


(Ubuntu Server 16.04) v1.0



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

JS Lab

안종석
james@jslab.kr

2018년 9월

목차



1. 실습 환경
2. vRouter (VyOS..)
3. Host (Ubuntu..)
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
10. Container Networking (Docker..)

목차



- 1. 실습 환경**
2. vRouter (VyOS..)
3. Host (Ubuntu..)
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
10. **Container Networking** (Docker..)

1. 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	redhat packaging (.rpm → yum(dnf) install)	RHEL, CentOS
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

1. 실습 환경

❖ 네트워킹 오픈소스 소프트웨어

이름	구분	라이선스	출범일
Edgent	Network Analytics	Apache	2016-12
linkerd	NFVI - Infrastructure Layer, VNF - Layer 4-7 Acceleration and Caching	Apache	2016-04
Cilium	NFVI - Infrastructure Layer, VNF - Layer 4-7 Security	Apache	2017-03
BiRD	NFVI - Switching and Routing	GPLv2	2013-03
NetBox	NFVI - Switching and Routing	Apache	2016-06
OSM (Open Source MANO)	NFV MANO - NFV Management and Orchestration	Apache	2016-05
Facebook Open Switching System (FBOSS)	NFVI - Switching and Routing, NFVI - Network Operating Systems	BSD	2015-03
Faucet SDN Controller	NFVI - Control	Apache	2015-03
GoBGP	NFVI - Switching and Routing	Apache	2017-02
HAProxy	VNF - Layer 4-7 Security, VNF - Layer 4-7 Acceleration and Caching	GPLv2, LGPL	2001-12
YANFF - Yet Another Network Function Framework	NFVI - Infrastructure Layer, VNF - Layer 4-7 Security, VNF - Layer 4-7 Acceleration and Caching	BSD	2017-03
OpenContrail	NFVI - Switching and Routing, NFVI - Control, NFV MANO	Apache	2013-09
OpenDataPlane Project	NFVI - Infrastructure Layer	BSD	2015-02
OpenSwitch	NFVI - Infrastructure Layer, NFVI - Switching and Routing, NFVI - Network Operating Systems	Apache	2016
OPNFV	NFVI - Hardware, NFVI - Infrastructure Layer, NFVI - Switching and Routing, NFVI - Network Operating Systems, NFVI - Control	Apache	2017-09
FD.io	NFVI - Infrastructure Layer, NFVI - Switching and Routing, VNF - Layer 4-7 Acceleration and Caching	Apache	2016-02
OpenDaylight	NFVI - Control	Eclipse Public License Version 1.0	2013-03
Open vSwitch	NFVI - Switching and Routing	Apache	2009-07
Open Network Automation Platform (ONAP)	NFVI - Control, NFV MANO, VNF - Layer 4-7 Security, VNF - Layer 4-7 Acceleration and Caching	Apache	2017-03
Data Plane Development Kit (DPDK)	NFVI - Infrastructure Layer, NFVI - Switching and Routing	BSD	2012-09
FRRouting (FRR)	NFVI - Switching and Routing	GPLv2	2017-10
OpenLSO	NFV MANO	OpenLSO components have individual licenses.	2016-03
NGINX Open Source (OSS)	VNF - Layer 4-7 Security, VNF - Layer 4-7 Acceleration and Caching	BSD	2011-07
Ryu Network Operating System	NFVI - Network Operating Systems, NFVI - Control	Apache	2011-12
Open Network Linux	NFVI - Network Operating Systems	GPLv2, GPLv3, Eclipse Public License 1.0	2014-01
Open Network Install Environment (ONIE)	NFVI - Hardware, Installation Environment	GPLv2	2013-06
SONiC	NFVI - Switching and Routing, NFVI - Network Operating Systems	Apache, GPLv2, Multiple different licenses for components	2016-03
OpenConfig Project	NFV MANO	Apache	2014-10
Central Office Re-architected as a Datacenter (CORD)	NFVI - Infrastructure Layer, NFVI - Network Operating Systems	Apache, ON.Lab Contributor License Agreement	Not provided
Open Networking Operating System (ONOS)	NFVI - Control	Apache, BSD, MIT, ON.Lab Contributor License Agreement	2014-12
OpenStack Neutron	NFVI - Infrastructure Layer	Apache	2013-07
OpenStack Tacker	NFV MANO	Apache	2015-12
P4	NFVI - Infrastructure Layer, NFVI - Switching and Routing	Apache	2015-02
Project Calico	NFVI - Switching and Routing	Apache	2014-07
Open Virtual Network (OVN)	NFVI - Infrastructure Layer, NFVI - Switching and Routing	Apache	2015-01

메모:

1. 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

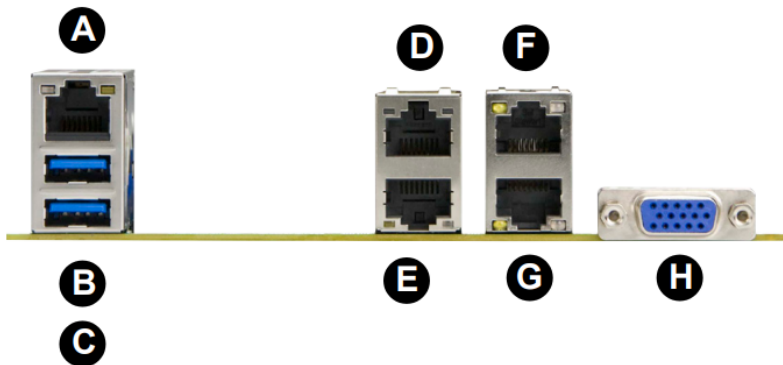
② RAM

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

1. 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 (SanDisk)
- ⑤ ESXi '6.0' vs '6.5' (실습 진행 편리를 위한 선택)
- ⑥ Windows Server 2016 Hyper-v 고려
- ⑦ 개인용 노트북 사용 (PDF viewer, Putty, WEB browser, Software Tools)



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

1. 실습 환경

❖ 실습 구성 @ KOREN AI Network Lab

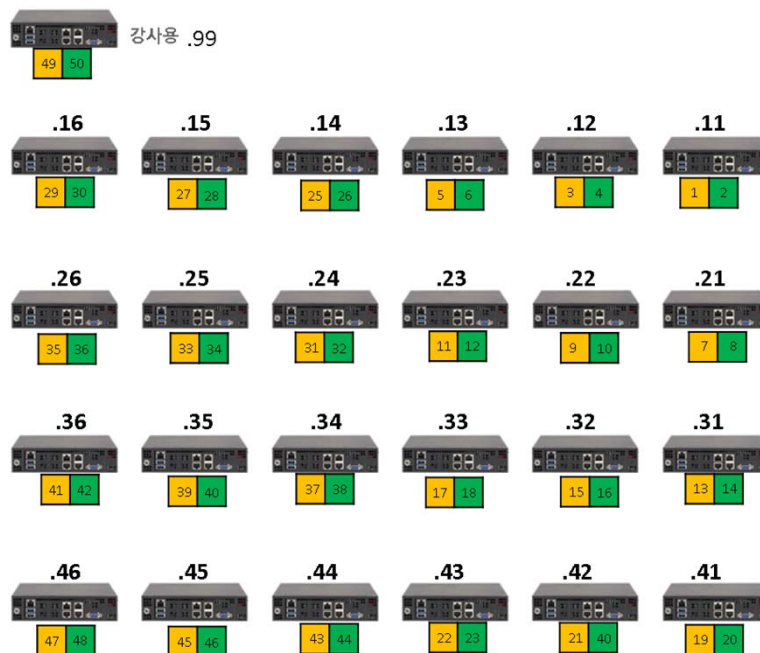
① USB 메모리

- OS
- 소프트웨어 도구 (Software Tools)

② IPMI 연결 이더넷 케이블

③ 인터넷 연결 케이블

④ 좌석 번호 별 서브넷의 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

1. 실습 환경

❖ 하이퍼바이저 비교

- ① Microsoft의 Hyper-v는 평가기간 무제한
- ② vSphere 6.5 평가판은 60일간 모든 기능 제공하며, 평가 기간 종료 후에 상용기능 정지 (**6.7 내용 업데이트 예정)
- ③ 하이퍼바이저 사용 실습에서는 LAN/웹브라우저/PDF뷰어 지원 개인 노트북 지참

기능 \ 제품	Microsoft	VMware vSphere 6.5		
	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/CP U
디스크 사이즈 조정	Hot-grow and shrink	Hot-grow	Hot-grow	Hot-grow
VM 암호화	Yes	No	No?	Yes

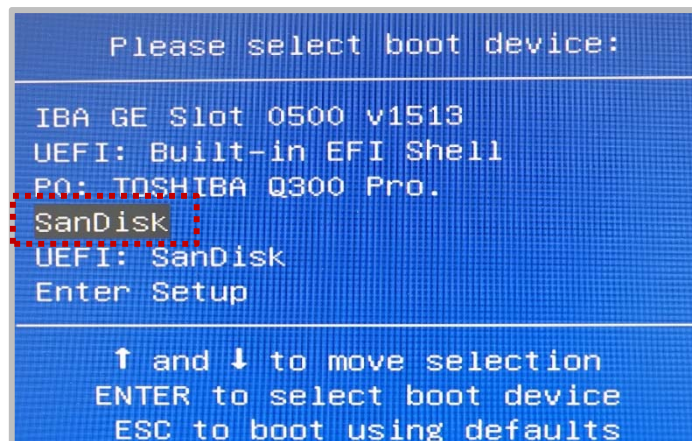
메모:

- 여러 명이 동시 실습을 진행하며 웹브라우저가 동일하지 않은 경우 vSphere 6.0과 전용 클라이언트 소프트웨어 사용 권장
- Type 2 Hypervisor는 VMware (WorkStation) Player 또는 VirtualBox 사용 가능 노트북 미지참 실습은 베어메탈 서버에 리눅스 설치 (USB 허브 필요)

1. 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 '6.0' vs '6.5' vs '6.7'
- ⑥ Windows Server 2016 Hyper-v



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

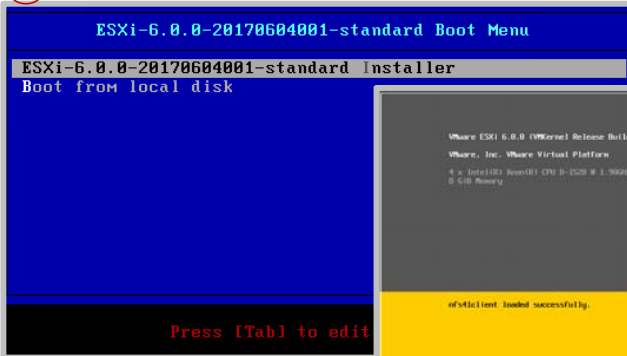
메모:

- 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

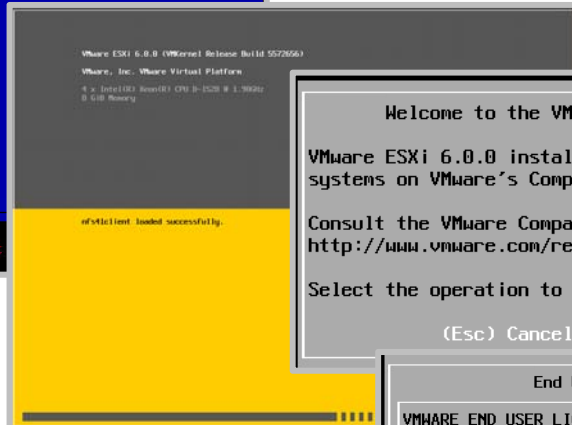
1. 실습 환경

❖ Hypervisor Installation (ESXi 6.0 예)

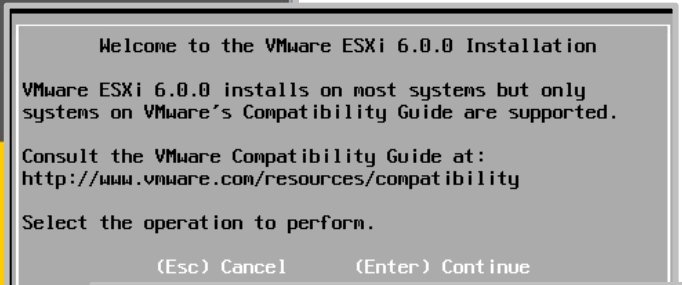
①



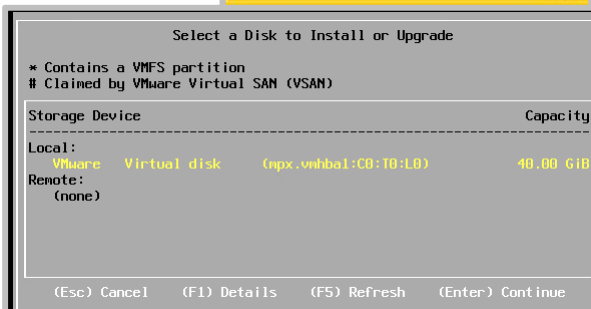
②



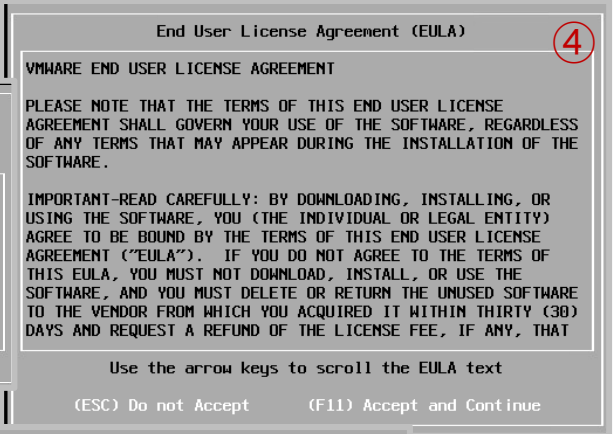
③



⑤



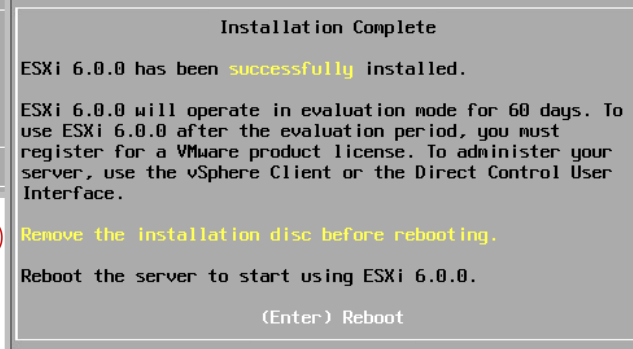
④



⑥



⑦



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>

1. 실습 환경

❖ Hypervisor Installation

① Configure Management Network 선택

② 좌석 번호 'TT' 이용 고정 IP 주소 설정 - 192.168.1.TT

The screenshot shows two panels from a system configuration interface. The left panel, titled 'System Customization', lists various options. The option 'Configure Management Network' is highlighted with a red dashed box and a circled '1'. The right panel, titled 'Configure Management Network', shows the configuration for the management network. The 'IPv4 Address' is set to '192.168.1.229', which is also highlighted with a red dashed box and a circled '2'. The 'Hostname' is set to 'localhost'. Below the IPv4 address, it states 'Network identity acquired from DHCP server 192.168.1.1'. The IPv6 addresses are listed as 'fe80::20c:29ff:fee5:4f66/64'. At the bottom of the right panel, it says 'To view or modify this host's management network settings in detail, press <Enter>.' The bottom of the left panel has '<Up/Down> Select' and the bottom of the right panel has '<Enter> More' and '<Esc> Log Out'.

메모:

1. 실습 환경

❖ Hypervisor Installation

- ① Networking 선택 확인
- ② Network Adaptor 선택 확인

192.168.1.14 - vSphere Client

File Edit View Inventory Administration Plug-ins Help

Home Inventory Inventory

localhost.localdomain VMware ESXi, 6.0.0, 3620759 | Evaluation (59 days remaining)

Getting Started Summary Virtual Machines Resource Allocation Performance Configuration Users Events Permissions

Hardware

- Health Status
- Processors
- Memory
- Storage
- Networking
- Storage Adapters
- Network Adapters
- Advanced Settings
- Power Management

Software

- Licensed Features
- Time Configuration
- DNS and Routing
- Authentication Services
- Virtual Machine Startup/Shutdown
- Virtual Machine Swapfile Location

View: vSphere Standard Switch

Networking Refresh Add Networking... Properties...

Standard Switch: vSwitch0 Remove... Properties...

Virtual Machine Port Group VM Network Physical Adapters vmnic0 1000 Full

VMkernel Port Management Network vmk0 : 192.168.1.14 fe80::ae1f:6bff:fe1b:8c8e

Recent Tasks

Name

localhost.localdomain VMware ESXi, 6.0.0, 3620759 | Evaluation (59 days remaining)

Getting Started Summary Virtual Machines Resource Allocation Performance Configuration Users Events Permissions

Hardware

- Health Status
- Processors
- Memory
- Storage
- Networking
- Storage Adapters
- Network Adapters
- Advanced Settings
- Power Management

Network Adapters

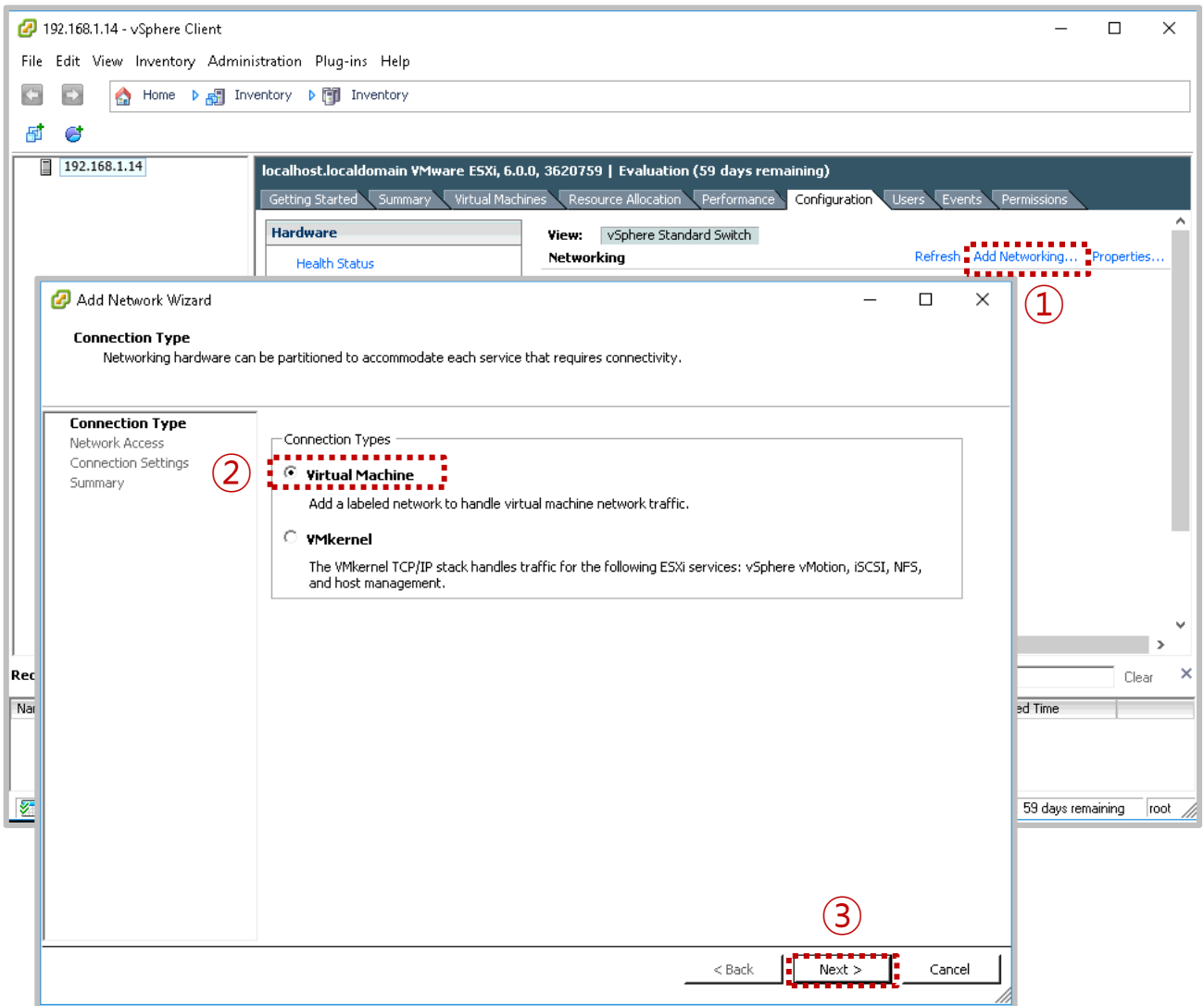
Device	Speed	Configured	Switch	MAC Address	Observed IP ranges
Intel Corporation I350 Gigabit Network Connection					
vmnic1	Down	Negotiate	None	ac:1f:6b:1b:8c:8f	None
vmnic0	1000 Full	Negotiate	vSwitch0	ac:1f:6b:1b:8c:8e	0.0.0.1-255.255.255

메모:

1. 실습 환경

❖ Hypervisor Installation

- ① Add Networking 선택
- ② Virtual Machine 선택

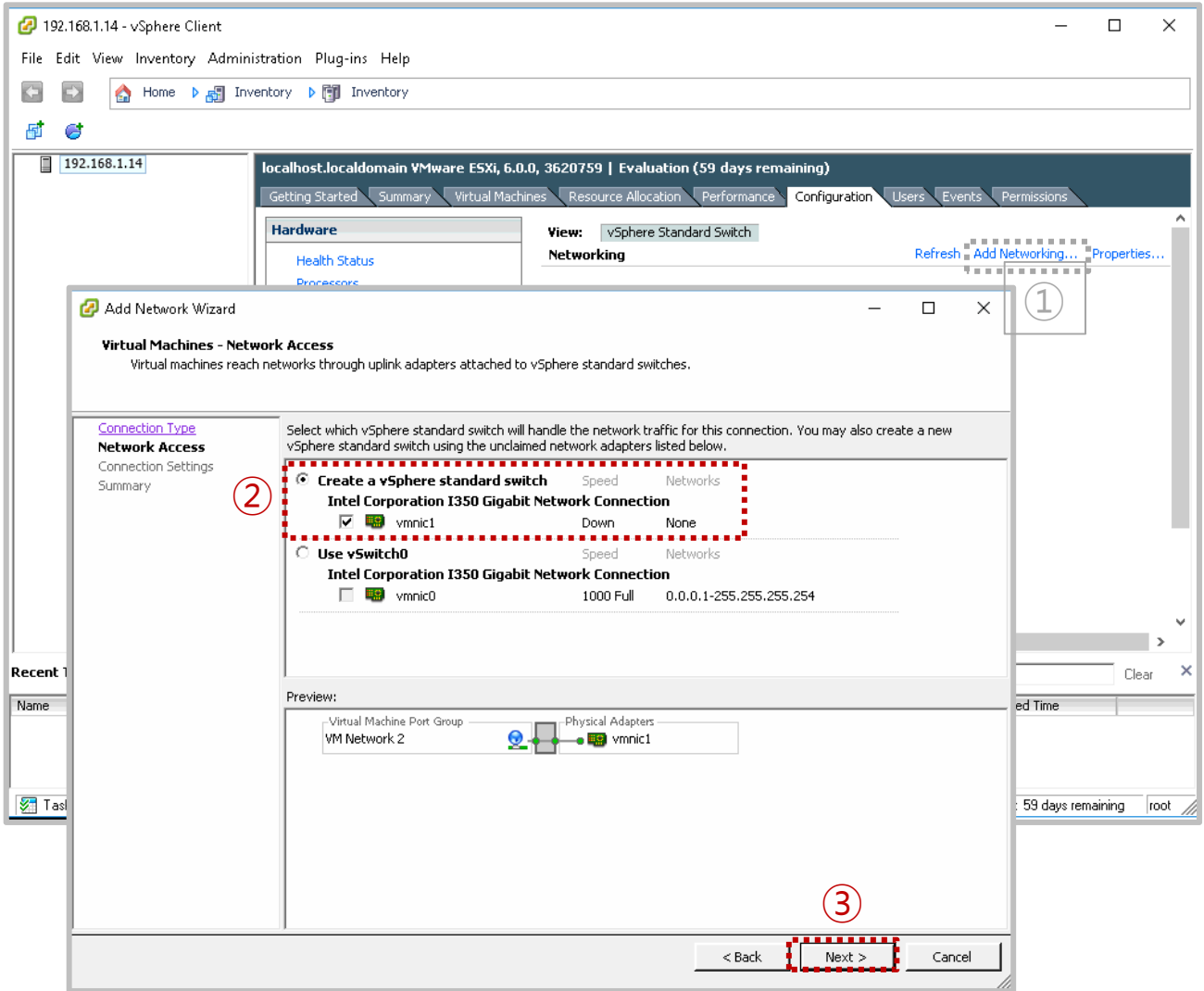


메모:

1. 실습 환경

❖ Hypervisor Installation

- ① 표준 스위치 생성 선택
- ② 유선랜 네트워크 연결 (내부 네트워크를 위한 선택)

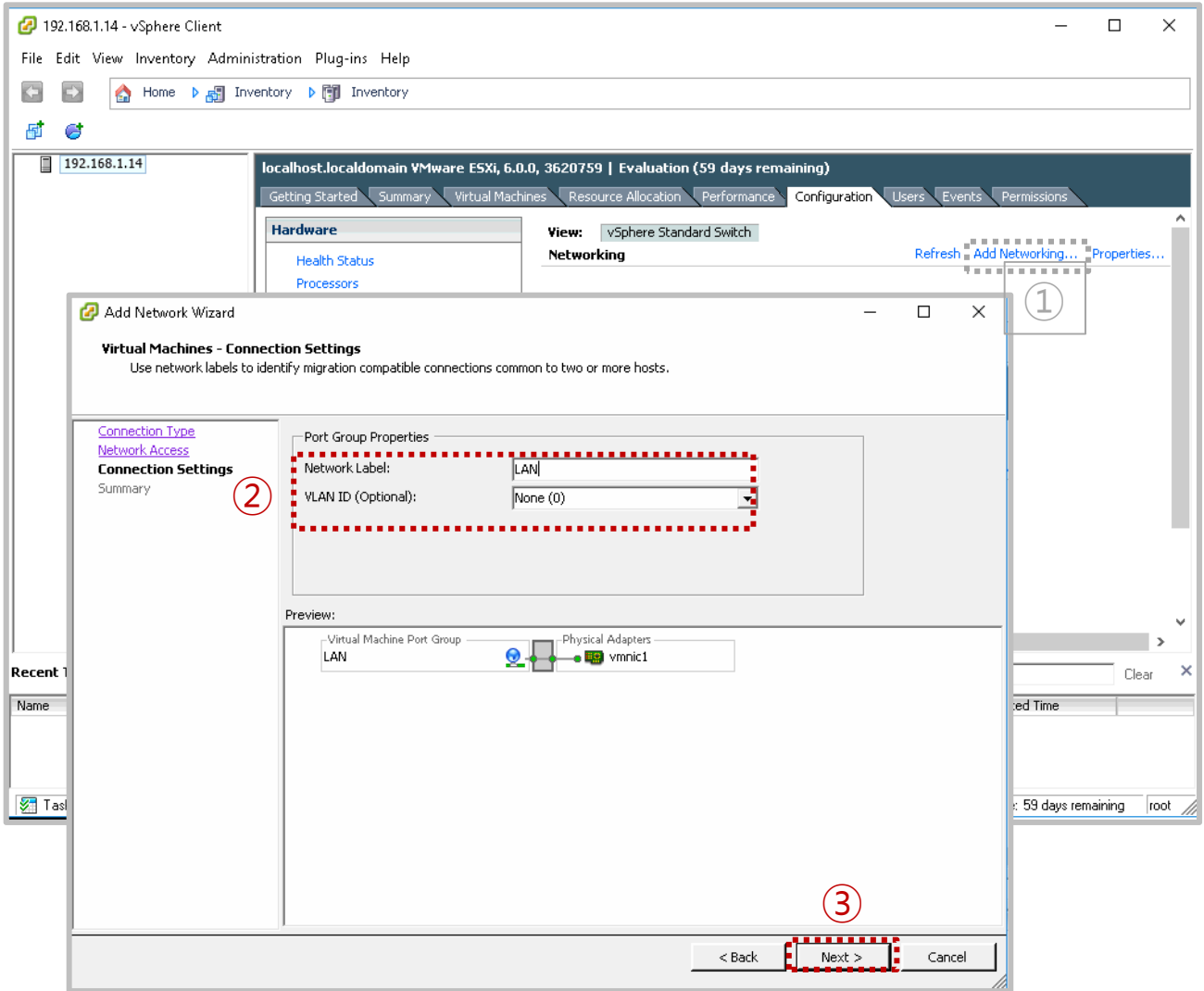


메모:

1. 실습 환경

❖ Hypervisor Installation

- ① 포트그룹 이름 설정
- ② 유선랜 네트워크 연결 (내부 네트워크를 위한 선택)

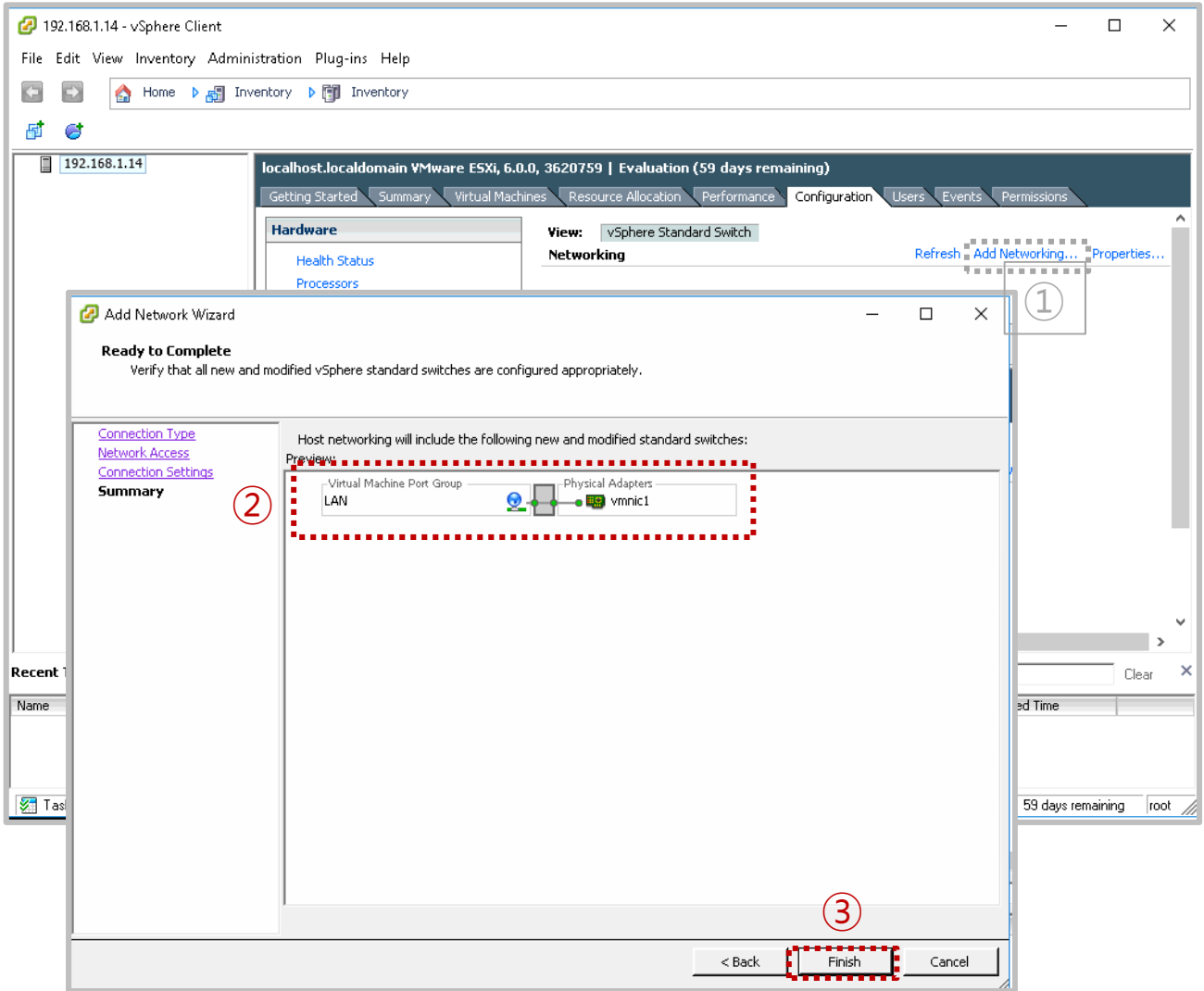


메모:

1. 실습 환경

❖ Hypervisor Installation

- ① 포트그룹 이름 설정
- ② 유선랜 네트워크 연결 (내부 네트워크를 위한 선택)

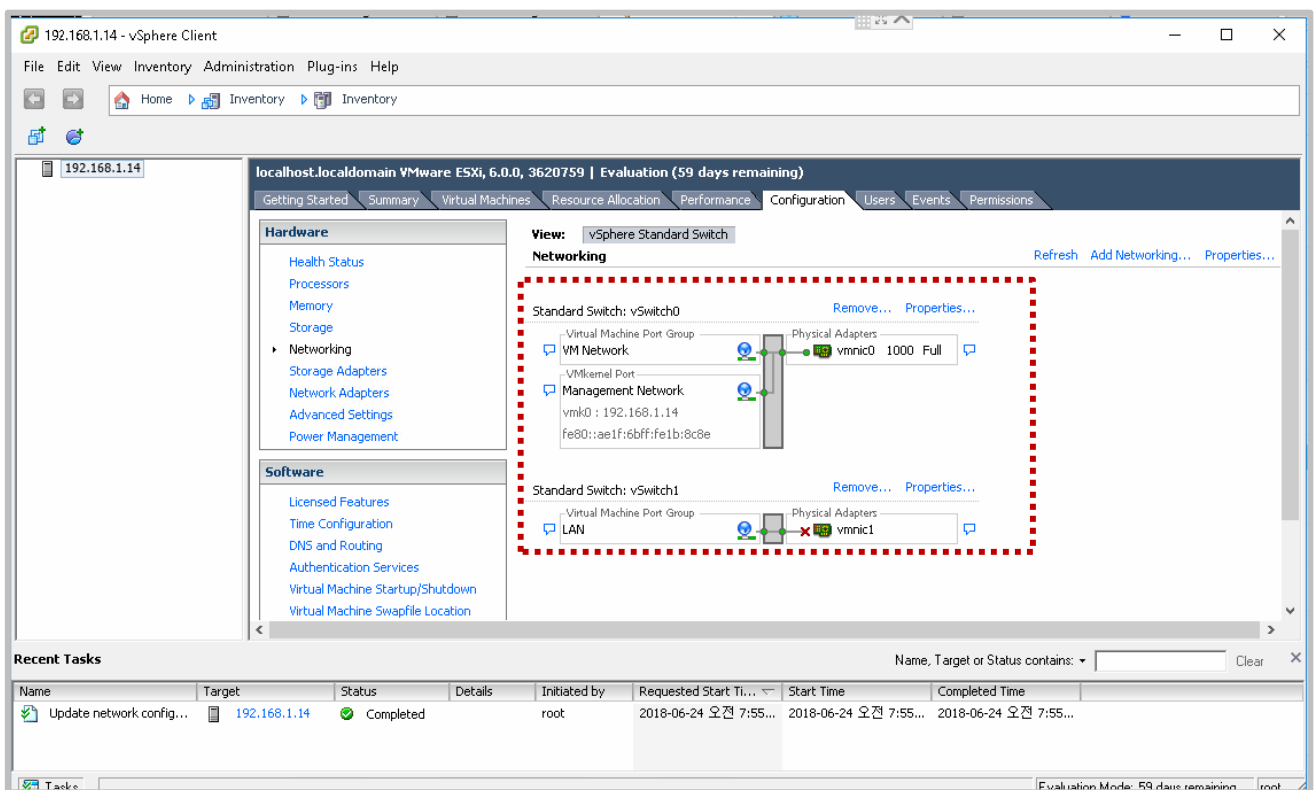


메모:

1. 실습 환경

❖ Hypervisor Installation

- ① 포트그룹 이름 설정
- ② 유선랜 네트워크 연결 (내부 네트워크를 위한 선택)

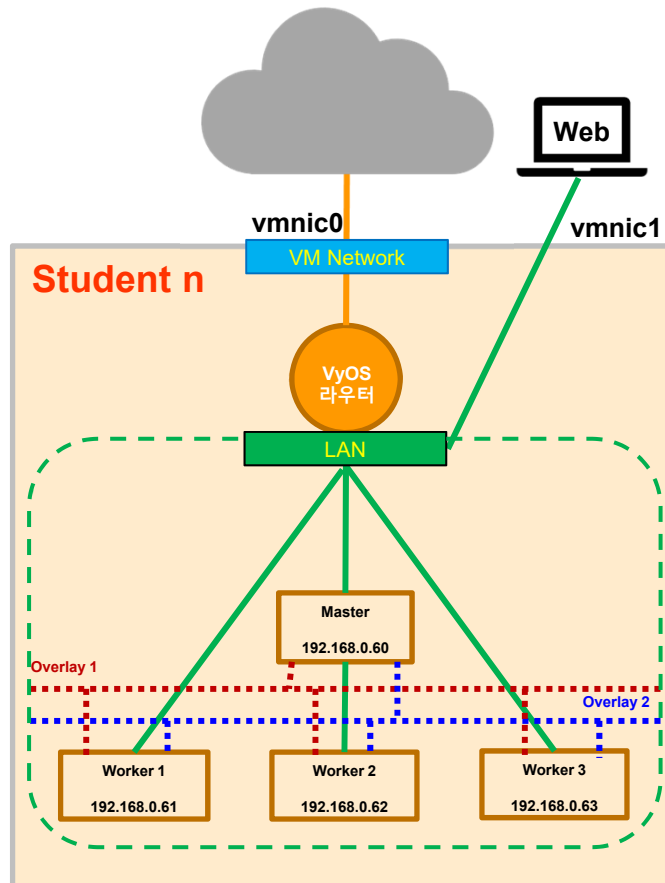


메모:

1. 실습 환경

❖ 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 (외부 유선랜 연결이 어려운 경우 Ubuntu Desktop VM 사용하거나 VyOS를 경유하는 SSH로 연결)



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)
- 호스트용 CentOS 이미지 : http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-Minimal-1708.iso (실습 시간 부족 시 초기설치 완료한 이미지 사용)

목차

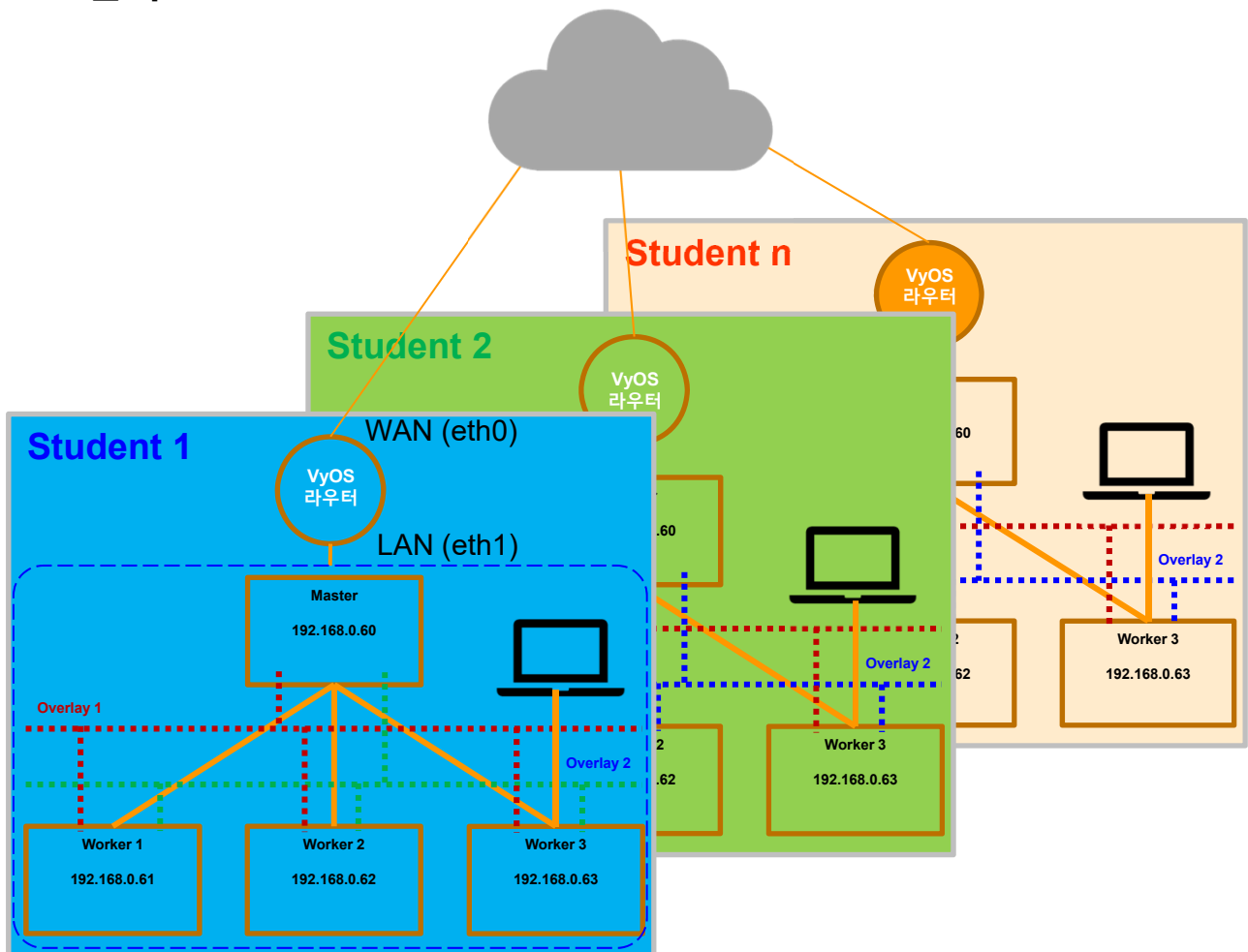


1. 실습 환경
- 2. vRouter (VyOS..)**
3. Host (Ubuntu..)
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
10. **Container Networking (Docker..)**

2. vRouter (VyOS)

❖ 라우터 'VyOS' 설치 환경

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



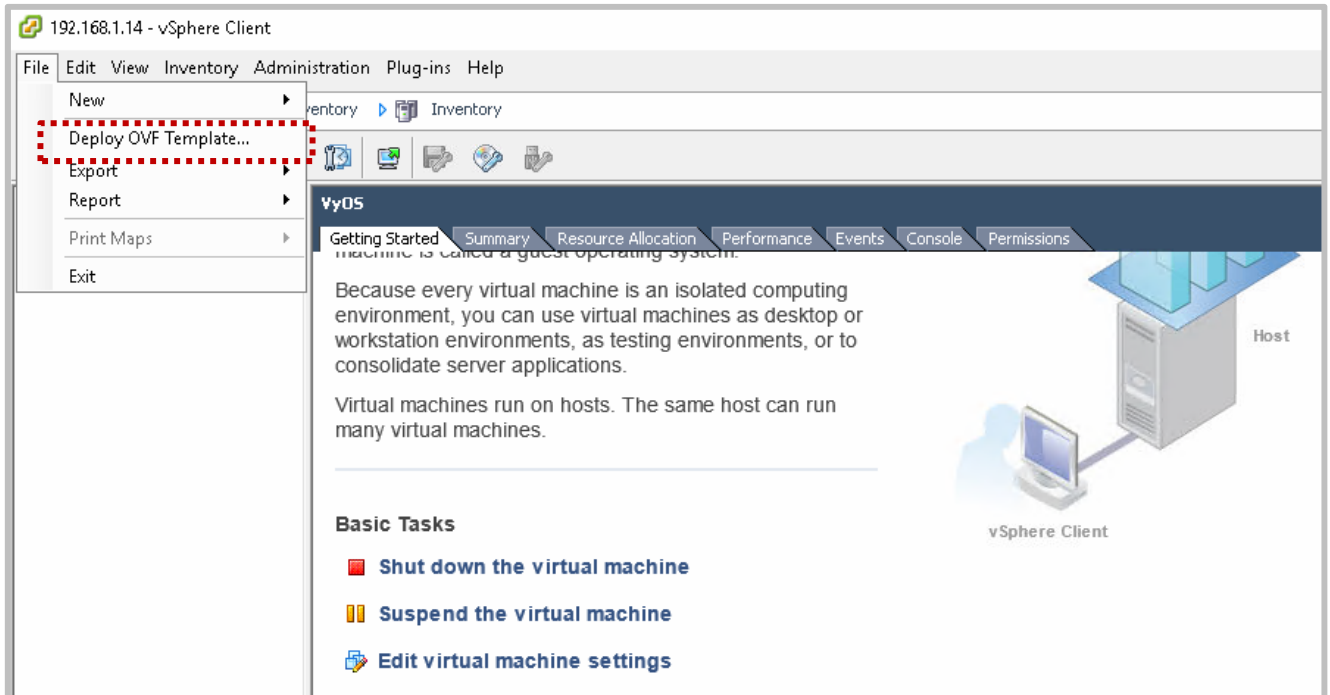
메모:

- 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)
- 호스트용 CentOS 이미지 : http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-Minimal-1708.iso (실습 시간 부족 시 초기설치 완료한 이미지 사용)

2. vRouter (VyOS)

❖ Router(VyOS) Installation

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

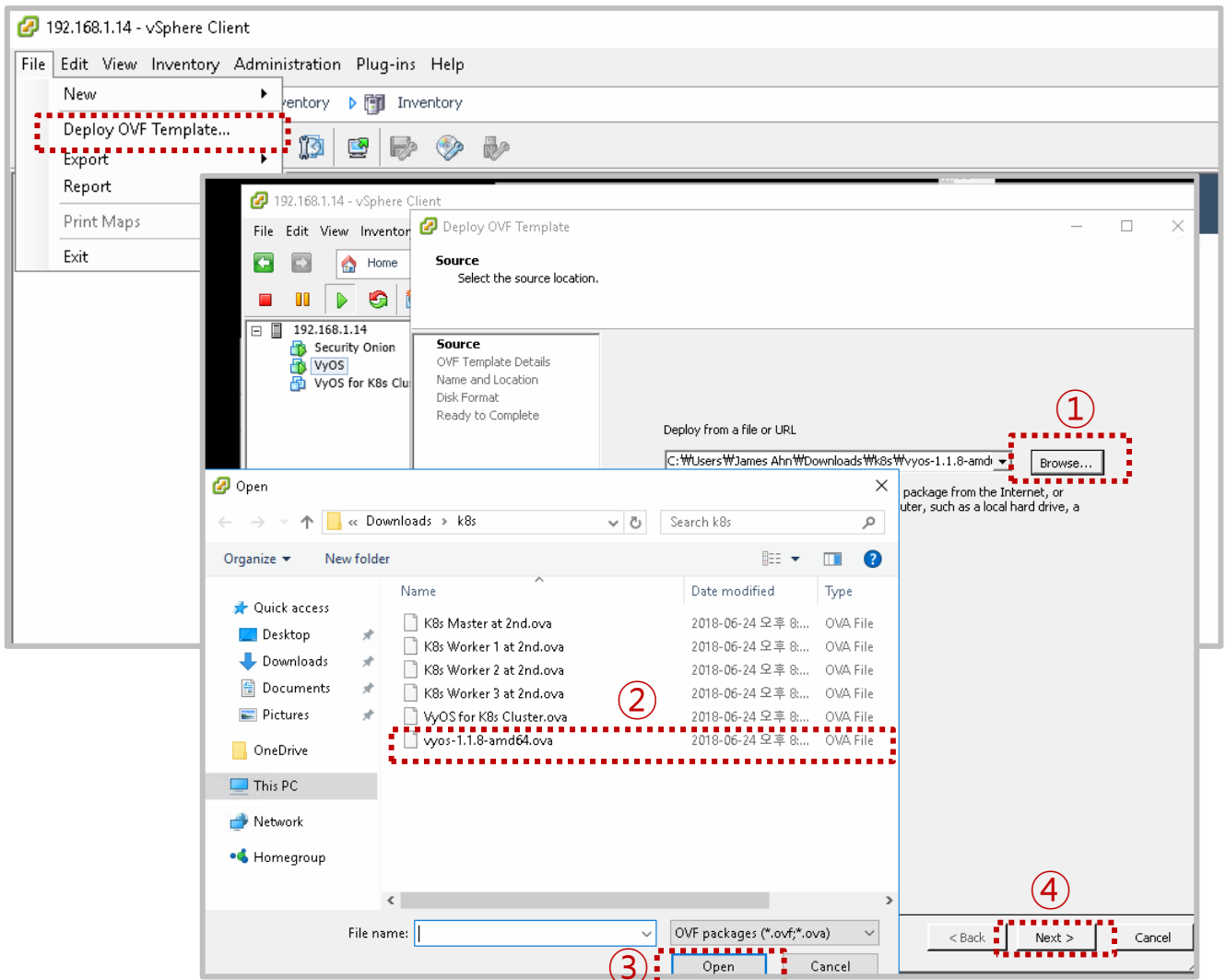


메모:

2. vRouter (VyOS)

❖ Router(VyOS) Installation

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



메모:

- VM 설치방법 1: 우분투(Ubuntu Server/Desktop) OVA 제공
- VM 설치방법 2: 우분투(Ubuntu Server/Desktop) ISO 제공 설치
- VMware Standalone Converter 사용하여 배포

2. vRouter (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
```

가상 라우터 VyOS 터미널 접속 (예)

가상 라우터 VyOS에 SSH 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:~$
```

메모:

- https://wiki.vyos.net/wiki/User_Guide

2. vRouter (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

2. vRouter (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)

2. vRouter (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

2. vRouter (VyOS)

❖ NetFlow @ VyOS 설치 (선택)

- ① config
- ② set system flow-accounting interface eth0
- ③ set system flow-accounting interface eth1
- ④ set system flow-accounting netflow engine-id 100
- ⑤ set system flow-accounting netflow version 5
- ⑥ set system flow-accounting netflow server 192.168.0.208 port 2055
- ⑦ set system flow-accounting netflow server 192.168.0.226 port 2055
- ⑧ set system flow-accounting sflow server 192.168.0.226 port 9996
- ⑨ commit
- ⑩ save

```
flow-accounting {  
    interface eth0  
    interface eth1  
    netflow {  
        engine-id 100  
        server 192.168.0.208 {  
            port 2055  
        }  
        version 5  
    }  
}
```

메모:

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

목차

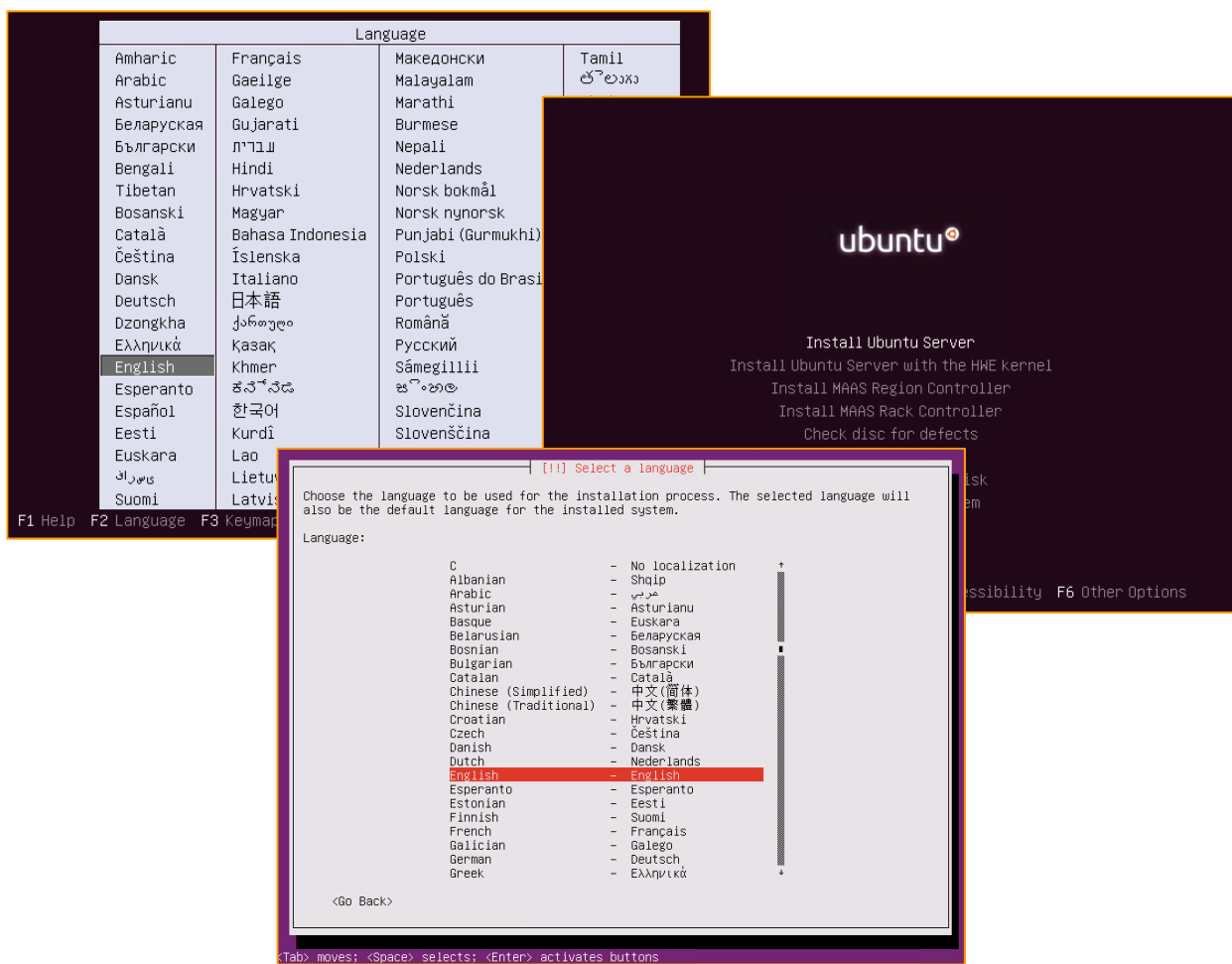


1. 실습 환경
2. vRouter (VyOS..)
- 3. Host (Ubuntu..)**
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
10. **Container Networking** (Docker..)

3. Host (Ubuntu)

❖ 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'**



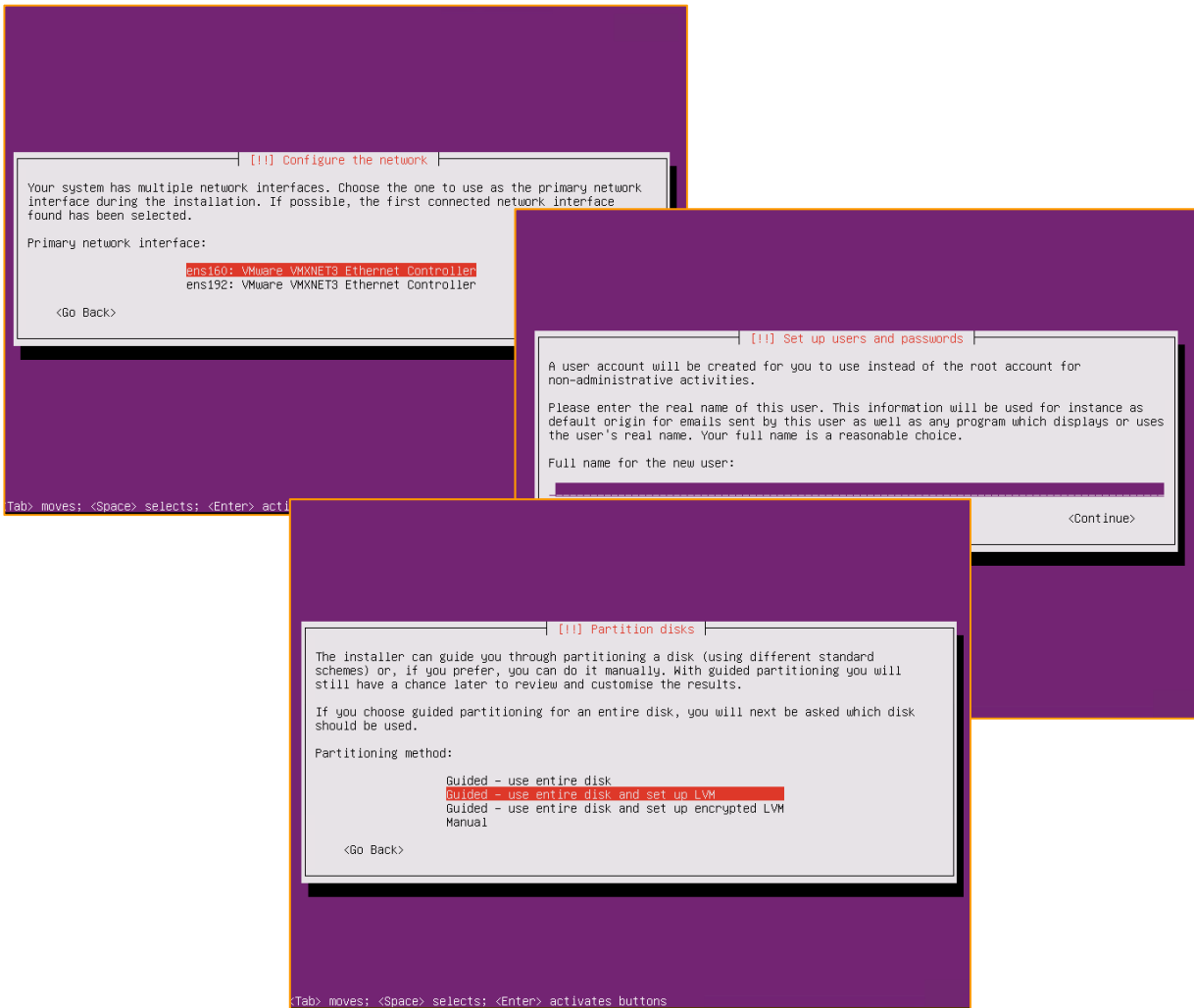
메모:

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

3. Host (Ubuntu)

❖ Ubuntu Server 16.04 Installation

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



메모:

3. Host (Ubuntu)

❖ Ubuntu Server 16.04 Installation

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



메모:

3. Host (Ubuntu)

❖ Ubuntu Server 16.04 Installation

- ① `sudo apt install lm-sensors` # sensors for Bare metal
- ② **ip link show** # Check Interfaces
- ③ **Static IP Address Setting**
- ④ **Host Name Setting**

- 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
```

- 고정 IP 주소 설정 -

```
sudo vi /etc/network/interfaces  
  
# Iface ens160 inet dhcp  
iface ens160 inet static  
    address 192.168.0.xx  
    netmask 255.255.255.0  
    gateway 192.168.0.1  
    dns-nameservers 8.8.8.8  
  
cntl+o → enter → cntl+x  
sudo /etc/init.d/networking restart (or reboot)
```

- Root 계정 생성 -

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

- 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'
- Root 계정으로 실행 필요시 (sudo 사용 일반 계정은 실행하지 못함)
루트계정 활성화: `sudo passwd root`

목차



1. 실습 환경
2. vRouter (VyOS..)
3. Host (Ubuntu..)
- 4. vSwitch (OVS..)**
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
10. **Container Networking** (Docker..)

4. vSwitch (OVS)

❖ Open vSwitch Installation (스위치 2개 예)

- ① **sudo apt install -y openvswitch-switch**
- ② **sudo ovs-vsctl show**
- ③ **ps -el | grep ovs**
- ④ **sudo ovs-vsctl add-br ovs1xx # ubuntu1**
- ⑤ **sudo ovs-vsctl show**
- ⑥ **sudo ovs-vsctl add-br ovs2xx # ubuntu2**
- ⑦ **show ovs-vsctl show**

```
james@ubuntu18:~$ sudo ovs-vsctl show
[sudo] password for james:
51da28cb-d5cf-4699-8a28-5045a5960295
  ovs_version: "2.9.0"
james@ubuntu18:~$ ps -el | grep ovs
5 S      0  2879    1  0  70 -10 - 5204 -    ?        00:00:00 ovsdb-server
5 S      0  2927    1  0  70 -10 - 6618 -    ?        00:00:00 ovs-vswitchd
```

```
james@ubuntu18:~$ sudo ovs-vsctl add-br ovs1qotom
james@ubuntu18:~$ sudo ovs-vsctl show
51da28cb-d5cf-4699-8a28-5045a5960295
  Bridge "ovs1qotom"
    Port "ovs1qotom"
      Interface "ovs1qotom"
        type: internal
  ovs_version: "2.9.0"
```

메모:

- sudo ovs-vsctl add-port ovs1xx patch-ovs1
- sudo ovs-vsctl add-port ovs2xx patch-ovs2
- ps -ef | grep onos

4. vSwitch (OVS)

❖ Open vSwitch Installation

- ① **sudo ovs-dpctl show**
- ② **sudo ovs-vsctl show**

```
james@ubuntu:~$ sudo ovs-dpctl show
system@ovs-system:
  lookups: hit:0 missed:0 lost:0
  flows: 0
  masks: hit:0 total:0 hit/pkt:0.00
  port 0: ovs-system (internal)
  port 1: enp2s0
  port 2: enp1s0
  port 3: enp3s0
  port 4: ovs2qotom (internal)
```

- 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 ovs1xx patch-ovs1
- sudo ovs-vsctl add-port ovs2xx patch-ovs2

4. vSwitch (OVS)

❖ Open vSwitch Installation

① Ifconfig

② `sudo ovs-vsctl add-port ovs1xx enp1s0`

```
james@ubuntu18:~$ ifconfig
enp1s0: flags=4099<UP, BROADCAST, MULTICAST> mtu 1500
ether 00:aa:2a:e8:34:20 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 16 memory 0xd0900000-d0920000

enp2s0: flags=4099<UP, BROADCAST, MULTICAST> mtu 1500
ether 00:aa:2a:e8:34:21 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 17 memory 0xd0800000-d0820000

enp3s0: flags=4099<UP, BROADCAST, MULTICAST> mtu 1500
ether 00:aa:2a:e8:34:22 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 18 memory 0xd0700000-d0720000

enp4s0: flags=4099<UP, BROADCAST, MULTICAST> mtu 1500
ether 00:aa:2a:e8:34:23 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 19 memory 0xd0600000-d0620000

lo: flags=73<UP, LOOPBACK, RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scope 0
loop txqueuelen 1000 (Local Loopback)
RX packets 259 bytes 19459 (19.4 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 259 bytes 19459 (19.4 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlx742f68923076: flags=4163<UP, BROADCAST, MULTICAST, RUNNING> mtu 1500
inet 192.168.0.208 netmask 255.255.255.0
inet6 fe80::9d60:5aa3:68:8705 prefixlen 64 scope 0
ether 74:2f:68:92:30:76 txqueuelen 1000 (Ethernet)
RX packets 212953 bytes 2900933 (2.9 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 103469 bytes 9175405 (9.1 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

james@ubuntu18:~$ sudo ovs-vsctl show
Bridge "ovs1qotom"
  Port "enp2s0"
    Interface "enp2s0"
  Port "ovs1qotom"
    Interface "ovs1qotom"
    type: internal
  Port "enp3s0"
    Interface "enp3s0"
  Port "patch-ovs1"
    Interface "patch-ovs1"
    error: "could not open network device patch-ovs1 (No such device)"
  Port "enp1s0"
    Interface "enp1s0"
  ovs_version: "2.9.0"

james@ubuntu18:~$ sudo ovs-vsctl add-port ovs1qotom enp1s0
```

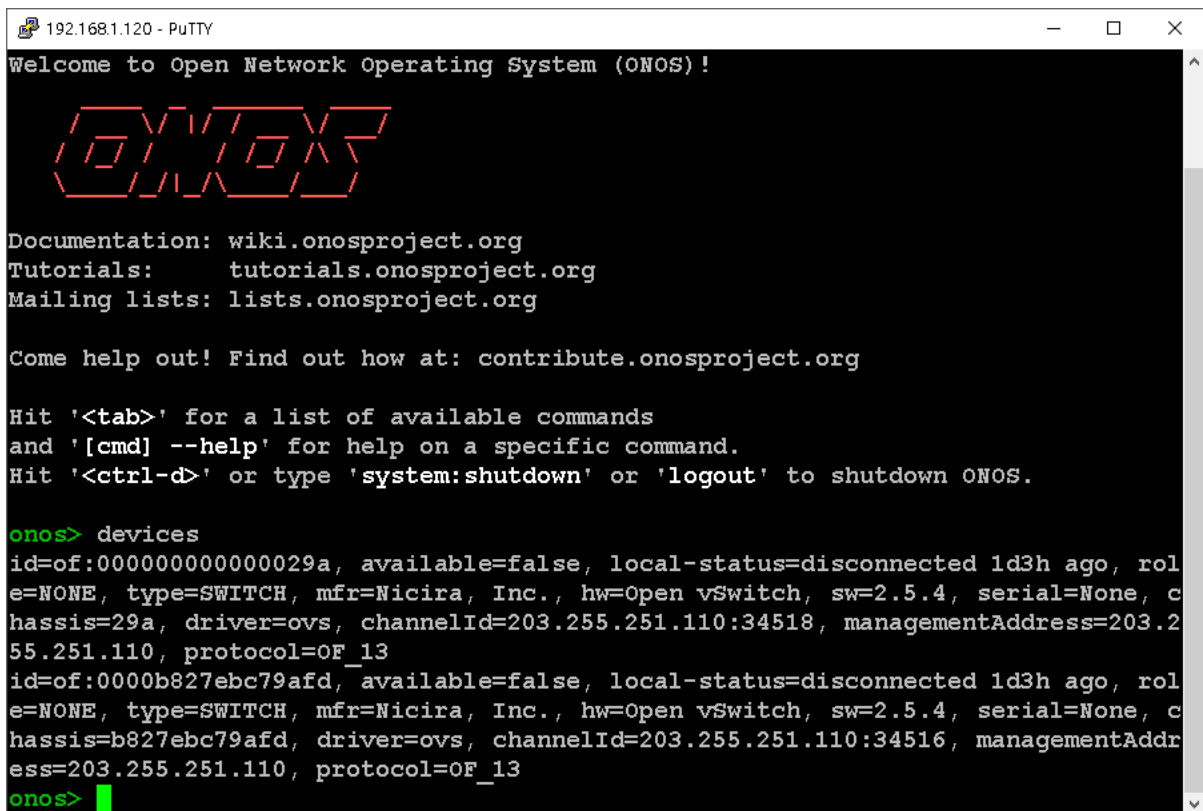
메모:

- `sudo ovs-vsctl add-port ovs1xx enp2s0`
- `sudo ovs-vsctl add-port ovs1xx enp3s0`
- `sudo ovs-vsctl add-port ovs2xx enp1s0`
- `sudo ovs-vsctl add-port ovs2xx enp2s0`
- `sudo ovs-vsctl add-port ovs2xx enp3s0`

4. vSwitch (OVS)

❖ Open vSwitch Installation (스위치 2개 예)

- ① `sudo ovs-vsctl set-controller ovs1xx tcp:203.255.252.51:6653`
- ② `sudo ovs-vsctl set-controller ovs2xx tcp:203.255.252.51:6653`
- ③ <http://203.255.252.xx:8181/onos/ui> # onos / rocks
- ④ `ssh james@203.255.252.xx:8101`



```
192.168.1.120 - PuTTY
Welcome to Open Network Operating System (ONOS)!

  ONOS

Documentation: wiki.onosproject.org
Tutorials:    tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown ONOS.

onos> devices
id=of:000000000000029a, available=false, local-status=disconnected 1d3h ago, role=NONE, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.5.4, serial=None, chassis=29a, driver=ovs, channelId=203.255.251.110:34518, managementAddress=203.255.251.110, protocol=OF_13
id=of:0000b827ebc79afd, available=false, local-status=disconnected 1d3h ago, role=NONE, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.5.4, serial=None, chassis=b827ebc79afd, driver=ovs, channelId=203.255.251.110:34516, managementAddress=203.255.251.110, protocol=OF_13
onos>
```

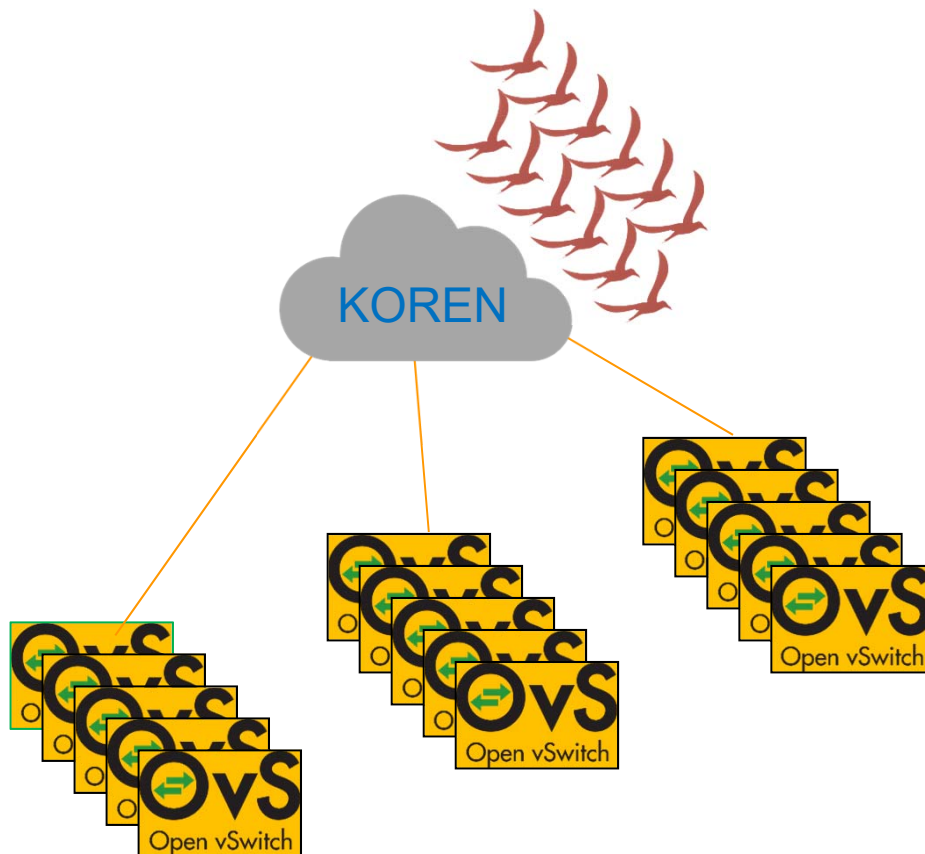
메모:

- `sudo docker run -t -d -p 1181:8181 -p 1101:8101 -p 1653:6653 --name onos1 onosproject/onos`
- `sudo docker run -t -d -p 2181:8181 -p 2101:8101 -p 2653:6653 --name onos2 onosproject/onos`

4. vSwitch (OVS)

❖ Open vSwitch Installation

- ① 개인별 제공 원격 ONOS SDN 제어기 연결
- ② 개인 스위치별 연결 주소 제공



메모:

4. vSwitch (OVS)

❖ Open vSwitch Installation

- ① <http://203.255.252.xx:8181/onos/ui> # onos / rocks
- ② `ssh james@203.255.252.xx:8101` # CLI for ONOS

		Port #8181 Dashboard	Port #8101 ONOS CLI	Port #6653 ONOS
1	203.255.252.51	1181	1101	1653
2	203.255.252.51	2181	2101	2653
3	203.255.252.51	3181	3101	3653
4	203.255.252.51	4181	4101	4653
5	203.255.252.51	5181	5101	5653
6	203.255.252.51	6181	6101	6653
7	203.255.252.51	7181	7101	7653
8	203.255.252.51	8181	8101	8653
9	203.255.252.51	9181	9101	9653
10	203.255.252.51	10181	10101	10653
11	203.255.252.52	1181	1101	1653
12	203.255.252.52	2181	2101	2653
13	203.255.252.52	3181	3101	3653
14	203.255.252.52	4181	4101	4653
15	203.255.252.52	5181	5101	5653

메모:

- `sudo ovs-ofctl dump-flows ovs1xx`
- `sudo ovs-ofctl show ovs1xx`
- `sudo ovs-ofctl dump-flows ovs2xx`
- `sudo ovs-ofctl show ovs2xx`

4. vSwitch (OVS)

❖ Open vSwitch Installation (원격 개인별 ONOS 설치 예)

- ① **sudo docker run -t -d -p 1181:8181 -p 1101:8101 -p 1653:6653 --name onos1 onosproject/onos # student 1**
- ② **sudo docker run -t -d -p 2181:8181 -p 2101:8101 -p 2653:6653 --name onos2 onosproject/onos # student 2**
- ③ **sudo docker run -t -d -p 3181:8181 -p 3101:8101 -p 3653:6653 --name onos3 onosproject/onos # student 3**
- ④ **sudo docker run -t -d -p 4181:8181 -p 4101:8101 -p 4653:6653 --name onos4 onosproject/onos # student 4**
- ⑤ **sudo docker run -t -d -p 5181:8181 -p 5101:8101 -p 5653:6653 --name onos5 onosproject/onos # student 5**
- ⑥ **sudo docker run -t -d -p 6181:8181 -p 6101:8101 -p 6653:6653 --name onos6 onosproject/onos # student 6**
- ⑦ **sudo docker run -t -d -p 7181:8181 -p 7101:8101 -p 7653:6653 --name onos7 onosproject/onos # student 7**
- ⑧ **sudo docker run -t -d -p 8181:8181 -p 8101:8101 -p 8653:6653 --name onos8 onosproject/onos # student 8**
- ⑨ **sudo docker run -t -d -p 9181:8181 -p 9101:8101 -p 9653:6653 --name onos9 onosproject/onos # student 9**
- ⑩ **sudo docker run -t -d -p 10181:8181 -p 10101:8101 -p 10653:6653 --name onos10 onosproject/onos # student 10**

메모:

- sudo systemctl stop ntopng
- sudo ntopng
- sudo docker start onosxx

4. vSwitch (OVS)

❖ Open vSwitch Installation

- ① **sudo ovs-vsctl set-controller ovs1xx tcp:203.255.252.51:10653**
- ② **sudo ovs-vsctl set-controller ovs2xx tcp:203.255.252.51:10653**
- ③ <http://203.255.252.51:8181/onos/ui> # onos / rocks

```
james@ubuntu18:~$ sudo ovs-ofctl show ovs1qotom
OFPT_FEATURES_REPLY (xid=0x2): dpid:000000aa2ae83420
n_tables:254, n_buffers:0
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
1(enp1s0): addr:00:aa:2a:e8:34:20
  config: 0
  state: LINK_DOWN
  current: COPPER AUTO_NEG
  advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
  supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
  speed: 0 Mbps now, 1000 Mbps max
2(enp2s0): addr:00:aa:2a:e8:34:21
  config: 0
  state: LINK_DOWN
  current: COPPER AUTO_NEG
  advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
  supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
  speed: 0 Mbps now, 1000 Mbps max
3(enp3s0): addr:00:aa:2a:e8:34:22
  config: 0
  state: LINK_DOWN
  current: COPPER AUTO_NEG
  advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
  supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
  speed: 0 Mbps now, 1000 Mbps max
LOCAL(ovs1qotom): addr:00:aa:2a:e8:34:20
  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
james@ubuntu18:~$
```

메모:

- sudo ovs-ofctl dump-flows ovs1qotom
- sudo ovs-ofctl show ovs1qotom
- sudo ovs-ofctl dump-flows ovs2qotom
- sudo ovs-ofctl show ovs2qotom

목차



1. 실습 환경
2. vRouter (VyOS..)
3. Host (Ubuntu..)
4. vSwitch (OVS..)
- 5. 컨테이너 (Docker..)**
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
10. Container Networking (Docker..)

5. 컨테이너 (Docker)

❖ Prerequisites @ Ubuntu 16.04

① **useradd jslab**

② **sudo visudo**

```
# User privilege specification
root ALL=(ALL:ALL) ALL
jslab ALL=(ALL:ALL) ALL
....
```

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

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

⑤ **sudo usermod -aG docker jslab**

⑥ **sudo docker version**

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

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

메모:

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

5. 컨테이너 (Docker)

❖ docker info

```
james@ubuntu-server:~$ sudo docker info
Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
Images: 0
Server Version: 18.05.0-ce
Storage Driver: aufs
  Root Dir: /var/lib/docker/aufs
  Backing Filesystem: extfs
  Dirs: 0
  Dirperm1 Supported: 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: 773c489c9c1b21a6d78b5c538cd395416ec50f88
runc version: 4fc53a81fb7c994640722ac585fa9ca548971871
init version: 949e6fa
Security Options:
  apparmor
  seccomp
    Profile: default
Kernel Version: 4.4.0-116-generic
Operating System: Ubuntu 16.04.4 LTS
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 3.859GiB
Name: ubuntu-server
ID: FY0C:6LZN:0IXJ:4YPQ:3RMB:MHMW:5MLW:2ZUD:ORS5:FSGH:14JH:G61Q
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
james@ubuntu-server:~$
```

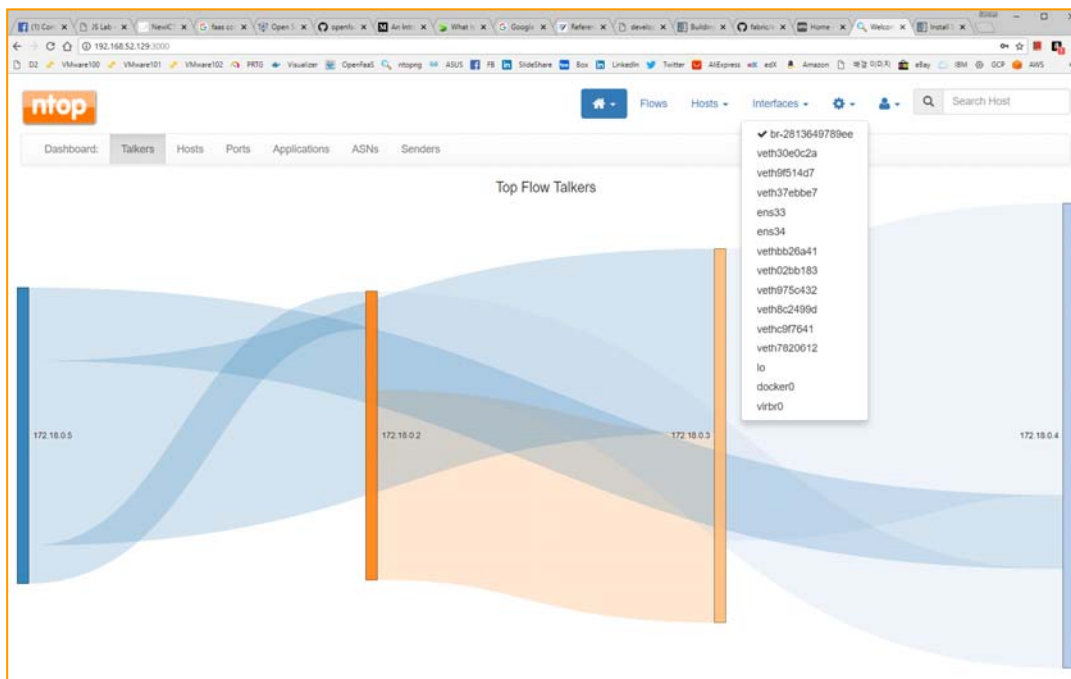
메모:

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

5. 컨테이너 (Docker)

❖ ntopng @ Ubuntu for flow monitoring

- ① **sudo apt install ntopng** # sudo ntopng
- ② **sudo systemctl enable ntopng**
- ③ **http://192.168.0.xx:3000** # admin / admin → password
- ④ **http://192.168.0.xx:3000** # admin / admin → password
- ⑤ **sudo apt install docker.io** # Optional for 1.13.1 (May 2018)
- ⑥ **sudo apt install curl**
- ⑦ **sudo curl -fsSL https://get.docker.com/ | sh** # latest
- ⑧ **sudo usermod -aG docker james**
- ⑨ **sudo docker version**



메모:

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

5. 컨테이너 (Docker)

❖ ntopng @ Ubuntu for flow monitoring

- ① All Hosts
- ② Active Flows

The screenshot shows the ntopng web interface with the 'All Hosts' view selected. The table lists various IP addresses, their locations (all 'Local'), alerts (all 0), names, and seen since times. The 'Breakdown' column shows 'Sent' and 'Rcvd' buttons for each host. The 'Throughput' and 'Traffic' columns show current and total data flow.

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

Local Hosts Active Flows Matrix

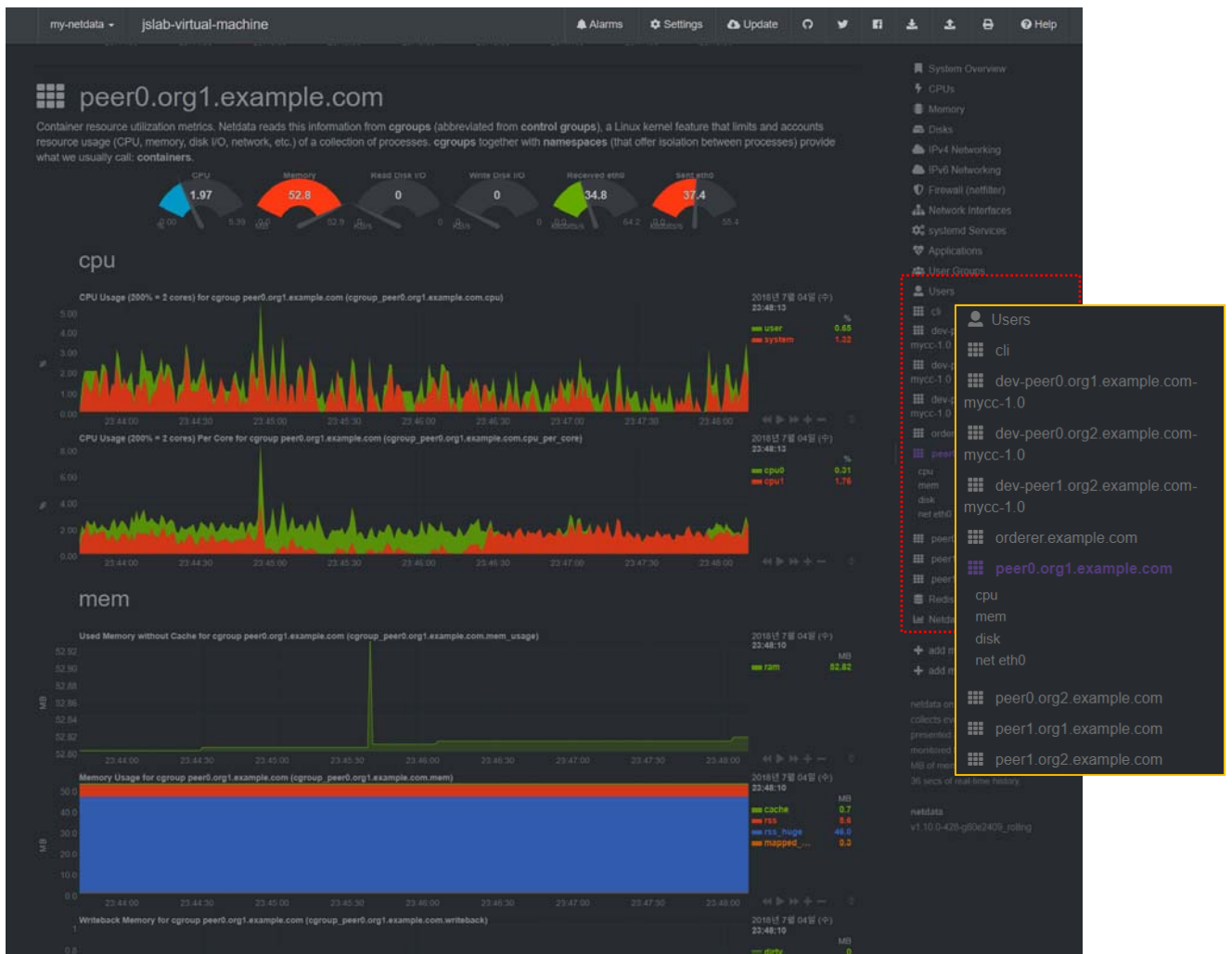
	172.18.0.3	172.18.0.4	172.18.0.6	172.18.0.5	172.18.0.8	172.18.0.2	172.18.0.10
172.18.0.3		497.3 KB 504.39 KB		437.58 KB 438.14 KB	112 B 178 B	434.39 KB 442.63 KB	
172.18.0.4	504.39 KB 497.3 KB		178 B 112 B	448.29 KB 437.77 KB		444.98 KB 442.93 KB	112 B 178 B
172.18.0.6		112 B 178 B		112 B 178 B			
172.18.0.5	438.14 KB 437.58 KB	437.77 KB 448.29 KB	178 B 112 B			499.65 KB 498.83 KB	
172.18.0.8	178 B 112 B						
172.18.0.2	442.83 KB 434.39 KB	442.93 KB 444.98 KB		498.83 KB 499.65 KB			
172.18.0.10		178 B 112 B					

메모:

5. 컨테이너 (Docker)

❖ netdata @ Ubuntu for resource monitoring

- ① **bash** `<(curl -Ss https://my-netdata.io/kickstart.sh)`
- ② **http://192.168.0.208:19999/** # http://192.168.0.214:19999/



메모:

5. 컨테이너 (Docker)

❖ 설치/실행 (예: Ubuntu @ www.docker.com)

- ① **sudo apt update**
- ② **sudo apt install -y apt-transport-https ca-certificates software-properties-common curl**
- ③ **curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**
- ④ **sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable"**
- ⑤ **sudo apt update**
- ⑥ **sudo apt install -y docker-ce**
- ⑦ **sudo usermod -aG docker userID**
- ⑧ **sudo systemctl restart ttyd**
- ⑨ **exit**

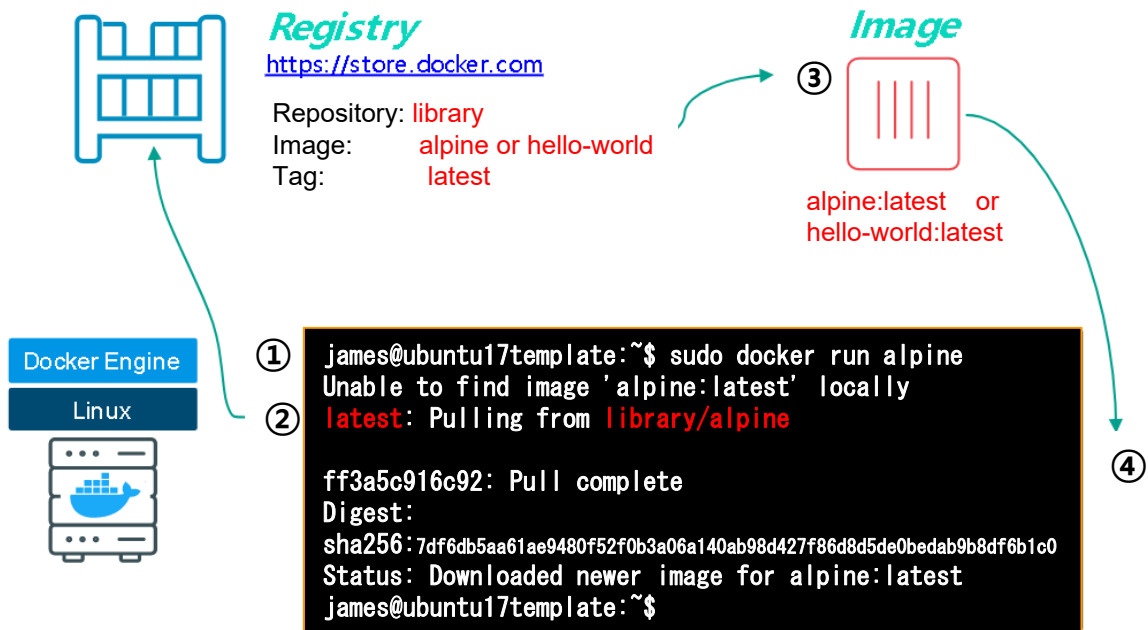
메모:

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

5. 컨테이너 (Docker)

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

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



메모:

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

5. 컨테이너 (Docker)

❖ Alpine Linux 컨테이너 @ Ubuntu (선택)

- ① `curl -fsSL https://get.docker.com/ | sh` # @ General
- ② `sudo docker image pull alpine`
- ③ `sudo docker image ls`
- ④ `sudo docker container run alpine ls -l`
- ⑤ `sudo docker container run alpine echo "hello from alpine"`

- ⑥ `sudo docker container run alpine /bin/sh`
- ⑦ `sudo docker container run -it alpine /bin/sh` # shell prompt
- ⑧ `/ # ls -l` # @ Alpine
- ⑨ `/ # uname -a` # @ Alpine
- ⑩ `/ # exit` # @ Alpine

- ⑪ `sudo docker container run -it alpine /bin/sh` # @ Alpine
- ⑫ `/ # echo "hello world" > hello.txt` # @ Alpine
- ⑬ `/ # ls` # @ Alpine
- ⑭ `/ # exit` # @ Alpine
- ⑮ `sudo docker container ls -a`

Question: hello.txt 생성 파일 확인 (선택: 반복 ⑪ ~ ⑭)

메모:

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

5. 컨테이너 (Docker)

❖ ghost

- ① **sudo docker run --name ghost1 -d ghost**
컨테이너 'ghost'는 포트 미지정시 기정포트(Default Port) 2368로 시작
- ② **sudo docker run --name ghost2 -p 8080:2368 -d ghost**
http://localhost:8080 or http://host-ip:8080 접속 가능 컨테이너
- ③ **sudo docker run --name ghost3 -v /path/to/ghost/blog:/var/lib/ghost ghost**
사용 호스트의 콘텐츠를 이미지에 지정하여 사용하는 컨테이너
데이터 컨테이너 'var/lib/ghost'로 대체하여 사용하는 컨테이너
- ④ **sudo docker run --name ghost4 --volumes-from some-ghost-data ghost**
데이터 컨테이너 'var/lib/ghost'로 대체하여 사용하는 컨테이너

메모:

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

5. 컨테이너 (Docker)

❖ 요약 (Basic commands)

- ① **docker images** # 현재 사용 가능한 image 목록을 출력. -a 옵션을 주면 모든 것을 보여줌
- ② **docker ps** # 현재 사용 가능한 컨테이너 목록을 출력. -a 옵션을 주면 모든 것을 보여줌
- ③ **docker pull <아이디>/<이미지 이름>:<태그>** # docker hub 이미지 가지고 올
- ④ **docker run -it <아이디>/<이미지 이름>:<태그> /bin/bash**
-it 실행한 명령이 Console에 붙어서 진행. i는 interactive, t는 tty를 의미
- ⑤ **docker container run <container 이름> ls -l**
ls -l 명령어를 실행하며 컨테이너를 실행
- ⑥ **docker container run -it --name <container 별명> <image 이름> /bin/ash**
--name# # 통해 container 이름 부여. container 이름을 부여하지 않으면 랜덤하게 생성
- ⑦ **docker container start <container ID>**
docker container에서 명령어 실행
- ⑧ **docker container exec <container ID> ls**
exec은 container에서 명령어를 실행
- ⑨ **docker diff <container 별명>**
컨테이너가 부모 이미지와 파일 변경 사항을 확인할 수 있는 명령어
- ⑩ **docker commit <container ID> <아이디>/<이미지 이름>:<태그>**
새로운 도커 이미지 생성
- ⑪ **docker push <아이디>/<이미지 이름>:<태그>**
docker hub에 이미지 업로드
- ⑫ **docker build --tag <아이디>/<이미지 이름>:<태그> .**
Dockerfile 생성 위치에서 실행하면 이미지 생성

메모:

5. 컨테이너 (Docker)

❖ 요약 (Basic commands)

① docker

Management Commands:

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

Commands:

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

메모:

목차



1. 실습 환경
2. vRouter (VyOS..)
3. Host (Ubuntu..)
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
- 6. 이미지 (Docker Image)**
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
10. Container Networking (Docker..)

6. 이미지 (Docker Image)

❖ 컨테이너에서 이미지 생성 @ Ubuntu (선택)

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

```
james@ubuntu17template:~$ docker container run -ti ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390aic435a: Pull complete
07844b14977e: Pull complete
b78396653dae: Pull complete
Digest: sha256:e348fbb0e0a0e73ab0370de151e7800684445c509d46195aef73e090e49bd6
Status: Downloaded newer image for ubuntu:latest
root@ba625ffe082:/# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:5 http://security.ubuntu.com/ubuntu xenial-security/universe Sources [77.2 kB]
Get:6 http://archive.ubuntu.com/ubuntu xenial/universe Sources [9802 kB]
Get:7 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [593 kB]
Get:8 http://security.ubuntu.com/ubuntu xenial-security/restricted amd64 Packages [12.7 kB]
Get:9 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 Packages [427 kB]
Get:10 http://security.ubuntu.com/ubuntu xenial-security/multiverse amd64 Packages [3492 B]
Get:11 http://archive.ubuntu.com/ubuntu xenial/main amd64 Packages [1558 kB]
Get:12 http://archive.ubuntu.com/ubuntu xenial/restricted amd64 Packages [14.1 kB]
Get:13 http://archive.ubuntu.com/ubuntu xenial/universe amd64 Packages [9827 kB]
Get:14 http://archive.ubuntu.com/ubuntu xenial/multiverse amd64 Packages [176 kB]
Get:15 http://archive.ubuntu.com/ubuntu xenial-updates/universe Sources [250 kB]
Get:16 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [962 kB]
Get:17 http://archive.ubuntu.com/ubuntu xenial-updates/restricted amd64 Packages [13.1 kB]
Get:18 http://archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [792 kB]
Get:19 http://archive.ubuntu.com/ubuntu xenial-updates/multiverse amd64 Packages [18.5 kB]
Get:20 http://archive.ubuntu.com/ubuntu xenial-backports/main amd64 Packages [6153 B]
Get:21 http://archive.ubuntu.com/ubuntu xenial-backports/universe amd64 Packages [7734 B]
Fetched 25.1 MB in 7s (3317 kB/s)
Reading package lists... Done
root@ba625ffe082:/# apt-get install -y figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  figlet
0 upgraded, 1 newly installed, 0 to remove and 9 not upgraded.
Need to get 190 kB of archives.
After this operation, 744 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu xenial/universe amd64 figlet amd64 2.2.5-2 [190 kB]
Fetched 190 kB in 1s (102 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package figlet.
(Reading database ... 4768 files and directories currently installed.)
Preparing to unpack .../figlet_2.2.5-2_amd64.deb ...
Unpacking figlet (2.2.5-2) ...
Setting up figlet (2.2.5-2) ...
update-alternatives: using /usr/bin/figlet-figlet to provide /usr/bin/figlet (figlet) in auto mode
root@ba625ffe082:/# figlet "hello james"

hello james
root@ba625ffe082:/#
```

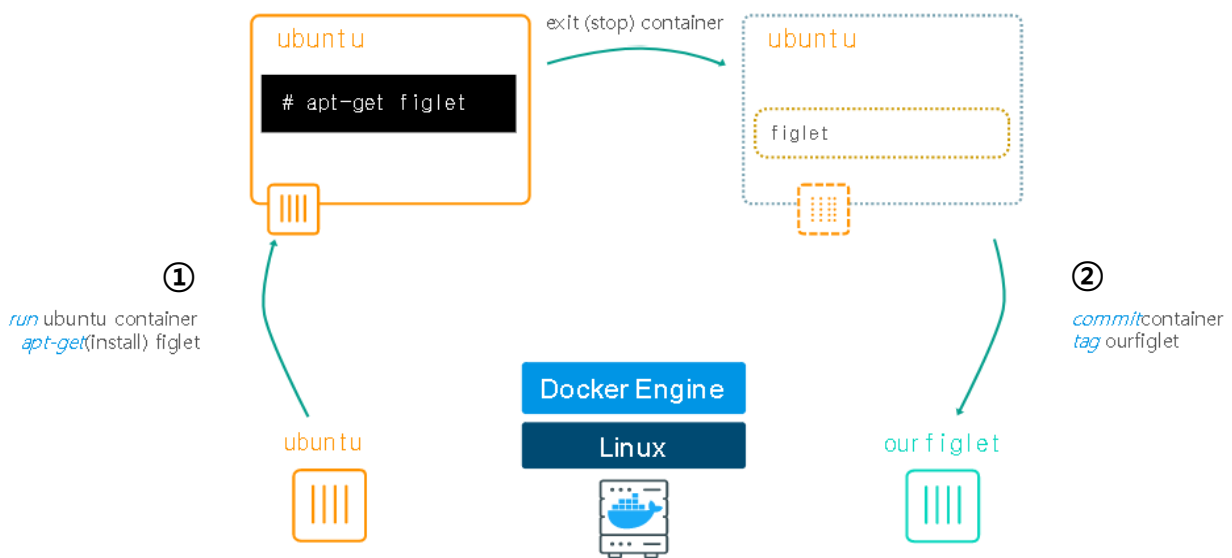
메모:

- 우분투(Ubuntu) 실행후 업데이트와 figlet 설치

6. 이미지 (Docker Image)

❖ 컨테이너에서 이미지 생성 @ Ubuntu (선택)

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



메모:

- It can be useful to commit a container's file changes or settings into a new image.
- Container ID와 Image ID는 다른 것과 겹치지 않는 1 글자 이상 가능

6. 이미지 (Docker Image)

❖ 생성 이미지 확인 @ Ubuntu (선택)

- ① `sudo docker container ls -a`
- ② `sudo docker image ls`
- ③ `sudo docker container commit CONTAINER_ID`
- ④ `sudo docker image ls`
- ⑤ `sudo docker image tag <IMAGE_ID> myfiglet`
- ⑥ `sudo docker image ls`
- ⑦ `sudo docker container run myfiglet figlet hello james`

```
james@ubuntu17template:~$ docker container ls -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
ba625ffee082   ubuntu   "bash"                  13 minutes ago Exited (2) 6 minutes ago
musing_colden

james@ubuntu17template:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest   f975c5035748   3 weeks ago   112MB

james@ubuntu17template:~$ docker container commit ba
sha256:4555e45525c1a53400e41436601b22789ce8cb645c1274eb86fb4e60f2c81742

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

james@ubuntu17template:~$ docker image tag 45 myfiglet
james@ubuntu17template:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
myfiglet      latest   4555e45525c1   4 minutes ago  154MB
ubuntu        latest   f975c5035748   3 weeks ago   112MB

james@ubuntu17template:~$
```

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

james@ubuntu17template:~$
james@ubuntu17template:~$ docker image tag 45 myfiglet
james@ubuntu17template:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
myfiglet      latest   4555e45525c1   4 minutes ago  154MB
ubuntu        latest   f975c5035748   3 weeks ago   112MB

james@ubuntu17template:~$ ^C
```

메모:

- 컨테이너에서 이미지 생성 @ Ubuntu (선택)

6. 이미지 (Docker Image)

❖ 이미지 생성 준비 @ Ubuntu (선택)

① vi index.js

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

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

vi 에디터 명령어 'esc' 후

- `:x` Exit, saving changes
- `:q` Exit as long as there have been no changes
- `ZZ` Exit and save changes if any have been made
- `:q!` Exit and ignore any changes

② vi Dockerfile

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

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

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

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

메모:

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

6. 이미지 (Docker Image)

❖ Dockerfile 명령어

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

메모:

6. 이미지 (Docker Image)

❖ 이미지 생성(Build) @ Ubuntu (선택)

- ① `sudo docker image build -t ubuntu:v0.1 .`
- ② `sudo docker images`

```
james@ubuntu17template:~$ sudo docker image build -t ubuntu:v0.1 .
Sending build context to Docker daemon 2.134MB
Step 1/5 : FROM alpine
latest: Pulling from library/alpine
ff3a5c916c92: Pull complete
Digest: sha256:7df6db5aa61ae9480f52f0b3a06a140ab98d427f86d8d5de0bedab9b8df6b1c0
Status: Downloaded newer image for alpine:latest
   -> 3fd9065eaf02
Step 2/5 : RUN apk update && apk add nodejs
   -> Running in 37fffa95be62
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz
v3.7.0-141-gd4baade662 [http://dl-cdn.alpinelinux.org/alpine/v3.7/main]
v3.7.0-141-gd4baade662 [http://dl-cdn.alpinelinux.org/alpine/v3.7/community]
OK: 9051 distinct packages available
(1/10) Installing ca-certificates (20171114-r0)
(2/10) Installing nodejs-npm (8.9.3-r1)
(3/10) Installing c-ares (1.13.0-r0)
(4/10) Installing libcrypto1.0 (1.0.2o-r0)
(5/10) Installing libgcc (6.4.0-r5)
(6/10) Installing http-parser (2.7.1-r1)
(7/10) Installing libssl1.0 (1.0.2o-r0)
(8/10) Installing libstdc++ (6.4.0-r5)
(9/10) Installing libuv (1.17.0-r0)
(10/10) Installing nodejs (8.9.3-r1)
Executing busybox-1.27.2-r7.trigger
Executing ca-certificates-20171114-r0.trigger
OK: 61 MiB in 21 packages
Removing intermediate container 37fffa95be62
   -> d8b0d85a540e
Step 3/5 : COPY . /app
   -> 978539efbb2b
Step 4/5 : WORKDIR /app
Removing intermediate container 91f6d535586e
   -> 85cce7cc18b8
Step 5/5 : CMD ["node", "index.js"]
   -> Running in c5de0ca210c2
Removing intermediate container c5de0ca210c2
   -> d45df6a1c291
Successfully built d45df6a1c291
Successfully tagged ubuntu:v0.1
james@ubuntu17template:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              v0.1               d45df6a1c291       12 minutes ago    52.3MB
myfiglet            latest             4555e45525c1       About an hour ago 154MB
ubuntu              latest             f975c5035748       3 weeks ago       112MB
alpine              latest             3fd9065eaf02       2 months ago      4.15MB
james@ubuntu17template:~$
```

메모:

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

6. 이미지 (Docker Image)

❖ 이미지 생성(Build) @ Ubuntu (선택)

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

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

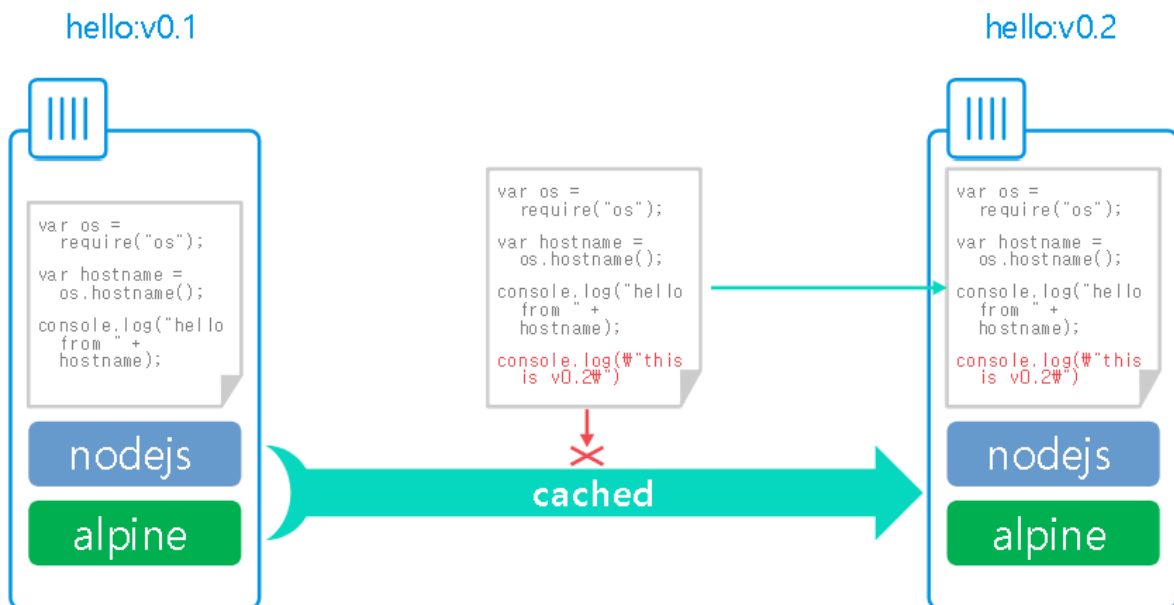
메모:

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

6. 이미지 (Docker Image)

❖ Image layers @ Ubuntu (선택)

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



메모:

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

6. 이미지 (Docker Image)

❖ Image inspect @ Ubuntu (선택)

- ① `sudo docker image pull alpine.`
- ② `sudo docker image inspect alpine`
- ③ `sudo docker image inspect --format "{{ json .RootFS.Layers }}" alpine`
- ④ `sudo docker image ls`
- ⑤ `sudo docker image inspect --format "{{ json .RootFS.Layers }}" <image ID>`

```
james@ubuntu17template:~$ docker image inspect --format "{{ json .RootFS.Layers }}" alpine
[{"sha256:cd7100a72410606589a54b932cabd804a17f9ae5b42a1882bd56d263e02b6215"}]
james@ubuntu17template:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello                v0.1               d45df6a1c291       36 minutes ago     52.3MB
ubuntu              v0.1               d45df6a1c291       36 minutes ago     52.3MB
myfiglet            latest             4555e45525c1       About an hour ago  154MB
ubuntu              latest             f975c5035748       3 weeks ago        112MB
alpine               latest             3fd9065eaf02       2 months ago       4.15MB
james@ubuntu17template:~$ docker image inspect --format "{{ json .RootFS.Layers }}" hello
Error: No such image: hello
james@ubuntu17template:~$ docker image inspect --format "{{ json .RootFS.Layers }}" hello:v0.1
[{"sha256:cd7100a72410606589a54b932cabd804a17f9ae5b42a1882bd56d263e02b6215"}, {"sha256:15975d6f3f707757bbbd49500c5b0b63b36aa92e11c35f7ff92f5ce0019981dd"}, {"sha256:371e14427d436b2a2e9c9b7c87227a22df2b39d3b46b61c13d10d2a71f382bb3"}]
```

메모:

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

목차



1. 실습 환경
2. vRouter (VyOS..)
3. Host (Ubuntu..)
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
- 7. 스웸 (Swarm)**
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
10. Container Networking (Docker..)

7. 스웜 (Swarm)

❖ Swarm Mode

- ① **sudo docker swarm init --advertise-addr \$(hostname -i)**
- ② **docker swarm join --token SWMTKN-1-133f2nioom30v47dr4c8j8q4uq5hhp3gn7su5tazj1a2oczomg-84iw7bynjt7f0qhy98u2mcou9 127.0.1.1:2377**
- ③ `git clone https://github.com/docker/example-voting-app`
- ④ `cd example-voting-app`
- ⑤ `cat docker-stack.yml`

```
james@ubuntu17template:~$ docker swarm init --advertise-addr $(hostname -i)
Swarm initialized: current node (9r7jspmooi98x7ubblc282jtq) 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@ubuntu17template:~$ sudo apt install git
```

```
james@ubuntu17template:~$ git clone https://github.com/docker/example-voting-app
Cloning into 'example-voting-app'...
remote: Counting objects: 482, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 482 (delta 4), reused 11 (delta 4), pack-reused 468
Receiving objects: 100% (482/482), 230.04 KiB | 555.00 KiB/s, done.
Resolving deltas: 100% (177/177), done.
```

```
james@ubuntu17template:~$ cd example-voting-app/
```

```
james@ubuntu17template:~/example-voting-app$ dir
architecture.png  docker-compose-javaworker.yml  docker-compose.yml  k8s-specifications  MAINTAINERS  result  worker
dockercloud.yml  docker-compose-simple.yml      docker-stack.yml    LICENSE              README.md    vote
```

```
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
- CentOS와 Ubuntu가 동일한 Docker Swarm 모드 명령어 사용
- 호스트에 복수 interface 시 `sudo docker swarm init --advertise-addr 192.168.0.xx` 지정

7. 스웜 (Swarm)

❖ 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

목차



1. 실습 환경
2. vRouter (VyOS..)
3. Host (Ubuntu..)
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
- 8. 스택과 서비스 (Stack/Service)**
9. FaaS (Open Function as a Service)
10. Container Networking (Docker..)

8. 스택과 서비스 (Stack/Service)

❖ stack 파일

⑤ cat docker-stack.yml

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

  redis:
    image: redis:alpine
    ports:
      - "6379"
    networks:
      - frontend
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
      delay: 10s
      restart_policy:
        condition: on-failure

  db:
    image: postgres:9.4
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - backend
    deploy:
      placement:
        constraints: [node.role == manager]

  vote:
    image: dockersamples/examplevotingapp_vote:before
    ports:
      - 5000:80
    networks:
      - frontend
    depends_on:
      - redis
    deploy:
      replicas: 2
      update_config:
        parallelism: 2
      restart_policy:
        condition: on-failure

  result:
    image: dockersamples/examplevotingapp_result:before
    ports:
      - 5001:80
    networks:
      - backend
    depends_on:
      - db

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

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

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

networks:
  frontend:
  backend:

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

메모:

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

8. 스택과 서비스 (Stack/Service)

❖ stack 실행

- ① **sudo docker ps** # check @ each host
- ② **sudo docker stack deploy --compose-file=docker-stack.yml voting_stack** # @ /example-voting-app
- ③ **sudo docker stack ls**
- ④ **sudo docker stack services voting_stack**
- ⑤ # <http://192.168.0.70:8080> for Visualizer @ Chrome
- ⑥ # <http://192.168.0.70:5000> for Vote
- ⑦ # <http://192.168.0.60:5001> for Result

```
james@ubuntu17template:~/example-voting-app$ docker stack deploy --compose-file=docker-stack.yml voting_stack
Creating network voting_stack_backend
Creating network voting_stack_frontend
Creating network voting_stack_default
Creating service voting_stack_worker
Creating service voting_stack_visualizer
Creating service voting_stack_redis
Creating service voting_stack_db
Creating service voting_stack_vote
Creating service voting_stack_result
james@ubuntu17template:~/example-voting-app$ docker stack ls
NAME                SERVICES
voting_stack        6
james@ubuntu17template:~/example-voting-app$ docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED             STATUS
PORTS              NAMES
fe58543baac4      dockersamples/examplevotingapp_result:before "node server.js"       53 seconds ago     Up 37 seconds
80/tcp
25e44cd8c1e6      postgres:9.4                             "docker-entrypoint.s..." About a minute ago Up 58 seconds
5432/tcp
691df72c91e6      dockersamples/examplevotingapp_vote:before "gunicorn app:app -b..." About a minute ago Up 57 seconds
80/tcp
483991c28cac      dockersamples/examplevotingapp_vote:before "gunicorn app:app -b..." About a minute ago Up 58 seconds
80/tcp
b71c0e3de445      dockersamples/examplevotingapp_worker:latest "/bin/sh -c 'dotnet ..." About a minute ago Up About a minute
voting_stack_worker.1.299ec7wqx8plcd2tvrjyelbrm
45ddb99341f      dockersamples/visualizer:stable         "npm start"            About a minute ago Up About a minute
8080/tcp
9ef47eedebbb      redis:alpine                             "docker-entrypoint.s..." About a minute ago Up About a minute
6379/tcp
voting_stack_redis.1.0gwixzc0z7wmd437udymaikl6
james@ubuntu17template:~/example-voting-app$
```

메모:

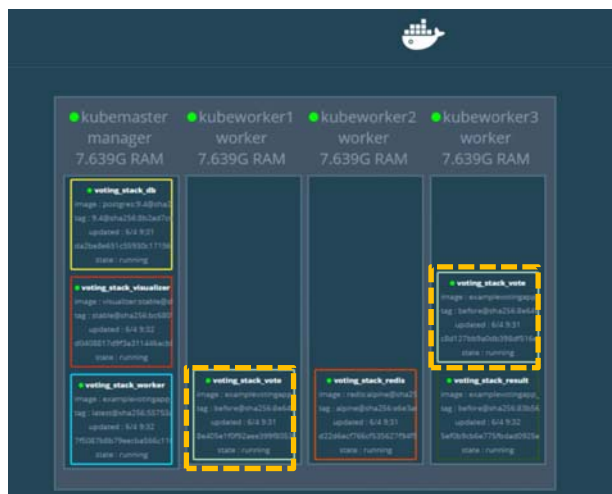
- `git clone https://github.com/docker/example-voting-app`
- `cd example-voting-app`
- `watch -n x <your command>`
- `watch -n 60 ls -l ~/Desktop`
- Check 'immutable infrastructure'

8. 스택과 서비스 (Stack/Service)

❖ stack Operations

- ① `sudo docker stack services voting_stack`
- ② `sudo docker service ps voting_stack_vote`

```
james@ubuntu17template:~/example-voting-app$ docker stack services voting_stack
ID                NAME                MODE                REPLICAS            IMAGE
PORTS
1fa0bp9x0a8y     voting_stack_vote    replicated          2/2                 dockersamples/examplevotingapp_vote:before
*:5000->80/tcp
1sm84ozd14vv     voting_stack_db      replicated          1/1                 postgres:9.4
ds790f0fcoxj     voting_stack_worker  replicated          1/1                 dockersamples/examplevotingapp_worker:latest
hqtmv6pgrlmw     voting_stack_result  replicated          1/1                 dockersamples/examplevotingapp_result:before
*:5001->80/tcp
sythupfy4m2i     voting_stack_visualizer replicated          1/1                 dockersamples/visualizer:stable
*:8080->8080/tcp
t0bcnmhrq4n6     voting_stack_redis   replicated          1/1                 redis:alpine
*:30000->6379/tcp
james@ubuntu17template:~/example-voting-app$
james@ubuntu17template:~/example-voting-app$ docker service ps voting_stack_vote
ID                NAME                IMAGE                NODE                DESIRED STATE        CURRENT
STATE            ERROR              PORTS
uk16gr193w3w     voting_stack_vote.1 dockersamples/examplevotingapp_vote:before ubuntu17template    Running              Running
20 minutes ago
xvly71tow6ac     voting_stack_vote.2 dockersamples/examplevotingapp_vote:before ubuntu17template    Running              Running
20 minutes ago
james@ubuntu17template:~/example-voting-app$
```



메모:

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

8. 스택과 서비스 (Stack/Service)

❖ scale

- ① `sudo docker service scale voting_stack_vote=5`
- ② `sudo docker stack services voting_stack`

```
[root@kubemaster example-voting-app]# docker service scale voting_stack_vote=5
voting_stack_vote scaled to 5
overall progress: 5 out of 5 tasks
1/5: running
2/5: running
3/5: running
4/5: running
5/5: running
verify: Service converged
[root@kubemaster example-voting-app]# docker stack services voting_stack
```

ID	NAME	MODE	REPLICAS	IMAGE
PORTS				
b0693htjttku	voting_stack_redis	replicated	1/1	redis:alpine
*:30000->6379/tcp				
d5hqegg0ckmq	voting_stack_db	replicated	1/1	postgres:9.4
n6s659sn5bhm	voting_stack_visualizer	replicated	1/1	dockersamples/visualizer:stable
*:8080->8080/tcp				
r1qha3ld9q4c	voting_stack_vote	replicated	5/5	dockersamples/examplevotingapp_vote:before
*:5000->80/tcp				
rhj71cxaysjy	voting_stack_worker	replicated	1/1	dockersamples/examplevotingapp_worker:latest
yk6k6vh0ornz	voting_stack_result	replicated	1/1	dockersamples/examplevotingapp_result:before
*:5001->80/tcp				

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



메모:

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

8. 스택과 서비스 (Stack/Service)

❖ docker network inspect ingress

- ① `sudo docker network ls`
- ② `sudo docker network inspect ingress`

```
[root@kubemaster example-voting-app]# docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
1fe249e36d43	bridge	bridge	local
05191e8b7e19	docker_gwbridge	bridge	local
06322c05f69e	host	host	local
33zsiip6je0ns	ingress	overlay	swarm
ed53abe4e032	none	null	local
7s7p1zaiqi7p	voting_stack_backend	overlay	swarm
n3sss1s7elwl	voting_stack_default	overlay	swarm
oao3jy8bdlzu	voting_stack_frontend	overlay	swarm

```
jslab@ubuntu70:~/example-voting-app$ sudo docker network inspect ingress
```

```
{
  "Name": "ingress",
  "Id": "m2ml5qlmdu3vxrm2kcmfyv78t",
  "Created": "2018-09-11T15:02:59.802809879+09:00",
  "Scope": "swarm",
  "Driver": "overlay",
  "EnableIPv6": false,
  "IPAM": {
    "Driver": "default",
    "Options": null,
    "Config": [
      {
        "Subnet": "10.255.0.0/16",
        "Gateway": "10.255.0.1"
      }
    ]
  },
  "Internal": false,
  "Attachable": false,
  "Ingress": true,
  "ConfigFrom": {
    "Network": ""
  },
  "ConfigOnly": false,
  "Containers": {
    "0020654177d1f8aae57b2dc5b9285cc69a9c3ee1bd35b1054e3945d994deac": {
      "Name": "voting_stack_result.1.fqmr20e2auv979tqbfg392g",
      "EndpointID": "6e3463c299ea9ed644deed9b0fd1275b4eeac470bb8549b8f06b73f17ba9c83",
      "MacAddress": "02:42:0a:ff:00:07",
      "IPv4Address": "10.255.0.7/16",
      "IPv6Address": ""
    },
    "316e68085cff970ff69925dc30e40aa9cf3ec567b198a866aeb3c8b84c19447c": {
      "Name": "voting_stack_redis.1.3wtwueh2y9v0z91zi8bzaidm9",
      "EndpointID": "c4ef905bf3f40791a2ceda7500f5e7159dac4609e91ace6d9630705b03bd74ba",
      "MacAddress": "02:42:0a:ff:00:0b",
      "IPv4Address": "10.255.0.11/16",
      "IPv6Address": ""
    },
    "56d302a0f2406f08883ed21a777dbbbe3e590b0218a329b9eb0fe620d1f60a": {
      "Name": "voting_stack_vote.2.tpfvsw2fyvgog5opeix56jmos",
      "EndpointID": "bc9a7a7e41003e322d40cb94bac8933670661295874367366043943aed7438a2",
      "MacAddress": "02:42:0a:ff:00:05",
      "IPv4Address": "10.255.0.5/16",
      "IPv6Address": ""
    },
    "b5f890af85ee5ad5178aaaf9aadafc9ca72c032642ab797b3c196c429d6c4": {
      "Name": "voting_stack_vote.1.jq3fscjgq87gmbvvo6xhu105",
      "EndpointID": "136e4fb25cbdc42e7d13729671df40621056e18faa4b2c1508a6d2b12d0c606",
      "MacAddress": "02:42:0a:ff:00:04",
      "IPv4Address": "10.255.0.4/16",
      "IPv6Address": ""
    },
    "4a492ce6a0e3bf36339fd9c1c66678cfae216c4bcf362ba2de8f6c30c92fd2": {
      "Name": "voting_stack_visualizer.1.jytdij70lbg448n04k59l7",
      "EndpointID": "21767399a07c1397e81182b1e5f483e43bd9e3d23974a121a1ec427f8e919584",
      "MacAddress": "02:42:0a:ff:00:09",
      "IPv4Address": "10.255.0.9/16",
      "IPv6Address": ""
    },
    "ingress-sbox": {
      "Name": "ingress-endpoint",
      "EndpointID": "9e3195e1db405c54758fea15f266255816a65a5f2b263d412e7906e7e51faf33",
      "MacAddress": "02:42:0a:ff:00:02",
      "IPv4Address": "10.255.0.2/16",
      "IPv6Address": ""
    }
  },
  "Options": {
    "com.docker.network.driver.overlay.vxlanid_list": "4096"
  },
  "Labels": {},
  "Peers": [
    {
      "Name": "917e8e1902a8",
      "IP": "192.168.0.70"
    },
    {
      "Name": "43c87f203409",
      "IP": "192.168.0.71"
    }
  ]
}
```

메모:

- Network Operations

8. 스택과 서비스 (Stack/Service)

❖ docker network inspect ingress

① docker network inspect ingress

```
[root@kubemaster example-voting-app]# docker network inspect ingress
[
  {
    "Name": "ingress",
    "Id": "33zsip6je0nseerjfs7iu6u59",
    "Created": "2018-04-05T20:22:13.859576938-04:00",
    "Scope": "swarm",
    "Driver": "overlay",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "10.255.0.0/16",
          "Gateway": "10.255.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": true,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "3fd994374ca181988a04bbd1746e680941d743b762898b391a543489c66c6113": {
        "Name": "voting_stack_vote.4.tgwqgjs7ut11grdue1oeltcr7",
        "EndpointID": "88f2f6cb7e8e67a03f1195a3256e5084b1d477dce4a28bcfd90c5cefba70ab",
        "MacAddress": "02:42:0a:ff:00:10",
        "IPv4Address": "10.255.0.16/16",
        "IPv6Address": ""
      },
      "d0408817d9f3a311446acb8b0e6e322cf7249b23755f26f61d8a3de27b6710c8": {
        "Name": "voting_stack_visualizer.1.xkme6y2zn4himbhvewo5fbz7s",
        "EndpointID": "7f7a4d6a1ba1fdee804ab7af215781d9a5e9781b028c896c1fb5f97bed711a0e",
        "MacAddress": "02:42:0a:ff:00:0e",
        "IPv4Address": "10.255.0.14/16",
        "IPv6Address": ""
      },
      "ingress-sbox": {
        "Name": "ingress-endpoint",
        "EndpointID": "2c1044a5161fb9e3a7c8705a3ae8f52a4aa76e0b11b14fc9d5aaaaf0aa5be55",
        "MacAddress": "02:42:0a:ff:00:02",
        "IPv4Address": "10.255.0.2/16",
        "IPv6Address": ""
      }
    }
  },
  {
    "Options": {
      "com.docker.network.driver.overlay.vxlanid_list":
      "4096"
    },
    "Labels": {},
    "Peers": [
      {
        "Name": "e529ff7ef122",
        "IP": "192.168.0.60"
      },
      {
        "Name": "eac1c7779806",
        "IP": "192.168.0.61"
      },
      {
        "Name": "67945b943ac1",
        "IP": "192.168.0.62"
      },
      {
        "Name": "1337293b54d9",
        "IP": "192.168.0.63"
      }
    ]
  }
]
```

메모:

- Network Operations

8. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성

④ sudo docker network inspect overnet

```
[root@kubemaster ~]# docker network inspect overnet
[
  {
    "Name": "overnet",
    "Id": "2n20w14b1ggir4ie2dok2tagz",
    "Created": "2018-04-04T03:57:19.826926805-04:00",
    "Scope": "swarm",
    "Driver": "overlay",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "10.0.0.0/24",
          "Gateway": "10.0.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "7doedf4eb9edcd271f4e9e16c078931205639bae90c5bc158f4d0a8b6ce04acf": {
        "Name": "myservice.2.qqxmz9c172rbssjnatk3t08sb",
        "EndpointID": "e7f646133243b5de9e66e50064973aa216c35d79e31e57d76fbc884a5d569b71",
        "MacAddress": "02:42:0a:00:00:06",
        "IPv4Address": "10.0.0.6/24",
        "IPv6Address": ""
      }
    },
    "Options": {
      "com.docker.network.driver.overlay.vxlanid_list": "4097"
    },
    "Labels": {},
    "Peers": [
      {
        "Name": "41816cd15b28",
        "IP": "192.168.0.60"
      },
      {
        "Name": "8ba267a3a74b",
        "IP": "192.168.0.61"
      }
    ]
  }
]
[root@kubemaster ~]#
```

메모:

- 생성 IP 주소 확인

8. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성

- ⑤ **sudo docker exec -it <CONTAINER ID> /bin/bash**
- ⑥ **apt-get update && apt-get install -y iptutils-ping**
- ⑦ **cat /etc/resolv.conf** # Check DNS Server @ 127.0.0.11:53
- ⑧ **ping -c5 myservice**

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

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

메모:

- DNS 동작 확인

8. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성

⑨ exit

⑩ sudo docker service inspect myservice

```
root@doedf4eb9ed:/# exit
exit
[root@kubemaster ~]# docker service inspect myservice
[
  {
    "ID": "3nzzhjmeoglebiq01y9w0mfu",
    "Version": {
      "Index": 21
    },
    "CreatedAt": "2018-04-04T07:57:19.663257975Z",
    "UpdatedAt": "2018-04-04T07:57:19.66539791Z",
    "Spec": {
      "Name": "myservice",
      "Labels": {},
      "TaskTemplate": {
        "ContainerSpec": {
          "Image": "ubuntu:latest@sha256:e348fbbca0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6",
          "Args": [
            "sleep",
            "infinity"
          ],
          "StopGracePeriod": 1000000000,
          "DNSConfig": {},
          "Isolation": "default"
        },
        "Resources": {
          "Limits": {},
          "Reservations": {}
        },
        "RestartPolicy": {
          "Condition": "any",
          "Delay": 500000000,
          "MaxAttempts": 0
        },
        "Placement": {
          "Platforms": [
            {
              "Architecture": "amd64",
              "OS": "linux"
            },
            {
              "Architecture": "linux",
              "OS": "linux"
            },
            {
              "Architecture": "arm64",
              "OS": "linux"
            },
            {
              "Architecture": "386",
              "OS": "linux"
            },
            {
              "Architecture": "ppc64le",
              "OS": "linux"
            },
            {
              "Architecture": "s390x",
              "OS": "linux"
            }
          ]
        }
      },
      "Networks": [
        {
          "Target": "2n20w14biggir4ie2dok2tagz",
          "ForceUpdate": 0,
          "Runtime": "container",
          "Mode": {
            "Replicated": {
              "Replicas": 2
            }
          },
          "UpdateConfig": {
            "Parallelism": 1,
            "FailureAction": "pause",
            "Monitor": 500000000,
            "MaxFailureRatio": 0,
            "Order": "stop-first"
          },
          "RollbackConfig": {
            "Parallelism": 1,
            "FailureAction": "pause",
            "Monitor": 500000000,
            "MaxFailureRatio": 0,
            "Order": "stop-first"
          },
          "EndpointSpec": {
            "Mode": "vip"
          },
          "Endpoint": {
            "Spec": {
              "Mode": "vip"
            }
          },
          "VirtualIPs": [
            {
              "NetworkID": "2n20w14biggir4ie2dok2tagz",
              "Addr": "10.0.0.4/24"
            }
          ]
        }
      ]
    }
  }
]
```

메모:

- 서비스 분석

8. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성 (예: visualizer)

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

메모:

- ONOS Install as a service

8. 스택과 서비스 (Stack/Service)

❖ 서비스 (Service) 생성 (예: ONOS)

- ① `sudo docker service create \`
- ② `--name onos \`
- ③ `--publish 8383:8181/tcp \`
- ④ `--publish 6653:6653/tcp \`
- ⑤ `--constraint node.role==manager \`
- ⑥ `--mount`
`type=bind,src=/var/run/docker.sock,dst=/var/run/docker.s`
`ock \`
- ⑦ `onosproject/onos:latest`
- ⑧ **Check Application started: OpenFlow Agent, Base Provider, LLDP Link Provider, Host Location Provider, Reactive Forwarding)**

메모:

- ONOS Install as a service

8. 스택과 서비스 (Stack/Service)

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

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

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

메모:

- Prometheus / Ghost Install as a service

8. 스택과 서비스 (Stack/Service)

❖ 마이크로서비스 App 실행 (중복)

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

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

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

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

메모:

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

8. 스택과 서비스 (Stack/Service)

❖ 마이크로서비스 App 실행 (중복)

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

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

메모:

- Check Local Host

8. 스택과 서비스 (Stack/Service)

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

- ① `sudo docker swarm join-token manager`
- ② `sudo docker swarm join-token worker`
- ③ `sudo docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-7rukwukuz7kmgnt0s1klrq5o2 192.168.0.60:2377 # @`
Manager
- ④ `sudo docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-7rukwukuz7kmgnt0s1klrq5o2 192.168.0.60:2377 # @`
Worker

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

```
docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-2a7m4ydly5j3hqgx7jdwyasg 192.168.0.60:2377
```

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

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

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

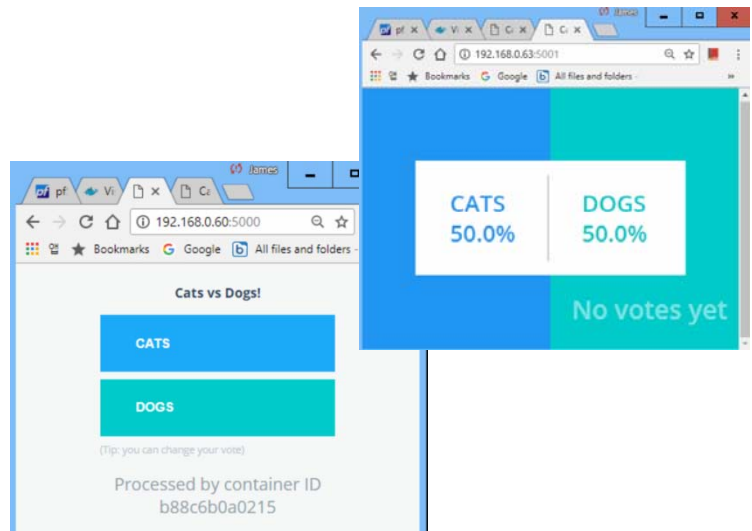
메모:

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

8. 스택과 서비스 (Stack/Service)

❖ 서비스 접속

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



메모:

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

목차



1. 실습 환경
2. vRouter (VyOS..)
3. Host (Ubuntu..)
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
- 9. FaaS (Open Function as a Service)**
10. Container Networking (Docker..)

9. FaaS (Open Function as a Service)

❖ OpenFaaS 실행

- ① **sudo docker ps** # check @ each host
- ② **sudo git clone https://github.com/openfaas/faas && **
**cd faas && **
./deploy_stack.sh --no-auth
- ③ **sudo docker stack ls**
- ④ **sudo docker stack services voting_stack**
- ⑤ **sudo docker stack rm func** # remove

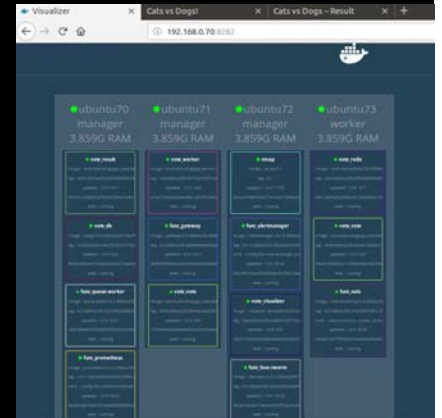
```

jslab@ubuntu71:/faas/faas$ sudo git clone https://github.com/openfaas/faas && cd faas && ./deploy_stack.sh --no-auth
[sudo] password for jslab:
Cloning into 'faas'...
remote: Counting objects: 16804, done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 16804 (delta 7), reused 8 (delta 3), pack-reused 16781
Receiving objects: 100% (16804/16804), 21.87 MiB | 9.88 MiB/s, done.
Resolving deltas: 100% (6697/6697), done.
Checking connectivity... done.
Attempting to create credentials for gateway..
Error response from daemon: rpc error: code = AlreadyExists desc = secret basic-auth-user already exists
Error response from daemon: rpc error: code = AlreadyExists desc = secret basic-auth-password already exists
[Credentials]
  already exist, not creating

Disabling basic authentication for gateway..

Deploying OpenFaaS core services
Creating config func_prometheus_rules
Creating config func_alertmanager_config
Creating config func_prometheus_config
Creating service func_prometheus
Creating service func_alertmanager
Creating service func_gateway
Creating service func_faas-swarm
Creating service func_nats
Creating service func_queue-worker
jslab@ubuntu71:/faas/faas/faas$

```



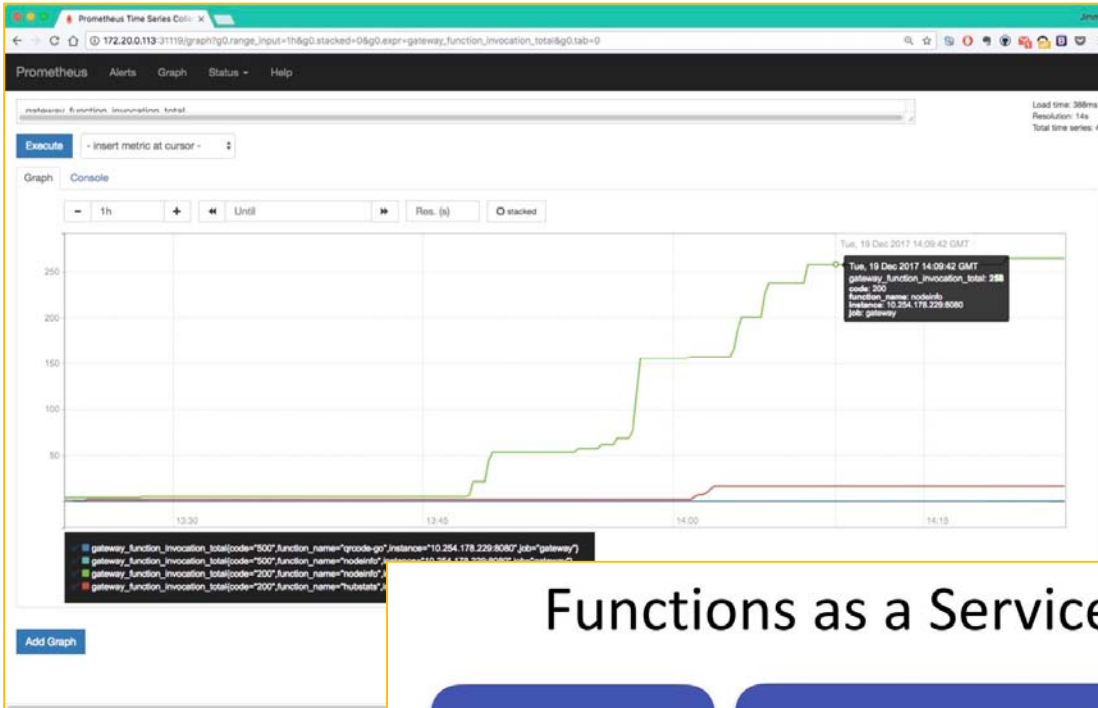
메모:

- Check port 8080 for other services
- Prometheus # <http://192.168.0.70:9090> @ Chrome
- <https://github.com/openfaas/workshop/blob/master/lab1.md>

9. FaaS (Open Function as a Service)

❖ Monitoring dashboard (Prometheus)

① # <http://192.168.0.70:9090> Prometheus @ Chrome



Functions as a Service

API Gateway

Function Watchdog



Prometheus



Swarm



Kubernetes



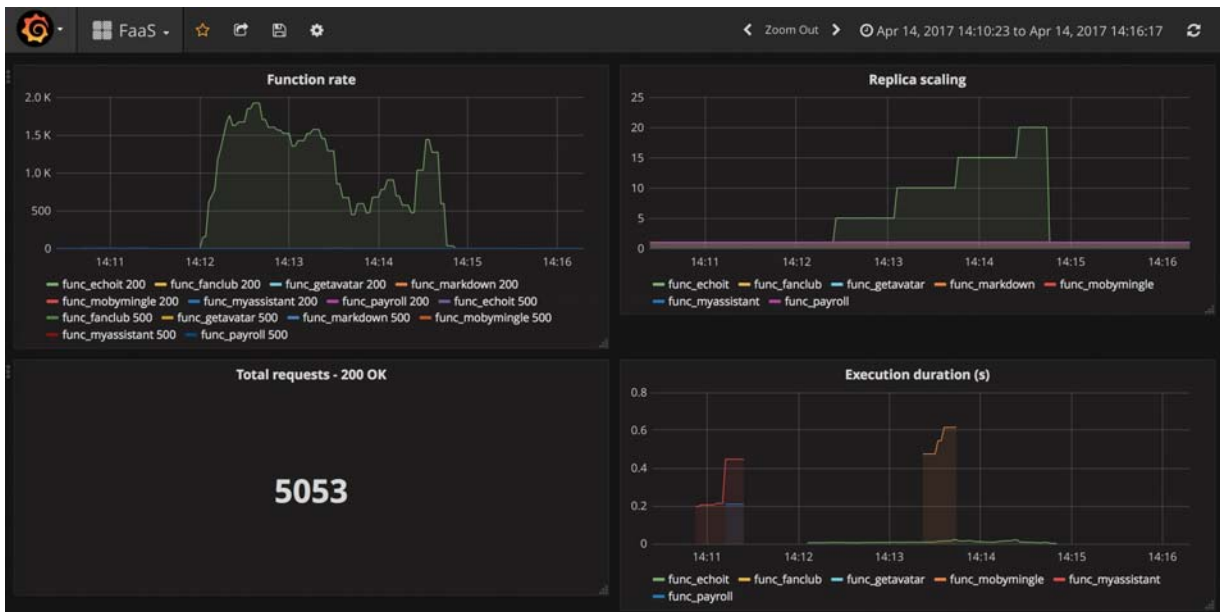
docker

메모:

9. FaaS (Open Function as a Service)

❖ Monitoring dashboard (Grafana)

- ① `sudo docker service create -d \`
- ② `--name=grafana \`
- ③ `--publish=3000:3000 \`
- ④ `--network=func_functions \`
- ⑤ `stefanprodan/faas-grafana:4.6.3`
- ⑥ <http://192.168.0.xx:3000> # UserID/Password=admin/admin



메모:

- <https://github.com/openfaas/workshop/blob/master/lab2.md>
- <http://127.0.0.1:3000/dashboard/db/openfaas>

9. FaaS (Open Function as a Service)

❖ Monitoring dashboard

- ① `sudo docker stack ls`
- ② `sudo docker service ls`

```

jslab@ubuntu70:~/example-voting-app$ sudo docker stack ls
NAME          SERVICES          ORCHESTRATOR
func          6                 Swarm
vote         6                 Swarm

jslab@ubuntu70:~/example-voting-app$ sudo docker service ls
ID            NAME              MODE             REPLICAS          IMAGE                                     PORTS
mdncce9m49qf func_alertmanager replicated        1/1               prom/alertmanager:v0.15.0
5vpyxdtq02y  func_faas-swarm  replicated        1/1               openfaas/faas-swarm:0.4.2
q8blgpkloqy3 func_gateway     replicated        1/1               openfaas/gateway:0.9.2
pmvkfebhgwag func_nats         replicated        1/1               nats-streaming:0.6.0
1n71jopatsh1 func_prometheus  replicated        1/1               prom/prometheus:v2.3.1
a8dx6f43w5k7 func_queue-worker replicated        1/1               openfaas/queue-worker:0.4.8
ub16urg0bf8v grafana          replicated        1/1               stefanprodan/faas-grafana:4.6.3
ubu6hyu429fe nmap            replicated        1/1               functions/nmap:0.1
3uz9qbuuqczg nodeinfo        replicated        1/1               functions/nodeinfo:latest
awmplayaybund vote_db          replicated        1/1               postgres:9.4
s031purbmo6u vote_redis      replicated        1/1               redis:alpine
jqhpg4lyi49q vote_result     replicated        1/1               dockersamples/examplevotingapp_result:before
magl363xug6i vote_visualizer replicated        1/1               dockersamples/visualizer:stable
00duyc4fa157 vote_vote       replicated        2/2               dockersamples/examplevotingapp_vote:before
4fypwyynec1  vote_worker    replicated        1/1               dockersamples/examplevotingapp_worker:latest

jslab@ubuntu70:~/example-voting-app$ sudo docker stack rm vote
Removing service vote_db
Removing service vote_redis
Removing service vote_result
Removing service vote_visualizer
Removing service vote_vote
Removing service vote_worker
Removing network vote_default
Removing network vote_backend
Removing network vote_frontend

jslab@ubuntu70:~/example-voting-app$ sudo docker stack rm func
Removing service func_alertmanager
Removing service func_faas-swarm
Removing service func_gateway
Removing service func_nats
Removing service func_prometheus
Removing service func_queue-worker
Removing config func_alertmanager_config
Removing config func_prometheus_rules
Removing config func_prometheus_config
Removing network func_functions
Failed to remove network aa8ld07mavezmq9ycxwjkp07l: Error response from daemon: rpc error: code = FailedPrecondition desc = network aa8ld07mavezmq9ycxwjkp07l
is in use by service 3uz9qbuuqczg5q7voqsafIarcFailed to remove some resources from stack: func
jslab@ubuntu70:~/example-voting-app$

```

메모:

- `sudo docker stack rm vote`
- `sudo docker stack rm func`

9. FaaS (Open Function as a Service)

❖ Remove Services

- ① **sudo docker service ls**
- ② **sudo docker service rm ub1** # foremost 3 bytes
- ③ **sudo docker service rm nmap** # name
- ④ **sudo docker service rm 3u** # foremost 2 bytes

```
jslab@ubuntu70:~/example-voting-app$ sudo docker service ls
ID                NAME          MODE          REPLICAS          IMAGE                                  PORTS
ub16urg0bf8v     grafana       replicated    1/1               stefanprodan/faas-grafana:4.6.3     *:3000->3000/tcp
ubu6hyu429fe     nmap         replicated    1/1               functions/nmap:0.1                  functions/nmap:0.1
3uz9qbuuqczg    nodeinfo     replicated    1/1               functions/nodeinfo:latest

jslab@ubuntu70:~/example-voting-app$ sudo docker service rm ub1
ub1
jslab@ubuntu70:~/example-voting-app$ sudo docker service rm nmap
nmap
jslab@ubuntu70:~/example-voting-app$ sudo docker service rm 3u
3u
jslab@ubuntu70:~/example-voting-app$
```

메모:

- sudo docker stack rm vote
- sudo docker stack rm func
- OpenFaaS 2018년 9월 현재 실행 Function(Service)을 수동 제거 필요

목차



1. 실습 환경
2. vRouter (VyOS..)
3. Host (Ubuntu..)
4. vSwitch (OVS..)
5. 컨테이너 (Docker..)
6. 이미지 (Docker Image)
7. 스웸 (Swarm)
8. 스택과 서비스 (Stack/Service)
9. FaaS (Open Function as a Service)
- 10. Container Networking (Docker..)**

10. 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:e5:68:b3 brd ff:ff:ff:ff:ff:ff link-netnsid 1
52: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
   link/ether 02:42:d7:eb:bb:ea brd ff:ff:ff:ff:ff:ff
54: veth33d2404@if53: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
   link/ether 06:1b:14:0c:7b:8e brd ff:ff:ff:ff:ff:ff link-netnsid 3
135: veth2716218@if134: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
   link/ether 32:0b:ee:24:65:eb brd ff:ff:ff:ff:ff:ff link-netnsid 5
139: vetha8b0fad@if138: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
   link/ether 9a:85:16:7a:6e:2f brd ff:ff:ff:ff:ff:ff link-netnsid 6
157: vethf2a0658@if156: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
   link/ether 5e:03:9f:bc:65:50 brd ff:ff:ff:ff:ff:ff link-netnsid 9
167: veth3c97a9a@if166: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP mode DEFAULT group default
   link/ether ca:b9:70:2c:c9:4e brd ff:ff:ff:ff:ff:ff link-netnsid 10
jslab@ubuntu70:~/example-voting-app$
```

메모:

- The Basics @ CentOS

10. 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
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

10. 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": {
      "7d3800792767f454cdf79d485000a62f5ceb993ac1146df03f8a4f66c7a8f5d8": {
        "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

10. Container Networking (Docker..)

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

```
james@masteratlocal:~$ sudo docker network inspect ingress
[
  {
    "Name": "ingress",
    "Id": "11yxmoq9eeyt066f00dv3jkfy",
    "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:

10. Container Networking (Docker..)

❖ Ping

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

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

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

메모:

- Ping

10. Container Networking (Docker..)

❖ Ping

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

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

메모:

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

10. Container Networking (Docker..)

❖ Ping

- ⑦ **exit**
- ⑧ **sudo docker ps**
- ⑨ **sudo docker stop <CONTAINER ID>**

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

메모:

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

10. Container Networking (Docker..)

❖ 외부 연결을 위한 NAT 구성

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

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

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

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

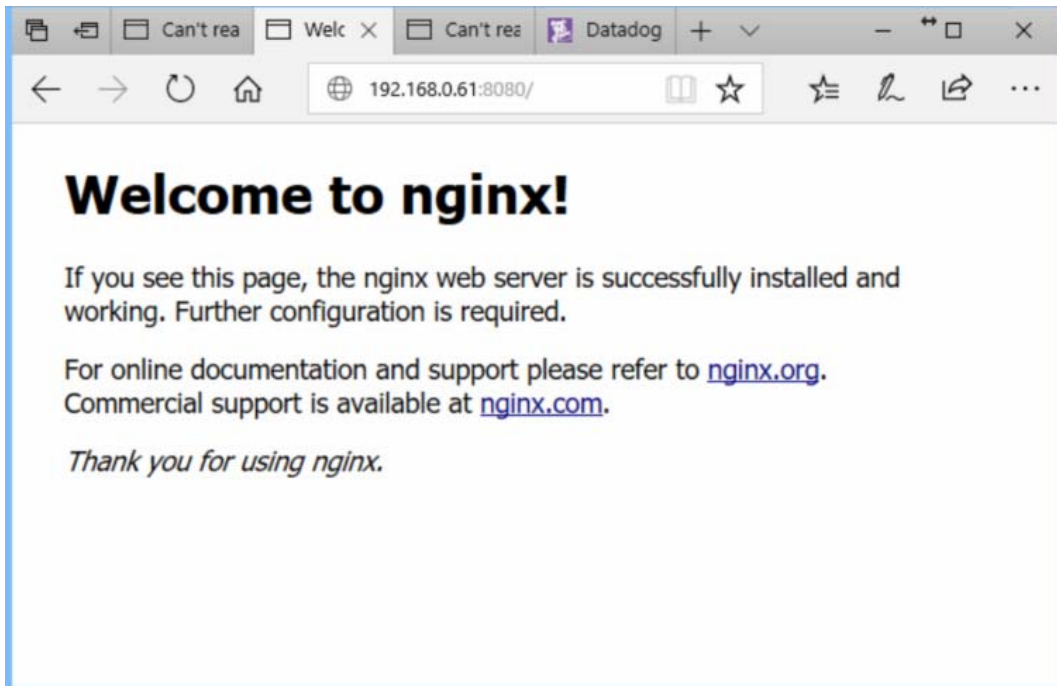
메모:

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

10. Container Networking (Docker..)

❖ 외부 연결을 위한 NAT 구성

④ **http://192.168.0.61:8080**



메모:

- 외부 연결을 위한 NAT 구성

10. Container Networking (Docker..)

❖ 오버레이(Overlay) 연결을 위한 구성

- ① `sudo docker swarm init --advertise-addr $(hostname -i)`
@ Manager
- ② `sudo docker swarm join --token SWMTKN-1-3our4qp38wf2qey61axjm13sp1g5gdup9gwvph6lmhp3zb3e2b-7rukukuz7kmgnt0s1klrq5o2 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-7rukukuz7kmgnt0s1klrq5o2 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-7rukukuz7kmgnt0s1klrq5o2 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
kb55f7sda5mduimloa2o5a9vx kubeworker1     Ready        Active           Leader          18.03.0-ce
[root@kubemaster ~]#
```

메모:

- Overlay Networking

10. Container Networking (Docker..)

❖ 오버레이(Overlay) 연결을 위한 구성

- ④ `sudo docker network create -d overlay overnet`
- ⑤ `sudo docker network ls`

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

메모:

- Overlay Networking

10. Container Networking (Docker..)

❖ 오버레이(Overlay) 연결을 위한 구성

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

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

메모:

- Overlay Networking

10. Container Networking (Docker..)

❖ 오버레이(Overlay) 연결을 위한 구성

- ① `sudo docker network create -d overlay overnet`
- ② `sudo docker service create --name myservice \`
`--network overnet \`
`--replicas 2 \`
`ubuntu sleep infinity`
- ③ `sudo docker service ps myservice`
- ④ `sudo docker network ls`

```
[root@kubemaster ~]# docker service create --name myservice ♣
> --network overnet ♣
> --replicas 2 ♣
> ubuntu sleep infinity
3nzzhjsoglebijq01y8w0mfu
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
qxxmz9c172rb     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

10. 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            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  --  172.18.0.2             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            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              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

10. 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::60d7:5bff:fed0:ac36/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:45 errors:0 dropped:0 overruns:0 frame:0
TX packets:78 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:4486 (4.4 KB) TX bytes:9393 (9.3 KB)

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

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

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

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

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

veth9f514d7 Link encap:Ethernet HWaddr d2:78:2c:67:91:6a
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>

10. 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$
```

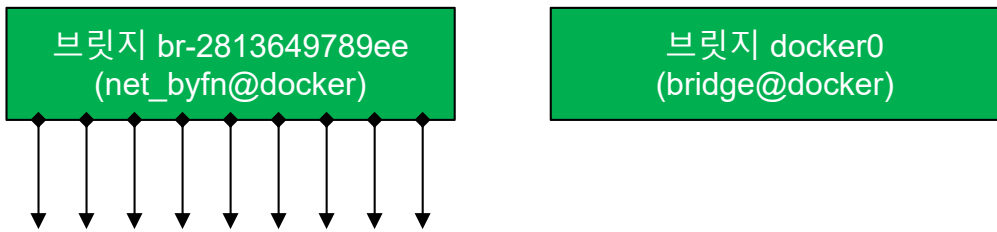
메모:

10. Container Networking (Docker..)

❖ sudo docker network ls & brctl show

- ① sudo apt install bridge-utils
- ② sudo docker network ls & brctl show

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

10. Container Networking (Docker..)

❖ brctl showmacs br-2813649789ee

① brctl showmacs br-2813649789ee

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

메모:

10. Container Networking (Docker..)

❖ sudo virsh net-list --all

- ① sudo apt-get install libvirt-bin
- ② sudo virsh net-list --all

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

메모:

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

10. Container Networking (Docker..)

❖ sudo docker network inspect bridge

① sudo docker network inspect bridge

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

메모:

10. Container Networking (Docker..)

❖ sudo docker network inspect net_byfn

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ sudo docker network inspect net_byfn
```

```
[
  {
    "Name": "net_byfn",
    "Id": "2813649789ee394b5ea3a548325f68dda03948da241a950d445e7d1ba463fddc",
    "Created": "2018-07-04T22:04:15.211034945+09:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,

```

```
    "Containers": {
      "2c94c8f7fa55275d1fc47a6eefd0044407f358ef4c7883b0b9502426526efbc7": {
        "Name": "dev-peer0.org2.example.com-mycc-1.0",
        "EndpointID": "0df17dc0e66d48d4d181c3a5e6b991b84f3ec07beac077678b33a7b702c2c817",
        "MacAddress": "02:42:ac:12:00:08",
        "IPv4Address": "172.18.0.8/16",
        "IPv6Address": ""
      },
      "3fd13dba67011dcdffa5855ce39dc63186d92ee5ed07cc9a61c0e50bd771fdec1": {
        "Name": "dev-peer1.org2.example.com-mycc-1.0",
        "EndpointID": "0b0cce7a0aefcbca5ac8244ab74106625825675a4626bd5127132f12ae7976e",
        "MacAddress": "02:42:ac:12:00:0a",
        "IPv4Address": "172.18.0.10/16",
        "IPv6Address": ""
      },
      "4cdaba34670ad245acc6476dd496c08afbe07e0f058dd79489347a134df0e1c1": {
        "Name": "peer0.org1.example.com",
        "EndpointID": "44e75ef1e1aaeb30d65392c715d0e2aeea696bbc73bfa45050a246205d91365",
        "MacAddress": "02:42:ac:12:00:05",
        "IPv4Address": "172.18.0.5/16",
        "IPv6Address": ""
      },
      "5559cf1bf28e4454b6e71f35d0b56f0c32462a1e661c15279a5300be74bea032": {
        "Name": "c1.1",
        "EndpointID": "98eea12635a61c8a5118594d434c133aeFeb6aa6d25f64d0f5fe45b8c1431c4c",
        "MacAddress": "02:42:ac:12:00:07",
        "IPv4Address": "172.18.0.7/16",
        "IPv6Address": ""
      },
      "5505ca81e17a97b8941cb721d63d5438cd2ed3db899990801a91c034226b477f": {
        "Name": "peer1.org1.example.com",
        "EndpointID": "58a08339cd5660c8bc10f6f78dac302940f637092600b9eea73d14f0f1be027",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
      },
      "57105838335f56fa09fe5afb4ace92df5c2d70ab7b5ce17344487aef1593c683": {
        "Name": "peer0.org2.example.com",
        "EndpointID": "840b2ceb01ead50ca4c1b0a07fcd31ede3b7842bcf18a2787fb3b4b0d91aae1",
        "MacAddress": "02:42:ac:12:00:04",
        "IPv4Address": "172.18.0.4/16",
        "IPv6Address": ""
      },
      "96690831f1d7c827284963a7a3d437c0b8c9bc817c9ee52d6b874ac5ee697fe": {
        "Name": "peer0.org2.example.com",
        "EndpointID": "ef3698b46f8df9ab8e41a3c3cbe8d8babdb9b082eba5d9af7ca62feead95cf54",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
      },
      "ae414d2b7eafde8f06b39009d0b7886e5d4491268bbe3f25f8fd54bbfb28f": {
        "Name": "orderer.example.com",
        "EndpointID": "9b27cb188990eb947d73bb3da3730e610868f2ae1972de00175400dd5ba5452",
        "MacAddress": "02:42:ac:12:00:06",
        "IPv4Address": "172.18.0.6/16",
        "IPv6Address": ""
      },
      "fc2d31807bd1a537f3c0106c5825767d9465728ab2e1de0f9fd011b09315d4db": {
        "Name": "dev-peer0.org1.example.com-mycc-1.0",
        "EndpointID": "88feb343b5b88aef8473712bae431e7ee872721fcedaf0a8d421fbae8a449ccc",
        "MacAddress": "02:42:ac:12:00:09",
        "IPv4Address": "172.18.0.9/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

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

NETWORK ID	NAME	DRIVER	SCOPE
2cc6ad951481	bridge	bridge	local
cc3648572554	host	host	local
2813649789ee	net_byfn	bridge	local
f3ac7c07b82c	none	null	local

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

메모:

10. Container Networking (Docker..)

❖ sudo docker image inspect hyperledger/fabric-peer

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ sudo docker image inspect h
[
  {
    "Id": "sha256:1be3342008805372667574cee7615f1c974a466aa768e7ec235afd6819f009b"
    "RepoTags": [
      "hyperledger/fabric-peer:1.2.0-rc1",
      "hyperledger/fabric-peer:latest"
    ],
    "RepoDigests": [
      "hyperledger/fabric-peer:sha256:16f70db0f825f0cee17cf7ca3c0d75e55e0c0ff5c3"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2018-06-22T00:36:35.531377378Z",
    "Container": "61bfc67d5b289680dcaebac1d7ea44d1fe1fc6a078367717bc18bd95a65a6984"
    "ContainerConfig": {
      "Hostname": "61bfc67d5b28",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
        "FABRIC_CFG_PATH=/etc/hyperledger/fabric"
      ],
      "Cmd": [
        "/bin/sh",
        "-c",
        "#(nop)",
        "LABEL org.hyperledger.fabric.version=1.2.0-rc1 org.hyperledger.fabric."
      ],
      "ArgsEscaped": true,
      "Image": "sha256:bbae1994a2485689541a095354949cc8bc580b21dc39f9dee96d35e4b"
      "Volumes": null,
      "WorkingDir": "",
      "Entrypoint": null,
      "OnBuild": null,
      "Labels": [
        "org.hyperledger.fabric.base.version": "0.4.8",
        "org.hyperledger.fabric.version": "1.2.0-rc1"
      ]
    },
    "Architecture": "amd64",
    "Os": "linux",
    "Size": 153884304,
    "VirtualSize": 153884304,
    "GraphDriver": {
      "Data": {
        "LowerDir":
          "/var/lib/docker/overlay2/908fc7152015bc795c08ae980209ac3067718a822a5c0df6954ca35c3dae42e/diff:/var/lib/dock
          er/overlay2/6979fc6f996624be151b4d8239cda9c3e8a9f7b54db958c89634929828b19f05/diff:/var/lib/dock
          er/overlay2/2623cd0772eaa2ad7117007e90740100140f57d4e779150f0342f377c1f4da/diff:/var/lib/docker/overl
          ay2/9c5549ade8dffa862bad81030b25ebced81c5c91a6ce1ef3592caef6c228f1/diff:/var/lib/docker/overlay2/e5
          e342d65fe4ff35bcef17e85617fdac4e32bc9bf03f38f5fa89c12fe2c8967/diff:/var/lib/docker/overlay2/015bb983
          a00bd655a58fdd53884753a4f5fbab97404c67f8284fc07e/diff:/var/lib/docker/overlay2/250891650f82b348
          792ed9f4a24e8aca16faf2328932d4fc5692dad692ec9b7d/diff:/var/lib/docker/overlay2/aa435110a944480779055f8
          d049f5647baa13dd552c0b1b8c48b5fa94dbc3717/diff:/var/lib/docker/overlay2/39f61cb8d991d20333db23a724a86
          6de5730bd81b1edfc8f36ec917714f71ed/diff",
        "MergedDir":
          "/var/lib/docker/overlay2/56b0cf35981707b7f1e7c55ee29f2ab3191409f5613452ff58a313b56a296810/merged",
        "UpperDir":
          "/var/lib/docker/overlay2/56b0cf35981707b7f1e7c55ee29f2ab3191409f5613452ff58a313b56a296810/diff",
        "WorkDir":
          "/var/lib/docker/overlay2/56b0cf35981707b7f1e7c55ee29f2ab3191409f5613452ff58a313b56a296810/work"
      }
    },
    "Name": "overlay2"
  },
  {
    "RootFS": {
      "Type": "layers",
      "Layers": [
        "sha256:fc0bfa2912f0cd6b9d13f91f288f112a2b825f3f758a4443aacb45bfc108cc74",
        "sha256:e1a9a6284d0d24d8194ac84b372619e75cd35a6866b74925b7274c7056561e4",
        "sha256:ac7299292f8b2f710d3b911c6a4e02ae8f06792c39822097f9c4e9c2672b32d",
        "sha256:a5e66470b2812e91798db36eb103c1f1e135bbe167e4b2ad5ba425b8db98ee8d",
        "sha256:a8de0e025d94b33db3542e1e8ce58829144b30c6cd1ff057ee05b1491933c3",
        "sha256:4ee119879d6e55ff932f0cc55950c2e0d72e45c6171e74538e423ae2c24417c4",
        "sha256:8da6ba42b46daadd9ec1306edccf88a8ac0be5808c461910050f3a09a709466",
        "sha256:5d58b558803d45bed2b7225a5f1a26f32fe860e3a0e3eeb51d3791d0cf2900",
        "sha256:0ebc7925211e82c4af72605fd1beb757b3a3e79736e55352f7b2eddb055120b",
        "sha256:681c3dde08b756b3fcc450b94d6f5f80d890baa98dfdbb1c3f6b39f79049559"
      ]
    },
    "Metadata": {
      "LastTagTime": "2018-07-04T21:55:14.550301563+09:00"
    }
  }
]
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

```
jslab@jslab-virtual-machine:~/fabric-samples/first-network$ sudo docker images
REPOSITORY          TAG         IMAGE ID      CREATED      SIZE
dev-peer1_or2_     latest     dev-peer1_or2_1-0-26c2ef32838554aa04f7ad6f100ca865e87959e9a126e86d764c8d01f8346ab
dev-peer0_or1_     latest     dev-peer0_or1_1-0-384f11f484b9302d190c453200cfb25174306f0e8f53f4e944d5ee36cab00c9
dev-peer0_or2_     latest     dev-peer0_or2_1-0-15b571b3c8a49066b7ec74497da3b27e54e0df1345daf3951b94245ce09e42b
hyperledger/fabric-ca          latest     hyperledger/fabric-ca-1.2.0-rc1
hyperledger/fabric-ca          latest     hyperledger/fabric-ca-1.2.0-rc1
hyperledger/fabric-tools       latest     hyperledger/fabric-tools-1.2.0-rc1
hyperledger/fabric-tools       latest     hyperledger/fabric-tools-1.2.0-rc1
hyperledger/fabric-cosmvn      latest     hyperledger/fabric-cosmvn-0.4.8
hyperledger/fabric-orderer     latest     hyperledger/fabric-orderer-0.4.8
hyperledger/fabric-orderer     latest     hyperledger/fabric-orderer-0.4.8
hyperledger/fabric-peer       latest     hyperledger/fabric-peer-1.2.0-rc1
hyperledger/fabric-peer       latest     hyperledger/fabric-peer-1.2.0-rc1
hyperledger/fabric-zookeeper   latest     hyperledger/fabric-zookeeper-0.4.8
hyperledger/fabric-kafka       latest     hyperledger/fabric-kafka-0.4.8
hyperledger/fabric-couchdb     latest     hyperledger/fabric-couchdb-0.4.8
hyperledger/fabric-couchdb     latest     hyperledger/fabric-couchdb-0.4.8
hyperledger/fabric-baseos      amd64     hyperledger/fabric-baseos-amd64-1.2.0-rc1
jslab@jslab-virtual-machine:~/fabric-samples/first-network$
```

메모:

10. Container Networking (Docker..)

❖ 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/erver.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$
```

메모: