


# Opensource Networking




엔터프라이즈 시스템/네트워크 운영자 대상  
**(for IT Pros and System Administrators)**

**JS Lab**


안종석  
james@jslab.kr

2019년 10월


# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

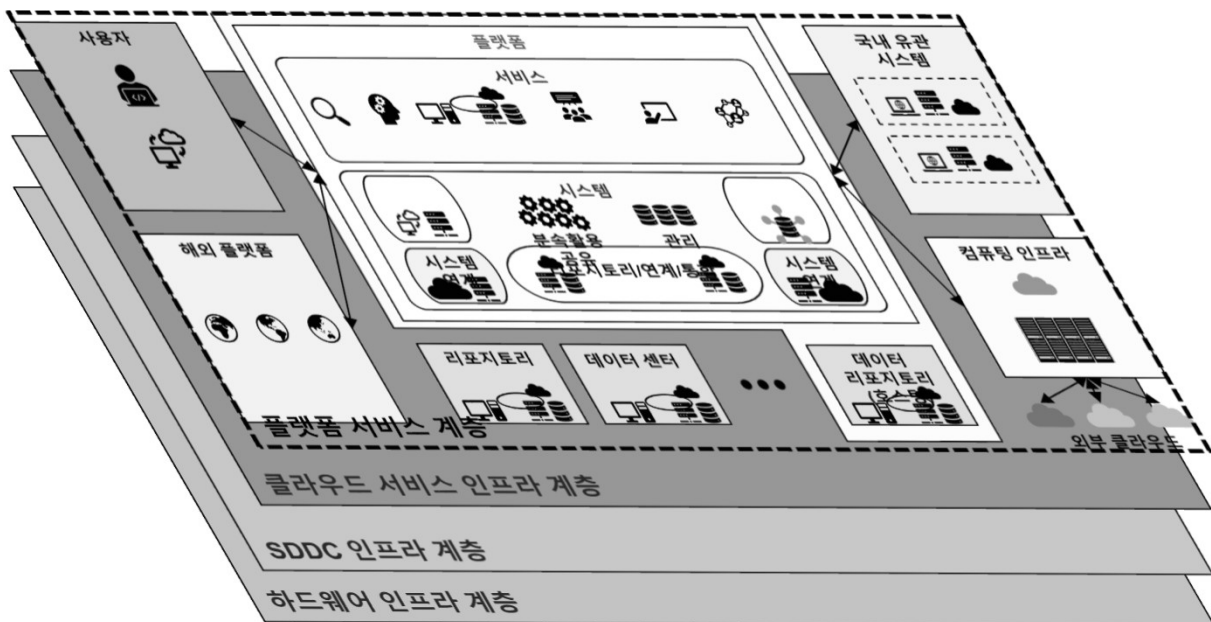


# I. 실습 환경

## ❖ 실습 환경 개요

### ▪ 각 계층별 네트워크 연계

- ✓ 하드웨어 계층: 서버, 네트워크, 스토리지 등
- ✓ 하드웨어 추상화 계층: SDDC(Software Defined Data Center)
- ✓ 클라우드 서비스 인프라 계층: Cloud Native
- ✓ 서비스: App



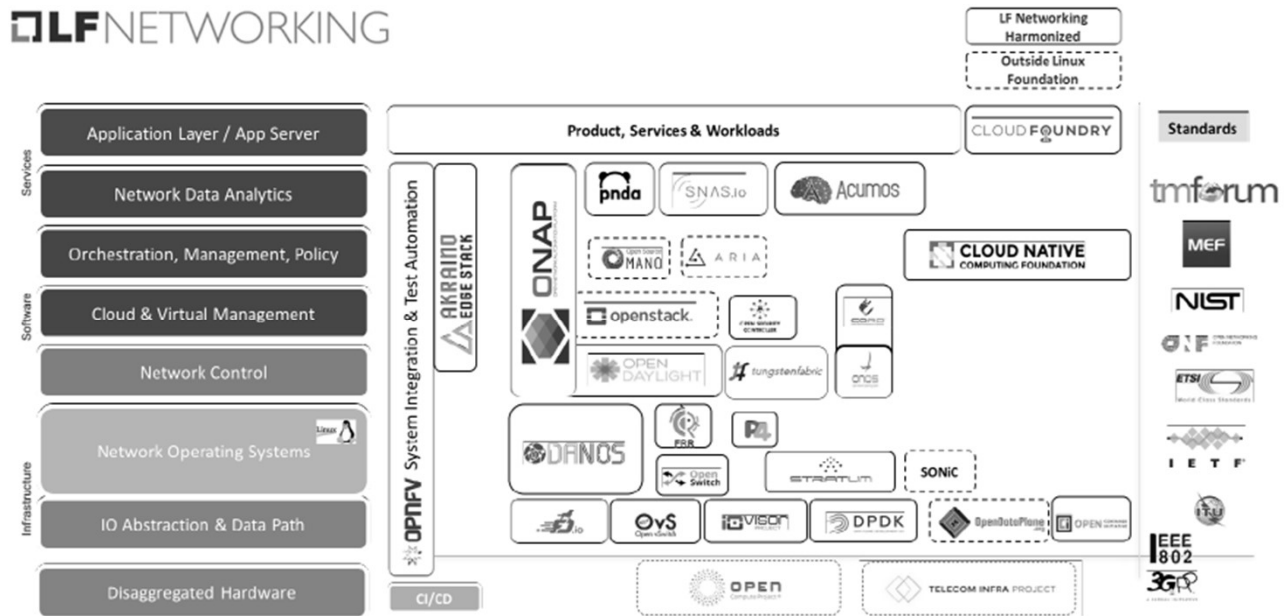
### 메모:

- 각 계층별 대표적인 Networking Opensource를 사용 실습
  - ✓ 가상스위치
  - ✓ 하이퍼바이저
  - ✓ 오케스트레이터, 서비스 메시
  - ✓ 각 네트워킹 오픈소스 프로젝트

# I. 실습 환경

## ❖ 실습 환경 개요

- **Open Networking - 계층구조** (리눅스재단의 오픈소스네트워킹)
  - ✓ **Disaggregated Hardware**
  - ✓ **IO Abstraction and Datapath**
  - ✓ **Network Operating Systems**
  - ✓ **Network Control**
  - ✓ **Network Virtualization**
  - ✓ **Cloud and Virtual Management**
  - ✓ **Orchestration, Management, Policy**
  - ✓ **Network Data Analytics**
  - ✓ **Application Layer.**



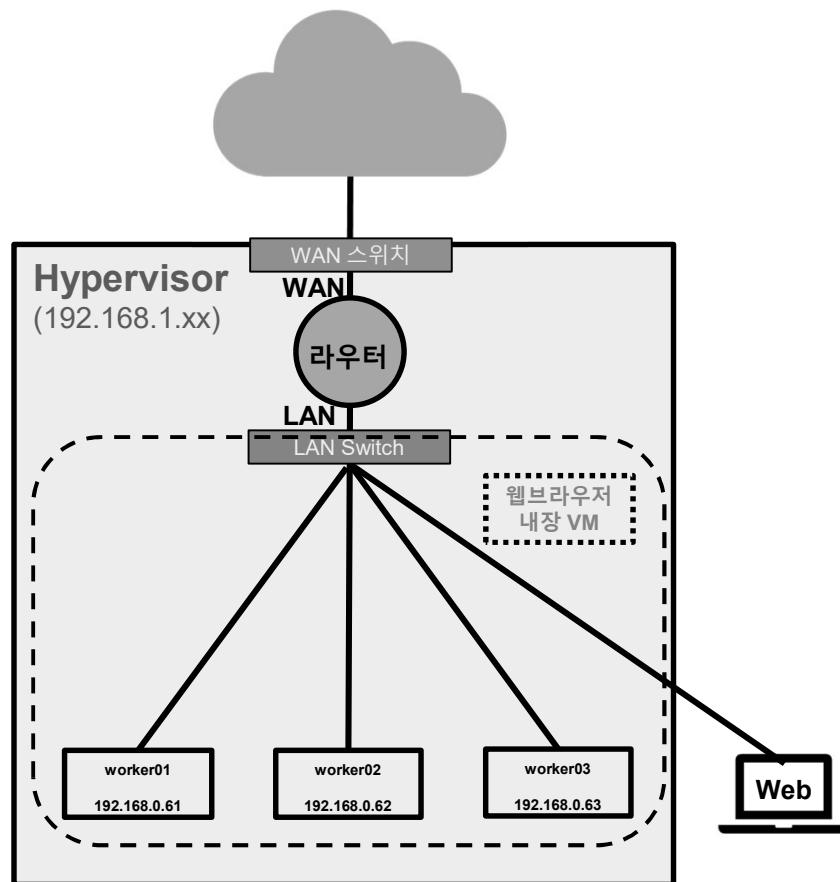
### 메모:

- Hybrid Cloud, Telco Cloud는 클라우드 네트워킹의 물리적 연계 이해가 필요
- 리눅스 재단의 오픈소스 네트워킹

# I. 실습 환경

## ❖ 실습 환경 개요

- 외부(WAN) IP 주소 1개 할당
- 내부(LAN)은 클러스터링 복수 호스트 연결 가상스위치 생성
- 설정을 위한 클라이언트는 VM 또는 유선랜 연결 PC 사용



### 메모:

- VM 시용시 VMRC 설치: VMware-VMRC-10.0.4-11818843
- 스위치는 2개 사용 (WAN/LAN)
- 하이퍼바이저 접속 <http://192.168.1.xy>
- 계정: ID / Password ( root / JSlab123)

# I. 실습 환경

## ❖ 사용 가능 소프트웨어

### ① Linux OS (Bare Metal 설치 Lab 환경 구성 고려)

- Fedora 또는 CentOS
- Ubuntu 또는 Debian
- Open Network Linux ( <https://opennetlinux.org/> )
- 기타

### ② Hardware 고려

- Intel 기반
- ARM 기반

### ③ 하이퍼바이저 기반 가상 네트워크 소프트웨어

- 가상화 보안 어플라이언스 (방화벽, IDS, SIEM등)
- 가상화 네트워크 어플라이언스 (라우터, SDN 제어기등)

OS	Packaging Tools	기타
Ubuntu	debian packaging (* .deb → apt-get install)	Debian
Fedora, CentOS	redhat packaging (.rpm → yum(dnf) install)	RHEL
Open Network Linux	nos-install-image (onie install)	Accton(7), Agema(1), Alpha Network(2), Dell(2), Penguin(3), Quanta(3)

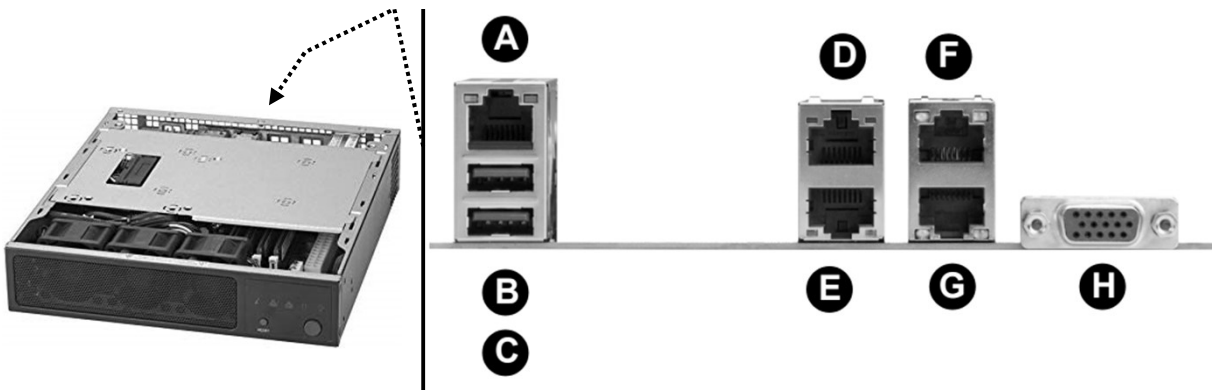
#### 메모:

- Current ONIE Hardware Status:  
[http://www.opencompute.org/wiki/Networking/ONIE/HW\\_Status](http://www.opencompute.org/wiki/Networking/ONIE/HW_Status)

# I. 실습 환경

## ❖ 하드웨어

- ① CPU w/Passive CPU heat sink
  - Intel® Xeon® processor D-1528
  - FCBGA 1667
  - CPU TDP support 35W, 9MB, 6 Cores, 12 Threads, 1.9-2.2GHz
- ② RAM
- ③ SSD
- ④ IPMI 2.0
- ⑤ 10GbE 2포트, 1 GbE LAN 2포트, IPMI 2.0 전용 LAN
- ⑥ SR-IOV (Single-Root Virtualization)



Back Panel I/O			
A	IPMI LAN	E	LAN Port 1 (-F, -LN2F, -TLN4F)
B	USB Port 1	F	LAN Port 4 (-TLN2F and -TLN4F)
C	USB Port 0	G	LAN Port 3 (-TLN2F and -TLN4F)
D	LAN Port 2 (-F, -LN2F, -TLN4F)	H	VGA Port

### 메모:

- Low noise fan speed control
- SR-IOV(Single Root I/O Virtualization): 시스템에서 여러 파티션이 동시에 실행되어 PCIe 장치를 공유할 수 있도록 PCI3 확장 스펙 정의 PCIe(Peripheral Component Interconnect express) 표준 아키텍처이며, 가상 함수(VF)로 알려진 PCI 함수의 가상 복제본을 정의함. 파티션에서 하이퍼바이저 또는 Virtual I/O Server 등을 거치지 않고도 SR-IOV 어댑터에 직접 연결할 수 있어 시간이 적고 CPU 이용률 감소

# I. 실습 환경

## ❖ 하이퍼바이저 설치 @ KOREN AI Network Lab

- ① Initial Powering Up (w/o Internet)
- ② USB booting Available
- ③ Alt-Ctrl-D로 Rebooting 하여 install 가능
- ④ Rebooting 시 'F11'에서 USB Booting 선택 (예: SanDisk)
- ⑤ ESXi '6.x' (원격콘솔 VMRC 사용)
- ⑥ Windows Server 2016 Hyper-v (선택)
- ⑦ 개인용 노트북 사용 (PDF 뷰어, Putty, WEB 브라우저, Software)



### 메모:

- ESXi 다운로드 주소: <https://my.vmware.com/en/web/vmware/evalcenter?p=free-esxi6>
- 디스크 이미지 굽기: Rufus 도구 사용 <https://rufus.akeo.ie/>
- Disk Imager <https://sourceforge.net/projects/win32diskimager/files/latest/download>
- USB 부팅 제가동은 전원 off/on (전원 케이블 포함)필요함

# I. 실습 환경

## ❖ 실습 구성 @ KOREN AI Network Lab

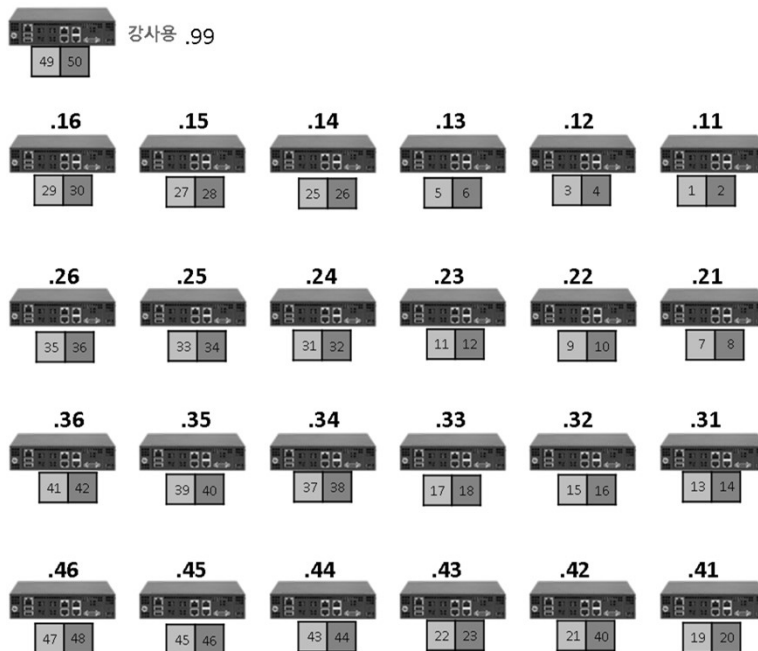
### ① USB 메모리

- OS
- 소프트웨어 도구 (Software Tools)

### ② IPMI 연결 이더넷 케이블

### ③ 인터넷 연결 케이블

### ④ 좌석 번호 별 서버넷의 해당 IP주소(x.x.x.nn) 설정 사용



#### 메모:

- 하이퍼바이저 설치 환경은 구조 분석을 고려하여 도구를 접속하여 미러링 가능한 네트워크 구성이 가능해야 함.
- 가상스위치는 분석을 위한 무작위(Promiscuous) 모드 설정 고려
- IPMI: Intelligent Platform Management Interface

# I. 실습 환경

## ❖ 하이퍼바이저 비교

- ① Microsoft의 Hyper-v는 평가기간 무제한
- ② vSphere 6.x 평가판은 60일간 모든 기능 제공하며, 평가 기간 종료 후에 상용기능 정지
- ③ 하이퍼바이저 사용 실습에서는 개인 노트북 필요 (LAN/웹브라우저/PDF뷰어 지원)

제품 기능	Microsoft	VMware vSphere 6.x		
	Hyper-V 2016	Free Hypervisor	Essential Plus	Enterprise Plus
VM 호스트 라이브 마이그레이션	Yes	No	Yes	Yes
VM 스토리지 라이브 마이그레이션	Yes	No	No	Yes
스토리지/네트워크 QoS	Yes	No (just disk shares)	No (just disk shares at host level)	Yes
하드웨어 패스드루	Discrete Device Assignment	PCI VM Direct Path USB redirection	PCI VM Direct Path USB redirection	PCI VM Direct Path USB redirection
운영 중 추가	Disks/vNIC/RAM	Disks/vNIC/USB	Disks/vNIC/USB	Disks/vNIC/USB/ CPU/RAM
운영 중 제거	Disks/vNIC/RAM	Disks/vNIC/USB	Disks/vNIC/USB	Disks/vNIC/USB/CPU
디스크 사이즈 조정	Hot-grow and shrink	Hot-grow	Hot-grow	Hot-grow
VM 암호화	Yes	No	No?	Yes

### 메모:













- Type 2 Hypervisor는 VMware (WorkStation) Player 또는 VirtualBox 사용 가능
- 노트북 미 지참 실습은 베어메탈 서버에 리눅스 또는 윈도우 OS 설치 (USB 허브 필요)



# I. 실습 환경

## ❖ 사용 소프트웨어

- ① 실습에 ESXi 6.7 사용시 웹 접속 또는 원격콘솔 VMRC 또는 VMware Player 사용
- ② Multi-host 클러스터링 환경을 위해 Super-putty (Multi Viewer)사용
- ③ Home Lab 구성에 사용 할 수 있는 소프트웨어 포함

이름	크기
 Xming-6-9-0-31-setup.exe	2,154KB
 X Window on Windows 1.20 setup-x86_64.e...	1,197KB
 WinSCP-5.15.4-Setup.exe	9,613KB
 VMware-player-15.5.0-14665864.exe	141,362KB
 VMware-converter-en-6.2.0-8466193.exe	176,047KB
 VirtualBox-6.0.12-133076-Win.exe	166,464KB
 SuperPuttySetup-1.4.0.9.msi	1,832KB
 rufus-3.8.exe	1,113KB
 putty-64bit-0.73-installer.msi	3,094KB
 DockerToolbox.exe	216,575KB
 Docker Desktop Installer.exe	856,885KB
 Advanced_IP_Scanner_2.5.3850.exe	19,908KB

메모:

# I. 실습 환경

---

## ❖ Hypervisor Installation

- ① Initial Powering Up (w/o Internet)
- ② USB booting Available
- ③ Alt-Ctrl-D로 Rebooting 하여 install 가능
- ④ Rebooting 시 'F11'에서 USB Booting 선택
- ⑤ ESXi '6.x' (6.x 설치 시연)
- ⑥ Windows Server 2016 Hyper-v (Option)

```
Please select boot device:

IBA GE Slot 0500 v1513
UEFI: Built-in EFI Shell
P0: TOSHIBA Q300 Pro.
SanDisk
UEFI: SanDisk
Enter Setup

↑ and ↓ to move selection
ENTER to select boot device
ESC to boot using defaults
```

**\*\* 실습 교육 진행은 OS나 웹브라우저 종류별로 다를 수 있는 동작을 고려하여 안정적 버전과 도구를 선택하여 진행 \*\***

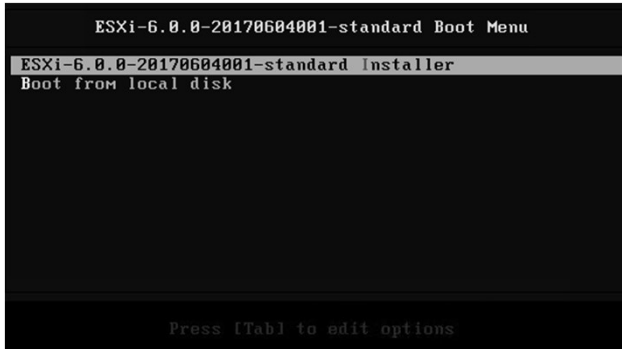
**메모:**

- Windows Containers on Windows Server: <https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/quick-start-windows-server>
- USB 부팅 재가동은 전원 off/on (전원 케이블 포함)필요함

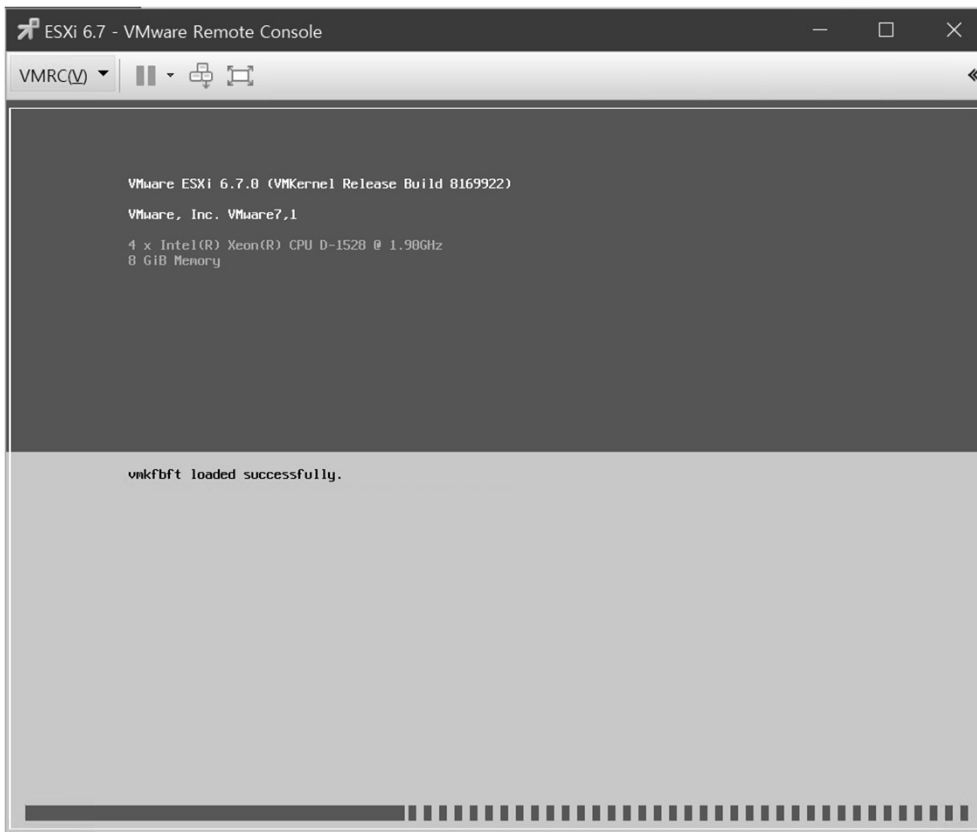
# I. 실습 환경

## ❖ Hypervisor Installation (ESXi 6.x 예)

①



②

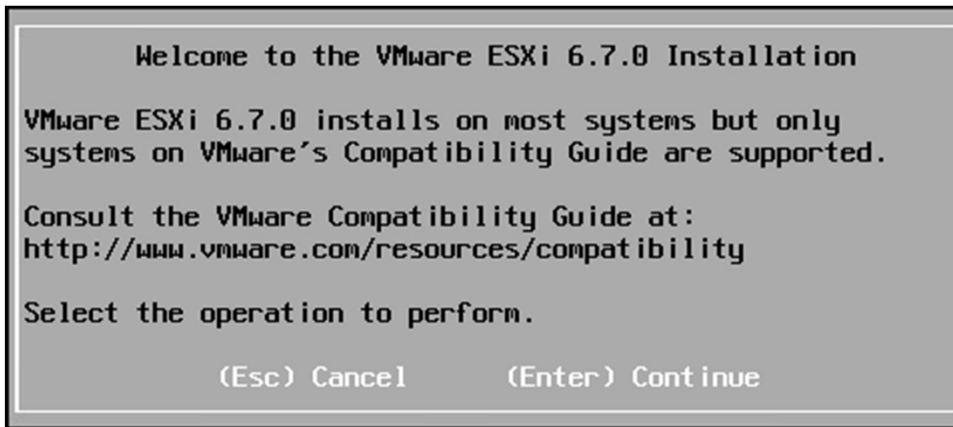


메모:

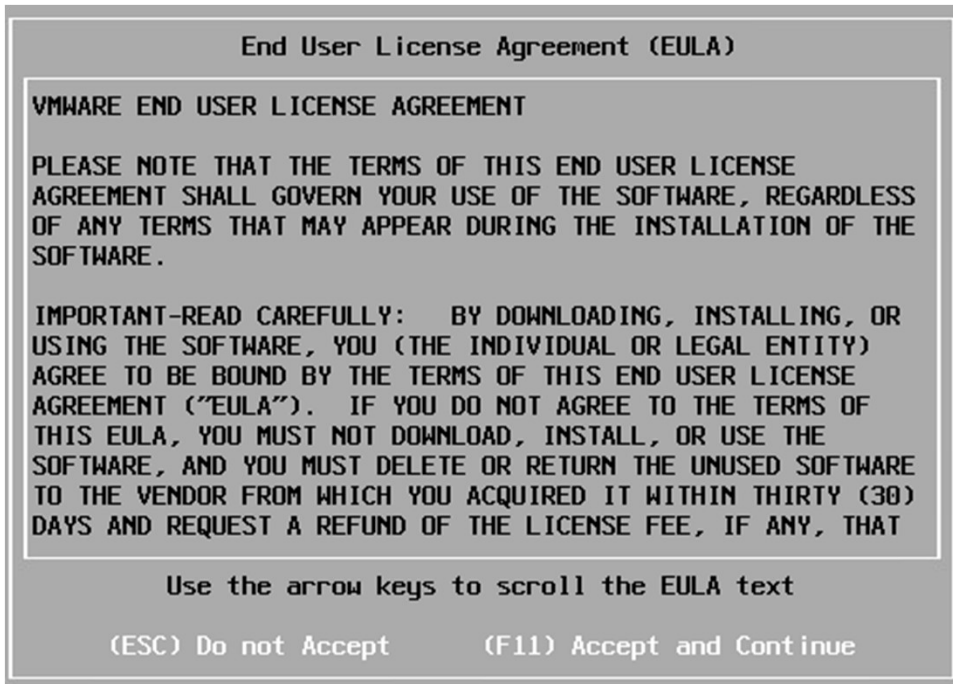
# I. 실습 환경

## ❖ Hypervisor Installation (ESXi 6.x 예)

①



②



### 메모:

- ESXi 6.7: Enter → F11 → US Default → Root Password (JSlab123) → F11

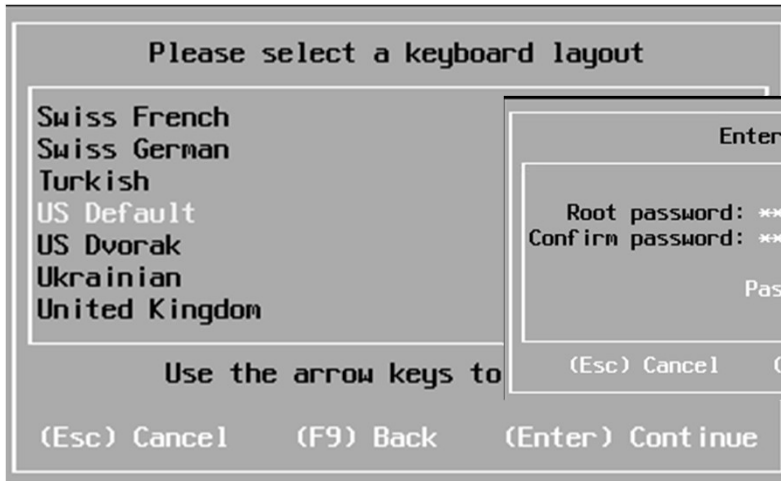
# I. 실습 환경

## ❖ Hypervisor Password Setting (ESXi 6.x 예)

①



②



③



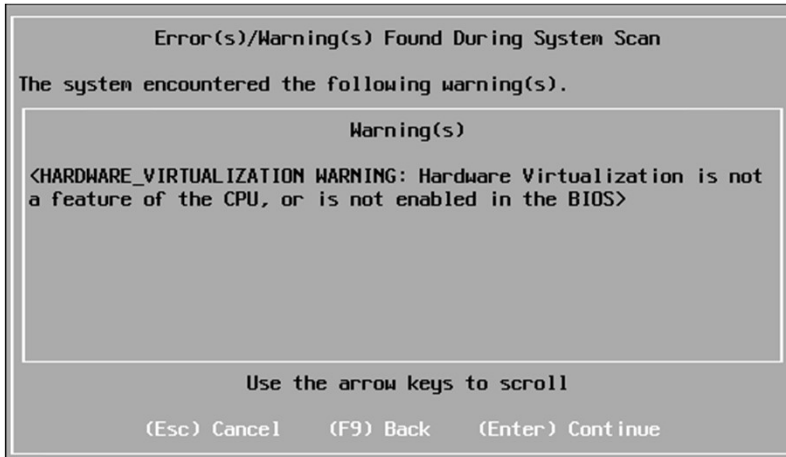
### 메모:

- ESXi 6.7: Enter → US Default → Root Password (JSlab123) → F11 → Enter

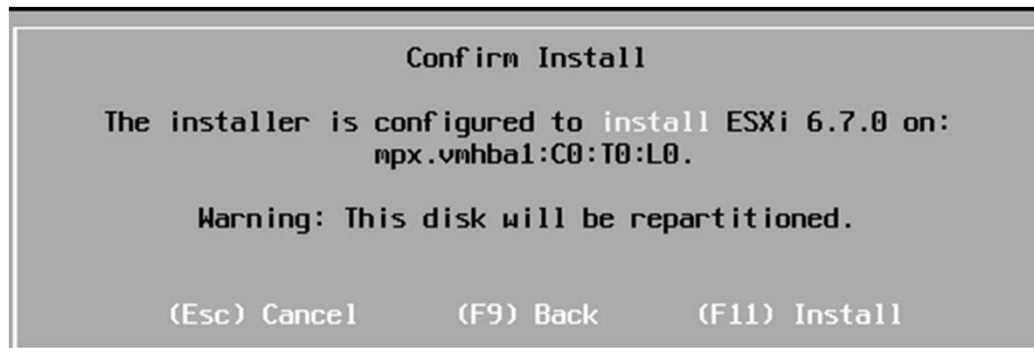
# I. 실습 환경

## ❖ Hypervisor Installation (ESXi 6.x 예)

①



②



### 메모:

- ESXi 6.7: Enter → F11

# I. 실습 환경

## ❖ Reboot Hypervisor Installation (ESXi 6.x 예)

①

```
Installation Complete

ESXi 6.0.0 has been successfully installed.

ESXi 6.0.0 will operate in evaluation mode for 60 days. To
use ESXi 6.0.0 after the evaluation period, you must
register for a VMware product license. To administer your
server, use the vSphere Client or the Direct Control User
Interface.

Remove the installation disc before rebooting.

Reboot the server to start using ESXi 6.0.0.

(Enter) Reboot
```

②

```
VMware ESXi 6.7.0 (VMKernel Release Build 8169922)

VMware, Inc. VMware7.1

4 x Intel(R) Xeon(R) CPU D-1528 @ 1.90GHz
8 GiB Memory

To manage this host go to:
http://192.168.55.121/ (DHCP)
http://1fe80::20c:29ff:fee4:834e1/ (STATIC)

<F2> Customize System/View Logs
<F12> Shut Down/Restart
```

메모:

# I. 실습 환경

## ❖ Hypervisor IP Address Setting

- ① Configure Management Network 선택
- ② 좌석 번호 'TT' 이용 고정 IP 주소 설정 - 192.168.XX.TT

The screenshot displays the 'System Customization' menu on the left and the 'Configure Management Network' configuration screen on the right. In the menu, 'Configure Management Network' is highlighted with a circled '1'. The configuration screen shows 'Hostname: localhost' and 'IPv4 Address: 192.168.1.229' with a circled '2'. Below the configuration screen is a grid of 48 IP address seats, each represented by a small device icon with a display showing the IP address. The seats are arranged in 5 rows and 6 columns. The top row has a header '강사용 .99' and seats for .16 to .11. The second row has seats for .26 to .21. The third row has seats for .36 to .31. The fourth row has seats for .46 to .41. The bottom row has seats for .47 to .40. At the bottom left of the grid is the text '<Up/Down> Select' and at the bottom right is '<Esc> Log Out'.

메모:



# I. 실습 환경

## ❖ Web Browser via WiFi

- ① 웹 브라우저로 접속: <http://192.168.xx.yy> (WiFi 접속 가능)
- ② 개선 프로그램 확인

vmware ESXi™ | root@192.168.0.232 | 도움말 | 검색

localhost.localdomain

vCenter Server 가져오기 | VM 생성/등록 | 종료 | 재부팅 | 새로 고침 | 작업

**localhost.localdomain**  
버전: 6.7.0 (Build 8169922)  
상태: 보통 (vCenter Server에 연결되지 않음)  
가동 시간: 0.01 일

CPU 사용 가능: 7.6 GHz | 사용량: 49 MHz (1%)  
메모리 사용 가능: 6.87 GB | 사용량: 1.13 GB (14%)  
스토리지 사용 가능: 31.09 GB | 사용량: 1.41 GB (4%)

현재 평가 모드에서 ESXi를 사용하고 있습니다. 이 라이선스는 60일 후에 만료됩니다.

하드웨어	
제조업체	VMware, Inc.
모델	VMware7,1
CPU	4 CPUs x Intel(R) Xeon(R) CPU D-1528 @ 1.90GHz
메모리	8 GB
영구 메모리	0 B
가상 플래시	0 B 사용됨, 0 B 용량
네트워킹	
호스트 이름	localhost.localdomain
IP 주소	1. vmk0: 192.168.0.232 2. vmk0: fe80::...

VMware Host Client를 개선할 수 있도록 도와주시십시오.

이 제품은 VMware의 "CEIP"(고객 환경 향상 프로그램)에 참여하고 있습니다. CEIP는 VMware에 제품 및 서비스를 개선하고, 문제를 해결하고, 제품을 배포하고 사용하는 최적의 방법을 사용자에게 알려주기 위한 정보를 제공합니다. CEIP의 일부로, VMware는 사용자 조직의 VMware 제품 및 서비스 사용에 대한 기술 정보를 사용자 조직의 VMware 라이선스 키와 함께 정기적으로 수집합니다. 이 정보는 사용자를 개인적으로 식별하지 않습니다. CEIP에 대한 자세한 내용은 VMware.com에서 신뢰 및 보증 센터를 참조하십시오. 참여 기본 설정은 아래에서 선택하거나 Host Client의 설정 메뉴에서 선택할 수 있습니다.

VMware 고객 환경 향상 프로그램에 참여

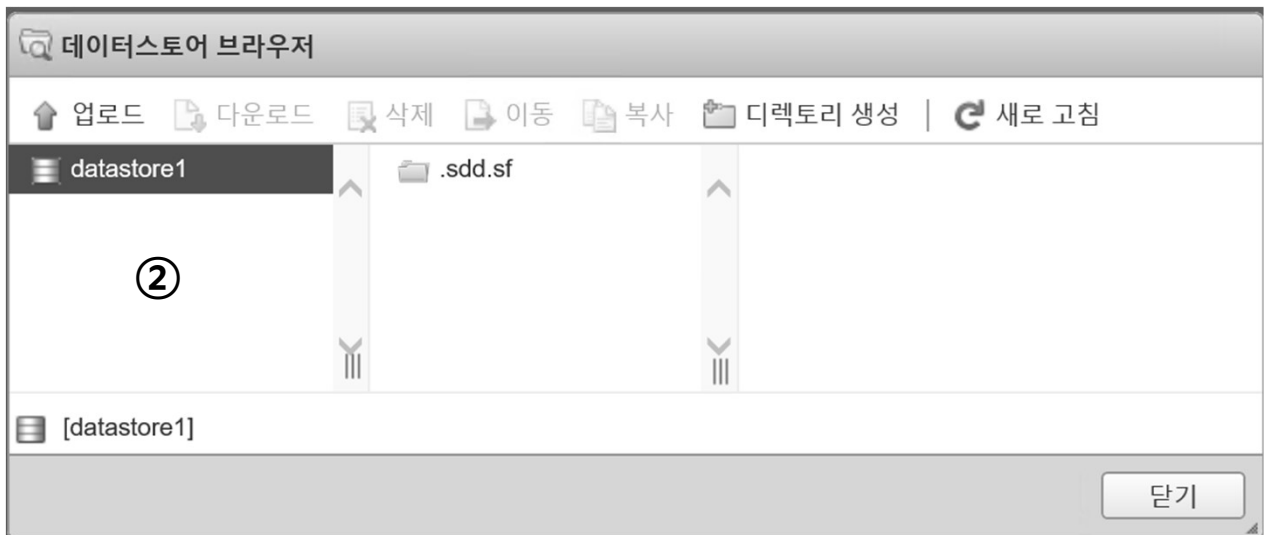
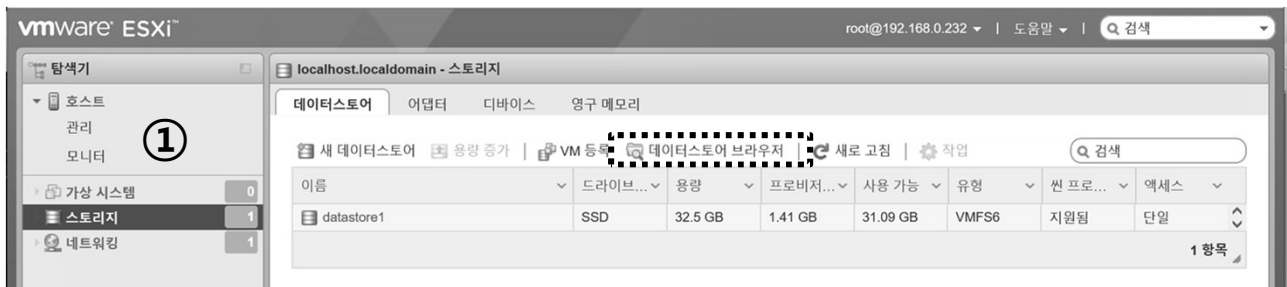
확인

메모:

# I. 실습 환경

## ❖ 스토리지

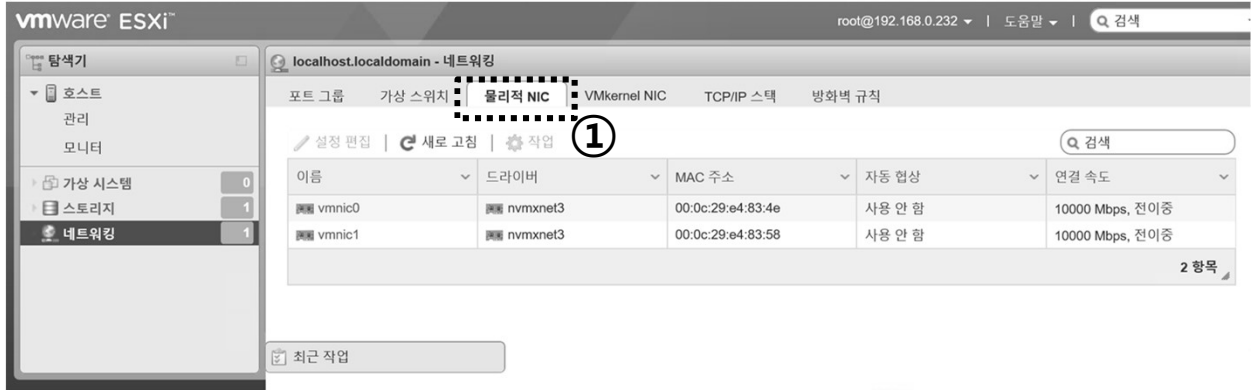
- ① 스토리지 선택
- ② 데이터 스토어 브라우저



메모:


# I. 실습 환경

## ❖ 네트워킹 (pNIC, 가상스위치, 포트그룹)



메모:

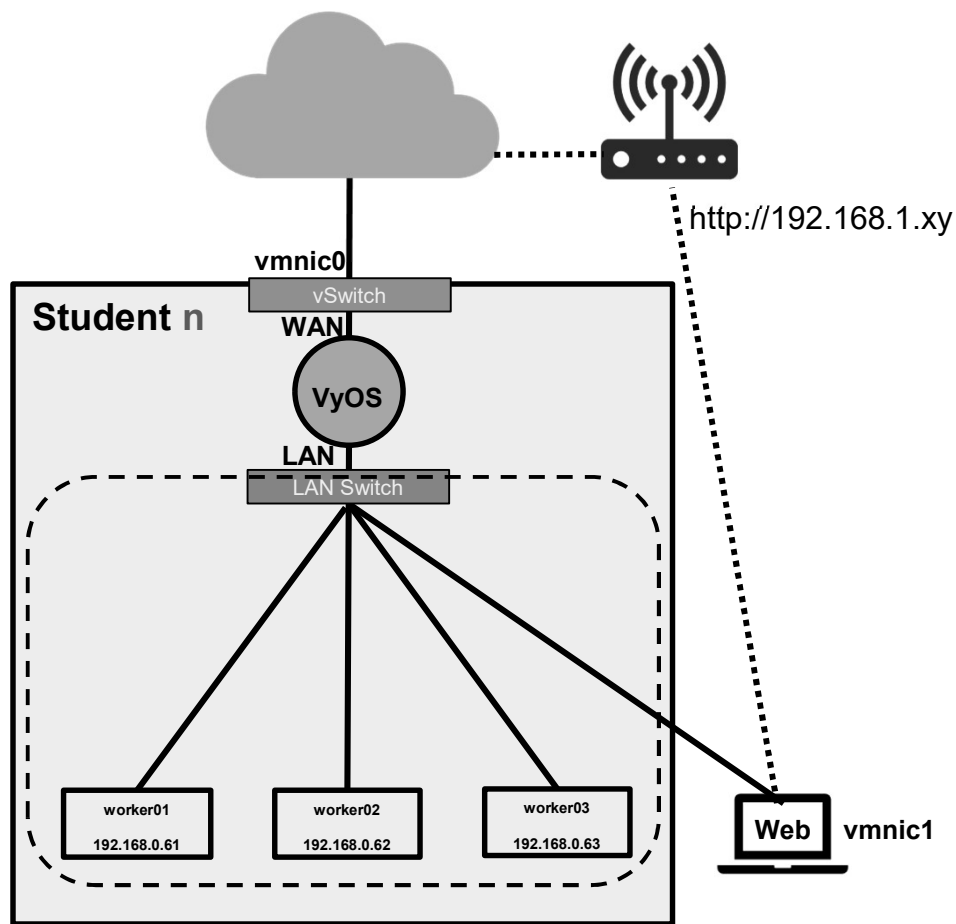
# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

## II. 라우터 (VyOS)

### ❖ 가상 스위치 설치 환경 (예)

- ① 하이퍼바이저 내 인터넷용과 호스트 연결 스위치 2개 필요
- ② WAN은 인터넷, LAN은 호스트 연결 vSwitch 별도 생성
- ③ 설정을 위한 클라이언트는 VM 또는 유선랜 연결 PC 사용 (외부 유선랜 연결이 어려운 경우 하이퍼바이저에 웹으로 연결 사용)



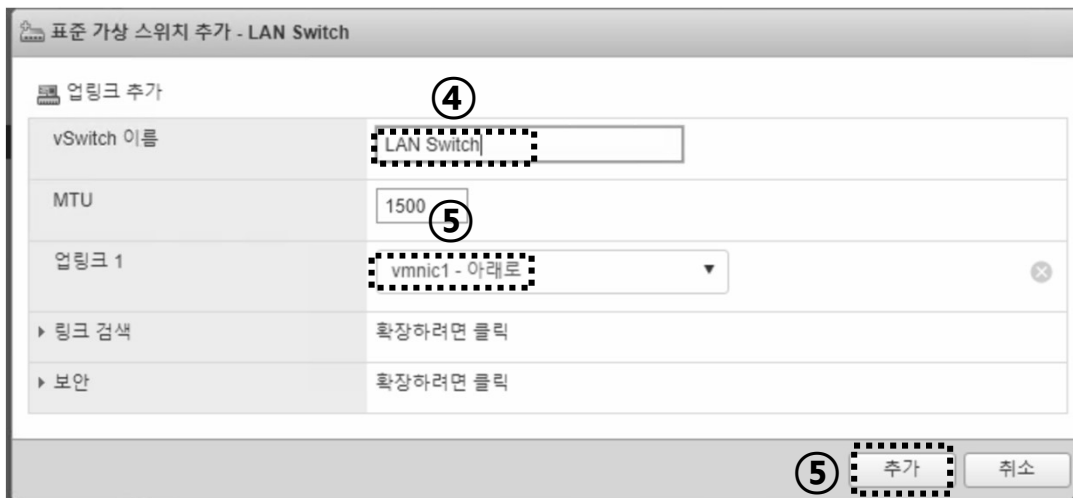
#### 메모:

- 라우터는 실습 중 발생 가능한 Loop 발생과 코어 네트워크의 DHCP 서버의 부담을 낮춤
- VMRC 설치: VMware-VMRC-10.0.4-11818843
- 스위치는 2개 사용 (WAN/LAN)
- 하이퍼바이저 접속 `http://192.168.1.xy`
- 하이퍼바이저 계정: ID / Password ( root / JSlab123)

## II. 라우터 (VyOS)

### ❖ 가상 스위치 설정

- ① 네트워킹(Networking) 선택 확인
- ② 가상스위치 선택
- ③ 표준 가상 스위치 추가
- ④ 이름 'LAN Switch' 설정
- ⑤ 업링크 'vmnic1' 확인 → 추가 단추



메모:

## II. 라우터 (VyOS)

### ❖ 가상 스위치 설정 상세 'LAN Switch'

표준 가상 스위치 추가 - LAN

업링크 추가

vSwitch 이름	① LAN
MTU	1500
업링크 1	vmnic0 - 위로, 10000 mbps
▶ 링크 검색	확장하려면 클릭
▶ 보안	확장하려면 클릭

추가 취소

▶ 링크 검색

모드	② 수신
프로토콜	CDP(Cisco Discovery Protocol)

▶ 보안

비규칙 모드	③ <input type="radio"/> 동의 <input checked="" type="radio"/> 거부
MAC 주소 변경	<input type="radio"/> 동의 <input checked="" type="radio"/> 거부
위조 전송	<input type="radio"/> 동의 <input checked="" type="radio"/> 거부

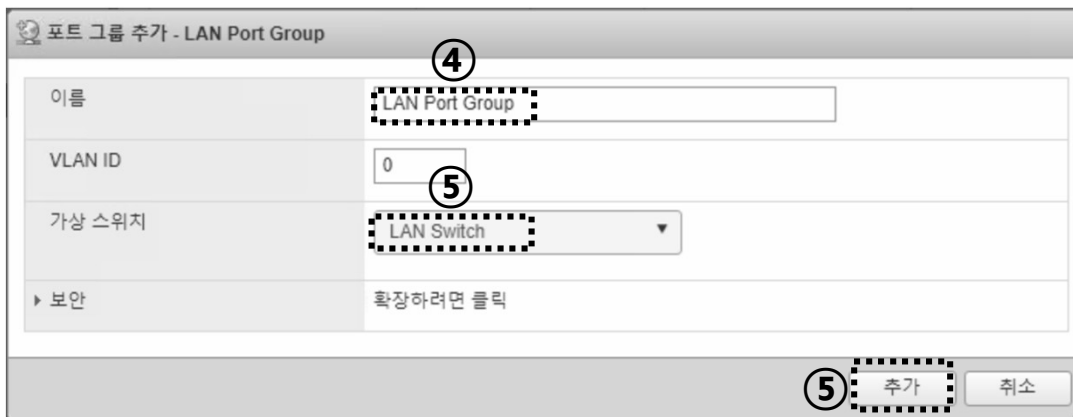
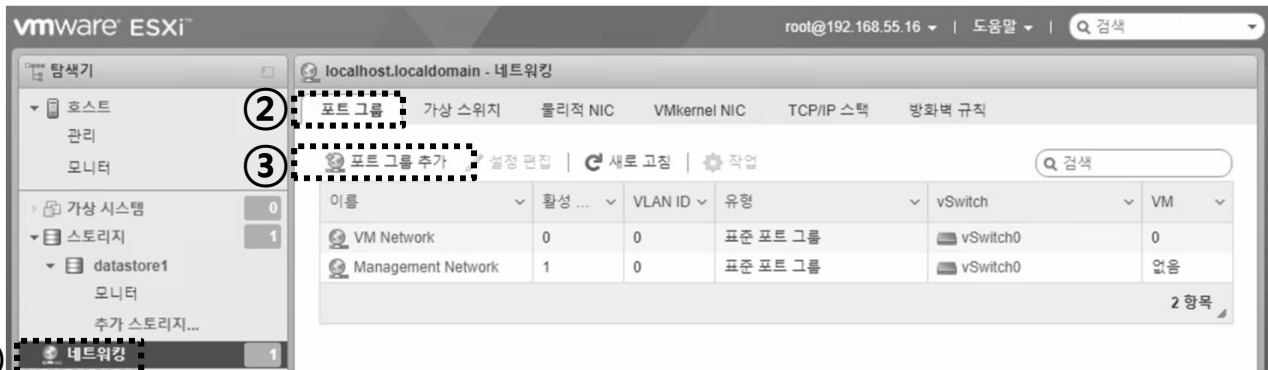
#### 메모:

- 스위치 설정 후 포트 그룹 추가 시 사용

## II. 라우터 (VyOS)

### ❖ 포트그룹 설정

- ① 네트워킹(Networking) 선택 확인
- ② 포트 그룹 선택
- ③ 포트 그룹 추가
- ④ 이름 'LAN Port Group' 설정
- ⑤ 생성한 가상 스위치 'LAN Switch' 확인 → 추가 단추



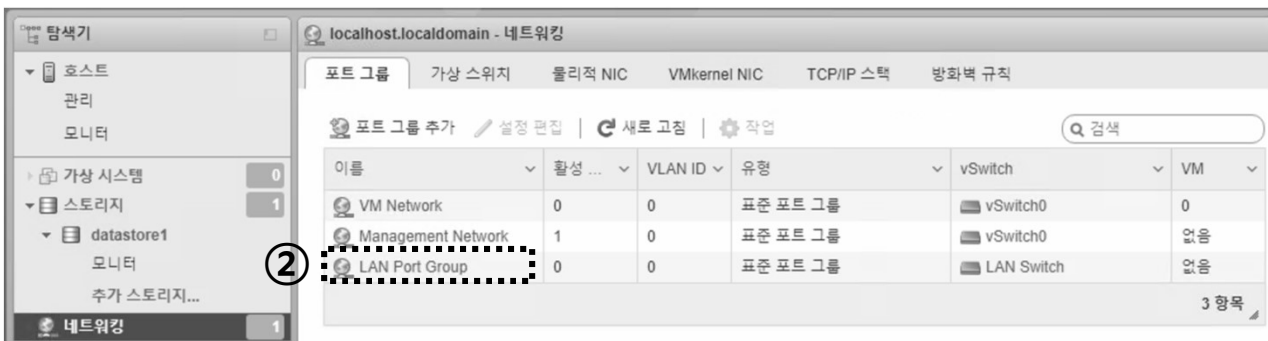
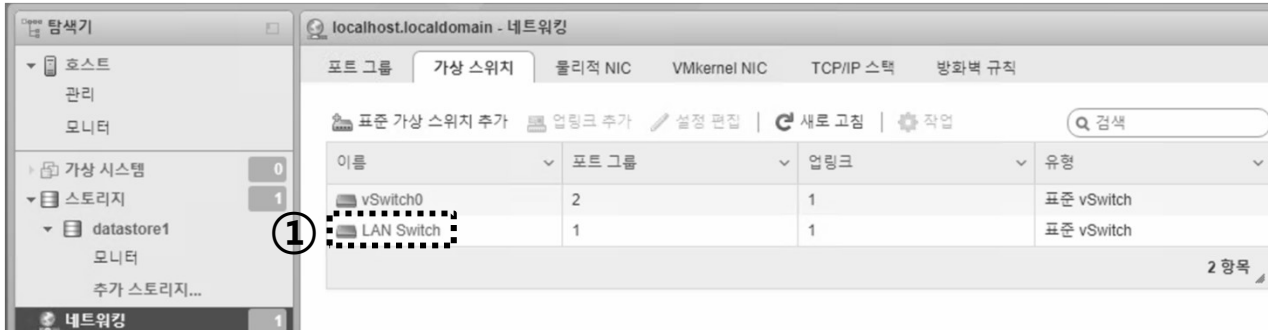
메모:



## II. 라우터 (VyOS)

### ❖ 가상 스위치/포트그룹 설정 확인

- ① 생성 가상 스위치 'LAN Switch' 확인
- ② 생성 포트 그룹 'LAN Port Group' 확인
- ③ 인터넷 연결되어 자동 생성한 'vSwitch0'는 WAN 스위치로 사용

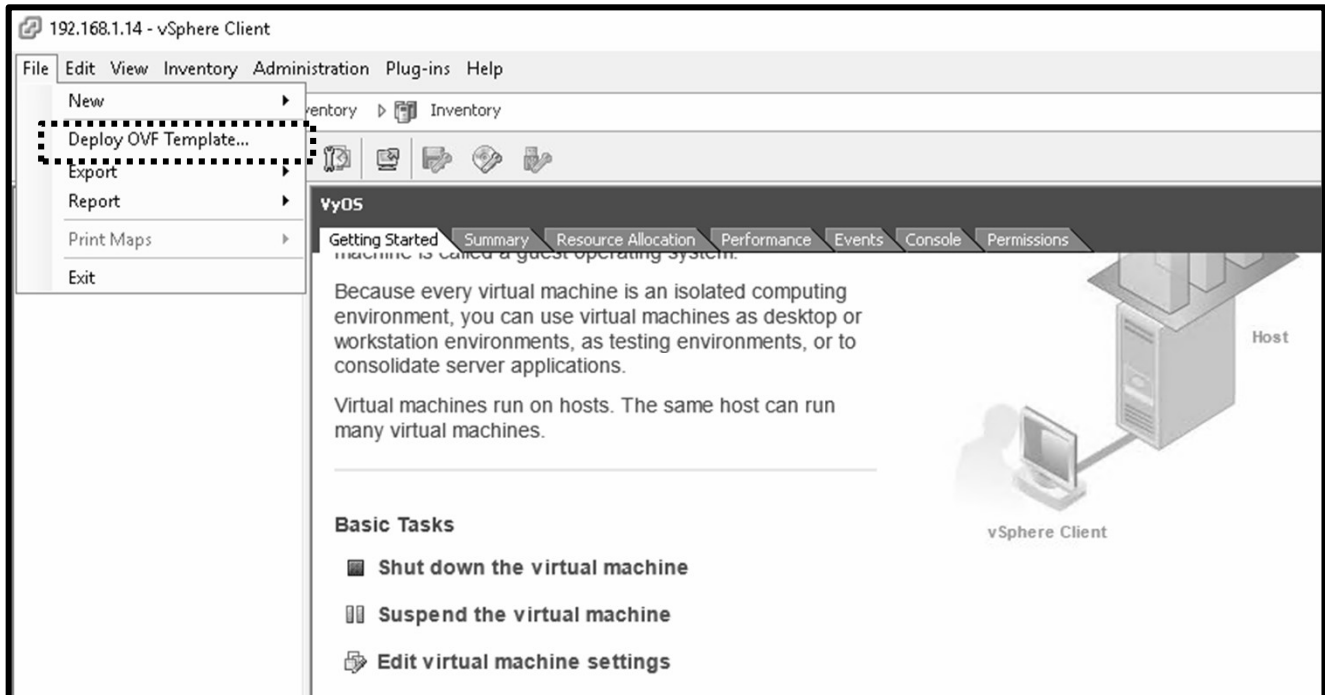


메모:

## II. 라우터 (VyOS)

### ❖ Router(VyOS) Installation (전용 Client 도구 사용)

- ① 'File' 선택
- ② 'Deploy OVF Template' 선택

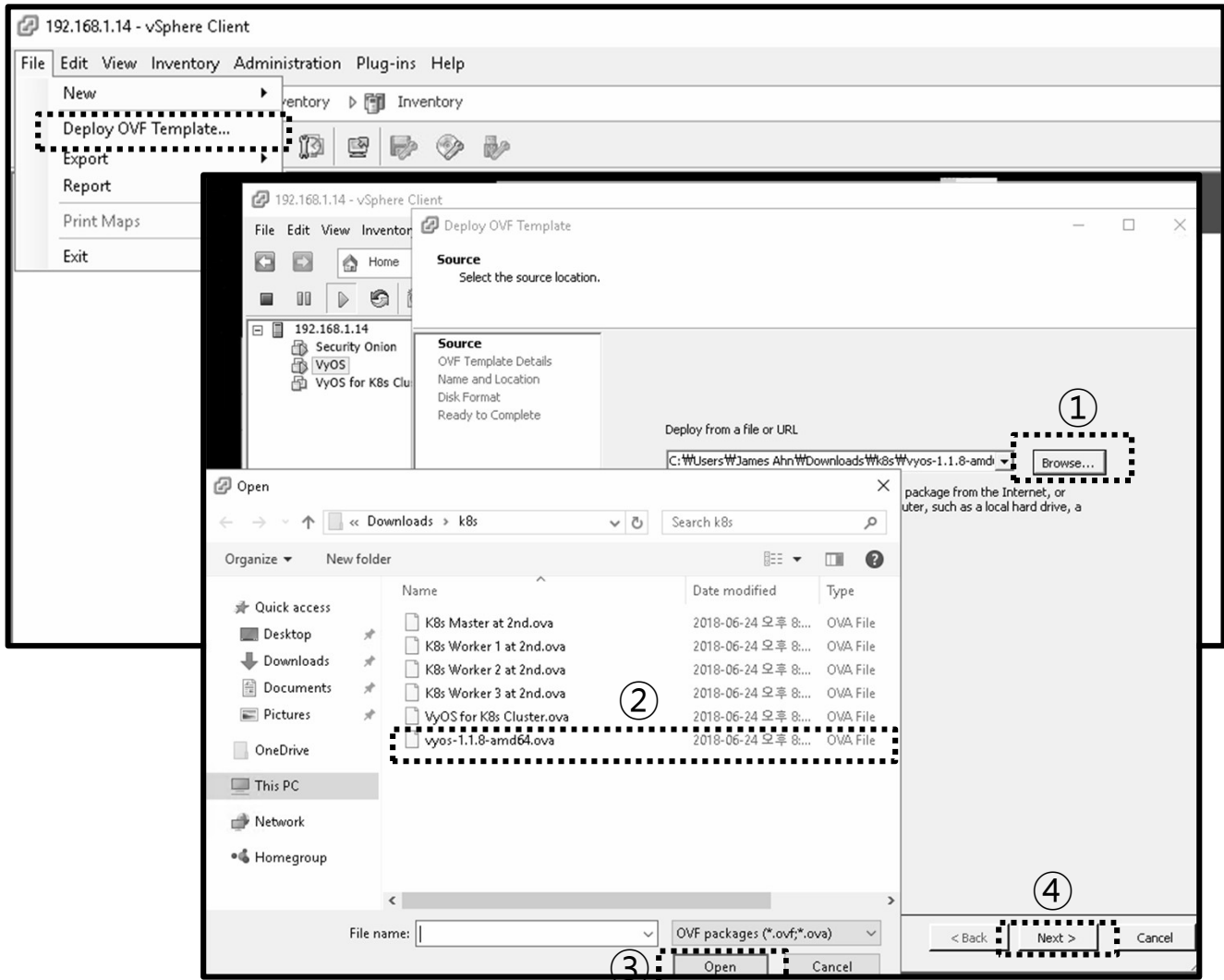


메모:

## II. 라우터 (VyOS)

### ❖ Router(VyOS) Installation (전용 Client 도구 사용)

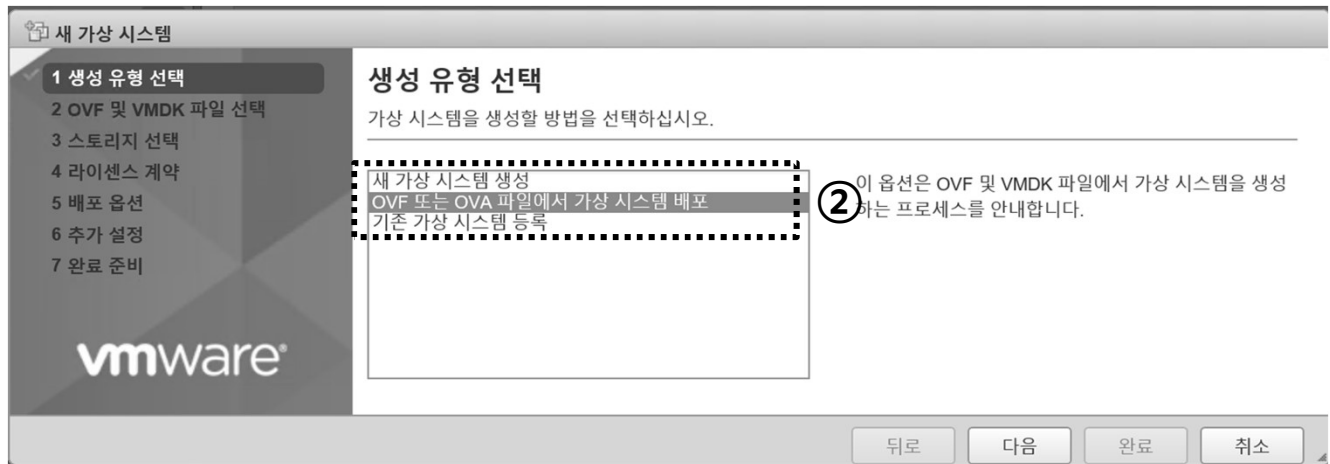
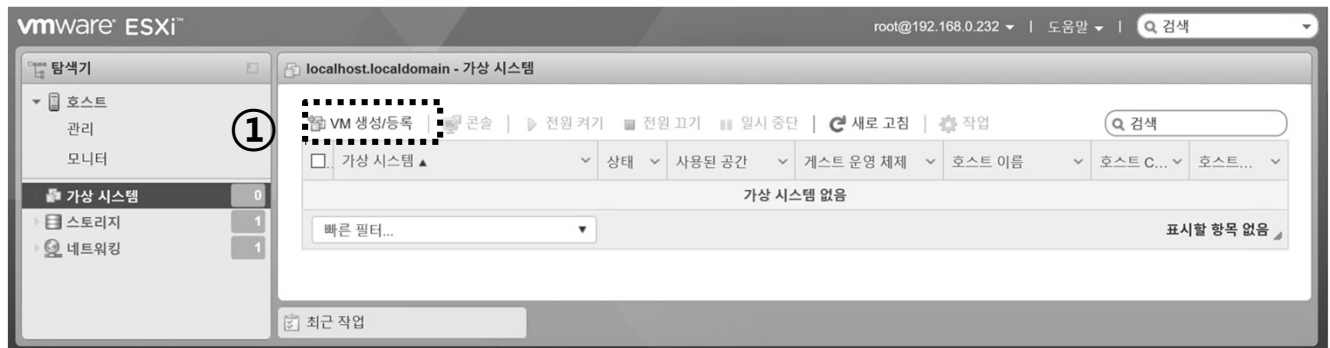
- ① VyOS OVA 선택
- ② 유선랜 네트워크 연결 (내부 네트워크를 위한 선택)



메모:

## II. 라우터 (VyOS)

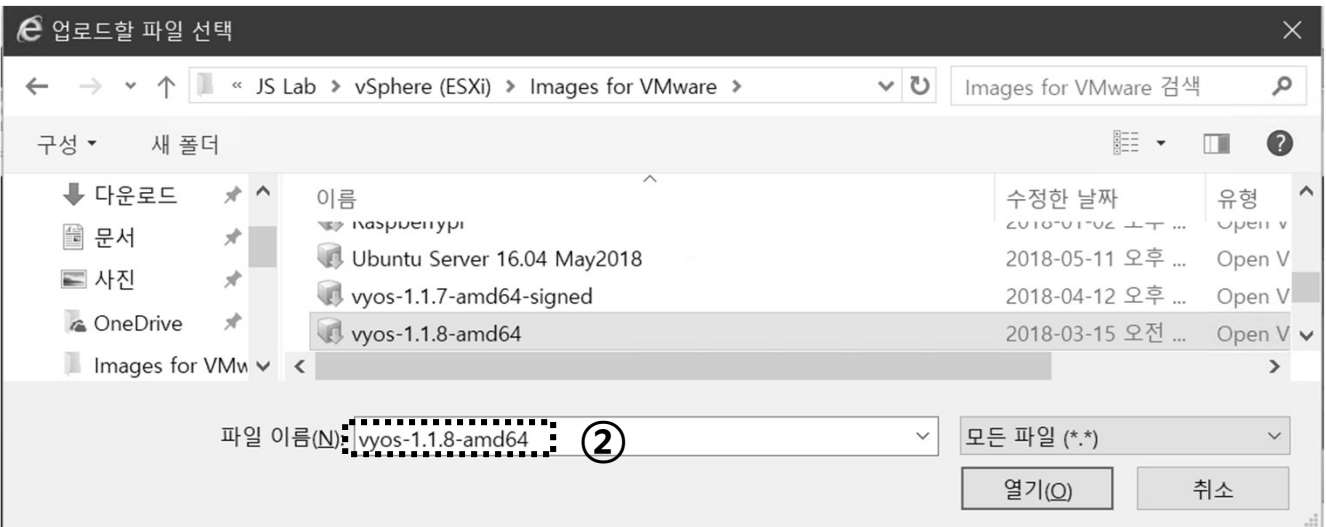
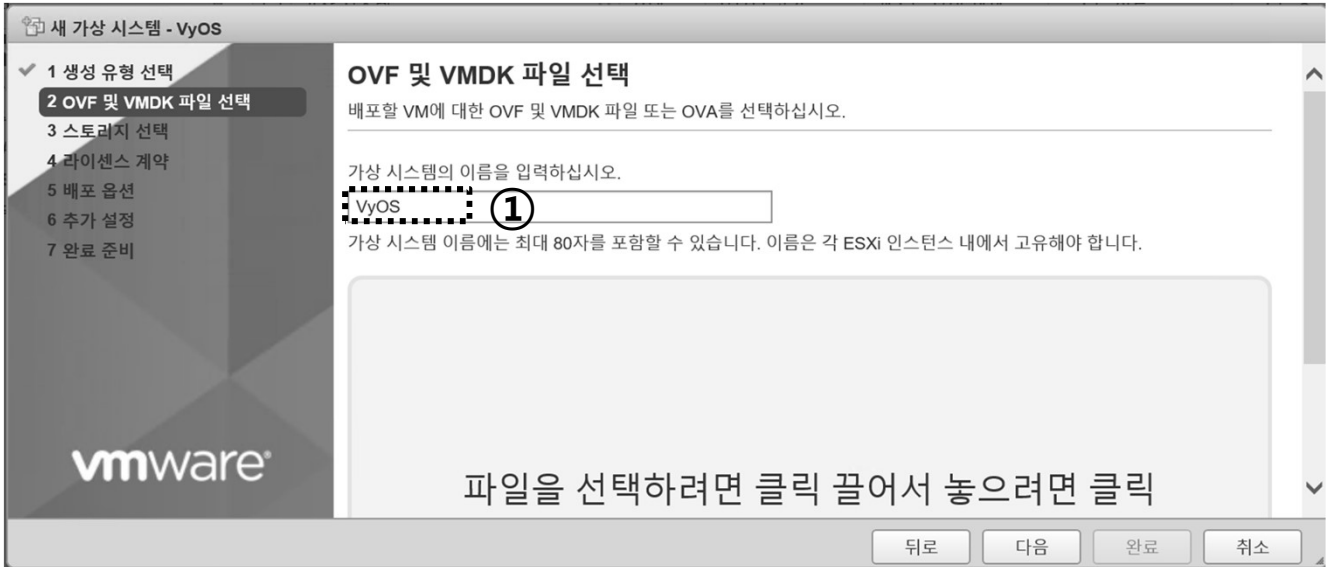
### ❖ 가상 시스템 생성 (웹브라우저 사용)



메모:

## II. 라우터 (VyOS)

### ❖ 가상 시스템 생성 (vRouter 이미지 'VyOS' 사용)

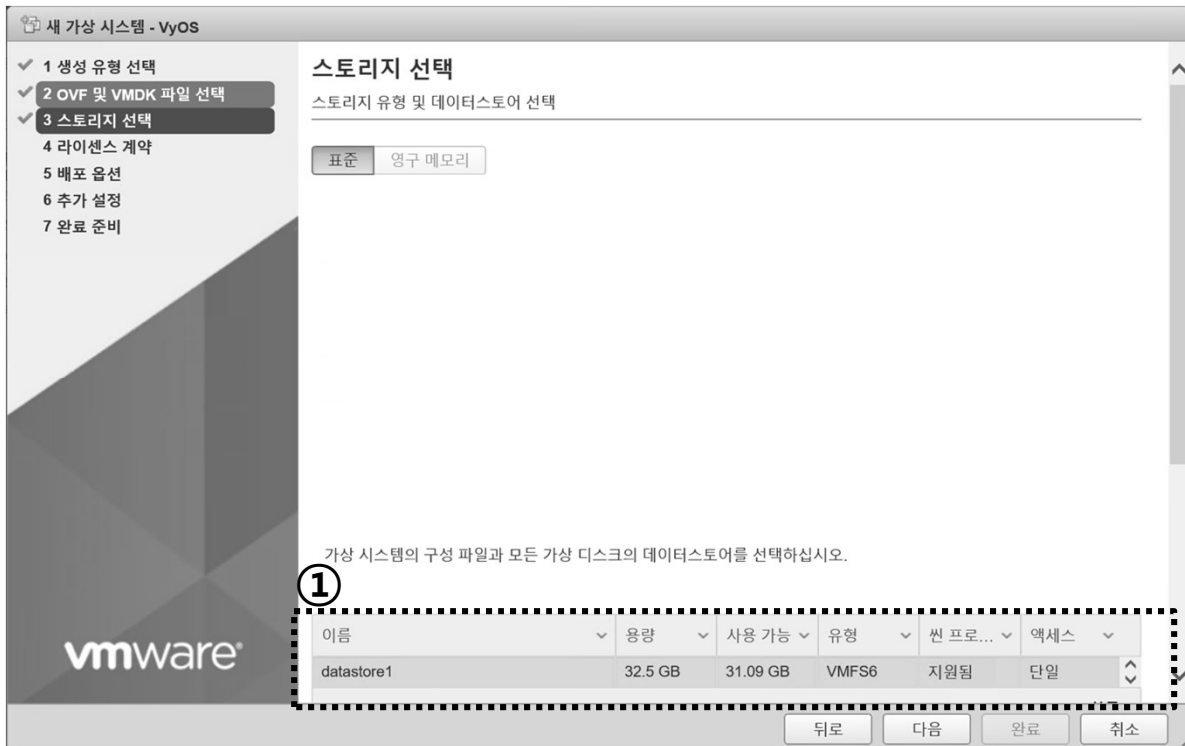


#### 메모:

- 라우터 VyOS 이미지 다운로드: <https://downloads.vyos.io/?dir=release/1.1.8>
- VMware OVA 템플릿 이미지 사용 가능 (예: vyos-1.1.8-amd64.ova, 약 230 MB)

## II. 라우터 (VyOS)

### ❖ 가상 시스템 생성 설치 위치 지정 및 네트워크 매핑



메모:

## II. 라우터 (VyOS)

### ❖ 가상 시스템 생성 설치 완료

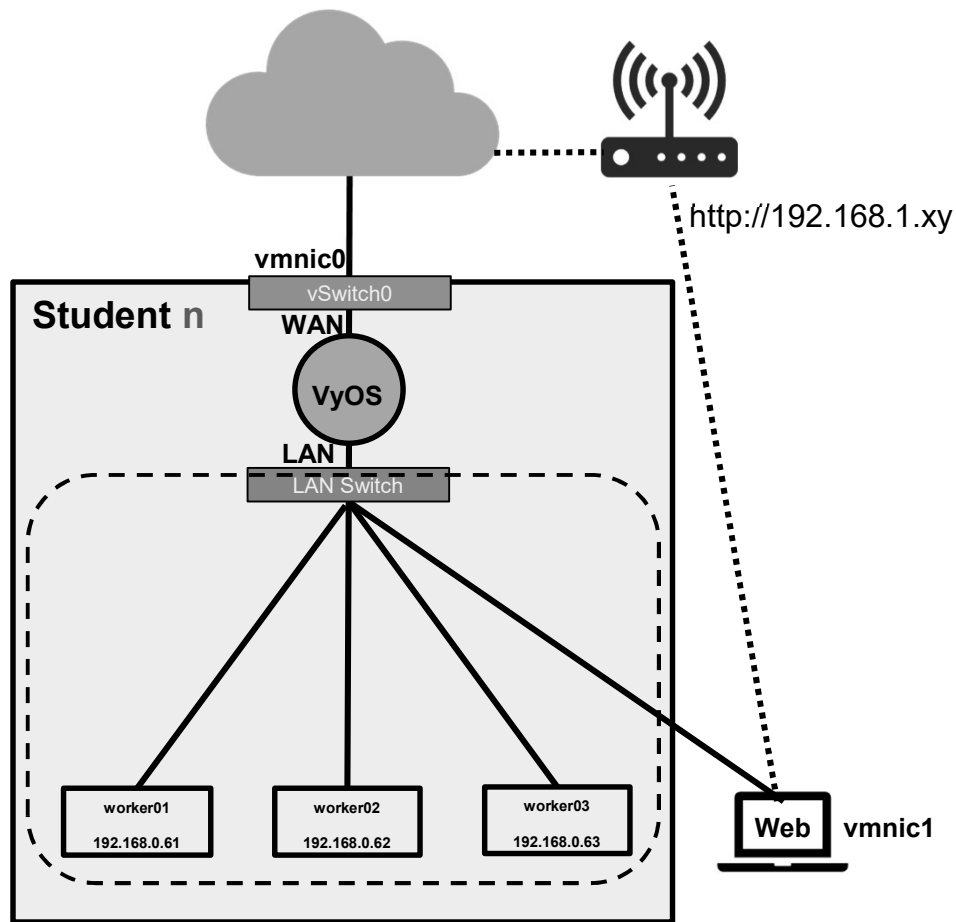


메모:

## II. 라우터 (VyOS)

### ❖ 라우터 'VyOS' 설치 환경

- ① 하이퍼바이저 내 인터넷용과 호스트 연결 스위치 2개 필요
- ② 라우터 WAN은 인터넷 스위치, LAN은 호스트 연결 스위치
- ③ 설정을 위한 클라이언트는 VM 또는 유선랜 연결 PC 사용



#### 메모:

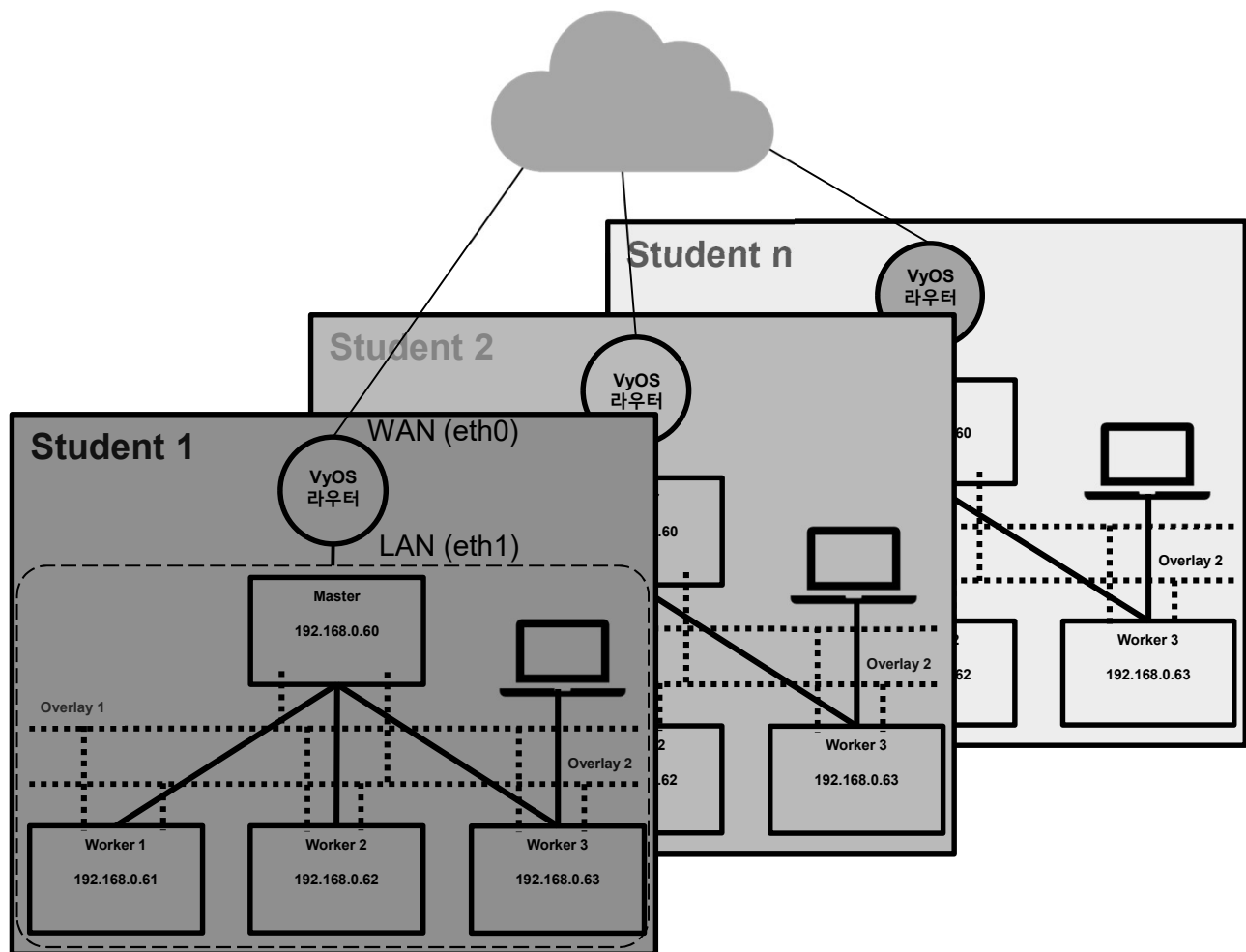
- 라우터는 실습 중 발생 가능한 Loop와 코어 네트워크의 DHCP 서버의 부담을 낮춤
- 라우터 VyOS 이미지 다운로드: <https://downloads.vyos.io/?dir=release/1.1.8>
- VMware OVA 템플릿 이미지 사용 가능 (예: vyos-1.1.8-amd64.ova, 약 230 MB)
- Ubuntu Server 16.04 이미지 : 실습 시간 부족 시 초기설치 완료한 이미지 사용



## II. 라우터 (VyOS)

### ❖ 라우터 'VyOS' 설치 환경

- ① 하이퍼바이저 내 인터넷용과 호스트 연결 스위치 2개 필요
- ② 라우터의 WAN은 인터넷, LAN은 호스트 연결 스위치 접속



#### 메모:

- Ubuntu Server 16.04 이미지 : 실습 시간 부족 시 초기설치 완료한 이미지 사용
- CentOS 7 minimal 이미지 : [http://isoredirect.centos.org/centos/7/isos/x86\\_64/CentOS-7-x86\\_64-Minimal-1708.iso](http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-Minimal-1708.iso) (실습 시간 2배 소요)

## II. 라우터 (VyOS)

### ❖ 라우터 'VyOS' 설치를 위한 접속

- ① 계정: ID / Password (vyos/vyos) 호스트 연결 스위치 접속
- ② configure
- ③ set service ssh
- ④ commit
- ⑤ save
- ⑥ exit
- ⑦ show interface (eth0의 DHCP 서버 할당 IP 주소 사용)
- ⑧ Putty 등으로 접속

```
Starting periodic command scheduler: cron.
Loading cpufreq kernel modules...done (none).
Starting routing daemons: ripd ripngd ospfd ospf6d bgpd.
Mounting VyOS Config...done.
Starting VyOS router: migrate rl-system firewall configure.
Starting vyos-intfwatcd: vyos-intfwatcd.

Welcome to VyOS - vyos tty1

vyos login: vyos
Password:
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Sat Nov 11 12:10:30 CET 2017 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address      S/L  Description
-----
eth0           192.168.1.109/24  u/u
eth1           -                u/u
lo             127.0.0.1/8     u/u
              ::1/128
vyos@vyos:~$
```

가상 라우터 VyOS 터미널 접속 (예)

```
vyos@vyos:~$ show interface
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address      S/L  Description
-----
eth0           192.168.1.109/24  u/u
eth1           -                u/u
lo             127.0.0.1/8     u/u
              ::1/128
vyos@vyos:~$
```

가상 라우터 VyOS에 SSH 접속 (예)

#### 메모:

- [https://wiki.vyos.net/wiki/User\\_Guide](https://wiki.vyos.net/wiki/User_Guide)

## II. 라우터 (VyOS)

### ❖ VyOS 컨피규레이션 세팅

- ① **configure**
- ② **set interfaces ethernet eth0 address dhcp # Internet**
- ③ **set interfaces ethernet eth0 description 'WAN'**
- ④ **set interfaces ethernet eth1 address '192.168.0.1/24'**
- ⑤ **set interfaces ethernet eth1 description 'LAN'**
- ⑥ **set nat source rule 100 outbound-interface 'eth0' # NAT**
- ⑦ **set nat source rule 100 source address '192.168.0.0/24'**
- ⑧ **set nat source rule 100 translation address masquerade**
- ⑨ **set service dhcp-server disabled 'false' # DHCP Server**
- ⑩ **set service dhcp-server shared-network-name LAN  
subnet 192.168.0.0/24 default-router '192.168.0.1'**
- ⑪ **set service dhcp-server shared-network-name LAN  
subnet 192.168.0.0/24 dns-server '192.168.0.1'**
- ⑫ **set service dhcp-server shared-network-name LAN  
subnet 192.168.0.0/24 domain-name 'internal-network'**
- ⑬ **set service dhcp-server shared-network-name LAN  
subnet 192.168.0.0/24 lease '86400'**
- ⑭ **set service dhcp-server shared-network-name LAN  
subnet 192.168.0.0/24 start '192.168.0.200' stop  
'192.168.0.232'**
- ⑮ **set service dns forwarding cache-size '0' # DNS**
- ⑯ **set service dns forwarding listen-on 'eth1'**
- ⑰ **set service dns forwarding name-server '8.8.8.8'**
- ⑱ **# commit → save → exit 후에 실행**

#### 메모:

- 라우터 이름(예): set system host-name 'vyos-1'
- 인터페이스 확인: 'show interface'
- 컨피규레이션 완료: 'commit' & 'save'
- DHCP IP주소 할당 확인: show dhcp server leases
- 업무 적용시: 고정 IP 주소 사용 권장

## II. 라우터 (VyOS)

### ❖ VyOS Operation

- ① `show dhcp server leases`      # commit → save → exit  
후에 실행
- ② `show interface`

```
vyos@vyos:~$ show interface
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address      S/L  Description
-----
eth0           192.168.99.114/24  u/u  WAN
eth1           192.168.0.1/24   u/u  LAN
lo             127.0.0.1/8      u/u
              ::1/128
```

```
vyos@vyos:~$
```

```
vyos@vyos:~$ show dhcp server leases
```

```
IP address      Hardware address  Lease expiration  Pool  Client Name
-----
vyos@vyos:~$
```

#### 메모:

- 실습용 호스트를 위한 DHCP 서버 설정
- VMware 이미지 사용 가능 (예: vyos-1.1.8-amd64.ova)

## II. 라우터 (VyOS)


### ❖ VyOS 세팅 후 컨피규레이션 확인

```
vyos@vyos:~$ show config
interfaces {
  ethernet eth0 {
    address dhcp
    description WAN
    duplex auto
    hw-id 00:0c:29:fd:c9:ca
    smp_affinity auto
    speed auto
  }
  ethernet eth1 {
    address 192.168.0.1/24
    description LAN
    duplex auto
    hw-id 00:0c:29:fd:c9:d4
    smp_affinity auto
    speed auto
  }
  loopback lo {
  }
}
nat {
  source {
    rule 100 {
      outbound-interface eth0
      source {
        address
        192.168.0.0/24
      }
      translation {
        address masquerade
      }
    }
  }
}
service {
  dhcp-server {
    disabled false
    shared-network-name LAN {
      authoritative disable
      subnet 192.168.0.0/24 {
        default-router 192.168.0.1
        dns-server 192.168.0.1
        domain-name internal-network
        lease 86400
        start 192.168.0.200 {
          stop 192.168.0.232
        }
      }
    }
  }
  dns {
    forwarding {
      cache-size 0
      listen-on eth1
      name-server 8.8.8.8
    }
  }
  ssh {
    port 22
  }
}
system {
  config-management {
    commit-revisions 100
  }
  console {
  }
  host-name vyos
  login {
    user vyos {
      authentication {
        encrypted-password *****
        plaintext-password *****
      }
      level admin
    }
  }
}
ntp {
  server 0.pool.ntp.org {
  }
  server 1.pool.ntp.org {
  }
  server 2.pool.ntp.org {
  }
}
package {
  auto-sync 1
  repository community {
    components main
    distribution helium
    password *****
    url http://packages.vyos.net/vyos
    username ""
  }
}
syslog {
  global {
    facility all {
      level notice
    }
    facility protocols {
      level debug
    }
  }
}
time-zone UTC
}
```

#### 메모:

- LAN/WAN 설정
- DHCP 서버 설정
- VMware 이미지 사용 가능

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

### III. vUTM (pfSense)

---

#### ❖ vUTM 개요

- 최고의 보안 인프라 실습 환경 제공
- UTM은 기본적인 보안 시스템 내장
  - ✓ 방화벽
  - ✓ 침입탐지/차단 (IDS/IPS)
  - ✓ L2/L3 라우팅
  - ✓ 무선랜 보안
  - ✓ 가상사설망(VPN)
  - ✓ 웹필터링 (Web Filtering)
  - ✓ 안티바이러스
  - ✓ DLP (Data Loss Prevention)
- 실습은 오픈소스 사용 (pfSense 소호 레퍼런스)
  - ✓ 라우터 모드, 브릿지 모드 제공
  - ✓ Stateful packet filtering
  - ✓ OS/Network 핑거프린팅 필터링
  - ✓ 방화벽 로그
  - ✓ 이중화 (고가용성)
  - ✓ 룰 그룹 관리 (Aliases)DDoS 방어 (SynProxy)
  - ✓ VPN (IPSEC/OpenVPN/PPTP/SSH 터널링 연동)
  - ✓ 웹필터링/웹프락시 (SquidGuard)
  - ✓ AntiVirus (ClamAV)
  - ✓ 모니터링 (CPU, Throughput, 그래프, 포털)

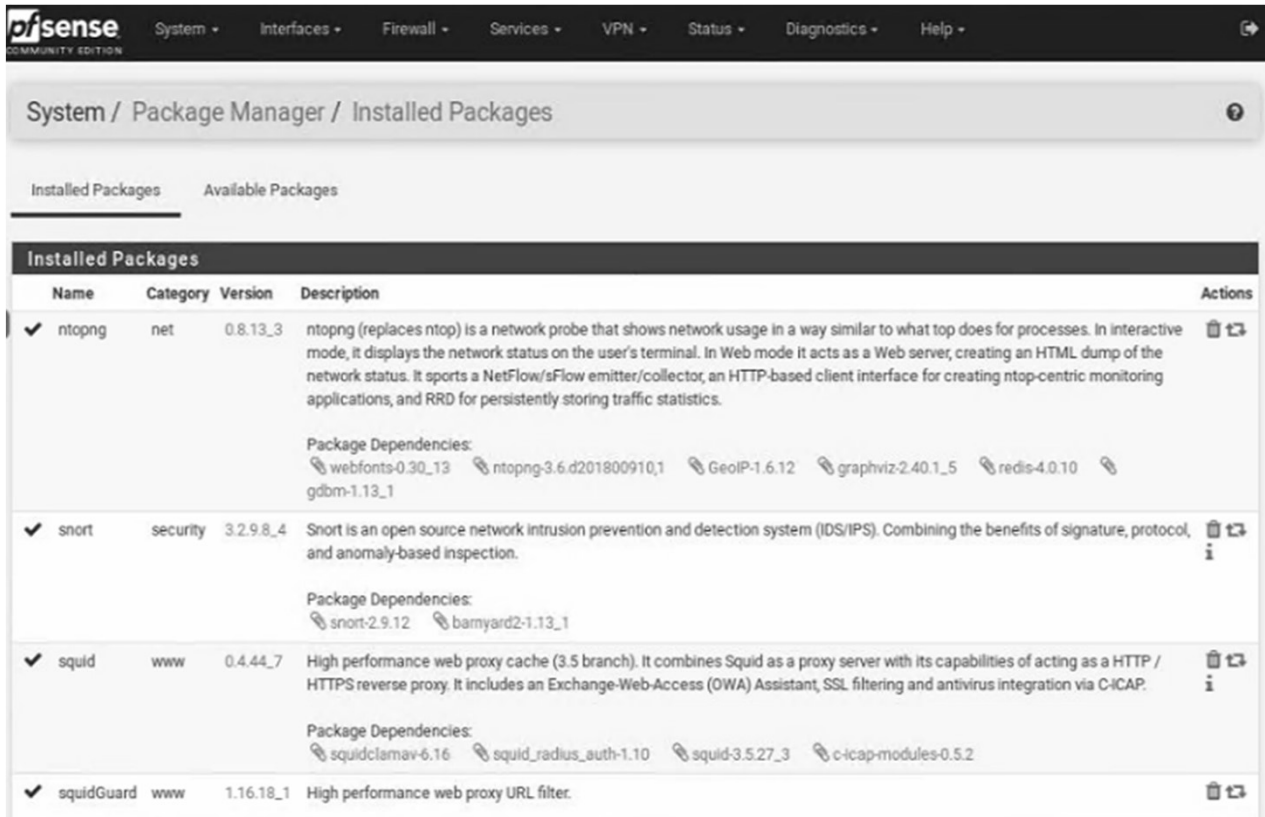
#### 메모:

- VyOS 라우터 대체 가능
- pfSense 이미지 다운로드: <https://www.pfsense.org/download/>
- 방화벽으로 사용 가능
- ISO 이미지 사용 (AMD64 64비트용)

# III. vUTM (pfSense)

## ❖ vUTM 개요

- **실습 설치 (pfSense)**
  - ✓ ntopng (플로우 모니터)
  - ✓ Snort (IDS/IPS)
  - ✓ Squid (프락시/웹 필터)
  - ✓ SquidGuard (웹 필터)



The screenshot shows the pfSense web interface, specifically the 'System / Package Manager / Installed Packages' page. It lists four installed packages with their details and dependencies.

Name	Category	Version	Description	Actions
✓ ntopng	net	0.8.13_3	ntopng (replaces ntop) is a network probe that shows network usage in a way similar to what top does for processes. In interactive mode, it displays the network status on the user's terminal. In Web mode it acts as a Web server, creating an HTML dump of the network status. It sports a NetFlow/sFlow emitter/collector, an HTTP-based client interface for creating ntop-centric monitoring applications, and RRD for persistently storing traffic statistics.  Package Dependencies: webfonts-0.30_13 ntopng-3.6.d201800910,1 GeoIP-1.6.12 graphviz-2.40.1_5 redis-4.0.10 gdbm-1.13_1	🗑️ ↻
✓ snort	security	3.2.9.8_4	Snort is an open source network intrusion prevention and detection system (IDS/IPS). Combining the benefits of signature, protocol, and anomaly-based inspection.  Package Dependencies: snort-2.9.12 barmyard2-1.13_1	🗑️ ↻ i
✓ squid	www	0.4.44_7	High performance web proxy cache (3.5 branch). It combines Squid as a proxy server with its capabilities of acting as a HTTP / HTTPS reverse proxy. It includes an Exchange-Web-Access (OWA) Assistant, SSL filtering and antivirus integration via C-ICAP.  Package Dependencies: squidclamav-6.16 squid_radius_auth-1.10 squid-3.5.27_3 c-icap-modules-0.5.2	🗑️ ↻ i
✓ squidGuard	www	1.16.18_1	High performance web proxy URL filter.	🗑️ ↻

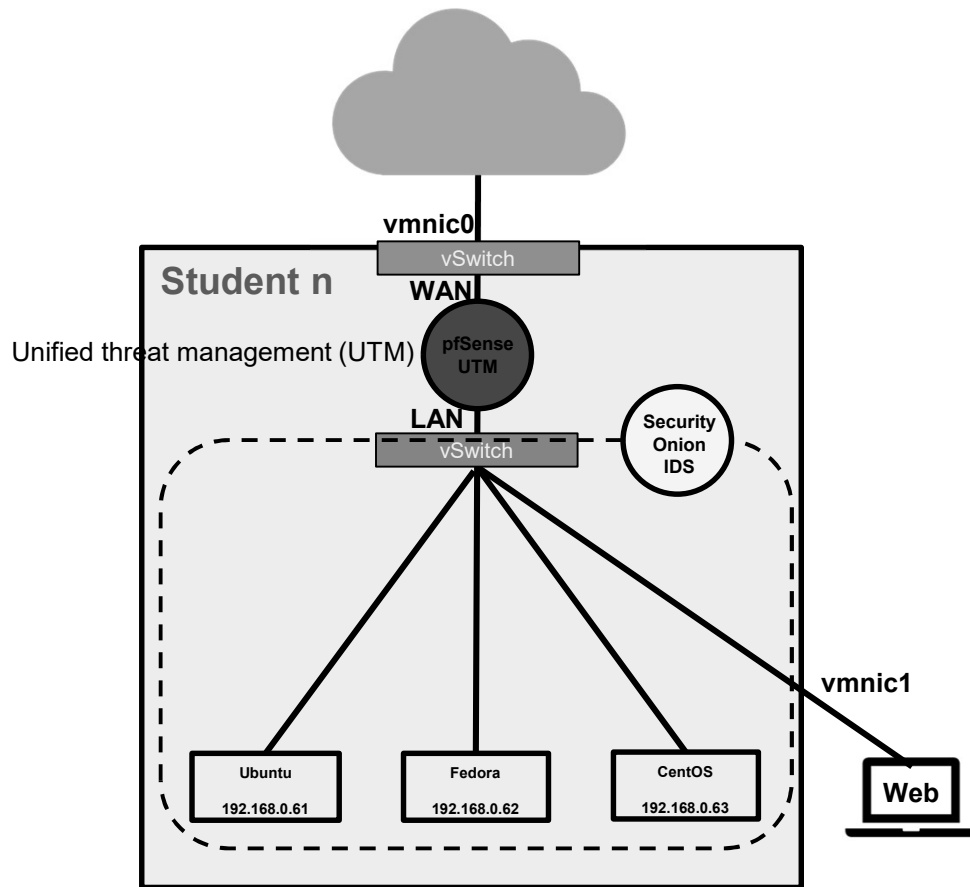
메모:



### III. vUTM (pfSense)

#### ❖ vUTM 'pfSense' 설치 환경

- ① 하이퍼바이저 내 인터넷용과 호스트 연결 스위치 2개 필요
- ② WAN은 인터넷, LAN은 호스트 연결 vSwitch 별도 생성
- ③ 센서 접속 부분의 스위치는 미리 기능 제공 세팅 필요
- ④ 설정을 위한 클라이언트는 VM 또는 유선랜 연결 PC 사용 (외부 유선랜 연결이 어려운 경우 하이퍼바이저에 웹으로 연결 사용)



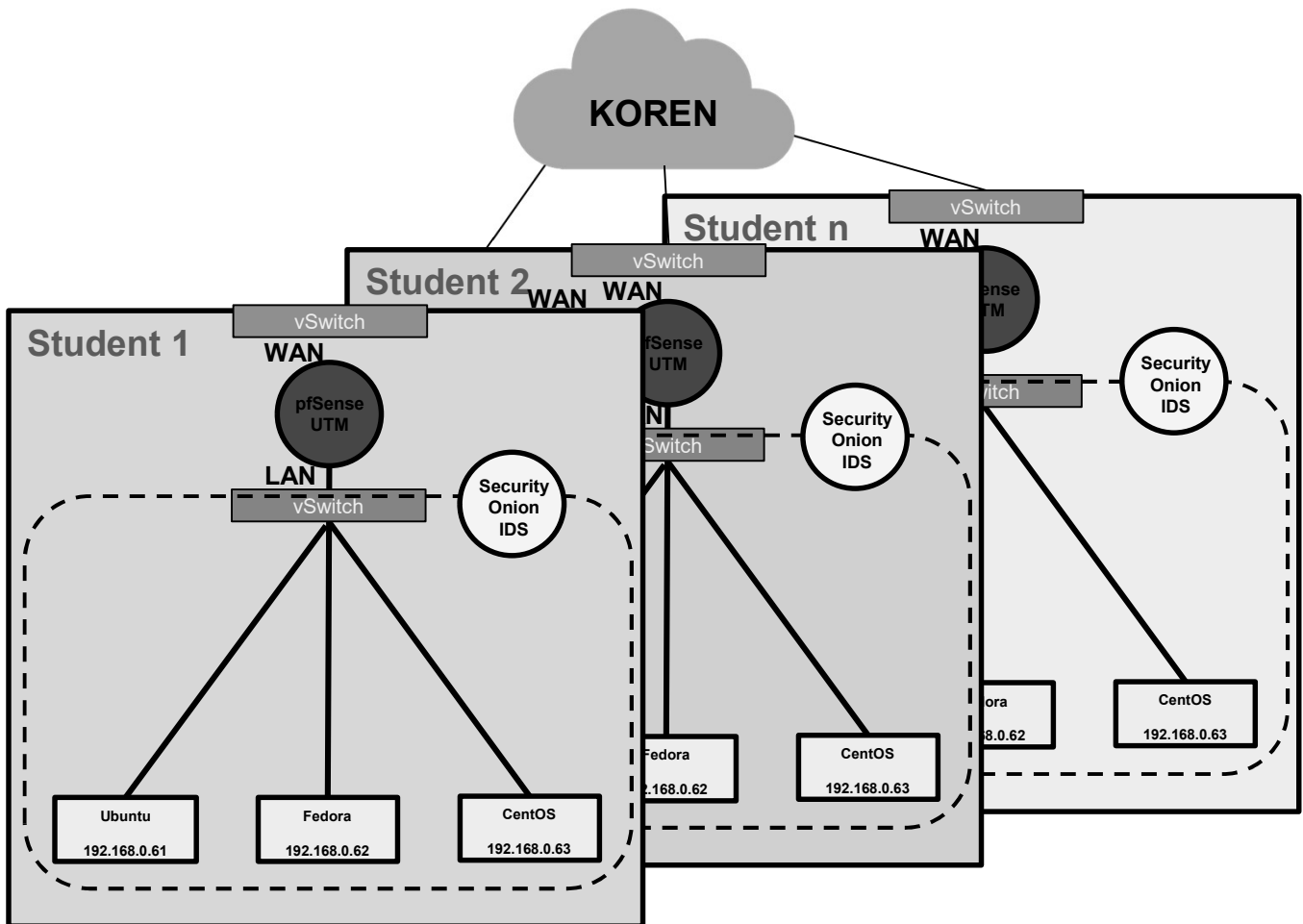
#### 메모:

- pfSense는 IDS/IPS, 방화벽, LB, 웹방화벽, NAT, DHCP 서버 등의 기능 제공

### III. vUTM (pfSense)

#### ❖ vUTM 'pfSense' 설치 환경

- ① WAN은 개인별 고정 IP주소 설정 권장
- ② LAN은 임의의 IP주소 설정 가능 (클라이언트를 위한 DHCP 서버 사용과 보안 기기를 위한 고정 IP 주소 사용)



#### 메모:

- Host는 Ubuntu Desktop과 Fedora Workstation ISO 이미지 제공
- vUTM과 vIDS용 ISO 이미지 사용 설치

### III. vUTM (pfSense)

#### ❖ pfSense 설치 준비

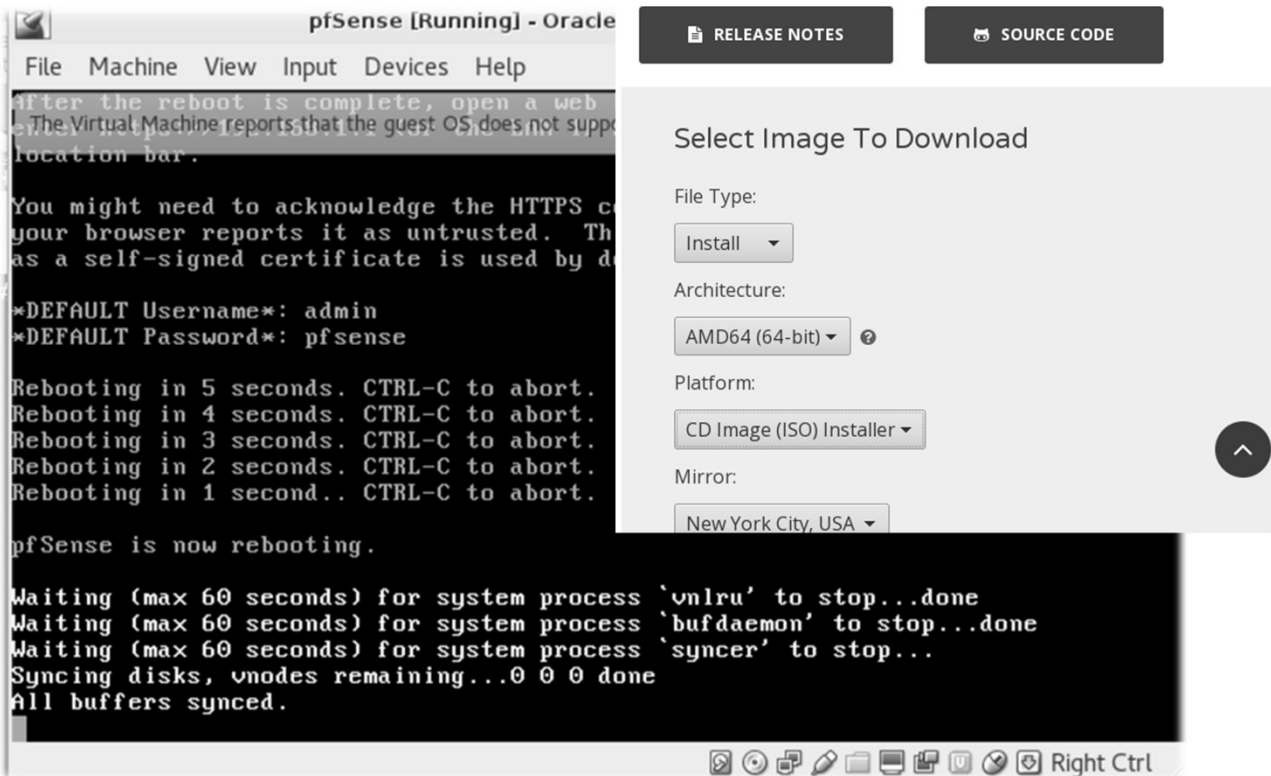
- pfSense 설치 (Type 1 또는 Type 2 하이퍼바이저 사용 가능)

① 다운로드: pfsense site ( <https://www.pfsense.org/> )

② 2개 이상 인터페이스 지정 (WAN/LAN)

③ ISO 이미지 다운로드 (또는 USB Memory)

④ pfSense 설치 (VirtualBox or VMWare 서버 or Type 1 하이퍼바이저)



#### 메모:

- pfSense 다운로드 주소: <https://www.pfsense.org/>
- ESXi 설치시 가상 스위치를 L2 Looping 을 방지하는 구성으로 해야함
- ESXi 설치시 동일 네트워크에 여러 사용자가 동시 접속 시 VyOS의 라우팅 사용 권장

# III. vUTM (pfSense)

## ❖ vUTM 'pfSense' 설치

- ① 이름과 운영체제 선택
- ② 자원 설정 (vCPU/vRAM/vHDD)
- ③ 설치



### 메모:

- pfSense 이미지 다운로드: <https://www.pfsense.org/download/>
- pfSense는 IDS/IPS, 방화벽, LB, 웹방화벽, NAT, DHCP 서버 등의 기능 제공
- ISO 이미지 사용 (AMD64 64비트용)

### III. vUTM (pfSense)

#### ❖ vUTM 'pfSense' 연결 설정

- 1) Assign Interfaces (LAN / WAN 설정)
- LAN / WAN MAC 주소 확인 @ 하이퍼바이저

```
pfSense
Starting syslog...done.
Starting CRON... done.
pfSense 2.4.4-RELEASE amd64 Thu Sep 20 09:03:12 EDT 2018
Bootup complete

FreeBSD/amd64 (pfSense.localdomain) (ttyv0)

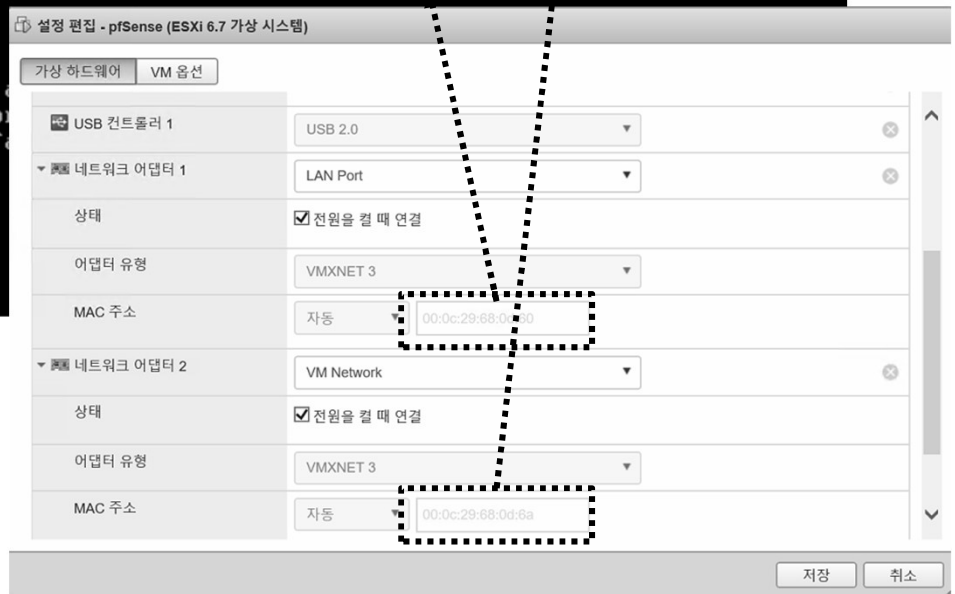
VMware Virtual Machine - Netgate Device ID: 9a48f6c634622f3c33a2

*** Welcome to pfSense 2.4.4-RELEASE (amd64) on pfSense ***

WAN (wan)      -> vmx1      -> v4/DHCP4: 192.168.1.189/24
LAN (lan)      -> vmx0      -> v4: 192.168.1.1/24

0) Logout (SSH only)
1) Assign Interfaces
2) Set interface(s) IP
3) Reset webConfigurator
4) Reset to factory default
5) Reboot system
6) Halt system
7) Ping host
8) Shell

Enter an option: |
```



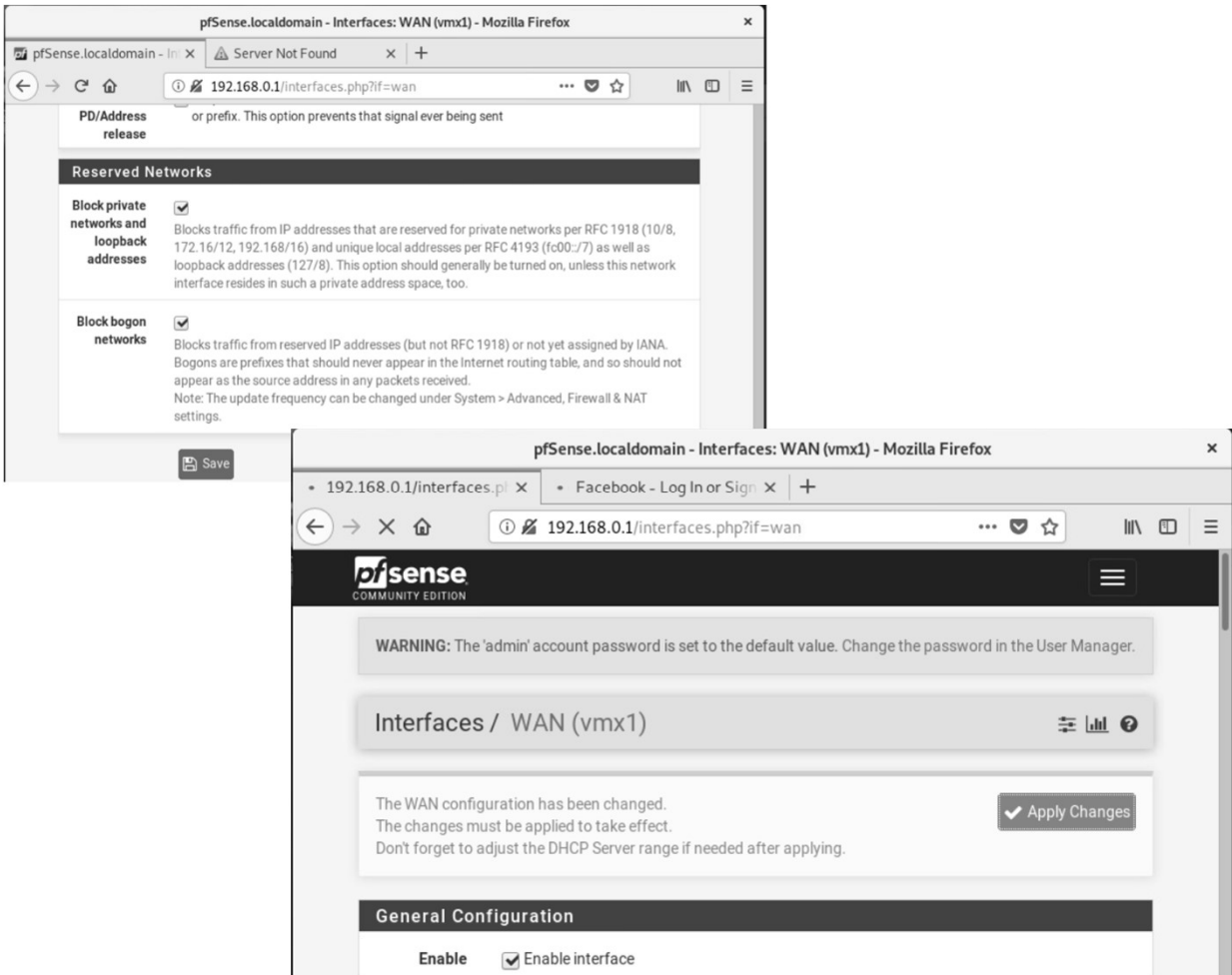
#### 메모:

- pfSense 이미지 다운로드: <https://www.pfsense.org/download/>
- pfSense는 IDS/IPS, 방화벽, LB, 웹방화벽, NAT, DHCP 서버 등의 기능 제공
- ISO 이미지 사용 (AMD64 64비트용)
- 초기 계정: admin / pfsense

### III. vUTM (pfSense)

#### ❖ vUTM 'pfSense' 설정 환경

- 초기 계정: admin / pfsense
- 사설 IP지원 설정 확인 (uncheck Block)
- Click Button “Apply Changes”




#### 메모:

- RFC1918: 인터넷 어드레싱 아키텍처에서 사설 IP 주소 공간을 이용하는 표준

RFC1918 이름	IP 주소 범위	주소 개수	클래스 내용	최대 사이더 블록 (서브넷 마스크)	호스트 ID 크기
24비트 블록	10.0.0.0 – 10.255.255.255	16,777,216	클래스 A 하나	10.0.0.0/8 (255.0.0.0)	24 비트
20비트 블록	172.16.0.0 – 172.31.255.255	1,048,576	16개의 인접 클래스 B	172.16.0.0/12 (255.240.0.0)	20 비트
16비트 블록	192.168.0.0 – 192.168.255.255	65,536	256개의 인접 클래스 C	192.168.0.0/16 (255.255.0.0)	16 비트

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

## IV. 리눅스 (Linux)

### ❖ Host 설치 환경

#### ① ISO 파일 선택 # Type 1 하이퍼바이저 설치 시

- CentOS7 minimal (실습 자료 포함)
- Ubuntu Desktop 18.04 (시연)
- Fedora Workstation 29 (선택)
- **Ubuntu Server 16.04** (Hyperledger, OpenStack, OpenFaaS 설치 가능)

#### ② ISO 파일 Upload

#### ③ Ubuntu Desktop과 Fedora Workstation 29는 시연으로 진행

#### ④ Ubuntu Server 는 설정 순서 제공

VM Name	Host Name	IP Address for 1	Interface 1 Name	Interface 2 Name
Master	master60	192.168.0.60		
Worker01	worker61	192.168.0.61		
Worker02	worker62	192.168.0.62		
Worker03	worker63	192.168.0.63		

#### 메모:

- Type 2 하이퍼바이저에서 VM 설치방법 1: 우분투(Ubuntu Server/Desktop) OVA 제공
- Type 2 하이퍼바이저에서 VM 설치방법 2: 우분투(Ubuntu Server/Desktop) ISO 제공 설치
- VMware Standalone Converter 사용하여 배포 가능
- 루트계정 활성화: `sudo su`



## IV. 리눅스 (Linux)

### ❖ Ubuntu Server 16.04 Installation @ vSphere

- ① ESXi 6.x 사용
- ② vCPU 2개, vRAM 4GB, 48 GB Storage (Thin)
- ③ 다운로드한 Ubuntu Server 16.04 ISO 파일 사용 설치

설정 편집 - Ansible (ESXi 6.7 가상 시스템)

가상 하드웨어 VM 옵션

하드 디스크 추가 네트워크 어댑터 추가 기타 디바이스 추가

CPU	2	
메모리	4096	MB
하드 디스크 1	48	GB
SCSI 컨트롤러 0	VMware Paravirtual	
SATA 컨트롤러 0		
USB 컨트롤러 1	USB 2.0	
네트워크 어댑터 1	VM Network	<input checked="" type="checkbox"/> 연결
CD/DVD 드라이브 1	데이터스토어 ISO 파일	<input checked="" type="checkbox"/> 연결
비디오 카드	사용자 지정 설정 지정	

저장 취소

#### 메모:

- VM 설치방법 1: 우분투(Ubuntu Server/Desktop) OVA 제공
- VM 설치방법 2: 우분투(Ubuntu Server/Desktop) ISO 제공 설치

## IV. 리눅스 (Linux)

### ❖ Ubuntu Server 16.04 Installation

- ① USB Booting 선택 # Bare-Metal
- ② ISO 파일 선택 # 4 GB RAM / 32 GB Storage
- ③ 언어 선택 'Korean (한국어)' and 'Continue'
- ④ 선택 'Install Ubuntu Server'



#### 메모:

- 멀티클러스터를 위해 이미지 복제는 VMware Standalone Converter 사용하여 배포
- 루트계정 활성화: `sudo passwd root`

# IV. 리눅스 (Linux)

## ❖ Ubuntu Server 16.04 Installation

- ① Full Name 'jalsb'
- ② User name 'jslab'
- ③ Password 'jslab123'



james@jslab.kr

메모:

# IV. 리눅스 (Linux)

## ❖ Ubuntu Server 16.04 Installation

- ① No automatic updates
- ② OpenSSH server
- ③ User name 'jslab'



james@jslab.kr

메모:

## IV. 리눅스 (Linux)

### ❖ Ubuntu Server 16.04 Installation (선택)

- ① **ip link show** # Check Interfaces
- ② **Static IP Address Setting**
- ③ **Host Name Setting**
- ④ **sudo reboot** # 인터페이스 생성 확인 후 재 리부팅 필요

#### - 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 vi /etc/hosts  
  
cntl+o → enter → cntl+x
```

#### - 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 1.1.1.1  
  
sudo /etc/init.d/networking restart (or reboot)
```

#### - Root 계정 생성 -

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

#### - Putty to VyOS for sshd -

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

#### 메모:

- Ubuntu Server 루트계정 활성화: sudo passwd root
- VM 이미지 Import 시 네트워크 인터페이스 확인 위한 명령어 'ip link show'
- Ping time 비교 1.1.1.1 vs. 8.8.8.8
- Root 계정으로 실행 필요시 (sudo 사용 일반 계정은 실행하지 못함)  
루트계정 활성화: sudo passwd root

## IV. 리눅스 (Linux)

### ❖ Static IP for WiFi (Ubuntu Desktop18.04)

- OVS (Open vSwitch) Mirroring (2.8.0)

#### 1. ip link show

```
james@ubuntu18:/etc/netplan$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 00:aa:2a:e8:34:21 brd ff:ff:ff:ff:ff:ff
4: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 00:aa:2a:e8:34:22 brd ff:ff:ff:ff:ff:ff
5: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 00:aa:2a:e8:34:23 brd ff:ff:ff:ff:ff:ff
7: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 96:be:89:0f:df:b5 brd ff:ff:ff:ff:ff:ff
8: ovs1qotom: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:aa:2a:e8:34:20 brd ff:ff:ff:ff:ff:ff
9: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default
    link/ether 02:42:ee:0f:69:c6 brd ff:ff:ff:ff:ff:ff
10: wlx742f68923076: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 74:2f:68:92:30:76 brd ff:ff:ff:ff:ff:ff
12: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel master ovs-system state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:aa:2a:e8:34:20 brd ff:ff:ff:ff:ff:ff
james@ubuntu18:/etc/netplan$
```

#### 2. cd /etc/netplan

#### 3. sudo nano 01-network-manager-all.yaml

```
network:
  version: 2
  renderer: networkd
  wifis:
    wlx742f68923076:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.0.18/24,]
      gateway4: 192.168.0.1
      nameservers:
        search: [vsphere.local]
        addresses: [192.168.0.1, 8.8.8.8]
      access-points:
        Tech-Support:
          password: 12345*****
```

#### 4. sudo netplan generate

#### 5. sudo netplan apply

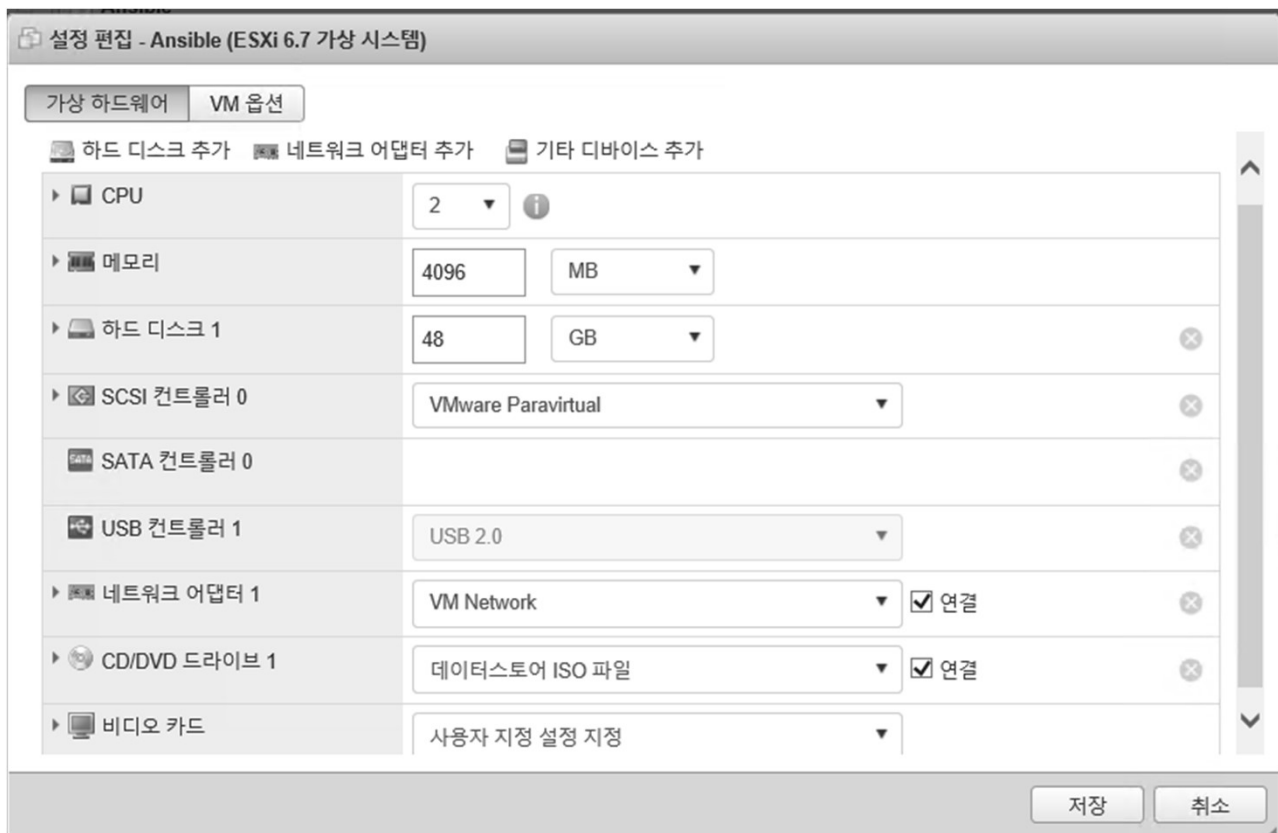
#### 메모:

- ❖ <https://www.tecmint.com/configure-network-static-ip-address-in-ubuntu/>

## IV. 리눅스 (Linux)

### ❖ CentOS7 Installation @ vSphere

- ① ESXi 6.7 사용
- ② vCPU 2개, vRAM 4GB, 48 GB Storage (Thin)
- ③ 다운로드한 CentOS7 Minimal ISO 파일 사용 설치



#### 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](http://ftp.kaist.ac.kr/CentOS/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso)

## IV. 리눅스 (Linux)

---

### ❖ CentOS7 Installation @ vSphere

- ① VM 전원 켜기
- ② Install CentOS 7
- ③ 시연 따라하기



#### 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](http://ftp.kaist.ac.kr/CentOS/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso)
- 계정 (예) : root/ password



# IV. 리눅스 (Linux)

## ❖ CentOS7 네트워크 설정

- ① **yum update** # Patch Update
- ② **nmtui** # IP 주소 설정 192.168.1.10 (Tab 키 사용 이동)
- ③ **ip add** # 설정한 IP 주소 확인 @ Terminal
- ④ **echo "nameserver 1.1.1.1">> /etc/resolv.conf** # 선택
- ⑤ **vi /etc/resolv.conf** # dns 주소 1.1.1.1 추가 확인



nmtui 명령어 수행 화면



Activate a connection



IP 주소 설정

```
[root@ansible ~]# ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
   link/ether 00:0c:29:ba:62:65 brd ff:ff:ff:ff:ff:ff
   inet 192.168.1.10/24 brd 192.168.1.255 scope global noprefixroute ens192
       valid_lft forever preferred_lft forever
   inet6 fe80::1c1b:c480:f0df:ea31/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
   inet6 fe80::9a43:9ba3:5dc5:a5cc/64 scope link tentative noprefixroute dadfailed
       valid_lft forever preferred_lft forever
[root@ansible ~]#
```

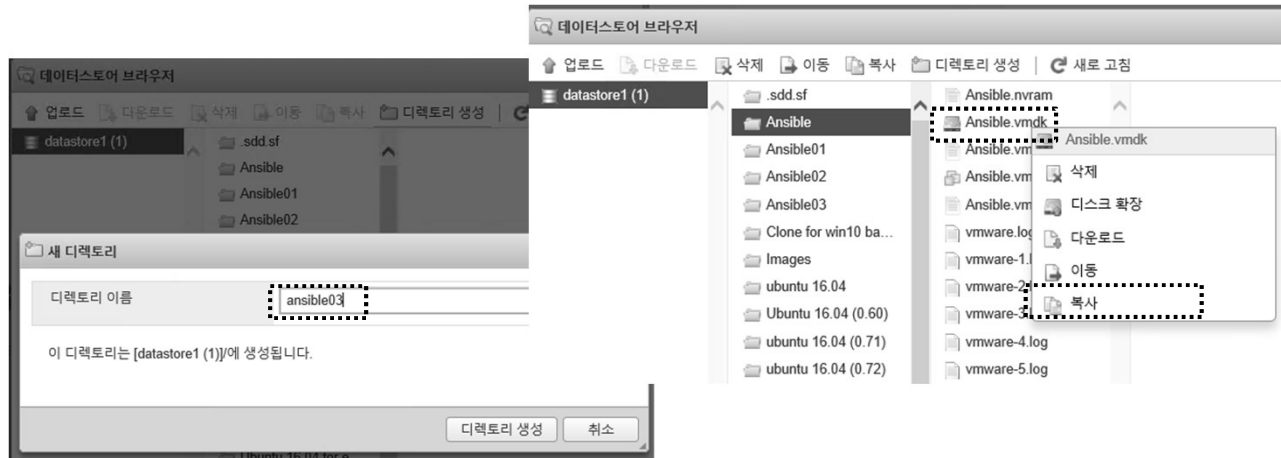
### 메모:

- 접속 후 계정 사용하여 로그인 계정 (예) : root/ password

## IV. 리눅스 (Linux)

### ❖ Cloning Host @ vSphere (예: CentOS 7)

- ① `hostnamectl set-hostname master` # @ master
- ② `su -`
- ③ `poweroff` # master
- ④ 복제를 위해 데이터스토어에서 디렉토리 생성 (3개)
- ⑤ `master.vmdk / master.vmx` 파일 선택후 디렉토리에 복제



VM Name	Host Name	IP Address for 1	Interface 1 Name	Interface 2 Name
Master	master60	192.168.0.60	ens33	ens35 1.60
Worker01	worker61	192.168.0.61	ens33	ens35 1.61
Worker02	worker62	192.168.0.62	ens33	ens35 1.62
Worker03	worker63	192.168.0.63	ens33	ens35 1.63

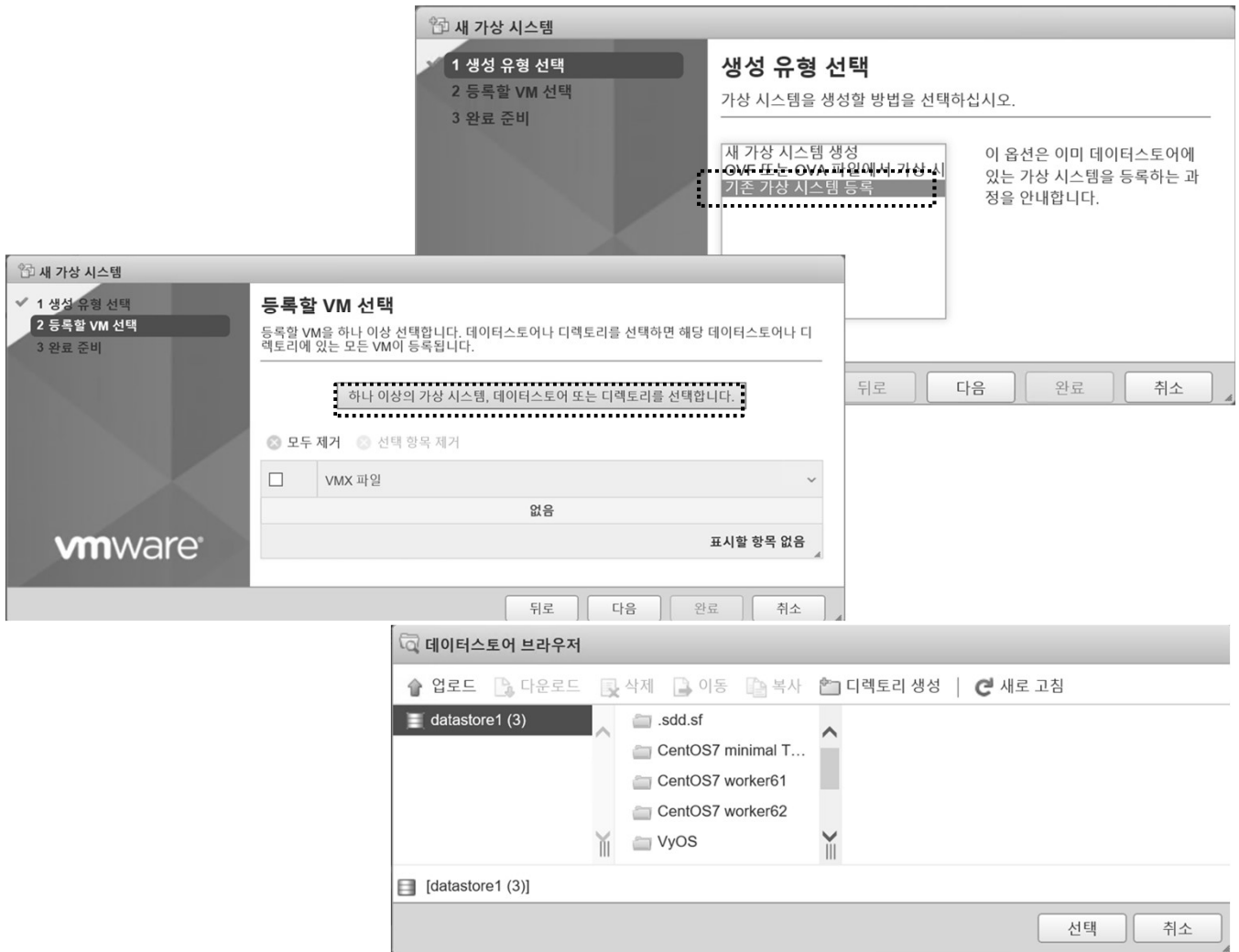
#### 메모:

- VMware vCenter Converter Standalone Client 사용 가능
- 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](http://ftp.kaist.ac.kr/CentOS/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso)

## IV. 리눅스 (Linux)

### ❖ Cloning Host @ vSphere (예: CentOS 7)

- ① VM 등록 @ 새가상 시스템 (3개)
- ② 기존 가상 시스템 등록
- ③ 디렉토리 선택



#### 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](http://ftp.kaist.ac.kr/CentOS/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso)

## IV. 리눅스 (Linux)

### ❖ Cloning Host @ vSphere (예: CentOS 7)

- ① VM 등록 (3개)
- ② 등록 VM 확인후 이름 변경
- ③ 복제 VM 실행 시 '복사함' 확인 (질문?)

설정 편집 - CentOS7 worker61 (ESXi 6.7 가상 시스템)

가상 하드웨어 VM 옵션

▼ 일반 옵션	
VM 이름:	CentOS7 worker61
VM 구성 파일	[datastore1 (3)] CentOS7 worker61/CentOS7 minimal
VM 작동 위치	[datastore1 (3)] CentOS7 worker61
게스트 운영 체제	Linux ▼
게스트 운영 체제 버전	CentOS 7(64비트) ▼
▶ VMware Remote Console 옵션	<input type="checkbox"/> 마지막 원격 사용자의 연결이 끊기면 게스트 운영 체제 잠금
▶ VMware Tools	VMware Tools 설정을 보려면 확장
▶ 전원 관리	전원 관리 설정을 보려면 확장
▶ 부팅 옵션	부팅 옵션을 보려면 확장

저장 취소

#### 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](http://ftp.kaist.ac.kr/CentOS/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso)

## IV. 리눅스 (Linux)

### ❖ Cloning Host @ vSphere (예: CentOS 7)


- 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 **cat /etc/resolv.conf** # dns 주소 1.1.1.1 추가 확인

VM Name	Host Name	IP Address for 1	Interface 1 Name	Interface 2 Name
Master	master60	192.168.0.60	ens33	ens35 1.60
Worker01	worker61	192.168.0.61	ens33	ens35 1.61
Worker02	worker62	192.168.0.62	ens33	ens35 1.62
Worker03	worker63	192.168.0.63	ens33	ens35 1.63

#### 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](http://ftp.kaist.ac.kr/CentOS/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso)
- SuperPutty 사용 가능

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

## V. 컨테이너 (Docker)

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

- ① **sudo apt install docker.io** # Ubuntu Server 16.04
- ② **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:~$
```

- **sudo curl -fsSL https://get.docker.com/ | sh** # latest (선택)
- **sudo usermod -aG docker jslab**

#### 메모:

- sudo apt install docker.io (Ubuntu 권장 @ Ubuntu Server 16.04)
- 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

## V. 컨테이너 (Docker)

---

### ❖ 도커(Docker) 설치/실행 by Vendor Docker (선택)

- 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:

- <https://docs.docker.com/install/linux/docker-ce/ubuntu/> (Docker 사 권장)
- '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`



## V. 컨테이너 (Docker)

---

### ❖ 각 호스트에 도커(Docker) 설치 @ CentOS7 (선택)

- ① **yum update**
- ② **sudo setenforce 0**      # Change SELinux mode to permissive
- ③ **sudo sed -i**  
**'s/^SELINUX=enforcing\$/SELINUX=permissive/'**  
**/etc/selinux/config**
- ④ **sudo yum update -y && yum install -y yum-utils device-**  
**mapper-persistent-data lvm2**
- ⑤ **sudo yum-config-manager --add-repo**  
**https://download.docker.com/linux/centos/docker-ce.repo**
- ⑥ **sudo yum install -y docker-ce**
- ⑦ **sudo systemctl start docker && sudo systemctl enable**  
**docker**
- ⑧ **docker version**
- ⑨ # Configure Firewall for Swarm
  - ✓ **firewall-cmd --permanent --add-port=2376/tcp**
  - ✓ **firewall-cmd --permanent --add-port=2377/tcp**
  - ✓ **firewall-cmd --permanent --add-port=7946/tcp**
  - ✓ **firewall-cmd --permanent --add-port=80/tcp**
  - ✓ **firewall-cmd --permanent --add-port=7946/udp**
  - ✓ **firewall-cmd --permanent --add-port=4789/udp**
- ⑩ **firewall-cmd --reload**
- ⑪ **systemctl restart docker**

#### 메모:

- 2019년 10월 현재 Docker 19.03.2 확인
- Swarm 사용 Web service를 위한 방화벽 정책 추가 필요

## V. 컨테이너 (Docker)

### ❖ 각 호스트에 도커(Docker) 설치 @ CentOS7 (선택)

- ① yum makecache fast
- ② yum install -y epel-release
- ③ yum provides docker
- ④ Loaded plugins: fastestmirror
- ⑤ Loading mirror speeds from cached hostfile
  - ✓ \* base: ftp.nara.wide.ad.jp
  - ✓ \* epel: ftp.kddilabs.jp
  - ✓ \* extras: ftp.iij.ad.jp
  - ✓ \* updates: ftp.nara.wide.ad.jp
- ⑥ **docker-1.12.6-71.git3e8e77d.el7.centos.1.x86\_64:**  
Automates deployment of containerized applications
- ⑦ Repo : extras ...
- ⑧ yum install -y docker-1.12.6-71.git3e8e77d.el7.centos.1.x86\_64 chrony # 표시된 리스트에서 확인
- ⑨ systemctl stop firewalld && systemctl disable firewalld
- ⑩ systemctl enable docker && systemctl start docker

```
# K8s 방화벽 세팅 권장 방법
1. exec bash
2. setenforce 0
3. sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux # Disable SELinux & setup firewall rules
4. firewall-cmd --permanent --add-port=6443/tcp
5. firewall-cmd --permanent --add-port=2379-2380/tcp
6. firewall-cmd --permanent --add-port=10250/tcp # Master and Worker
7. firewall-cmd --permanent --add-port=10251/tcp
8. firewall-cmd --permanent --add-port=10252/tcp
9. firewall-cmd --permanent --add-port=10255/tcp # Master and Worker
10. firewall-cmd --reload
```

#### 메모:

- yum provides docker 로 확인하여 표시되는 docker 릴리즈 버전 중에서 원하는 것을 선택
- 2018년 2월 현재 K8s는 Docker 1.12.6 까지 테스트 확인
- curl -fsSL https://get.docker.com/ | sh # @ General for New
- sudo apt install docker.io # @ Ubuntu
- yum install -y docker-1.13.1-53.git774336d.el7.centos.x86\_64 chrony # 23 Apr. 2018

## V. 컨테이너 (Docker)

---

### ❖ 각 호스트에 도커(Docker) 설치 @ CentOS7 (선택)

- ① `yum -y install docker`
- ② `systemctl start docker`
- ③ `systemctl enable docker`
- ④ `systemctl status docker`
- ⑤ `docker --version`                      # or `docker version`

메모:

## V. 컨테이너 (Docker)

---

### ❖ Check Docker Installation (예: CentOS 7)

#### ① docker info

```
[root@master ~]# docker info
Client:
 Debug Mode: false

Server:
 Containers: 1
  Running: 1
  Paused: 0
  Stopped: 0
 Images: 1
 Server Version: 19.03.2
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 894b81a4b802e4eb2a91d1ce216b8817763c29fb
 runc version: 425e105d5a03fabd737a126ad93d62a9eede87f
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 3.10.0-1062.1.2.el7.x86_64
 Operating System: CentOS Linux 7 (Core)
 OSType: linux
 Architecture: x86_64
 CPUs: 2
 Total Memory: 3.701GiB
 Name: master.localdomain
 ID: OWVZ:YPZV:Z454:7DCP:3AFM:3346:ZGIR:IBI3:GW60:BKBU:G76K:SWIN
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

WARNING: bridge-nf-call-ip6tables is disabled
```

## V. 컨테이너 (Docker)

---

### ❖ Check Docker Installation (예: CentOS 7)

#### ① docker version

```
[root@master ~]# docker version
Client: Docker Engine - Community
 Version:      19.03.2
 API version:  1.40
 Go version:   go1.12.8
 Git commit:   6a30dfc
 Built:        Thu Aug 29 05:28:55 2019
 OS/Arch:      linux/amd64
 Experimental: false

Server: Docker Engine - Community
 Engine:
  Version:      19.03.2
  API version:  1.40 (minimum version 1.12)
  Go version:   go1.12.8
  Git commit:   6a30dfc
  Built:        Thu Aug 29 05:27:34 2019
  OS/Arch:      linux/amd64
  Experimental: false
 containerd:
  Version:      1.2.6
  GitCommit:    894b81a4b802e4eb2a91d1ce216b8817763c29fb
 runc:
  Version:      1.0.0-rc8
  GitCommit:    425e105d5a03fabd737a126ad93d62a9eeede87f
 docker-init:
  Version:      0.18.0
  GitCommit:    fec3683
[root@master ~]#
```

# V. 컨테이너 (Docker)

## ❖ Check Docker Installation (예: CentOS 7)

① **docker** # checking CLI

```
[root@master ~]# docker
```

Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:

--config string	Location of client config files (default "/root/.docker")
-D, --debug	Enable debug mode
-H, --host list	Daemon socket(s) to connect to
-l, --log-level string	Set the logging level ("debug" "info" "warn" "error" "fatal") (default "info")
--tls	Use TLS; implied by --tlsverify
--tlscacert string	Trust certs signed only by this CA (default "/root/.docker/ca.pem")
--tlscert string	Path to TLS certificate file (default "/root/.docker/cert.pem")
--tlskey string	Path to TLS key file (default "/root/.docker/key.pem")
--tlsverify	Use TLS and verify the remote
-v, --version	Print version information and quit

Management Commands:

builder	Manage builds
config	Manage Docker configs
container	Manage containers
engine	Manage the docker engine
image	Manage images
network	Manage networks
node	Manage Swarm nodes
plugin	Manage plugins
secret	Manage Docker secrets
service	Manage services
stack	Manage Docker stacks
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
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

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
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

# V. 컨테이너 (Docker)

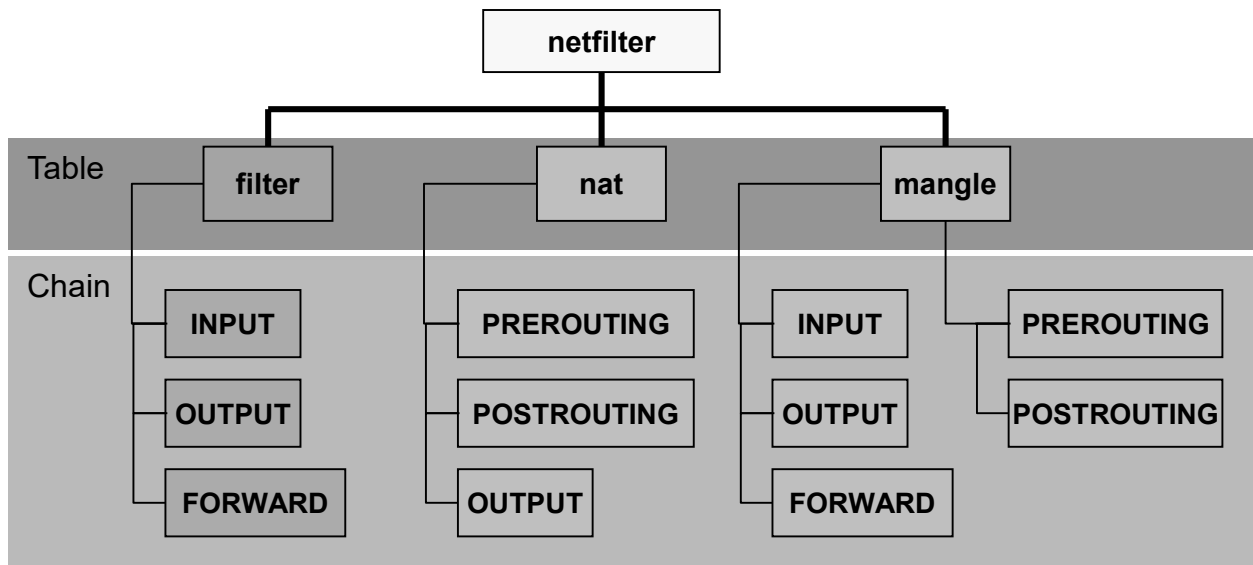
## ❖ Check Docker Installation

### ① iptables – Tables and Chains ( 3 built -in tables )

- Filter
- NAT
- Mangle (TTL or TOS 값을 변경하거나 매칭 시에 사용)

### ② 각 테이블은 체인 세트가 있으며, 각체인에는 룰 세트를 할당 할 수 있음

### ③ Tables → Chains → Rules



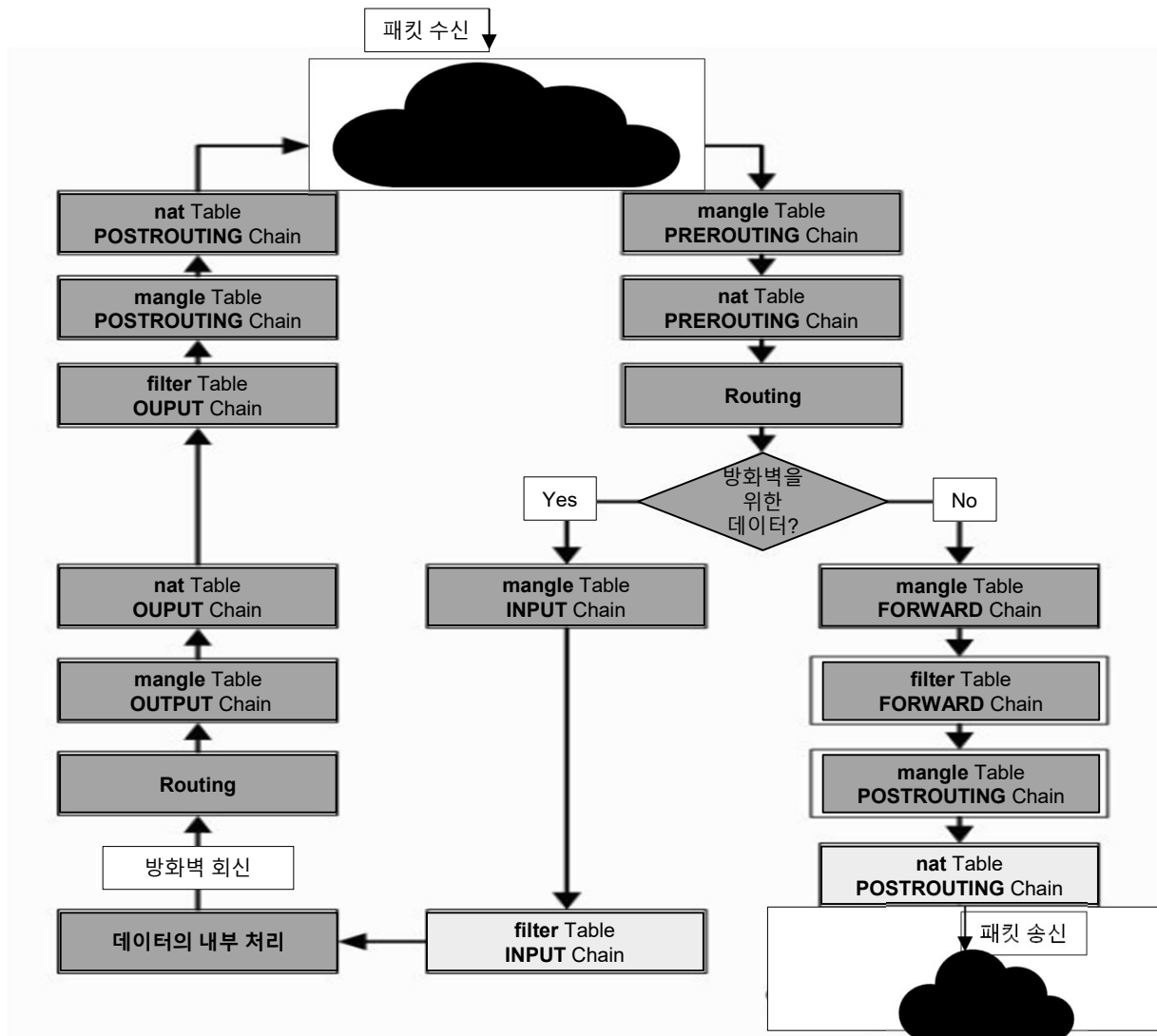
#### 메모:

- Post Routing (snat)
- Pre Routing
  - ✓ MASQUERADE Port (1 to 65535)
  - ✓ Masquerade (Port Address Translation (PAT))
  - ✓ Port Address Table / IP Translation

# V. 컨테이너 (Docker)

## ❖ 계층 분리에 사용하는 Linux의 IPTable

### ① iptables -t nat -L -n



### 메모:

- -t, --table table
- -L, --list [chain]
- -n, --numeric
- -n — Displays IP addresses and port numbers in numeric format, rather than the default hostname and network service format.



## V. 컨테이너 (Docker)

### ❖ Check Docker Installation

- ① `docker network ls`
- ② `docker network inspect bridge`

```
[root@master ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
5e395d3047db       bridge             bridge              local
d07b05617fe2       host               host                local
a5f7e9e9a037       none               null                local
[root@master ~]# docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "5e395d3047dbab93e7ccafb3c9beb30c1b2d66eaa20ae44ada8bd7594e5b9024",
    "Created": "2019-05-23T08:40:08.758643877-04:00",
    "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": {},
    "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@master ~]#
```



# V. 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

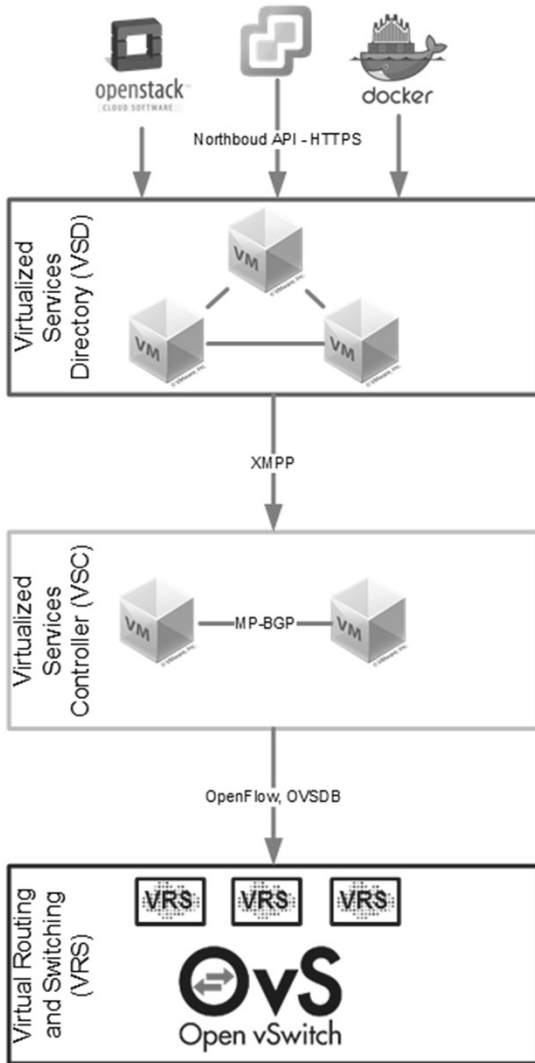
# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

# VI. OVS (Open vSwitch)

## ❖ Vendor Architecture (예: VMware)

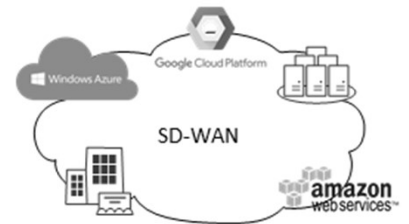
- VMware NSX-T 와 KVM 연동
- Kubernetes 연동



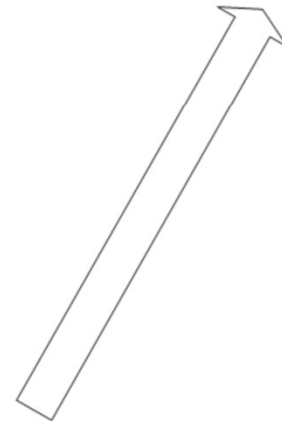
- Management
- GUI
- RESTful API
- Multi-Tenant
- Cluster
- Quorum

- Control Plane
- SSH
- SNMP
- NETCONF

- VXLAN
- Service Insertion
- Security
- Policy Routing
- QoS
- FW Filters – L4 Stateful



- IPsec
- VRS -> NSG
- Branch
- Public Cloud



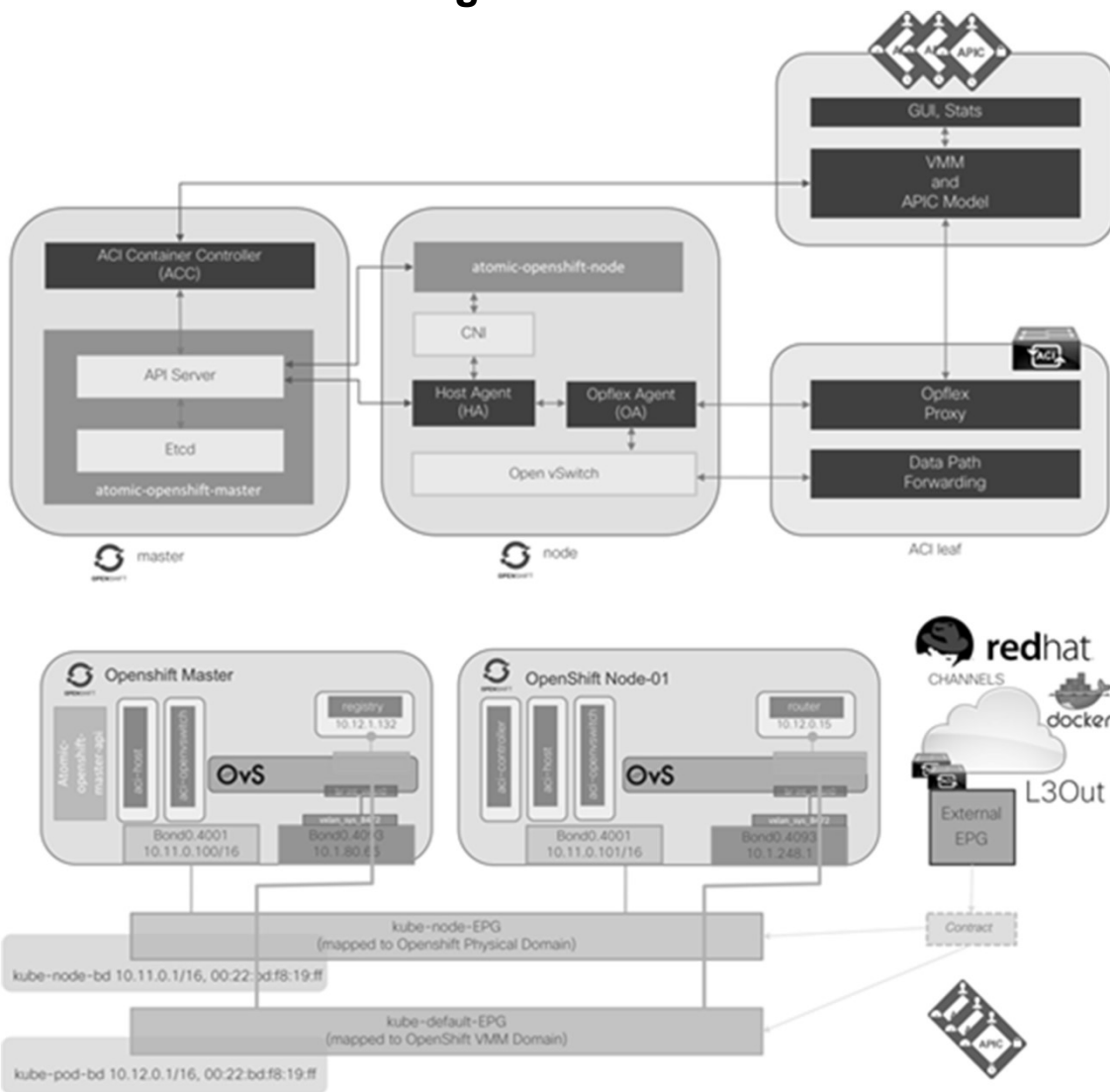
### 메모:

- <https://thedataplumber.net/nsx-t-vs-nsx-v-and-a-little-bit-of-nuage-vsp/>
- <https://blogs.vmware.com/networkvirtualization/files/2017/09/Figure1-Topology.png>
- <http://www.routetocloud.com/2017/10/introduction-to-nsx-and-kubernetes/>

# VI. OVS (Open vSwitch)

## ❖ Vendor Architecture (예: Cisco)

- Cisco Application Policy Infrastructure Controller (APIC)
- General ACI CNI Plugin architecture

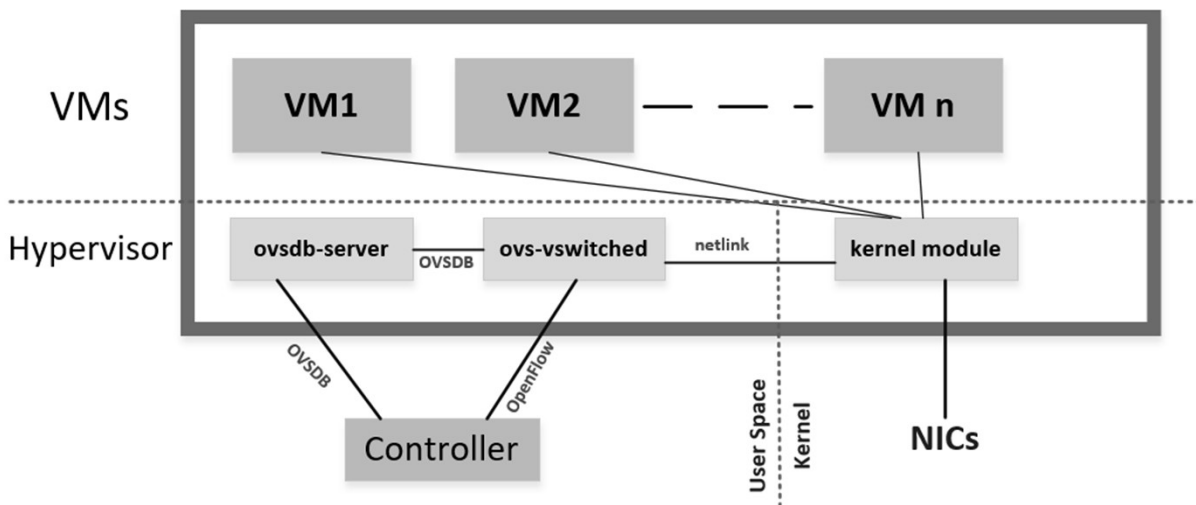
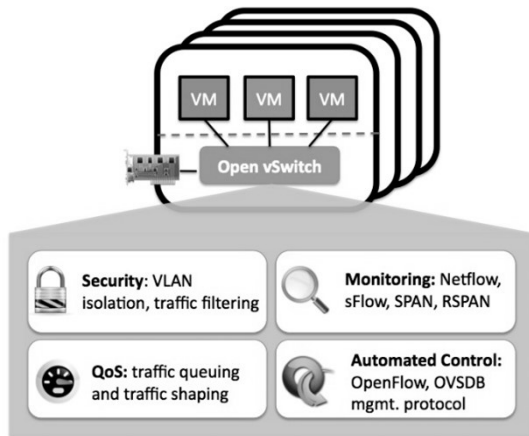


### 메모:

- [https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/white\\_papers/Cisco-ACI-CNI-Plugin-for-OpenShift-Architecture-and-Design-Guide.html](https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/white_papers/Cisco-ACI-CNI-Plugin-for-OpenShift-Architecture-and-Design-Guide.html)

# VI. OVS (Open vSwitch)

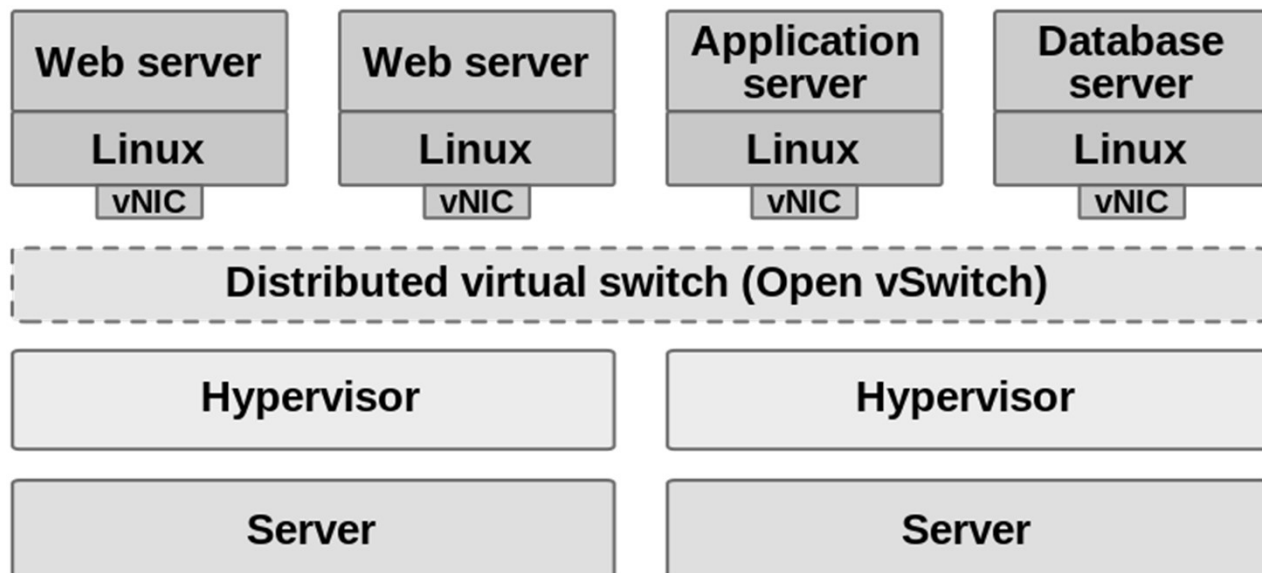
## ❖ Open vSwitch Architecture



메모:

## VI. OVS (Open vSwitch)

### ❖ Distributed Open vSwitch Instance



- Visibility into inter-VM communication via NetFlow, sFlow(R), IPFIX, SPAN, RSPAN, and GRE-tunnelled mirrors
- LACP (IEEE 802.1AX-2008)
- Standard 802.1Q VLAN model with trunking
- Multicast snooping
- IETF Auto-Attach SPBM and rudimentary required LLDP support
- BFD and 802.1ag link monitoring
- STP (IEEE 802.1D-1998) and RSTP (IEEE 802.1D-2004)
- Fine-grained QoS control
- Support for HFSC qdisc
- Per VM interface traffic policing
- NIC bonding with source-MAC load balancing, active backup, and L4 hashing
- OpenFlow protocol support (including many extensions for virtualization)
- IPv6 support
- Multiple tunnelling protocols (GRE, VXLAN, STT, and Geneve, with IPsec support)
- Remote configuration protocol with C and Python bindings
- Kernel and user-space forwarding engine options
- Multi-table forwarding pipeline with flow-caching engine
- Forwarding layer abstraction to ease porting to new software and hardware platforms

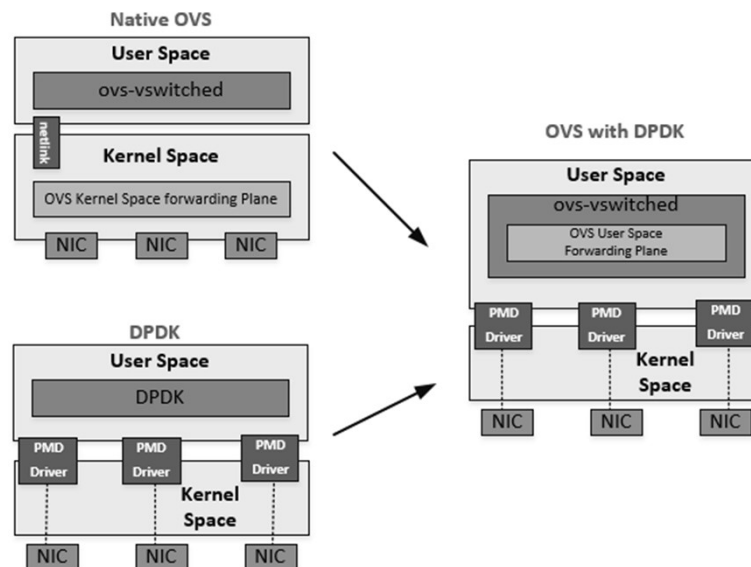
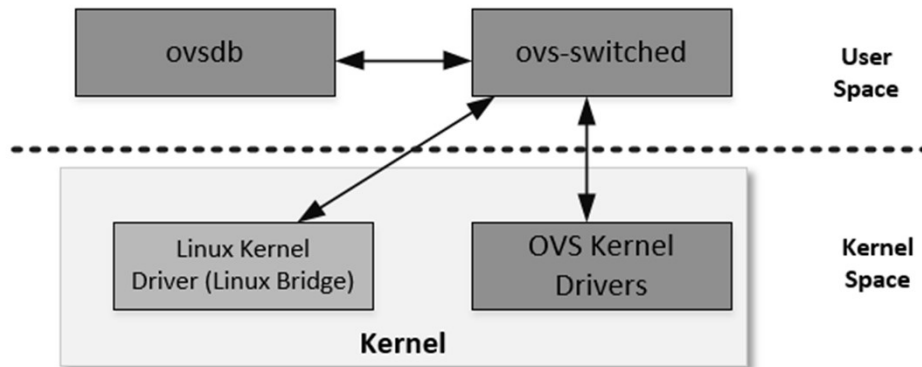
메모:



## VI. OVS (Open vSwitch)

### ❖ Integration of DPDK Data Plane with Open vSwitch

- DPDK는 커널 바이패스로 Latency를 빠르게하고, CPU를 복수로 할당하여 성능을 증가



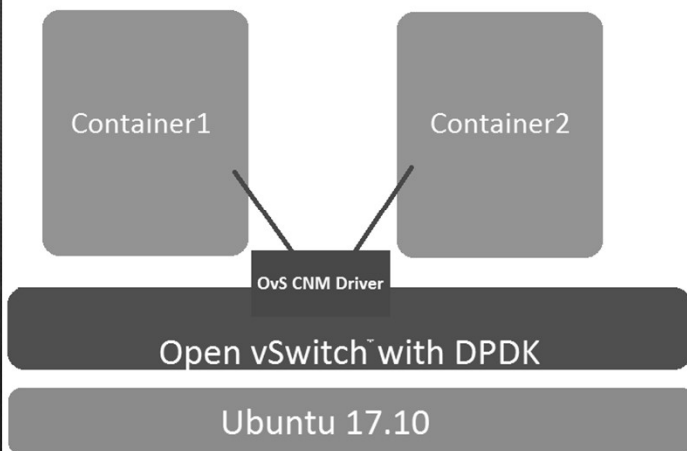
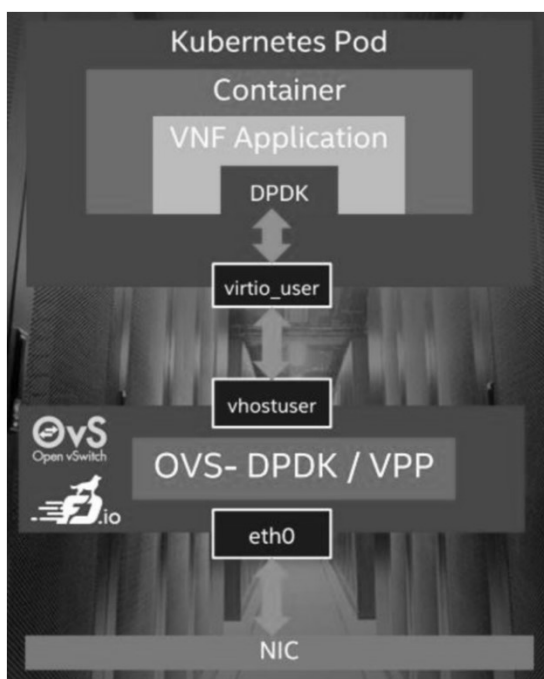
#### 메모:

- <https://software.intel.com/en-us/articles/set-up-open-vswitch-with-dpdk-on-ubuntu-server>
- sudo apt-get install openvswitch-switch-dpdk
- sudo update-alternatives --set ovs-vswitchd /usr/lib/openvswitch-switch-dpdk/ovs-vswitchd-dpdk
- sudo systemctl restart openvswitch-switch.service

## VI. OVS (Open vSwitch)

### ❖ Installing Docker and OVS-DPDK (예: Intel)

- ① `sudo apt install docker.io`
- ② `sudo apt install openvswitch-switch-dpdk`
- ③ `sudo update-alternatives --set OvS-vswitchd /usr/lib/openvswitch-switch`
- ④ `-dpdk/OvS-vswitchd-dpdk`
- ⑤ `sudo systemctl restart openvswitch-switch.service`



#### 메모:

- <https://software.intel.com/en-us/articles/using-docker-containers-with-open-vswitch-and-dpdk-on-ubuntu-1710>
- <https://01.org/kubernetes/building-blocks/userspace-cni>

## VI. OVS (Open vSwitch)

---

### ❖ Installing Docker and OVS-DPDK (예: OVS)

#### ① Install DPDK

- **Download the DPDK sources**  

```
$ cd /usr/src/  
$ wget http://fast.dpdk.org/rel/dpdk-18.11.2.tar.xz  
$ tar xf dpdk-18.11.2.tar.xz  
$ export DPDK_DIR=/usr/src/dpdk-stable-18.11.2  
$ cd $DPDK_DIR
```
- **Configure and install DPDK**  

```
$ export DPDK_TARGET=x86_64-native-linuxapp-gcc  
$ export DPDK_BUILD=$DPDK_DIR/$DPDK_TARGET  
$ make install T=$DPDK_TARGET DESTDIR=install
```

#### ② Install OVS

- **Ensure the standard OVS requirements, described in Build Requirements, are installed**
- **Bootstrap, if required, as described in Bootstrapping**
- **Configure the package using the --with-dpdk flag:**  

```
$. /configure --with-dpdk=$DPDK_BUILD
```
- **Build and install OVS, as described in Building**

#### ③ Setup

- **Setup Hugepages**
- **Setup DPDK devices using VFIO**
- **Setup OVS**

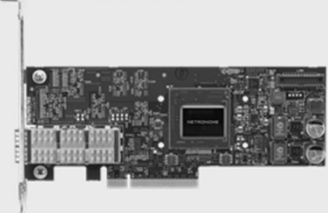
#### 메모:

- <http://docs.openvswitch.org/en/latest/intro/install/dpdk/>

# VI. OVS (Open vSwitch)

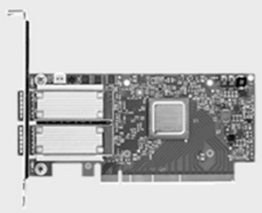
## ❖ SmartNIC and SR-IOV @ VMware

### SmartNIC



- Multi Core network processor chip
- Crypto IPSEC, SSL
- OVS Offloading
- Custom programming at data plane
- FPGA like flexibility

### Standard NIC



- General purpose
- 1G/10G/40G/100G/400G

### Common features

- CRC, Checksum offloading (TCP, etc)
- Tunneling, Overlay offloading (VXLAN, NVGRE, GENEVE)
- SR-IOV
- DPDK



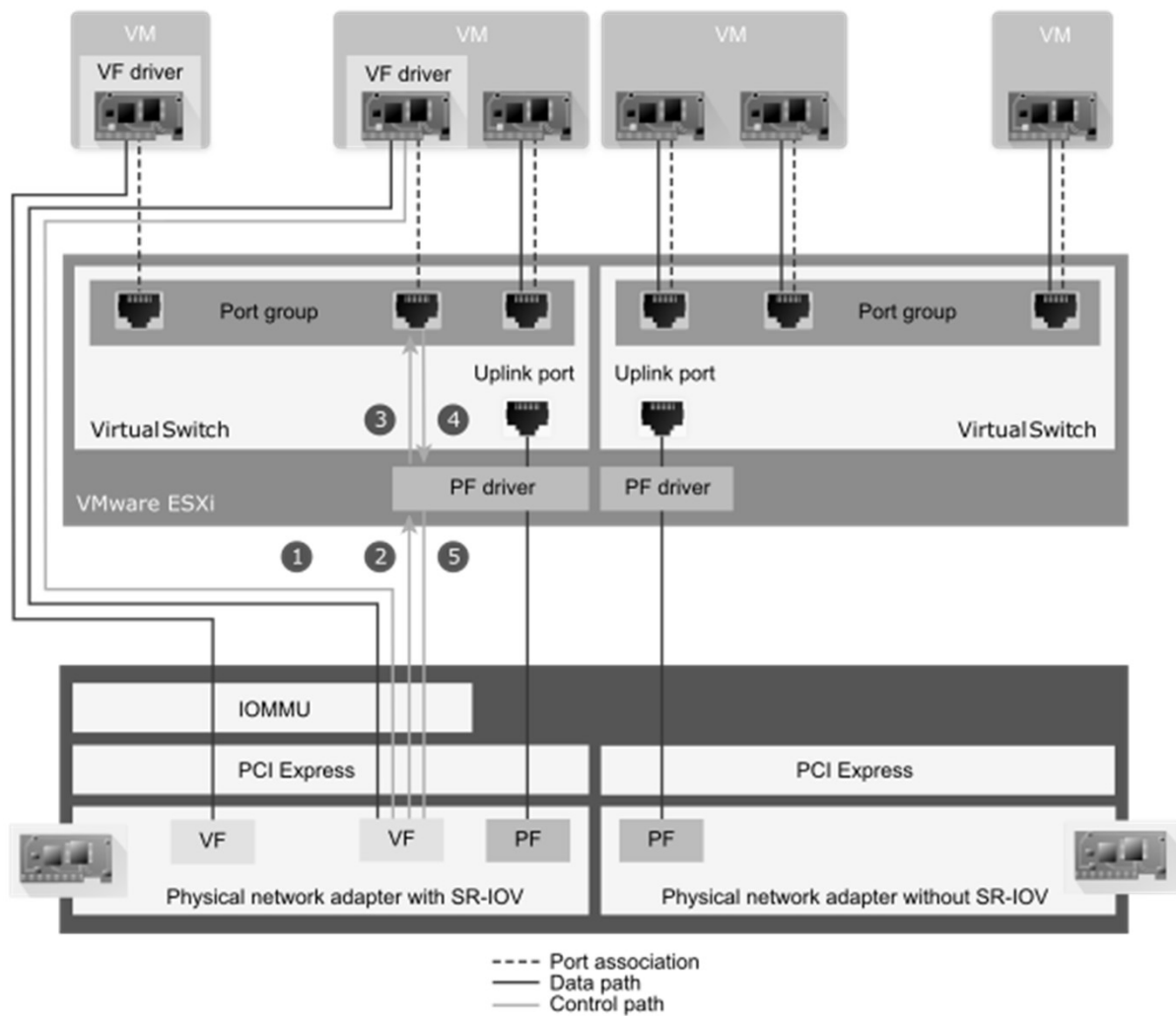
Network Adapter 2 configuration:

- Network: LAN1
- State:  전원을 켤 때 연결
- Adapter type: SR-IOV 패스스루 (dropdown menu open)
- Physical features: E1000, E1000e, AMD Lance PCNet 32, SR-IOV 패스스루 (selected), VMXNET 2(고급), VMXNET 3
- Buttons: 저장, 취소

메모:

# VI. OVS (Open vSwitch)

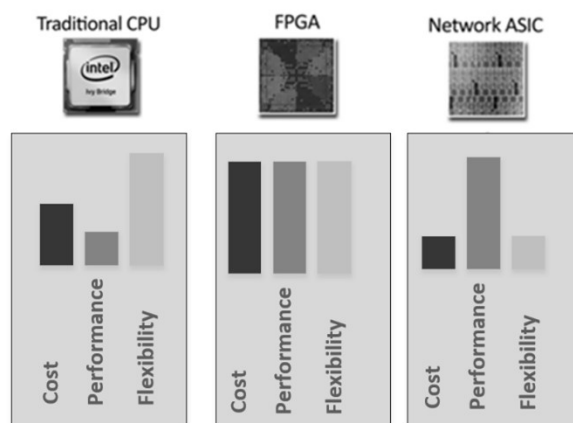
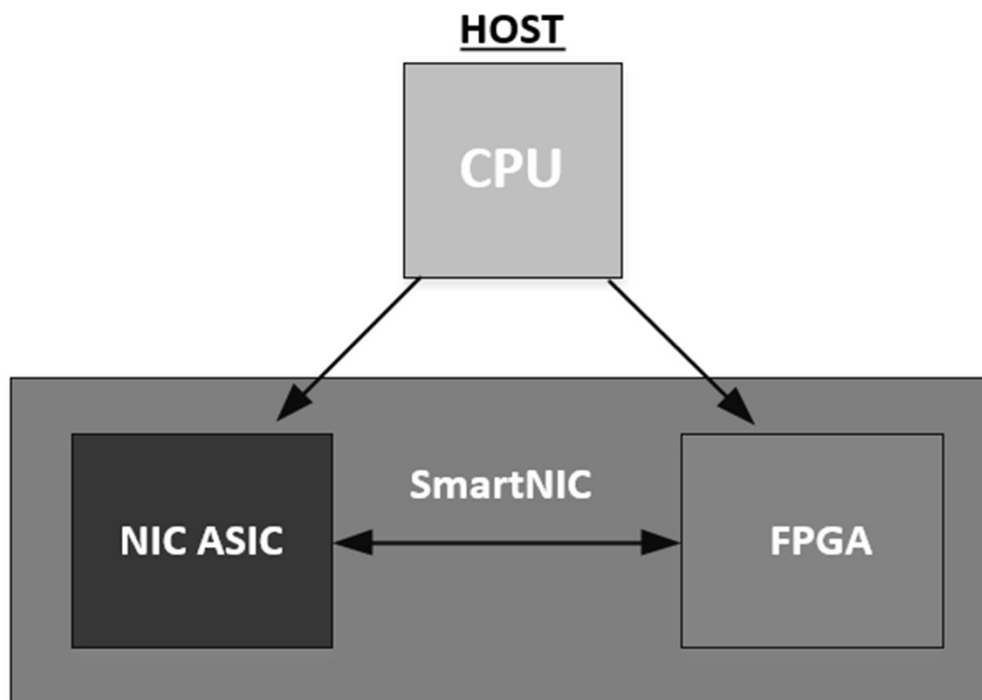
## ❖ SR-IOV Diagram for VMware ESXi



메모:

# VI. OVS (Open vSwitch)

## ❖ SmartNIC with FPGA



메모:

## VI. OVS (Open vSwitch)

### ❖ OVS(Open vSwitch) Installation (예: 스위치 2개)

- ① **sudo apt install -y openvswitch-switch**
- 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)
- 포트 추가: `sudo ovs-vsctl add-port ovs1 patch-ovs1`
- 포트 추가: `sudo ovs-vsctl add-port ovs2 patch-ovs2`
- `ps -ef | grep onos`

# VI. OVS (Open vSwitch)

## ❖ OVS(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#
```

- ovs-appctl
- ovsdb-client
- ovsdb-tool
- ovs-docker
- ovs-dpctl
- ovs-dpctl-top
- ovs-ofctl
- ovs-parse-backtrace
- ovs-pcap
- ovs-pki
- ovs-tcpdump
- ovs-tcpundump
- ovs-vlan-test
- ovs-vsctl

### 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



# VI. OVS (Open vSwitch)

## ❖ OVS with Docker Containers

- ① **sudo apt install docker.io** # just try 'docker' command
- ② **ifconfig**
- ③ **cd /usr/bin** # Install ovs-docker utility.
- ④ **sudo wget**  
<https://raw.githubusercontent.com/openvswitch/ovs/master/utilities/ovs-docker>
- ⑤ **ovs-vsctl add-br ovs1** # Create an OVS bridge.
- ⑥ **ifconfig ovs1 173.16.1.1 netmask 255.255.255.0 up**
- ⑦ **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  Mask:255.255.255.0
           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)

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)

enp0s3    Link encap:Ethernet  HWaddr 08:00:27:3e:9b:25
           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:1232 errors:0 dropped:0 overruns:0
           TX packets:998 errors:0 dropped:0 overruns:0
           collisions:0 txqueuelen:1000
           RX bytes:114040 (114.0 KB)  TX bytes:142092

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:797 errors:0 dropped:0 overruns:0
           TX packets:396 errors:0 dropped:0 overruns:0
           collisions:0 txqueuelen:1000
           RX bytes:744812 (744.8 KB)  TX bytes:29115

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:

- sudo docker version
- sudo docker info
- [http://containertutorials.com/network/ovs\\_docker.html](http://containertutorials.com/network/ovs_docker.html)

## VI. OVS (Open vSwitch)

### ❖ OVS with 'Docker Networking' 소개 (선택)

- ① 생성한 ovs1에 Ping 가능한 리눅스 OS 구동 컨테이너 접속

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

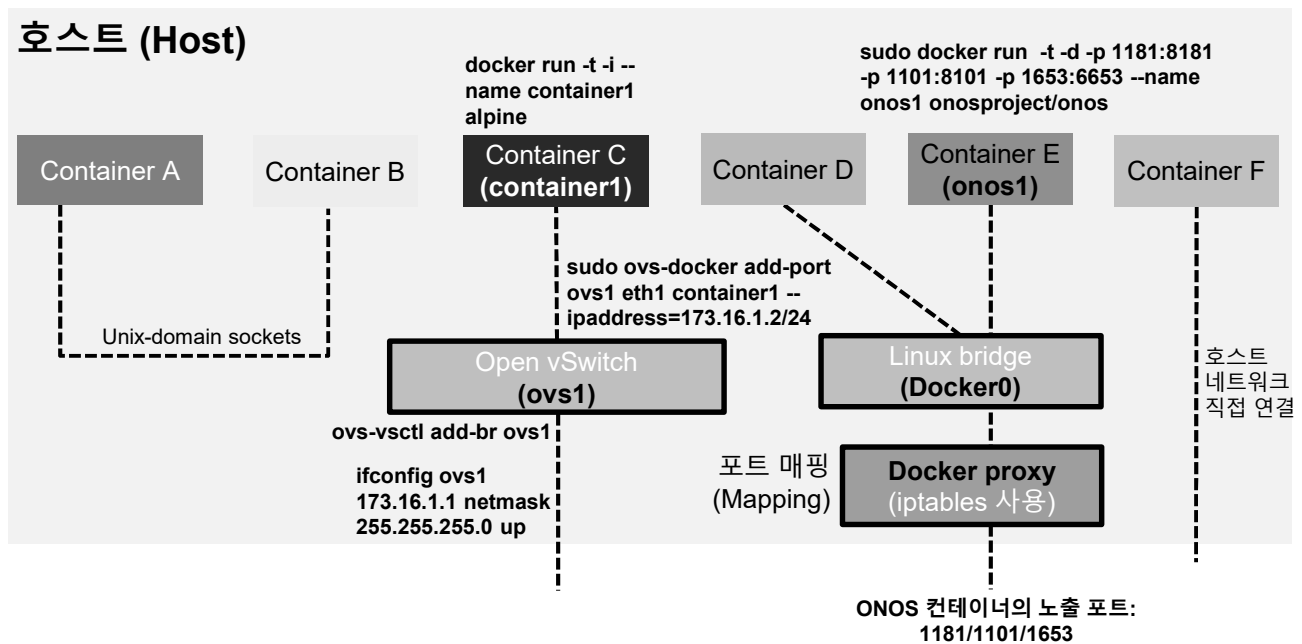
- ② ovs1 스위치에 호스트 외부 접속 인터페이스 생성

```
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:

## VI. OVS (Open vSwitch)

### ❖ Using OVS bridge with Docker Containers

- 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
16ddc135de29	alpine	"/bin/sh"	seconds ago	Up 6 seconds	7
564a21911e7f	alpine	"/bin/sh"	minutes ago	Up 5 minutes	5

```
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

## VI. OVS (Open vSwitch)

### ❖ Using OVS bridge with Docker Containers (선택)

- 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:

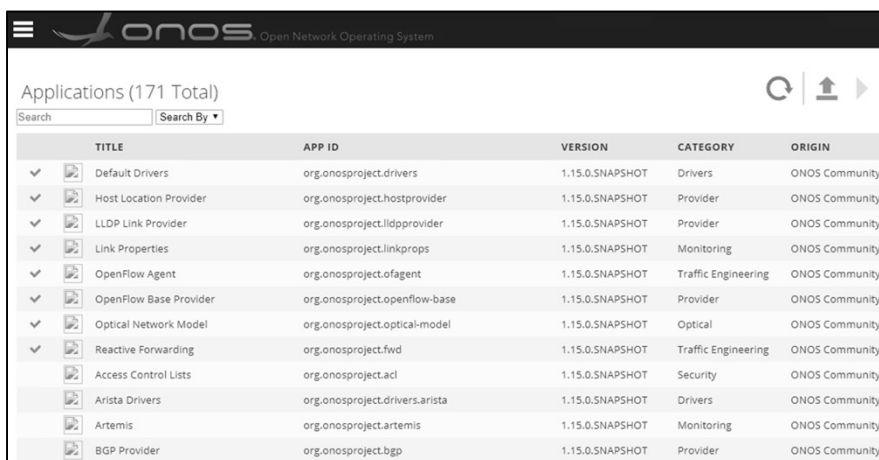
- 포트 삭제: `sudo ovs-docker del-port ovs2 eth1 container2 --ipaddress=173.16.1.3/24`
- It will be back after docker container
- 미사용 컨테이너 삭제: `sudo docker system prune`
- [http://containertutorials.com/network/ovs\\_docker.html](http://containertutorials.com/network/ovs_docker.html)

## VI. OVS (Open vSwitch)

### ❖ Using OVS bridge with SDN Controller 'ONOS' (선택)

- 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_1"
        Interface "78d47c80d9c54_1"
    ovs_version: "2.8.1"
```



The screenshot shows the ONOS web interface with a table of applications. The table has columns for Title, App ID, Version, Category, and Origin. The following table represents the data visible in the screenshot:

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**

## VI. OVS (Open vSwitch)

### ❖ 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:

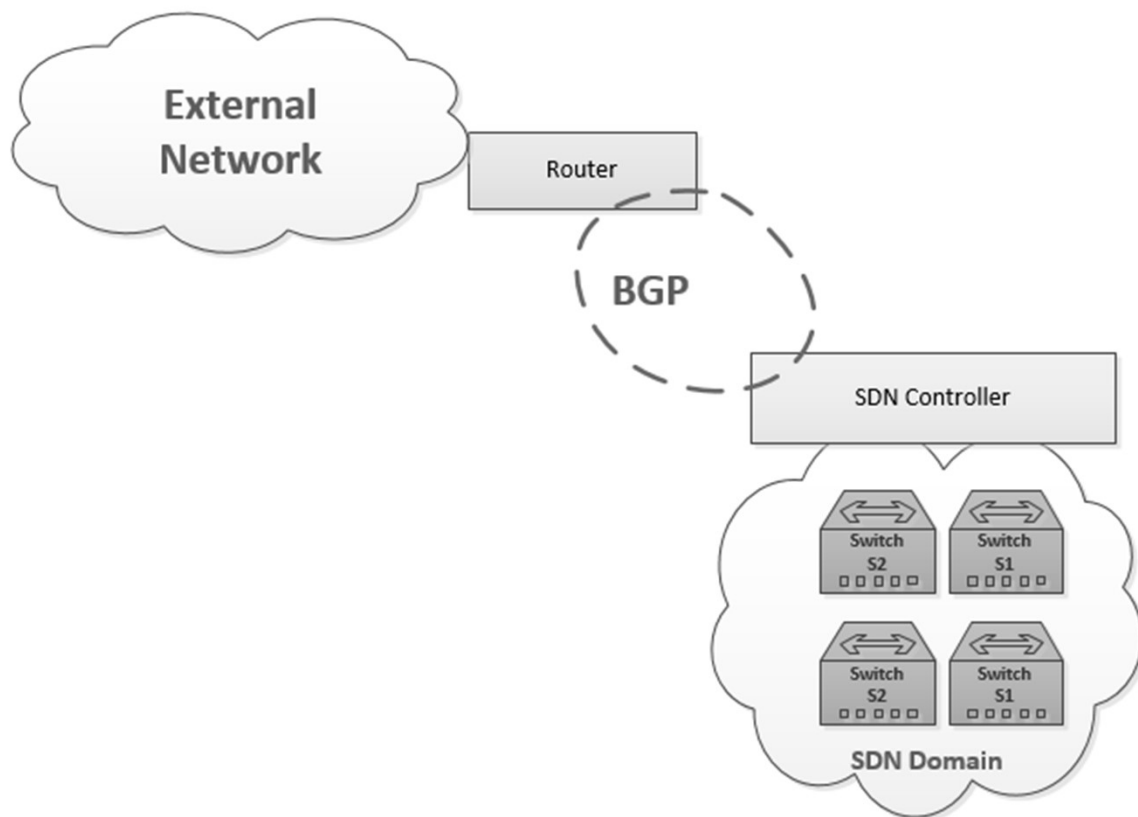
- Port 서로 연결 전에는 오류 발생 (정상)
- ONOS 에서 연결 확인

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

## VII. SDN 제어기 (ONOS)

### ❖ Relationship between a Network Managed by an SDN Controller and External Networks



#### 메모:

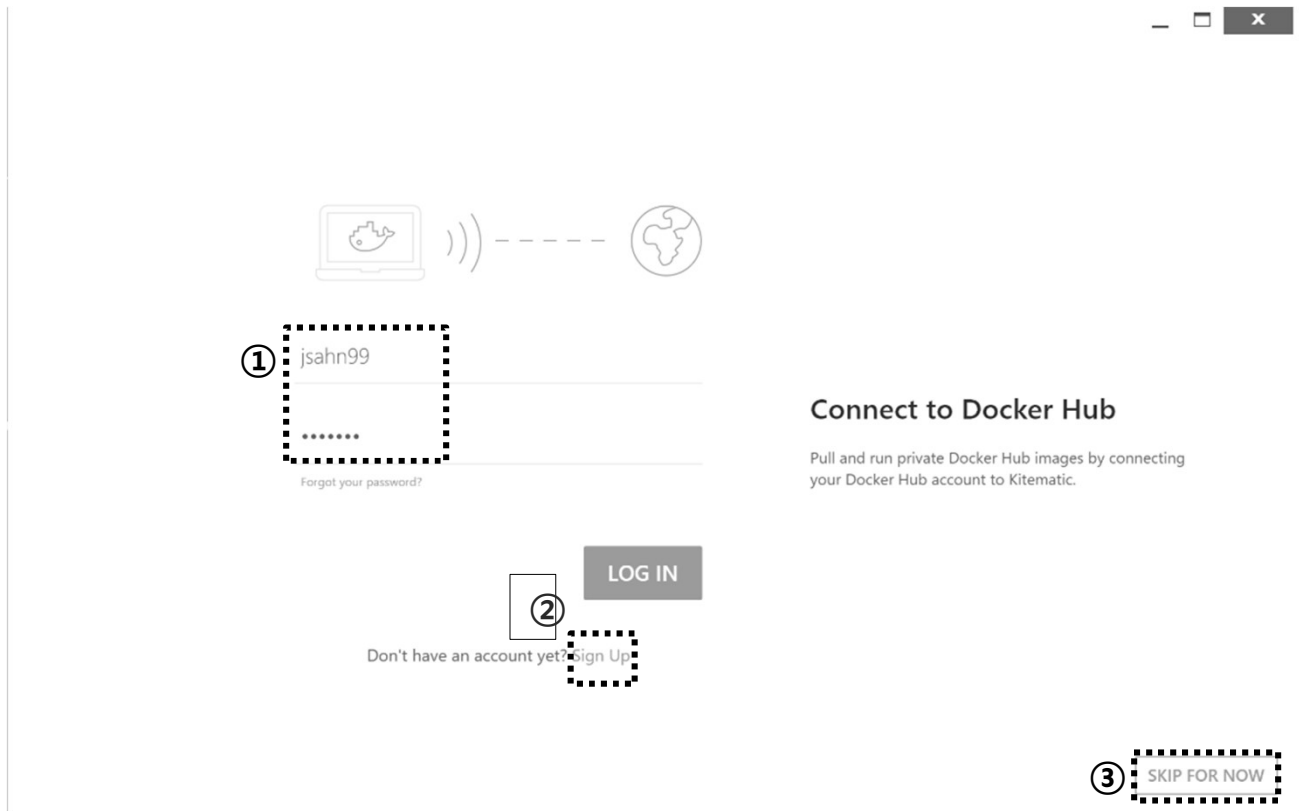
- BGP App @ ODL
- SDN-IP App @ ONOS



## VII. SDN 제어기 (ONOS)

### ❖ Docker Toolbox or Decsktop 사용 (Docker Hub 접속)

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



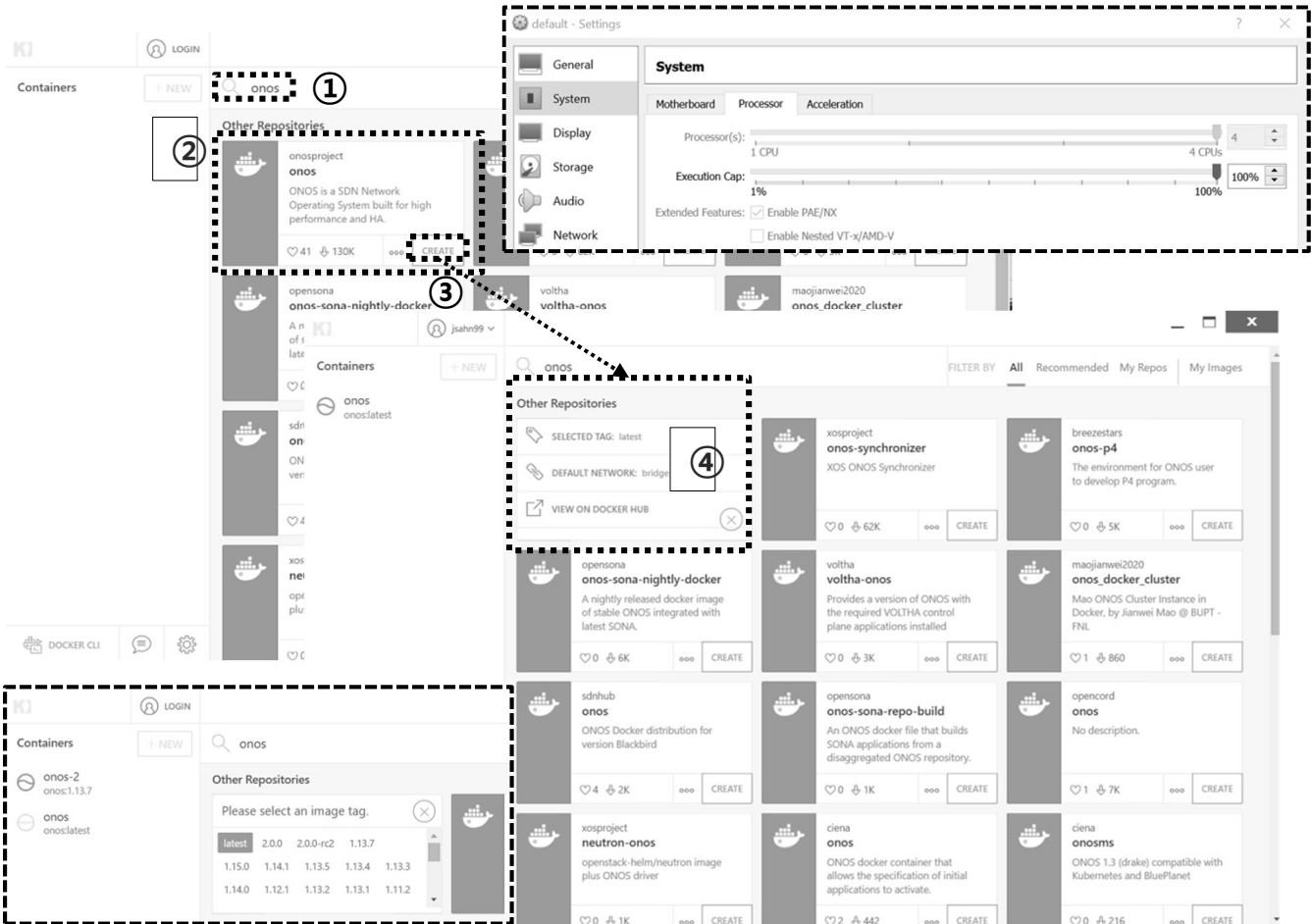
#### 메모:

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

# VII. SDN 제어기 (ONOS)

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

- ① Search 'onos'
- ② Check 'onosproject/onos'
- ③ Check 'ooo'
- ④ Check 'Default Network'



### 메모:

- 도커 ONOS 검색(Search) CLI: `sudo docker search onos`
- 도커 ODL 검색(Search) 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

## VII. SDN 제어기 (ONOS)

### ❖ 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

The screenshot shows the Docker Desktop interface for a container named 'onos'. The 'CONTAINER LOGS' panel displays the following information:

```
org.onosproject.mobility has been installed
07-21-12-485 INFO [onos-store-app-app-activation] Application
org.onosproject.configsync-netconf has been installed
07-21-12-497 INFO [onos-store-app-app-activation] Application
org.onosproject.odtn-service has been installed
07-21-12-508 INFO [onos-store-app-app-activation] Application
org.onosproject.hostprobingprovider has been installed
07-21-12-514 INFO [onos-store-app-app-activation] Application
org.onosproject.drivers.lumentum has been installed
07-21-12-527 INFO [onos-store-app-app-activation] Application
org.onosproject.drivers.barefoot has been installed
07-21-12-537 INFO [onos-store-app-app-activation] Application
org.onosproject.loadtest has been installed
07-21-12-578 INFO [onos-store-app-app-activation] Application
org.onosproject.newoptical has been installed
07-21-12-585 INFO [onos-store-app-app-activation] Application
org.onosproject.p4tutorial.mytunnel has been installed
07-21-12-605 INFO [onos-store-app-app-activation] Application
org.onosproject.pathpainter has been installed
07-21-12-617 INFO [onos-store-app-app-activation] Application
org.onosproject.tetunnel has been installed
07-21-12-627 INFO [onos-store-app-app-activation] Application
org.onosproject.netcfghostprovider has been installed
07-21-12-645 INFO [tcp1010590001-125] Registering topology session
[UITopoSession for user <onos>]
07-21-12-642 WARN [tcp1010590001-125] added overlay:
UITopo2Overlay[id="traffic-2-overlay", class="Traffic2Overlay"]
07-21-12-658 INFO [tcp1010590001-125] GUI client connected -- us
07-21-12-733 INFO [tcp1010590001-125] Session token authenticat
07-26-12-468 INFO [tcp1010590001-275] Session token revoked
07-26-12-473 INFO [tcp1010590001-275] Unregistering topology ses
[UITopoSession for user <onos>]
07-26-12-481 INFO [tcp1010590001-275] GUI client disconnected [c
code=1001, message=null]
```

The 'IP & PORTS' panel shows the following configuration:

DOCKER PORT	ACCESS URL
6640/tcp	192.168.99.103:32772
6653/tcp	192.168.99.103:32771
8101/tcp	192.168.99.103:32770
8181/tcp	192.168.99.103:32769
9876/tcp	192.168.99.103:32768

The browser window shows the ONOS web interface at <http://192.168.99.103:32769/onos/ui/index.html>. The ONOS Summary panel shows:

```
ONOS Summary
Version: 2.1.0.a80d3f9553
Devices: 0
Links: 0
Hosts: 0
Topology SCCs: 0
Intents: 0
Flows: 0
```

The status bar at the bottom indicates "No Devices Are Connected".

#### 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`

## VII. SDN 제어기 (ONOS)

### ❖ Linux Host 의 ONOS 컨테이너 실행 @ Ubuntu or CentOS

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`

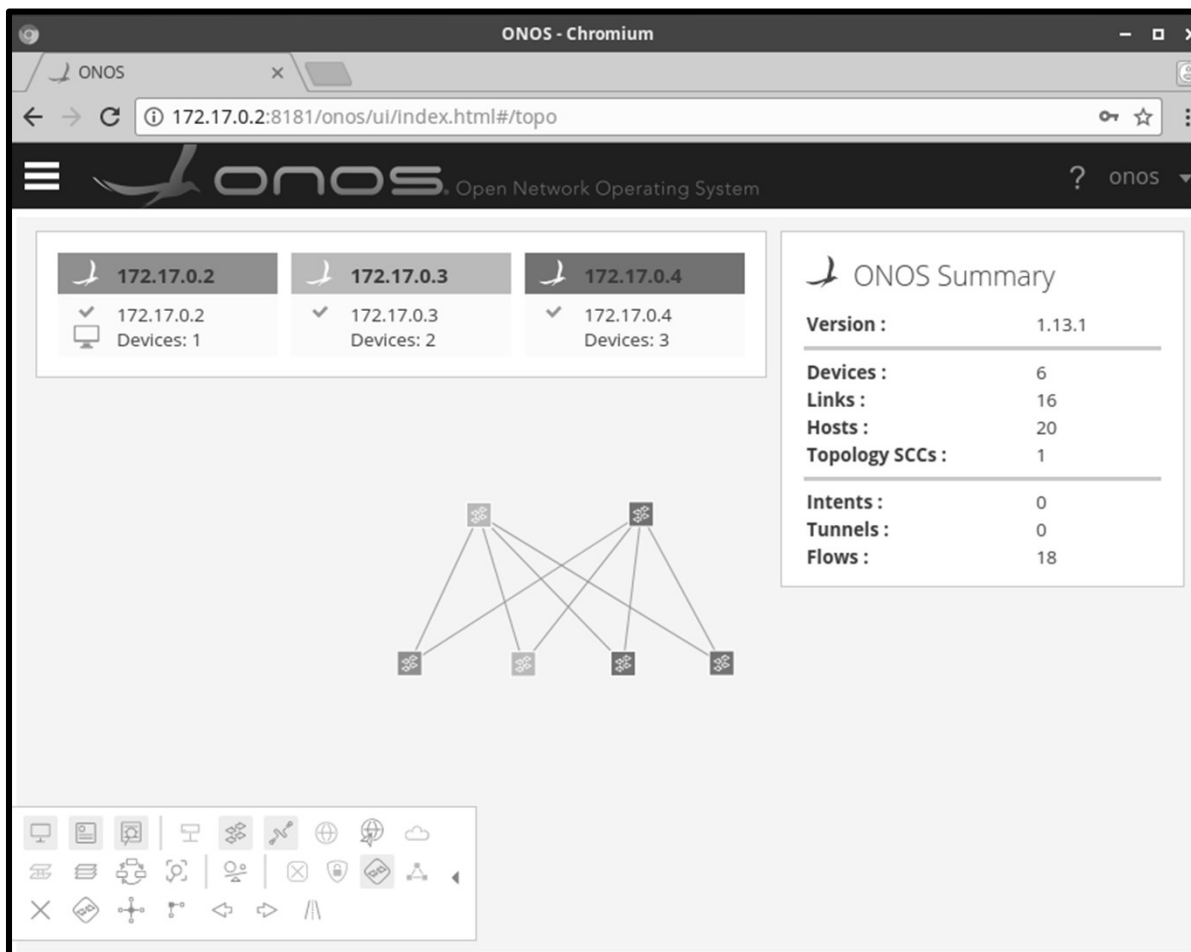
#### 메모:

- 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)

## VII. SDN 제어기 (ONOS)

### ❖ Launch ONOS (예: ONOS SDN Container Cluster)

- ① **User ID / Password: onos / rocks**
- ② **Ifconfig**
- ③ **docker ps**
- ④ **/ key, L key, H Key, E key**



#### 메모:

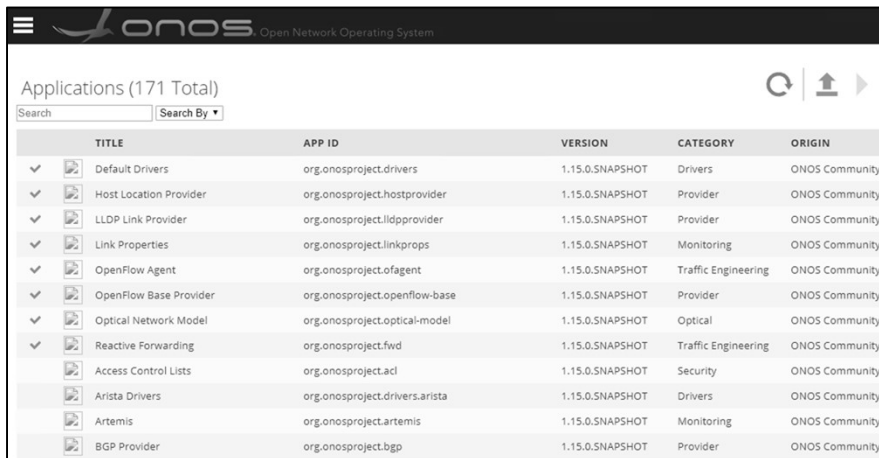
- **Mastership re-balancing:** The network devices will be roughly equally divided between all nodes in the ONOS cluster. To do this from the GUI, press the E key.

## VII. SDN 제어기 (ONOS)

### ❖ ONOS ← → OVS 연결 확인

- ① **sudo ovs-vsctl set-controller ovs1 tcp:192.168.0.102:6653**
- 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_1"
      Interface "78d47c80d9c54_1"
    ovs_version: "2.8.1"
```



The screenshot shows the ONOS web interface with a table of applications. The table has columns for Title, App ID, Version, Category, and Origin. The following table represents the data shown in the screenshot:

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

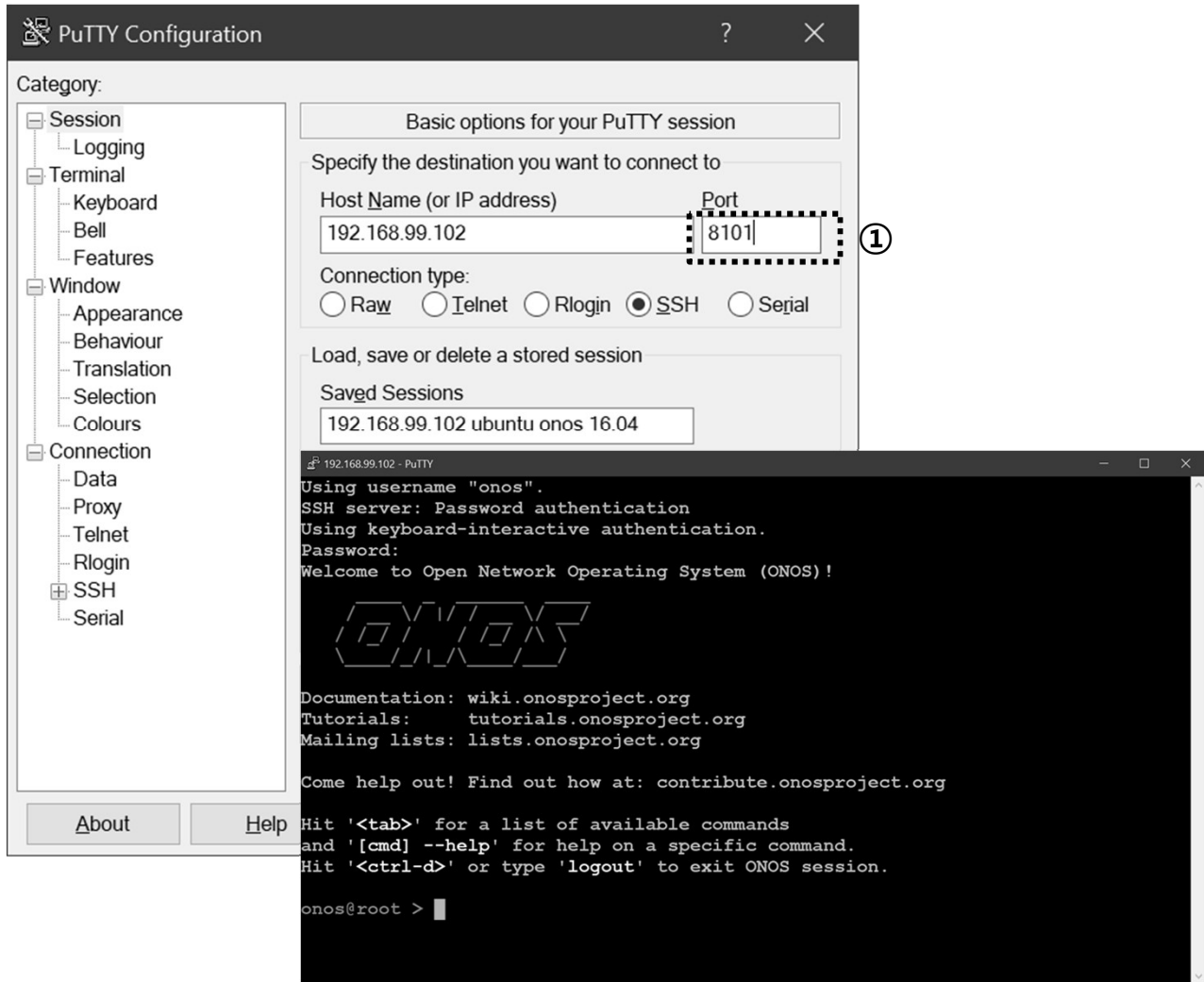
#### 메모:

- **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**

# VII. SDN 제어기 (ONOS)

## ❖ ONOS ← → OVS 연결 확인


- ① **ssh james@192.168.99.xxx:8101** # 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

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨



# VIII. 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  8000.0242d7ebbba       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:a5:68: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

## VIII. 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

## VIII. Container Networking (Docker)

### ❖ 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": {
      "7d3800792767f454cdf79d48500a62f5ceb993ac1146df03f8a4f66c7a8f5d8": {
        "Name": "determined_wiles",
        "EndpointID": "00c397c6642f38fcf3cddf205c9a7a0c5ac3d34e894fa79672bfc5c6e228e99",
        "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

## VIII. Container Networking (Docker)

### ❖ 'docker network inspect ingress' (도커 설치 후 확인)

```
james@masteratlocal:~$ sudo docker network inspect ingress
[
  {
    "Name": "ingress",
    "Id": "11yxmoq9eeyt066f00dv3jkyf",
    "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:

## VIII. Container Networking (Docker)

---

### ❖ Ping (Self-study Sample)

- 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

## VIII. Container Networking (Docker)

---

### ❖ Ping (Self-study Sample)

#### ⑥ apt-get update && apt-get install -y iptutils-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 iptutils-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 iptutils-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!

## VIII. Container Networking (Docker)

---

### ❖ Ping (Self-study Sample)

- 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

## VIII. Container Networking (Docker)

### ❖ 외부 연결을 위한 NAT 구성 (Self-study Sample)

- 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
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.

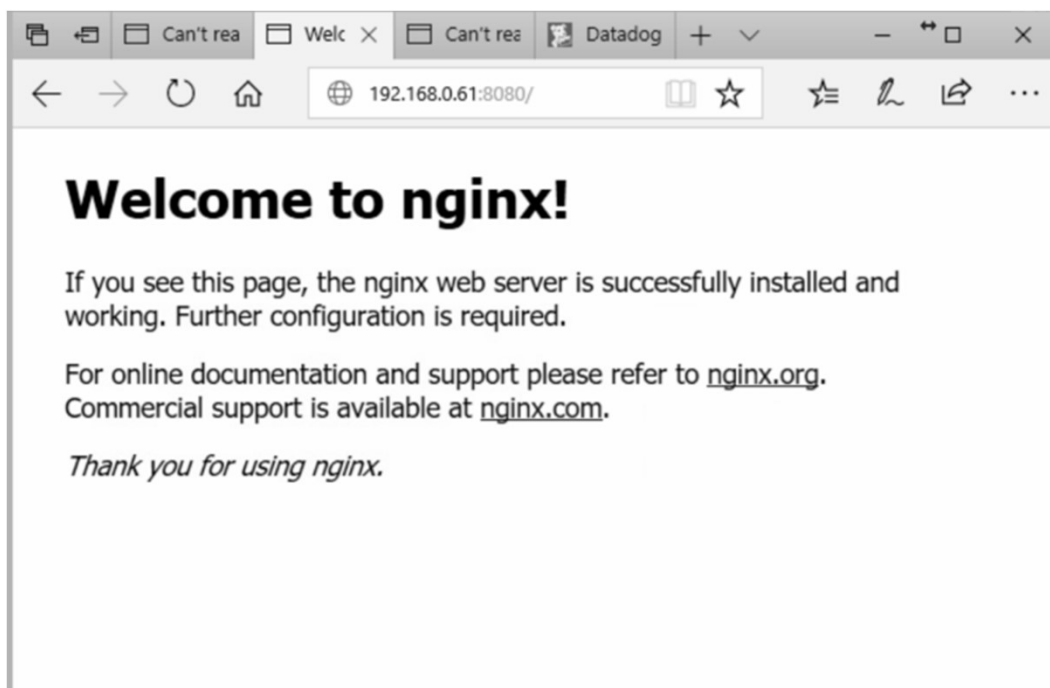


## VIII. Container Networking (Docker)

---

### ❖ 외부 연결을 위한 NAT 구성 (Self-study Sample)

④ <http://192.168.0.61:8080>



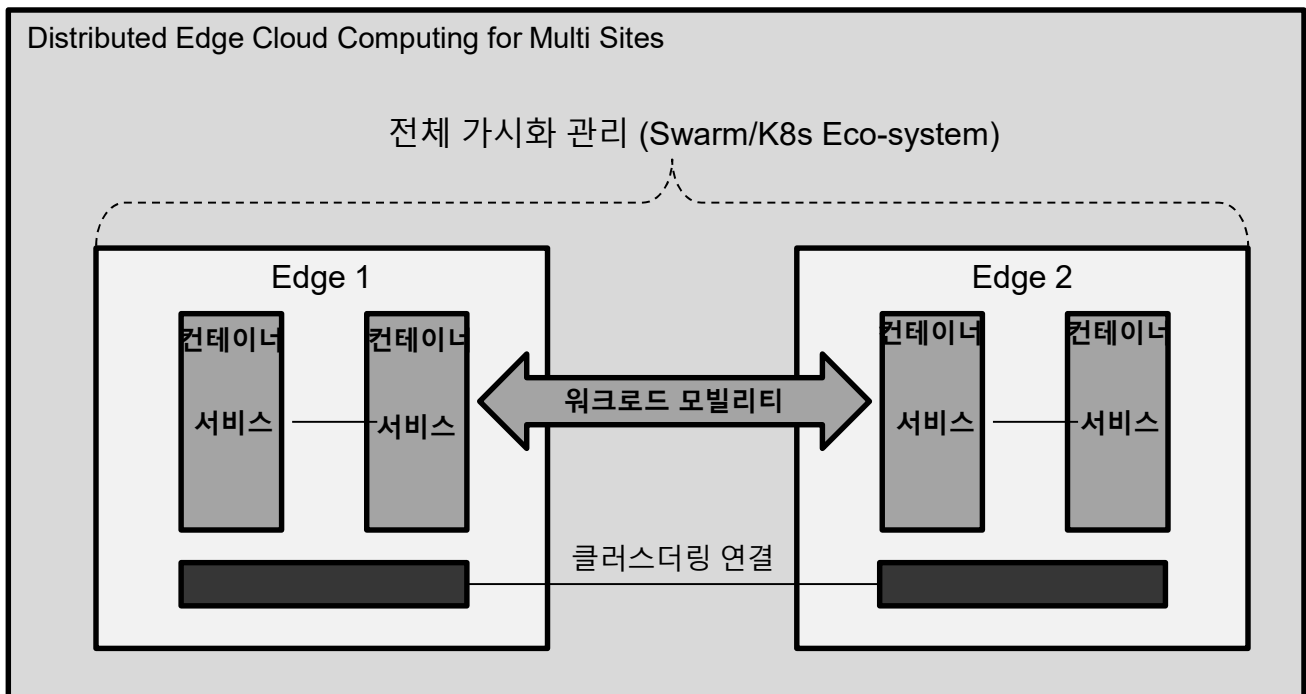
#### 메모:

- 외부 연결을 위한 NAT 구성

## VIII. Container Networking (Docker)

### ❖ Container Based Edge Cloud Computing Infra

- Swarm / K8s Orchestration (High Level over API)
- CNCF Eco-system
- Container based infrastructure



#### 메모:

- `docker swarm join \`
    - > `--token SWMTKN-1-69b2x1u2wtjdmot0oqxjw1r2d27f0lbnhfxhvj83chln1l6es5-37ykdpul0vylenefef2439cqp \`
    - > `10.0.0.5:2377`
- This node joined a swarm as a worker.

# VIII. Container Networking (Docker)

## ❖ Swarm Mode

- ① `docker swarm init --advertise-addr 192.168.0.yy`
- ② `docker node ls`

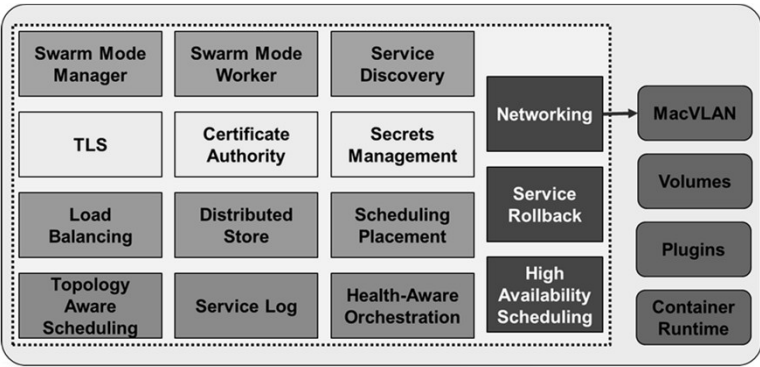
```
docker swarm init --advertise-addr 192.168.1.233
Swarm initialized: current node (j2l913nyay7n5vvn843djp91r) is now a manager.

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

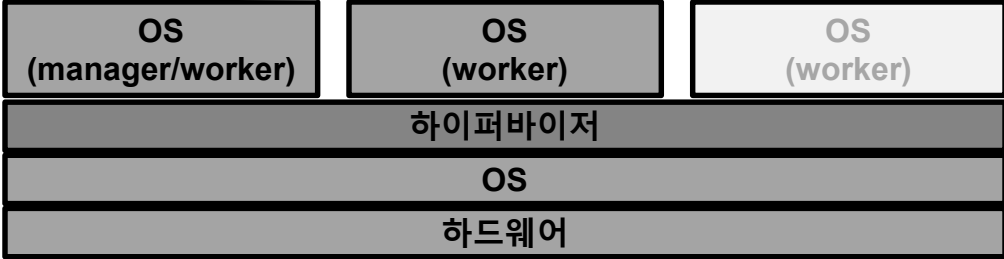
    docker swarm join --token SWMTKN-1-4w4rb969oyf96q2zfw65lfmg5xgksaczgkexxbzxc7l8lhw09-3l1uwi7aje3r7ljvtx0rwt237
    192.168.1.233:2377

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

root@ubuntutemplate:/home/jslab#
```



## Swarm vs. K8s



**메모:**

- `docker swarm join \`
  - > `--token SWMTKN-1-69b2x1u2wtjdmot0oqxjw1r2d27f0lbnhfxhvj83chln1l6es5-37ykdpul0vylenefe2439cqp \`
  - > `10.0.0.5:2377`

This node joined a swarm as a worker.

## VIII. Container Networking (Docker)

### ❖ Swarm Mode

- ① `docker swarm init --advertise-addr $(hostname -i)`
- ② `docker swarm join --token SWMTKN-1-133f2nioom30v47dr4c8j8q4uq5hhp3gn7su5tazj1a2oczomg-84iw7bynjt7f0qhy98u2mcou9 127.0.1.1:2377`
- ③ `docker node ls` # @ Manager node

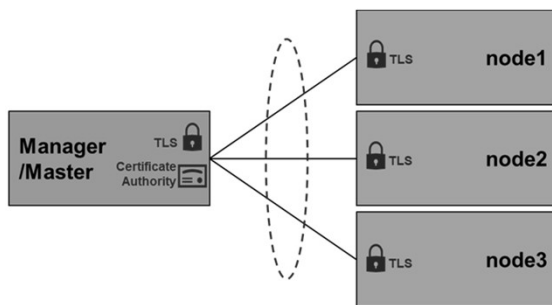
```
james@ubuntu17template:~$ docker swarm init --advertise-addr $(hostname -i)
Swarm initialized: current node (9r7jspmooi98x7ubb1c282jta) is now a manager.
```

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

```
docker swarm join --token SWMTKN-1-133f2nioom30v47dr4c8j8q4uq5hhp3gn7su5tazj1a2oczomg-84iw7bynjt7f0qhy98u2mcou9 127.0.1.1:2377
```

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

```
james@ubuntu4k8s-1:~$ sudo docker swarm init --advertise-addr 192.168.0.30
.....
james@ubuntu4k8s-1:~$ sudo docker node ls
[sudo] password for james:
ID                HOSTNAME          STATUS  AVAILABILITY  MANAGER STATUS
1ckstnt9bgujuu2kdfn4yzpsa  kubeworker1     Ready   Active
wpj943eytbj91e3s8c1bugizz *  ubuntu4k8s-1   Ready   Active         Leader
james@ubuntu4k8s-1:~$
```



#### 메모:

- Docker는 Kubernetes 지원 기능을 출시
- 호스트에 복수 interface 시 `sudo docker swarm init --advertise-addr 192.168.0.xx` 지정

## VIII. Container Networking (Docker)

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

- ① `docker swarm join-token manager`
- ② `docker swarm join-token worker`
- ③ `docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwwvph6lmhp3zb3e2b-7rukukuz7kmgnt0s1klrq5o2 192.168.0.60:2377 # @`  
Manager
- ④ `docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwwvph6lmhp3zb3e2b-7rukukuz7kmgnt0s1klrq5o2 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-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwwvph6lmhp3zb3e2b-
    2a7m4ydl5j3hqx7jdwyasg 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-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwwvph6lmhp3zb3e2b-
    7rukukuz7kmgnt0s1klrq5o2 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`

# VIII. Container Networking (Docker)

## ❖ Clone the Voting App

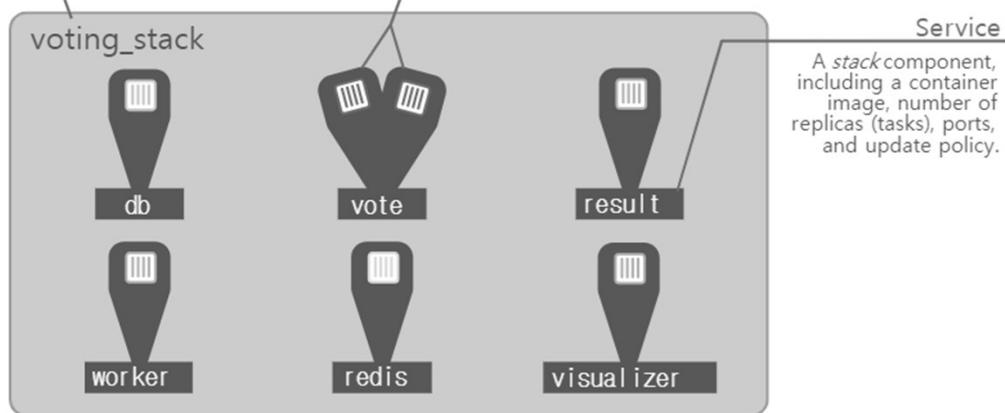
- ① `git clone https://github.com/docker/example-voting-app`
- ② `cd example-voting-app`
- ③ `cat docker-stack.yml`
- ④ `docker stack deploy --compose-file=docker-stack.yml voting_stack`
- ⑤ `docker stack ls`
- ⑥ `docker stack services voting_stack`
- ⑦ `docker service ps voting_stack_vote`

### Stack

Group of interrelated *services* & dependencies. Orchestrated as a unit. Production applications are one stack, and sometime more.

### Tasks

Atomic unit of a *service* and scheduling in Docker. One container instance per task.



### Service

A *stack* component, including a container image, number of replicas (tasks), ports, and update policy.

메모:

# VIII. Container Networking (Docker)

## ❖ cat docker-stack.yml

### ① cat docker-stack.yml

```
root@ubuntutemplate:/home/jslab/example-voting-app# cat docker-stack.yml
version: "3"
services:

  redis:
    image: redis:alpine
    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]

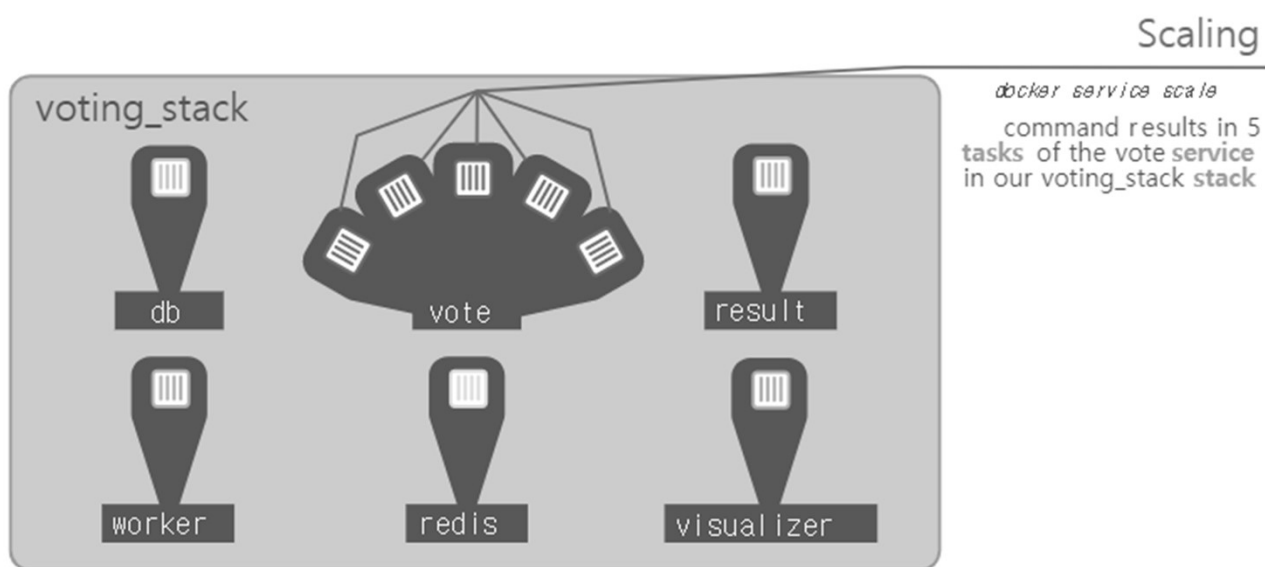
networks:
  frontend:
  backend:

volumes:
  db-data:
root@ubuntutemplate:/home/jslab/example-voting-app#
```

# VIII. Container Networking (Docker)

## ❖ Scaling An Application

- ① `docker service scale voting_stack_vote=5`
- ② `docker stack`
- ③ `docker stack ls`
- ④ `docker stack rm voting_stack`



### 메모:

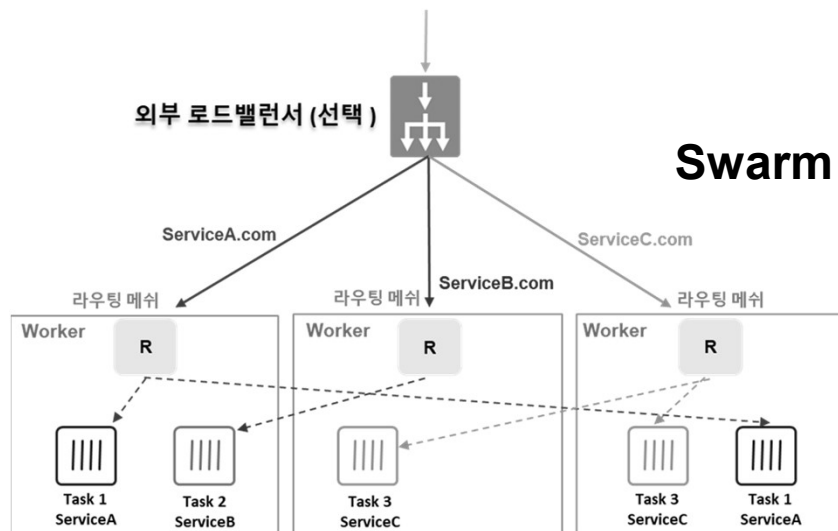
- `docker image history <image ID>`



## VIII. Container Networking (Docker)

### ❖ 서비스 접속 (Routing Mesh for 1 Manager and 3 Worker)

- 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

## VIII. Container Networking (Docker)

---

### ❖ 스웜 종료 (선택)

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

#### 메모:

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

## VIII. Container Networking (Docker)

### ❖ 오버레이(Overlay) 연결을 위한 구성 (Self-study Sample)

- 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 (19e8wqyjw00ogjl092n0eyymr) 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
19e8wqyjw00ogjl092n0eyymr * kubemaster      Ready        Active           Leader           18.03.0-ce
kb55f7sda5mduiml0a2o5a9vx kubeworker1     Ready        Active           18.03.0-ce
[root@kubemaster ~]#
```

#### 메모:

- Overlay Networking

## VIII. Container Networking (Docker)

---

### ❖ 오버레이(Overlay) 연결을 위한 구성 (Self-study Sample)

- 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

## VIII. Container Networking (Docker)

### ❖ 오버레이(Overlay) 연결을 위한 구성 (Self-study Sample)

- ⑥ `docker network create -d overlay overnet`
- ⑦ `docker network inspect overnet`

```
[root@kubemaster ~]# docker network inspect overnet
[
  {
    "Name": "overnet",
    "Id": "2n20w14b1ggir4ie2dok2tagz",
    "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
  }
]
```

#### 메모:

- Overlay Networking

## VIII. Container Networking (Docker)

### ❖ 오버레이(Overlay) 연결을 위한 구성 (Self-study Sample)

- 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
3nzzhjsoglebi jq01y8w0mfu
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
3nzzhjsogle      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

## VIII. 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

# VIII. Container Networking (Docker)

## ❖ 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::60d7:5bfff: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:fe: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>



## VIII. Container Networking (Docker)

---

### ❖ 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$
```

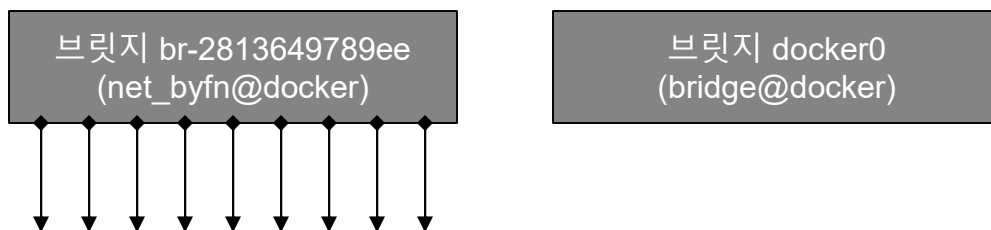
메모:

## VIII. Container Networking (Docker)

### ❖ sudo docker network ls & brctl show (Self-study Sample)

- 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                   veth02bb183
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:

## VIII. Container Networking (Docker)

### ❖ brctl showmacs br-2813649789ee (Self-study Sample)

#### 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:

## VIII. Container Networking (Docker)

---

❖ **sudo virsh net-list --all** (Self-study Sample)

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

## VIII. Container Networking (Docker)

### ❖ sudo docker network inspect bridge (Self-study Sample)

#### 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:

# VIII. Container Networking (Docker)

## ❖ sudo docker image inspect onosproject/onos

```
sdn@sdn:~$ sudo docker image inspect onosproject/onos
[
  {
    "Id": "sha256:ec5d8befbdf5846d2713c9865e9e9ec812ecc5dec269a643bbe89d4bb099e2",
    "RepoTags": [
      "onosproject/onos:latest"
    ],
    "RepoDigests": [
      "onosproject/onos@sha256:c9153ec30673500315aa8685e6bf5f72ca0c5f24c7857e7c46364382b4f"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2018-10-11T15:05:57.706099077Z",
    "Container": "fbdcef50a024a7b2cfb0d2a4b850a00fb1337a607914db4106b8613789b750af",
    "ContainerConfig": {
      "Hostname": "fbdcef50a024",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "ExposedPorts": {
        "9940/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-schema.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-schema.vendor": "Open Networking Foundation"
      },
      "Architecture": "amd64",
      "Os": "linux",
      "Size": 476856426,
      "VirtualSize": 476856426,
      "GraphDriver": {
        "Data": {
          "LowerDir":
            "/var/lib/docker/overlay2/6e162c562b6974d4c4880477b570d5b5e066052d56e64eeaf30a45d34527f42/diff:/var/lib/docker/overlay2/d5da7cb928321e61a9810097e2eaa9ee8d5099f170908a04f3f82574db685a4b/diff:/var/lib/docker/overlay2/67b2b2dfc8ceb60636fed430674aa632d289af15ba7e03f9879632141ff2979f/diff:/var/lib/docker/overlay2/798b3a994c1c063515f8137d727a15950ac6d8d76baaad2e263f79c124b0b31d/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:ebf12965380b39889c9a9c02e82ba465f887b45975b6e389d42e9e6a3857888",
          "sha256:f81d498bb79e7f680c98171ecb36651ff959610646aa1092e195a58c25fc7918",
          "sha256:478189a15cc48c5c8ff88a91b01034c5ad9afd78b5cdd4b01687afcb378faac4",
          "sha256:6de6049b450a58e07ccb96b23670a1343e2c5e4437fc6426e165a0bd7065ed81",
          "sha256:5c681a837481e881c103e2d4d1b30822dfb00da49e027a6dceb3776f4f68ab2"
        ]
      },
      "Metadata": {
        "LastTagTime": "0001-01-01T00:00:00Z"
      }
    }
  ]
}
```

james@jslab.kr

메모:

# VIII. Container Networking (Docker)

## ❖ Hyperledger (예): 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/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/server.key
    - CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/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$
```

james@jslab.kr

메모:

## VIII. Container Networking (Docker)

---

### ❖ Test network connectivity (Self-study Sample)

- ① **docker network**
- ② **docker network ls**
- ③ **docker network inspect bridge**
- ④ **docker info**
- ⑤ **sudo apt-get install bridge-utils**
- ⑥ **apk update**
- ⑦ **apk add bridge**
- ⑧ **brctl show**
- ⑨ **ip a**
- ⑩ **docker run -dt ubuntu sleep infinity**
- ⑪ **docker ps**
- ⑫ **docker network inspect bridge**
- ⑬ **ping -c5 172.17.0.2**
- ⑭ **docker ps**
- ⑮ **docker exec -it yourcontainerid /bin/bash**
- ⑯ **/ # ping -c5 www.github.com**
- ⑰ **/ # apt-get update && apt-get install -y iputils-ping**
- ⑱ **/ # ping -c5 www.github.com**
- ⑲ **/ # exit**

#### 메모:

- `docker image history <image ID>`



## VIII. Container Networking (Docker)

---

- ❖ **Configure NAT for external connectivity**
- ❖ **Create an overlay network (Self-study Sample)**

- ① **docker run --name web1 -d -p 8080:80 nginx**
- ② **docker ps**
- ③ **curl 127.0.0.1:8080**
- ④ **docker swarm init --advertise-addr \$(hostname -i)**
- ⑤ **docker node ls**
- ⑥ **docker network create -d overlay overnet**
- ⑦ **docker network ls**
- ⑧ **docker network inspect overnet**

### 메모:

- docker swarm join \  
> --token SWMTKN-1-69b2x1u2wtjdmot0oqxjw1r2d27f0lbnhfxhvj83chln1l6es5-37ykdpul0vylenefe2439cqp \ \  
> 10.0.0.5:2377

This node joined a swarm as a worker.

## VIII. Container Networking (Docker)

---


### ❖ Network for service (Self-study Sample)

- ① **docker service create --name myservice \  
--network overnet \  
--replicas 2 \  
ubuntu sleep infinity**
- ② **docker service ls**
- ③ **docker service ps myservice**
- ④ **docker network ls**
- ⑤ **docker network inspect overnet**
- ⑥ **docker ps**
- ⑦ **docker exec -it yourcontainerid /bin/bash**
- ⑧ **/ # ping -c5 10.0.0.3**
- ⑨ **/ # apt-get update && apt-get install -y iputils-ping**
- ⑩ **/ # ping -c5 10.0.0.3**
- ⑪ **/ # ping -c5 myservice**
- ⑫ **/ # exit**
- ⑬ **docker service rm myservice**
- ⑭ **docker swarm leave --force # on node 1**
- ⑮ **docker swarm leave --force # on node 2**

#### 메모:

- `docker image history <image ID>`

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

## IX. Cloud Networking (Rancher/K8s/Istio)

### ❖ Rancher 2.x installation (예: CentOS 7)

- ① `docker run -d --restart=unless-stopped -p 9999:80 -p 9443:443 rancher/rancher`
- ② `http://192.168.0.xx:9999`

```
[root@master ~]# docker run -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
docker run -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
Unable to find image 'rancher/rancher:latest' locally
Trying to pull repository docker.io/rancher/rancher ...
latest: Pulling from docker.io/rancher/rancher
38e2e6cd5626: Pull complete
705054bc3f5b: Pull complete
c7051e069564: Pull complete
7308e914506c: Pull complete
0cfcb3cfb94b: Pull complete
49cb3551f487: Pull complete
8856f5defe68: Pull complete
d50abe29b623: Pull complete
297145c80f79: Pull complete
b6b66b1777e8: Pull complete
7edf8d37c2b7: Pull complete
511c877e7916: Pull complete
Digest: sha256:924b8acaa169821c86b840c33e1d79d87db0dfbb84dae6c102cc7c196811230f
Status: Downloaded newer image for docker.io/rancher/rancher:latest
721da9cd2b74a152481cf22e0a47191afe832a17fa333b5a3baca483266f5a5f
You have new mail in /var/spool/mail/root
[root@master ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
721da9cd2b74      rancher/rancher   "entrypoint.sh doc... 3 minutes ago      Up 3
minutes           0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp   elated_kowalevski
[root@master ~]#
```

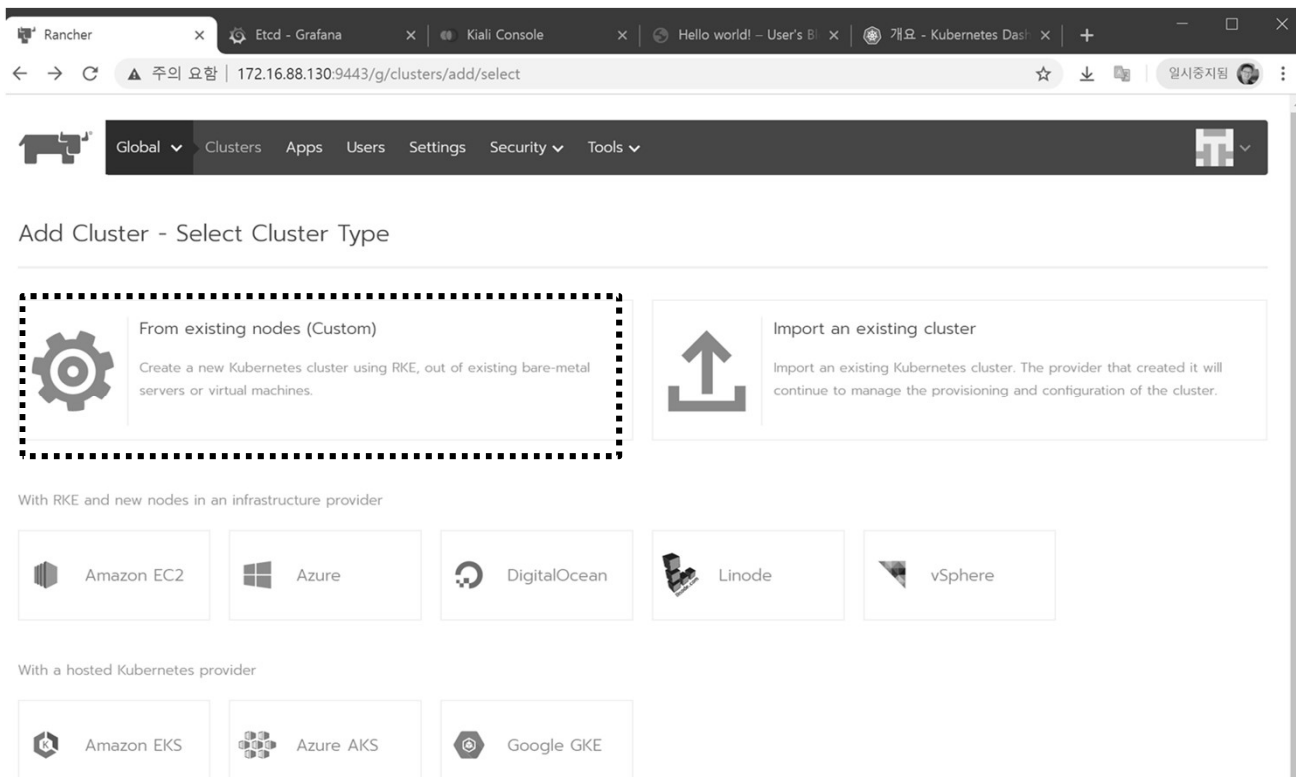
#### 메모:

- `systemctl disable firewalld`
- `systemctl stop firewalld`
- `systemctl status firewalld`

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher Clusters

- ① From the Clusters page, click Add Cluster.
- ② Select “From existing nodes (Custom)”



### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher Clusters

### ① Cluster Name

### ② Next

The screenshot shows the Rancher web interface for adding a custom cluster. The 'Cluster Name' field is highlighted with a dashed box and contains the text 'Kubernetes'. The page shows various configuration options like 'Kubernetes Version', 'Network Provider', and 'Cloud Provider'.

### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher Clusters

- ① Copy command
- ② Paste command @ workers

Global Clusters Apps Users Settings Security Tools

Add Cluster - Custom

Cluster Options Edit as YAML

Customize Node Run Command  
Editing node options will update the command you will run on your existing machines

1 Node Options  
Choose what roles the node will have in the cluster

Node Role

etcd  Control Plane  Worker Show advanced options

2 Run this command on one or more existing machines already running a supported version of Docker.

```
sudo docker run -d --privileged --restart=unless-stopped --net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run rancher/rancher-agent:v2.3.0 --server https://172.16.88.130:9443 --token krqmd8b7wthknr97vrtwz5c6mqjcl7qgmpsrq6pjk2v7m9gwnvg8 --ca-checksum d667145049471010aceaf5506d5a827d133ebf5baa3ead50510f32c194cae155 --worker
```

Done

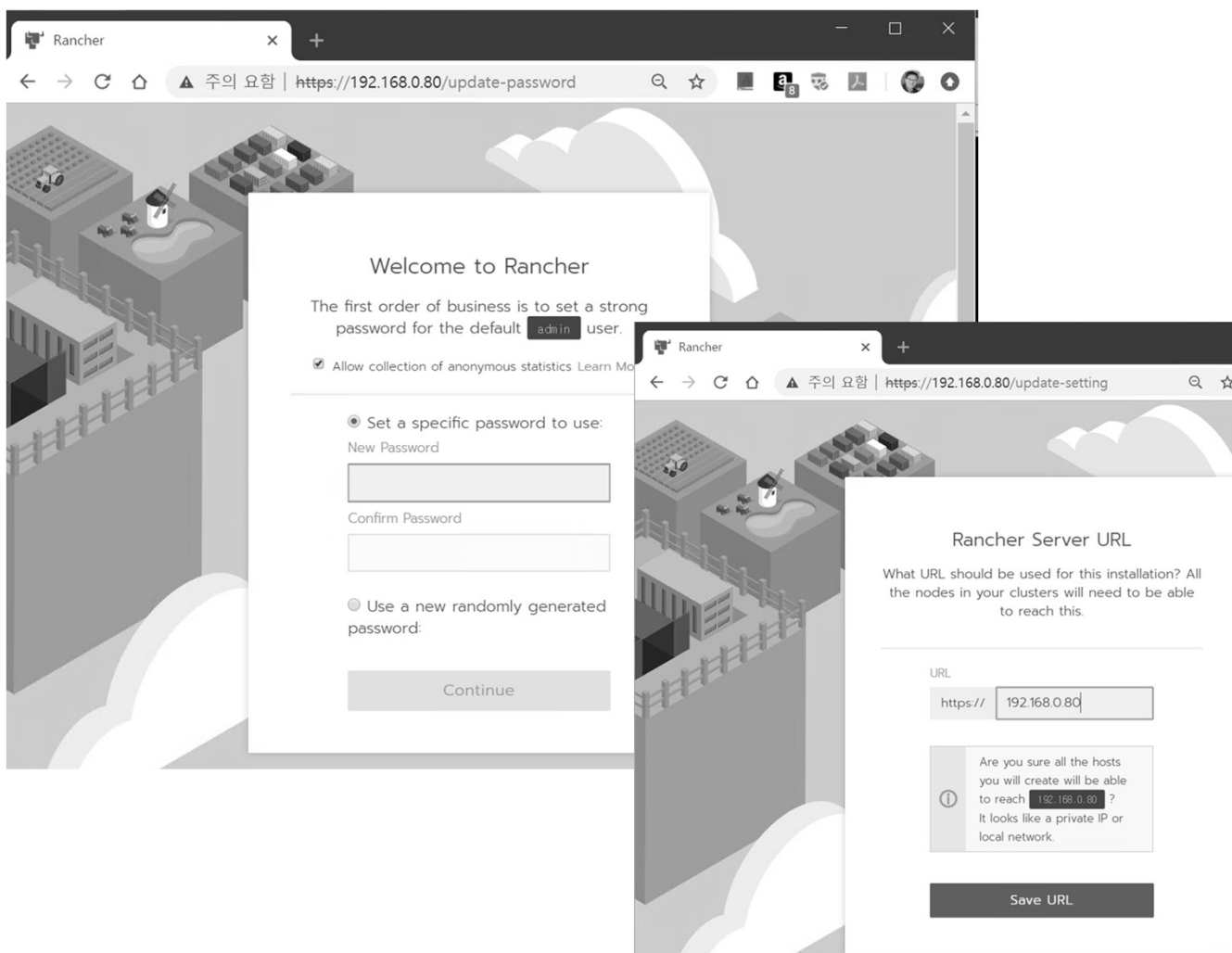
### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher 2.x installation (선택: CentOS 7 사용시)

- ① <http://192.168.0.xx:80>
- ② admin / jslab123
- ③ save



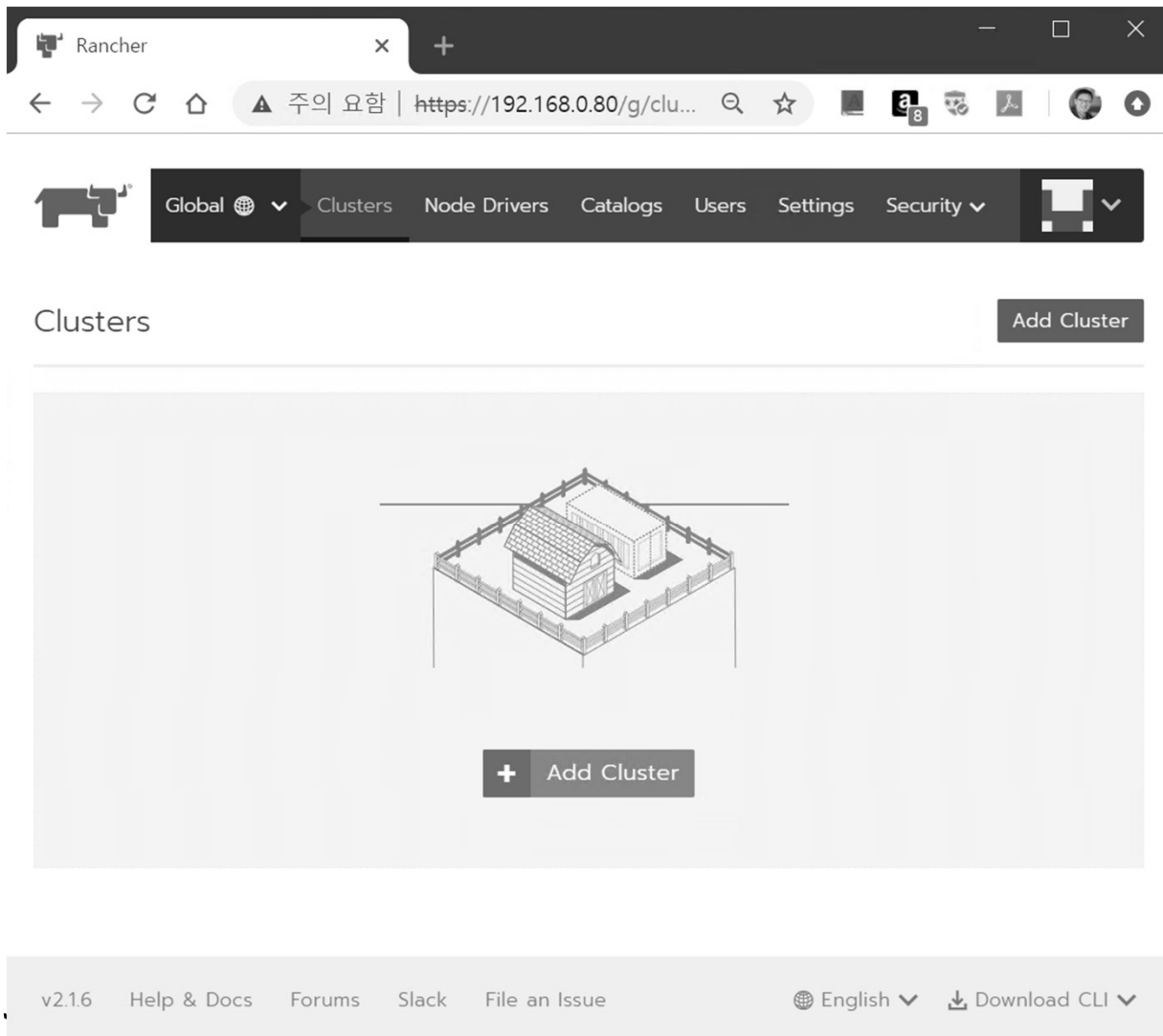
메모:



# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher 2.x installation (선택: CentOS 7 사용시)

- ① <http://192.168.0.xx:80>
- ② admin / jslab123
- ③ save



메모:

## IX. Cloud Networking (Rancher/K8s/Istio)

---

### ❖ Rancher 2.x Clusters

- ① From the Clusters page, click Add Cluster.
- ② Choose Custom.
- ③ Enter a Cluster Name.
- ④ Skip Member Roles and Cluster Options. We'll tell you about them later.
- ⑤ Click Next.
- ⑥ From Node Role, select all the roles: etcd, Control, and Worker.
- ⑦ Optional: Rancher auto-detects the IP addresses used for Rancher communication and cluster communication. You can override these using Public Address and Internal Address in the Node Address section.
- ⑧ Skip the Labels stuff. It's not important for now.
- ⑨ Copy the command displayed on screen to your clipboard.
- ⑩ Log in to your Linux host using your preferred shell, such as PuTTY or a remote Terminal connection. Run the command copied to your clipboard.
- ⑪ When you finish running the command on your Linux host, click Done.

#### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

## IX. Cloud Networking (Rancher/K8s/Istio)

---

### ❖ Linux Networking Operations @ Rancher 2.x

- ① **ifconfig**
- ② **ip route**
- ③ **docker network ls**
- ④ **docker network inspect xxxxxxx**
- ⑤ **sudo iptables -t nat -L -n** # K8s에서 생성한 NAT 확인

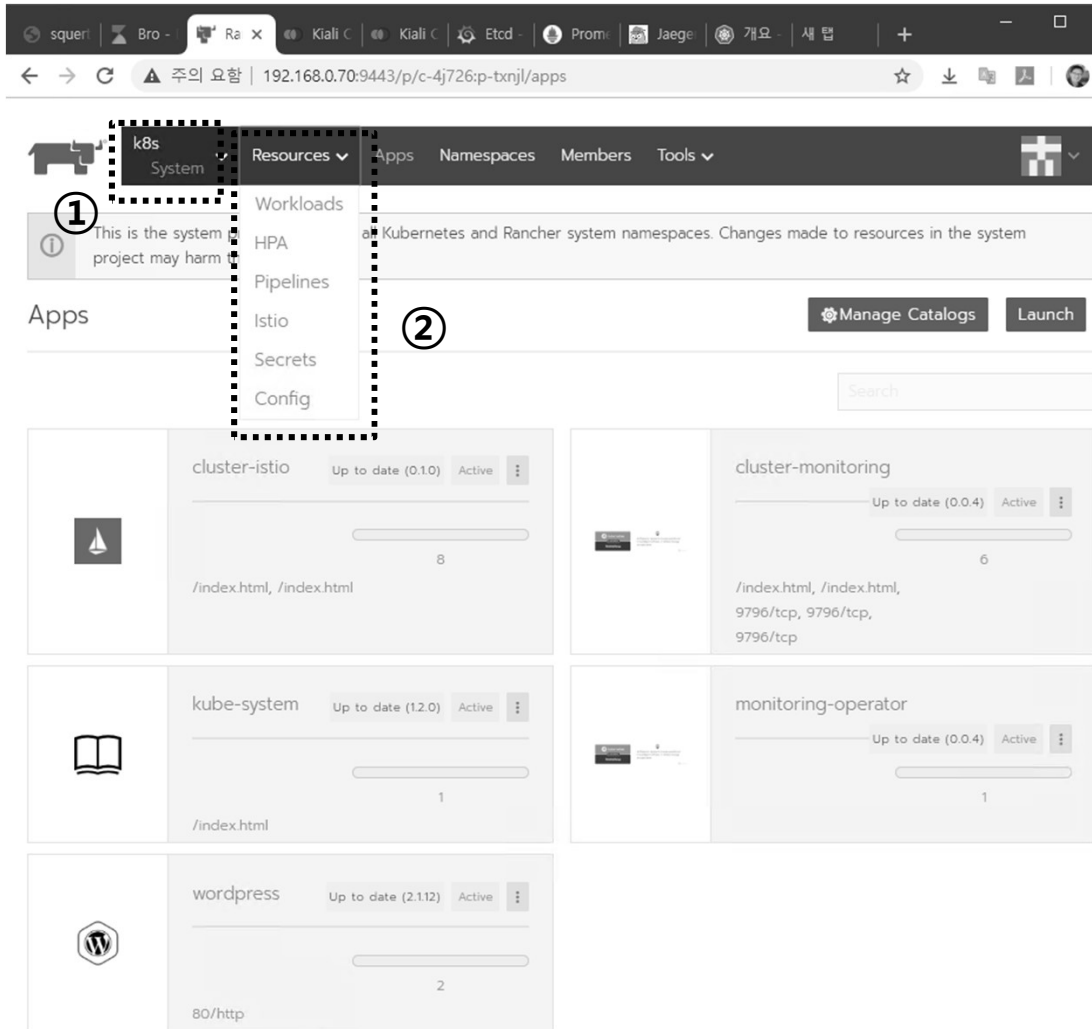
#### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Istio Installation @ Rancher 2.3

- ① Kubernetes Project 'System' 선택
- ② Resources 탭에서 Istio 선택



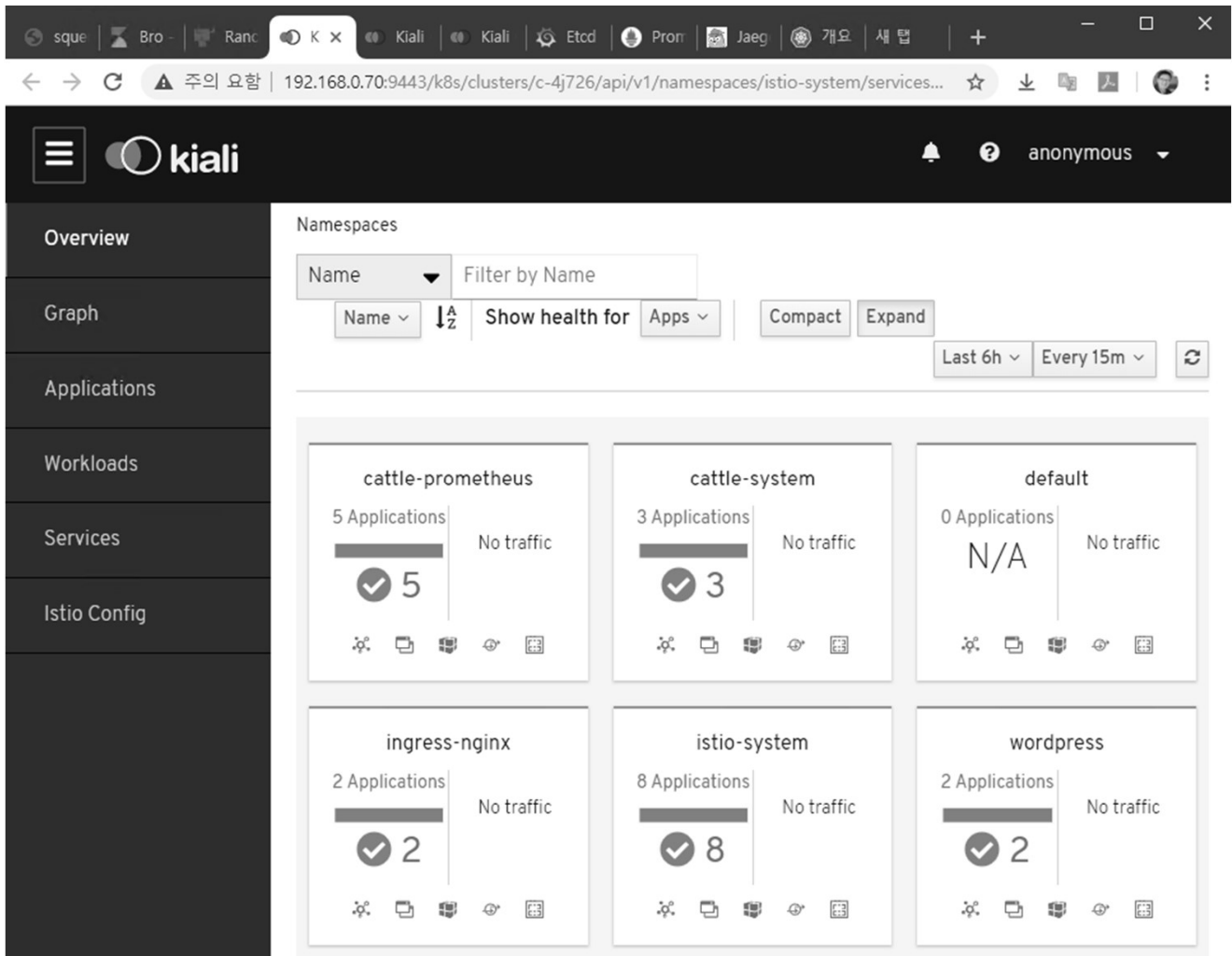
### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Istio Installation @ Rancher 2.3

### ① Kiali @ Istio



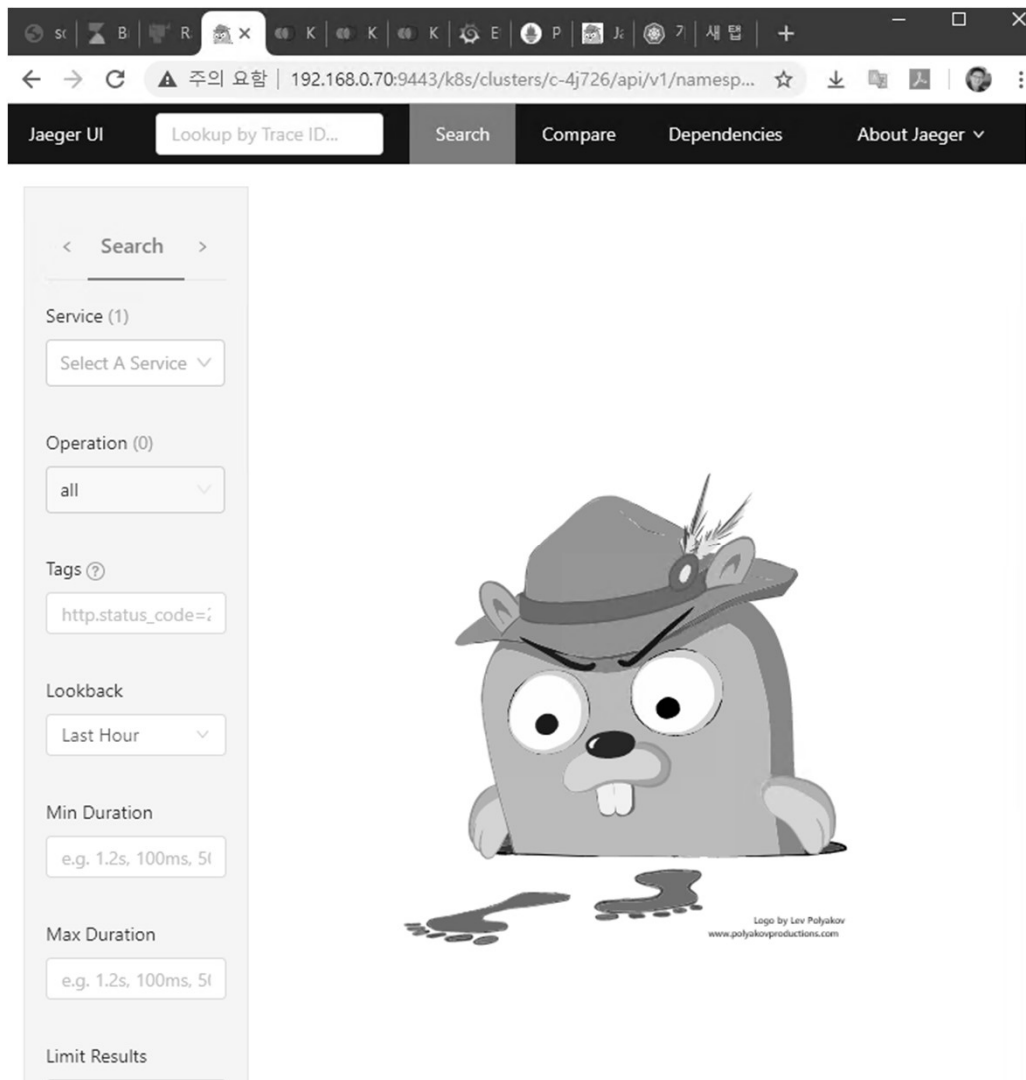
#### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Istio Installation @ Rancher 2.3

### ① Jaeger @ Istio



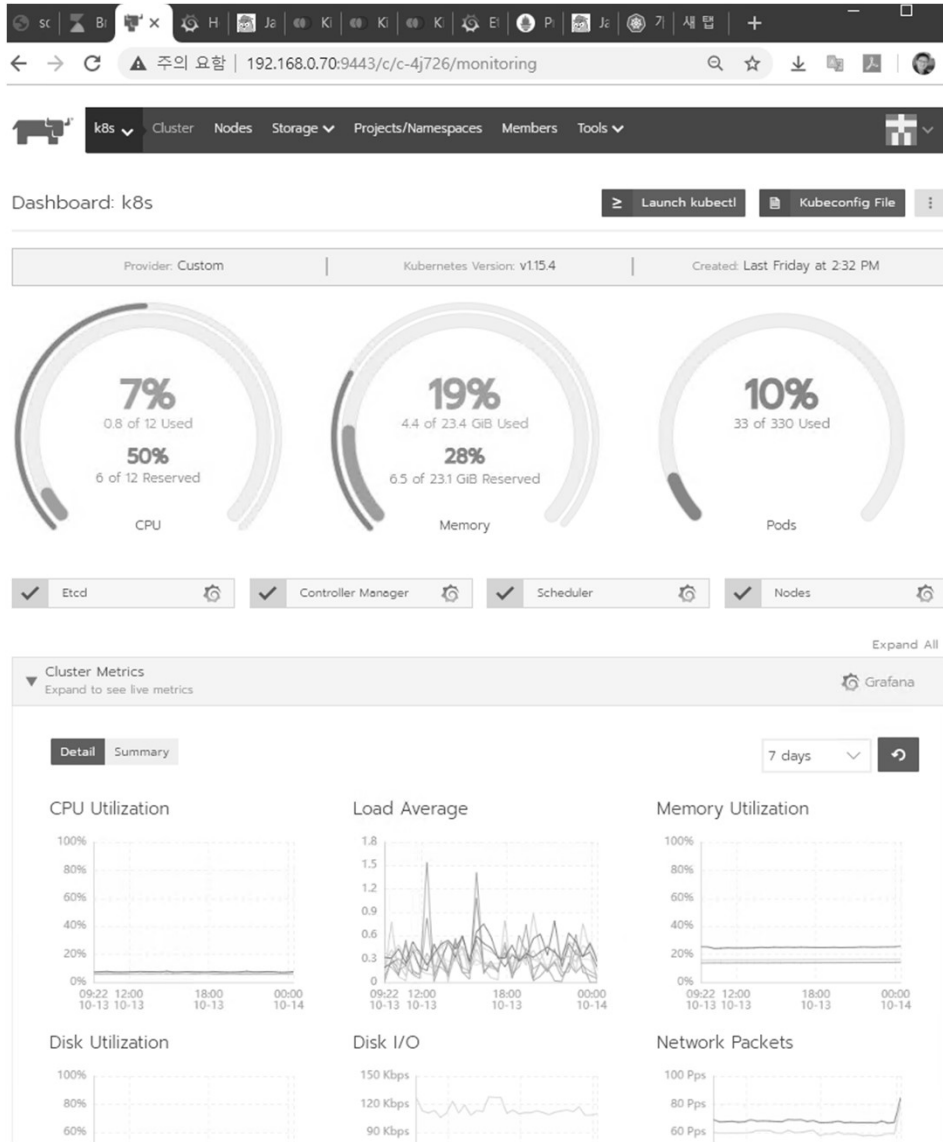
#### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Istio Installation @ Rancher 2.3

### ① Grafana @ Istio



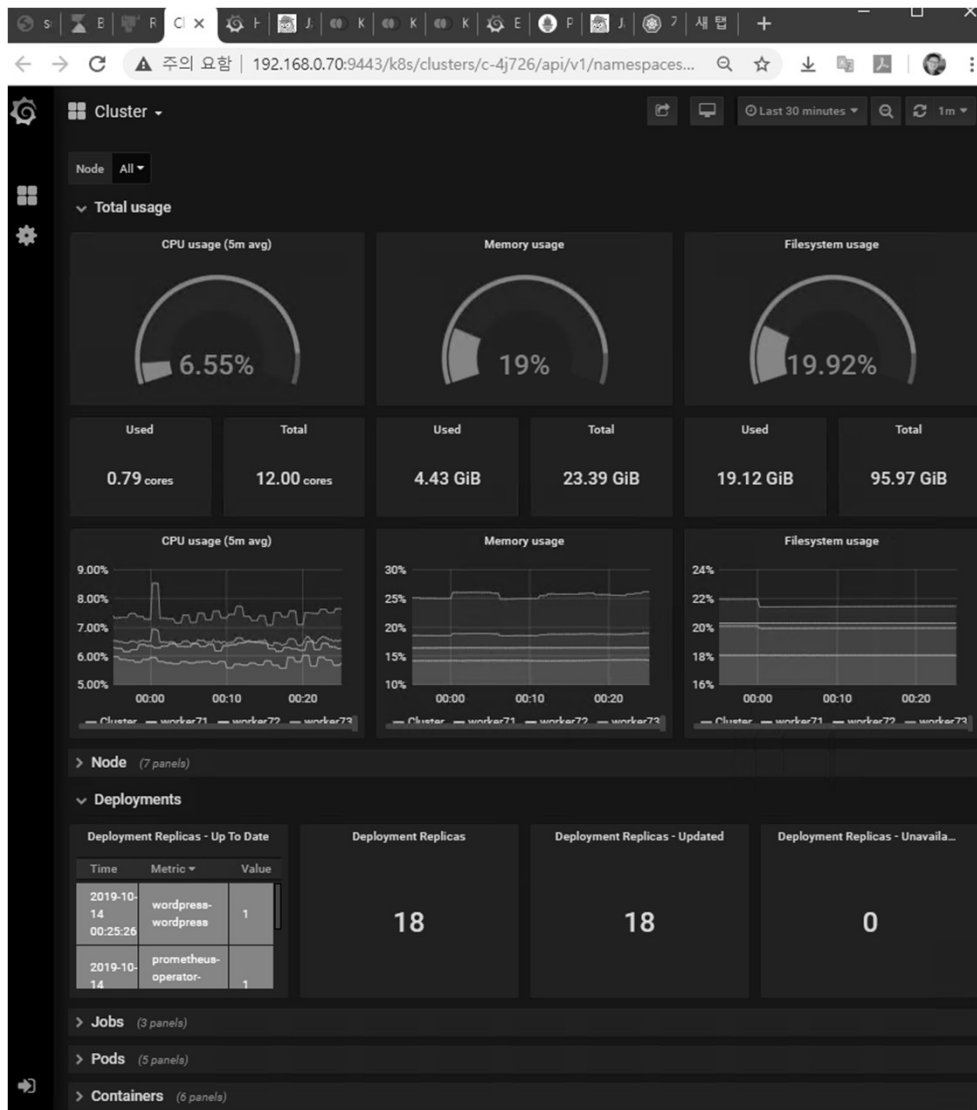
#### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Istio Installation @ Rancher 2.3

### ① Grafana @ Istio



#### 메모:

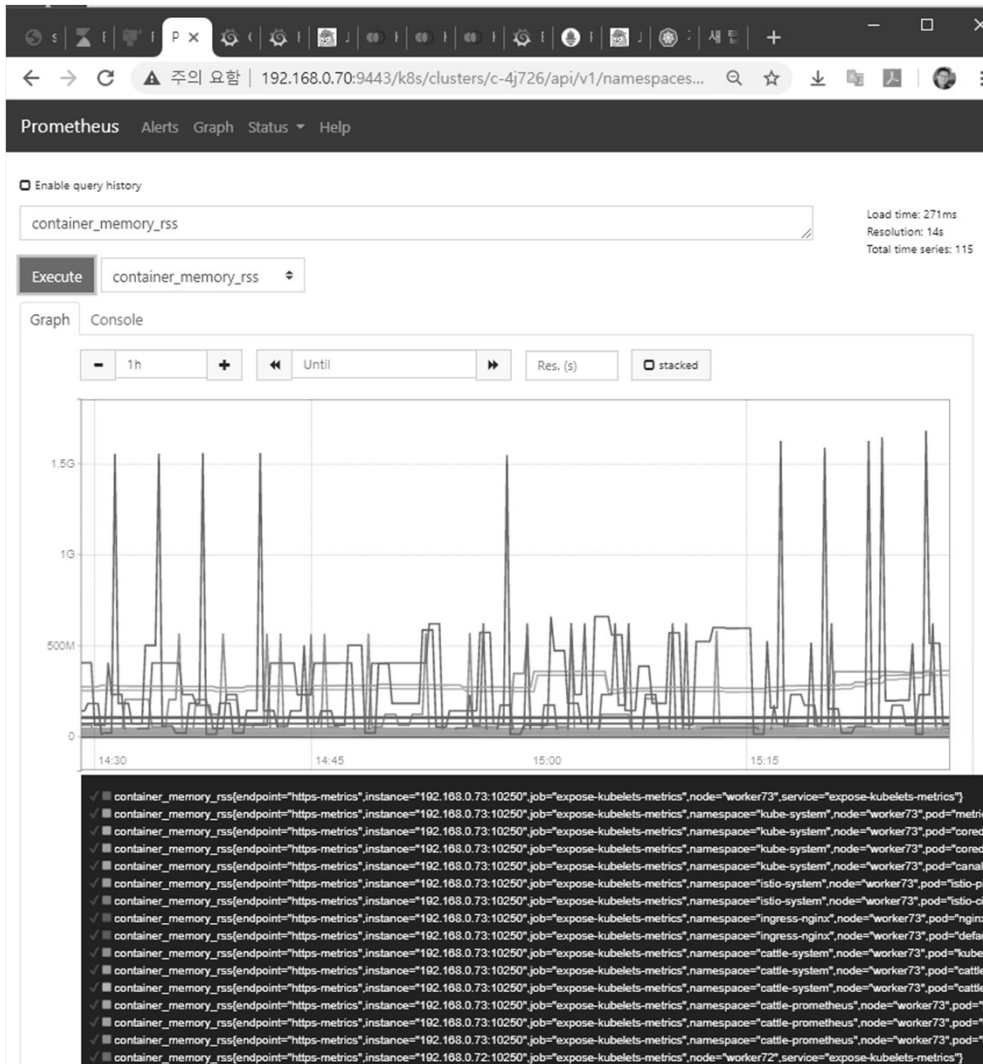
- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>



# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Istio Installation @ Rancher 2.3

### ① Prometheus @ Istio



#### 메모:

- <https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

# IX. Cloud Networking (Rancher/K8s/Istio)

---

## ❖ Rancher 1.x installation

- 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/**

james@jslab.kr

- 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`

# IX. Cloud Networking (Rancher/K8s/Istio)

---

## ❖ Rancher 1.x installation

- 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.howtforge.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`

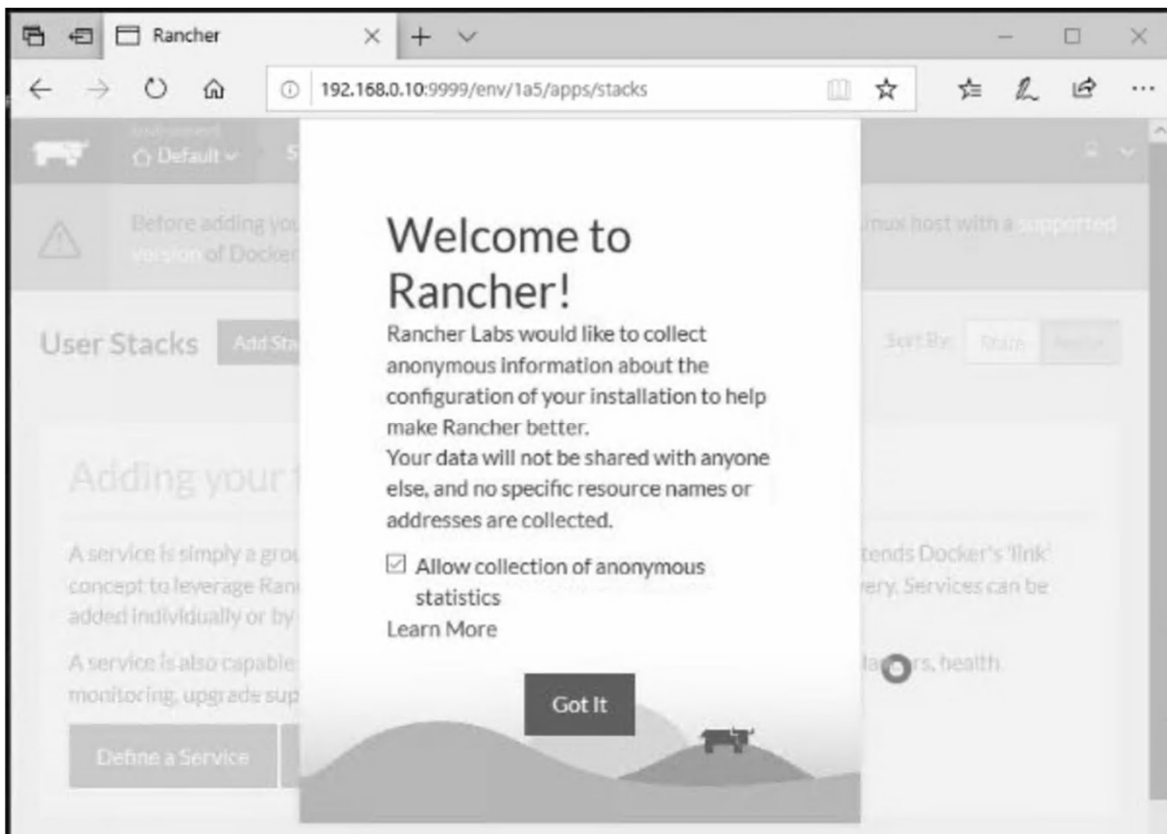
### 메모:

- 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`

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher 1.x installation

① <http://192.168.xx.xx:9999/>

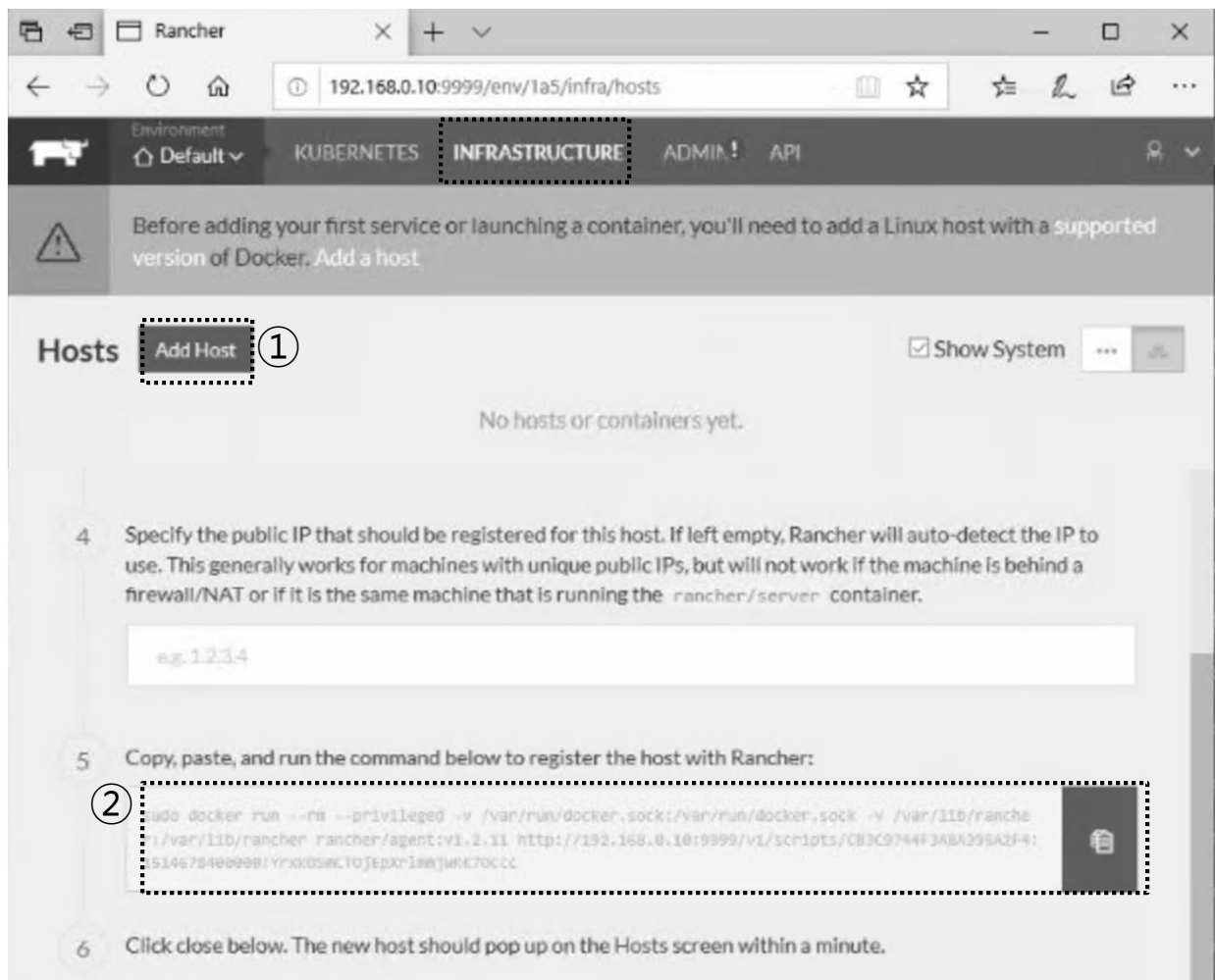


메모:

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher 1.x installation

- ① Add host @ Infrastructure
- ② Copy key for Paste @ worker01, worker02, worker03



메모:

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher 1.x installation

- ① Copy key
- ② Paste key @ worker01, worker02, worker03

```
root@worker01:~# sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.11 http://192.168.0.10:9999/v1/scripts/CB3C9744F3A8A395A2F4:1514678400000:YrxKOSmCTOjE pXrlmmjwKK7Occc
unable to find image 'rancher/agent:v1.2.11' locally
Trying to pull repository docker.io/rancher/agent:v1.2.11: Pulling from docker.io/rancher/agent
b3e1c725a85f: Pull complete
6a710864a9fc: Pull complete
d0ac3b234321: Pull complete
87f567b5cf58: Pull complete
063e24b217c4: Pull complete
d0a3f58caef0: Pull complete
16914729cfd3: Pull complete
bbad862633b9: Pull complete
3cf9849d7f3c: Pull complete
Digest: sha256:0fba3fb10108f7821596de5ad4bfa30e93426d034cd3471f6c0d3afb5f87a963
Status: Downloaded newer image for docker.io/rancher/agent:v1.2.11

INFO: Running Agent Registration Process, CATTLE_URL=http://192.168.0.10:9999/v1
INFO: Attempting to connect to: http://192.168.0.10:9999/v1
INFO: http://192.168.0.10:9999/v1 is accessible
INFO: Configured Host Registration URL info: CATTLE_URL=http://192.168.0.10:9999/v1 ENV_URL=http://192.168.0.10:9999/v1
INFO: Inspecting host capabilities
INFO: Boot2Docker: false
INFO: Host writable: true
INFO: Token: xxxxxxxx
INFO: Running registration
INFO: Printing Environment
INFO: ENV: CATTLE_ACCESS_KEY=19771183F76725F6DD2D
INFO: ENV: CATTLE_HOME=/var/lib/cattle
INFO: ENV: CATTLE_REGISTRATION_ACCESS_KEY=registrationToken
INFO: ENV: CATTLE_REGISTRATION_SECRET_KEY=xxxxxxx
INFO: ENV: CATTLE_SECRET_KEY=xxxxxxx
INFO: ENV: CATTLE_URL=http://192.168.0.10:9999/v1
INFO: ENV: DETECTED CATTLE_AGENT_IP=192.168.0.11
```

Copy, paste, and run the command below to register the host with Rancher.

```
sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.11 http://192.168.0.10:9999/v1/scripts/CB3C9744F3A8A395A2F4:1514678400000:YrxKOSmCTOjE pXrlmmjwKK7Occc
```

Click close below. The new host should pop up on the Hosts screen within a minute.

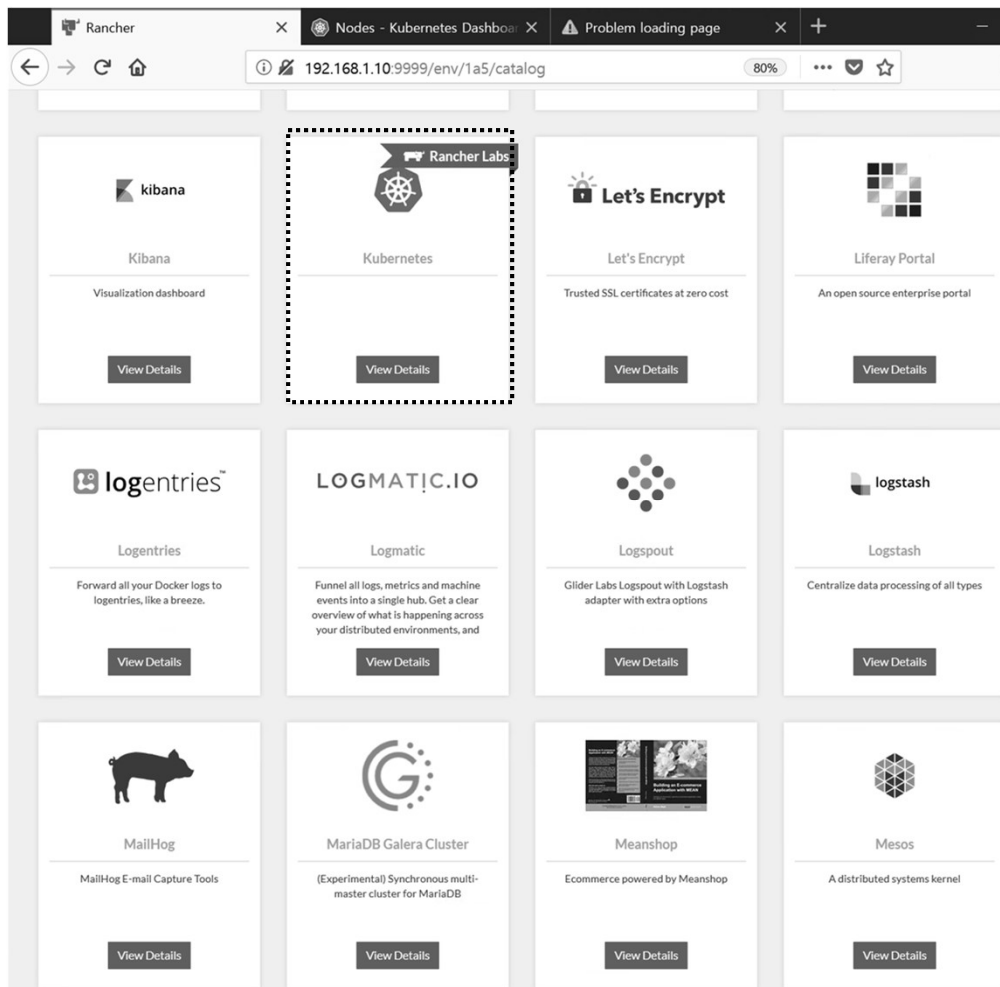
### 메모:

- 아마존 클라우드 서비스 AWS 상의 Docker Hub가 연결 되어야 함

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Rancher 1.x installation

- ① Catalog @ Infrastructure
- ② Check Kubernetes

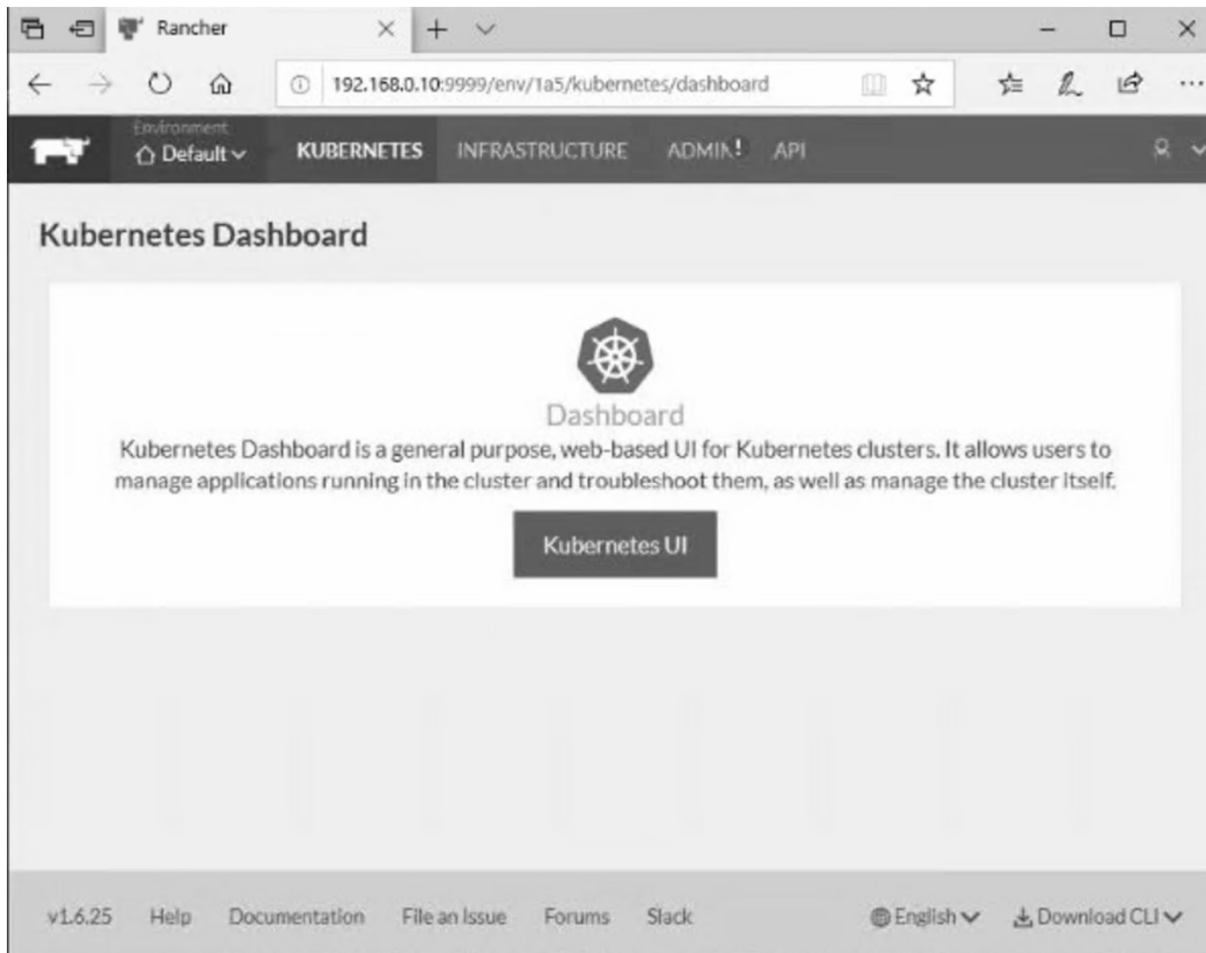


메모:

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ K8s 1.x installation

- ① Dashboard @ Kubernetes
- ② CLI @ Kubernetes



james@jslab.kr

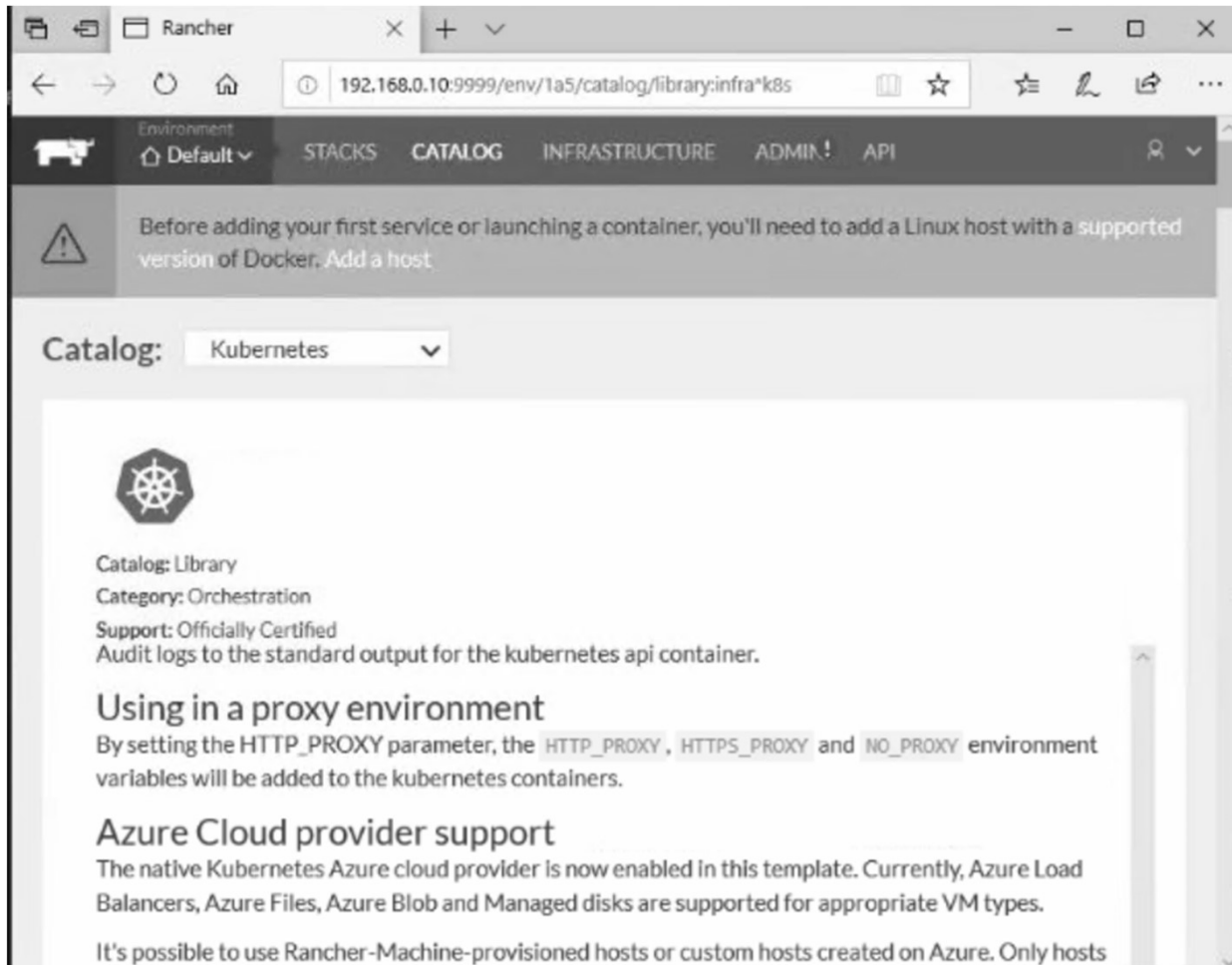
메모:



# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ K8s 1.x installation

- ① <http://192.168.0.10:9999/>
- ② K8s @ Catalog 선택 Launch

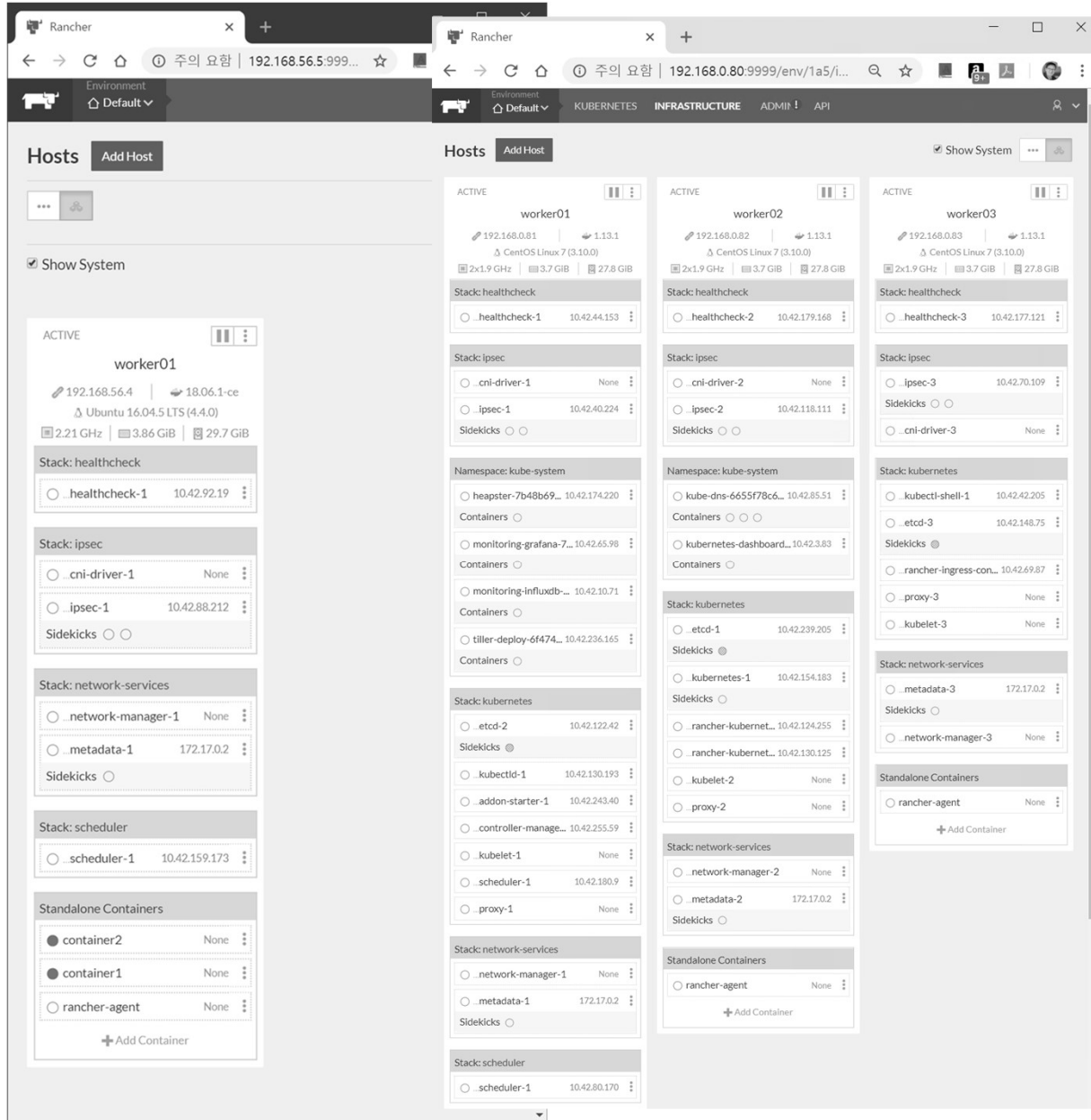


### 메모:

- Kubernetes 대쉬보드 접속은 K8s 실행 후 수분 소요

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Infrastructure/Hosts after K8s installation

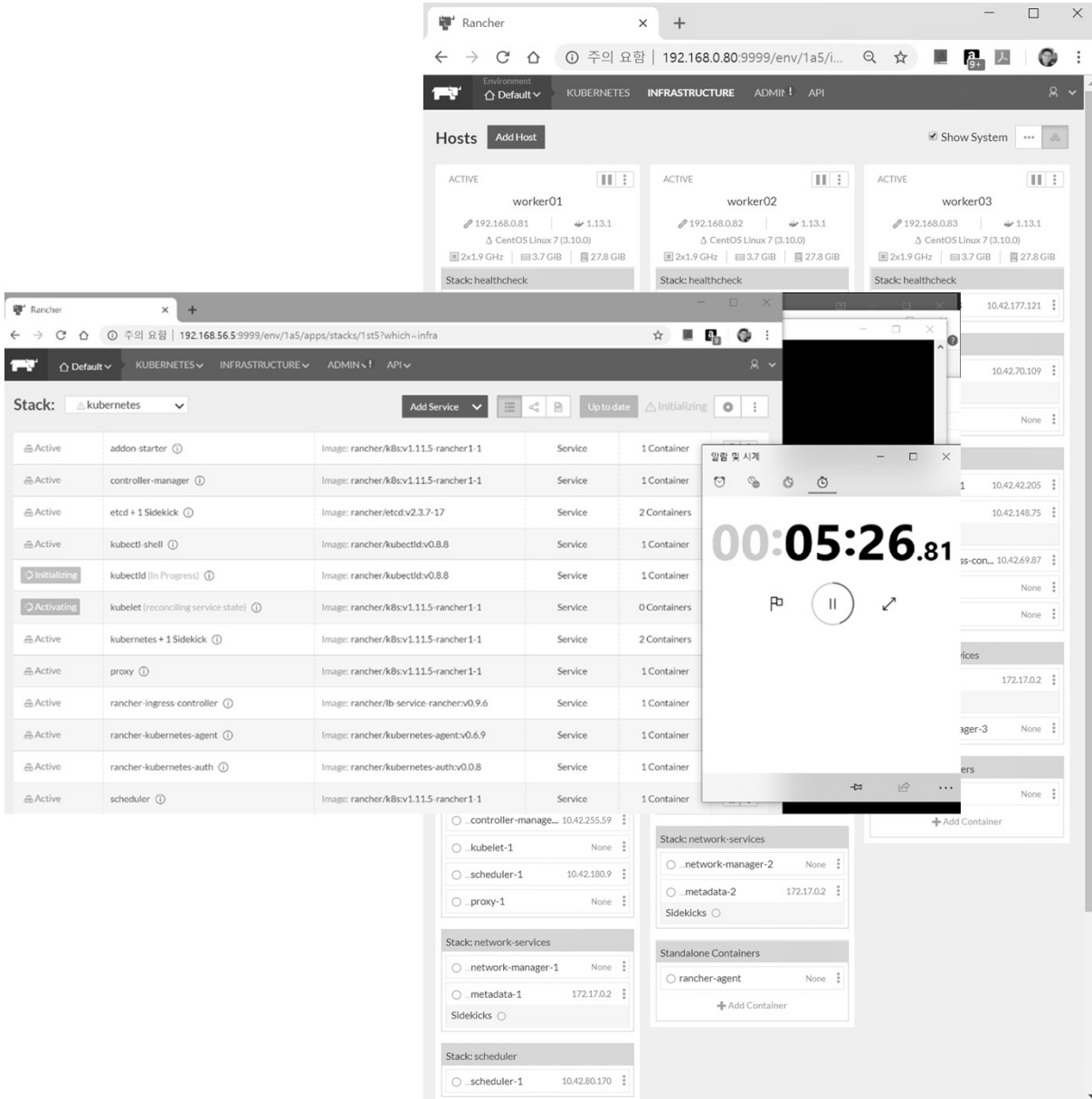


### 메모:

- 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)

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ 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

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Infrastructure/Hosts K8s @ CentOS7 minimal

The screenshot shows the Rancher web interface. The main view displays the 'Stack: kubernetes' with a list of services including 'addon-starter', 'controller-manager', 'etcd + 1 Sidekick', 'kubectl-shell', 'kubectld', 'kubelet', 'kubernetes + 1 Sidekick', 'proxy', 'rancher-ingress-controller', 'rancher-kubernetes-agent', 'rancher-kubernetes-auth', and 'scheduler'. A '작업 관리자' (Task Manager) window is overlaid, showing system resource usage: CPU at 16% (1.57GHz), Memory at 11.6/15.9GB (73%), and Disk usage at 0% for both C: and D: drives. The '이더넷' (Network) section is expanded, showing a table for 'Npcap Loopback Adapter' with columns for '처리량' (Throughput) and '100Kbps'. Below the table, network details are listed: '보내기' (Send) at 0Kbps, '받기' (Receive) at 0Kbps, '어댑터 이름: Npcap Loopback Adapter', '연결 형식: 이더넷', 'IPv4 주소: 169.254.147.86', and 'IPv6 주소: fe80:a0d3:b244cb2:9356%9'.

### 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

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Deployment

- ① + Create
- ② Copy and Paste 'deployment of nginx'
- ③ Upload

```
1 apiVersion: apps/v1 # for versions before 1.9.8 use apps/v1beta2
2 kind: Deployment
3 metadata:
4   name: nginx-deployment
5 spec:
6   selector:
7     matchLabels:
8       app: nginx
9   replicas: 2
10  template:
11    metadata:
12      labels:
13        app: nginx
14    spec:
15      containers:
16        - name: nginx
17          image: nginx:1.8 # Update the version of nginx from 1.7.9 to 1.8
18          ports:
19            - containerPort: 80
```

### 메모:

- Replicas Updates

# IX. Cloud Networking (Rancher/K8s/Istio)

## ❖ Infrastructure Stacks

### ① ingress

The screenshot shows the Rancher web interface for Infrastructure Stacks. The page title is 'Infrastructure Stacks' and it includes buttons for 'Add Stack' and 'Add from Catalog'. A table lists several stacks with their status and resource counts:


Stack Name	Status	Services	Containers
healthcheck	Up to date	1	3
ipsec	Up to date	2	12
kubernetes	Up to date	12	22
kubernetes-ingress-lbs	Up to date	0	0
istio	Up to date	2	9
scheduler	Up to date	1	1

A dropdown menu is open for the 'Add Service' button, showing options: 'Add Load Balancer', 'Add Service Alias', and 'Add External Service'. A circled '2' highlights this menu.

#### 메모:

- CLI
- Replicas Updates

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

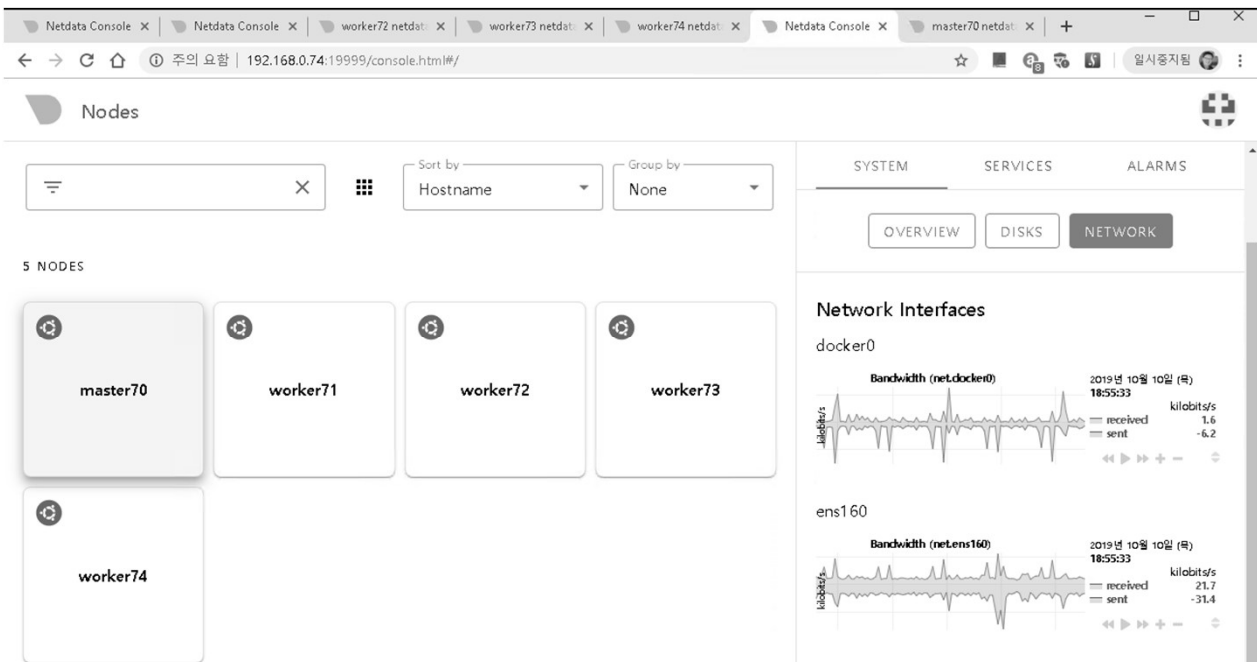




# X. 도구

## ❖ netdata

- ① <http://127.0.0.1:19999>
- ② 클러스터 구성 Host Node 등록 w/Account



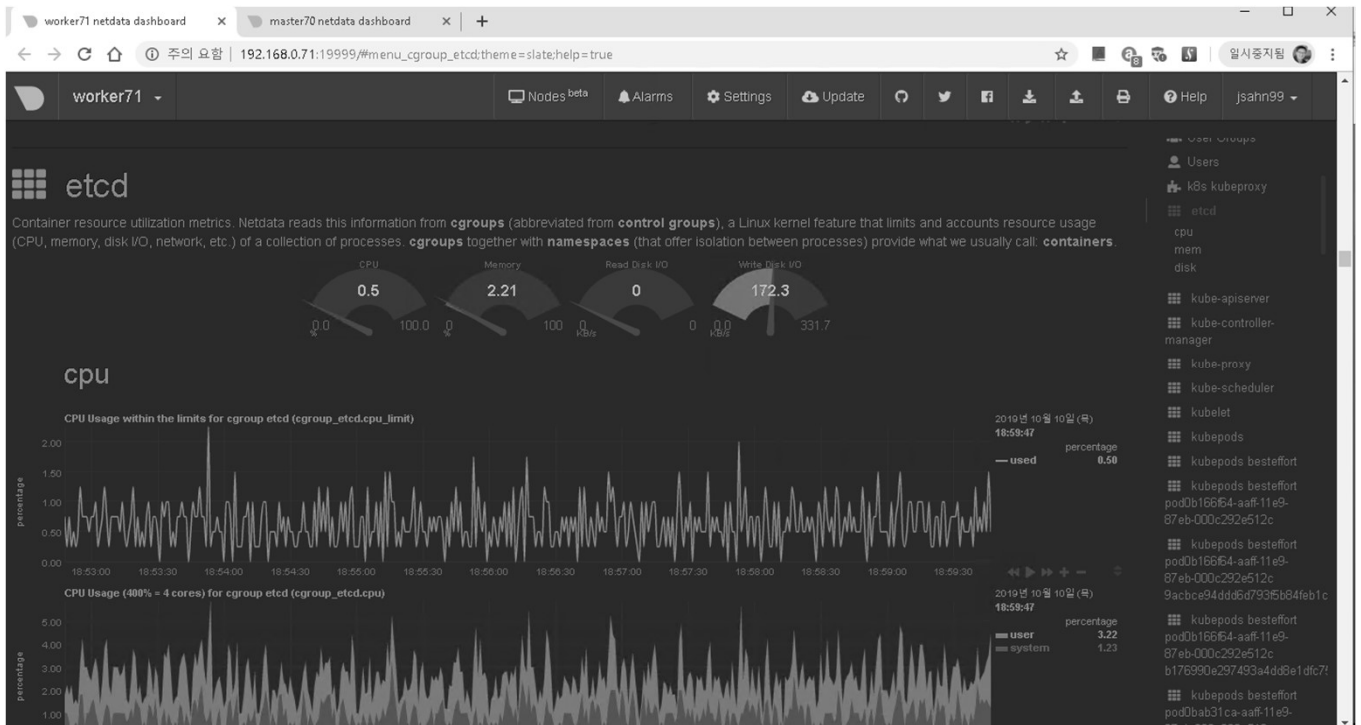
### 메모:

- Ubuntu Server 16.04 확인

# X. 도구

## ❖ netdata

- ① <http://127.0.0.1:19999>
- ② Users (namespace for K8s)



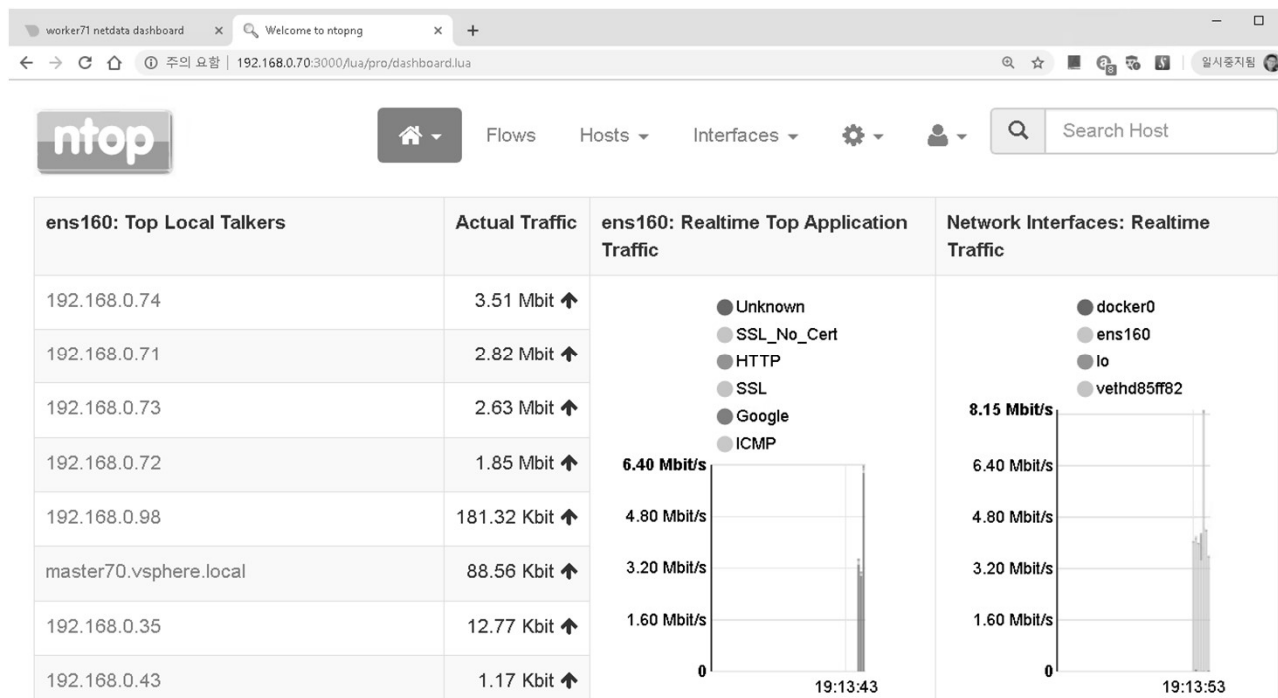
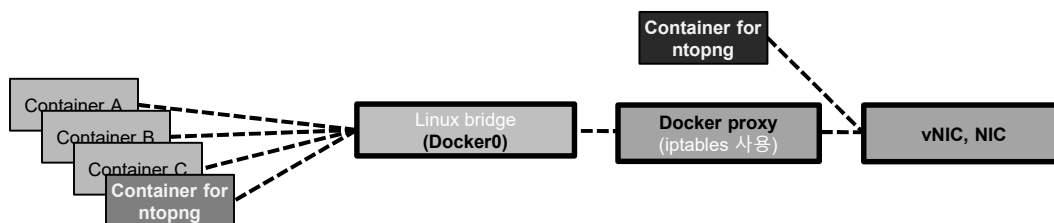
### 메모:

- Ubuntu Server 16.04 확인

# X. 도구

## ❖ ntopng (선택 설치 별 설명)

- ① **sudo docker run -t -p 3000:3000 lucaderi/ntopng-docker**
- ② **sudo docker run --net=host -t lucaderi/ntopng-docker**
- ③ **http://hostIPaddress:3000 # admin/admin**



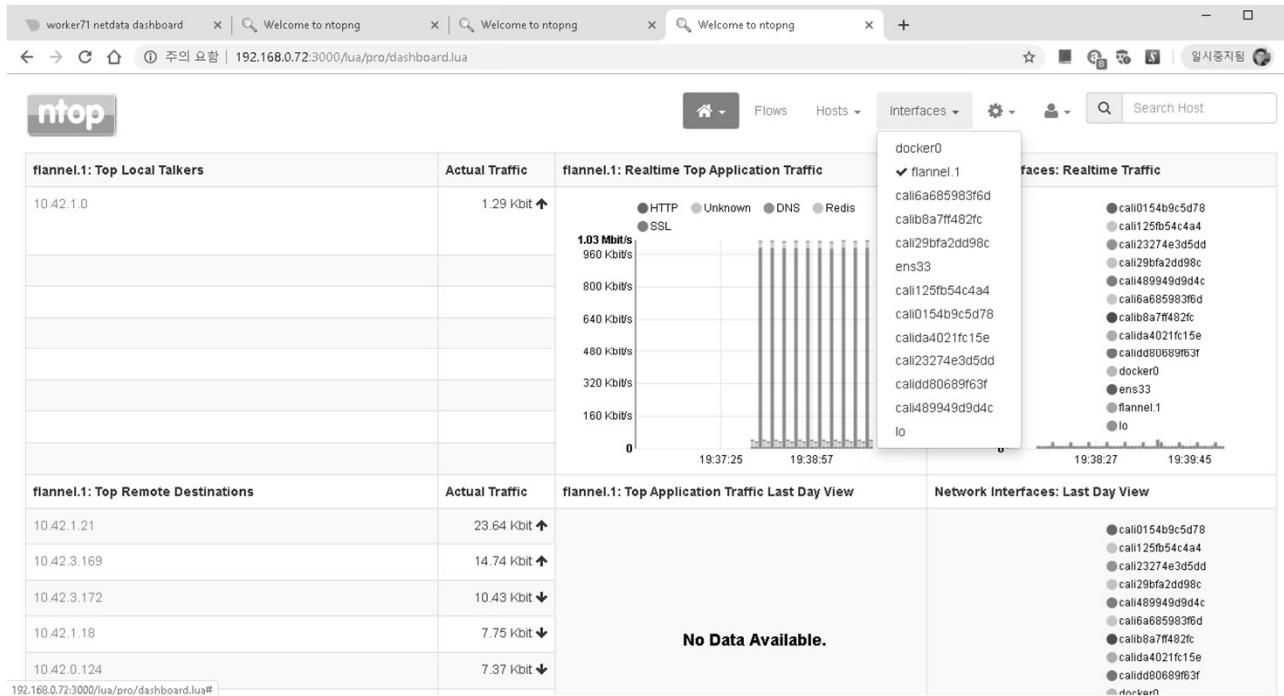
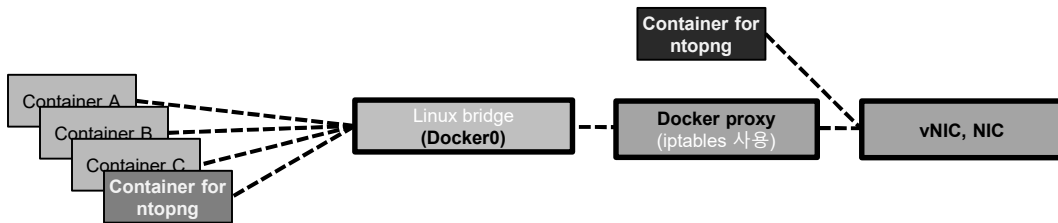
### 메모:

- Ubuntu Server 16.04 확인
- Interface 확인

# X. 도구

## ❖ ntopng (선택 설치 별 설명)

- ① **sudo docker run -t -p 3000:3000 lucaderi/ntopng-docker**
- ② **sudo docker run --net=host -t lucaderi/ntopng-docker**
- ③ **http://hostIPaddress:3000 # admin/admin**



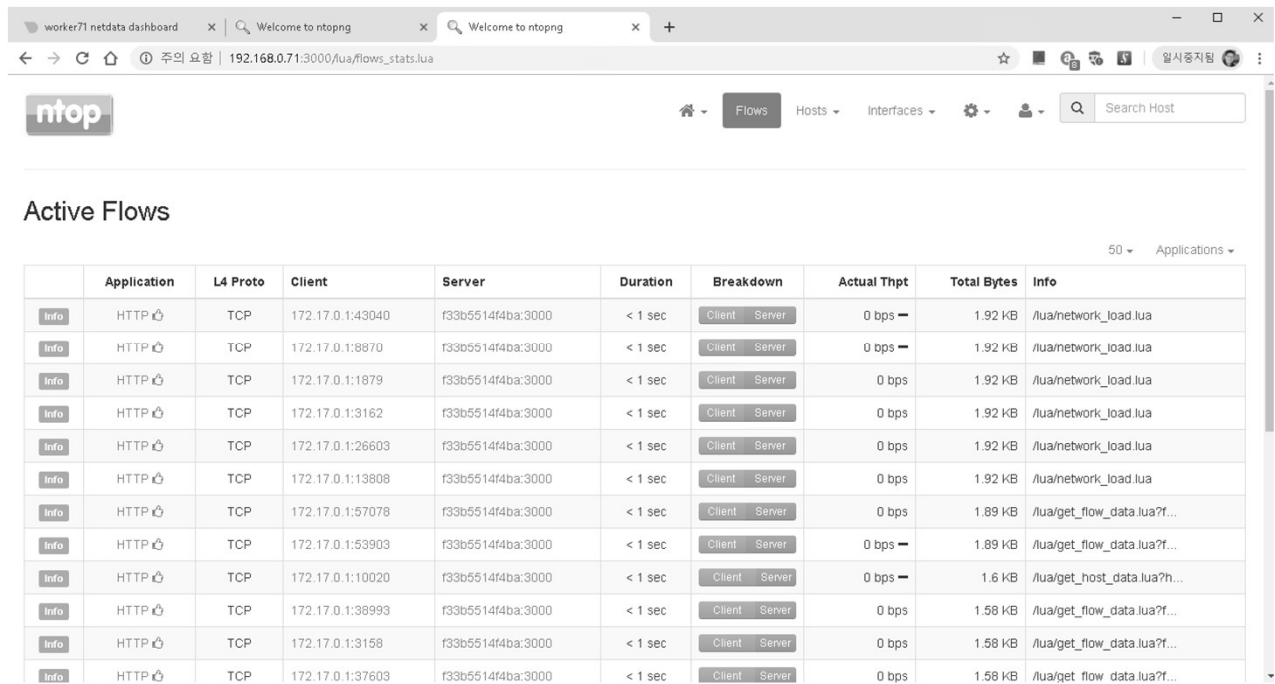
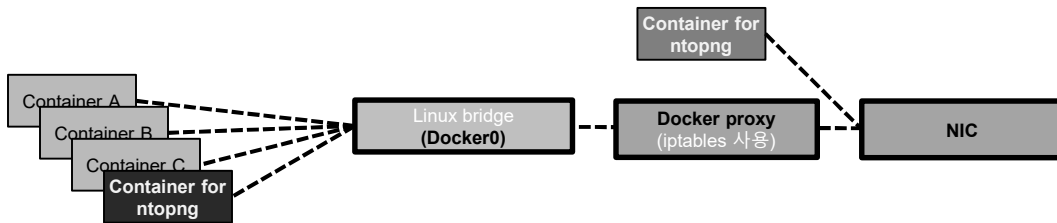
### 메모:

- Ubuntu Server 16.04 확인
- Interface 확인

# X. 도구

## ❖ ntopng (선택 설치 별 설명)

- ① **sudo docker run -t -p 3000:3000 lucaderi/ntopng-docker**
- ② **sudo docker run --net=host -t lucaderi/ntopng-docker**
- ③ **http://hostIPaddress:3000 # admin/admin**



### 메모:

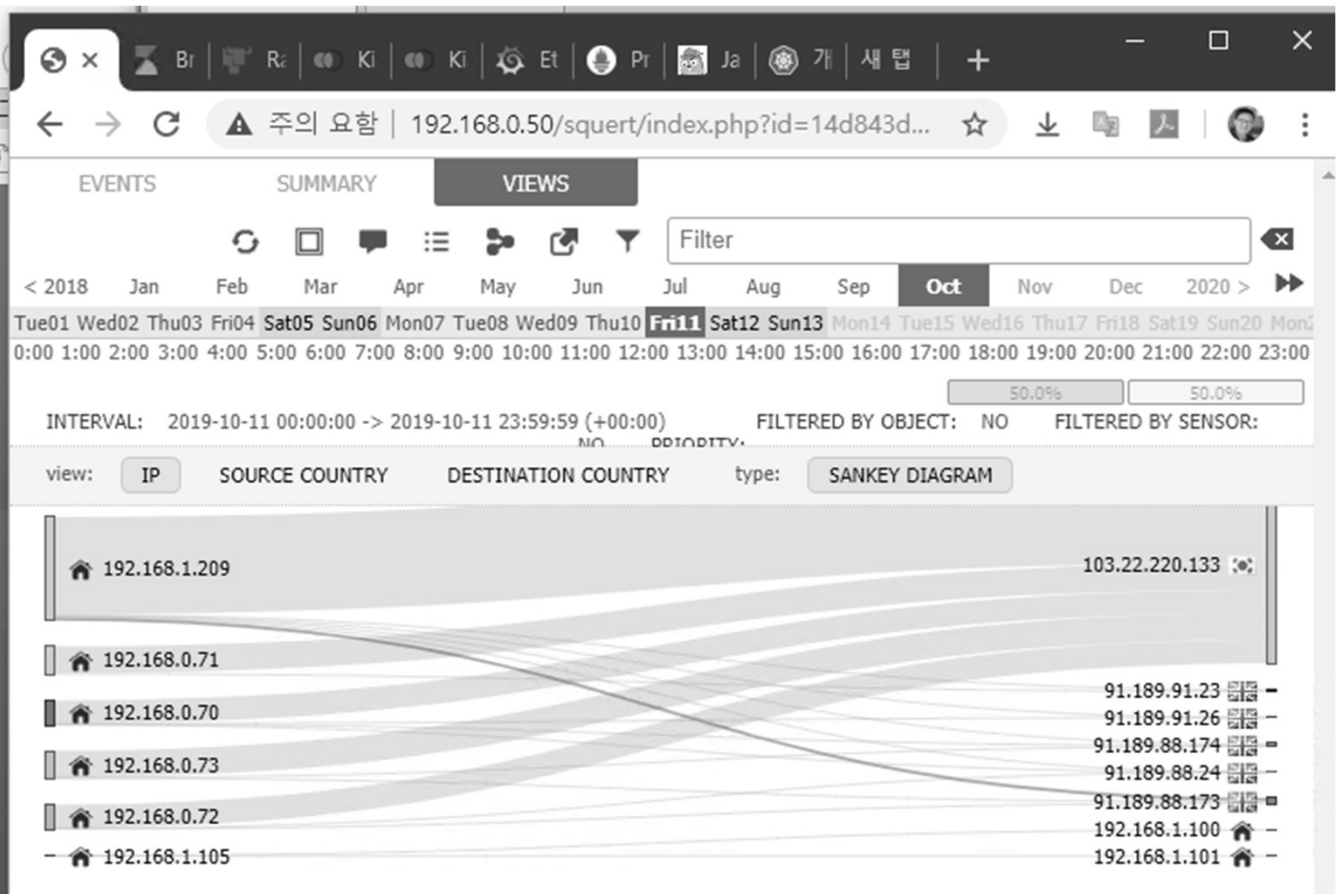
- Ubuntu Server 16.04 확인
- Interface 확인

# X. 도구

## ❖ Security Onion @ Hypervisor

- ① Squert
- ② Kibana
- ③ Snort / Suricata / Bro

james@jslab.kr



메모:

# X. 도구

## ❖ Security Onion @ Hypervisor

### ① Squert

QUEUE	SC	DC	ACTIVITY	LAST EVENT	SIGNATURE	ID	PROTO	% TOTAL
21	4	4		23:50:07	ET CINS Active Threat Intelligence Poor Reputation IP TCP group 38	2403374	6	21.212%
2	1	2		23:06:36	ET CINS Active Threat Intelligence Poor Reputation IP TCP group 10	2403318	6	2.020%
30	2	2		22:22:33	GPL SNMP public access udp	2101411	17	30.303%
4	1	1		21:25:35	ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management	2013504	6	4.040%
33	16	4		20:18:59	ET SCAN Suspicious inbound to MSSQL port 1433	2010935	6	33.333%
1	1							%
6	7	1						%
1	1							%
1	1							%

CATEGORIZE 33 EVENT(S)    CREATE FILTER: [src](#) [dst](#) [both](#)

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	%
1		2019-10-11 20:18:59	211.69.161.100	2	CHINA (.cn)	203.255.251	%
1		2019-10-11 20:18:58	211.69.163.226	2	CHINA (.cn)	203.255.251	%
1		2019-10-11 18:16:57	211.69.163.226	2	CHINA (.cn)	203.255.251	%
1		2019-10-11 18:16:54	211.69.161.100	2	CHINA (.cn)	203.255.251	%
1		2019-10-11 16:51:25	161.53.116.99	2	CROATIA (.hr)	203.255.251	%
1		2019-10-11 15:31:11	193.193.244.196	2	KAZAKHSTAN (.kz)	203.255.251	%
1		2019-10-11 15:28:38	161.53.116.99	2	CROATIA (.hr)	203.255.251	%
1		2019-10-11 14:29:51	193.193.244.196	2	KAZAKHSTAN (.kz)	203.255.251	%
1		2019-10-11 14:27:05	161.53.116.99	2	CROATIA (.hr)	203.255.251	%
1		2019-10-11 12:53:32	171.67.70.84	3	UNITED STATES (.us)	203.255.251	%
1		2019-10-11 12:39:32	171.67.70.86	3	UNITED STATES (.us)	203.255.251	%
1		2019-10-11 12:33:39	171.67.70.92	3	UNITED STATES (.us)	203.255.251	%
1		2019-10-11 12:08:38	77.235.23.197	2	KYRGYZSTAN (.kg)	203.255.251	%

메모:

# X. 도구

## ❖ Security Onion @ Hypervisor

- ① Squert
- ② Kibana
- ③ Snort / Suricata / Bro

The screenshot shows the Kibana dashboard interface. The main content area displays several widgets:

- Total Number of Logs:** A large central widget showing the count **170,230**.
- Total Log Count Over Time:** A line graph showing log counts over a 24-hour period, with a peak around 12:00.
- All Sensors - Log Type:** A table listing various log types and their counts.
- DNS - Server:** A table listing IP addresses and their corresponding log counts.
- NIDS - Classification:** A table showing the classification of network intrusion detection system alerts.

Log Type(s)	Count
bro_conn	40,046
ossec	33,818
bro_files	26,092
CRON	24,750
bro_ssl	15,193
bro_dns	12,330

Server	Count
8.8.8.8	4,221
1.1.1.1	3,290
192.168.0.233	2,788
8.8.4.4	1,431
224.0.0.251	232
224.0.0.252	161
203.255.251.111	123
ff02::fb	34
ff02::1:3	17
203.255.251.99	8

Classification	Count
Misc Attack	63
Attempted Information Leak	31
Potentially Bad Traffic	29
Not Suspicious Traffic	4
Misc activity	1

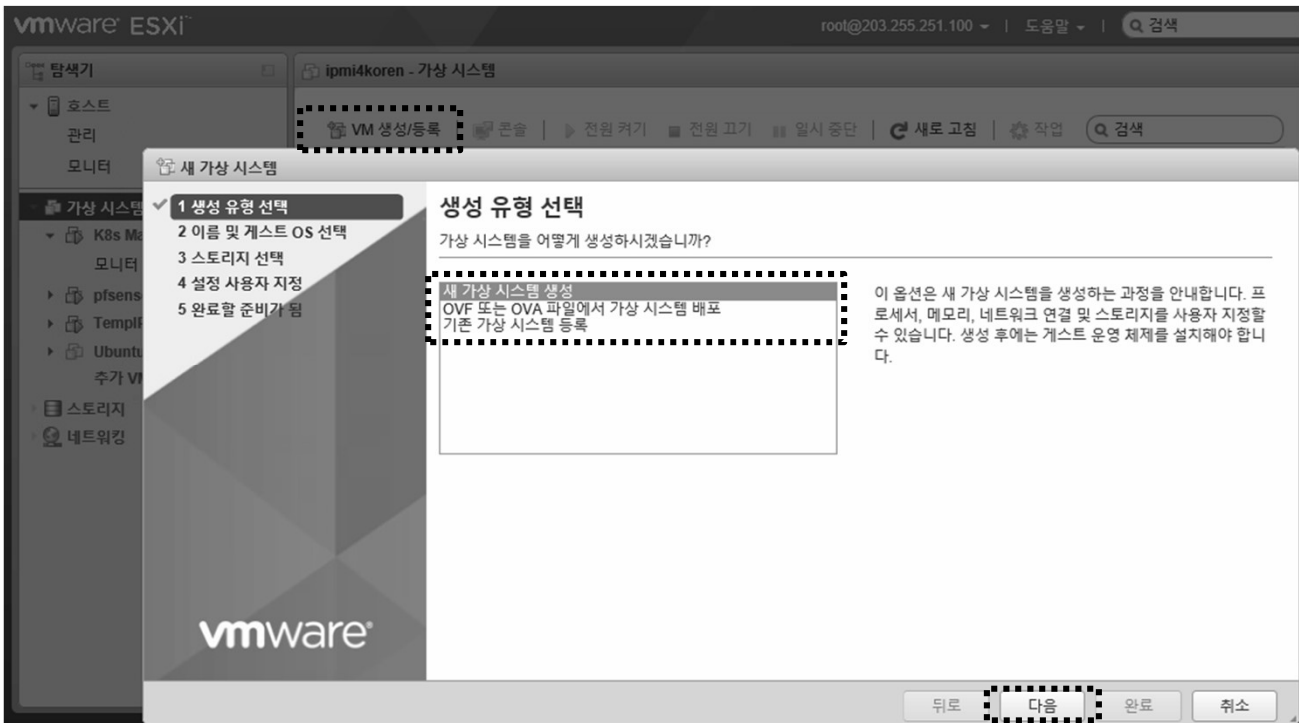
메모:



# X. 도구

## ❖ Security Onion @ Hypervisor

- ① VM 생성 등록
- ② 새 가상 시스템 생성
- ③ 다음



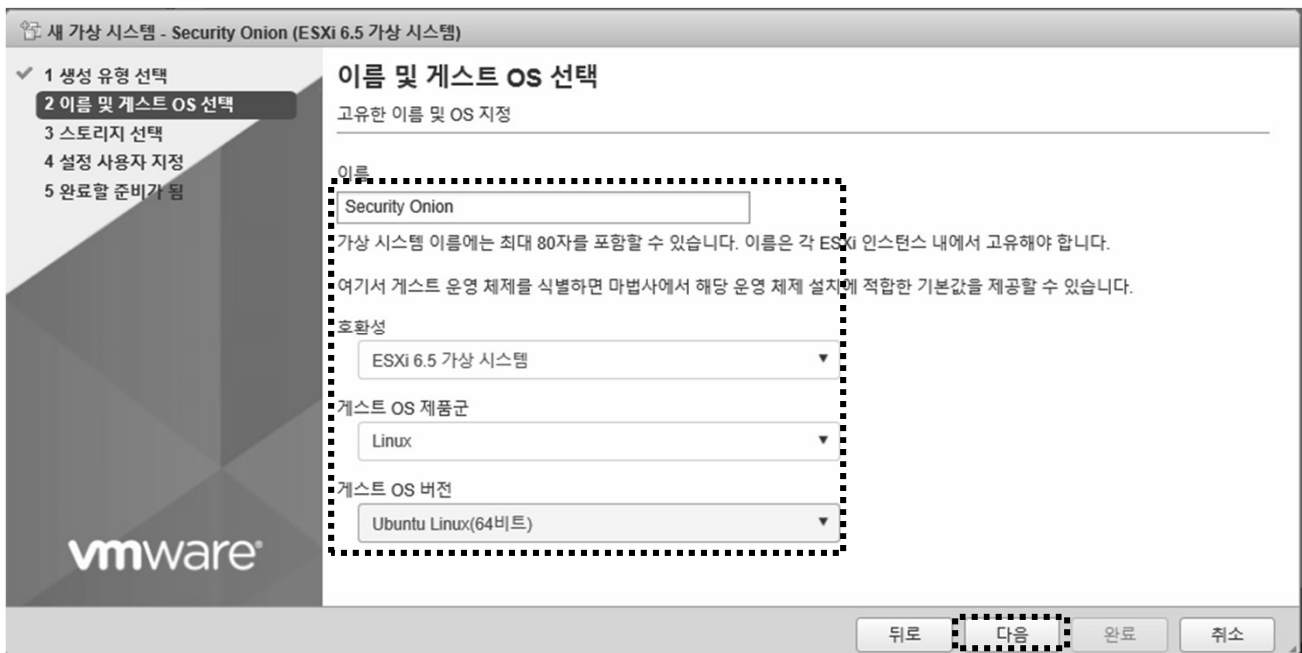
### 메모:

- ESXi 다운로드 주소: <https://my.vmware.com/en/web/vmware/evalcenter?p=free-esxi6>
- 디스크 이미지 굽기: Rufus 도구 사용 <https://rufus.akeo.ie/>
- Disk Imager <https://sourceforge.net/projects/win32diskimager/files/latest/download>
- USB 부팅 제가동은 전원 off/on (전원 케이블 포함)필요함
- [https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify\\_ISO.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md)

# X. 도구

## ❖ Security Onion @ Hypervisor

- ① 이름
- ② 호환성
- ③ 게스트 OS 제품군
- ④ 게스트 OS 버전
- ⑤ 다음



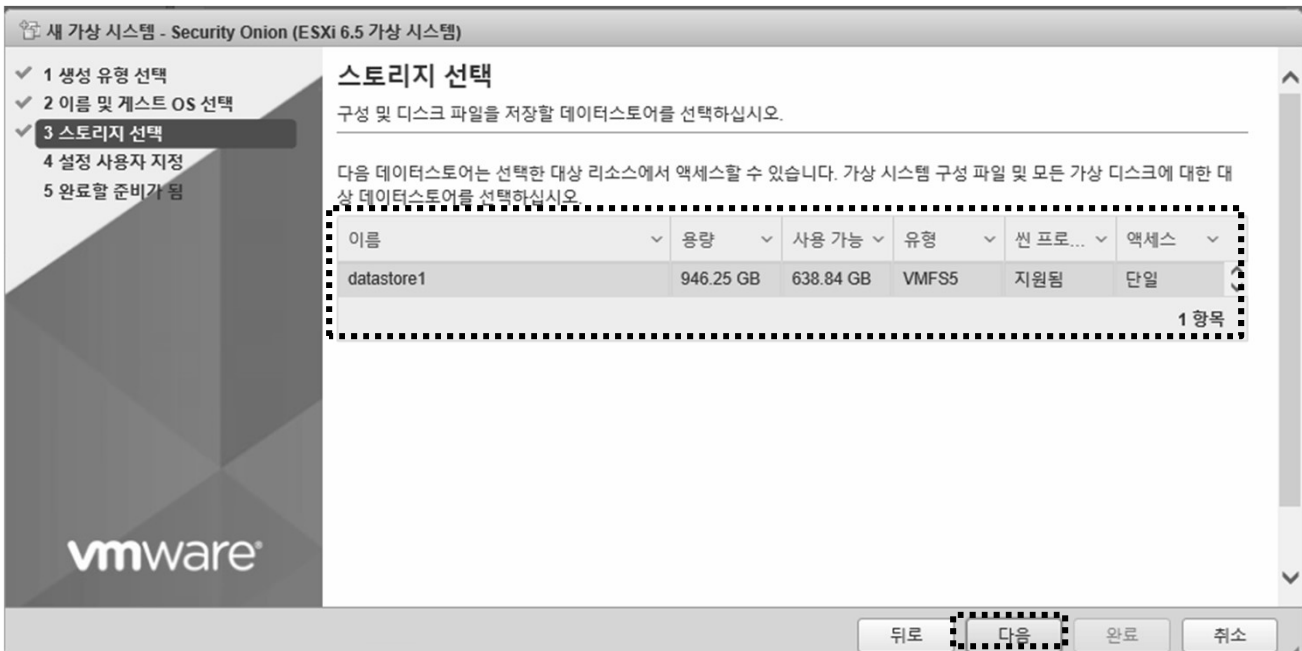
### 메모:

- ESXi 다운로드 주소: <https://my.vmware.com/en/web/vmware/evalcenter?p=free-esxi6>
- 디스크 이미지 굽기: Rufus 도구 사용 <https://rufus.akeo.ie/>
- Disk Imager <https://sourceforge.net/projects/win32diskimager/files/latest/download>
- USB 부팅 제가동은 전원 off/on (전원 케이블 포함)필요함
- [https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify\\_ISO.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md)

# X. 도구

## ❖ Security Onion @ Hypervisor

- ① 스토리지 선택
- ② 다음



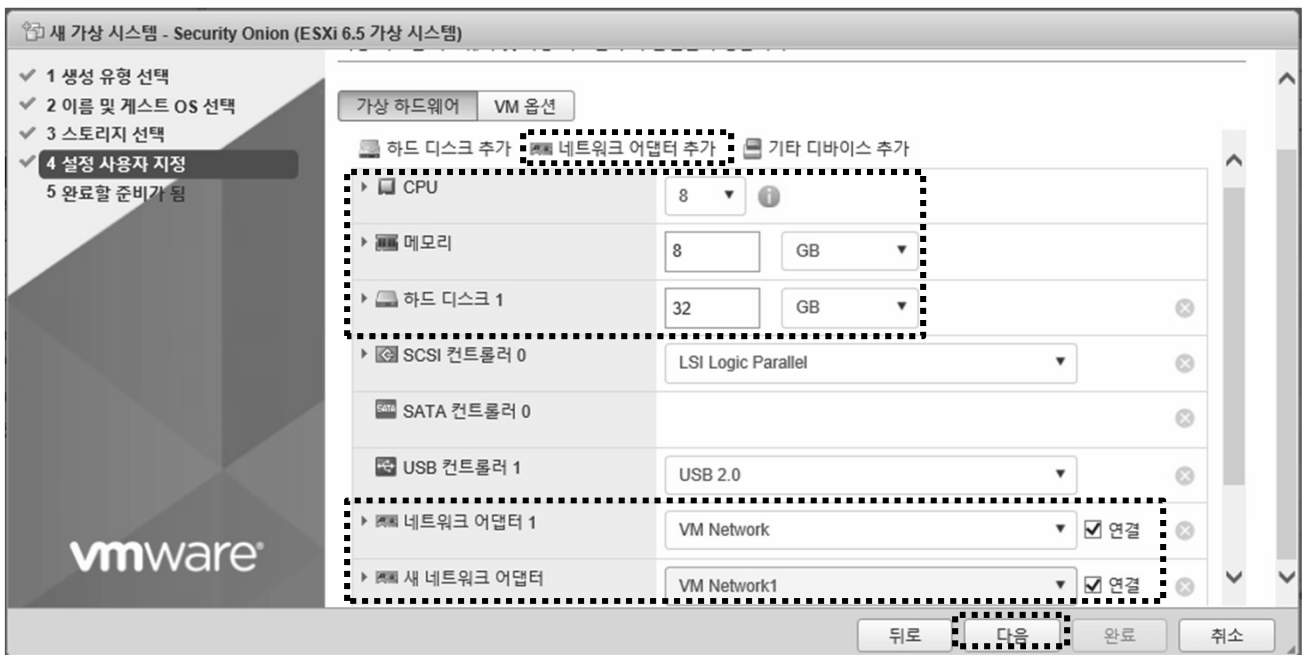
### 메모:

- ESXi 다운로드 주소: <https://my.vmware.com/en/web/vmware/evalcenter?p=free-esxi6>
- 디스크 이미지 굽기: Rufus 도구 사용 <https://rufus.akeo.ie/>
- Disk Imager <https://sourceforge.net/projects/win32diskimager/files/latest/download>
- USB 부팅 제가동은 전원 off/on (전원 케이블 포함)필요함
- [https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify\\_ISO.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md)

# X. 도구

## ❖ Security Onion @ Hypervisor

- ① 네트워크 어댑터 추가
- ② CPU/메모리/하드디스크
- ③ 네트워크 어댑터 선택
- ④ 다음



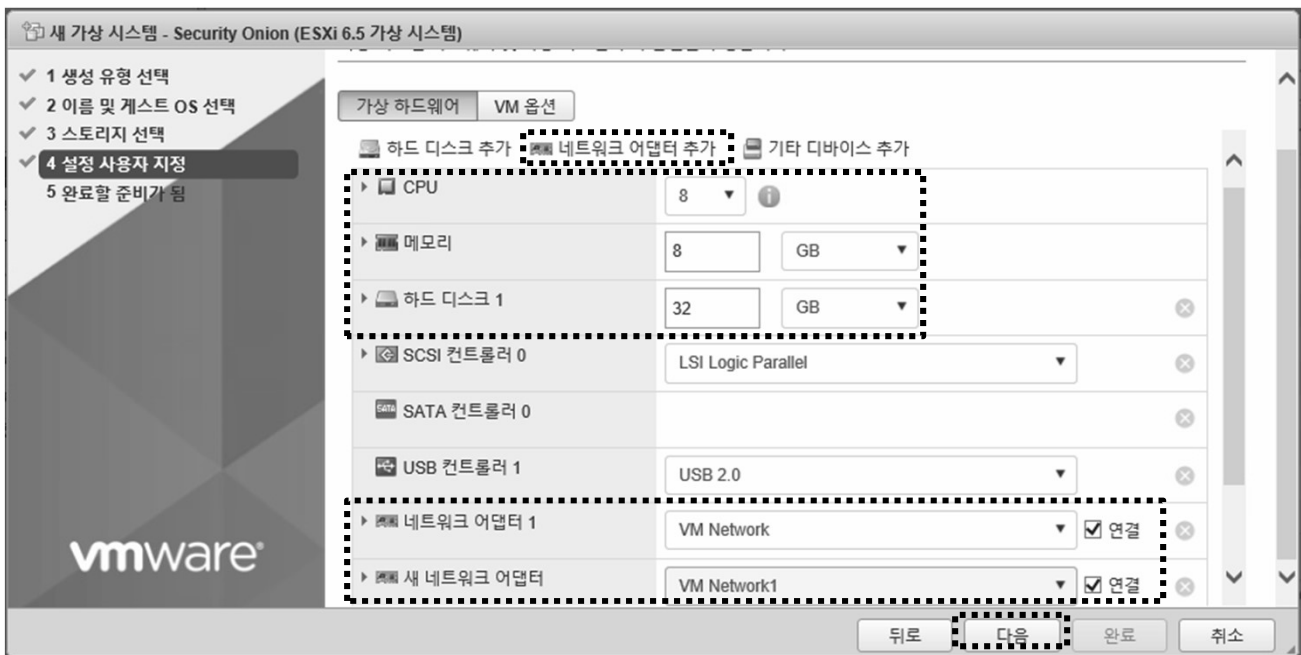
### 메모:

- 하드웨어 규격: <https://github.com/Security-Onion-Solutions/security-onion/wiki/Hardware>
- 최소 규격: RAM needed is 8GB
- [https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify\\_ISO.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md)

# X. 도구

## ❖ Security Onion @ Hypervisor

- ① 네트워크 어댑터 추가
- ② CPU 8 Core / 메모리 8GB / 하드디스크 썸(Thin)
- ③ 네트워크 어댑터 선택
- ④ 다음



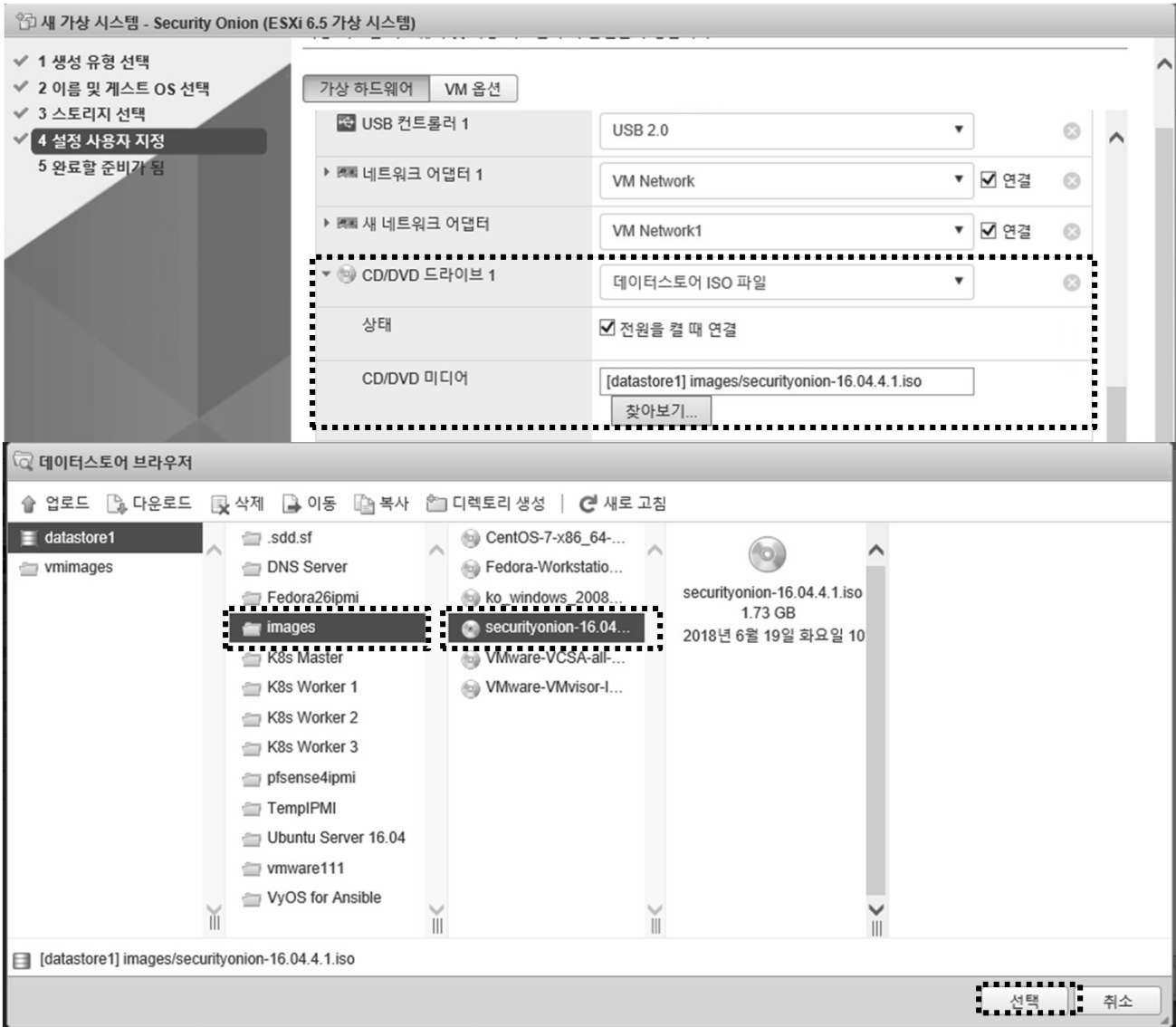
### 메모:

- 하드웨어 규격: <https://github.com/Security-Onion-Solutions/security-onion/wiki/Hardware>
- 최소 규격: RAM needed is 8GB
- [https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify\\_ISO.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md)
- 스위치의 무작위 모드 확인 (미러링 효과)

# X. 도구

## ❖ Security Onion @ Hypervisor

### ① 설치 이미지 선택

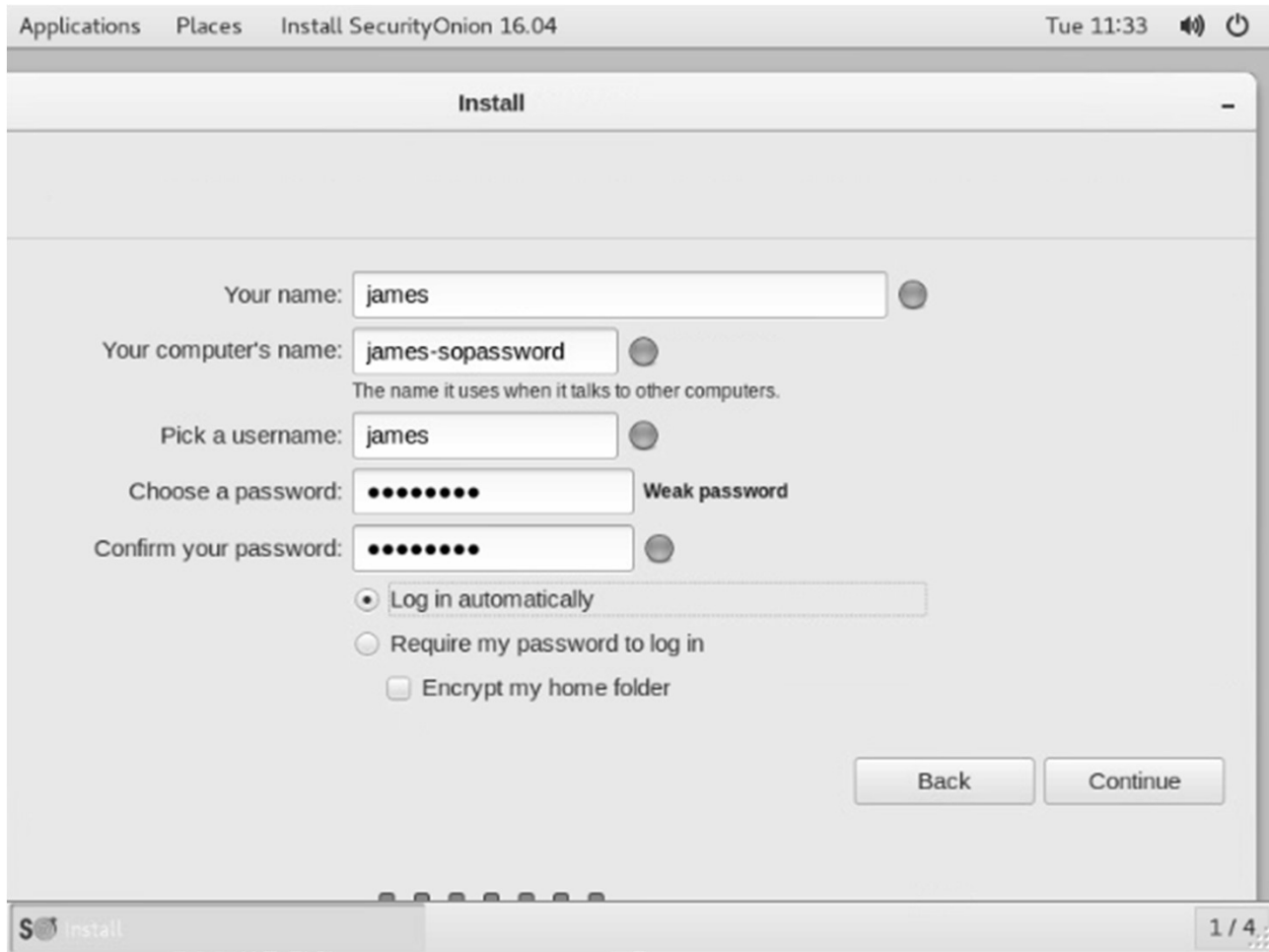


#### 메모:

- 하드웨어 규격: <https://github.com/Security-Onion-Solutions/security-onion/wiki/Hardware>
- 최소 규격: RAM needed is 8GB
- [https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify\\_ISO.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md)

## X. 도구

### ❖ Security Onion @ Hypervisor



#### 메모:

- 하드웨어 규격: <https://github.com/Security-Onion-Solutions/security-onion/wiki/Hardware>
- 최소 규격: RAM needed is 8GB
- [https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify\\_ISO.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md)

# X. 도구

## ❖ Security Onion @ Hypervisor

- ① 하이퍼바이저 확인
- ② 인터페이스 확인

The screenshot shows a Linux desktop environment with a terminal window and a 'Security Onion Setup' dialog box. The terminal displays the following network interface information:

```
ens160  Link encap:Ethernet  HWaddr 00:0c:29:34:a8:92
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:487 errors:0 dropped:0 overruns:0 frame:0
        TX packets:48 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:43852 (43.8 KB)  TX bytes:9184 (9.1 KB)

ens192  Link encap:Ethernet  HWaddr 00:0c:29:34:a8:9c
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:4 errors:0 dropped:0 overruns:0 frame:0
        TX packets:48 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:240 (240.0 B)  TX bytes:9184 (9.1 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:560 errors:0 dropped:0 overruns:0
        TX packets:560 errors:0 dropped:0 overruns:0
        collisions:0 txqueuelen:1000
```

The 'Security Onion Setup' dialog box asks: "Which network interface should be the management interface?" with radio buttons for 'ens160' and 'ens192'. A table in the foreground provides details for two network adapters:

네트워크 어댑터 1	
네트워크	VM Network1 (연결됨)
연결됨	예
MAC 주소	00:0c:29:34:a8:92
패스스루(Direct-path I/O)	예

네트워크 어댑터 2	
네트워크	VM Network (연결됨)
연결됨	예
MAC 주소	00:0c:29:34:a8:9c
패스스루(Direct-path I/O)	예

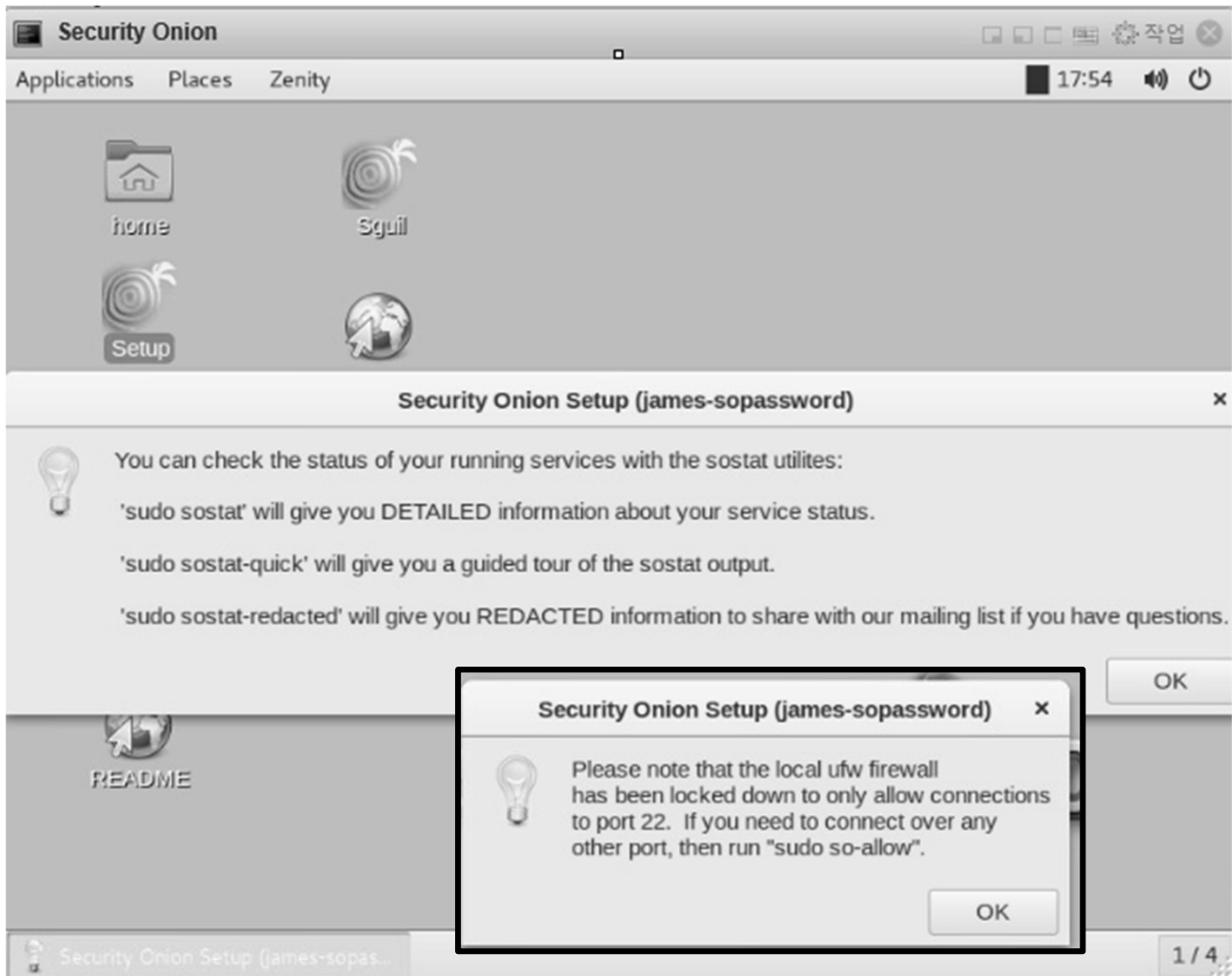
메모:



# X. 도구

## ❖ Security Onion @ Hypervisor

- ① 설치 이미지 선택
- ② Restart
- ③ Setup



### 메모:

- 하드웨어 규격: <https://github.com/Security-Onion-Solutions/security-onion/wiki/Hardware>
- 최소 규격: RAM needed is 8GB
- [https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify\\_ISO.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md)

# X. 도구

## ❖ Security Onion @ Hypervisor

- ① sudo apt get update @ Ubuntu Desktop
- ② Check event @ Security Onion

The screenshot shows a Kibana dashboard with three main panels: Files - MIME Type, Files - Source IP Address, and Files - Destination IP Address. Below the dashboard is a terminal window showing the execution of 'sudo apt-get update' and 'ifconfig' commands. A sidebar on the right contains various tool categories like SMTP, SNMP, Software, SSH, SSL, Syslog, Tunnels, Weird, and X.509. A 'NIDS - Alerts' panel shows an alert with the keyword 'Descending' and a count of 8.

MIME Type	Count	File IP Address	Count	IP Address	Count
application/pkix-cert	151	54.230.181.25	24	192.168.10.100	129
application/ocsp-request	17	192.168.10.100	17	192.168.0.100	49
application/ocsp-response	17	52.85.118.205	12	117.18.237.29	9
text/plain	3	54.230.181.131	12	172.217.26.14	6
text/html	1	117.18.237.29	9	54.192.183.96	2

```
James@james-virtual-machine:~$ sudo apt-get update
Hit:1 http://kr.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://kr.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://kr.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Fetched 247 kB in 2s (141 kB/s)
Reading package lists... Done
James@james-virtual-machine:~$ ifconfig
ens160: flags=4163<UP,BROADCAST,ST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.10.100 netmask 255.255.255.0 broadcast 192.168.10.255
    inet6 fe80::1cc1:dceb:5420:b4d4 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:d7:79:b6 txqueuelen 1000 (Ethernet)
    RX packets 106069 bytes 124995387 (124.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 49846 bytes 4690304 (4.6 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
```

메모:

## X. 도구

---

### ❖ X-RDP for Security Onion (선택)

# VM manager 사용하는 KVM/QEMU에서 마우스 인식 어려워 xrdp 설치

- ① **sudo ufw allow in 3389seconds**
- ② **sudo ufw allow ot 3389sword**
- ③ **sudo apt-get install xrdp**
- ④ **sudo apt-get install xfce4**
- ⑤ **sudo service xrdp restart**
  
- ⑥ **Remote Desktop 실행**

메모:

# X. 도구

## ❖ sudo so-allow for Remote Access

- ① sudo so-allow
- ② IP address for Remote Access

```
jslab@jslab-virtual-machine:~$ sudo so-allow
[sudo] password for jslab:
This program allows you to add a firewall rule to allow connections
from a new IP address.
```

What kind of device do you want to allow?

- [a] - Analyst - ports 22/tcp, 443/tcp, and 7734/tcp
- [b] - Logstash Beat - port 5044/tcp
- [c] - apt-cacher-ng client - port 3142/tcp
- [e] - Elasticsearch REST endpoint - port 9200
- [f] - Logstash forwarder - standard - port 6050/tcp
- [j] - Logstash forwarder - JSON - port 6051/tcp
- [l] - Syslog device - port 514
- [n] - Elasticsearch node-to-node communication - port 9300
- [o] - OSSEC agent - port 1514
- [s] - Security Onion sensor - 22/tcp, 4505/tcp, 4506/tcp, and

If you need to add any ports other than those listed above, you can do so using the standard 'ufw' utility.

For more information, please see the Firewall page on our Wik <https://github.com/Security-Onion-Solutions/security-onion/wiki/Firewall>

Please enter your selection (a - analyst, c - apt-cacher-ng c syslog, o - ossec, or s - Security Onion sensor, etc.):

a  
Please enter the IP address of the analyst you'd like to allow connect to port(s) 22, 443, 7734:

192.168.55.100

We're going to allow connections from 192.168.55.100 to port(s) 22, 443, 7734.

Here's the firewall rule we're about to add:  
sudo ufw allow proto tcp from 192.168.55.100 to any port 22, 4

We're also whitelisting 192.168.55.100 in /var/ossec/etc/ossec.conf to prevent OSSEC Active Response from blocking it. Keep in mind, the OSSEC server will be restarted once configuration is complete.

To continue and add this rule, press Enter. Otherwise, press Ctrl-c to exit.

Rule added  
Rule has been added.

Here is the entire firewall ruleset:

### UFW Rules

To	Action	From
22/tcp	ALLOW	Anywhere
22, 443, 7734/tcp	ALLOW	192.168.55.122
22, 443, 7734/tcp	ALLOW	192.168.10.100
22, 443, 7734/tcp	ALLOW	192.168.55.100
22/tcp (v6)	ALLOW	Anywhere (v6)

### Docker IPTables Rules

To	Action	From
----	--------	------

Added whitelist entry for 192.168.55.100 in /var/ossec/etc/ossec.conf.

Restarting OSSEC Server...  
jslab@jslab-virtual-machine:~\$

메모:

## X. 도구

---

### ❖ Squert for Security Onion (선택)

1. Squert is a web application that is used to query and view event data stored in a Sguil database (typically IDS alert data). Squert is a visual tool that attempts to provide additional context to events through the use of metadata, time series representations and weighted and logically grouped result sets.
2. Security Onion maintains its own fork of Squert
3. Squert authenticates against the Sguil user database, so you should be able to login to Squert using the same username/password you use to login to Sguil.
4. Data Type
  - NIDS alerts
  - HIDS alerts
  - Asset data from PRADS (if PRADS and pads\_agent are enabled)
  - HTTP logs from Bro (if http\_agent is enabled)

#### 메모:

- Squert: <http://www.squertproject.org/>
- Security Onion maintains its own fork of Squert: <https://blog.securityonion.net/2016/09/squert-development.html>

# X. 도구

## ❖ sudo docker info

```
jslab@jslab-virtual-machine:~$ sudo docker info
Containers: 7
  Running: 7
  Paused: 0
  Stopped: 0
Images: 7
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: 4.15.0-36-generic
Operating System: Ubuntu 16.04.5 LTS
OSType: linux
Architecture: x86_64
CPUs: 8
Total Memory: 11.73GiB
Name: jslab-virtual-machine
ID: UDLG:YGGR:VHYI:DNNS:3GER:63BY:KNR4:AIN4:EYA2:F6GY:VOXU:SYWZ
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:

- sudo docker version

## X. 도구

### ❖ sudo docker info

① sudo iptables -t nat -L -n

② sudo docker ps

```
jslab@jslab-virtual-machine:~$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9872b6d8bc21       bridge             bridge             local
80a0d461c98d       host              host              local
8400d338e2a3       none              null              local
8d1ed97b634e       so-elastic-net    bridge            local
jslab@jslab-virtual-machine:~$
```

```
jslab@jslab-virtual-machine:~$ sudo docker ps
CONTAINER ID        IMAGE                                     COMMAND              CREATED            STATUS
PORTS              NAMES
92fd22d9e34d       securityionsolutions/so-curator         "/bin/bash"        11 hours ago      Up 11
hours                                                     so-curator
93764999e697       securityionsolutions/so-elastalert      "/opt/start-elastale... 11 hours ago      Up 11
hours                                                     so-elastalert
419f8db86c1e       securityionsolutions/so-kibana         "/bin/sh -c /usr/loc... 11 hours ago      Up 11
hours                                                     so-kibana
35fde0562d89       securityionsolutions/so-logstash       "/usr/local/bin/dock... 11 hours ago      Up 11
hours 0.0.0.0:5044->5044/tcp, 0.0.0.0:6050-6053->6050-6053/tcp, 0.0.0.0:9600->9600/tcp so-logstash
a541ecde19ef       securityionsolutions/so-elasticsearch  "/bin/bash bin/es-do... 11 hours ago      Up 11
hours 127.0.0.1:9200->9200/tcp, 127.0.0.1:9300->9300/tcp so-elasticsearch
c4fd232d54dc       securityionsolutions/so-domainstats    "/bin/sh -c '/usr/bi... 11 hours ago      Up 11
hours 20000/tcp                                               so-domainstats
27e1571a4038       securityionsolutions/so-freqserver     "/bin/sh -c '/usr/bi... 11 hours ago      Up 11
hours 10004/tcp                                               so-freqserver
jslab@jslab-virtual-machine:~$
```

메모:

## X. 도구

### ❖ sudo iptables -t nat -L -n

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

Chain INPUT (policy ACCEPT)
target     prot opt source                destination

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

Chain POSTROUTING (policy ACCEPT)
target     prot opt source                destination            ADDRTYPE match dst-type LOCAL
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  --  172.17.0.4             172.17.0.4            tcp dpt:9300
MASQUERADE tcp  --  172.17.0.4             172.17.0.4            tcp dpt:9200
MASQUERADE tcp  --  172.17.0.5             172.17.0.5            tcp dpt:9600
MASQUERADE tcp  --  172.17.0.5             172.17.0.5            tcp dpt:6053
MASQUERADE tcp  --  172.17.0.5             172.17.0.5            tcp dpt:6052
MASQUERADE tcp  --  172.17.0.5             172.17.0.5            tcp dpt:6051
MASQUERADE tcp  --  172.17.0.5             172.17.0.5            tcp dpt:6050
MASQUERADE tcp  --  172.17.0.5             172.17.0.5            tcp dpt:5044
MASQUERADE tcp  --  172.17.0.6             172.17.0.6            tcp dpt:5601

Chain DOCKER (2 references)
target     prot opt source                destination            ADDRTYPE match dst-type LOCAL
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              127.0.0.1              tcp dpt:9300 to:172.17.0.4:9300
DNAT       tcp  --  0.0.0.0/0              127.0.0.1              tcp dpt:9200 to:172.17.0.4:9200
DNAT       tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:9600 to:172.17.0.5:9600
DNAT       tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:6053 to:172.17.0.5:6053
DNAT       tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:6052 to:172.17.0.5:6052
DNAT       tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:6051 to:172.17.0.5:6051
DNAT       tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:6050 to:172.17.0.5:6050
DNAT       tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:5044 to:172.17.0.5:5044
DNAT       tcp  --  0.0.0.0/0              127.0.0.1              tcp dpt:5601 to:172.17.0.6:5601
jslab@jslab-virtual-machine:~$
```

메모:



# X. 도구

## ❖ sudo iptables -L -n

```
jslab@jslab-virtual-machine:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
ufw-before-logging-input all -- 0.0.0.0/0 0.0.0.0/0
ufw-before-input all -- 0.0.0.0/0 0.0.0.0/0
ufw-after-input all -- 0.0.0.0/0 0.0.0.0/0
ufw-after-logging-input all -- 0.0.0.0/0 0.0.0.0/0
ufw-logging-input all -- 0.0.0.0/0 0.0.0.0/0
ufw-track-input all -- 0.0.0.0/0 0.0.0.0/0

Chain FORWARD (policy DROP)
target prot opt source destination
DOCKER-USER all -- 0.0.0.0/0 0.0.0.0/0
DOCKER-ISOLATION-STAGE-1 all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 ctstate RELATED,ESTABLISHED
DOCKER all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 ctstate RELATED,ESTABLISHED
DOCKER all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0
ufw-before-logging-forward all -- 0.0.0.0/0 0.0.0.0/0
ufw-before-forward all -- 0.0.0.0/0 0.0.0.0/0
ufw-after-forward all -- 0.0.0.0/0 0.0.0.0/0
ufw-after-logging-forward all -- 0.0.0.0/0 0.0.0.0/0
ufw-logging-forward all -- 0.0.0.0/0 0.0.0.0/0
ufw-reject-forward all -- 0.0.0.0/0 0.0.0.0/0
ufw-track-forward all -- 0.0.0.0/0 0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
ufw-before-logging-output all -- 0.0.0.0/0 0.0.0.0/0
ufw-before-output all -- 0.0.0.0/0 0.0.0.0/0
ufw-after-output all -- 0.0.0.0/0 0.0.0.0/0
ufw-after-logging-output all -- 0.0.0.0/0 0.0.0.0/0
ufw-logging-output all -- 0.0.0.0/0 0.0.0.0/0
ufw-reject-output all -- 0.0.0.0/0 0.0.0.0/0
ufw-track-output all -- 0.0.0.0/0 0.0.0.0/0

Chain DOCKER (2 references)
target prot opt source destination
ACCEPT top -- 0.0.0.0/0 172.17.0.4 top dpt:9300
ACCEPT top -- 0.0.0.0/0 172.17.0.4 top dpt:9300
ACCEPT top -- 0.0.0.0/0 172.17.0.5 top dpt:9600
ACCEPT top -- 0.0.0.0/0 172.17.0.5 top dpt:6053
ACCEPT top -- 0.0.0.0/0 172.17.0.5 top dpt:6052
ACCEPT top -- 0.0.0.0/0 172.17.0.5 top dpt:6051
ACCEPT top -- 0.0.0.0/0 172.17.0.5 top dpt:6050
ACCEPT top -- 0.0.0.0/0 172.17.0.5 top dpt:5044
ACCEPT top -- 0.0.0.0/0 172.17.0.5 top dpt:5041

Chain DOCKER-ISOLATION-STAGE-1 (1 reference)
target prot opt source destination
DOCKER-ISOLATION-STAGE-2 all -- 0.0.0.0/0 0.0.0.0/0
RETURN all -- 0.0.0.0/0 0.0.0.0/0

Chain DOCKER-ISOLATION-STAGE-2 (2 references)
target prot opt source destination
DROP all -- 0.0.0.0/0 0.0.0.0/0
DROP all -- 0.0.0.0/0 0.0.0.0/0
RETURN all -- 0.0.0.0/0 0.0.0.0/0

Chain DOCKER-USER (1 reference)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
DROP all -- 0.0.0.0/0 0.0.0.0/0
RETURN all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-after-forward (1 reference)
target prot opt source destination

Chain ufw-after-input (1 reference)
target prot opt source destination
ufw-skip-to-policy-input udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:137
ufw-skip-to-policy-input udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:138
ufw-skip-to-policy-input tcp -- 0.0.0.0/0 0.0.0.0/0 top dpt:139
ufw-skip-to-policy-input tcp -- 0.0.0.0/0 0.0.0.0/0 top dpt:445
ufw-skip-to-policy-input udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:67
ufw-skip-to-policy-input udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:68
ufw-skip-to-policy-input all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type BROADCAST

Chain ufw-after-logging-forward (1 reference)
target prot opt source destination
LOG all -- 0.0.0.0/0 0.0.0.0/0 limit: avg 3/min burst 10 LOG flags 0 level 4 prefix "[UFW BLOCK]"

Chain ufw-after-logging-input (1 reference)
target prot opt source destination
LOG all -- 0.0.0.0/0 0.0.0.0/0 limit: avg 3/min burst 10 LOG flags 0 level 4 prefix "[UFW BLOCK]"

Chain ufw-after-logging-output (1 reference)
target prot opt source destination

Chain ufw-after-output (1 reference)
target prot opt source destination

Chain ufw-before-forward (1 reference)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 ctstate RELATED,ESTABLISHED
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 3
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 4
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 11
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 12
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 8
ufw-user-forward all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-before-input (1 reference)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 ctstate RELATED,ESTABLISHED
```

```
ufw-logging-deny all -- 0.0.0.0/0 0.0.0.0/0 ctstate INVALID
DROP all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 3
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 4
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 11
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 12
ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 8
ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0 udp spt:67 dpt:68
ufw-not-local all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT udp -- 0.0.0.0/0 224.0.0.251 udp dpt:5353
ACCEPT udp -- 0.0.0.0/0 239.255.255.250 udp dpt:1900
ufw-user-input all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-before-logging-forward (1 reference)
target prot opt source destination

Chain ufw-before-logging-input (1 reference)
target prot opt source destination

Chain ufw-before-logging-output (1 reference)
target prot opt source destination

Chain ufw-before-output (1 reference)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 ctstate RELATED,ESTABLISHED
ufw-user-output all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-logging-allow (0 references)
target prot opt source destination
LOG all -- 0.0.0.0/0 0.0.0.0/0 limit: avg 3/min burst 10 LOG flags 0 level 4 prefix "[UFW ALLOW]"

Chain ufw-logging-deny (2 references)
target prot opt source destination
RETURN all -- 0.0.0.0/0 0.0.0.0/0 ctstate INVALID limit: avg 3/min burst 10
LOG all -- 0.0.0.0/0 0.0.0.0/0 limit: avg 3/min burst 10 LOG flags 0 level 4 prefix "[UFW BLOCK]"

Chain ufw-not-local (1 reference)
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 ADDRTYPE match dst-type LOCAL
RETURN all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type BROADCAST
ufw-logging-deny all -- 0.0.0.0/0 0.0.0.0/0 limit: avg 3/min burst 10
DROP all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-reject-forward (1 reference)
target prot opt source destination

Chain ufw-reject-input (1 reference)
target prot opt source destination

Chain ufw-reject-output (1 reference)
target prot opt source destination

Chain ufw-skip-to-policy-forward (0 references)
target prot opt source destination
DROP all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-skip-to-policy-input (7 references)
target prot opt source destination
DROP all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-skip-to-policy-output (0 references)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-track-forward (1 reference)
target prot opt source destination

Chain ufw-track-input (1 reference)
target prot opt source destination

Chain ufw-track-output (1 reference)
target prot opt source destination
ACCEPT top -- 0.0.0.0/0 0.0.0.0/0 ctstate NEW
ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0 ctstate NEW

Chain ufw-user-forward (1 reference)
target prot opt source destination

Chain ufw-user-input (3 references)
target prot opt source destination
ACCEPT top -- 0.0.0.0/0 0.0.0.0/0 top dpt:22
ACCEPT top -- 192.168.55.122 0.0.0.0/0 multiport dports 22,443,7734
ACCEPT top -- 192.168.10.100 0.0.0.0/0 multiport dports 22,443,7734
ACCEPT top -- 192.168.55.100 0.0.0.0/0 multiport dports 22,443,7734

Chain ufw-user-limit (0 references)
target prot opt source destination
LOG all -- 0.0.0.0/0 0.0.0.0/0 limit: avg 3/min burst 5 LOG flags 0 level 4 prefix "[UFW LIMIT BLOCK]"
REJECT all -- 0.0.0.0/0 0.0.0.0/0 reject-with icmp-port-unreachable

Chain ufw-user-limit-accept (0 references)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0

Chain ufw-user-logging-forward (0 references)
target prot opt source destination

Chain ufw-user-logging-input (0 references)
target prot opt source destination

Chain ufw-user-logging-output (0 references)
target prot opt source destination

Chain ufw-user-output (1 reference)
target prot opt source destination
jslab@jslab-virtual-machine:~$
```

메모:

# X. 도구

## ❖ ip route

- ① ip route
- ② brctl show
- ③ Check 'sudo docker network ls' # 도커의 리눅스 브릿지 사용

```
jslab@jslab-virtual-machine:~$ ip route
default via 192.168.55.1 dev ens224 onlink
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
172.18.0.0/16 dev br-8d1ed97b634e proto kernel scope link src 172.18.0.1
192.168.55.0/24 dev ens224 proto kernel scope link src 192.168.55.43
jslab@jslab-virtual-machine:~$
```

```
jslab@jslab-virtual-machine:~$ brctl show
bridge name      bridge id        STP enabled      interfaces
br-8d1ed97b634e  8000.02429b7f90e0  no               veth0a8d905
                                                         veth2fc6972
                                                         veth3b98e4f
                                                         veth5284a6f
                                                         veth783c90b
                                                         veth7a5200b
                                                         vethcdb21af
docker0          8000.0242d38891bc  no               veth4021b3b
                                                         veth591b8ce
                                                         veth7ef17b0
                                                         veth8d500af
                                                         vetha1d41ca
                                                         vethbc57b2b
                                                         vethebd422
```

```
jslab@jslab-virtual-machine:~$ sudo docker network ls
NETWORK ID      NAME          DRIVER        SCOPE
9872b6d8bc21   bridge       bridge        local
80a0d461c98d   host         host          local
8400d338e2a3   none         null          local
8d1ed97b634e   so-elastic-net  bridge        local
jslab@jslab-virtual-machine:~$
```

메모:

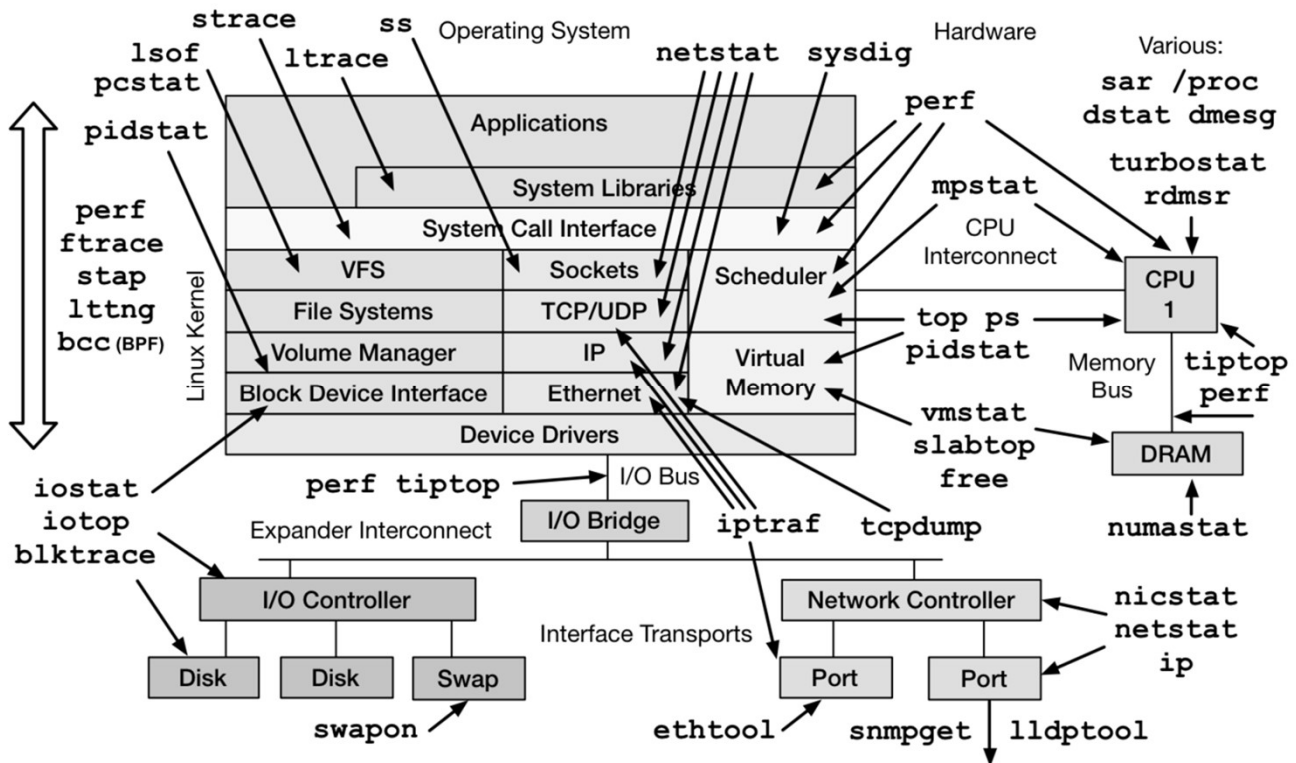
# X. 도구

## ❖ Linux Perf Tools

① `sudo apt-get install nmh`

② **Fedora:** 다음쪽 참조

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



### 메모:

- 컨테이너 고려사항
  - ✓ Netdata 등의 트래픽 변동 확인 `http://127.0.0.1:19999/`
  - ✓ Host PID는 컨테이너 ID와 연동하지 않음
  - ✓ 커널에 컨테이너 ID 표시 없음
  - ✓ 리눅스 도구에 컨테이너를 위한 설명 없음 (netdata는 docker0 트래픽 통계 확인 가능)

## X. 도구

### ❖ sudo docker network ls & brctl show

1. sudo docker network ls
2. 'brctl show' & 'virsh net-list --all'

```
jslab@jslab-virtual-machine:~$ ip route
default via 192.168.55.1 dev ens224 onlink
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
172.18.0.0/16 dev br-8d1ed97b634e proto kernel scope link src 172.18.0.1
192.168.55.0/24 dev ens224 proto kernel scope link src 192.168.55.43
jslab@jslab-virtual-machine:~$
```

```
jslab@jslab-virtual-machine:~$ brctl show
bridge name      bridge id                STP enabled   interfaces
br-8d1ed97b634e  8000.02429b7f90e0        no            veth0a8d905
                                                         veth2fc6972
                                                         veth3b98e4f
                                                         veth5284a6f
                                                         veth783c90b
                                                         veth7a5200b
                                                         vethcdb21af
                                                         veth4021b3b
                                                         veth591b8ce
                                                         veth7ef17b0
                                                         veth8d500af
                                                         vetha1d41ca
                                                         vethbc57b2b
                                                         vethebda422
docker0          8000.0242d38891bc        no            vethc0a8d905
                                                         veth2fc6972
                                                         veth3b98e4f
                                                         veth5284a6f
                                                         veth783c90b
                                                         veth7a5200b
                                                         vethcdb21af
                                                         veth4021b3b
                                                         veth591b8ce
                                                         veth7ef17b0
                                                         veth8d500af
                                                         vetha1d41ca
                                                         vethbc57b2b
                                                         vethebda422
```

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

#### 메모:

- virsh is a command line interface tool for managing guests and the hypervisor

## X. 도구

---

### ❖ brctl showmacs docker0

#### ① brctl showmacs docker0

```
jslab@jslab-virtual-machine:~$ brctl showmacs docker0
port no mac addr          is local?    ageing timer
 4    02:42:ac:11:00:05      no           0.90
 5    02:42:ac:11:00:06      no           16.37
 6    06:e9:55:0d:c7:4a      yes          0.00
 6    06:e9:55:0d:c7:4a      yes          0.00
 1    42:2c:60:88:9a:65      yes          0.00
 1    42:2c:60:88:9a:65      yes          0.00
 3    4e:b4:78:52:47:4b      yes          0.00
 3    4e:b4:78:52:47:4b      yes          0.00
 2    7a:02:82:10:c9:70      yes          0.00
 2    7a:02:82:10:c9:70      yes          0.00
 4    82:f5:84:ad:6b:f5      yes          0.00
 4    82:f5:84:ad:6b:f5      yes          0.00
 7    a6:f3:3a:e2:05:6f      yes          0.00
 7    a6:f3:3a:e2:05:6f      yes          0.00
 5    e2:d3:a5:2f:33:52      yes          0.00
 5    e2:d3:a5:2f:33:52      yes          0.00
jslab@jslab-virtual-machine:~$
```

#### 메모:

- Network : bridge(bridge), host(host), none(null), so-elastic-net(bridge)

# X. 도구

## ❖ sudo docker network inspect bridge

```
jslab@jslab-virtual-machine:~$ sudo docker network inspect bridge
[sudo] password for jslab:
[
  {
    "Name": "bridge",
    "Id": "9372b6d8bc21ff54e2d204efd323ba6bef8791f497598c15d115bbe1cb477826",
    "Created": "2018-11-18T19:35:19.141183196Z",
    "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": {
      "27e1571a40380bbc40c2db9161e65b666cac2e48b1c01020799a563c7c767163": {
        "Name": "so-freqserver",
        "EndpointID": "93d76dc8ad119431f94e1c179586a6d1495e7db487a1fd3bf7e4036241f15555",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      },
      "35fde0562d899b0d6cd0e7ac820dfd73a60bde230c170210f35cea7c255b5131": {
        "Name": "so-logstash",
        "EndpointID": "c7678ac47cbb18705e788c3e0652ef7749451a01e7db96c21d66484f1217bea3",
        "MacAddress": "02:42:ac:11:00:05",
        "IPv4Address": "172.17.0.5/16",
        "IPv6Address": ""
      },
      "419f8db86c1ed5fccc28d0a41fb4f059c54dcbf038c3d7056e59956bfa3c0a3c": {
        "Name": "so-kibana",
        "EndpointID": "79bc394b8a8538a1f409e0f4e59ab442c26ab77b5173ce00d172cbf0c98d8f35",
        "MacAddress": "02:42:ac:11:00:06",
        "IPv4Address": "172.17.0.6/16",
        "IPv6Address": ""
      },
      "92fd22d9e34ddc90af681aae94dd9e6d6ad208b4df87eb3e549d83064a5371d": {
        "Name": "so-curator",
        "EndpointID": "6b8bc81f87fc1f44a7f5e8e0a87309e7fa5a16e0295f78f73935801ab17454a",
        "MacAddress": "02:42:ac:11:00:08",
        "IPv4Address": "172.17.0.8/16",
        "IPv6Address": ""
      },
      "93764999e6975359edeb3a03ca8908b5358b640cbbc21b5793b57463bdfbf7c0": {
        "Name": "so-elastalert",
        "EndpointID": "8469cb9216c93abc9f4cc790511bce46870eda81785a2250f2bd06c3c9d67f7a",
        "MacAddress": "02:42:ac:11:00:07",
        "IPv4Address": "172.17.0.7/16",
        "IPv6Address": ""
      },
      "a541ecde19ef87b73595305e0958d7208458d745eb66ac5ee66d07d084d8e0b6": {
        "Name": "so-elasticsearch",
        "EndpointID": "e0dc9a32ad25a1f66e73fc5bfb608e7d634ded6a5aad60ae135e99ccd505a58d",
        "MacAddress": "02:42:ac:11:00:04",
        "IPv4Address": "172.17.0.4/16",
        "IPv6Address": ""
      },
      "c4fd232d54dc571b105c590742dda1f32635e65bbc85c065ddf536d104dfd6": {
        "Name": "so-domainstats",
        "EndpointID": "cd0e6530e299debd3ee1a15860438298d942c1200634e2d16b1ef4f",
        "MacAddress": "02:42:ac:11:00:03",
        "IPv4Address": "172.17.0.3/16",
        "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": {}
  }
]
```

james@jslab.kr

### 메모:

- Containers @ Bridge : so-curator (172.17.0.8/16), so-elastalert (172.17.0.7/16), so-kibana (172.17.0.6/16), so-logstash (172.17.0.5/16), so-elasticsearch (172.17.0.4/16), so-domainstats (172.17.0.3/16), so-freqserver (172.17.0.2/16)

# X. 도구

## ❖ sudo docker network inspect so-elastic-net

```
jslab@jslab-virtual-machine:~$ sudo docker network inspect so-elastic-net
[
  {
    "Name": "so-elastic-net",
    "Id": "8d1ed97b634e480e725e4033b3fde3c2e382765b5f1274daf7533281c0070a",
    "Created": "2018-11-18T19:38:51.794605818Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "27e1571a40380bbc40c2db9161e65b666cac2e48b1c01020799a563c7c767163": {
        "Name": "so-freqserver",
        "EndpointID": "5443b5e27f18c283aac673a5158c680f1cd16ab7473b96d43ac2cf643d78f488",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
      },
      "35fde0562d899b0d6cd0e7ac820dfd73a60bde230c170210f35cea7c255b5131": {
        "Name": "so-logstash",
        "EndpointID": "f0dfcc32661fc7fb04e8f5a08f24cbe4d1545f6b82b56375990a4367dbd417a3",
        "MacAddress": "02:42:ac:12:00:05",
        "IPv4Address": "172.18.0.5/16",
        "IPv6Address": ""
      },
      "419f8db86c1ed5fccc28d0a41fb4f059c54dcbf038c3d7056e59956bfa3c0a3c": {
        "Name": "so-kibana",
        "EndpointID": "1af0dd03cc5aadd4ea15f55758e752b1ae45b2fd9f8f647c43c6447259333bab",
        "MacAddress": "02:42:ac:12:00:06",
        "IPv4Address": "172.18.0.6/16",
        "IPv6Address": ""
      },
      "92fd22d9e34ddc90af681aae94dd9e6d6ad208b4df87eb3e549d83064a5371d": {
        "Name": "so-curator",
        "EndpointID": "ed12e89b81146df9beeb46156d778a2554fa8f0bedc43be7956f56c656e009b",
        "MacAddress": "02:42:ac:12:00:08",
        "IPv4Address": "172.18.0.8/16",
        "IPv6Address": ""
      },
      "93764999e6975359edeb3a03ca8908b5358b640cbbc21b5793b57463bdfbf7c0": {
        "Name": "so-elastalert",
        "EndpointID": "3d308fafa699188adce990fdd9b4d13aa30bc38595e19b93b3fa286962b7387a",
        "MacAddress": "02:42:ac:12:00:07",
        "IPv4Address": "172.18.0.7/16",
        "IPv6Address": ""
      },
      "a541ecde19ef87b73595305e0958d7208458d745eb66ac5ee66d07d084d8e0b6": {
        "Name": "so-elasticsearch",
        "EndpointID": "ed4125e7734b9c3f46adc328fa8edc61c72f18ec644b605708590893c25512f6",
        "MacAddress": "02:42:ac:12:00:04",
        "IPv4Address": "172.18.0.4/16",
        "IPv6Address": ""
      },
      "c4fd232d54dc571b105c5909742dda1f32635e65bbc85c065ddef536d104dfd6": {
        "Name": "so-domainstats",
        "EndpointID": "4384c3eeee12b13bdf61f9b88b92bf1a58e5ba8988f4076333a87080a2ce6584",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
      }
    }
  }
]
jslab@jslab-virtual-machine:~$
```

james@jslab.kr

### 메모:

- Containers @ so-elastic-net (bridge) : so-curator (172.18.0.8/16), so-elastalert (172.18.0.7/16), so-kibana (172.18.0.6/16), so-logstash (172.17.0.5/16), so-elasticsearch (172.18.0.4/16), so-domainstats (172.18.0.3/16), so-freqserver (172.18.0.2/16)





# X. 도구

---

## ❖ sudo docker image ls

① sudo docker image ls

② sudo docker image inspect c6

```
jslab@jslab-virtual-machine:~$ sudo docker image ls
REPOSITORY                                TAG                IMAGE ID           CREATED           SIZE
securityonionsolutions/so-freqserver      latest            7430335b16aa      2 months ago    312MB
securityonionsolutions/so-domainstats    latest            0497f0bbe842      2 months ago    400MB
securityonionsolutions/so-elastalert     latest            0ee1d4814674      2 months ago    418MB
securityonionsolutions/so-curator        latest            c1e5b6c06aad      2 months ago    324MB
securityonionsolutions/so-kibana         latest            ce42f28e58ab      2 months ago    800MB
securityonionsolutions/so-logstash       latest            c6f488b28175      2 months ago    708MB
securityonionsolutions/so-elasticsearch  latest            862bec843f98      2 months ago    432MB
```

### 메모:

- Images: so-curator, so-elastalert, so-kibana, so-logstash, so-elasticsearch, so-domainstats, so-freqserver ( @ /securityonionsolutions/)

# X. 도구

## ❖ sudo docker image inspect d9

```
james@jkoron:~$ sudo docker inspect d9
[{"id": "sha256:d9c0d19555e5a5eaf07aeeefb0367ab27d6c34cb5553df69a047ea",
  "rootfs": {
    "type": "overlay2",
    "diff_ids": [
      "sha256:09c4472f224ba8f02bae0d1fa651e94456c289e908162620b45d5"
    ]
  },
  "parent": "",
  "comment": "",
  "created": "2018-03-21T11:27:07.34218792Z",
  "container": "24006fe709ed3179bc089c028b074743ab2b4196612b33426961248ee75",
  "container_config": {
    "hostname": "24006fe709ed",
    "domainname": "",
    "user": "freqserver",
    "attach_stdin": false,
    "attach_stdout": false,
    "attach_stderr": false,
    "exposed_ports": {
      "10004/tcp": []
    },
    "tty": false,
    "open_stdin": false,
    "stdin_once": false,
    "env": [
      "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
    ]
  },
  "os": {
    "arch": "amd64",
    "os": "linux"
  },
  "cmd": [
    "/bin/sh",
    "-c",
    "#(nop)",
    "CMD ["/bin/sh" "$@" -c "$@" /usr/bin/python /opt/freq_server/freq_server.py -ip 0.0.0.0 10004 /opt/freq_server/freq_table.freq]"
  ],
  "arg_scope": {
    "image": "sha256:60a541b1100f94e71439ad0247e7049e2a4b43707bcf50a847072ab2bd957a",
    "volumes": null,
    "networking": null,
    "entrypoint": null,
    "debug": null,
    "labels": {
      "build-date": "20180302",
      "license": "GPLv2",
      "maintainer": "Security Onion Solutions, LLC",
      "name": "Security Base Image",
      "vendor": "CentOS"
    }
  },
  "stop_signal": "SIGTERM",
  "docker_version": "17.12.1-ce",
  "author": "",
  "config": {
    "hostname": "",
    "domainname": "",
    "user": "freqserver",
    "attach_stdin": false,
    "attach_stdout": false,
    "attach_stderr": false,
    "exposed_ports": {
      "10004/tcp": []
    },
    "tty": false,
    "open_stdin": false,
    "stdin_once": false,
    "env": [
      "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
    ]
  },
  "os": {
    "arch": "amd64",
    "os": "linux"
  },
  "cmd": [
    "/usr/bin/python /opt/freq_server/freq_server.py -ip 0.0.0.0 10004 /opt/freq_server/freq_table.freq"
  ],
  "arg_scope": {
    "image": "sha256:60a541b1100f94e71439ad0247e7049e2a4b43707bcf50a847072ab2bd957a",
    "volumes": null,
    "networking": null,
    "entrypoint": null,
    "debug": null,
    "labels": {
      "build-date": "20180302",
      "license": "GPLv2",
      "maintainer": "Security Onion Solutions, LLC",
      "name": "Security Base Image",
      "vendor": "CentOS"
    }
  },
  "stop_signal": "SIGTERM",
  "architecture": "amd64",
  "os": "linux",
  "size": 364767066,
  "virtual_size": 364767066,
  "graph_driver": {
    "data": null,
    "name": "aufs"
  },
  "rootfs": {
    "type": "layers",
    "layers": [
      "sha256:020995543b7956d23a5ef4220c959a950a01d8644044c0073a793841884",
      "sha256:09c4472f224ba8f02bae0d1fa651e94456c289e908162620b45d5",
      "sha256:c21691efae209419907aa3a0010d816d8a097150d7b3035819238f19e1a",
      "sha256:52002c7fd1e11509398a79d999e4c3a7b0c0d54c921607187fa08b0",
      "sha256:19623d8e97545780ca1f4c5c20a105d802d14e4101f18a972e0a0ba",
      "sha256:11603c10f4a0000a72300a1621e70470a116604d800845210a09",
      "sha256:c01826e1817e70cf27d4f1ead0c0c3551fa81c09175d8bd051122001a"
    ]
  },
  "metadata": {
    "last_commit": "2018-04-16T14:21:01.09380059-04:00"
  }
}]
```

```

"ContainerConfig": {
  "Hostname": "24006fe709ed",
  "Domainname": "",
  "User": "freqserver",
  "AttachStdin": false,
  "AttachStdout": false,
  "AttachStderr": false,
  "ExposedPorts": {
    "10004/tcp": []
  },
  "Tty": false,
  "OpenStdin": false,
  "StdinOnce": false,
  "Env": [
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  ],
  "Cmd": [
    "/bin/sh",
    "-c",
    "#(nop)",
    "CMD ["/bin/sh" "$@" -c "$@" /usr/bin/python /opt/freq_server/freq_server.py -ip 0.0.0.0 10004 /opt/freq_server/freq_table.freq]"
  ],
  "ArgScope": {
    "image": "sha256:60a541b1100f94e71439ad0247e7049e2a4b43707bcf50a847072ab2bd957a",
    "volumes": null,
    "networking": null,
    "entrypoint": null,
    "debug": null,
    "labels": {
      "build-date": "20180302",
      "license": "GPLv2",
      "maintainer": "Security Onion Solutions, LLC",
      "name": "Security Base Image",
      "vendor": "CentOS"
    }
  },
  "stop_signal": "SIGTERM",
  "architecture": "amd64",
  "os": "linux",
  "size": 364767066,
  "virtual_size": 364767066,
  "graph_driver": {
    "data": null,
    "name": "aufs"
  },
  "rootfs": {
    "type": "layers",
    "layers": [
      "sha256:020995543b7956d23a5ef4220c959a950a01d8644044c0073a793841884",
      "sha256:09c4472f224ba8f02bae0d1fa651e94456c289e908162620b45d5",
      "sha256:c21691efae209419907aa3a0010d816d8a097150d7b3035819238f19e1a",
      "sha256:52002c7fd1e11509398a79d999e4c3a7b0c0d54c921607187fa08b0",
      "sha256:19623d8e97545780ca1f4c5c20a105d802d14e4101f18a972e0a0ba",
      "sha256:11603c10f4a0000a72300a1621e70470a116604d800845210a09",
      "sha256:c01826e1817e70cf27d4f1ead0c0c3551fa81c09175d8bd051122001a"
    ]
  },
  "metadata": {
    "last_commit": "2018-04-16T14:21:01.09380059-04:00"
  }
}

```

james@jslab.kr

### 메모:

- X-RDP for Security Onion

## X. 도구

---

### ❖ netdata / ntopng / sshd / Net-tools (Ubuntu 17.20)

#### 1. netdata

- `bash <(curl -Ss https://my-netdata.io/kickstart.sh)`
- `http://127.0.0.1:19999/`

#### 2. ntopng

- `sudo apt install ntopng`
- `sudo ntopng`
- `http://127.0.0.1:3000/ (admn/admin)`

#### 3. SSH server

- `sudo apt install openssh-server`

#### 4. Net tools for 'ifconfig'

- `sudo apt install net-tools`

#### 메모:

- 다운로드 주소:
  - ✓ [https://github.com/Security-Onion-Solutions/security-onion/blob/master/old/Verify\\_ISO\\_14.04.5.2.md](https://github.com/Security-Onion-Solutions/security-onion/blob/master/old/Verify_ISO_14.04.5.2.md)
  - ✓ <https://github.com/Security-Onion-Solutions/security-onion/releases/download/v14.04.5.3/securityonion-14.04.5.3.iso>

## X. 도구


---

### ❖ Side-Kick

- ① `sudo docker run -t -i -d -p 3331:3000 --name ntopng1 lucaderi/ntopng-docker`
- ② `sudo docker run -t -i -d --net=host --name ntopng2 lucaderi/ntopng-docker`
- ③ `sudo docker run --privileged -it -d --net=host --name ntopng lucaderi/ntopng-docker # @ cumulus`

메모:

# JS Lab

- 
- I. 실습 환경
  - II. 라우터 (VyOS)
  - III. vUTM (pfSense)
  - IV. 리눅스 (Linux)
  - V. 컨테이너 (Docker)
  - VI. OVS (Open vSwitch)
  - VII. SDN 제어기 (ONOS)
  - VIII. **Container Networking** (Docker)
  - IX. **Cloud Networking** (Rancher/K8s/Istio)
  - X. 도구 (NetData, ntopng, Security Onion)
  - ❖ 별첨

# ❖ 별첨 1. Installing the Chart

## ❖ Helm chart

① `helm install stable/wordpress`

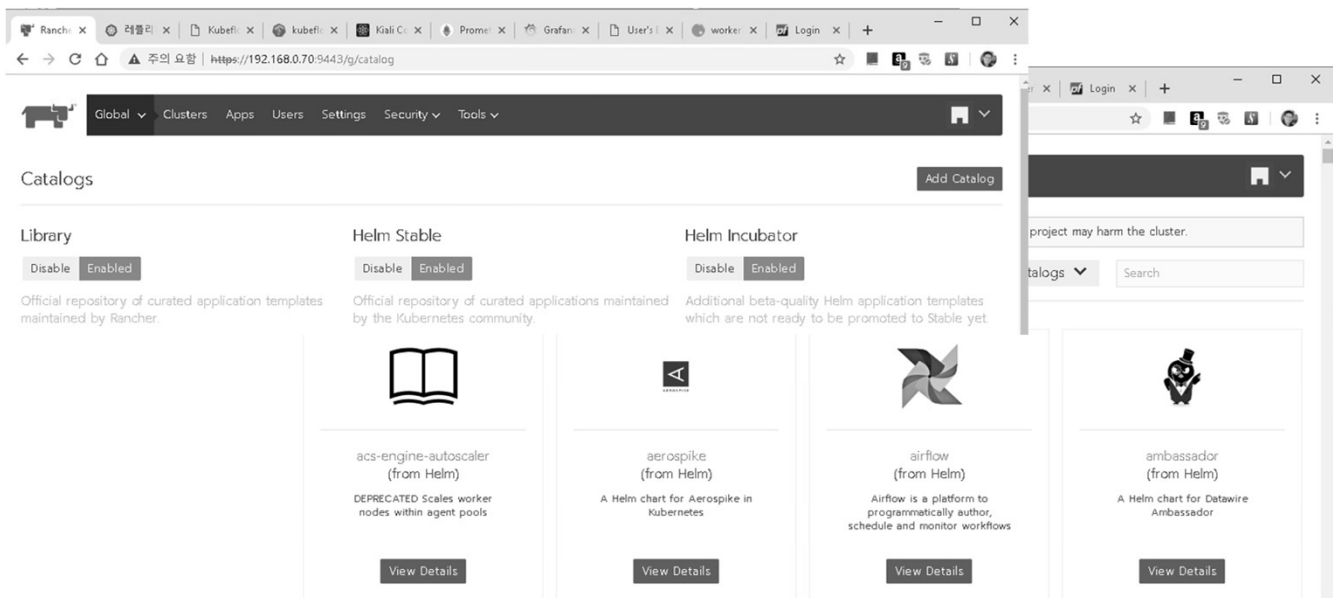
② `helm install --name my-release stable/wordpress`

③ `helm delete my-release`

④ `helm install --name my-release \`  
`--set`

`wordpressUsername=admin,wordpressPassword=password,mariadb.mariadbRootPassword=secretpassword \`  
`stable/wordpress`

⑤ `helm install --name my-release -f values.yaml`  
`stable/wordpress`



### 메모:

- 비교: Docker Compose, Ansible, OpenStack Heat, K8s, TOSCA
- Rancher의 Helm 사용 App Catalog 비교

# ❖ 별첨 1. Installing the Chart

## ❖ Helm chart

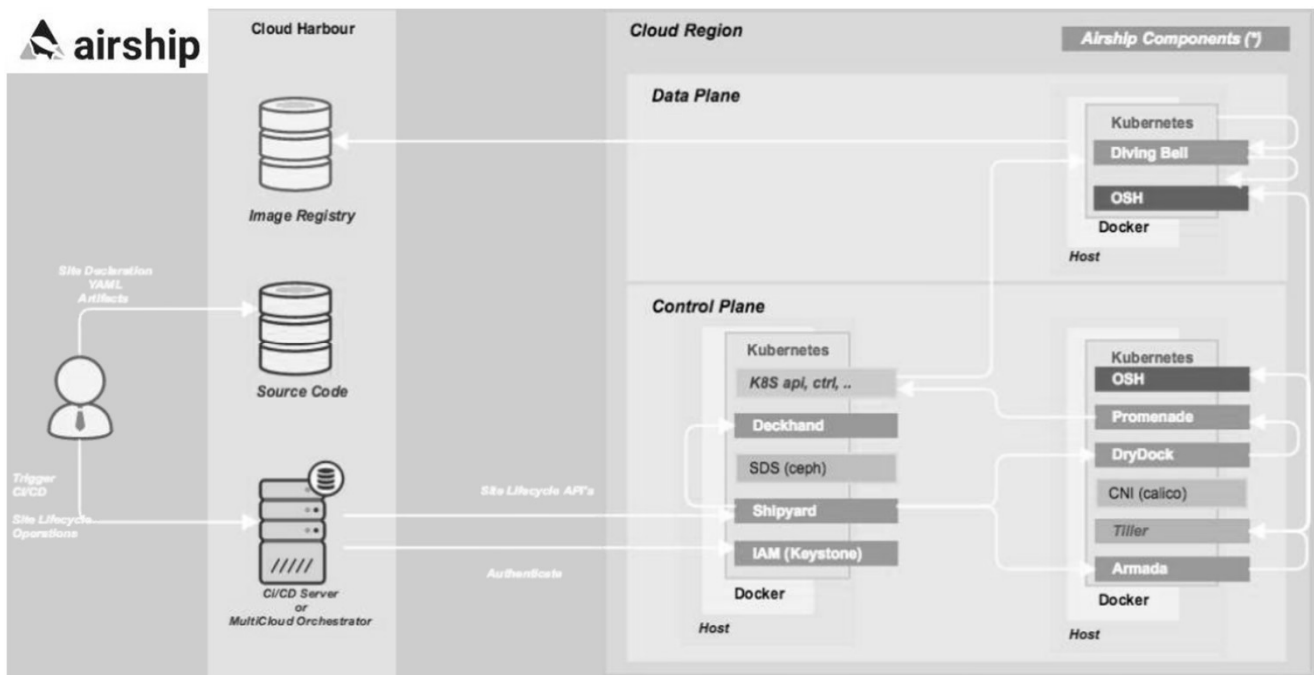
Parameter	Description	Default
global.imageRegistry	Global Docker image registry	nil
global.imagePullSecrets	Global Docker registry secret names as an array	[] (does not add image pull secrets to deployed pods)
image.registry	WordPress image registry	docker.io
image.repository	WordPress image name	bitnami/wordpress
image.tag	WordPress image tag	{VERSION}
image.pullPolicy	Image pull policy	Always if imageTag is latest, else IfNotPresent
image.pullSecrets	Specify docker-registry secret names as an array	[] (does not add image pull secrets to deployed pods)
wordpressUsername	User of the application	user
wordpressPassword	Application password	random 10 character long alphanumeric string
wordpressEmail	Admin email	user@example.com
wordpressFirstName	First name	FirstName
wordpressLastName	Last name	LastName
wordpressBlogName	Blog name	User's Blog!
wordpressTablePrefix	Table prefix	wp_
allowEmptyPassword	Allow DB blank passwords	true
allowOverrideNone	Set Apache AllowOverride directive to None	no
customHTAccessCM	Configmap with custom wordpress-htaccess.conf directives	nil
smtpHost	SMTP host	nil
smtpPort	SMTP port	nil
smtpUser	SMTP user	nil
smtpPassword	SMTP password	nil
smtpUsername	User name for SMTP emails	nil
smtpProtocol	SMTP protocol [tls, ssl]	nil
replicaCount	Number of WordPress Pods to run	1
mariadb.enabled	Deploy MariaDB container(s)	true
mariadb.rootUser.password	MariaDB admin password	nil
mariadb.db.name	Database name to create	bitnami_wordpress
mariadb.db.user	Database user to create	bn_wordpress
mariadb.db.password	Password for the database	random 10 character long alphanumeric string
externalDatabase.host	Host of the external database	localhost
externalDatabase.user	Existing username in the external db	bn_wordpress
externalDatabase.password	Password for the above username	nil
externalDatabase.database	Name of the existing database	bitnami_wordpress
externalDatabase.port	Database port number	3306
service.annotations	Service annotations	{}
service.type	Kubernetes Service type	LoadBalancer
service.port	Service HTTP port	80
service.httpsPort	Service HTTPS port	443
service.externalTrafficPolicy	Enable client source IP preservation	Cluster
service.nodePorts.http	Kubernetes http node port	""
service.nodePorts.https	Kubernetes https node port	""
service.extraPorts	Extra ports to expose in the service (normally used with the sidecar value)	nil
healthcheckHttps	Use https for liveness and readiness	false
livenessProbeHeaders	Headers to use for livenessProbe	nil
readinessProbeHeaders	Headers to use for readinessProbe	nil
ingress.enabled	Enable ingress controller resource	false
ingress.certManager	Add annotations for cert-manager	false
ingress.annotations	Ingress annotations	{}
ingress.hosts[0].name	Hostname to your Wordpress installation	wordpress.local
ingress.hosts[0].path	Path within the url structure	/
ingress.tls[0].hosts[0]	TLS hosts	wordpress.local
ingress.tls[0].secretName	TLS Secret (certificates)	wordpress.local-tls
ingress.secrets[0].name	TLS Secret Name	nil
ingress.secrets[0].certificate	TLS Secret Certificate	nil
ingress.secrets[0].key	TLS Secret Key	nil
persistence.enabled	Enable persistence using PVC	true
persistence.existingClaim	Enable persistence using an existing PVC	nil
persistence.storageClass	PVC Storage Class	nil (uses alpha storage class annotation)
persistence.accessMode	PVC Access Mode	ReadWriteOnce
persistence.size	PVC Storage Request	10Gi
nodeSelector	Node labels for pod assignment	{}
tolerations	List of node taints to tolerate	[]
affinity	Map of node/pod affinities	{}
podAnnotations	Pod annotations	{}
metrics.enabled	Start a side-car prometheus exporter	false
metrics.image.registry	Apache exporter image registry	docker.io
metrics.image.repository	Apache exporter image name	lusotycoon/apache-exporter
metrics.image.tag	Apache exporter image tag	v0.5.0
metrics.image.pullPolicy	Image pull policy	IfNotPresent
metrics.image.pullSecrets	Specify docker-registry secret names as an array	[] (does not add image pull secrets to deployed pods)
metrics.podAnnotations	Additional annotations for Metrics exporter pod	{prometheus.io/scrape: "true", prometheus.io/port: "9117"}
metrics.resources	Exporter resource requests/limit	{}
sidecars	Attach additional containers to the pod	nil

## ❖ 별첨 2. Airship

❖ Ubuntu 16.04 VM (Min 4vCPU/20GB RAM/32GB disk)

❖ This will deploy Airship and Openstack Helm (OSH)

- ① `sudo -i`
- ② `mkdir -p /root/deploy && cd "$_"`
- ③ `git clone https://git.openstack.org/openstack/airship-in-a-bottle`
- ④ `cd /root/deploy/airship-in-a-bottle/manifests/dev_single_node`
- ⑤ `./airship-in-a-bottle.sh`



### 메모:

- <https://github.com/openstack/airship-in-a-bottle>
- Virtualization enabled Ubuntu Host
- `curl -O https://git.airshipit.org/cgit/airship-in-a-bottle/plain/Vagrantfile`
- `vagrant up`



## ❖ 별첨 2. Airship

```

root@airship:~# kubectl get services --all-namespaces
NAMESPACE   NAME                                TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
default     kubernetes                          ClusterIP  10.96.0.1        <none>           443/TCP          1d
kube-system calico-etcd                         ClusterIP  10.96.232.136   <none>           6666/TCP         1d
kube-system coredns                             ClusterIP  10.96.0.10      <none>           53/UDP,53/TCP    1d
kube-system ingress                   ClusterIP  None            <none>           80/TCP,443/TCP,18080/TCP 1d
kube-system ingress-error-pages      ClusterIP  None            <none>           80/TCP           1d
kube-system ingress-exporter         ClusterIP  10.96.226.29   <none>           10254/TCP        1d
kube-system kubernetes-apiservert    ClusterIP  10.96.230.68   <none>           6443/TCP         1d
kube-system kubernetes-etcd          ClusterIP  10.96.0.2       <none>           2379/TCP         1d
kube-system nfs-provisioner           ClusterIP  10.96.154.39   <none>           2049/TCP,20048/TCP,111/TCP,111/UDP 1d
openstack   airship-rabb-dsv-5c0ce1             ClusterIP  None            <none>           5672/TCP,25672/TCP,15672/TCP 1d
openstack   airship-rabb-mgr-5c0ce1            ClusterIP  10.96.191.94   <none>           80/TCP,443/TCP   1d
openstack   cloudformation                     ClusterIP  10.96.122.200   <none>           80/TCP,443/TCP   1d
openstack   glance                              ClusterIP  10.96.75.218   <none>           80/TCP,443/TCP   1d
openstack   glance-api                          ClusterIP  10.96.246.179   <none>           9292/TCP         1d
openstack   glance-registry                    ClusterIP  10.96.169.125   <none>           9191/TCP         1d
openstack   heat                                 ClusterIP  10.96.132.160   <none>           80/TCP,443/TCP   1d
openstack   heat-api                            ClusterIP  10.96.111.46   <none>           8004/TCP         1d
openstack   heat-cfn                            ClusterIP  10.96.102.38   <none>           8000/TCP         1d
openstack   horizon                             ClusterIP  10.96.251.9     <none>           80/TCP,443/TCP   1d
openstack   horizon-dashboard                  NodePort  10.96.197.72   <none>           80:31030/TCP     1d
openstack   horizon-int                         ClusterIP  10.96.44.137   <none>           80/TCP           1d
openstack   ingress                             ClusterIP  10.96.133.162   <none>           80/TCP,443/TCP,18080/TCP 1d
openstack   ingress-error-pages                ClusterIP  None            <none>           80/TCP           1d
openstack   ingress-exporter                   ClusterIP  10.96.26.54     <none>           10254/TCP        1d
openstack   keystone                            ClusterIP  10.96.107.227   <none>           80/TCP,443/TCP   1d
openstack   keystone-api                       ClusterIP  10.96.234.74   <none>           5000/TCP         1d
openstack   mariadb                             ClusterIP  10.96.86.196   <none>           3306/TCP         1d
openstack   mariadb-discovery                  ClusterIP  None            <none>           3306/TCP,4567/TCP 1d
openstack   mariadb-ingress-error-pages        ClusterIP  None            <none>           80/TCP           1d
openstack   mariadb-server                     ClusterIP  10.96.22.67     <none>           3306/TCP         1d
openstack   memcached                           ClusterIP  10.96.64.153   <none>           11211/TCP        1d
openstack   metadata                            ClusterIP  10.96.112.161   <none>           80/TCP,443/TCP   1d
openstack   neutron                             ClusterIP  10.96.219.163   <none>           80/TCP,443/TCP   1d
openstack   neutron-server                     ClusterIP  10.96.106.117   <none>           9696/TCP         1d
openstack   nova                                 ClusterIP  10.96.40.151   <none>           80/TCP,443/TCP   1d
openstack   nova-api                            ClusterIP  10.96.99.235   <none>           8774/TCP         1d
openstack   nova-metadata                      ClusterIP  10.96.251.50   <none>           8775/TCP         1d
openstack   nova-novncproxy                    ClusterIP  10.96.52.174   <none>           6080/TCP         1d
openstack   novncproxy                          ClusterIP  10.96.244.157   <none>           80/TCP,443/TCP   1d
openstack   placement                           ClusterIP  10.96.12.252   <none>           80/TCP,443/TCP   1d
openstack   placement-api                      ClusterIP  10.96.172.114   <none>           8778/TCP         1d
openstack   rabbitmq                             ClusterIP  10.96.193.158   <none>           5672/TCP,25672/TCP,15672/TCP 1d
ucp         airflow-flower                      ClusterIP  10.96.22.100   <none>           5555/TCP         1d
ucp         airflow-web                          ClusterIP  10.96.83.4     <none>           80/TCP           1d
ucp         airflow-web-int                      NodePort  10.96.200.229   <none>           8080:30004/TCP   1d
ucp         airflow-worker                       ClusterIP  10.96.138.1    <none>           8793/TCP         1d
ucp         airflow-worker-discovery             ClusterIP  None            <none>           8793/TCP         1d
ucp         airship-ucp--dsv-8e72c0             ClusterIP  None            <none>           5672/TCP,25672/TCP,15672/TCP 1d
ucp         airship-ucp--mgr-8e72c0            ClusterIP  10.96.58.52    <none>           80/TCP,443/TCP   1d
ucp         armada                               ClusterIP  10.96.226.57   <none>           80/TCP,443/TCP   1d
ucp         armada-api                          ClusterIP  10.96.69.184   <none>           8000/TCP         1d
ucp         barbican                             ClusterIP  10.96.156.213   <none>           80/TCP,443/TCP   1d
ucp         barbican-api                        ClusterIP  10.96.5.47     <none>           9311/TCP         1d
ucp         deckhand-api                        ClusterIP  10.96.65.235   <none>           80/TCP,443/TCP   1d
ucp         deckhand-int                         ClusterIP  10.96.67.0     <none>           9000/TCP         1d
ucp         drydock-api                         ClusterIP  10.96.129.232   <none>           9000/TCP         1d
ucp         ingress                             ClusterIP  10.96.164.53   <none>           80/TCP,443/TCP,18080/TCP 1d
ucp         ingress-error-pages                 ClusterIP  None            <none>           80/TCP           1d
ucp         ingress-exporter                     ClusterIP  10.96.229.232   <none>           10254/TCP        1d
ucp         keystone                            ClusterIP  10.96.131.108   <none>           80/TCP,443/TCP   1d
ucp         keystone-api                        ClusterIP  10.96.243.126   <none>           5000/TCP         1d
ucp         maas-ingress-error                  ClusterIP  10.96.51.219   <none>           8080/TCP         1d
ucp         maas-region                         ClusterIP  10.96.27.104   <none>           80/TCP,8000/TCP   1d
ucp         mariadb                              ClusterIP  10.96.55.104   <none>           3306/TCP         1d
ucp         mariadb-discovery                   ClusterIP  None            <none>           3306/TCP,4567/TCP 1d
ucp         mariadb-ingress-error-pages         ClusterIP  None            <none>           80/TCP           1d
ucp         mariadb-server                       ClusterIP  10.96.248.182   <none>           3306/TCP         1d
ucp         memcached                           ClusterIP  10.96.227.155   <none>           11211/TCP        1d
ucp         postgresql                           ClusterIP  10.96.228.132   <none>           5432/TCP         1d
ucp         promenade-api                       ClusterIP  10.96.115.230   <none>           80/TCP           1d
ucp         rabbitmq                             ClusterIP  10.96.139.237   <none>           5672/TCP,25672/TCP,15672/TCP 1d
ucp         shipyard-api                        ClusterIP  10.96.190.85   <none>           80/TCP           1d
ucp         shipyard-int                         NodePort  10.96.78.89    <none>           9000:30003/TCP   1d
root@airship:~#

```



## ❖ 별첨 3. Armada Integration

---

### ❖ Airship

### ❖ Containerization – Armada Integration

- ① `$ sudo docker run -d --net host -p 8000:8000 --name armada \`  
`-v ~/.kube/config:/armada/.kube/config \`  
`-v $(pwd)/examples/./examples`  
`quay.io/airshipit/armada:latest`
- ② `armada apply examples/openstack-helm.yaml [ --debug ]`

#### 메모:

- <https://github.com/openstack/airship-armada>
- <https://docs.starlingx.io/specs/specs/2019.03/approved/containerization-2003908-armada-integration.html>

# ❖ 별첨 4. TCP/UDP Port Number

## ❖ TCP/UDP Port Number

7 Echo	554 RTSP	2745 Bagle.H	6891-6901 Windows Live
19 Chargen	546-547 DHCPv6	2967 Symantec AV	6970 Quicktime
20-21 FTP	560 rmonitor	3050 Interbase DB	7212 GhostSurf
22 SSH/SCP	563 NNTP over SSL	3074 XBOX Live	7648-7649 CU-SeeMe
23 Telnet	587 SMTP	3124 HTTP Proxy	8000 Internet Radio
25 SMTP	591 FileMaker	3127 MyDoom	8080 HTTP Proxy
42 WINS Replication	593 Microsoft DCOM	3128 HTTP Proxy	8086-8087 Kaspersky AV
43 WHOIS	631 Internet Printing	3222 GLBP	8118 Privoxy
49 TACACS	636 LDAP over SSL	3260 iSCSI Target	8200 VMware Server
53 DNS	639 MSDP (PIM)	3306 MySQL	8500 Adobe ColdFusion
67-68 DHCP/BOOTP	646 LDP (MPLS)	3389 Terminal Server	8767 TeamSpeak
69 TFTP	691 MS Exchange	3689 iTunes	8866 Bagle.B
70 Gopher	860 iSCSI	3690 Subversion	9100 HP JetDirect
79 Finger	873 rsync	3724 World of Warcraft	9101-9103 Bacula
80 HTTP	902 VMware Server	3784-3785 Ventrilo	9119 MxIt
88 Kerberos	989-990 FTP over SSL	4333 mSQL	9800 WebDAV
102 MS Exchange	993 IMAP4 over SSL	4444 Blaster	9898 Dabber
110 POP3	995 POP3 over SSL	4664 Google Desktop	9988 Rbot/Spybot
113 Ident	1025 Microsoft RPC	4672 eMule	9999 Urchin
119 NNTP (Usenet)	1026-1029 Windows Messenger	4899 Radmin	10000 Webmin
123 NTP	1080 SOCKS Proxy	5000 UPnP	10000 BackupExec
135 Microsoft RPC	1080 MyDoom	5001 Slingbox	10113-10116 NetIQ
137-139 NetBIOS	1194 OpenVPN	5001 Iperf	11371 OpenPGP
143 IMAP4	1214 Kazaa	5004-5005 RTP	12035-12036 Second Life
161-162 SNMP	1241 Nessus	5050 Yahoo! Messenger	12345 NetBus
177 XDMCP	1311 Dell OpenManage	5060 SIP	13720-13721 NetBackup
179 BGP	1337 WASTE	5190 AIM/ICQ	14567 Battlefield
201 AppleTalk	1433-1434 Microsoft SQL	5222-5223 XMPP/Jabber	15118 Dipnet/Oddbob
264 BGMP	1512 WINS	5432 PostgreSQL	19226 AdminSecure
318 TSP	1589 Cisco VQP	5500 VNC Server	19638 Ensism
381-383 HP Openview	1701 L2TP	5554 Sasser	20000 Usermin
389 LDAP	1723 MS PPTP	5631-5632 pcAnywhere	24800 Synergy
411-412 Direct Connect	1725 Steam	5800 VNC over HTTP	25999 Xfire
443 HTTP over SSL	1741 CiscoWorks 2000	5900+ VNC Server	27015 Half-Life
445 Microsoft DS	1755 MS Media Server	6000-6001 X11	27374 Sub7
464 Kerberos	1812-1813 RADIUS	6112 Battle.net	28960 Call of Duty
465 SMTP over SSL	1863 MSN	6129 DameWare	31337 Back Orifice
497 Retrospect	1985 Cisco HSRP	6257 WinMX	33434+ traceroute
500 ISAKMP	2000 Cisco SCCP	6346-6347 Gnutella	
512 rexec	2002 Cisco ACS	6500 GameSpy Arcade	<b>Legend</b>
513 rlogin	2049 NFS	6566 SANE	Chat
514 syslog	2082-2083 cPanel	6588 AnalogX	Encrypted
515 LPD/LPR	2100 Oracle XDB	6665-6669 IRC	Gaming
520 RIP	2222 DirectAdmin	6679/6697 IRC over SSL	Malicious
521 RIPng (IPv6)	2302 Halo	6699 Napster	Peer to Peer
540 UUCP	2483-2484 Oracle DB	6881-6999 BitTorrent	Streaming

### 메모:

- 컨테이너 환경에서는 Well-known TCP/UDP Port와 겹치지 않게 관리 필요

# JS Lab

