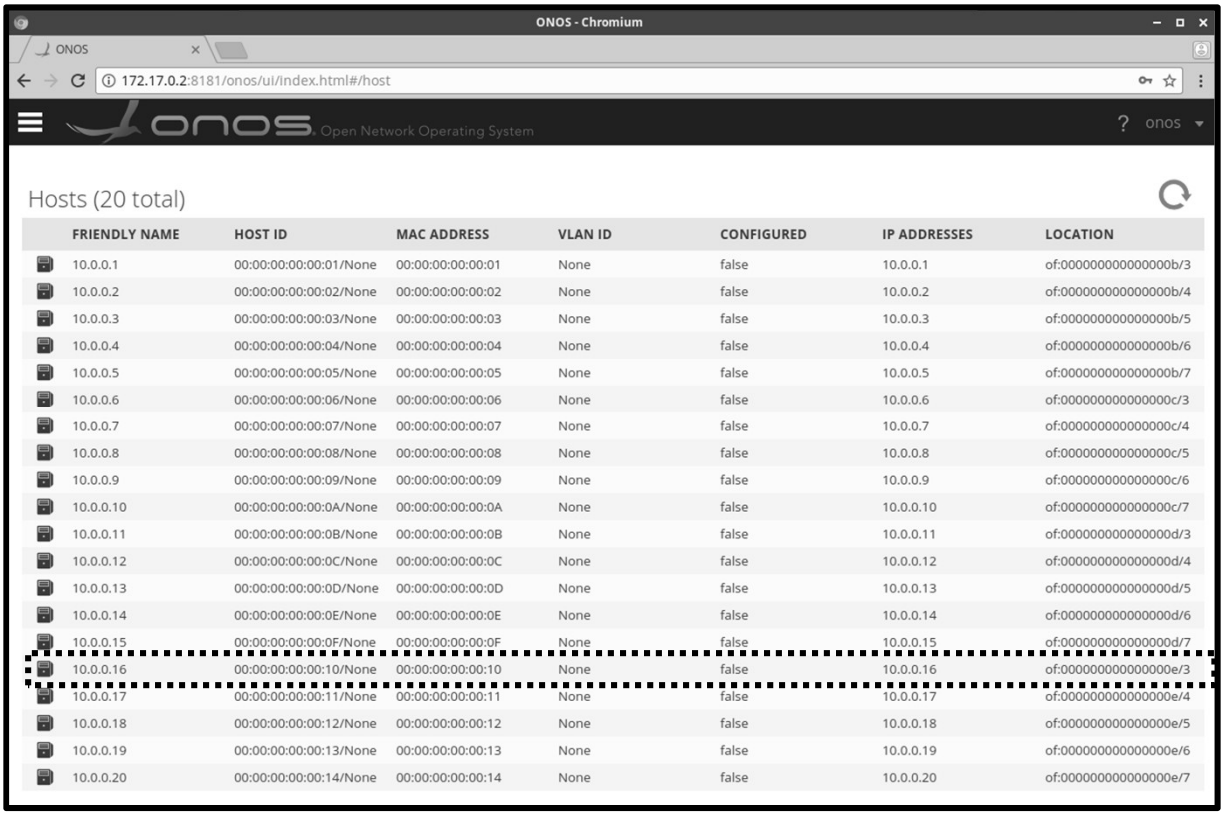
메모:

5. mininet (w/ONOS)

❖ ONOS 명령어 help, devices, links, hosts

① **onos> hosts**

② **Check Host ID '00:00:00:00:00:10/None'**



FRIENDLY NAME	HOST ID	MAC ADDRESS	VLAN ID	CONFIGURED	IP ADDRESSES	LOCATION
10.0.0.1	00:00:00:00:00:01/None	00:00:00:00:00:01	None	false	10.0.0.1	of:000000000000000b/3
10.0.0.2	00:00:00:00:00:02/None	00:00:00:00:00:02	None	false	10.0.0.2	of:000000000000000b/4
10.0.0.3	00:00:00:00:00:03/None	00:00:00:00:00:03	None	false	10.0.0.3	of:000000000000000b/5
10.0.0.4	00:00:00:00:00:04/None	00:00:00:00:00:04	None	false	10.0.0.4	of:000000000000000b/6
10.0.0.5	00:00:00:00:00:05/None	00:00:00:00:00:05	None	false	10.0.0.5	of:000000000000000b/7
10.0.0.6	00:00:00:00:00:06/None	00:00:00:00:00:06	None	false	10.0.0.6	of:000000000000000c/3
10.0.0.7	00:00:00:00:00:07/None	00:00:00:00:00:07	None	false	10.0.0.7	of:000000000000000c/4
10.0.0.8	00:00:00:00:00:08/None	00:00:00:00:00:08	None	false	10.0.0.8	of:000000000000000c/5
10.0.0.9	00:00:00:00:00:09/None	00:00:00:00:00:09	None	false	10.0.0.9	of:000000000000000c/6
10.0.0.10	00:00:00:00:00:0A/None	00:00:00:00:00:0A	None	false	10.0.0.10	of:000000000000000c/7
10.0.0.11	00:00:00:00:00:0B/None	00:00:00:00:00:0B	None	false	10.0.0.11	of:000000000000000d/3
10.0.0.12	00:00:00:00:00:0C/None	00:00:00:00:00:0C	None	false	10.0.0.12	of:000000000000000d/4
10.0.0.13	00:00:00:00:00:0D/None	00:00:00:00:00:0D	None	false	10.0.0.13	of:000000000000000d/5
10.0.0.14	00:00:00:00:00:0E/None	00:00:00:00:00:0E	None	false	10.0.0.14	of:000000000000000d/6
10.0.0.15	00:00:00:00:00:0F/None	00:00:00:00:00:0F	None	false	10.0.0.15	of:000000000000000d/7
10.0.0.16	00:00:00:00:00:10/None	00:00:00:00:00:10	None	false	10.0.0.16	of:000000000000000e/3
10.0.0.17	00:00:00:00:00:11/None	00:00:00:00:00:11	None	false	10.0.0.17	of:000000000000000e/4
10.0.0.18	00:00:00:00:00:12/None	00:00:00:00:00:12	None	false	10.0.0.18	of:000000000000000e/5
10.0.0.19	00:00:00:00:00:13/None	00:00:00:00:00:13	None	false	10.0.0.19	of:000000000000000e/6
10.0.0.20	00:00:00:00:00:14/None	00:00:00:00:00:14	None	false	10.0.0.20	of:000000000000000e/7

메모:

5. mininet (w/ONOS)

❖ Flow 명령어

① **onos> flows**

② **mininet> h11 ping h41**

③ **onos> flows**

④ **paths <TAB>**

```
Terminal
File  Edit  View  Terminal  Tabs  Help
    id=10000fd22f071, state=ADDED, bytes=196, packets=2, duration=623, liveType=UNKNOWN, priority=5, tableId=0,
appId=org.onosproject.core, payLoad=null, selector=[ETH_TYPE:ipv4], treatment=DefaultTrafficTreatment{immediate=
[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
onos> flows
deviceId=of:0000000000000001, flowRuleCount=4
    id=100007a585b6f, state=ADDED, bytes=167986, packets=1846, duration=1428, liveType=UNKNOWN, priority=40000,
tableId=0, appId=org.onosproject.core, payLoad=null, selector=[ETH_TYPE:bddp], treatment=DefaultTrafficTreatment
{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=null, metadata
=null}
    id=100009465555a, state=ADDED, bytes=167986, packets=1846, duration=1428, liveType=UNKNOWN, priority=40000,
tableId=0, appId=org.onosproject.core, payLoad=null, selector=[ETH_TYPE:lldp], treatment=DefaultTrafficTreatment
{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=null, metadata
=null}
    id=10000ea6f4b8e, state=ADDED, bytes=0, packets=0, duration=1428, liveType=UNKNOWN, priority=40000, tableId=
0, appId=org.onosproject.core, payLoad=null, selector=[ETH_TYPE:arp], treatment=DefaultTrafficTreatment{immediat
e=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=10000021b41dc, state=ADDED, bytes=0, packets=0, duration=643, liveType=UNKNOWN, priority=5, tableId=0, ap
pId=org.onosproject.core, payLoad=null, selector=[ETH_TYPE:ipv4], treatment=DefaultTrafficTreatment{immediate=[O
UTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
deviceId=of:0000000000000002, flowRuleCount=4
    id=1000002bbd8d4, state=ADDED, bytes=167895, packets=1845, duration=1428, liveType=UNKNOWN, priority=40000,
tableId=0, appId=org.onosproject.core, payLoad=null, selector=[ETH_TYPE:lldp], treatment=DefaultTrafficTreatment
{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=null, metadata
=null}
```

```
onos> paths
Error executing command onos.paths: argument src is required
onos> paths of:000000000000000
of:0000000000000001   of:0000000000000002   of:000000000000000b   of:000000000000000c   of:000000000000000d
of:000000000000000e
onos> paths of:0000000000000001 of:000000000000000
of:0000000000000001   of:0000000000000002   of:000000000000000b   of:000000000000000c   of:000000000000000d
of:000000000000000e
onos> paths of:0000000000000001 of:0000000000000002
of:0000000000000001/4-of:000000000000000e/1==>of:000000000000000e/2-of:0000000000000002/4; cost=2.0
of:0000000000000001/3-of:000000000000000d/1==>of:000000000000000d/2-of:0000000000000002/3; cost=2.0
of:0000000000000001/2-of:000000000000000c/1==>of:000000000000000c/2-of:0000000000000002/2; cost=2.0
of:0000000000000001/1-of:000000000000000b/1==>of:000000000000000b/2-of:0000000000000002/1; cost=2.0
onos>
onos>
onos>
```

메모:

- **PENDING_ADD:** The flow has been submitted and forwarded to the switch.
- **ADDED:** The flow has been added to the switch.
- **PENDING_REMOVE**: The request to remove the flow has been submitted and forwarded to the switch.
- **REMOVED:** The rule has been removed.

5. mininet (w/ONOS)

❖ Intent 명령어 (1)

① **onos> app deactivate fwd**

② **onos> add-host-intent 00:00:00:00:00:01/None 00:00:00:00:00:02/None**

③ **mininet> h11 ping h41** # Ping 실패

④ **mininet> h11 ping h12** # Ping 성공

```
onos> add-host-intent 00:00:00:00:00:
00:00:00:00:00:01/None   00:00:00:00:00:02/None   00:00:00:00:00:03/None   00:00:00:00:00:04/None
00:00:00:00:00:05/None   00:00:00:00:00:06/None   00:00:00:00:00:07/None   00:00:00:00:00:08/None
00:00:00:00:00:09/None   00:00:00:00:00:0A/None   00:00:00:00:00:0B/None   00:00:00:00:00:0C/None
00:00:00:00:00:0D/None   00:00:00:00:00:0E/None   00:00:00:00:00:0F/None   00:00:00:00:00:10/None
00:00:00:00:00:11/None   00:00:00:00:00:12/None   00:00:00:00:00:13/None   00:00:00:00:00:14/None
onos> add-host-intent 00:00:00:00:00:01/none 00:00:00:00:00:02/none
Error executing command: java.lang.NumberFormatException: For input string: "none"
onos> add-host-intent 00:00:00:00:00:01/None 00:00:00:00:00:02/None
Host to Host intent submitted:
HostToHostIntent{id=0x0, key=0x0, appId=DefaultApplicationId{id=26, name=org.onosproject.cli}, priority=100, res
ources=[00:00:00:00:00:01/None, 00:00:00:00:00:02/None], selector=DefaultTrafficSelector{criteria=[]}, treatment
=DefaultTrafficTreatment{immediate=[NOACTION], deferred=[], transition=None, meter=[], cleared=false, StatTrigge
r=null, metadata=null}, constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}], resourceGroup=null,
one=00:00:00:00:00:01/None, two=00:00:00:00:00:02/None}
onos> intents
Id: 0x0
State: INSTALLED
Key: 0x0
Intent type: HostToHostIntent
Application Id: org.onosproject.cli
Leader Id: 172.17.0.4
Resources: [00:00:00:00:00:01/None, 00:00:00:00:00:02/None]
Treatment: [NOACTION]
Constraints: [LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
Source host: 00:00:00:00:00:01/None
Destination host: 00:00:00:00:00:02/None

onos>
```

- WITHDRAWING - The intent is being withdrawn.
- WITHDRAWN - The intent has been removed.
- FAILED - The intent is in a failed state because it cannot be satisfied.

메모:

- **SUBMITTED:** The intent has been submitted and will be processed soon.
- **COMPILING:** The intent is being compiled. This is a transient state.
- **INSTALLING:** The intent is in the process of being installed.
- **INSTALLED:** The intent has been installed.
- **RECOMPILING:** The intent is being recompiled after a failure.

james@jslab.kr

5. mininet (w/ONOS)

❖ Intent 명령어 (2)

① **onos> app deactivate fwd**

② **onos> add-host-intent 00:00:00:00:00:01/None 00:00:00:00:00:10/None**

③ **mininet> h11 ping h41**

④ **mininet> link s2 s11 down**

```
onos> add-host-intent 00:00:00:00:00:
00:00:00:00:00:01/None   00:00:00:00:00:02/None   00:00:00:00:00:03/None
00:00:00:00:00:04/None   00:00:00:00:00:05/None   00:00:00:00:00:06/None
00:00:00:00:00:07/None   00:00:00:00:00:08/None   00:00:00:00:00:09/None
00:00:00:00:00:0A/None   00:00:00:00:00:0B/None   00:00:00:00:00:0C/None
00:00:00:00:00:0D/None   00:00:00:00:00:0E/None   00:00:00:00:00:0F/None
00:00:00:00:00:10/None   00:00:00:00:00:11/None   00:00:00:00:00:12/None
00:00:00:00:00:13/None   00:00:00:00:00:14/None
onos> add-host-intent 00:00:00:00:00:01/None 00:00:00:00:00:
00:00:00:00:00:01/None   00:00:00:00:00:02/None   00:00:00:00:00:03/None
00:00:00:00:00:04/None   00:00:00:00:00:05/None   00:00:00:00:00:06/None
00:00:00:00:00:07/None   00:00:00:00:00:08/None   00:00:00:00:00:09/None
00:00:00:00:00:0A/None   00:00:00:00:00:0B/None   00:00:00:00:00:0C/None
00:00:00:00:00:0D/None   00:00:00:00:00:0E/None   00:00:00:00:00:0F/None
00:00:00:00:00:10/None   00:00:00:00:00:11/None   00:00:00:00:00:12/None
00:00:00:00:00:13/None   00:00:00:00:00:14/None
onos> add-host-intent 00:00:00:00:00:01/None 00:00:00:00:00:10/None
Host to Host intent submitted:
HostToHostIntent{id=0x0, key=0x0, appId=DefaultApplicationId{id=2, name=org.onos
project.cli}, priority=100, resources=[00:00:00:00:00:01/None, 00:00:00:00:00:10
/None], selector=DefaultTrafficSelector{criteria=[]}, treatment=DefaultTrafficTr
eatment{immediate=[NOACTION], deferred=[], transition=None, meter=[], cleared=fa
lse, StatTrigger=null, metadata=null}, constraints=[LinkTypeConstraint{inclusive
=false, types=[OPTICAL]}], resourceGroup=null, one=00:00:00:00:00:01/None, two=0
0:00:00:00:00:10/None}
onos> intents
Id: 0x0
State: INSTALLED
Key: 0x0
Intent type: HostToHostIntent
Application Id: org.onosproject.cli
Leader Id: 172.17.0.2
Resources: [00:00:00:00:00:01/None, 00:00:00:00:00:10/None]
Treatment: [NOACTION]
Constraints: [LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
Source host: 00:00:00:00:00:01/None
Destination host: 00:00:00:00:00:10/None

onos>
```

메모:

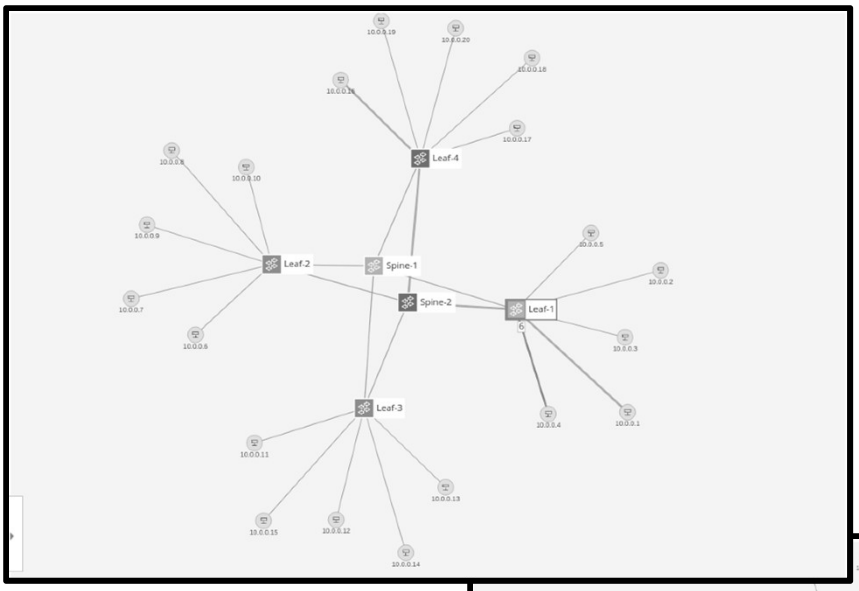
- We can see that the traffic flows between dpid 00:00:00:00:00:00:00:02 (Spine-2) and 00:00:00:00:00:00:00:0b (Leaf-1) and similarly between 00:00:00:00:00:00:00:02 (Spine-2) and 00:00:00:00:00:00:00:0e (Leaf-4).

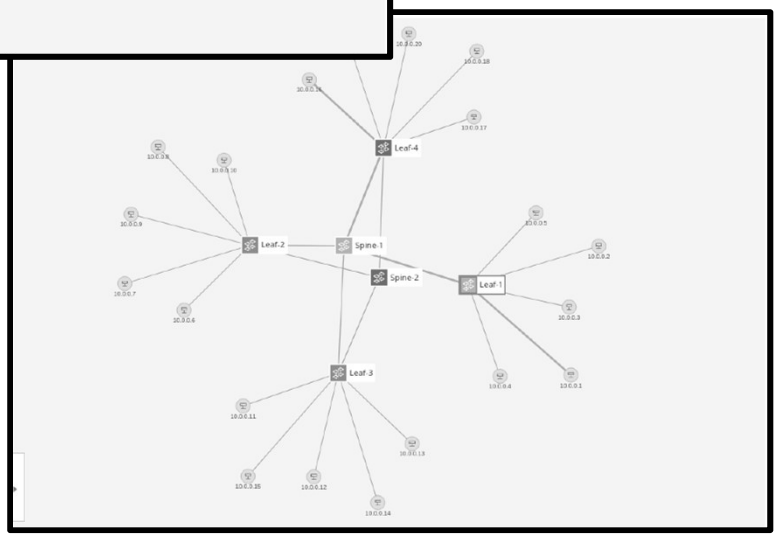
5. mininet (w/ONOS)

❖ Intent 명령어 (3)

① **mininet> h11 ping h41**

② **mininet> link s2 s11 down**





메모:

- **UI Autolayout:** onos> topo-layout access
- You can visualize the intent using ONOS GUI by selecting the Leaf-1 node in the GUI and press the ***V*** key to show paths provisioned by intents that pass thru the selected node
- To send a more significant amount of traffic, than a mere ICMP ping, between the 2 hosts you can use the bgPerf mininet command as follows: mininet> bgIperf h11 h41

james@jslab.kr

5. mininet (w/ONOS)

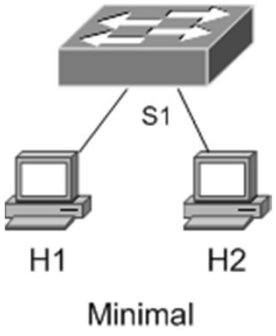
❖ Mininet 명령어와 구성 (예)

① sudo mn --topo minimal

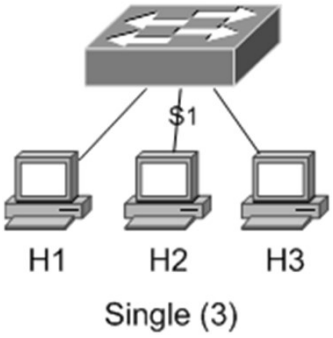
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0

② sudo mn --topo single,3

mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0



xterm h1 h2



xterm h1 h2 h3

메모:

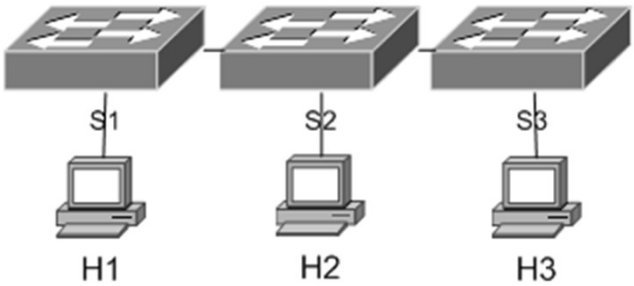
- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip

james@jslab.kr

❖ Mininet 명령어와 구성 (예)

① sudo mn --topo linear,3

```
mininet&gt; net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3
```



Linear (3)

xterm h1 h2 h3 @PuTTY

- ifconfig
- ping

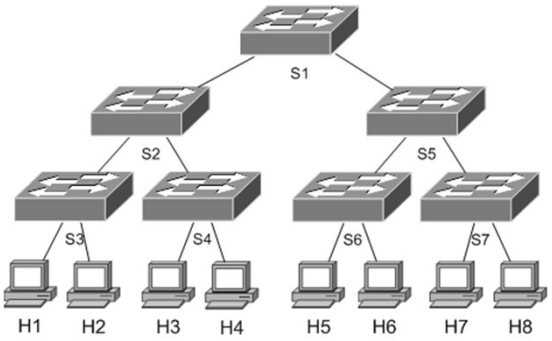
메모:

- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip

❖ Mininet 명령어와 구성 (예)

① sudo mn --topo tree,3

```
mininet&gt; net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s5-eth3
s2 lo:  s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h1-eth0 s3-eth2:h2-eth0 s3-eth3:s2-eth1
s4 lo:  s4-eth1:h3-eth0 s4-eth2:h4-eth0 s4-eth3:s2-eth2
s5 lo:  s5-eth1:s6-eth3 s5-eth2:s7-eth3 s5-eth3:s1-eth2
s6 lo:  s6-eth1:h5-eth0 s6-eth2:h6-eth0 s6-eth3:s5-eth1
s7 lo:  s7-eth1:h7-eth0 s7-eth2:h8-eth0 s7-eth3:s5-eth2
```



Tree (3)

메모:

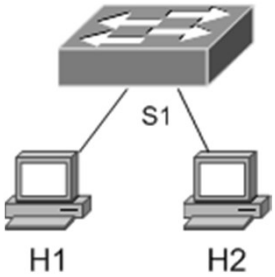
- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip

5. mininet (w/ONOS)

❖ Mininet Operations

① **sudo mn**
② **mininet> nodes**
③ **mininet> net**
④ **mininet> dump**
⑤ **mininet> xterm h1 h2 s1**

⑥ **mininet> pingall**
⑦ **mininet> link h1 s1 down**
⑧ **mininet> h1 ping -c 1 h2**
⑨ **connect: Network is unreachable**



Minimal

메모:

- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip

james@jslab.kr

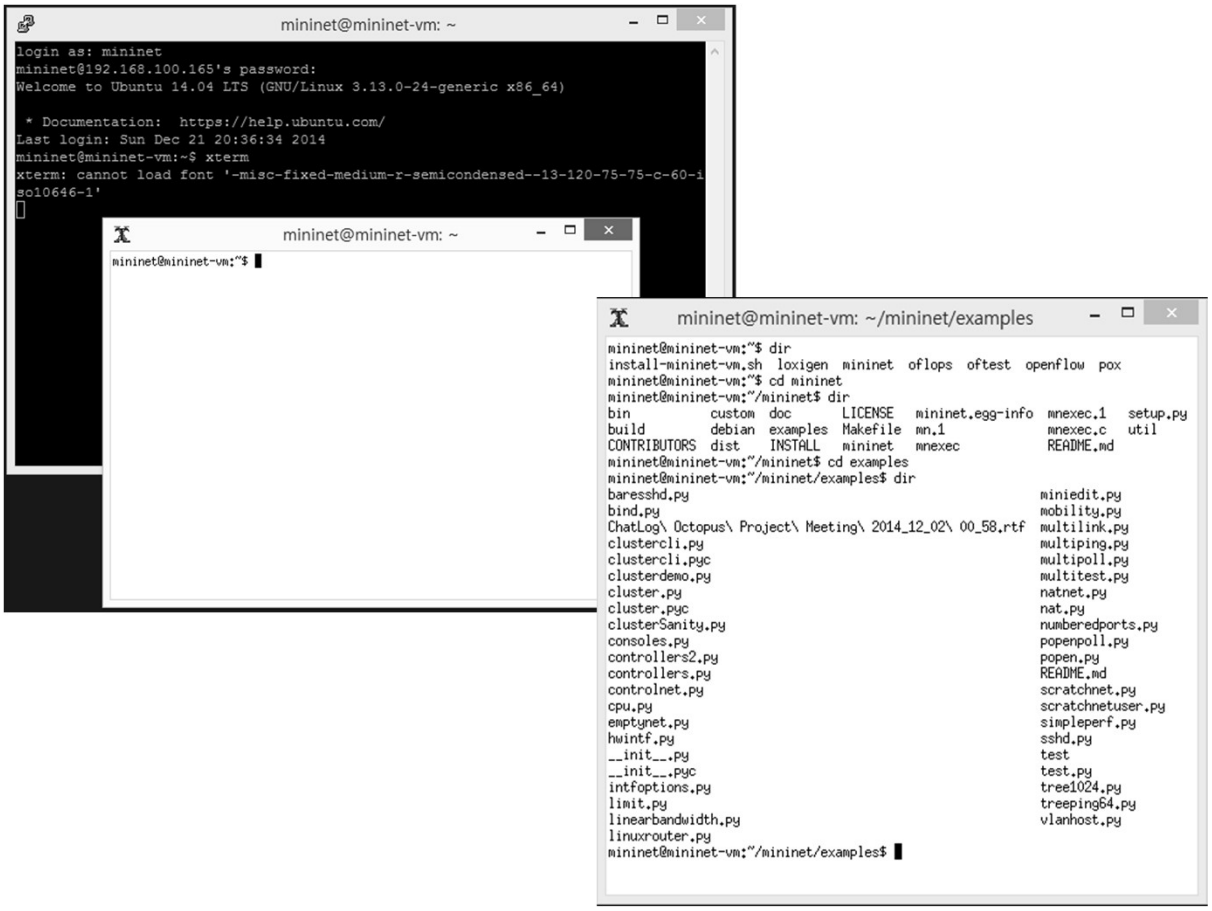
5. mininet (w/ONOS)

❖ Miniedit 실행

① **Running PuTTY w/X11**

② **'xtrem' @ Mininet (PuTTY w/X11)**

③ **'sudo ./miniedit.py' (@ mininet@mininet-vm:~mininet/examples$)**



메모:

- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip

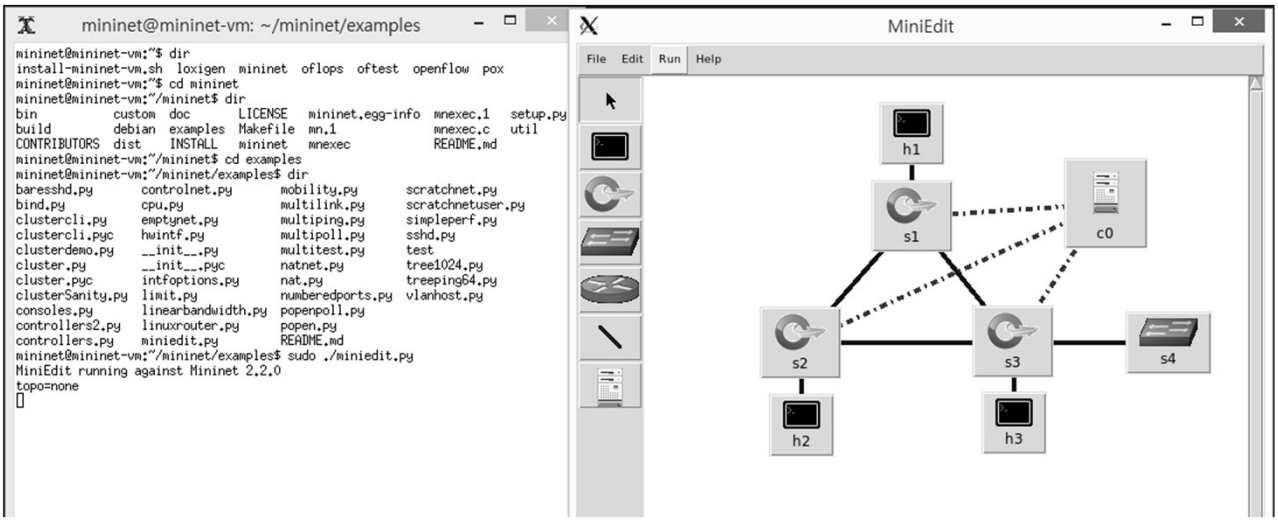
5. mininet (w/ONOS)

❖ Miniedit Operations

① **Configuration @ MiniEdit**

② **Run the Configuration @ MiniEdit**

③ **Check w/xterm**



메모:

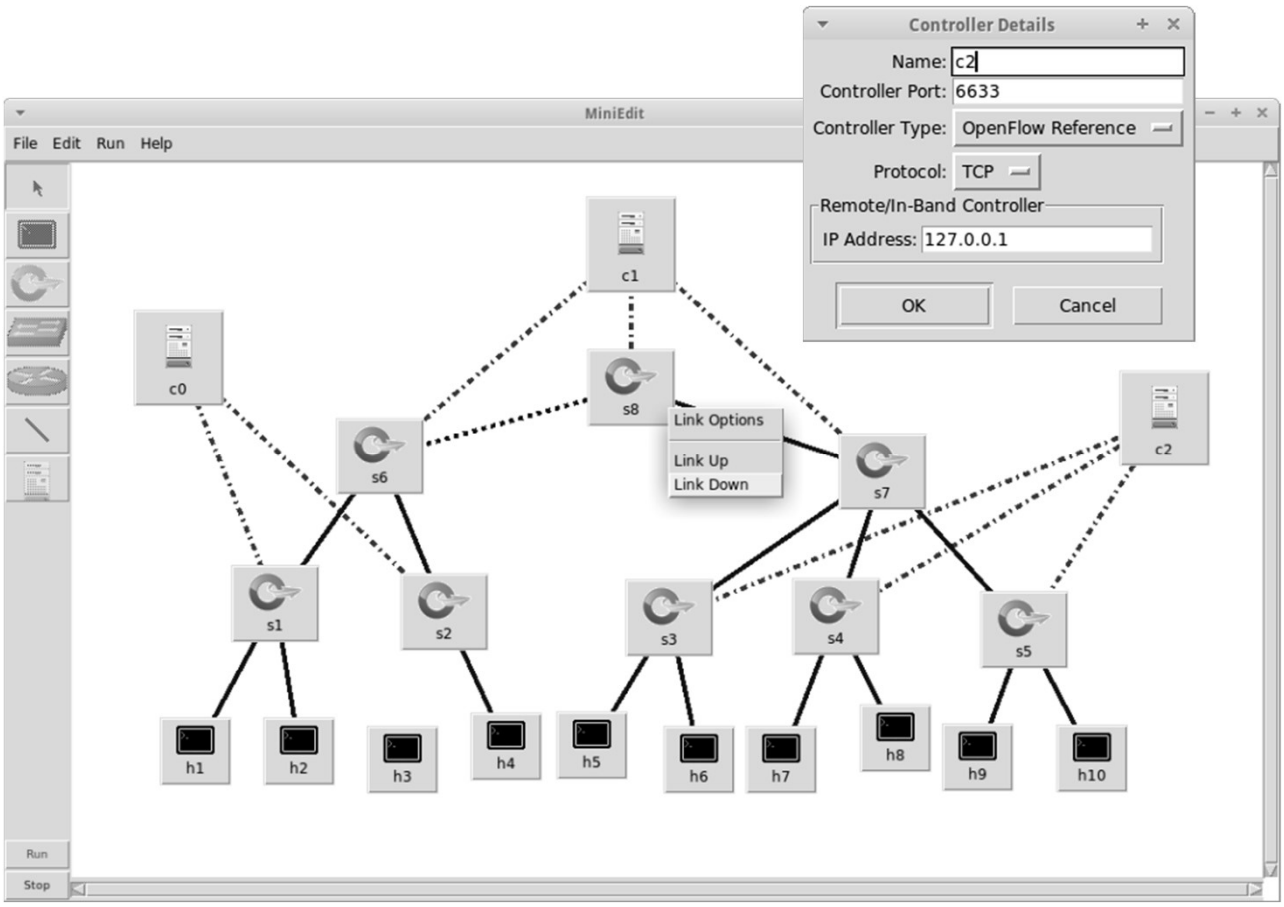
- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip

5. mininet (w/ONOS)

❖ Miniedit Operations

① **Check controller details**

② **Set IP address with remote controller option**



메모:

- SDN Controller는 VM 'UbuntuServer16.04 ONOS and Rancher with 2 ports.ova' 사용
- Ubuntu ONOS VM의 호스트 어댑터 확인 (연결 하는 2개의 VM 은 동일 어댑터 사용)
- sudo docker run -t -d -p 8181:8181 -p 8101:8101 -p 6653:6653 --name onos1 onosproject/onos
- sudo docker ps로 노출 TCP 포트 확인

james@jslab.kr

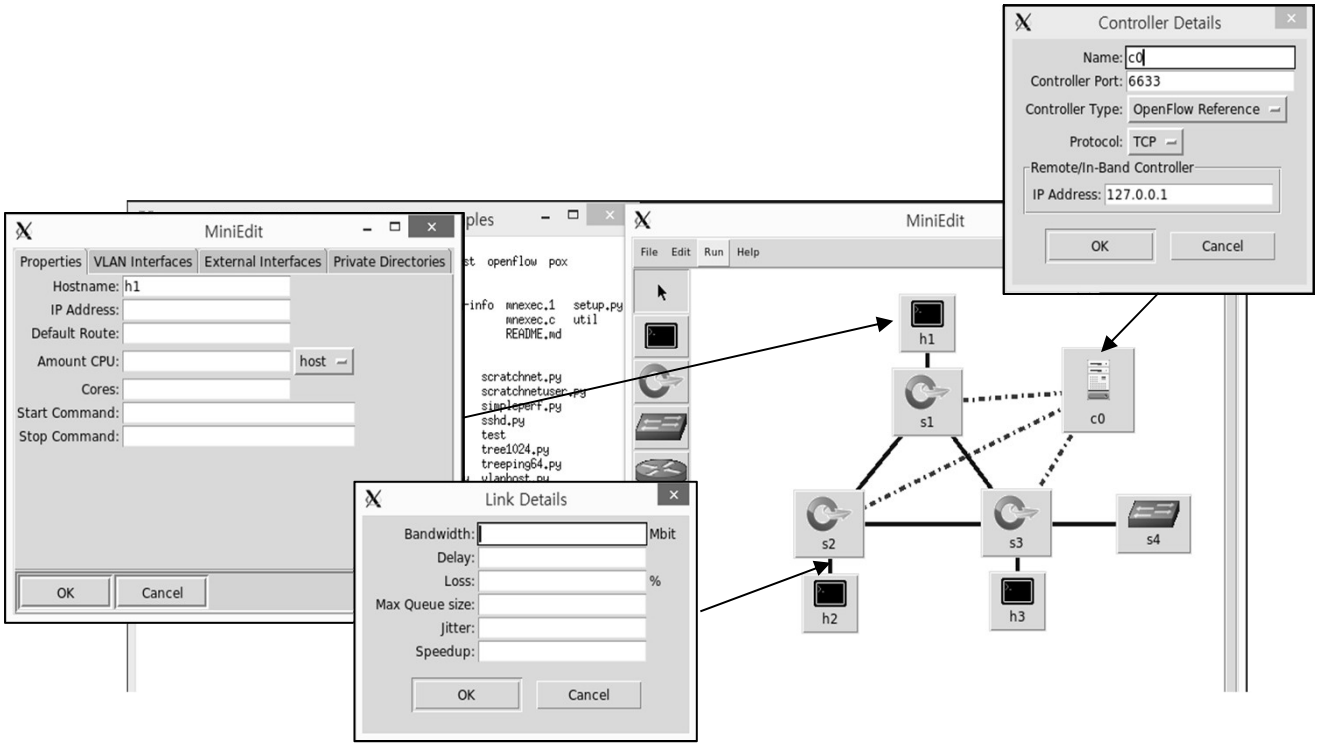
5. mininet (w/ONOS)

❖ Miniedit Operations

① **Configuration @ MiniEdit**

② **Check Options (for Remote Controller)**

③ **Save**



메모:

- http://192.168.xx.xxx:8181/onos/ui/
- ONOS 의 Applications 확인 (OpenFlow, agent, lldp, host, switch)

james@jslab.kr

5. mininet (w/ONOS)

❖ Mininet with 10X/50X/200X OpenFlow switches

① **sudo mn --topo=linear,10 --switch=ovsk,protocols=OpenFlow13 --mac**

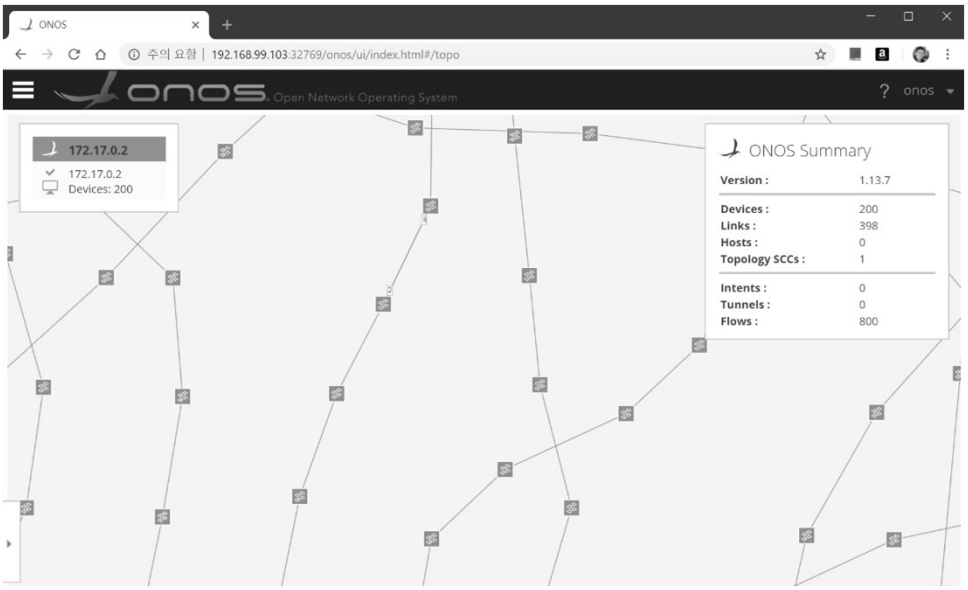
② **sudo mn --topo=linear,50 --switch=ovsk,protocols=OpenFlow13 --mac**

③ **# sudo mn --controller=remote,ip=192.168.0.211 --mac --topo=linear,200**

④ **sudo mn --controller=remote,ip=192.168.99.100:32771 --mac --topo=linear,200**

⑤ **sudo mn --topo=linear,200**

⑥ **h1 ping h200**



메모:

- Mininet Download: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip
- ### $ sudo mn --topo single,3 --mac --controller=remote, ip=x.x.x.x, port=6633
- ONOS @ Docker http://192.168.99.100:3276x/onos/ui/ w/apps

❖ OVS의 Operations @ MiniEdit

① **sudo ovs-vsctl show**
② **sudo ovs-ofctl dump-flows s1**
③ **sudo mn**
④ **sudo ovs-vsctl show**
⑤ **sudo ovs-ofctl dump-flows s1**
⑥ **h1 ping h2**
⑦ **sudo ovs-vsctl show**
⑧ **sudo ovs-ofctl dump-flows s1**

메모:

- ovs-vsctl : ovs-switchd 유틸리티 (querying and configuring)
- ovs-ofctl : OpenFlow 스위치 관리
- ovs-appctl : Open vSwitch 데몬 유틸리티 (동작중인 Open vSwitch 데몬 configuring)

1. 실습 환경
2. Host
3. Open vSwitch
4. SDN Controller (Docker)
5. mininet (w/ONOS)
6. Rancher 설치
7. Kubernetes 설치

8. 부록: Docker

6. Rancher 설치

❖ Rancher installation (Ubuntu Server 16.04)

① **docker --version** # or docker version

② **docker run -i -t -d -p 9999:8080 rancher/server**

③ **http://192.168.0.10:9999**

④ **docker ps**

⑤ **http://192.168. 0.10:8080/**

- 호스트 이름 변경 -

```
/etc/hostname
/etc/hosts
sudo nano /etc/hostname
sudo vi /etc/hosts
```

메모:

- 외부 Stateful 스토리지 사용
  - ✓ HOST_VOLUME=$HOME/rancher-data/mysql
  - ✓ mkdir -p $HOST_VOLUME
  - ✓ docker run -d -v $HOST_VOLUME:/var/lib/mysql --restart=unless-stopped -p 8080:8080 rancher/server

6. Rancher 설치

❖ Rancher installation (선택: CentOS 7 사용시)

① **yum -y install docker** # Docker version 확인 필요
② **systemctl start docker**
③ **systemctl enable docker**
④ **systemctl status docker**
⑤ **docker --version** # or docker version

⑥ **docker run -i -t -d -p 9999:8080 rancher/server**
⑦ **docker ps**
⑧ **ip addr**
⑨ **http://192.168. 56.x0:9999/** # master

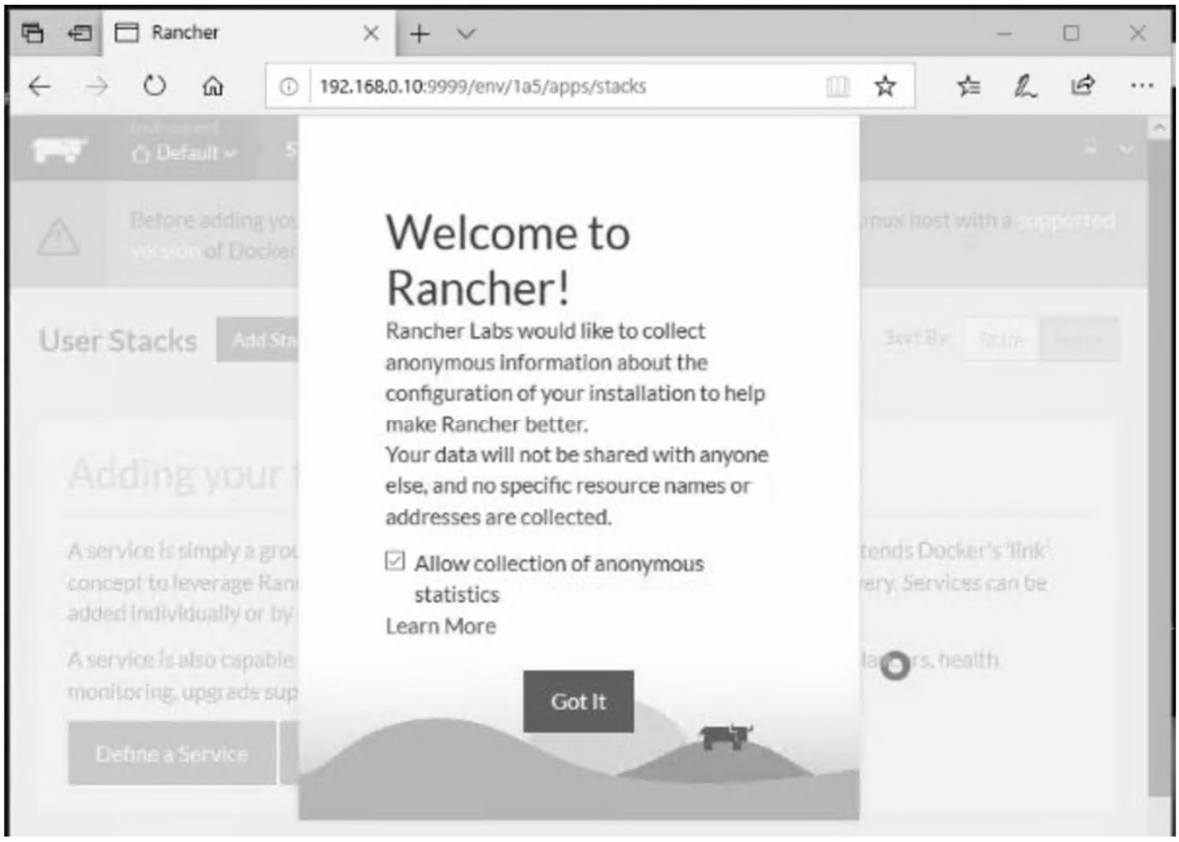
- 참조: https://www.howtoforge.com/tutorial/centos-rancher-docker-container-management-platform/
- 실습은 Container Local Storage 용 사용: docker run -i -t -d -p 9999:8080 rancher/server
- Rancher 실행 후 수분 후에 접속 가능: http://192.168.0.10:9999

메모:
- 외부 Stateful 스토리지 사용
  - ✓ HOST_VOLUME=$HOME/rancher-data/mysql
  - ✓ mkdir -p $HOST_VOLUME
  - ✓ docker run -d -v $HOST_VOLUME:/var/lib/mysql --restart=unless-stopped -p 8080:8080 rancher/server

6. Rancher 설치

❖ Rancher installation

① **http://192.168. xx.xx:9999/**



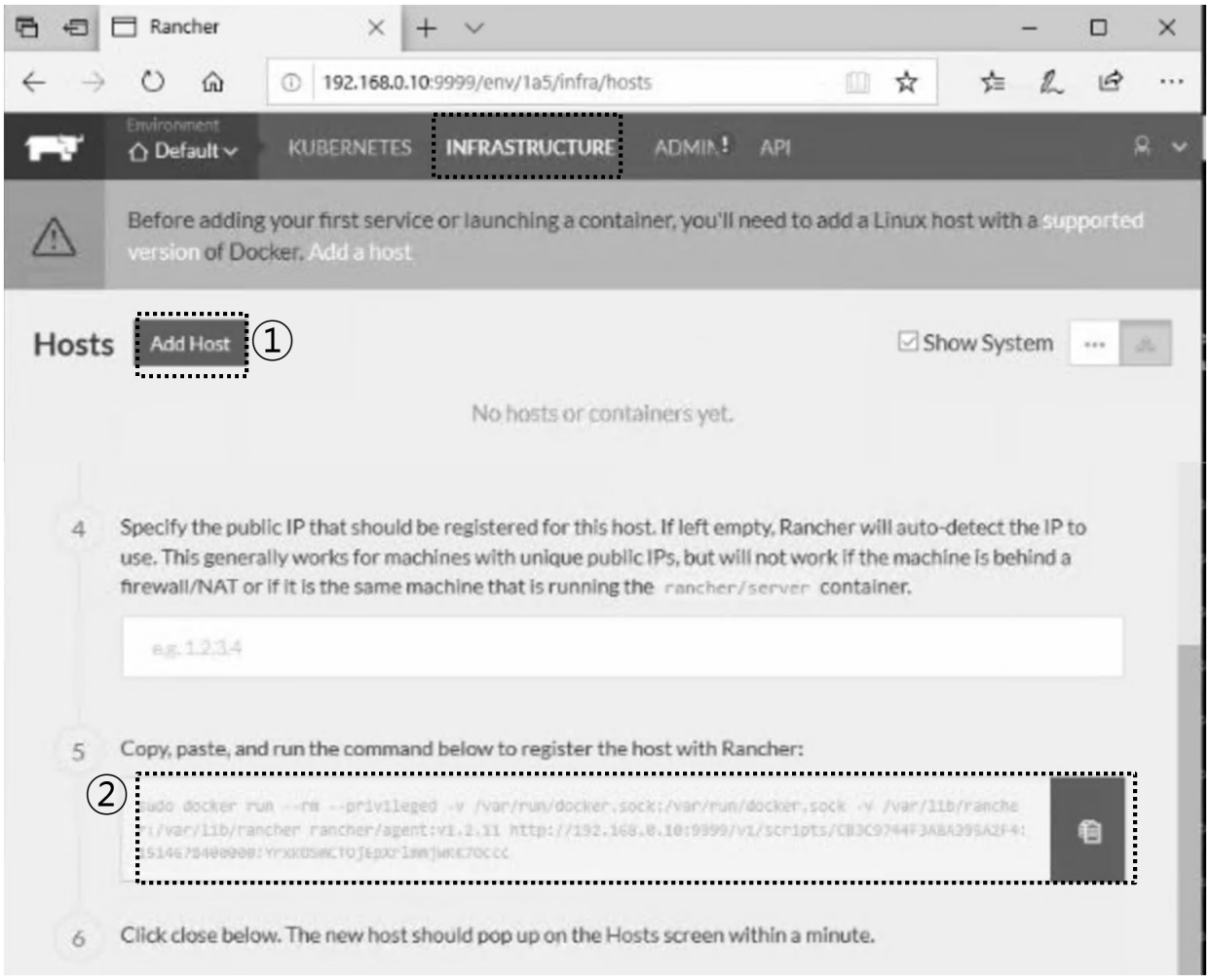
메모:

6. Rancher 설치

❖ Rancher installation

① **Add host @ Infrastructure**

② **Copy key for Paste @ worker01, worker02, worker03**



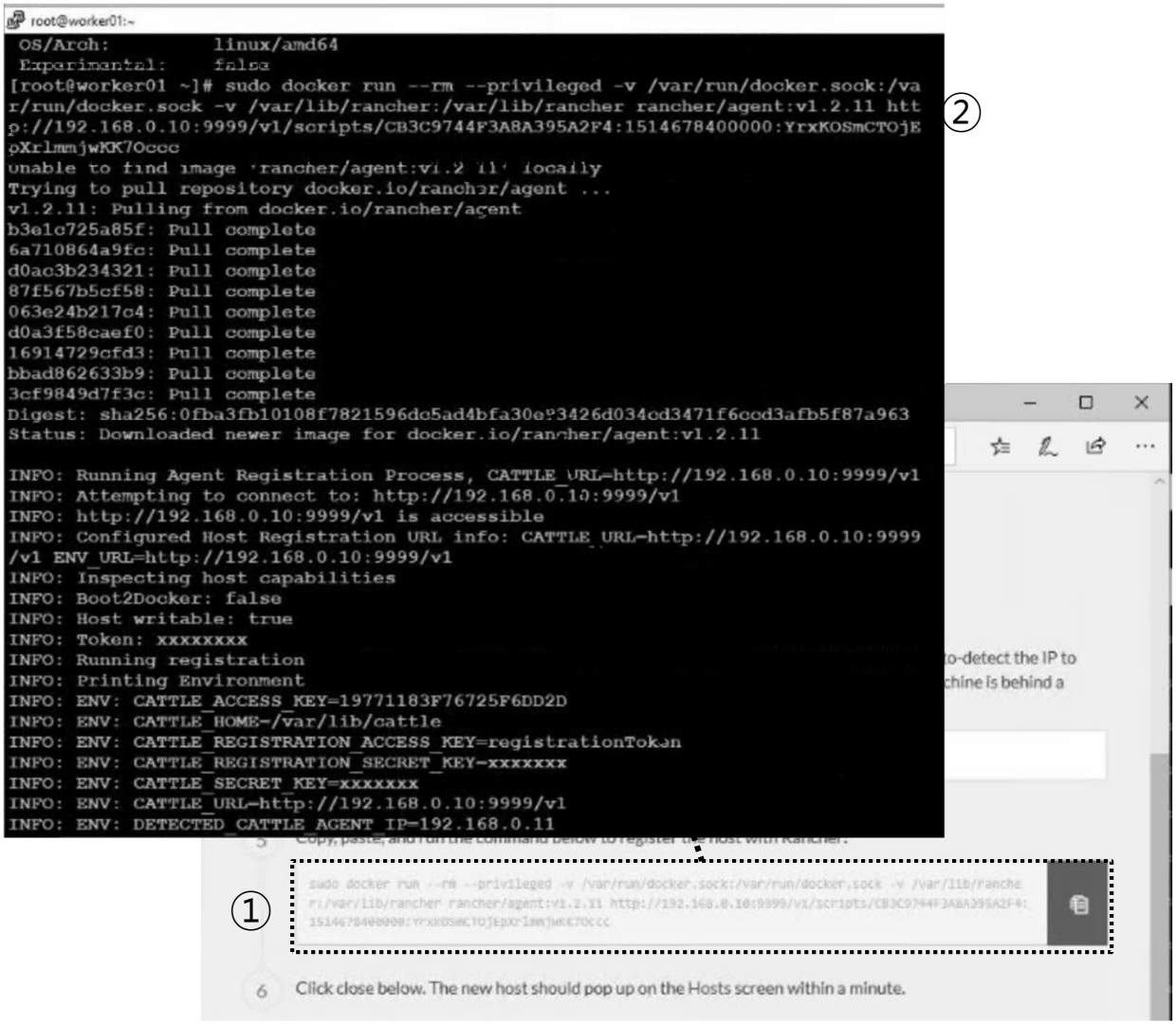
메모:

6. Rancher 설치

❖ Rancher installation

① **Copy key**

② **Paste key @ worker01, worker02, worker03**



메모:

- 아마존 클라우드 서비스 AWS 상의 Docker Hub가 연결 되어야 함

1. 실습 환경
2. Host
3. Open vSwitch
4. SDN Controller (Docker)
5. mininet (w/ONOS)
6. Rancher
7. Kubernetes 설치

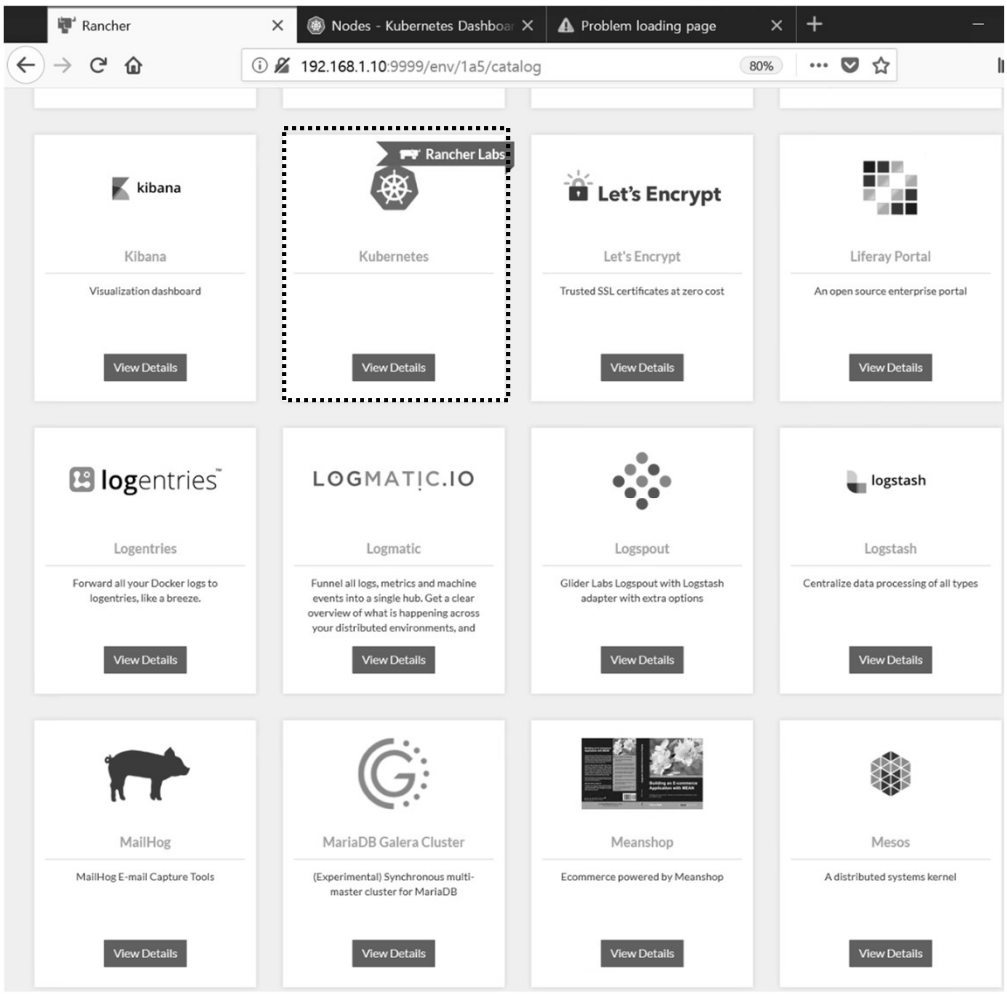
❖ 부록: Docker

7. Kubernetes 설치

❖ Rancher installation

① **Catalog @ Infrastructure**

② **Check Kubernetes**

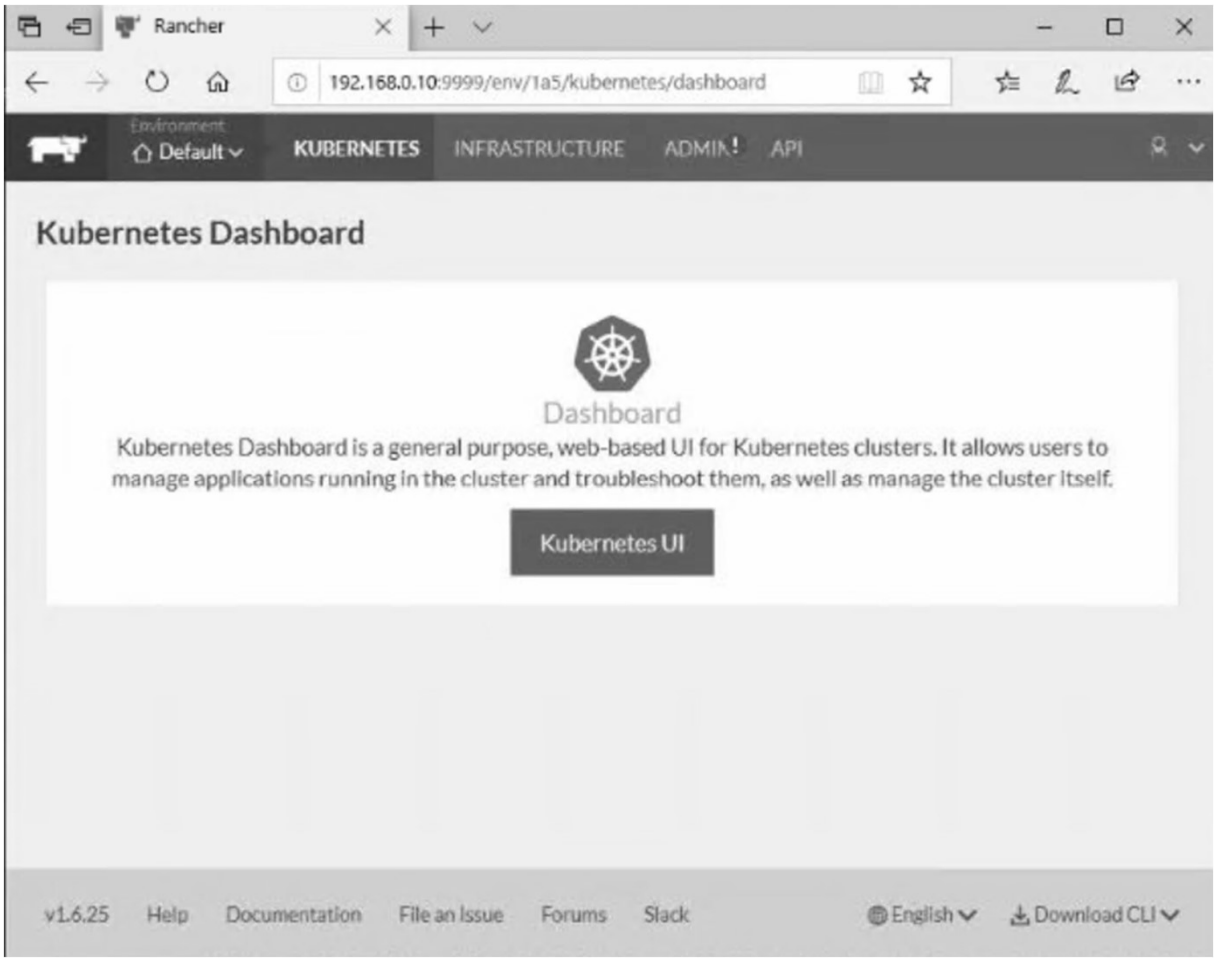


메모:

7. Kubernetes 설치

❖ K8s installation

① **Dashboard @ Kubernetes**
② **CLI @ Kubernetes**



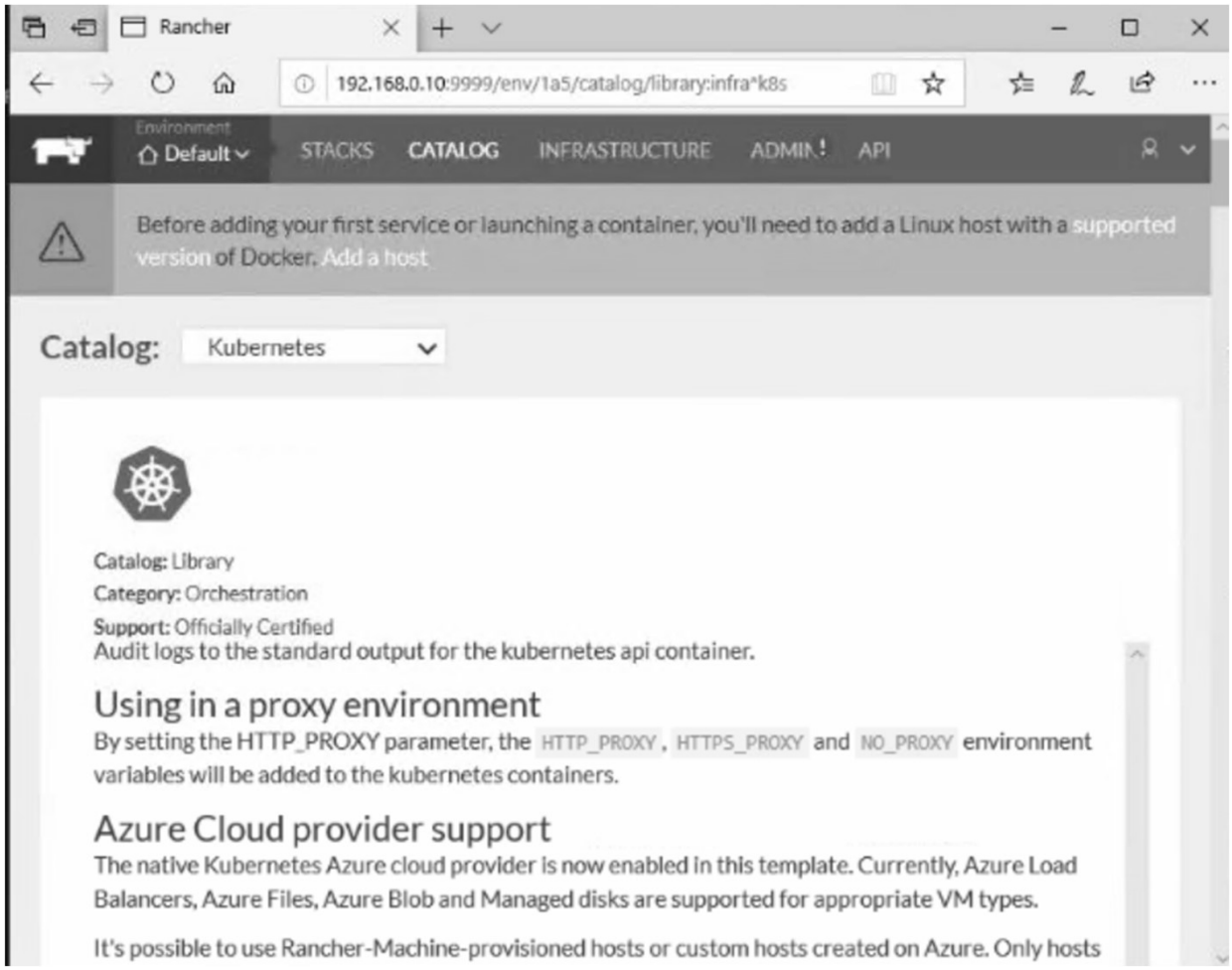
메모:

7. Kubernetes 설치

❖ K8s installation

① **http://192.168. 0.10:9999/**

② **K8s @ Catalog 선택 Launch**

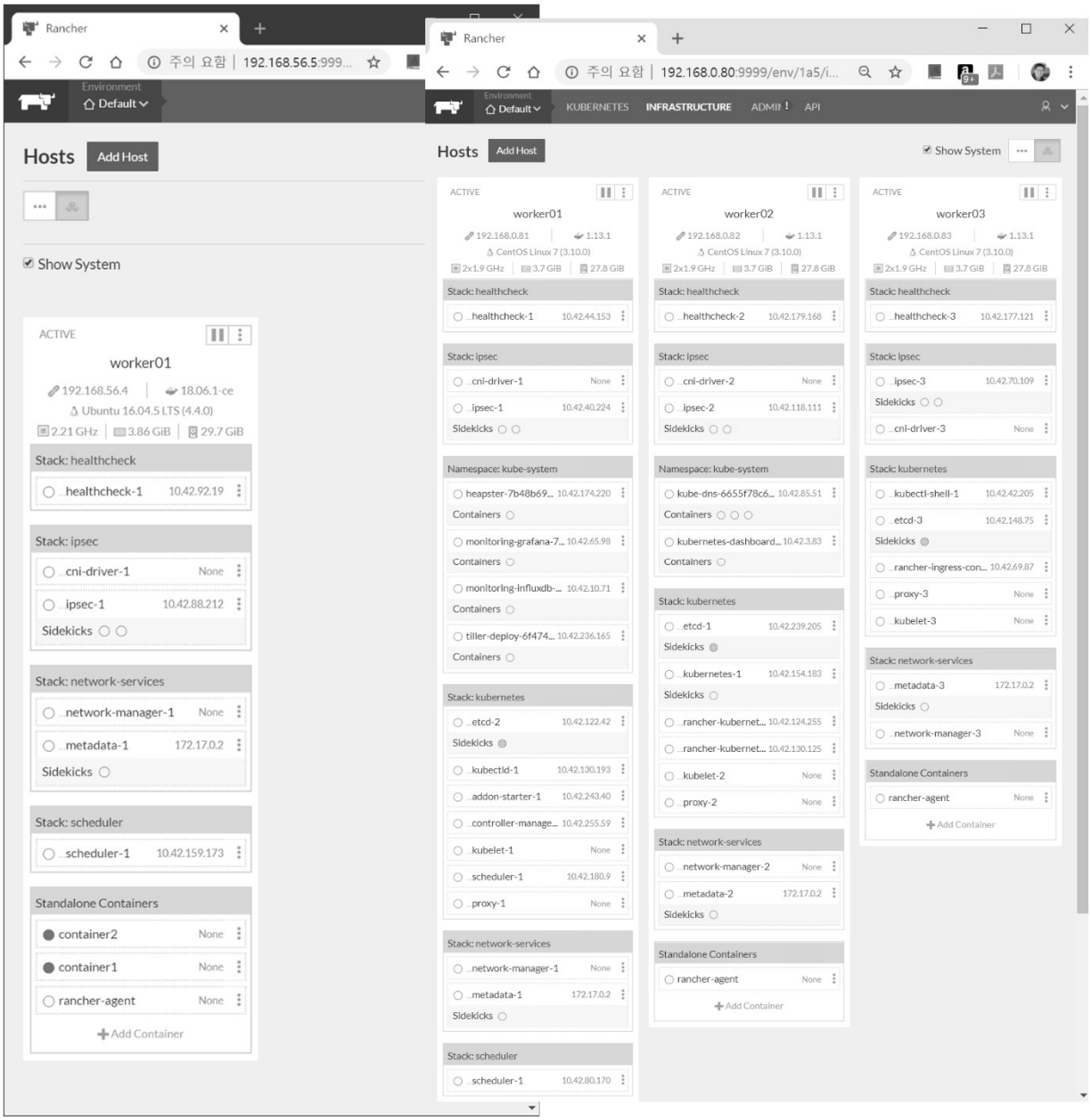


메모:

- Kubernetes 대쉬보드 접속은 K8s 실행 후 수분 소요

7. Kubernetes 설치

❖ Infrastructure/Hosts after K8s installation

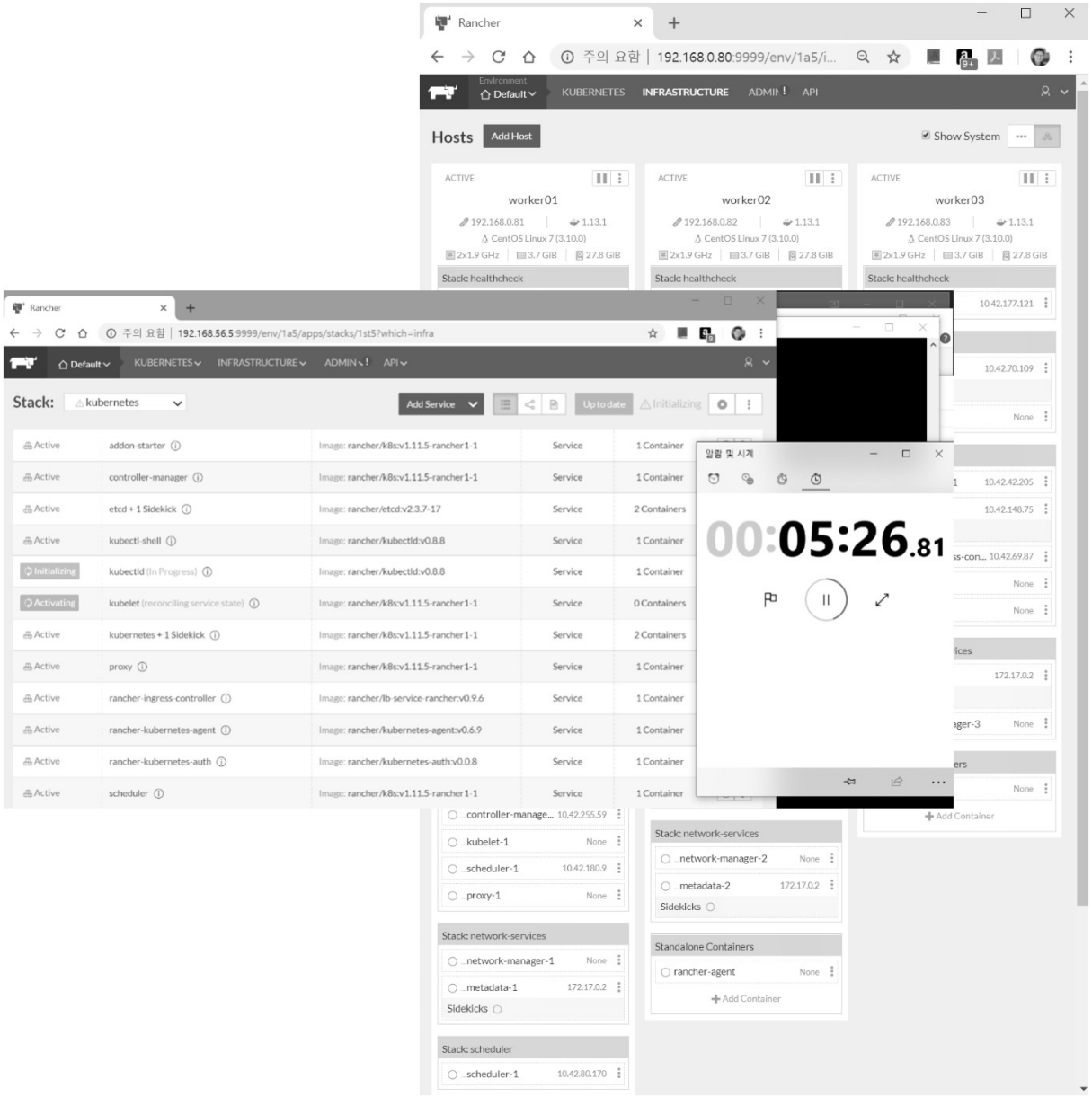


메모:

- 실습장비 RAM 16 GB 이상에서 Host (Worker node) 3개
- 실습장비 RAM 8 GB 에서 Host (Worker node) 1 개
- Host 이미지 크기 작은 것이 유리 (CentOS 7 minimal 권장)

7. Kubernetes 설치

❖ Infrastructure/Hosts K8s @ Ubuntu Server 16.04

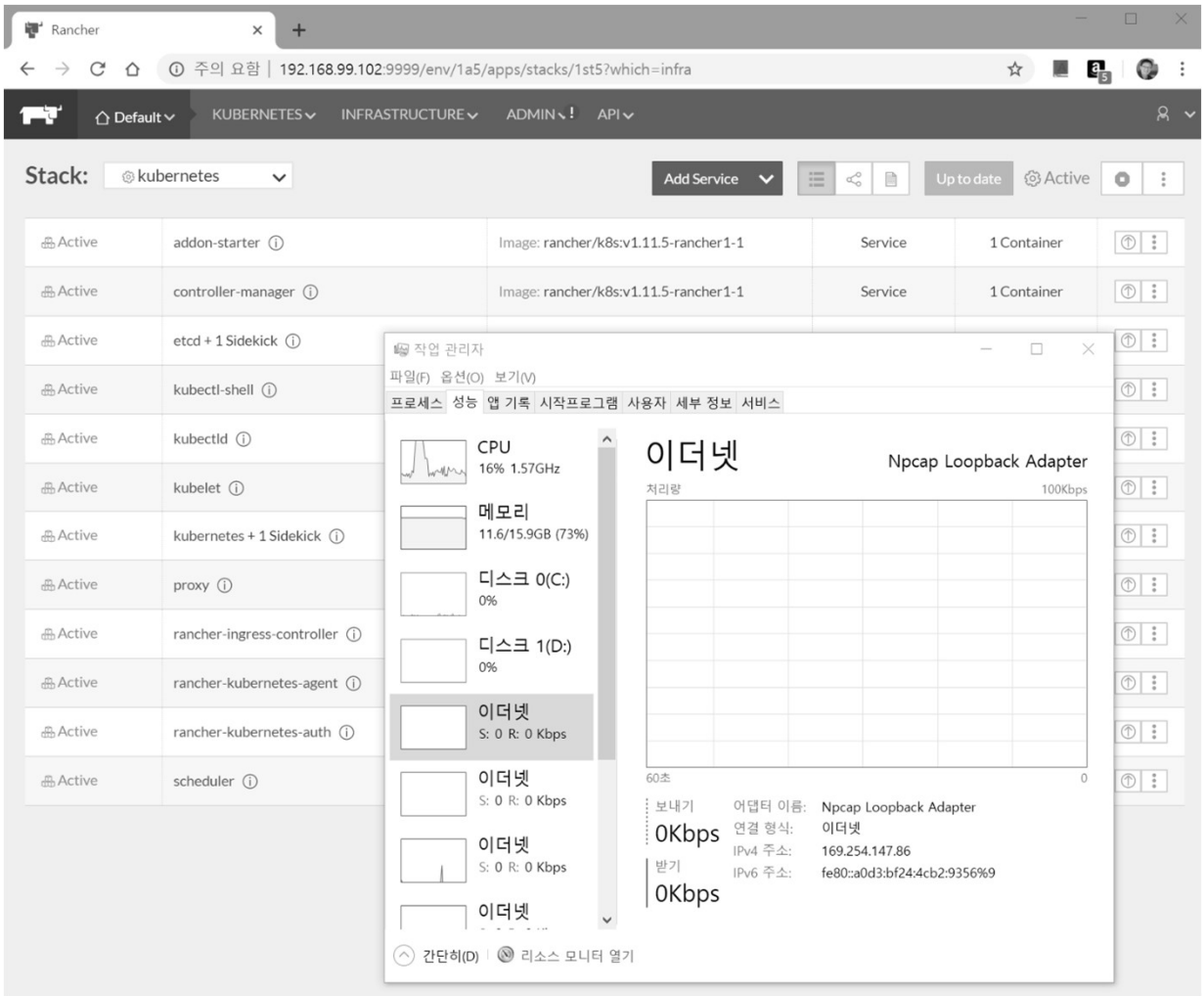


메모:

- 실습장비 RAM 8 GB 에서 Host (Worker node) 1 개
- Worker node 1개 시 RAM 4 GB 이상에 확장 권장
- Master node 는 실습에서 RAM 2 GB 가능

7. Kubernetes 설치

❖ Infrastructure/Hosts K8s @ CentOS7 minimal



메모:

- 실습장비 RAM 8 GB 에서 Host (Worker node) 1 개
- Worker node 1개 시 RAM 4 GB 이상에 확장 권장
- Master node 는 실습에서 RAM 2 GB 가능
- RAM Memory 크기가 작은 경우 Paging으로 속도 저하 가능

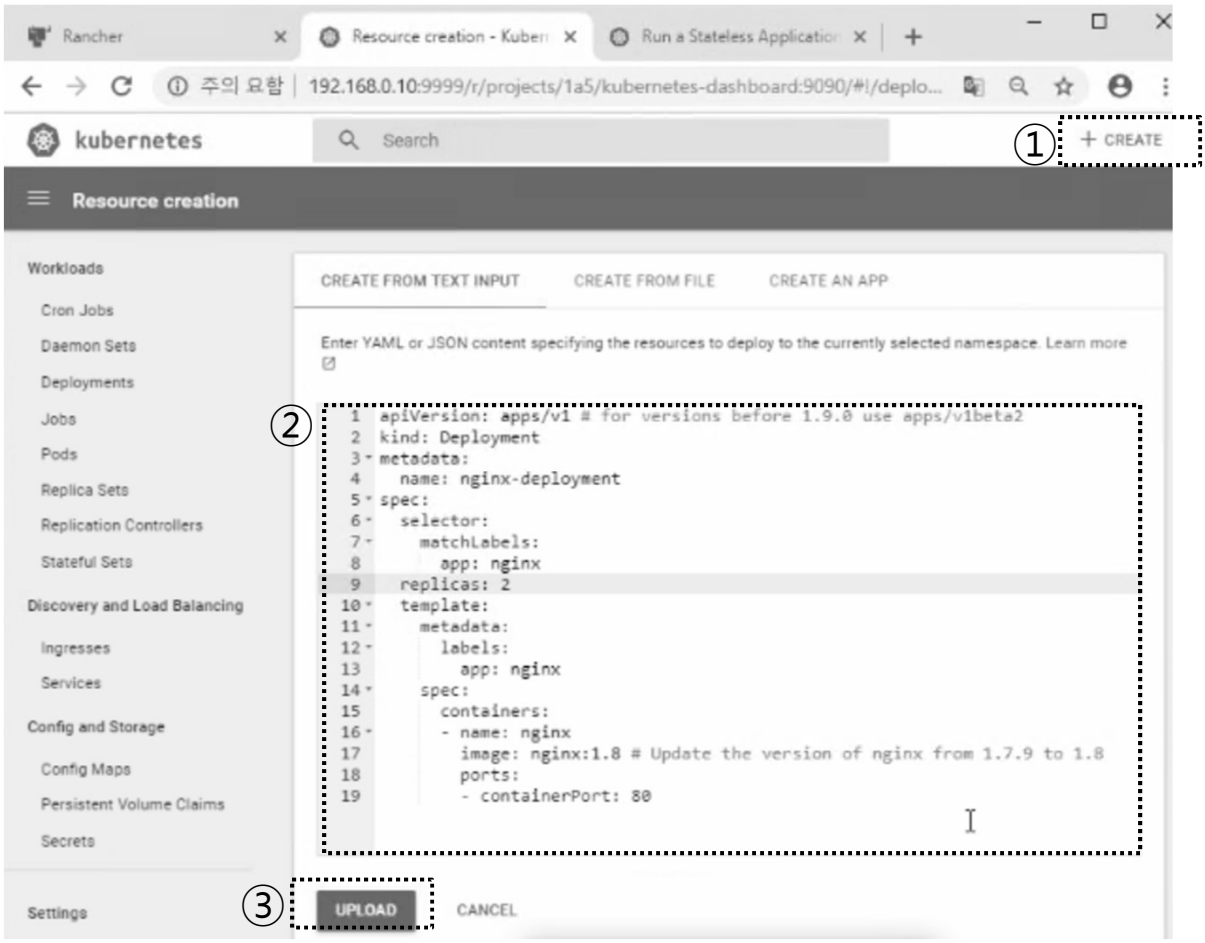
7. Kubernetes 설치

❖ Deployment

① **+ Create**

② **Copy and Paste ‘deployment of nginx’**

③ **Upload**



메모:

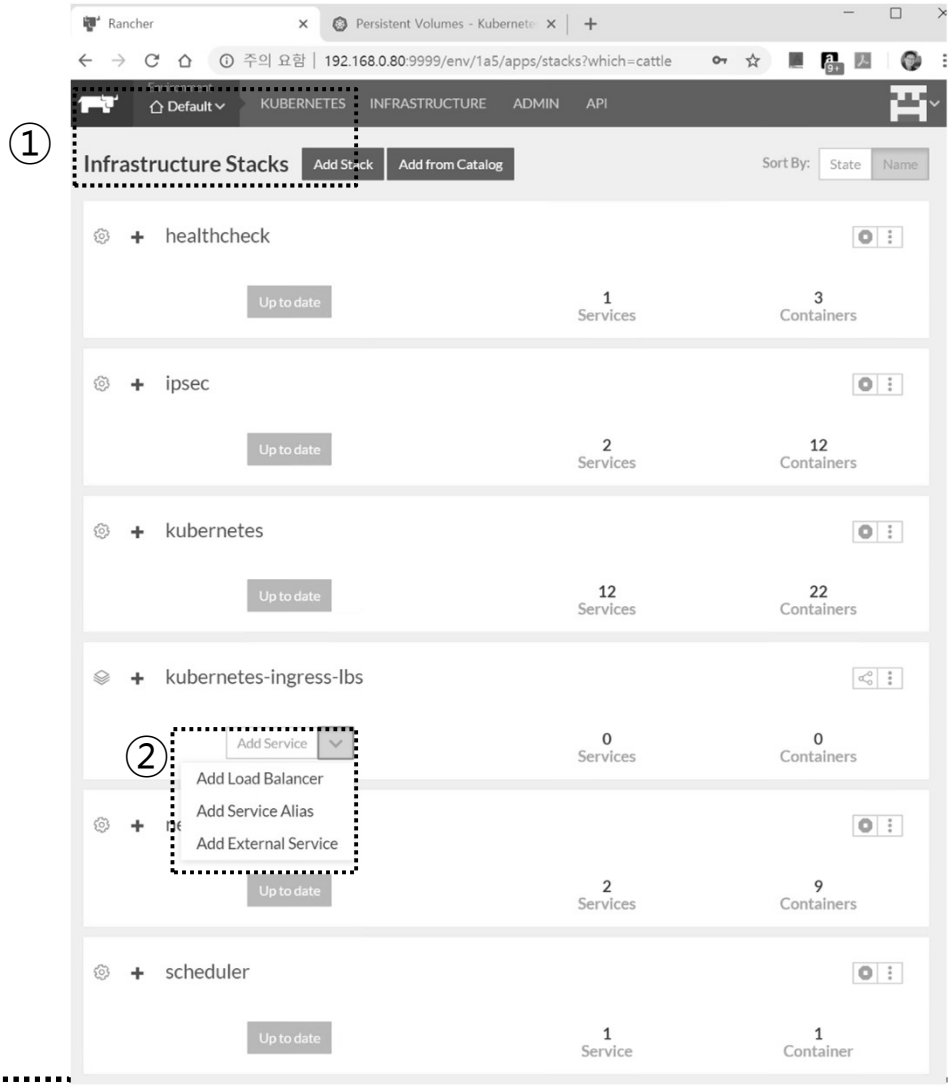
- Replicas Updates

james@jslab.kr

7. Kubernetes 설치

❖ Infrastructure Stacks

① ingress



메모:

- CLI
- Replicas Updates

1. 실습 환경
2. Host
3. Open vSwitch
4. SDN Controller (Docker)
5. mininet (w/ONOS)
6. Rancher
7. Kubernetes 설치

❖ 부록: Docker

부록: Docker

1. **컨테이너** (Docker..)
2. **이미지** (Docker Image)
3. **스웜** (Swarm)
4. **스택과 서비스** (Stack/Service)
5. **Container Networking** (Docker..)

1. 컨테이너 (Docker)

❖ Prerequisites @ Ubuntu 16.04

① **useradd sdn**

② **sudo visudo**

```
# User privilege specification

root ALL=(ALL:ALL) ALL
sdn ALL=(ALL:ALL) ALL
....
```

③ **sudo apt install docker.io** # Optional for 1.13.1 (May 2018)

④ **sudo curl -fsSL https://get.docker.com/ | sh** # latest

⑤ **sudo usermod -aG docker jslab**

⑥ **sudo docker version**

```
james@ubuntu-server:~$ sudo docker version
Client:
 Version:      18.05.0-ce
 API version:  1.37
 Go version:   go1.9.5
 Git commit:   f150324
 Built:        Wed May  9 22:16:25 2018
 OS/Arch:      linux/amd64
 Experimental: false
 Orchestrator: swarm

Server:
 Engine:
  Version:      18.05.0-ce
  API version:  1.37 (minimum version 1.12)
  Go version:   go1.9.5
  Git commit:   f150324
  Built:        Wed May  9 22:14:32 2018
  OS/Arch:      linux/amd64
  Experimental: false
james@ubuntu-server:~$
```

메모:

- sudo apt install docker.io (설치 Docker Version 1.13.1. / hyperledger 17.06.2-ce 이상 권장)
- 실습 교재 cut & paste 사용시 외부에서 putty등을 사용
- Ubuntu Desktop 은 'sudo apt install openssh-server' 로 sshd 설치
- Ubuntu Desktop 은 'sudo apt install curl' 로 curl 설치

1. 컨테이너 (Docker)

❖ docker info

```
sdn@sdn:~$ docker info
Containers: 3
 Running: 3
 Paused: 0
 Stopped: 0
Images: 3
Server Version: 18.06.1-ce
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 468a545b9edcd5932818eb9de8e72413e616e86e
runc version: 69663f0bd4b60df09991c08812a60108003fa340
init version: fec3683
Security Options:
 apparmor
 seccomp
  Profile: default
Kernel Version: 4.4.0-131-generic
Operating System: Ubuntu 16.04.5 LTS
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 3.859GiB
Name: sdn
ID: MC5D:ZY3Q:LSI5:MO36:72FG:UDLE:TVZ7:7I7X:4AO6:ISBA:7FV4:OAIB
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false

WARNING: No swap limit support
```

메모:

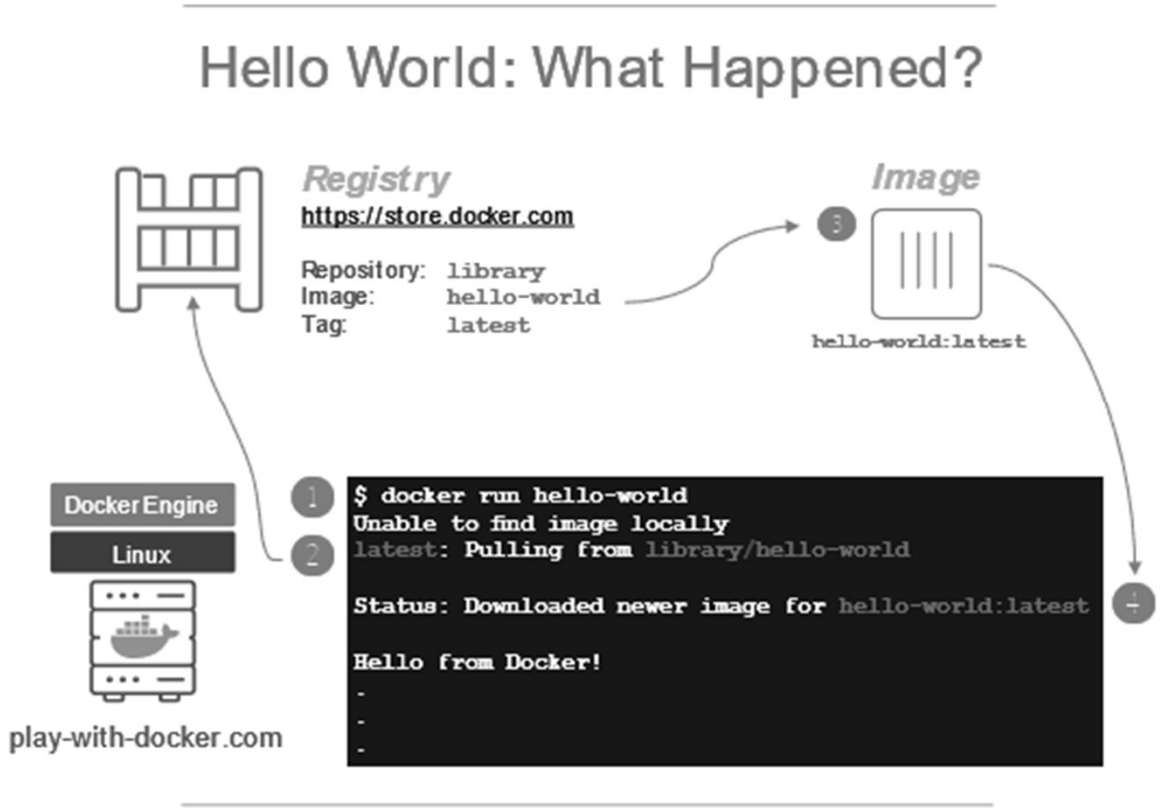
- http://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html

1. 컨테이너 (Docker)

❖ First Container

① **uname -a**

② **docker container run hello-world**



메모:

- https://training.play-with-docker.com/ops-s1-hello/
- docker container start <container ID>

james@jslab.kr

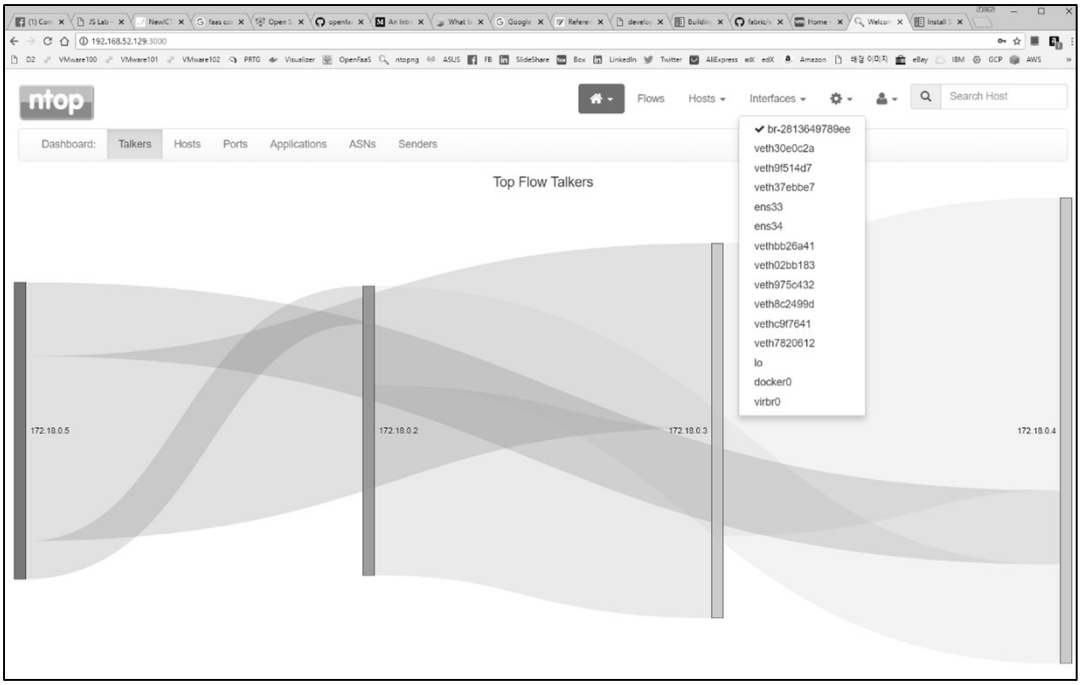
1. 컨테이너 (Docker)

❖ ntopng @ Ubuntu for flow monitoring

① **sudo apt install ntopng** # sudo ntopng

② **sudo systemctl enable ntopng**

③ **http://192.168.99.xx:3000** # admin / admin ➔ password



메모:

- 3장 ovs-docker 실행 후 확인
- Ubuntu Desktop 은 ‘sudo apt install openssh-server’ 로 sshd 설치
- Ubuntu Desktop 은 ‘sudo apt install curl’ 로 curl 설치

james@jslab.kr

1. 컨테이너 (Docker)

❖ ntopng @ Ubuntu for flow monitoring

① **All Hosts**

② **Active Flows**

ntop

Flows | Hosts | Interfaces | Search Host

All Hosts

10 | Filter Hosts

IP Address	Location	Alerts	Name	Seen Since	ASN	Breakdown	Throughput	Traffic
172.18.0.9	Local	0	172.18.0.9	7 min, 46 sec		Sent Rcvd	0 bps	1.7 KB
172.18.0.8	Local	0	172.18.0.8	8 min, 3 sec		Sent Rcvd	463.35 bps	1.98 KB
172.18.0.6	Local	0	172.18.0.6	7 min, 17 sec		Sent Rcvd	0 bps	3.4 KB
172.18.0.5	Local	0	172.18.0.5	8 min, 19 sec		Sent Rcvd	68.45 Kbit	3.88 MB
172.18.0.4	Local	0	172.18.0.4	8 min, 19 sec		Sent Rcvd	78.6 Kbit	3.89 MB
172.18.0.3	Local	0	172.18.0.3	8 min, 19 sec		Sent Rcvd	75.09 Kbit	3.86 MB
172.18.0.2	Local	0	172.18.0.2	8 min, 19 sec		Sent Rcvd	68.14 Kbit	3.87 MB
172.18.0.10	Local	0	172.18.0.10	7 min, 28 sec		Sent Rcvd	0 bps	1.7 KB

Local Hosts Active Flows Matrix

	172.18.0.3	172.18.0.4	172.18.0.6	172.18.0.5	172.18.0.8	172.18.0.2	172.18.0.10
172.18.0.3		497.3 KB 504.39 KB		437.58 KB 438.14 KB	112 B 178 B	434.39 KB 442.83 KB	
172.18.0.4	504.39 KB 497.3 KB		178 B 112 B	448.29 KB 437.77 KB		444.98 KB 442.93 KB	112 B 178 B
172.18.0.6		112 B 178 B		112 B 178 B			
172.18.0.5	438.14 KB 437.58 KB	437.77 KB 448.29 KB	178 B 112 B			499.65 KB 498.83 KB	
172.18.0.8	178 B 112 B						
172.18.0.2	442.83 KB 434.39 KB	442.93 KB 444.98 KB		498.83 KB 499.65 KB			
172.18.0.10		178 B 112 B					

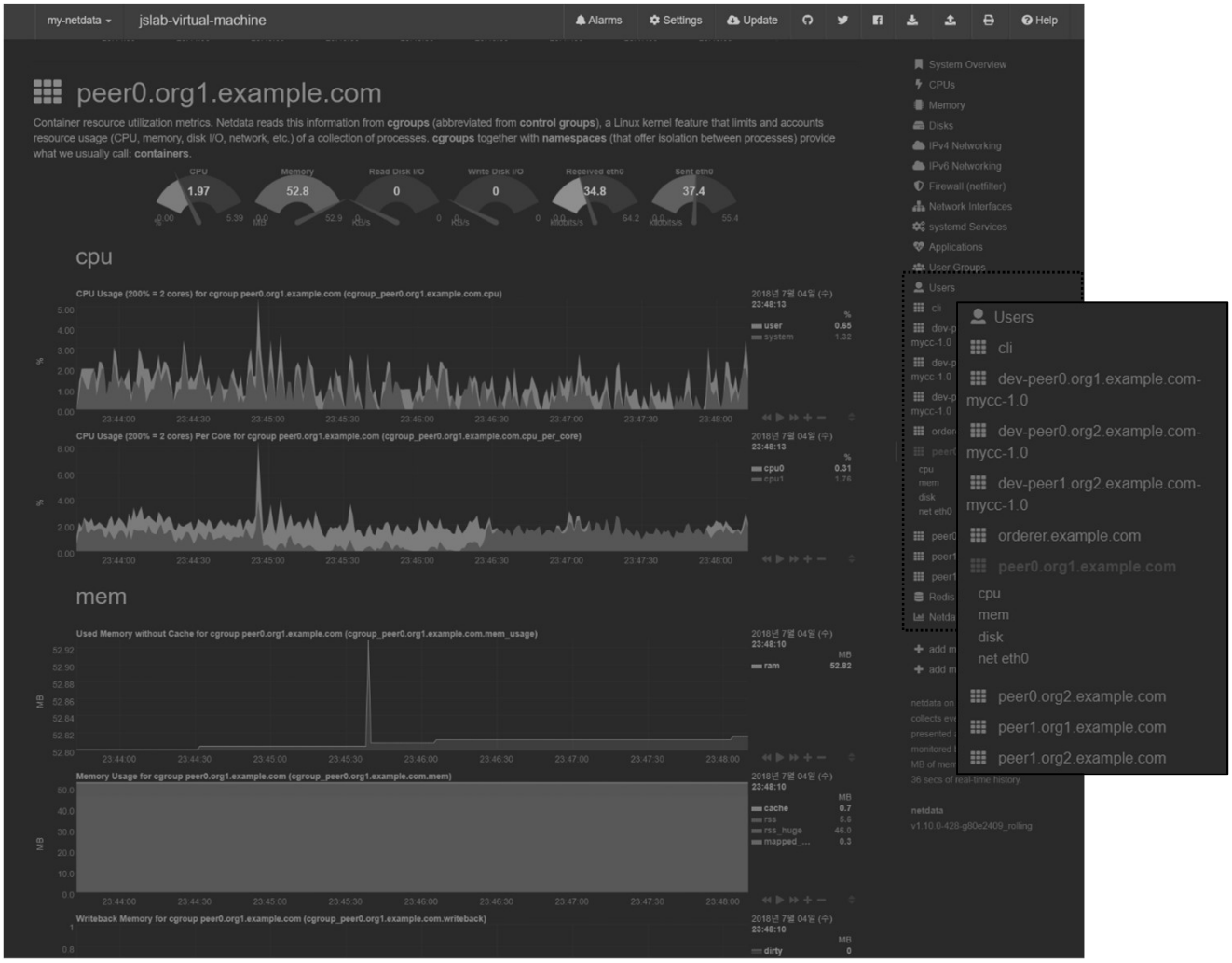
메모:

1. 컨테이너 (Docker)

❖ netdata @ Ubuntu for resource monitoring

① **bash <(curl -Ss https://my-netdata.io/kickstart.sh)**

② **http://192.168.99.100:19999/**



메모:

1. 컨테이너 (Docker)

❖ **설치/실행** (예: Ubuntu @ www.docker.com)

① **sudo apt update**

② **sudo apt install -y apt-transport-https ca-certificates software-properties-common curl**

③ **curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**

④ **sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"**

⑤ **sudo apt update**

⑥ **sudo apt install -y docker-ce**

⑦ **sudo usermod -aG docker userID**

⑧ **sudo systemctl restart ttyd**

⑨ **exit**

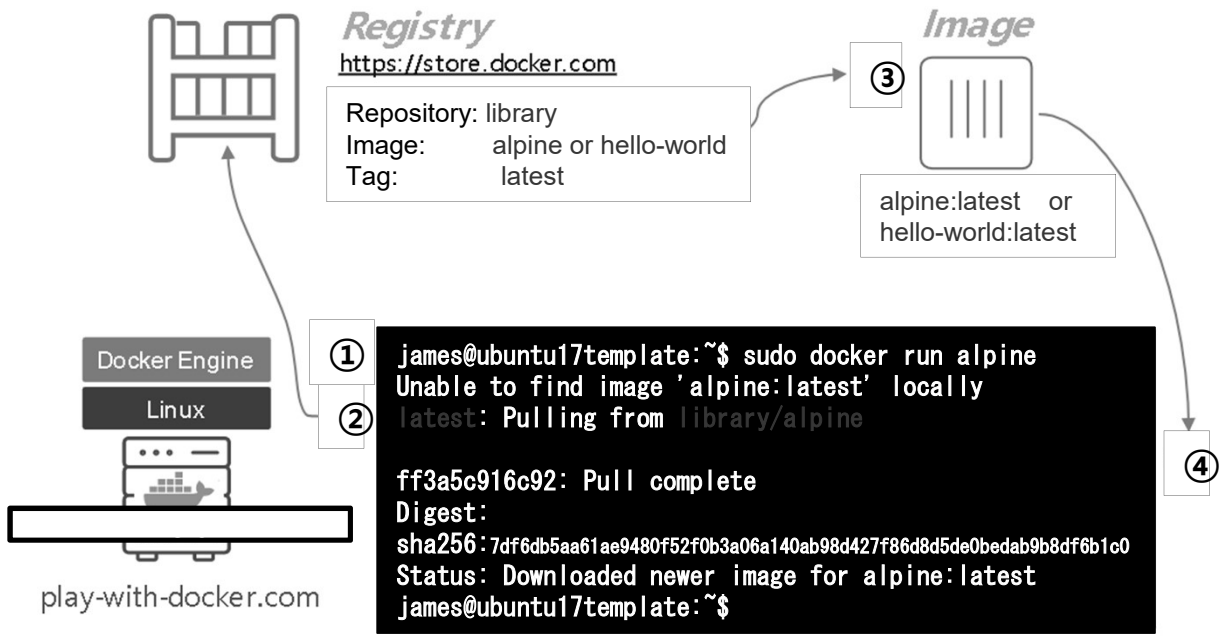
메모:

- ‘curl -fsSL https://get.docker.com/ | sh’ 명령어는 최신 버전의 Docker 설치
- docker container run alpine # before issuing ‘sudo docker image pull alpine’
- Alpine Linux 기반 Docker 이미지는 5 MB 크기임
- Id # for checking id

1. 컨테이너 (Docker)

❖ 각 호스트에 도커(Docker) 설치/실행 @ Ubuntu (선택)

① **curl -fsSL https://get.docker.com/ | sh** # @ General (선택)
② **systemctl stop firewalld && systemctl disable firewalld**
③ **sudo systemctl enable docker**
④ **sudo systemctl start docker**
⑤ **sudo docker image pull alpine**
⑥ **sudo docker image ls**
⑦ **sudo docker container run alpine**



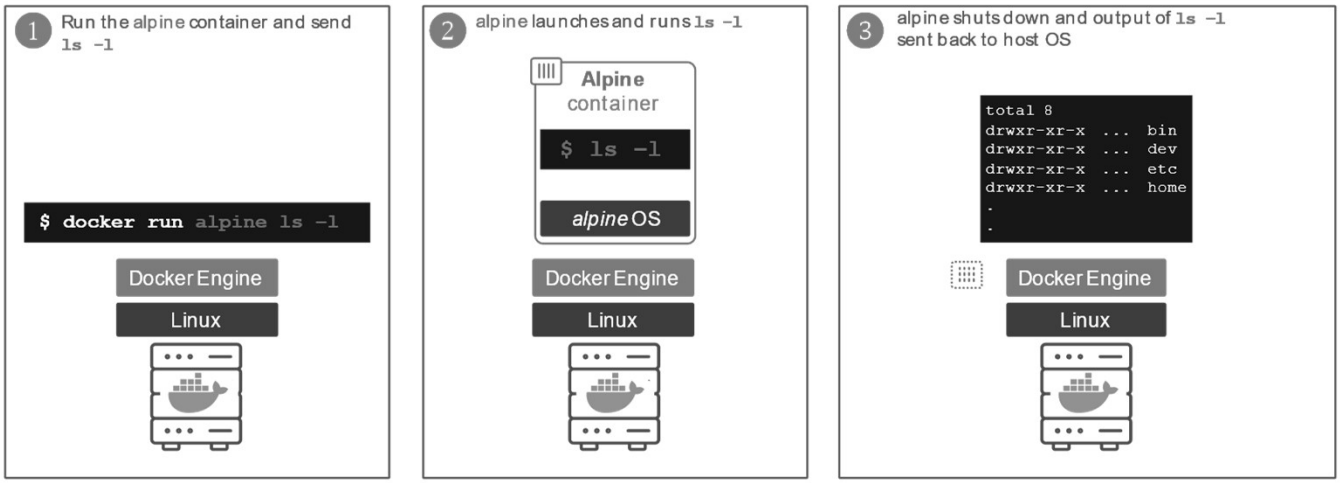
메모:

- 'curl -fsSL https://get.docker.com/ | sh' 명령어는 최신 버전의 Docker 설치
- docker container run alpine # before issuing 'sudo docker image pull alpine'
- Alpine Linux 기반 Docker 이미지는 5 MB 크기임
- 실제 적용시 firewalld 사용 권장
- http://play-with-docker.com

1. 컨테이너 (Docker)

❖ Alpine Linux 컨테이너 @ Ubuntu (선택)

① **sudo docker image pull alpine**
② **sudo docker image ls**
③ **sudo docker container run alpine ls -l**
④ **sudo docker container run alpine echo "hello from alpine“**
⑤ **sudo docker image ls**



메모:

- uname (short for unix name) is a computer program in Unix and Unix-like computer operating systems that prints the name, version and other details about the current machine and the operating system running on it.
- sudo docker attach ‘CONTAINER ID’
- checking hello.txt (반복 ⑪ ~ ⑭)

1. 컨테이너 (Docker)

❖ Alpine Linux 컨테이너 @ Ubuntu (선택)

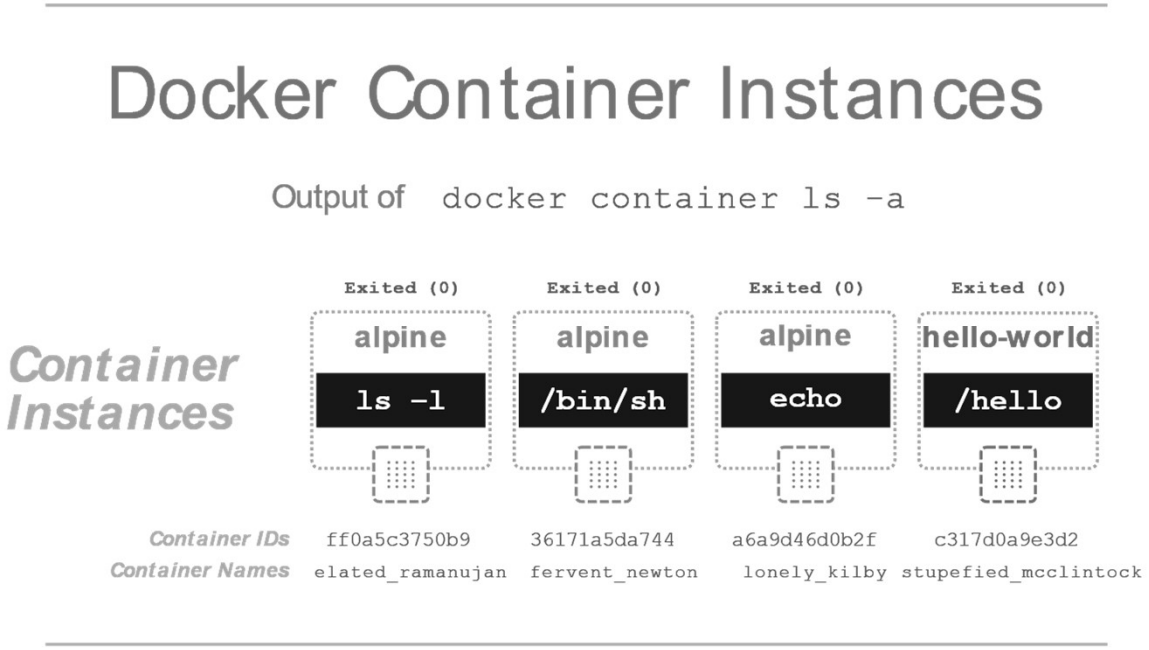
① **sudo docker container run alpine /bin/sh**

② **sudo docker container run -it alpine /bin/sh** # shell prompt

③ **/ # ls -l** # @ Alpine

④ **/ # uname -a** # @ Alpine

⑤ **/ # exit** # @ Alpine



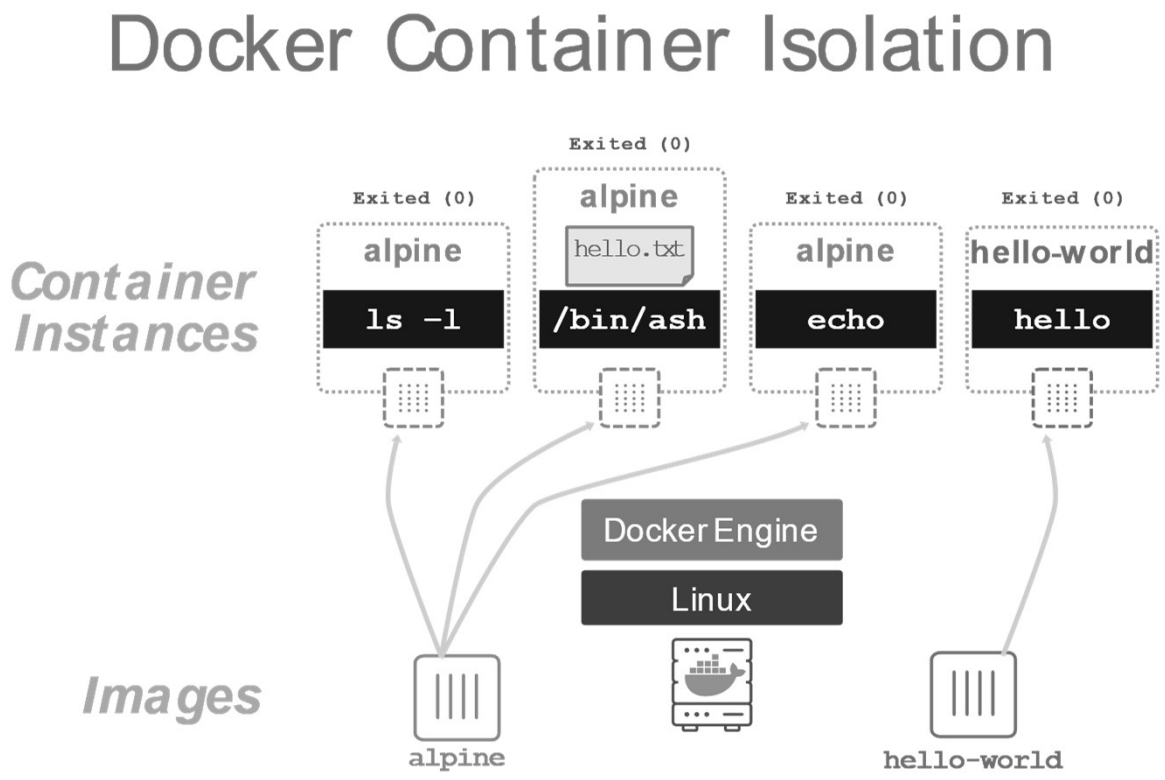
메모:

- uname (short for unix name) is a computer program in Unix and Unix-like computer operating systems that prints the name, version and other details about the current machine and the operating system running on it.
- sudo docker attach 'CONTAINER ID'

1. 컨테이너 (Docker)

❖ Alpine Linux 컨테이너 @ Ubuntu (선택)

① **sudo docker container run -it alpine /bin/sh** # @ Alpine
② **/ # echo "hello world" > hello.txt** # @ Alpine
③ **/ # ls** # @ Alpine
④ **/ # exit** # @ Alpine
⑤ **sudo docker container ls -a**



메모:

- sudo docker attach ‘CONTAINER ID’
- checking hello.txt (순서 반복)

1. 컨테이너 (Docker)

❖ ghost

① **sudo docker run --name ghost1 -d ghost**

컨테이너 ‘ghost’는 포트 미지정시 기정포트(Default Port) 2368로 시작

② **sudo docker run --name ghost2 -p 8080:2368 -d ghost**

http://localhost:8080 or http://host-ip:8080 접속 가능 컨테이너

③ **sudo docker run --name ghost3 -v /path/to/ghost/blog:/var/lib/ghost ghost**

사용 호스트의 컨텐츠를 이미지에 지정하여 사용하는 컨테이너
데이터 컨테이너 ‘/var/lib/ghost’로 대체하여 사용하는 컨테이너

④ **sudo docker run --name ghost4 --volumes-from some-ghost-data ghost**

데이터 컨테이너 ‘/var/lib/ghost’로 대체하여 사용하는 컨테이너

메모:

- docker run [options] image: tag [command, args]
- docker restart [Options] Container ID (s)
- docker attach[Options] Container ID
- docker rm [Options] Container(s)
- # 생성한 모든 컨테이너 보기: sudo docker container ls -a

1. 컨테이너 (Docker)

❖ 요약 (Basic commands)

① **docker images** # 현재 사용 가능한 image 목록을 출력. -a 옵션을 주면 모든 것을 보여줌

② **docker ps** # 현재 사용 가능한 컨테이너 목록을 출력. -a 옵션을 주면 모든 것을 보여줌

③ **docker pull <아이디>/<이미지 이름>:<태그>** # docker hub 이미지 가지고 옴

④ **docker run -it <아이디>/<이미지 이름>:<태그> /bin/bash**
-it 실행한 명령이 Console에 붙어서 진행. i는 interactive, t는 tty를 의미

⑤ **docker container run <container 이름> ls -l**
ls -l 명령어를 실행하며 컨테이너를 실행

⑥ **docker container run -it --name <container 별명> <image 이름> /bin/ash**
--name# # 통해 container 이름 부여. container 이름을 부여하지 않으면 랜덤하게 생성

⑦ **docker container start <container ID>**
docker container에서 명령어 실행

⑧ **docker container exec <container ID> ls**
exec은 container에서 명령어를 실행

⑨ **docker diff <container 별명>**
컨테이너가 부모 이미지와 파일 변경 사항을 확인할 수 있는 명령어

⑩ **docker commit <container ID> <아이디>/<이미지 이름>:<태그>**
새로운 도커 이미지 생성

⑪ **docker push <아이디>/\<이미지 이름\>:<태그>**
docker hub에 이미지 업로드

⑫ **docker build --tag <아이디>/\<이미지 이름\>:<태그> .**
Dockerfile 생성 위치에서 실행하면 이미지 생성

메모:

1. 컨테이너 (Docker)

❖ 요약 (Basic commands)

① docker

```
Management Commands:
  config      Manage Docker configs
  container   Manage containers
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container (creates a new writeable container layer)
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes
```

메모:

부록: Docker

1. **컨테이너** (Docker..)
2. **이미지** (Docker Image)
3. **스웜** (Swarm)
4. **스택과 서비스** (Stack/Service)
5. **Container Networking** (Docker..)

2. 이미지 (Docker Image)

❖ 컨테이너에서 이미지 생성 @ Ubuntu (선택)

① **sudo docker container run -ti ubuntu bash**
② **/# apt-get update**
③ **/# apt-get install -y figlet**
④ **/# figlet "hello james“**
⑤ **/# exit**

```
james@ubuntu17template:~$ docker container run -ti ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete
b78396653dae: Pull complete
Digest: sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6
Status: Downloaded newer image for ubuntu:latest
root@ba625ffee082:/# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:5 http://security.ubuntu.com/ubuntu xenial-security/universe Sources [77.2 kB]
Get:6 http://archive.ubuntu.com/ubuntu xenial/universe Sources [9802 kB]
Get:7 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [593 kB]
Get:8 http://security.ubuntu.com/ubuntu xenial-security/restricted amd64 Packages [12.7 kB]
Get:9 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 Packages [427 kB]
Get:10 http://security.ubuntu.com/ubuntu xenial-security/multiverse amd64 Packages [3492 B]
Get:11 http://archive.ubuntu.com/ubuntu xenial/main amd64 Packages [1558 kB]
Get:12 http://archive.ubuntu.com/ubuntu xenial/restricted amd64 Packages [14.1 kB]
Get:13 http://archive.ubuntu.com/ubuntu xenial/universe amd64 Packages [9827 kB]
Get:14 http://archive.ubuntu.com/ubuntu xenial/multiverse amd64 Packages [176 kB]
Get:15 http://archive.ubuntu.com/ubuntu xenial-updates/universe Sources [250 kB]
Get:16 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [962 kB]
Get:17 http://archive.ubuntu.com/ubuntu xenial-updates/restricted amd64 Packages [13.1 kB]
Get:18 http://archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [792 kB]
Get:19 http://archive.ubuntu.com/ubuntu xenial-updates/multiverse amd64 Packages [18.5 kB]
Get:20 http://archive.ubuntu.com/ubuntu xenial-backports/main amd64 Packages [5153 B]
Get:21 http://archive.ubuntu.com/ubuntu xenial-backports/universe amd64 Packages [7734 B]
Fetched 25.1 MB in 7s (3317 kB/s)
Reading package lists... Done
root@ba625ffee082:/# apt-get install -y figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  figlet
0 upgraded, 1 newly installed, 0 to remove and 9 not upgraded.
Need to get 190 kB of archives.
After this operation, 744 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu xenial/universe amd64 figlet amd64 2.2.5-2 [190 kB]
Fetched 190 kB in 1s (102 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package figlet.
(Reading database ... 4768 files and directories currently installed.)
Preparing to unpack .../figlet_2.2.5-2_amd64.deb ...
Unpacking figlet (2.2.5-2) ...
Setting up figlet (2.2.5-2) ...
update-alternatives: using /usr/bin/figlet-figlet to provide /usr/bin/figlet (figlet) in auto mode
root@ba625ffee082:/# figlet "hello james"
 _          _ _          _
| |__   ___| | | ___    (_) __ _ _ __ ___   ___  ___
| '_ ¥ / _ ¥ | |/ _ ¥   | |/ _` | '_ ` _ ¥ / _ ¥/ __|
| | | |  __/ | | (_) |  | | (_| | | | | | |  __/¥__ ¥
|_| |_|¥___|_|_|¥___/  _/ |¥__,_|_| |_| |_|¥___||___/
                      |__/
root@ba625ffee082:/#
```

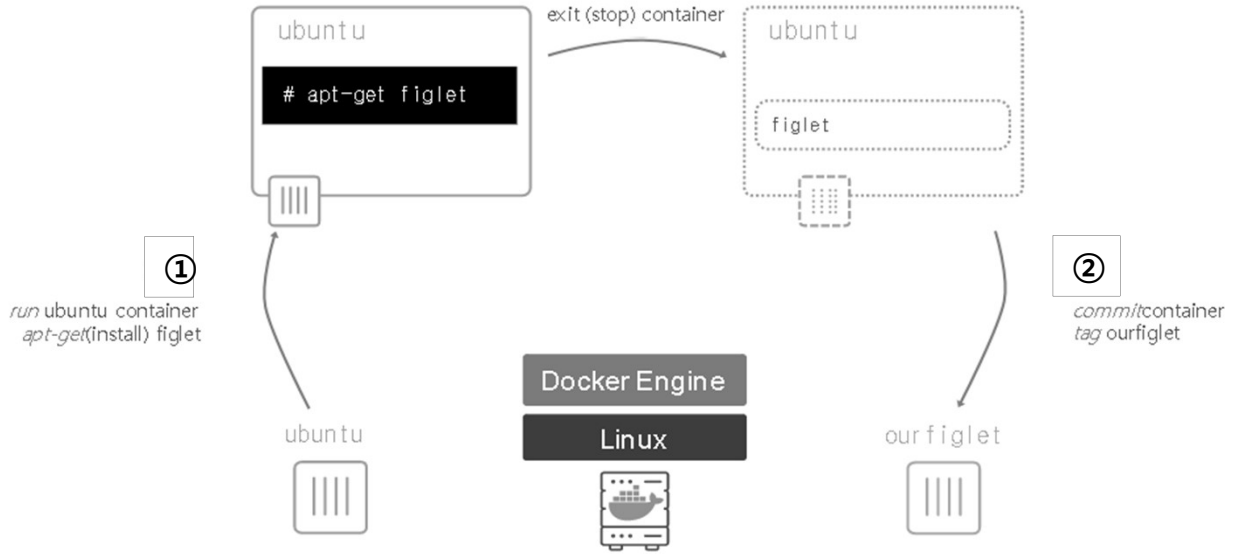
메모:

- 우분투(Ubuntu) 실행후 업데이트와 figlet 설치

2. 이미지 (Docker Image)

❖ 컨테이너에서 이미지 생성 @ Ubuntu (선택)

① **sudo docker container ls -a**
② **sudo docker image ls**
③ **sudo docker container commit CONTAINER_ID**
④ **sudo docker image ls**
⑤ **sudo docker image tag <IMAGE_ID> myfiglet**
⑥ **sudo docker image ls**



메모:

- docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
- It can be useful to commit a container's file changes or settings into a new image.
- Container ID와 Image ID는 다른 것과 겹치지 않는 1 글자 이상 가능

2. 이미지 (Docker Image)

❖ 생성 이미지 확인 @ Ubuntu (선택)

① **sudo docker container ls -a**

② **sudo docker image ls**

③ **sudo docker container commit CONTAINER_ID**

④ **sudo docker image ls**

⑤ **sudo docker image tag <IMAGE_ID> myfiglet**

⑥ **sudo docker image ls**

⑦ **sudo docker container run myfiglet figlet hello james**

```
james@ubuntu17template:~$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
ba625ffee082        ubuntu              "bash"              13 minutes ago      Exited (2) 6 minutes ago
musing_colden
james@ubuntu17template:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              f975c5035748        3 weeks ago         112MB
james@ubuntu17template:~$ docker container commit ba
sha256:4555e45525c1a53400e41436601b22789ce8cb645c1274eb86fb4e60f2c81742
james@ubuntu17template:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              4555e45525c1        4 seconds ago       154MB
ubuntu              latest              f975c5035748        3 weeks ago         112MB
james@ubuntu17template:~$ docker image tag 45 myfiglet
james@ubuntu17template:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myfiglet            latest              4555e45525c1        4 minutes ago       154MB
ubuntu              latest              f975c5035748        3 weeks ago         112MB
james@ubuntu17template:~$
```

```
james@ubuntu17template:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              4555e45525c1        4 seconds ago       154MB
ubuntu              latest              f975c5035748        3 weeks ago         112MB
james@ubuntu17template:~$
james@ubuntu17template:~$ docker image tag 45 myfiglet
james@ubuntu17template:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myfiglet            latest              4555e45525c1        4 minutes ago       154MB
ubuntu              latest              f975c5035748        3 weeks ago         112MB
james@ubuntu17template:~$ ^C
```

메모:

- 컨테이너에서 이미지 생성 @ Ubuntu (선택)

2. 이미지 (Docker Image)

❖ 이미지 생성 준비 @ Ubuntu (선택)

① vi index.js

- var os = require("os");
- var hostname = os.hostname();
- console.log("hello from " + hostname);

```
var os = require("os");
var hostname = os.hostname();
console.log("hello from " + hostname);
```

vi 에디터 명령어 'esc' 후		
-	:x	Exit, saving changes
-	:q	Exit as long as there have been no changes
-	ZZ	Exit and save changes if any have been made
-	:q!	Exit and ignore any changes

② vi Dockerfile

- FROM alpine
- RUN apk update && apk add nodejs
- COPY . /app
- WORKDIR /app
- CMD ["node","index.js"]

i	Insert before cursor
I	Insert before line
a	Append after cursor
A	Append after line

o	Open a new line after current line
O	Open a new line before current line
r	Replace one character
R	Replace many characters

```
FROM alpine
RUN apk update && apk add nodejs
COPY . /app
WORKDIR /app
CMD ["node","index.js"]
```

메모:

- expose ALL ports: EXPOSE 1-65535
- Dockerfile 사용 이미지(Image) 생성 @ Ubuntu (선택)

2. 이미지 (Docker Image)

❖ Dockerfile 명령어

- ADD copies the file(s) from the specified source on the host system or a URL to the specified destination within the container. (Dockerfile 이 위치한 디렉토리의 파일 -> 이미지에 추가)
- CMD executes the specified command when the container is instantiated. There can be only one CMD inside a Dockerfile. If there's more than one CMD instruction, then the last appearing CMD instruction in the DOCKERFILE will be executed. (컨테이너가 시작될 때 실행되는 명령설정, 한번만 사용가능)
- ENTRYPOINT specifies the default executable that should be run when the container is started. This is a must if you want your image to be runnable or you use CMD.
- ENV sets the environment variables in the Dockerfile, which then can be used as part of the instructions—for example, ENV MySQL_ROOT_PASSWORD mypassword.
- EXPOSE specifies the port number where the container will listen. (생성한 이미지에서 노출할 포트 정의)
- FROM specifies the base image to use to start the build image. This is the very first command, and a mandatory one in the Dockerfile. (베이스가 될 이미지 정의)
- MAINTAINER sets the author information in the generated images—for example, MAINTAINER pkocher@domain.com. (이미지를 생성한 개발자 정보, 도커 1.13.0 버전 이후 사용하지 않음)
- RUN executes the specified command(s) and creates a layer for every RUN instruction. The next layer will be built on the previous committed layer. (이미지를 만들기 위해 컨테이너 내부에서 명령어 실행 명령어의 옵션/인자 값은 배열형태로 전달)
- USER sets the user name or user ID to be used when running the image or various instructions such as RUN, CMD, and ENTRYPOINT.
- VOLUME specifies one or more shared volumes on the host machine that can be accessed from the containers.
- WORKDIR sets the working directory for any RUN, CMD, ENTRYPOINT, COPY, or ADD instruction. (명령어를 실행할 디렉토리 정의, cd 명령과 같은 기능)

메모:

2. 이미지 (Docker Image)

❖ 이미지 생성(Build) @ Ubuntu (선택)

① **sudo docker image build -t ubuntu:v0.1 .**

② **sudo docker images**

```
james@ubuntu17template:~$ sudo docker image build -t ubuntu:v0.1 .
Sending build context to Docker daemon  2.134MB
Step 1/5 : FROM alpine
latest: Pulling from library/alpine
ff3a5c916c92: Pull complete
Digest: sha256:7df6db5aa61ae9480f52f0b3a06a140ab98d427f86d8d5de0bedab9b8df6b1c0
Status: Downloaded newer image for alpine:latest
 ---> 3fd9065eaf02
Step 2/5 : RUN apk update && apk add nodejs
 ---> Running in 37fffa95be62
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz
v3.7.0-141-gd4baade662 [http://dl-cdn.alpinelinux.org/alpine/v3.7/main]
v3.7.0-141-gd4baade662 [http://dl-cdn.alpinelinux.org/alpine/v3.7/community]
OK: 9051 distinct packages available
(1/10) Installing ca-certificates (20171114-r0)
(2/10) Installing nodejs-npm (8.9.3-r1)
(3/10) Installing c-ares (1.13.0-r0)
(4/10) Installing libcrypto1.0 (1.0.2o-r0)
(5/10) Installing libgcc (6.4.0-r5)
(6/10) Installing http-parser (2.7.1-r1)
(7/10) Installing libssl1.0 (1.0.2o-r0)
(8/10) Installing libstdc++ (6.4.0-r5)
(9/10) Installing libuv (1.17.0-r0)
(10/10) Installing nodejs (8.9.3-r1)
Executing busybox-1.27.2-r7.trigger
Executing ca-certificates-20171114-r0.trigger
OK: 61 MiB in 21 packages
Removing intermediate container 37fffa95be62
 ---> d8b0d85a540e
Step 3/5 : COPY . /app
 ---> 978539afbb2b
Step 4/5 : WORKDIR /app
Removing intermediate container 91f6d535586e
 ---> 85cce7cc18b8
Step 5/5 : CMD ["node","index.js"]
 ---> Running in c5de0ca210c2
Removing intermediate container c5de0ca210c2
 ---> d45df6a1c291
Successfully built d45df6a1c291
Successfully tagged ubuntu:v0.1
james@ubuntu17template:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              v0.1                d45df6a1c291        12 minutes ago      52.3MB
myfiglet            latest              4555e45525c1        About an hour ago   154MB
ubuntu              latest              f975c5035748        3 weeks ago         112MB
alpine              latest              3fd9065eaf02        2 months ago        4.15MB
james@ubuntu17template:~$
```

메모:

- Dockerfile 사용 이미지(Image) 생성 @ Ubuntu (선택)

2. 이미지 (Docker Image)

❖ 이미지 생성(Build) @ Ubuntu (선택)

① **sudo docker image build -t hello:v0.1 .**

② **sudo docker images**

```
[root@kubemaster ~]# dir
anaconda-ks.cfg     Dockerfile          index.js
docker-compose.yml  example-voting-app  labs
[root@kubemaster ~]# docker image build -t hello:v0.1 .
Sending build context to Docker daemon  387.5MB
Step 1/5 : FROM node
 ---> 42e85254dd8f
Step 2/5 : RUN mkdir -p /usr/src/app
 ---> Using cache
 ---> f1a45f7964aa
Step 3/5 : COPY index.js /usr/src/app
 ---> Using cache
 ---> 0a20b9b48378
Step 4/5 : EXPOSE 8080
 ---> Using cache
 ---> e93372b9a659
Step 5/5 : CMD [ "node", "/usr/src/app/index" ]
 ---> Using cache
 ---> fbdc46fcb363
Successfully built fbdc46fcb363
Successfully tagged hello:v0.1
[root@kubemaster ~]# docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
awesome                             latest              fbdc46fcb363        18 hours ago        673MB
hello                               v0.1                fbdc46fcb363        18 hours ago        673MB
myfiglet                            latest              c93f86228b61        21 hours ago        154MB
node                                latest              42e85254dd8f        3 days ago          673MB
postgres                            <none>              ed5db6e669ff        3 weeks ago         263MB
ubuntu                              latest              f975c5035748        4 weeks ago         112MB
alpine                              latest              3fd9065eaf02        2 months ago        4.15MB
dockercloud/haproxy                 <none>              4d6ae6c16c4d        3 months ago        42.6MB
dockersamples/visualizer            <none>              8dbf7c60cf88        8 months ago        148MB
dockersamples/examplevotingapp_worker <none>            2b1e6048c539        12 months ago       962MB
dockersamples/examplevotingapp_vote <none>              f6e8af4562c1        15 months ago       83.6MB
[root@kubemaster ~]#
```

메모:

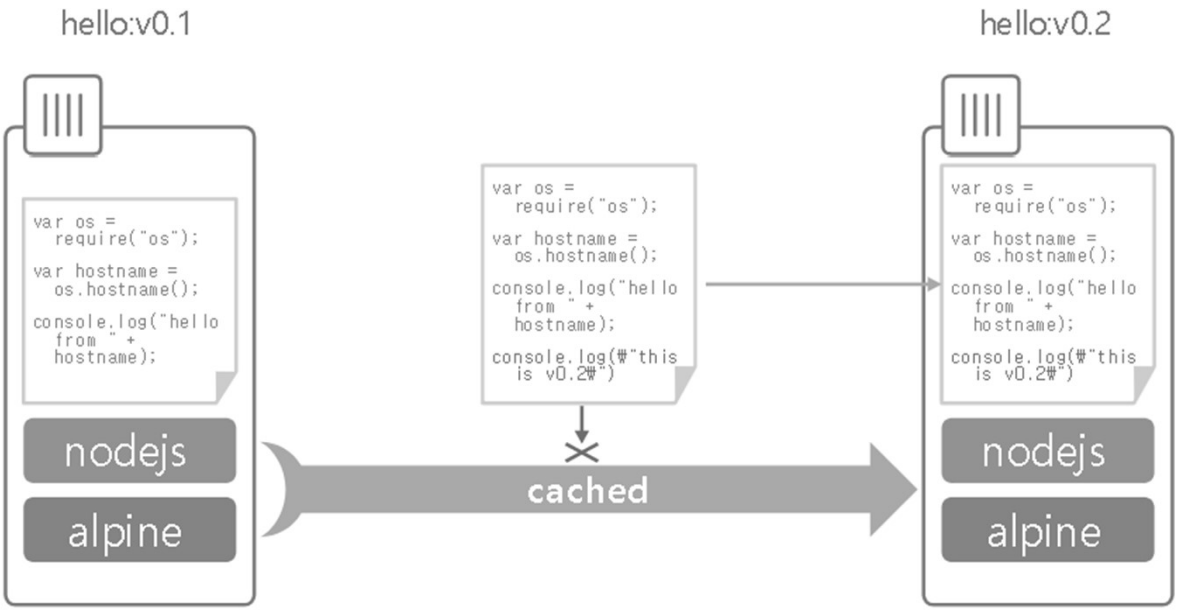
- Dockerfile 사용 이미지(Image) 생성 @ Ubuntu (선택)
- docker rmi[options] image [image, image...]

2. 이미지 (Docker Image)

❖ Image layers @ Ubuntu (선택)

① **sudo docker image build -t hello:v0.2 .**

② **sudo docker images**



메모:

- http://play-with-docker.com

2. 이미지 (Docker Image)

❖ Image inspect @ Ubuntu (선택)

① **sudo docker image pull alpine.**

② **sudo docker image inspect alpine**

③ **sudo docker image inspect --format "{{ json .RootFS.Layers }}" alpine**

④ **sudo docker image ls**

⑤ **sudo docker image inspect --format "{{ json .RootFS.Layers }}" <image ID>**

```
james@ubuntu17template:~$ docker image inspect --format "{{ json .RootFS.Layers }}" alpine
["sha256:cd7100a72410606589a54b932cabd804a17f9ae5b42a1882bd56d263e02b6215"]
james@ubuntu17template:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello               v0.1                d45df6a1c291        36 minutes ago      52.3MB
ubuntu              v0.1                d45df6a1c291        36 minutes ago      52.3MB
myfiglet            latest              4555e45525c1        About an hour ago   154MB
ubuntu              latest              f975c5035748        3 weeks ago         112MB
alpine              latest              3fd9065eaf02        2 months ago        4.15MB
james@ubuntu17template:~$ docker image inspect --format "{{ json .RootFS.Layers }}" hello

Error: No such image: hello
james@ubuntu17template:~$ docker image inspect --format "{{ json .RootFS.Layers }}" hello:v0.1
["sha256:cd7100a72410606589a54b932cabd804a17f9ae5b42a1882bd56d263e02b6215","sha256:15975d6f3f707757bbbd49500c5b0b63b36aa92e11c35f7ff92
f5ce0019981dd","sha256:371e14427d436b2a2e9c9b7c87227a22df2b39d3b46b61c13d10d2a71f382bb3"]
```

메모:

- Image Inspection @ Ubuntu (선택)
- docker logs [Options] Container (docker log eded3539719c)

부록: Docker

1. **컨테이너** (Docker..)
2. **이미지** (Docker Image)
3. **스웜** (Swarm)
4. **스택과 서비스** (Stack/Service)
5. **Container Networking** (Docker..)

3. 스웜 (Swarm)

❖ Swarm Mode

① **sudo docker swarm init --advertise-addr 192.168.99.100**
② **git clone https://github.com/docker/example-voting-app**
③ **cd example-voting-app**
④ **cat docker-stack.yml**
⑤ **sudo docker node ls**

```
sdn@sdn:~$ sudo docker swarm init --advertise-addr 192.168.99.100
Swarm initialized: current node (5hggzgml14qvq37ulgd78xuto) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-01iluiub1y65nuau4roreavt0jp5aowi13fekf4tgl1hgtf12p-e31nhziaqgba3yv4xltt3hur6
192.168.99.100:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

sdn@sdn:~$

sdn@sdn:~$ git clone https://github.com/docker/example-voting-app
Cloning into 'example-voting-app'...
remote: Enumerating objects: 228, done.
remote: Counting objects: 100% (228/228), done.
remote: Compressing objects: 100% (139/139), done.
remote: Total 746 (delta 74), reused 217 (delta 70), pack-reused 518
Receiving objects: 100% (746/746), 847.12 KiB | 446.00 KiB/s, done.
Resolving deltas: 100% (257/257), done.
Checking connectivity... done.
sdn@sdn:~$ dir
example-voting-app
```

```
sdn@sdn:~$ sudo docker node ls
ID                          HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ENGINE VERSION
5hggzgml14qvq37ulgd78xuto *   sdn                 Ready               Active              Leader
18.06.1-ce
sdn@sdn:~$
```

메모:

- Docker는 Kubernetes 지원 기능을 출시
- CentOS와 Ubuntu가 동일한 Docker Swarm 모드 명령어 사용
- sudo docker swarm init --advertise-addr $(hostname -i)

3. 스웜 (Swarm)

❖ 스웜 종료 (선택)

① **docker swarm leave --force** # @ Worker 1
② **docker swarm leave --force** # @ Worker 2
③ **docker swarm leave --force** # @ Manager 3
④ **docker swarm leave --force** # @ Manager

메모:

- Manager 노드 구동 호스트의 리부팅시 Swarm 모드 자동 실행 / 서비스 복구

부록: Docker

1. **컨테이너** (Docker..)
2. **이미지** (Docker Image)
3. **스웜** (Swarm)
4. **스택과 서비스** (Stack/Service)
5. **Container Networking** (Docker..)

4. 스택과 서비스 (Stack/Service)

❖ stack 파일

⑤ cat docker-stack.yml

```
james@ubuntu17template:~/example-voting-app$ cat docker-
stack.yml
version: "3"
services:

  redis:
    image: redis:alpine
    ports:
      - "6379"
    networks:
      - frontend
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure
  db:
    image: postgres:9.4
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - backend
    deploy:
      placement:
        constraints: [node.role == manager]
  vote:
    image: dockersamples/examplevotingapp_vote:before
    ports:
      - 5000:80
    networks:
      - frontend
    depends_on:
      - redis
    deploy:
      replicas: 2
      update_config:
        parallelism: 2
      restart_policy:
        condition: on-failure
  result:
    image: dockersamples/examplevotingapp_result:before
    ports:
      - 5001:80
    networks:
      - backend
    depends_on:
      - db
```

```
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure

  worker:
    image: dockersamples/examplevotingapp_worker
    networks:
      - frontend
      - backend
    deploy:
      mode: replicated
      replicas: 1
      labels: [APP=VOTING]
      restart_policy:
        condition: on-failure
        delay: 10s
        max_attempts: 3
        window: 120s
      placement:
        constraints: [node.role == manager]

  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    stop_grace_period: 1m30s
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]

networks:
  frontend:
  backend:

volumes:
  db-data:
james@ubuntu17template:~/example-voting-app$
```

메모:

- git clone https://github.com/docker/example-voting-app
- cd example-voting-app
- Docker는 Kubernetes 지원 기능을 출시예정 (2018년 4월 현재 Beta)
- 8080은 다른 서비스 사용 가능하여 8181 등으로 변환 필요 할 수 있음

4. 스택과 서비스 (Stack/Service)

❖ stack 실행

① **sudo docker ps** # check @ each host

② **sudo docker stack deploy --compose-file=docker-stack.yml voting_stack** # @ /example-voting-app

③ **sudo docker stack ls**

④ **sudo docker stack services voting_stack**

⑤ # http://192.168.0.70:8080 for Visualizer @ Chrome

⑥ # http://192.168.0.70:5000 for Vote

⑦ # http://192.168.0.60:5001 for Result

```
james@ubuntu17template:~/example-voting-app$ docker stack deploy --compose-file=docker-stack.yml voting_stack
Creating network voting_stack_backend
Creating network voting_stack_frontend
Creating network voting_stack_default
Creating service voting_stack_worker
Creating service voting_stack_visualizer
Creating service voting_stack_redis
Creating service voting_stack_db
Creating service voting_stack_vote
Creating service voting_stack_result
james@ubuntu17template:~/example-voting-app$ docker stack ls
NAME                SERVICES
voting_stack        6
james@ubuntu17template:~/example-voting-app$ docker ps
CONTAINER ID        IMAGE                                       COMMAND                  CREATED             STATUS
PORTS               NAMES
fe58543baac4        dockersamples/examplevotingapp_result:before   "node server.js"         53 seconds ago      Up 37 seconds
80/tcp              voting_stack_result.1.e2wih2eg3c65wmnjzdl7eythi
25e44cd8c1e6        postgres:9.4                                "docker-entrypoint.s…"   About a minute ago   Up 58 seconds
5432/tcp            voting_stack_db.1.z7boxqf07ad8b5iiwra3fd7rb
691df72c91e6        dockersamples/examplevotingapp_vote:before     "gunicorn app:app -b…"   About a minute ago   Up 57 seconds
80/tcp              voting_stack_vote.2.xvly71tow6acj7eunn6m89b51
483991c28cac        dockersamples/examplevotingapp_vote:before     "gunicorn app:app -b…"   About a minute ago   Up 58 seconds
80/tcp              voting_stack_vote.1.uk16grl93w3wxsvnrntuamsws
b71c0e3de445        dockersamples/examplevotingapp_worker:latest   "/bin/sh -c 'dotnet …"   About a minute ago   Up About a minute
voting_stack_worker.1.299ec7wqx8plcd2tvrjyelbrm
45ddbd99341f        dockersamples/visualizer:stable             "npm start"              About a minute ago   Up About a minute
8080/tcp            voting_stack_visualizer.1.6q2yyrm6wb4ykh9afyynrvk8a
9ef47eedebbb        redis:alpine                                "docker-entrypoint.s…"   About a minute ago   Up About a minute
6379/tcp            voting_stack_redis.1.0gwixzc0z7wmd437udymaikl6
james@ubuntu17template:~/example-voting-app$
```

메모:

- git clone https://github.com/docker/example-voting-app
- cd example-voting-app
- watch -n x <your command>
- watch -n 60 ls -l ~/Desktop
- Check ‘immutable infrastructure’

4. 스택과 서비스 (Stack/Service)

❖ stack Operations

① **sudo docker stack services voting_stack**

② **sudo docker service ps voting_stack_vote**

```
james@ubuntu17template:~/example-voting-app$ docker stack services voting_stack
ID              NAME                MODE            REPLICAS        IMAGE
PORTS
1fa0bp9x0a8y    voting_stack_vote   replicated      2/2             dockersamples/examplevotingapp_vote:before
*:5000->80/tcp
1sm84ozdl4vv    voting_stack_db     replicated      1/1             postgres:9.4
ds790f0fcoxj    voting_stack_worker replicated      1/1             dockersamples/examplevotingapp_worker:latest
hqtmv6pgrlmw    voting_stack_result replicated      1/1             dockersamples/examplevotingapp_result:before
*:5001->80/tcp
sythupfy4m2i    voting_stack_visualizer replicated  1/1             dockersamples/visualizer:stable
*:8080->8080/tcp
t0bcnmhrq4n6    voting_stack_redis  replicated      1/1             redis:alpine
*:30000->6379/tcp
james@ubuntu17template:~/example-voting-app$
james@ubuntu17template:~/example-voting-app$ docker service ps voting_stack_vote
ID              NAME                IMAGE                                       NODE            DESIRED STATE   CURRENT
STATE           ERROR               PORTS
uk16grl93w3w    voting_stack_vote.1 dockersamples/examplevotingapp_vote:before  ubuntu17template  Running       Running
20 minutes ago
xvly71tow6ac    voting_stack_vote.2 dockersamples/examplevotingapp_vote:before  ubuntu17template  Running       Running
20 minutes ago
james@ubuntu17template:~/example-voting-app$
```

메모:

- http://play-with-docker.com 참조
- http://192.168.0.60:8080 for Visualizer

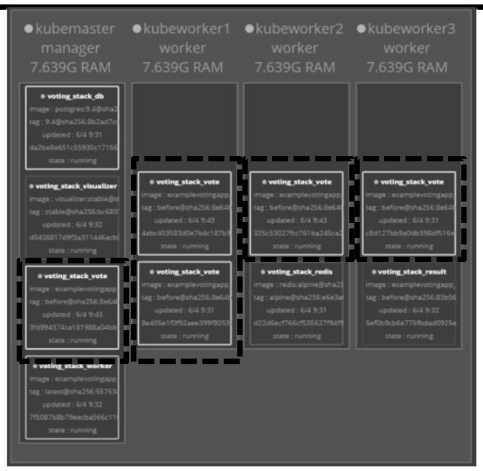
4. 스택과 서비스 (Stack/Service)

❖ scale

① **sudo docker service scale voting_stack_vote=5**

② **sudo docker stack services voting_stack**

```
[root@kubemaster example-voting-app]# docker service scale voting_stack_vote=5
voting_stack_vote scaled to 5
overall progress: 5 out of 5 tasks
1/5: running
2/5: running
3/5: running
4/5: running
5/5: running
verify: Service converged
[root@kubemaster example-voting-app]# docker stack services voting_stack
ID              NAME                MODE         REPLICAS   IMAGE
PORTS
b0693htjttku    voting_stack_redis  replicated   1/1        redis:alpine
*:30000->6379/tcp
d5hqeqg0ckmq    voting_stack_db     replicated   1/1        postgres:9.4
n6s659sn5bhm    voting_stack_visualizer  replicated   1/1   dockersamples/visualizer:stable
*:8080->8080/tcp
r1qha3ld9q4c    voting_stack_vote   replicated   5/5        dockersamples/examplevotingapp_vote:before
*:5000->80/tcp
rhj71cxaysjy    voting_stack_worker replicated   1/1        dockersamples/examplevotingapp_worker:latest
yk6k6vh0ornz    voting_stack_result replicated   1/1        dockersamples/examplevotingapp_result:before
*:5001->80/tcp
[root@kubemaster example-voting-app]#
```



메모:

- http://play-with-docker.com 참조
- git clone https://github.com/docker/example-voting-app
- 'cd example-voting-app' 후 docker stack 실행
- Manager 키 확인: sudo docker swarm join-token manager
- Stack 중지: sudo docker stack rm voting_stack

4. 스택과 서비스 (Stack/Service)

❖ docker network inspect ingress

① **sudo docker network ls**

② **sudo docker network inspect ingress**

```
[root@kubemaster example-voting-app]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
1fe249e36d43        bridge              bridge              local
05191e8b7e19        docker_gwbridge     bridge              local
06322c05f69e        host                host                local
33zsip6je0ns        ingress             overlay             swarm
ed53abe4e032        none                null                local
7s7p1zaiqi7p        voting_stack_backend   overlay          swarm
n3sss1s7elwl        voting_stack_default   overlay          swarm
oao3jy8bdlzu        voting_stack_frontend  overlay          swarm
```

```
jslab@ubuntu70:~/example-voting-app$ sudo docker network inspect ingress
[
    {
        "Name": "ingress",
        "Id": "mzlm5qlmdu3vxrm2kcmfvy78t",
        "Created": "2018-09-11T15:02:59.802809879+09:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.255.0.0/16",
                    "Gateway": "10.255.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": true,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "0020654177d1f8aae57b2dc5b9285cc69a9c3ee1bd35b1054ece3945d994deac": {
                "Name": "voting_stack_result.1.fqmbr20e2auv979tqbf6g392g",
                "EndpointID": "6e3463c299ea9ed644deed9b0fd1275bf4eeac470bb8549b8f06b73f17ba9c83",
                "MacAddress": "02:42:0a:ff:00:07",
                "IPv4Address": "10.255.0.7/16",
                "IPv6Address": ""
            },
            "316e68085cff970ff69925dc30e40aa9cf3ec567b198a866aeb3c8b84c19447c": {
                "Name": "voting_stack_redis.1.3wtwueh2y9v0z91zi8bzaidm9",
                "EndpointID": "c4ef905bf3f40791a2ceda7500f5e7159dac4609e91ace6d9630705b03bd74ba",
                "MacAddress": "02:42:0a:ff:00:0b",
                "IPv4Address": "10.255.0.11/16",
                "IPv6Address": ""
            },
            "56d302a0f2406f08883edf21a777dbbbe3e590b0218a329b9eb0fe620d1f6f0a": {
                "Name": "voting_stack_vote.2.tvpfws2fvygog5opeix56jmos",
                "EndpointID": "bc9a7a7a41003e322d40cb94bac693367066129587436736604394 3aed7438a2",
                "MacAddress": "02:42:0a:ff:00:05",
                "IPv4Address": "10.255.0.5/16",
                "IPv6Address": ""
            },"b5f890af85eeee5ad51786aaf8aadafc9ca72c032d642ab797b3c196c429d6c4": {
                "Name": "voting_stack_vote.1.jq3fscjqgx87gmtvvo6xhu105",
                "EndpointID": "136e4fb25cbdc4c2e7d13729671df40621056e18faa4b2c1508a6d2b12dcc606",
                "MacAddress": "02:42:0a:ff:00:04",
                "IPv4Address": "10.255.0.4/16",
                "IPv6Address": ""
            },
            "f4a492ce6a0e3bbf36339fd9c1c66678cfae216c4bcf362ba2de8f6c30c92fd2": {
                "Name": "voting_stack_visualizer.1.1ytndij70lblg448n0j4k59l7",
                "EndpointID": "217e7399ad7c1397e81182b1e5f483e43bd9e3d23974a121a1ec427f8e919584",
                "MacAddress": "02:42:0a:ff:00:09",
                "IPv4Address": "10.255.0.9/16",
                "IPv6Address": ""
            },
            "ingress-sbox": {
                "Name": "ingress-endpoint",
                "EndpointID": "9e3195e1db405c54758fea15f266255816a65a5f2b263d412e7906e7e51faf33",
                "MacAddress": "02:42:0a:ff:00:02",
                "IPv4Address": "10.255.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4096"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "917e8e1902a8",
                "IP": "192.168.0.70"
            },
            {
                "Name": "43c87f203409",
                "IP": "192.168.0.71"
            }
        ]
    }
]
jslab@ubuntu70:~/example-voting-app$
```

메모:

- Network Operations

4. 스택과 서비스 (Stack/Service)

❖ docker network inspect ingress

① docker network inspect ingress

```
[root@kubemaster example-voting-app]# docker network inspect ingress
[
    {
        "Name": "ingress",
        "Id": "33zsip6je0nseerjfs7iu6u59",
        "Created": "2018-04-05T20:22:13.859576938-04:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.255.0.0/16",
                    "Gateway": "10.255.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": true,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "3fd994374ca181988a04bbd1746e680941d743b762898b391a543489c66c6113": {
                "Name": "voting_stack_vote.4.tgwqgjs7ut11grdue1oeltcr7",
                "EndpointID": "88f2f6cb7e8e67a03f1195a3256e5084b1d477ddce4a28bcfd90c5cefba70ab",
                "MacAddress": "02:42:0a:ff:00:10",
                "IPv4Address": "10.255.0.16/16",
                "IPv6Address": ""
            },
            "d0408817d9f3a311446acb8b0e6e322cf7249b23755f26f61d8a3de27b6710c8": {
                "Name": "voting_stack_visualizer.1.xkme6y2zn4himbhvewo5fbz7s",
                "EndpointID": "7f7a4d6a1ba1fdee804ab7af215781d9a5e9781b028c896c1b5f97bed711a0e",
                "MacAddress": "02:42:0a:ff:00:0e",
                "IPv4Address": "10.255.0.14/16",
                "IPv6Address": ""
            },
            "ingress-sbox": {
                "Name": "ingress-endpoint",
                "EndpointID": "2c1044a5161fb9e3a7c8705a3ae8f52a4aa76e0b1b14fc9d5aaaaf0aa5be55",
                "MacAddress": "02:42:0a:ff:00:02",
                "IPv4Address": "10.255.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4096"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "e529ff7ef122",
                "IP": "192.168.0.60"
            },
            {
                "Name": "eac1c7779806",
                "IP": "192.168.0.61"
            },
            {
                "Name": "67945b943ac1",
                "IP": "192.168.0.62"
            },
            {
                "Name": "1337293b54d9",
                "IP": "192.168.0.63"
            }
        ]
    }
]
[root@kubemaster example-voting-app]#
```

메모:

- Network Operations

4. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성

④ sudo docker network inspect overnet

```
[root@kubemaster ~]# docker network inspect overnet
[
    {
        "Name": "overnet",
        "Id": "2n2Ow14b1ggir4ie2dok2tagz",
        "Created": "2018-04-04T03:57:19.826926805-04:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "7dcedf4eb9edcd271f4e9e16c078931205639bae90c5bc158f4d0a8b6ce04acf": {
                "Name": "myservice.2.qqxmz9cl72rbssjnatk3t08sb",
                "EndpointID": "e7f646133243b5de9e66e50064973aa216c35d79e31e57d76fbc884a5d569b71",
                "MacAddress": "02:42:0a:00:00:06",
                "IPv4Address": "10.0.0.6/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "41816cd15b28",
                "IP": "192.168.0.60"
            },
            {
                "Name": "8ba267a3a74b",
                "IP": "192.168.0.61"
            }
        ]
    }
]
[root@kubemaster ~]#
```

메모:

- 생성 IP 주소 확인

4. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성

⑤ **sudo docker exec -it <CONTAINER ID> /bin/bash**

⑥ **apt-get update && apt-get install -y iputils-ping**

⑦ **cat /etc/resolv.conf** # Check DNS Server @ 127.0.0.11:53

⑧ **ping -c5 myservice**

```
root@7dcedf4eb9ed:/# cat /etc/resolv.conf
search internal-network
nameserver 127.0.0.11
options ndots:0
root@7dcedf4eb9ed:/# ping -c5 myservice
PING myservice (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.068 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.080 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.075 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.067 ms

--- myservice ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.067/0.071/0.080/0.011 ms
root@7dcedf4eb9ed:/#
```

메모:

- DNS 동작 확인

4. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성

⑨ **exit**

⑩ **sudo docker service inspect myservice**

```
root@7dcedf4eb9ed:/# exit
exit
[root@kubemaster ~]# docker service inspect myservice
[
    {
        "ID": "3nzzhjmsoglebijq01y8w0mfu",
        "Version": {
            "Index": 21
        },
        "CreatedAt": "2018-04-04T07:57:19.663257975Z",
        "UpdatedAt": "2018-04-04T07:57:19.66539791Z",
        "Spec": {
            "Name": "myservice",
            "Labels": {},
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image": "ubuntu:latest@sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6",
                    "Args": [
                        "sleep",
                        "infinity"
                    ],
                    "StopGracePeriod": 10000000000,
                    "DNSConfig": {},
                    "Isolation": "default"
                },
                "Resources": {
                    "Limits": {},
                    "Reservations": {}
                },
                "RestartPolicy": {
                    "Condition": "any",
                    "Delay": 5000000000,
                    "MaxAttempts": 0
                },
                "Placement": {
                    "Platforms": [
                        {
                            "Architecture": "amd64",
                            "OS": "linux"
                        },
                        {
                            "OS": "linux"
                        },
                        {
                            "Architecture": "arm64",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "386",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "ppc64le",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "s390x",
                            "OS": "linux"
                        }
                    ]
                },
```

```
                "Networks": [
                        {
                            "Target": "2n20w14b1ggir4ie2dok2tagz"
                        }
                    ],
                    "ForceUpdate": 0,
                    "Runtime": "container"
                },
                "Mode": {
                    "Replicated": {
                        "Replicas": 2
                    }
                },
                "UpdateConfig": {
                    "Parallelism": 1,
                    "FailureAction": "pause",
                    "Monitor": 5000000000,
                    "MaxFailureRatio": 0,
                    "Order": "stop-first"
                },
                "RollbackConfig": {
                    "Parallelism": 1,
                    "FailureAction": "pause",
                    "Monitor": 5000000000,
                    "MaxFailureRatio": 0,
                    "Order": "stop-first"
                },
                "EndpointSpec": {
                    "Mode": "vip"
                }
            },
            "Endpoint": {
                "Spec": {
                    "Mode": "vip"
                },
                "VirtualIPs": [
                    {
                        "NetworkID": "2n20w14b1ggir4ie2dok2tagz",
                        "Addr": "10.0.0.4/24"
                    }
                ]
            }
        }
    ]
    [root@kubemaster ~]#
```

메모:

- 서비스 분석

4. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성 (예: visualizer)

```
① visualizer:
②     image: dockersamples/visualizer:stable
③     ports:
④       - "8282:8080"
⑤     stop_grace_period: 1m30s
⑥     volumes:
⑦       - "/var/run/docker.sock:/var/run/docker.sock"
⑧     deploy:
⑨       placement:
⑩         constraints: [node.role == manager]
```

메모:

- ONOS Install as a service

4. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성 (예: ONOS)

```
① sudo docker service create \
②         --name onos \
③         --publish 8383:8181/tcp \
④         --publish 6653:6653/tcp \
⑤         --constraint node.role==manager \
⑥         --mount
type=bind,src=/var/run/docker.sock,dst=/var/run/docker.s
ock \
⑦         onosproject/onos:latest
```

메모:

- ONOS Install as a service
- Check Application started: OpenFlow Agent, Base Provider, LLDP Link Provider, Host Location Provider, Reactive Forwarding)

4. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성 (예: Prometheus, ghost)

```
① sudo docker service create \
②       --name prom \
③       --publish 9090:9090/tcp \
④       --constraint node.role==manager \
⑤       --mount
type=bind,src=/var/run/docker.sock,dst=/var/run/docker.s
ock \
⑥       prom/prometheus:latest

⑦ sudo docker service create \
⑧  --name ghost \
⑨       --publish 8080:2368/tcp \
⑩       --constraint node.role==worker \
⑪       --mount
type=bind,src=/var/run/docker.sock,dst=/var/run/docker.s
ock \
⑫       /path/to/ghost/blog:/var/lib/ghost
```

메모:

- Prometheus / Ghost Install as a service

4. 스택과 서비스 (Stack/Service)

❖ 요약

① **docker ps**
② **docker kill yourcontainerid1 yourcontainerid2**
③ **docker swarm leave --force** # @ Manager
④ **docker swarm leave --force** # @ Worker

⑤ **git clone https://github.com/docker/example-voting-app**
⑥ **cd example-voting-app**
⑦ **cat docker-stack.yml**

⑧ **docker stack deploy --compose-file=docker-stack.yml voting_stack**
⑨ **docker stack ls**
⑩ **docker stack services voting_stack**

⑪ # http://192.168.0.60:8080 for Visualizer
⑫ # http://192.168.0.60:5000 for vote
⑬ # http://192.168.0.60:5001 for result

메모:

- Microservices is a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services should be fine-grained and the protocols should be lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity and makes the application easier to understand, develop and test.
(https://en.wikipedia.org/wiki/Microservices 참조)

4. 스택과 서비스 (Stack/Service)

❖ 블로그 App 실행 (예)

① **docker stack deploy --compose-file=ghost-stack.yml ghost-stack**

```
version: '3.1'
services:
  ghost:
    image: ghost:1-alpine
    restart: always
    ports:
      - 8585:2368
    environment:
      # see https://docs.ghost.org/docs/config#section-running-ghost-with-config-env-variables
      database__client: mysql
      database__connection__host: db
      database__connection__user: root
      database__connection__password: example
      database__connection__database: ghost
  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: example
```

메모:

- Check Local Host

4. 스택과 서비스 (Stack/Service)

❖ 서비스(service)를 위한 Manager/Worker 노드 추가

① **sudo docker swarm join-token manager**

② **sudo docker swarm join-token worker**

③ **sudo docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-7rukwukuz7kmgnt0s1klrq5o2 192.168.0.60:2377** # @ Manager

④ **sudo docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-7rukwukuz7kmgnt0s1klrq5o2 192.168.0.60:2377** # @ Worker

```
[root@kubemaster example-voting-app]# docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-
2a7m4ydly5j3hqgx7jdwyyasg 192.168.0.60:2377

[root@kubemaster example-voting-app]# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-
7rukwukuz7kmgnt0s1klrq5o2 192.168.0.60:2377

[root@kubemaster example-voting-app]#
```

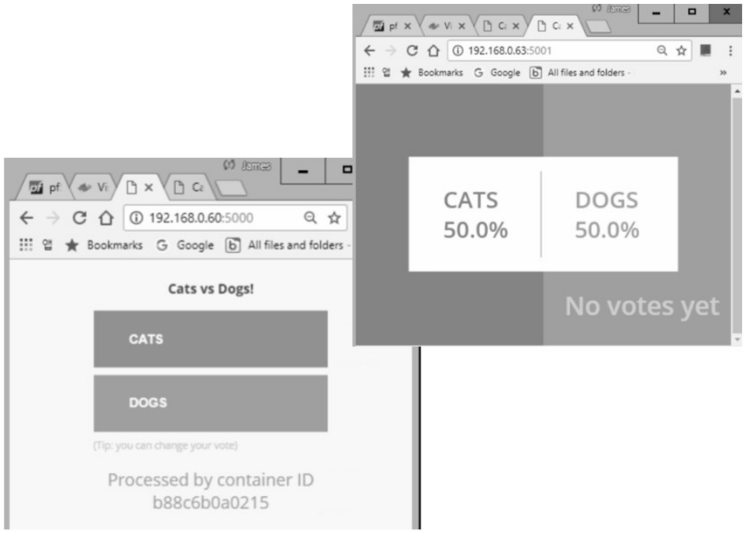
메모:

- 스웜(Swarm) 모드 지원 최신 Docker 버전 설치: curl -fsSL https://get.docker.com/ | sh
- usermod -aG docker root
- systemctl stop firewalld && systemctl disable firewalld
- systemctl enable docker && systemctl start docker

4. 스택과 서비스 (Stack/Service)

❖ 서비스 접속

① # http://192.168.0.60:8080 for Visualizer
② # http://192.168.0.60:5000 for vote
③ # http://192.168.0.60:5001 for result
④ # http://192.168.0.61:8080 for Visualizer
⑤ # http://192.168.0.61:5000 for vote
⑥ # http://192.168.0.61:5001 for result
⑦ # http://192.168.0.62:8080 for Visualizer
⑧ # http://192.168.0.62:5000 for vote
⑨ # http://192.168.0.62:5001 for result
⑩ # http://192.168.0.63:8080 for Visualizer
⑪ # http://192.168.0.63:5000 for vote
⑫ # http://192.168.0.63:5001 for result



메모:

- Routing mesh: Docker Engine swarm mode makes it easy to publish ports for services to make them available to resources outside the swarm. All nodes participate in an ingress routing mesh
- Port 7946 TCP/UDP 는 컨테이너 네트워크 발견(container network discovery)에 사용
- Port 4789 UDP 는 컨테이너 진입(Ingress) 네트워크(container ingress network)에 사용

부록: Docker

1. **컨테이너** (Docker..)
2. **이미지** (Docker Image)
3. **스웜** (Swarm)
4. **스택과 서비스** (Stack/Service)
5. **Container Networking** (Docker..)

5. Container Networking (Docker..)

❖ 도커 브릿지 (Docker Bridge)

① **sudo docker network**

② **sudo docker network ls**

③ **sudo docker network inspect bridge**

④ **sudo docker info**

⑤ **sudo docker network ls**

⑥ **sudo apt install bridge-utils**

⑦ **ip link show**

```
jslab@ubuntu70:~/example-voting-app$ brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242b7693044       no              veth30909d0
                                                        vethe398f5d
docker_gwbridge         8000.0242d7ebbbea       no              veth2716218
                                                        veth33d2404
                                                        veth3c97a9a
                                                        vetha8b0fad
                                                        vethf2a0658
jslab@ubuntu70:~/example-voting-app$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:1e:79:0b brd ff:ff:ff:ff:ff:ff
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 02:42:b7:69:30:44 brd ff:ff:ff:ff:ff:ff
25: veth30909d0@if24: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP mode DEFAULT group default
    link/ether 1a:9c:f0:55:0b:84 brd ff:ff:ff:ff:ff:ff link-netnsid 0
27: vethe398f5d@if26: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP mode DEFAULT group default
    link/ether ba:e5:0b:e5:68:b3 brd ff:ff:ff:ff:ff:ff link-netnsid 1
52: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 02:42:d7:eb:bb:ea brd ff:ff:ff:ff:ff:ff
54: veth33d2404@if53: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether 06:1b:14:0c:7b:8e brd ff:ff:ff:ff:ff:ff link-netnsid 3
135: veth2716218@if134: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether 32:0b:ee:24:65:eb brd ff:ff:ff:ff:ff:ff link-netnsid 5
139: vetha8b0fad@if138: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether 9a:85:16:7a:6e:2f brd ff:ff:ff:ff:ff:ff link-netnsid 6
157: vethf2a0658@if156: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether 5e:03:9f:bc:65:50 brd ff:ff:ff:ff:ff:ff link-netnsid 9
167: veth3c97a9a@if166: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
    link/ether ca:b9:70:2c:c9:4c brd ff:ff:ff:ff:ff:ff link-netnsid 10
jslab@ubuntu70:~/example-voting-app$
```

메모:

- The Basics @ CentOS

5. Container Networking (Docker..)

❖ 도커 브릿지 (Docker Bridge)

① **sudo docker run -dt ubuntu sleep infinity**

② **sudo docker ps**

③ **sudo brctl show**

```
[root@kubeworker1 ~]# docker run -dt ubuntu sleep infinity
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete
b78396653dae: Pull complete
Digest: sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6
Status: Downloaded newer image for ubuntu:latest
7d3800792767f454cdf79d485000a62f5ceb993ac1146df03f8a4f66c7a8f5d8
[root@kubeworker1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
7d3800792767        ubuntu              "sleep infinity"    13 seconds ago      Up 13 seconds
determined_wiles
[root@kubeworker1 ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.02426d0da0e5       no              veth7169caf
```

메모:

- 컨테이너 연결

❖ 도커 브릿지 (Docker Bridge)

④ docker network inspect bridge

```
[root@kubeworker1 ~]# docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "9d00fa54875a2fc19f0b782fbbc080de9e5b4b0899a38d1e9564db6b3e27aa52",
        "Created": "2018-04-04T03:00:12.771895121-04:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "7d3800792767f454cdf79d485000a62f5ceb993ac1146df03f8a4f66c7a8f5d8": {
                "Name": "determined_wiles",
                "EndpointID": "00c397c6642f38fcf3cdddf205c9a7a0c5ac3d34e894fa79672bfc5c6e228e99",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
[root@kubeworker1 ~]#
```

메모:

- 컨테이너 연결

5. Container Networking (Docker..)

❖ ‘docker network inspect ingress’ (도커 설치 후 확인)

```
james@masteratlocal:~$ sudo docker network inspect ingress
[
    {
        "Name": "ingress",
        "Id": "l1yxmoq9eeyt066f00dv3jkfy",
        "Created": "2018-04-09T22:31:55.942519097+09:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.255.0.0/16",
                    "Gateway": "10.255.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": true,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "ingress-sbox": {
                "Name": "ingress-endpoint",
                "EndpointID": "9dfbeb73b9d41cfd650a75132616072b329eb1e2267bd0923733a86285e86ca0",
                "MacAddress": "02:42:0a:ff:00:02",
                "IPv4Address": "10.255.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4096"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "b14075486730",
                "IP": "192.168.0.61"
            },
            {
                "Name": "e6a823a6f7fa",
                "IP": "192.168.33.61"
            }
        ]
    }
]
james@masteratlocal:~$
```

메모:

5. Container Networking (Docker..)

❖ Ping

① ping -c5 <IPv4 Address>
② sudo docker ps
③ sudo docker exec -it <CONTAINER ID> /bin/bash
④ apt-get update && apt-get install -y iputils-ping
⑤ exit

```
[root@kubeworker1 ~]# ping -c5 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.197 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.087 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.096 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.076 ms

--- 172.17.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.073/0.105/0.197/0.048 ms
[root@kubeworker1 ~]# ^C
[root@kubeworker1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
7d3800792767        ubuntu              "sleep infinity"    7 minutes ago       Up 7 minutes
determined_wiles
[root@kubeworker1 ~]# docker exec -it 7d /bin/bash
root@7d3800792767:/# apt-get update && apt-get install -y iputils-ping
```

메모:
- Ping

❖ Ping

⑥ apt-get update && apt-get install -y iputils-ping

```
[[root@kubeworker1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
7d3800792767        ubuntu              "sleep infinity"    7 minutes ago       Up 7 minutes
determined_wiles
[root@kubeworker1 ~]# docker exec -it 7d /bin/bash
root@7d3800792767:/# apt-get update && apt-get install -y iputils-ping
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
...
...
...
Setting up libffi6:amd64 (3.2.1-4) ...
Setting up libp11-kit0:amd64 (0.23.2-5~ubuntu16.04.1) ...
Setting up libtasn1-6:amd64 (4.7-3ubuntu0.16.04.3) ...
Setting up libgnutls30:amd64 (3.4.10-4ubuntu1.4) ...
Setting up libgnutls-openssl27:amd64 (3.4.10-4ubuntu1.4) ...
Setting up iputils-ping (3:20121221-5ubuntu2) ...
Setcap is not installed, falling back to setuid
Processing triggers for libc-bin (2.23-0ubuntu10) ...
root@7d3800792767:/#
```

메모:

- A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!

5. Container Networking (Docker..)

❖ Ping

⑦ **exit**

⑧ **sudo docker ps**

⑨ **sudo docker stop <CONTAINER ID>**

```
root@7d3800792767:/# exit
exit
[root@kubeworker1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
7d3800792767        ubuntu              "sleep infinity"    12 minutes ago      Up 12 minutes
determined_wiles
[root@kubeworker1 ~]# docker stop 7d
7d
```

메모:

- <CONTAINER ID> 는 다른 컨테이너와 겹치지 않는 ID 앞부분 1글자 이상이면 가능

5. Container Networking (Docker..)

❖ 외부 연결을 위한 NAT 구성

① **sudo docker run --name web1 -d -p 8080:80 nginx**
② **sudo docker ps**
③ **sudo curl 127.0.0.1:8080**

```
[root@kubeworker1 ~]# docker run —name web1 -d -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
2a72cbf407d6: Pull complete
e19f9e910af9: Pull complete
2f3d26a87e79: Pull complete
Digest: sha256:d0468eaec1ef818af05f85ac00e484fd5a2ae75dd567dc9f7ccf5f68a60351fb
Status: Downloaded newer image for nginx:latest
06082a1850464843a3d0ac641c77816e8f3b7a5d8a363bc7016c9cd81bef34a1
[root@kubeworker1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
06082a185046        nginx               "nginx -g 'daemon of…"   13 seconds ago      Up 12 seconds       0.0.0.0:8080-
>80/tcp   web1
[root@kubeworker1 ~]# curl 127.0.0.1:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@kubeworker1 ~]#
```

메모:

- curl: command lines or scripts to transfer data. It is also used in cars, television sets, routers, printers, audio equipment, mobile phones, tablets, settop boxes, media players and is the internet transfer backbone for thousands of software applications.
- curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, HTTP/2, cookies, user+password authentication (Basic, Plain, Digest, CRAM-MD5, NTLM, Negotiate and Kerberos), file transfer, proxy tunneling and more.

❖ 외부 연결을 위한 NAT 구성

④ **http://192.168.0.61:8080**

메모:

- 외부 연결을 위한 NAT 구성

❖ 오버레이(Overlay) 연결을 위한 구성

① **sudo docker swarm init --advertise-addr $(hostname -i)**

@ Manager

② **sudo docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-7rukwukuz7kmgnt0s1klrq5o2 192.168.0.60:2377**

@ Worker

```
[root@kubemaster ~]# docker swarm init --advertise-addr $(hostname -i)
Swarm initialized: current node (l9e8wqyjw00ogjl092n0eyymr) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-
7rukwukuz7kmgnt0s1klrq5o2 192.168.0.60:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the
instructions.

[root@kubemaster ~]#
```

```
[root@kubeworker1 ~]#  docker swarm join --token SWMTKN-1-
3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-7rukwukuz7kmgnt0s1klrq5o2 192.168.0.60:2377
This node joined a swarm as a worker.
[root@kubeworker1 ~]#
```

```
[[root@kubemaster ~]# docker node ls
ID                          HOSTNAME          STATUS            AVAILABILITY      MANAGER STATUS      ENGINE
VERSION
l9e8wqyjw00ogjl092n0eyymr *  kubemaster        Ready             Active            Leader              18.03.0-ce
kb55f7sda5mduimloa2o5a9vx    kubeworker1       Ready             Active                                18.03.0-ce
[root@kubemaster ~]#
```

메모:

- Overlay Networking

❖ 오버레이(Overlay) 연결을 위한 구성

④ **sudo docker network create -d overlay overnet**

⑤ **sudo docker network ls**

```
[root@kubemaster ~]# docker network create -d overlay overnet
2n20w14b1ggir4ie2dok2tagz
[root@kubemaster ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
07476b48b3b6        bridge              bridge              local
05191e8b7e19        docker_gwbridge     bridge              local
06322c05f69e        host                host                local
mt37ijy3elpt        ingress             overlay             swarm
ed53abe4e032        none                null                local
2n20w14b1ggi        overnet             overlay             swarm
```

메모:

- Overlay Networking

❖ 오버레이(Overlay) 연결을 위한 구성

⑥ **docker network create -d overlay overnet**

⑦ **docker network inspect overnet**

```
[root@kubemaster ~]# docker network inspect overnet
[
    {
        "Name": "overnet",
        "Id": "2n2Ow14b1ggir4ie2dok2tagz",
        "Created": "2018-04-04T07:48:55.65703066Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": []
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": null,
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": null
    }
]
[root@kubemaster ~]#
```

메모:

- Overlay Networking

5. Container Networking (Docker..)

❖ 오버레이(Overlay) 연결을 위한 구성

① **sudo docker network create -d overlay overnet**

② **sudo docker service create --name myservice **

--network overnet \

--replicas 2 \

ubuntu sleep infinity

③ **sudo docker service ps myservice**

④ **sudo docker network ls**

```
[root@kubemaster ~]# docker service create —name myservice ¥
> —network overnet ¥
> —replicas 2 ¥
> ubuntu sleep infinity
3nzzhjmsoglebijq01y8w0mfu
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
verify: Service converged
[root@kubemaster ~]# docker service ls
ID              NAME            MODE            REPLICAS        IMAGE           PORTS
3nzzhjmsogle    myservice       replicated      2/2             ubuntu:latest
[root@kubemaster ~]#
[root@kubemaster ~]# docker service ps myservice
ID              NAME            IMAGE           NODE            DESIRED STATE   CURRENT STATE
ERROR           PORTS
3rnwogesfguo    myservice.1     ubuntu:latest   kubeworker1     Running         Running about a minute
ago
qqxmz9cl72rb    myservice.2     ubuntu:latest   kubemaster      Running         Running about a minute
ago
[root@kubemaster ~]#
[root@kubemaster ~]# docker network ls
NETWORK ID      NAME            DRIVER          SCOPE
07476b48b3b6    bridge          bridge          local
05191e8b7e19    docker_gwbridge bridge          local
06322c05f69e    host            host            local
mt37ijy3elpt    ingress         overlay         swarm
ed53abe4e032    none            null            local
2n20w14b1ggi    overnet         overlay         swarm
```

메모:

- 서비스 (Service) 생성

5. Container Networking (Docker..)

❖ sudo iptables -t nat -L -n # 도커에서 생성한 NAT 확인

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ sudo iptables -t nat -L -n
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  0.0.0.0/0            0.0.0.0/0            ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  0.0.0.0/0           !127.0.0.0/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.18.0.0/16        0.0.0.0/0
MASQUERADE  all  --  172.17.0.0/16        0.0.0.0/0
MASQUERADE  tcp  --  172.18.0.2           172.18.0.2           tcp dpt:7053
MASQUERADE  tcp  --  172.18.0.2           172.18.0.2           tcp dpt:7051
MASQUERADE  tcp  --  172.18.0.3           172.18.0.3           tcp dpt:7053
MASQUERADE  tcp  --  172.18.0.3           172.18.0.3           tcp dpt:7051
MASQUERADE  tcp  --  172.18.0.4           172.18.0.4           tcp dpt:7053
MASQUERADE  tcp  --  172.18.0.4           172.18.0.4           tcp dpt:7051
MASQUERADE  tcp  --  172.18.0.5           172.18.0.5           tcp dpt:7053
MASQUERADE  tcp  --  172.18.0.5           172.18.0.5           tcp dpt:7051
MASQUERADE  tcp  --  172.18.0.6           172.18.0.6           tcp dpt:7050

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:8053 to:172.18.0.2:7053
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:8051 to:172.18.0.2:7051
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:9053 to:172.18.0.3:7053
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:9051 to:172.18.0.3:7051
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:10053 to:172.18.0.4:7053
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:10051 to:172.18.0.4:7051
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:7053 to:172.18.0.5:7053
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:7051 to:172.18.0.5:7051
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:7050 to:172.18.0.6:7050
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

메모:

- Hyperledger Fabric

❖ ifconfig

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ ifconfig
br-2813649789ee Link encap:Ethernet  HWaddr 02:42:52:b5:7b:fc
          inet addr:172.18.0.1  Bcast:172.18.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:52ff:feb5:7bfc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:55 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:448 (448.0 B)  TX bytes:6548 (6.5 KB)

docker0   Link encap:Ethernet  HWaddr 02:42:40:02:84:ad
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:40ff:fe02:84ad/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:1193 (1.1 KB)

ens33     Link encap:Ethernet  HWaddr 00:0c:29:04:6f:d8
          inet addr:192.168.52.129  Bcast:192.168.52.255
Mask:255.255.255.0
          inet6 addr: fe80::f3b5:51eb:563f:dc41/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:575324 errors:0 dropped:0 overruns:0 frame:0
          TX packets:136202 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:864390894 (864.3 MB)  TX bytes:8768964 (8.7 MB)

ens34     Link encap:Ethernet  HWaddr 00:0c:29:04:6f:e2
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1144 (1.1 KB)  TX bytes:7515 (7.5 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:960 errors:0 dropped:0 overruns:0 frame:0
          TX packets:960 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:103681 (103.6 KB)  TX bytes:103681 (103.6 KB)

veth7820612 Link encap:Ethernet  HWaddr 62:d7:5b:d0:ac:36
          inet6 addr: fe80::60d7:5bff:fed0:ac36/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:78 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4486 (4.4 KB)  TX bytes:9393 (9.3 KB)

veth02bb183 Link encap:Ethernet  HWaddr f2:21:d9:80:36:fd
          inet6 addr: fe80::f021:d9ff:fe80:36fd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15159 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15256 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2762725 (2.7 MB)  TX bytes:2764978 (2.7 MB)
```

```
veth30e0c2a Link encap:Ethernet  HWaddr 1e:dc:d2:ba:25:52
          inet6 addr: fe80::1cdc:d2ff:feba:2552/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14954 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15286 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2781766 (2.7 MB)  TX bytes:2795729 (2.7 MB)

veth37ebbe7 Link encap:Ethernet  HWaddr b2:e8:fc:49:14:11
          inet6 addr: fe80::b0e8:fcff:fe49:1411/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:65 errors:0 dropped:0 overruns:0 frame:0
          TX packets:107 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7065 (7.0 KB)  TX bytes:13824 (13.8 KB)

veth8c2499d Link encap:Ethernet  HWaddr ca:23:30:1c:89:ab
          inet6 addr: fe80::c823:30ff:fe1c:89ab/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15169 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15201 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2709149 (2.7 MB)  TX bytes:2793392 (2.7 MB)

veth975c432 Link encap:Ethernet  HWaddr fa:83:8e:75:a6:d7
          inet6 addr: fe80::f883:8eff:fe75:a6d7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14991 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14880 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2673221 (2.6 MB)  TX bytes:2755827 (2.7 MB)

veth9f514d7 Link encap:Ethernet  HWaddr d2:78:2c:57:91:6a
          inet6 addr: fe80::d078:2cff:fe57:916a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:217 errors:0 dropped:0 overruns:0 frame:0
          TX packets:344 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:137964 (137.9 KB)  TX bytes:56184 (56.1 KB)

vethbb26a41 Link encap:Ethernet  HWaddr 76:57:33:dc:26:d6
          inet6 addr: fe80::7457:33ff:fedc:26d6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:408 errors:0 dropped:0 overruns:0 frame:0
          TX packets:431 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:144171 (144.1 KB)  TX bytes:91507 (91.5 KB)

vethc9f7641 Link encap:Ethernet  HWaddr 9a:09:cf:75:d7:50
          inet6 addr: fe80::9809:cfff:fe75:d750/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:62 errors:0 dropped:0 overruns:0 frame:0
          TX packets:101 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6717 (6.7 KB)  TX bytes:13075 (13.0 KB)

jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

메모:

- http://hyperledger-fabric.readthedocs.io/en/release-1.1/samples.html#binaries

❖ ip route

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ ip route
default via 192.168.52.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1 linkdown
172.18.0.0/16 dev br-2813649789ee  proto kernel  scope link  src 172.18.0.1
192.168.52.0/24 dev ens33  proto kernel  scope link  src 192.168.52.129  metric 100
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

메모:

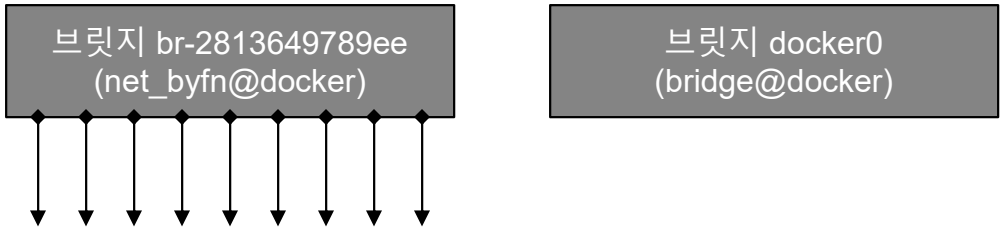
5. Container Networking (Docker..)

❖ sudo docker network ls & brctl show

① **sudo apt install bridge-utils**

② **sudo docker network ls & brctl show**

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
2cc6ad351481        bridge              bridge              local
cc3648572554        host                host                local
2813649789ee        net_byfn            bridge              local
f3ac7c07b82c        none                null                local
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ sudo brctl show
bridge name         bridge id               STP enabled     interfaces
br-2813649789ee     8000.024252b57bfc       no              veth02bb183
                                                            veth30e0c2a
                                                            veth37ebbe7
                                                            veth7820612
                                                            veth8c2499d
                                                            veth975c432
                                                            veth9f514d7
                                                            vethbb26a41
                                                            vethc9f7641
docker0             8000.0242400284ad       no
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```




```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ ip route
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1 linkdown
172.18.0.0/16 dev br-2813649789ee  proto kernel  scope link  src 172.18.0.1
```

메모:

james@jslab.kr

❖ brctl showmacs br-2813649789ee

① brctl showmacs br-2813649789ee

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ brctl showmacs br-2813649789ee
port no mac addr                is local?       ageing timer
  1     02:42:ac:12:00:02       no                 0.18
  2     02:42:ac:12:00:03       no                 0.23
  3     02:42:ac:12:00:04       no                 0.23
  4     02:42:ac:12:00:05       no                 0.23
  5     02:42:ac:12:00:06       no                62.58
  7     02:42:ac:12:00:08       no                38.26
  8     02:42:ac:12:00:09       no                20.85
  9     02:42:ac:12:00:0a       no                 3.18
  4     1e:dc:d2:ba:25:52       yes                0.00
  4     1e:dc:d2:ba:25:52       yes                0.00
  9     62:d7:5b:d0:ac:36       yes                0.00
  9     62:d7:5b:d0:ac:36       yes                0.00
  6     76:57:33:dc:26:d6       yes                0.00
  6     76:57:33:dc:26:d6       yes                0.00
  8     9a:09:cf:75:d7:50       yes                0.00
  8     9a:09:cf:75:d7:50       yes                0.00
  7     b2:e8:fc:49:14:11       yes                0.00
  7     b2:e8:fc:49:14:11       yes                0.00
  2     ca:23:30:1c:89:ab       yes                0.00
  2     ca:23:30:1c:89:ab       yes                0.00
  5     d2:78:2c:57:91:6a       yes                0.00
  5     d2:78:2c:57:91:6a       yes                0.00
  3     f2:21:d9:80:36:fd       yes                0.00
  3     f2:21:d9:80:36:fd       yes                0.00
  1     fa:83:8e:75:a6:d7       yes                0.00
  1     fa:83:8e:75:a6:d7       yes                0.00
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

메모:

5. Container Networking (Docker..)

❖ sudo virsh net-list --all

① sudo apt-get install libvirt-bin

② sudo virsh net-list --all

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ sudo virsh net-list --all
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     yes           yes

jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

메모:

- The libvirt project: is a toolkit to manage virtualization platforms

❖ sudo docker network inspect bridge

① sudo docker network inspect bridge

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ sudo docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "2cc6ad351481d6c6fc91bb106eda985e3e6f9c256ac7faf4c1c87094e9ce3bd6",
        "Created": "2018-07-04T21:51:46.258574047+09:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

메모:

5. Container Networking (Docker..)

❖ sudo docker image inspect onosproject/onos

```
sdn@sdn:~$ sudo docker image inspect onosproject/onos
[
    {
        "Id": "sha256:ec5d8befbddf5846d2713c986b5e9e9ec812ecc5dec269a643bbe89d4bb099e2",
        "RepoTags": [
            "onosproject/onos:latest"
        ],
        "RepoDigests": [
            "onosproject/onos@sha256:c9153ec30673500315aa8685e6bf5f72ca0c5f24c7857e7c46364382b4(
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2018-10-11T15:05:57.706099077Z",
        "Container": "fbdcef50a024a7b2cfb0d2a4b850a00fb1337a607914db4106b8613789b750af",
        "ContainerConfig": {
            "Hostname": "fbdcef50a024",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "ExposedPorts": {
                "6640/tcp": {},
                "6653/tcp": {},
                "8101/tcp": {},
                "8181/tcp": {},
                "9876/tcp": {}
            },
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/jdk/bin'
                "JAVA_VERSION_MAJOR=8",
                "JAVA_VERSION_MINOR=181",
                "JAVA_VERSION_BUILD=13",
                "JAVA_PACKAGE=server-jre",
                "JAVA_JCE=standard",
                "JAVA_HOME=/opt/jdk",
                "GLIBC_REPO=https://github.com/sgerrand/alpine-pkg-glibc",
                "GLIBC_VERSION=2.28-r0",
                "LANG=C.UTF-8"
            ],
            "Cmd": [
                "/bin/sh",
                "-c",
                "#(nop) ",
                "CMD [¥"server¥"]"
            ],
            "ArgsEscaped": true,
            "Image": "sha256:6b9683aeb06f0e77919304651719e237d86b59d36ac7f7df5ae243eb43d81b9a",
            "Volumes": null,
            "WorkingDir": "/root/onos",
            "Entrypoint": [
                "./bin/onos-service"
            ],
            "OnBuild": null,
            "Labels": {
                "org.label-schema.description": "SDN Controller",
                "org.label-schema.name": "ONOS",
                "org.label-schema.schema-version": "1.0",
                "org.label-schema.url": "http://onosproject.org",
                "org.label-schema.usage": "http://wiki.onosproject.org",
                "org.label-scheme.vendor": "Open Networking Foundation"
            }
        },
        "DockerVersion": "18.03.1-ee-1-tp5",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "ExposedPorts": {
                "6640/tcp": {},
                "6653/tcp": {},
                "8101/tcp": {},
                "8181/tcp": {},
                "9876/tcp": {}
            },
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/jdk/bin",
                "JAVA_VERSION_MAJOR=8",
                "JAVA_VERSION_MINOR=181",
                "JAVA_VERSION_BUILD=13",
                "JAVA_PACKAGE=server-jre",
                "JAVA_JCE=standard",
                "JAVA_HOME=/opt/jdk",
                "GLIBC_REPO=https://github.com/sgerrand/alpine-pkg-glibc",
                "GLIBC_VERSION=2.28-r0",
                "LANG=C.UTF-8"
            ],
            "Cmd": [
                "server"
            ],
            "ArgsEscaped": true,
            "Image": "sha256:6b9683aeb06f0e77919304651719e237d86b59d36ac7f7df5ae243eb43d81b9a",
            "Volumes": null,
            "WorkingDir": "/root/onos",
            "Entrypoint": [
                "./bin/onos-service"
            ],
            "OnBuild": null,
            "Labels": {
                "org.label-schema.description": "SDN Controller",
                "org.label-schema.name": "ONOS",
                "org.label-schema.schema-version": "1.0",
                "org.label-schema.url": "http://onosproject.org",
                "org.label-schema.usage": "http://wiki.onosproject.org",
                "org.label-scheme.vendor": "Open Networking Foundation"
            }
        },
        "Architecture": "amd64",
        "Os": "linux",
        "Size": 476856426,
        "VirtualSize": 476856426,
        "GraphDriver": {
            "Data": {
                "LowerDir":
"/var/lib/docker/overlay2/6e162c562b6974d4c4880477b570d5b5e066052d56e64eeafb30a45d34527f42/diff:/var/l
ib/docker/overlay2/d5da7cb928321e61a9810097e2eea9ee8d5099f170908a04f3f82574db685a4b/diff:/var/lib/dock
er/overlay2/67b2b2dfc8ceb60636fed430674aa632d289af15ba7e03f9879632141ff2979f/diff:/var/lib/docker/over
lay2/798b3a994c1c063515f8137d727a15950ac6d8d76baaad2e2e3f79c124b0b31d/diff",
                "MergedDir":
"/var/lib/docker/overlay2/a207fce93e8cd455d657800a1dbbab8714fd3d925a79eedfb0c79b2cc43a96f4/merged",
                "UpperDir":
"/var/lib/docker/overlay2/a207fce93e8cd455d657800a1dbbab8714fd3d925a79eedfb0c79b2cc43a96f4/diff",
                "WorkDir":
"/var/lib/docker/overlay2/a207fce93e8cd455d657800a1dbbab8714fd3d925a79eedfb0c79b2cc43a96f4/work"
            },
            "Name": "overlay2"
        },
        "RootFS": {
            "Type": "layers",
            "Layers": [
                "sha256:ebf12965380b39889c99a9c02e82ba465f887b45975b6e389d42e9e6a3857888",
                "sha256:f81d498bb79e7f680c98171ecb36651ff959610646aa1092e195a58c25fc7918",
                "sha256:478189a15cc48c5cff88a91b01034c5ad9afd78b5cdcd4b01687afcb378faac4",
                "sha256:6de6049b450a58e07ccb96b23670a1343e2c5e4437fc6426e165a0bd7065ed81",
                "sha256:5c681a837481e881c103e2d4d1b30822dfb00da49e027a6dcebb3776f4f68ab2"
            ]
        },
        "Metadata": {
            "LastTagTime": "0001-01-01T00:00:00Z"
        }
    }
]
```

메모:

5. Container Networking (Docker..)

❖ Hyperledger: vi docker-compose-cli.yaml

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#

version: '2'

volumes:
  orderer.example.com:
  peer0.org1.example.com:
  peer1.org1.example.com:
  peer0.org2.example.com:
  peer1.org2.example.com:

networks:
  byfn:

services:

  orderer.example.com:
    extends:
      file:   base/docker-compose-base.yaml
      service: orderer.example.com
    container_name: orderer.example.com
    networks:
      - byfn

  peer0.org1.example.com:
    container_name: peer0.org1.example.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.org1.example.com
    networks:
      - byfn

  peer1.org1.example.com:
    container_name: peer1.org1.example.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer1.org1.example.com
    networks:
      - byfn

  peer0.org2.example.com:
    container_name: peer0.org2.example.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.org2.example.com
    networks:
      - byfn

  peer1.org2.example.com:
    container_name: peer1.org2.example.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer1.org2.example.com
    networks:
      - byfn
```

```
  cli:
     container_name: cli
     image: hyperledger/fabric-tools:$IMAGE_TAG
     tty: true
     stdin_open: true
     environment:
       - GOPATH=/opt/gopath
       - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
       #- CORE_LOGGING_LEVEL=DEBUG
       - CORE_LOGGING_LEVEL=INFO
       - CORE_PEER_ID=cli
       - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
       - CORE_PEER_LOCALMSPID=Org1MSP
       - CORE_PEER_TLS_ENABLED=true
       -
CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/
crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/
server.crt
       -
CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/c
rypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/s
erver.key
       -
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/p
eer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/
tls/ca.crt
       -
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/
crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
     working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
     command: /bin/bash
     volumes:
         - /var/run/:/host/var/run/
         - ./../chaincode/:/opt/gopath/src/github.com/chaincode
         - ./crypto-
config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
         -
./scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scripts/
         - ./channel-
artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-
artifacts
     depends_on:
       - orderer.example.com
       - peer0.org1.example.com
       - peer1.org1.example.com
       - peer0.org2.example.com
       - peer1.org2.example.com
     networks:
       - byfn
```

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ dir
base     channel-artifacts  crypto-config       docker-compose-cli.yaml         docker-compose-
couch.yaml          docker-compose-e2e.yaml    eyfn.sh          README.md
byfn.sh  configtx.yaml      crypto-config.yaml  docker-compose-couch-org3.yaml  docker-compose-e2e-
template.yaml  docker-compose-org3.yaml  org3-artifacts  scripts
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

메모: