Installation Kubernetes 1.29 on VirtualBox Guide

설치 전 과정

1. 권장 실습 환경

   ① Windows 11

   ② 한글 계정명 사용 금지

   ③ C Drive 여유공간 50GB 이상

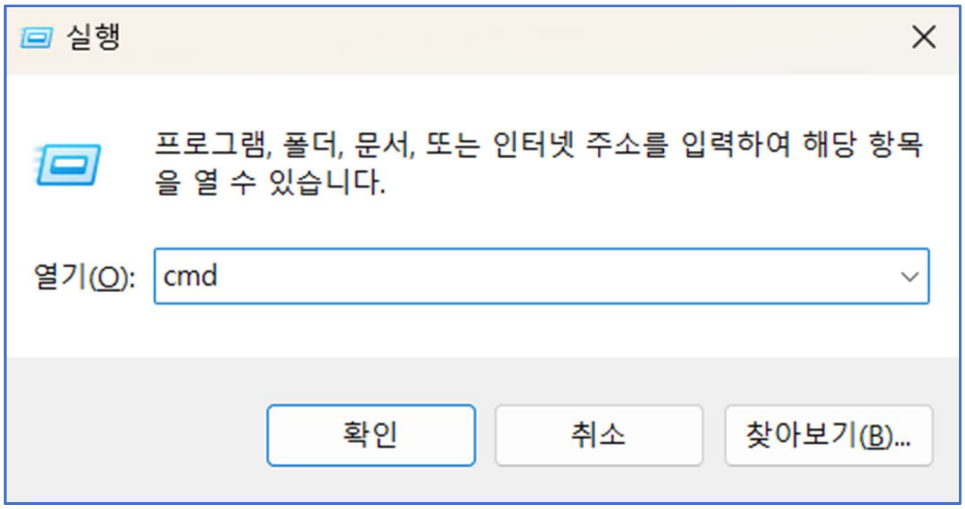
   ④ RAM 16GB 이상 권장

   ⑤ 안정적인 Network 환경(가급적 유선연결)

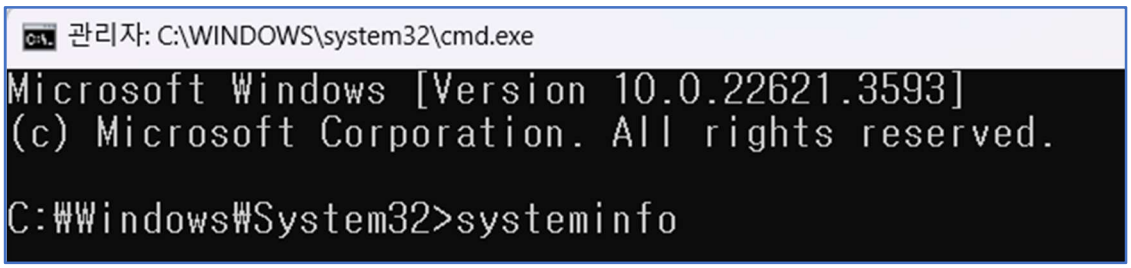
2. Hyper-V 설정 확인

   ※ 실습용 Windows 시스템에 Hyper-V 기능이 활성화되어 있으면 실습 불가

   ① Windows Key + r > 실행 창 > cmd > Ctrl + Shift + Enter



   ② **systeminfo**

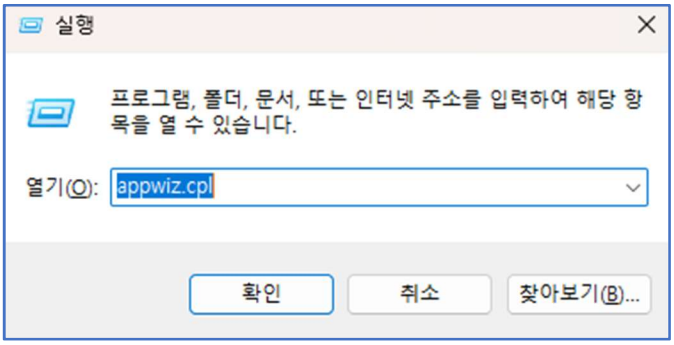


③ 출력 결과 확인

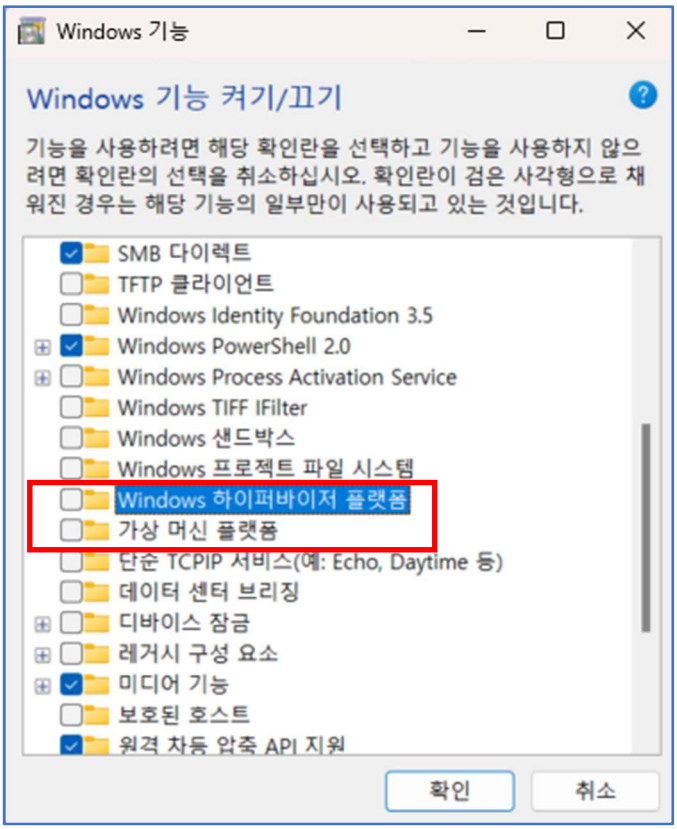
1) 추가 작업 필요

```
                                상태:         미디어 연결이 끊어짐
Hyper-V 요구 사항:      하이퍼바이저가 검색되었습니다. Hyper-V에 필요한 기능이 표시되지 않습니다.
```

- 실행 창에 **[appwiz.cpl]** 입력 > **[Windows 기능 켜기/끄기]**

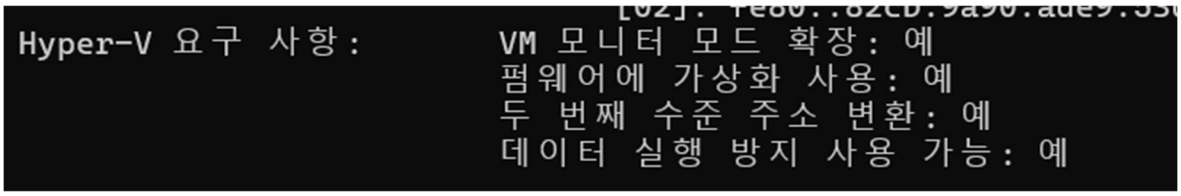


- 아래 그림과 같이 **[Windows 하이퍼바이저 플랫폼]**과 **[가상 머신 플랫폼]**을 체크 해제한다.



- [확인] 버튼을 클릭하고 시스템을 재부팅한다.

2) 정상인 경우



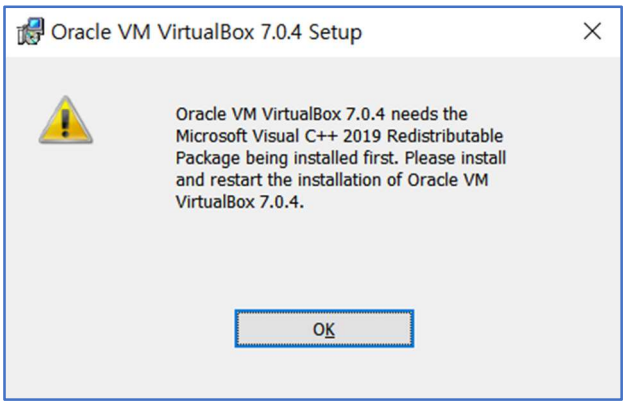
3. Oracle VirtualBox 설치

① VirtualBox 다운로드 및 설치 (VMware, Hyper-V 제거 필요)

② https://www.virtualbox.org, VirtualBox-7.0.6-155176-Win.exe

③ 설치파일 다운로드 후 관리자 권한으로 실행 (한글 계정명 사용 금지)

④ Microsoft Visual C++ 2019 Redistributable Package Error 발생 시



- Microsoft Visual C++ 2019 Redistributable Package 설치 후 진행

https://learn.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170

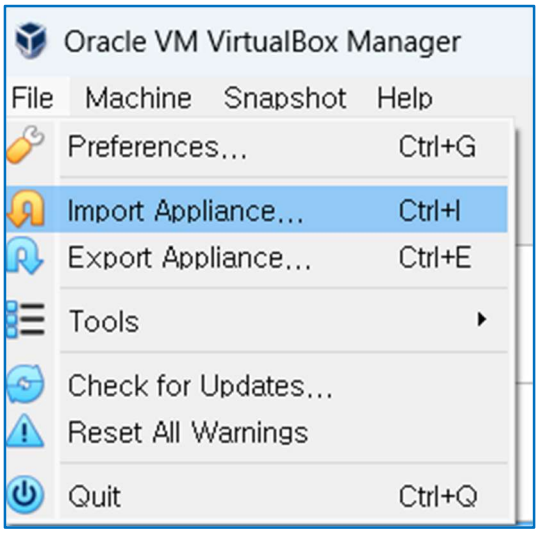
- VC_redist.x64.exe 설치 후 다시 VirtualBox 설치할 것

⑤ VirtualBox 호스트키 조합 설정

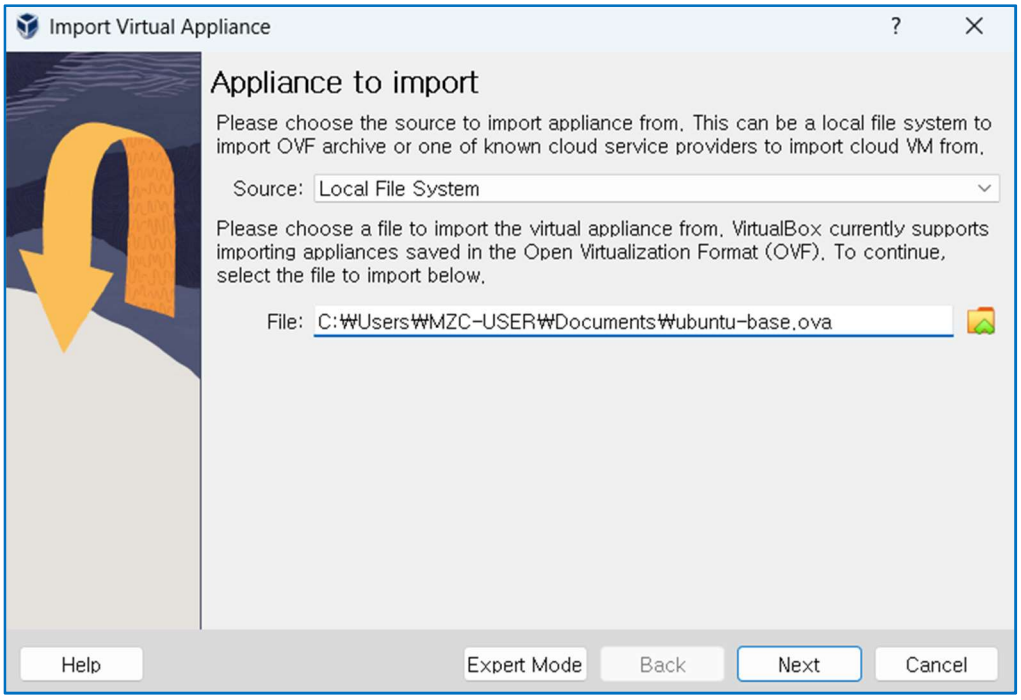
- 파일 > 환경 설정 > 입력 > 가상 머신
- 호스트 키 조합 > **F12**

4. Ubuntu VM Import – Ubuntu Linux Server 22.04 LTS

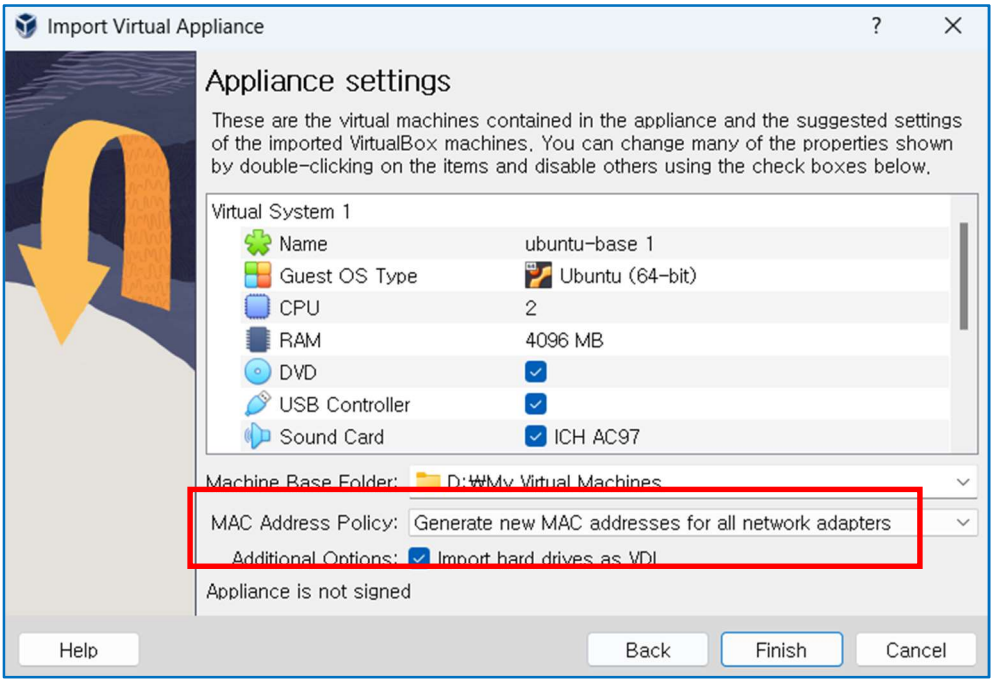
① **[파일]** > **[가상 시스템 가져오기(Import Appliance)]** > 파일



② "ubuntu-base.ova" 선택 > **[열기]** > **[다음]**



③ **[Appliance settings]** 창에서, **[MAC 주소 정책(MAC Address Policy)] : [모든 네트워크 어댑터의 새 MAC 주소 생성(Generate new MAC addresses for all network adapters)]** → 반드시 선택하여 변경할 것 > **[Finish]**



5. VirtualBox - NatNetwork 설정(192.168.137.0/24)

① **[도구(Tools)]** > **[network]** > **[만들기(Create)]** > **NatNetwork**

② **[General Options]** > **[IP4 Prefix]** : **192.168.137.0/24** > **[적용(Apply)]**

③ **[포트 포워딩(Port Forwarding)]** > 새 포트 포워딩 규칙 추가(+)

- **[이름]** : ubuntu
- **[호스트 IP]** : 192.168.56.1
- **[호스트 포트]** : 100
- **[게스트 IP]** : 192.168.137.100
- **[게스트 포트]** : 22

이름	프로토콜	호스트 IP	호스트 포트	게스트 IP	게스트 포트
ubuntu	TCP	192.168.56.1	100	192.168.137.100	22

④ **[적용(Apply)]**

6. Ubuntu VM IP 설정 및 Hostname 변경하기

① 로그인 후, **00-installer-config.yaml** 파일 수정

$ **sudo vi /etc/netplan/00-installer-config.yaml** ← 각 라인의 들여쓰기는 반드시 2칸이다.

```
# This is the network config written by 'subiquity'
network:
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
      - 192.168.137.100/24
      routes:
      - to: default
        via: 192.168.137.1
      nameservers:
        addresses: [8.8.8.8,8.8.7.7]
  version: 2
```

$ **sudo netplan try**

$ **sudo netplan apply**

$ **ip a**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:0c:17:c5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.137.100/24 brd 192.168.137.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe0c:17c5/64 scope link tentative
       valid_lft forever preferred_lft forever
```

② Ubuntu VM Network Adapter 변경하기

- Ubuntu VM > **[Settings]** > **[Network]** > **[Apater1]** > **[Attached to]** : NAT에서 **NAT Network**로 변경
- 이름이 **NatNetwork**로 변경 확인 **[OK]** 버튼 클릭하여 적용

③ 호스트 이름은 다음의 명령으로 변경한다.

$ **sudo hostnamectl set-hostname master**

④ 호스트 이름 변경 후 확인한다.

```
$ sudo hostname
```

⑤ Network Test를 위해 다음의 명령을 수행한다.

```
$ sudo apt update
```

⑥ 만일 오류가 발생하면 다음의 명령을 수행한다.

```
$ sudo vi /etc/resolv.conf
nameserver 8.8.8.8
```

⑦ 저장하면 바로 적용됨

```
$ ping -c 4 www.google.com
```

⑧ Ubuntu VM 종료 후 Ubuntu VM의 이름은 **master**로 변경 후 **[시작]** > **[Headless Start]** 선택 > SSH Client Tool(PuTTY, Tabby, MobaXterm 등)으로 로그인하기

7. Docker Installation(https://docs.docker.com/engine/install/ubuntu/ 참조)

① Set up Docker's `apt` repository.

```
$ sudo apt-get update

$ sudo apt-get install ca-certificates curl

$ sudo install -m 0755 -d /etc/apt/keyrings

$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

$ sudo chmod a+r /etc/apt/keyrings/docker.asc


$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

$ sudo apt-get update
```

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo ₩
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu ₩
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | ₩
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

② Install the Docker packages.

$ **sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin**

③ Docker 설치 후 작업

$ **sudo usermod -aG docker $USER**

$ **logout** > 다시 로그인

$ **id** > 확인

④ 상세 정보 확인

$ **docker version**

$ **docker info**

⑤ Docker 사용량 확인

$ **docker system df**

⑥ $ **sudo poweroff**

8. 가상 머신 Snapshot

① Ubuntu의 가상 머신의 이름이 **master**로 변경 확인 후,

② master > **[Snapshots]** > **[Name]** : Before K8s, **[설명]** : Kubernets 설치 전 > **[Take]**

Kubernetes 설치하기

1. 쿠버네티스 클러스터 구성 전 요구사항 확인

   ① 관리자로 변경

   $ **sudo -i**

   ② 2GB 이상 메모리

   # **free**

   ③ 2 CPUs or more

   # **lscpu**

   ④ /etc/hosts 수정하기

   # **vi /etc/hosts**

   127.0.0.1 localhost

   127.0.1.1. master

   192.168.137.100 master

   192.168.137.101 worker1

   192.168.137.102 worker2

   ⑤ IP 주소 및 UUID 확인

   # **ip a**

   # **ifconfig**

   # **cat /sys/class/dmi/id/product_uuid**

2. Swap Disable

   ① Swap 가동 중인지 확인하기

   # **free -h**

```
root@master:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       256Mi       3.1Gi       1.0Mi       450Mi       3.4Gi
Swap:          4.0Gi          0B       4.0Gi
```

cat /proc/swaps 로도 확인 가능

```
root@master:~# cat /proc/swaps
Filename                                Type            Size            Used            Priority
/dev/sda2                               partition       4194300         0               -2
root@master:~#
```

② Swap 가동 중지시키기

swapoff --all(-가 2개)

```
root@master:~# swapoff --all
root@master:~#
```

③ Swap 가동 중지 확인하기

free -h or # **cat /proc/swaps**

```
root@master:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       252Mi       3.1Gi       1.0Mi       451Mi       3.4Gi
Swap:             0B          0B          0B
root@master:~#
root@master:~#
root@master:~# cat /proc/swaps
Filename                                Type            Size            Used            Priority
root@master:~#
```

④ /etc/fstab 파일에서 swap 부분 주석처리하기

sed -i '/swap/s/^/#/' /etc/fstab

```
root@master:~# sed -i '/swap/s/^/#/' /etc/fstab
root@master:~#
```

⑤ /etc/fstab 파일 확인

cat /etc/fstab

```
root@master:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
#/dev/disk/by-uuid/48a16ab8-3ced-4eeb-bd5f-21f8f75c9e4a none swap sw 0 0
# / was on /dev/sda4 during curtin installation
/dev/disk/by-uuid/1f72770a-f49b-471a-8ea0-2fc1ffc0adf3 / ext4 defaults 0 1
# /boot was on /dev/sda3 during curtin installation
/dev/disk/by-uuid/7a3de8a3-e4e6-47d6-bcbe-dac2cdcbc1aa /boot ext4 defaults 0 1
root@master:~#
```

⑥ master VM 재 부팅

shutdown -r now

3. Bridge Network 설정

① 재 부팅 후 다시 다음 명령으로 Swap Disable

swapoff --all(-가 2개)

② iptables가 bridege traffic을 확인하고 제어할 수 있도록 설정

cat <<EOF | tee /etc/modules-load.d/k8s.conf

overlay

br_netfilter

EOF ← **마지막 줄 EOF 뒤쪽에 공백이 없도록 주의할 것**

```
root@master:~# cat <<EOF | tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
overlay
br_netfilter
root@master:~#
```

sudo modprobe overlay

sudo modprobe br_netfilter

```
root@master:~# modprobe overlay
root@master:~# modprobe br_netfilter
root@master:~#
```

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

net.ipv4.ip_forward   = 1

EOF ← 마지막 줄 EOF 뒤쪽에 공백이 없도록 주의할 것

```
root@master:~# cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
root@master:~#
```

③ 다음과 같이 시스템에 적용한다.

sysctl --system(-가 2개)

```
root@master:~# sysctl --system
* Applying /etc/sysctl.d/10-console-messages.conf ...
kernel.printk = 4 4 1 7
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
kernel.kptr_restrict = 1
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
kernel.sysrq = 176
* Applying /etc/sysctl.d/10-network-security.conf ...
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.all.rp_filter = 2
* Applying /etc/sysctl.d/10-ptrace.conf ...
kernel.yama.ptrace_scope = 1
* Applying /etc/sysctl.d/10-zeropage.conf ...
vm.mmap_min_addr = 65536
* Applying /usr/lib/sysctl.d/50-default.conf ...
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.default.accept_source_route = 0
sysctl: setting key "net.ipv4.conf.all.accept_source_route": Invalid argument
net.ipv4.conf.default.promote_secondaries = 1
sysctl: setting key "net.ipv4.conf.all.promote_secondaries": Invalid argument
net.ipv4.ping_group_range = 0 2147483647
net.core.default_qdisc = fq_codel
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.protected_regular = 1
fs.protected_fifos = 1
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
kernel.pid_max = 4194304
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
* Applying /etc/sysctl.conf ...
root@master:~#
```

4. 인증 키 추가

```
# apt update

# apt upgrade -y

# apt install apt-transport-https ca-certificates curl jq gpg -y

# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | tee /etc/apt/sources.list.d/kubernetes.list

# cat /etc/apt/sources.list.d/kubernetes.list
```

```
# apt update
# apt upgrade -y
# apt install apt-transport-https ca-certificates curl jq gpg -y
# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | tee /etc/apt/sources.list.d/kubernetes.list

# cat /etc/apt/sources.list.d/kubernetes.list
```

5. Kubernetes 설치(https://v1-29.docs.kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/ 참조)

```
# apt update

# apt install containerd kubelet kubeadm kubectl
```

① 설치 확인

```
# apt-mark hold kubelet kubeadm kubectl
```

kubelet set on hold.

kubeadm set on hold.

kubectl set on hold.

```
root@master:~# apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@master:~#
```

② 버전 확인

kubelet --version(-가 2개)

kubeadm version

kubectl version --output=yaml(-가 2개)

```
root@master:~# kubelet --version
Kubernetes v1.29.5
root@master:~# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"29", GitVersion:"v1.29.5", GitCommit:"59755ff595fa452
 Compiler:"gc", Platform:"linux/amd64"}
root@master:~#
root@master:~# kubectl version --output=yaml
clientVersion:
  buildDate: "2024-05-14T10:46:12Z"
  compiler: gc
  gitCommit: 59755ff595fa4526236b0cc03aa2242d941a5171
  gitTreeState: clean
  gitVersion: v1.29.5
  goVersion: go1.21.9
  major: "1"
  minor: "29"
  platform: linux/amd64
kustomizeVersion: v5.0.4-0.20230601165947-6ce0bf390ce3

The connection to the server localhost:8080 was refused - did you specify the right host or port?
root@master:~#
```

③ master VM poweroff

poweroff

6. config.toml 파일 수정하기

① 다음과 같이 config.toml 파일을 수정한다.

```
# containerd config default > /etc/containerd/config.toml

# vi /etc/containerd/config.toml
```

② 다음과 같이 SystemdCgroup의 값을 false에서 true로 수정한 후, 저장한다.

```
SystemdCgroup = true
```

```
        [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
          BinaryName = ""
          CriuImagePath = ""
          CriuPath = ""
          CriuWorkPath = ""
          IoGid = 0
          IoUid = 0
          NoNewKeyring = false
          NoPivotRoot = false
          Root = ""
          ShimCgroup = ""
          SystemdCgroup = true
      [plugins."io.containerd.grpc.v1.cri".containerd.untrusted_workload_runtime]
        base_runtime_spec = ""
```

③ containerd 서비스를 재시작하고 시스템을 확인한다.

```
# systemctl restart containerd

# free -h

# sysctl --system
```

```
root@master:~# systemctl restart containerd
root@master:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       225Mi       3.0Gi       1.0Mi       589Mi       3.4Gi
Swap:             0B          0B          0B
root@master:~# sysctl --system
* Applying /etc/sysctl.d/10-console-messages.conf ...
kernel.printk = 4 4 1 7
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
```

④ 다시 config.toml 파일을 오픈하여 다음과 같이 sandbox_image의 값을 수정한 후, 저장한다.

```
# /etc/containerd/config.toml

# sandbox_image = "registry.k8s.io/pause:3.8" -> "registry.k8s.io/pause:3.9"
```

Worker1 & Worker2 복제 및 환경 설정

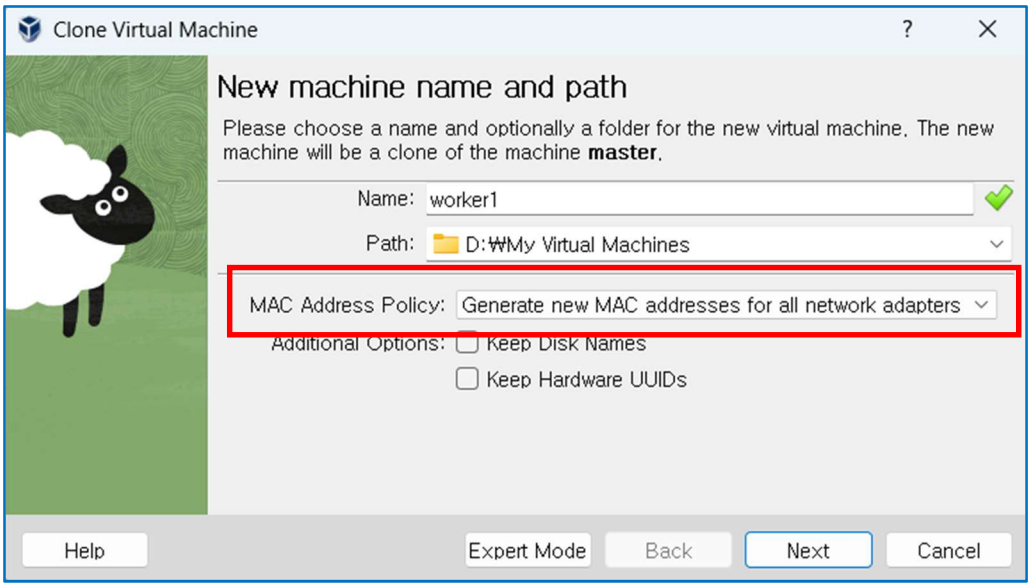
1. Worker1 복제

① master (마우스 오른쪽 버튼 클릭) > **[복제(Clone)]**

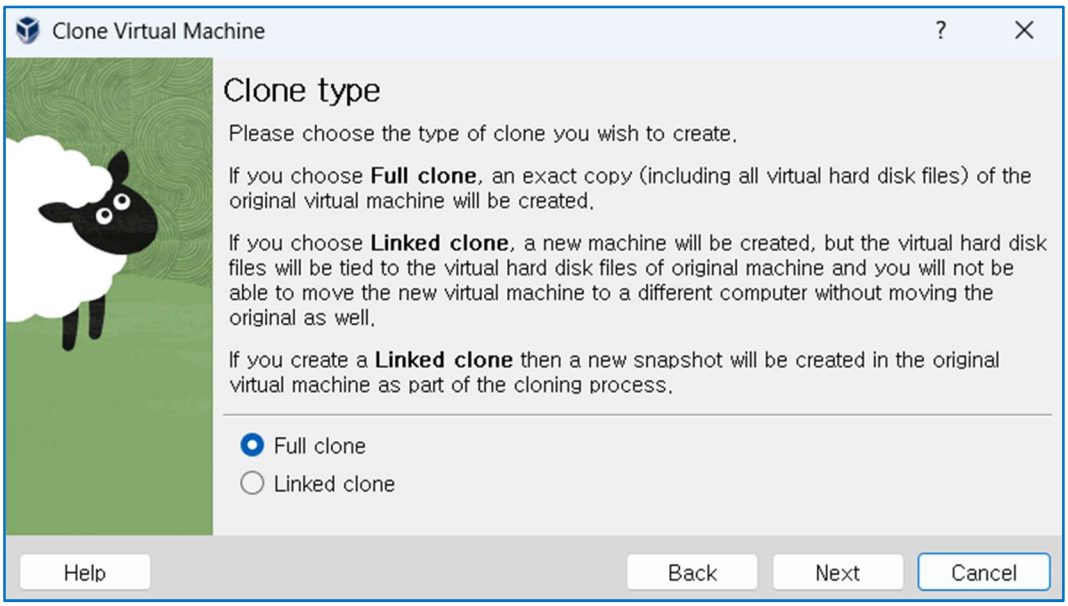
② **[이름]**: worker1

③ **[MAC 주소 정책]**: **[모든 네트워크 어댑터의 새 MAC 주소 생성]** (반드시 선택하여 변경)

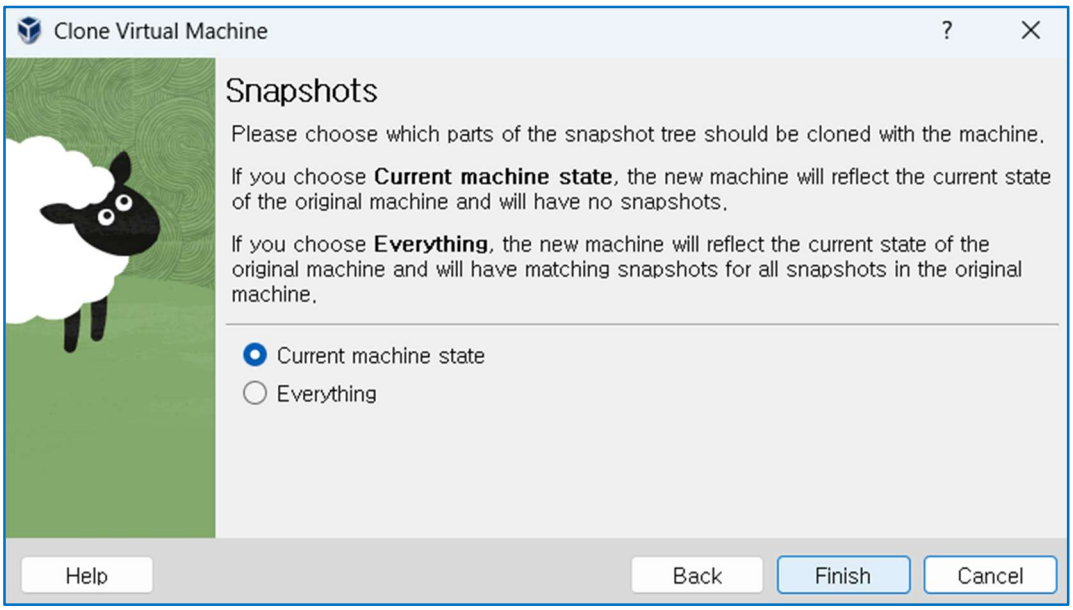
④ 추가 옵션: 체크 안함



⑤ **[다음]** > **[완전한 복제]**



⑥ **[다음] > [현재 머신 상태(Current machine state)] > [Finish]**



2. worker1 환경 설정

① worker1 시작

② 로그인 : ubuntu/P@$$W0rd

③ hostname 변경

```
# sudo -i

# hostname

# hostnamectl set-hostname worker1

# hostname
```

```
root@master:~# hostnamectl set-hostname worker1
root@master:~# hostname
worker1
root@master:~# _
```

④ IP Address 변경

ip a

vi /etc/netplan/00-installer-config.yaml

- addresses: [192.168.137.100/24] > 100을 **101**로 변경

```
# This is the network config written by 'subiquity'
network:
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
      - 192.168.137.101/24
      routes:
      - to: default
        via: 192.168.137.1
      nameservers:
        addresses: [8.8.8.8,8.8.7.7]
  version: 2
```

- 변경될 IP 주소 적용 및 확인

netplan apply

ip a

```
root@master:~# netplan try
Do you want to keep these settings?


Press ENTER before the timeout to accept the new configuration


Changes will revert in 119 seconds
Configuration accepted.
root@master:~# netplan apply
root@master:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:a9:5c:10 brd ff:ff:ff:ff:ff:ff
    inet 192.168.137.101/24 brd 192.168.137.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fea9:5c10/64 scope link
       valid_lft forever preferred_lft forever
root@master:~#
```

⑤ NatNetwork 설정 (192.168.137.0/24)

- **[파일]** > **[환경 설정]**
- **[네트워크]** > NatNetwork (더블클릭)
- **[포트 포워딩]** > 새 포트 포워딩 규칙 추가(+)
- **[이름]** > **worker1**
- **[호스트 IP]** > 192.168.56.1
- **[호스트 포트]** > 101
- **[게스트 IP]** > 192.168.137.101
- **[게스트 포트]** > 22
- **[확인]**

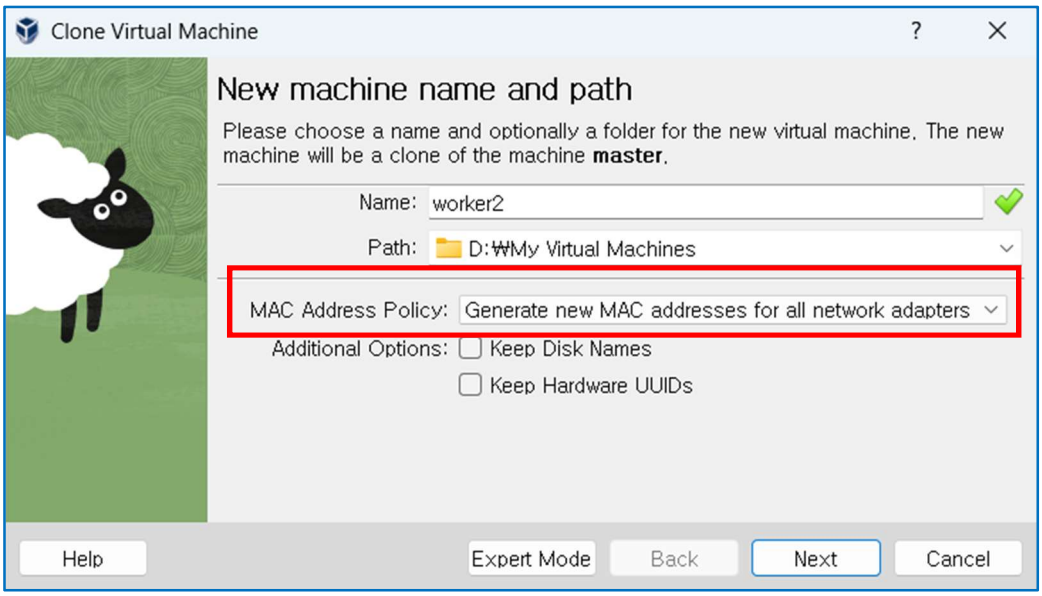
3. Worker2 복제

① master (마우스 오른쪽 버튼 클릭) > **[복제(Clone)]**

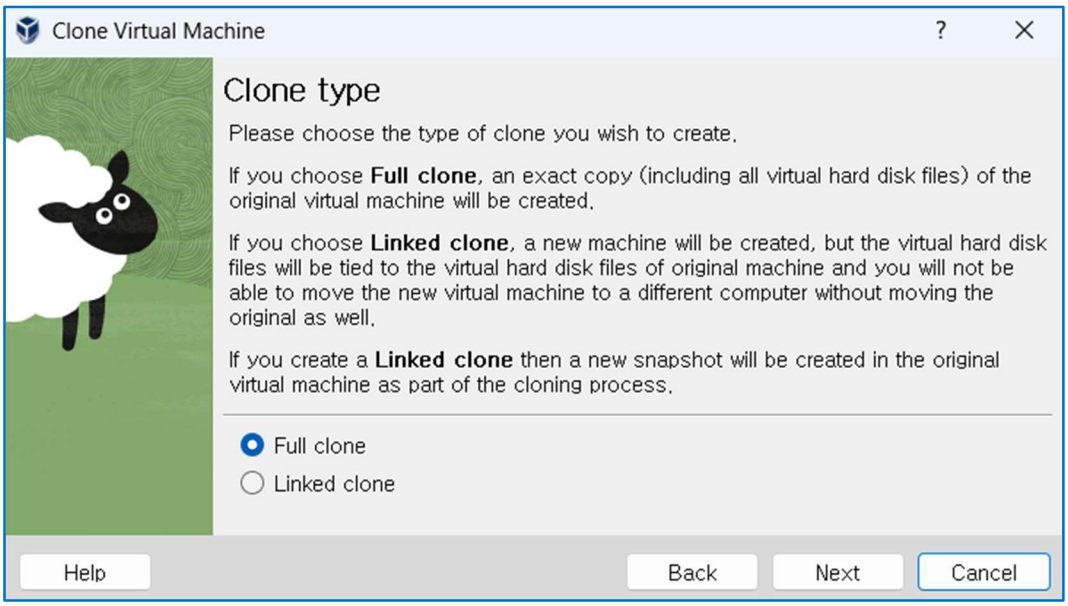
② **[이름]**: worker2

③ **[MAC 주소 정책]**: **[모든 네트워크 어댑터의 새 MAC 주소 생성]** (반드시 선택하여 변경)

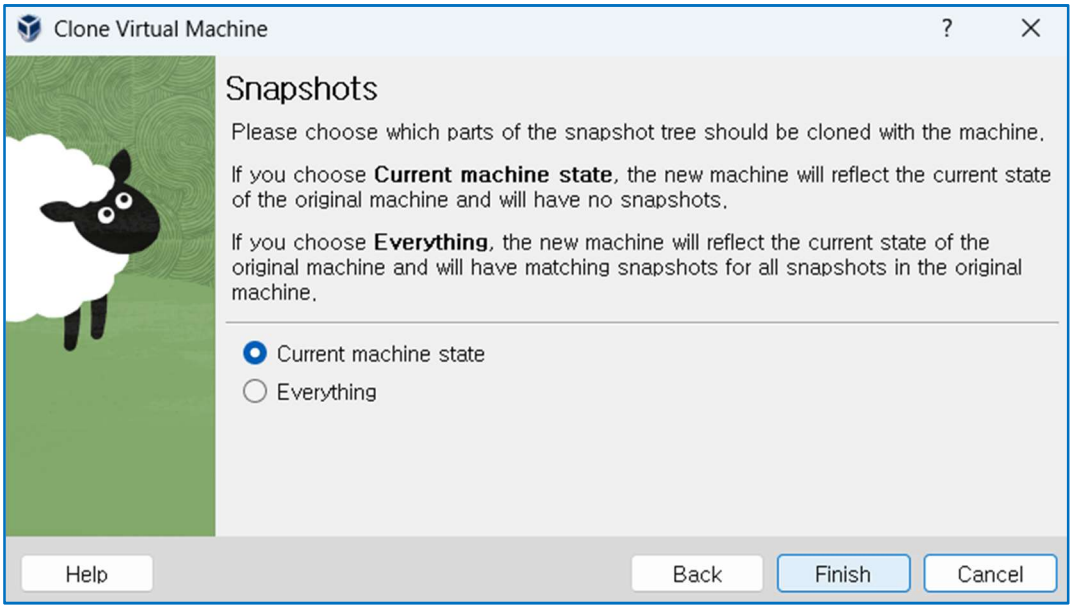
④ 추가 옵션: 체크 안함



⑤ **[다음] > [완전한 복제]**



⑥ **[다음] > [현재 머신 상태(Current machine state)] > [Finish]**



4. worker2 환경 설정

① worker2 시작

② 로그인 : ubuntu/P@$$W0rd

③ hostname 변경

```
# sudo -i

# hostname

# hostnamectl set-hostname worker2

# hostname
```

```
root@master:~# hostnamectl set-hostname worker2
root@master:~# hostname
worker2
root@master:~#
```

④ IP Address 변경

```
# ip a

# vi /etc/netplan/00-installer-config.yaml
```

- addresses: [192.168.137.100/24] > 100을 **102**로 변경

```
# This is the network config written by 'subiquity'
network:
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
      - 192.168.137.102/24
      routes:
      - to: default
        via: 192.168.137.1
      nameservers:
        addresses: [8.8.8.8,8.8.7.7]
  version: 2
```

- 변경될 IP 주소 적용 및 확인

netplan apply

ip a

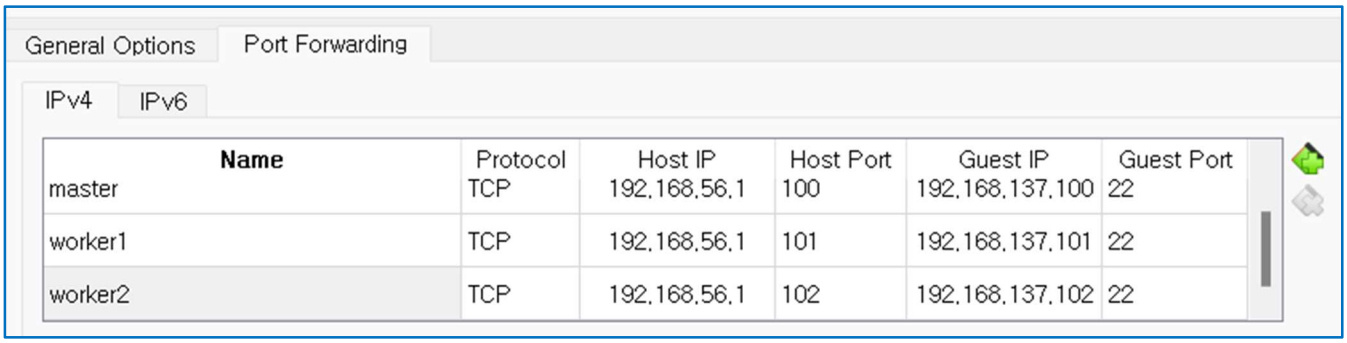
```
root@master:~# netplan try
Do you want to keep these settings?


Press ENTER before the timeout to accept the new configuration


Changes will revert in 119 seconds
Configuration accepted.
root@master:~#
root@master:~# netplan apply
root@master:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:6c:bd:79 brd ff:ff:ff:ff:ff:ff
    inet 192.168.137.102/24 brd 192.168.137.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe6c:bd79/64 scope link
       valid_lft forever preferred_lft forever
root@master:~#
```

⑤ NatNetwork 설정 (192.168.137.0/24)

- **[파일]** > **[환경 설정]**
- **[네트워크]** > NatNetwork (더블클릭)
- **[포트 포워딩]** > 새 포트 포워딩 규칙 추가(+)
- **[이름]** > worker2
- **[호스트 IP]** > 192.168.56.1
- **[호스트 포트]** > 102
- **[게스트 IP]** > 192.168.137.102
- **[게스트 포트]** > 22
- **[확인]**



- worker1, worker2 모두 shutdown

5. master, worker1, worker2 모두 Start

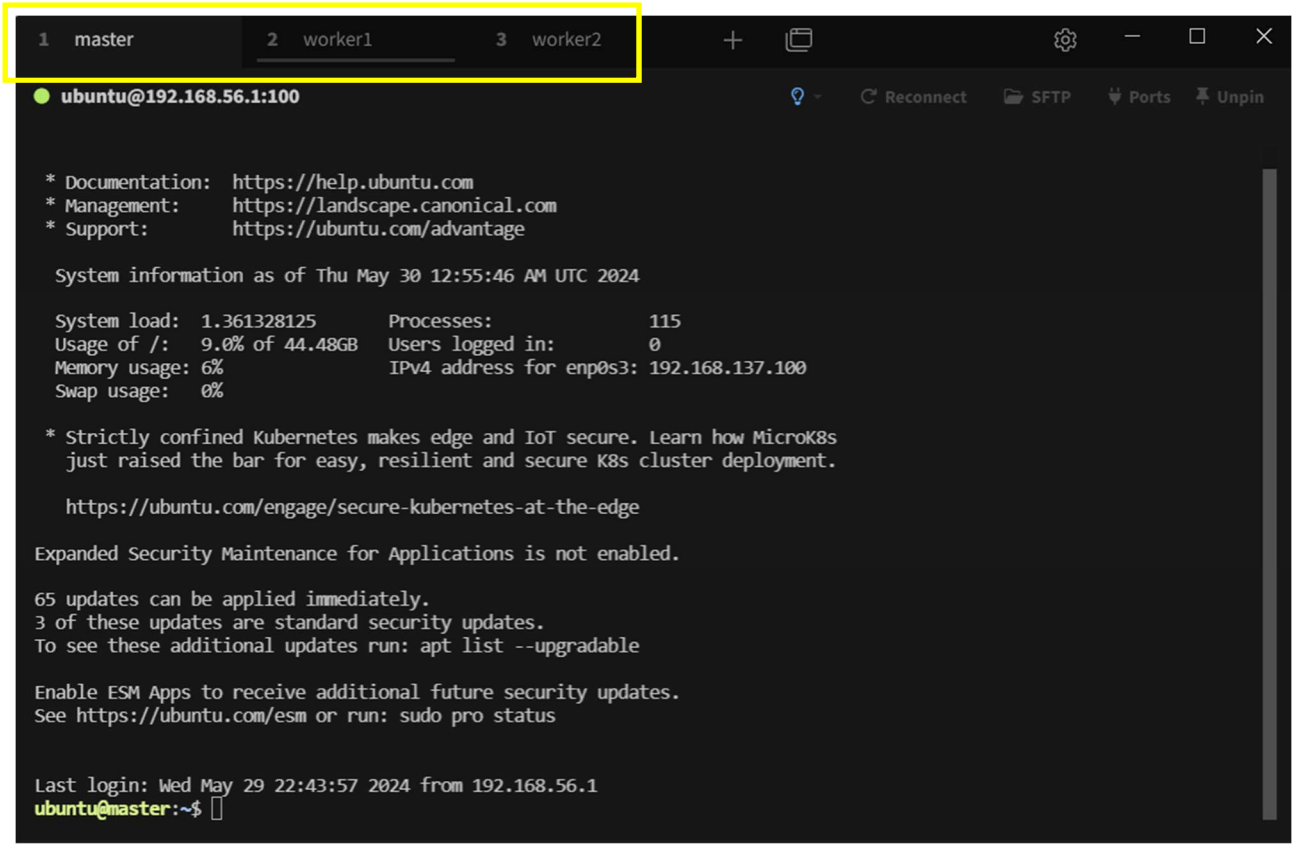
① 모두 **[시작]** > **[헤드리스 시작]**

② 각각 SSH Client Tools 로 연결할 것

③ master : 192.168.56.1:100

④ worker1 : 192.168.56.1:101

⑤ worker2 : 192.168.56.1:102



⑥ master에서 worker1, worker2로, worker1에서 master, worker2로, worker2에서 master, worker1으로 각각 ping test

```
ubuntu@master:~$ ping worker1
PING worker1 (192.168.137.101) 56(84) bytes of data.
64 bytes from worker1 (192.168.137.101): icmp_seq=1 ttl=64 time=1.24 ms
64 bytes from worker1 (192.168.137.101): icmp_seq=2 ttl=64 time=0.983 ms
64 bytes from worker1 (192.168.137.101): icmp_seq=3 ttl=64 time=1.32 ms
64 bytes from worker1 (192.168.137.101): icmp_seq=4 ttl=64 time=0.912 ms
^C
--- worker1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.912/1.113/1.317/0.169 ms
ubuntu@master:~$ ping worker2
PING worker2 (192.168.137.102) 56(84) bytes of data.
64 bytes from worker2 (192.168.137.102): icmp_seq=1 ttl=64 time=1.28 ms
64 bytes from worker2 (192.168.137.102): icmp_seq=2 ttl=64 time=0.914 ms
64 bytes from worker2 (192.168.137.102): icmp_seq=3 ttl=64 time=0.946 ms
^C
--- worker2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.914/1.047/1.283/0.166 ms
ubuntu@master:~$
```

```
[plugins."io.containerd.grpc.v1.cri"]
  cdi_spec_dirs = ["/etc/cdi", "/var/run/cdi"]
  device_ownership_from_security_context = false
  disable_apparmor = false
  disable_cgroup = false
  disable_hugetlb_controller = true
  disable_proc_mount = false
  disable_tcp_service = true
  drain_exec_sync_io_timeout = "0s"
  enable_cdi = false
  enable_selinux = false
  enable_tls_streaming = false
  enable_unprivileged_icmp = false
  enable_unprivileged_ports = false
  ignore_image_defined_volumes = false
  image_pull_progress_timeout = "1m0s"
  max_concurrent_downloads = 3
  max_container_log_line_size = 16384
  netns_mounts_under_state_dir = false
  restrict_oom_score_adj = false
  sandbox_image = "registry.k8s.io/pause:3.9"
  selinux_category_range = 1024
  stats_collect_period = 10
  stream_idle_timeout = "4h0m0s"
```

6. master node 설정(kubeadm init)

① 다음과 같이 kubeadm로 마스터 노드 초기화를 진행한다.

kubeadm init --pod-network-cidr 10.10.0.0/16 --node-name master

```
root@master:~# kubeadm init --pod-network-cidr 10.10.0.0/16 --node-name master
I0530 06:15:26.949758    7953 version.go:256] remote version is much newer: v1.30.1; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.5
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local master] and IPs [10.96.0.1 192.168.137.100]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master] and IPs [192.168.137.100 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IPs [192.168.137.100 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 11.005413 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
```

```
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.137.100:6443 --token 1vdw0p.u618ys5628kc18un \
        --discovery-token-ca-cert-hash sha256:a7df00fa129621e980614bcf4165477ee17174368ca3139e04fe20a4ab192fa5
root@master:~#
```

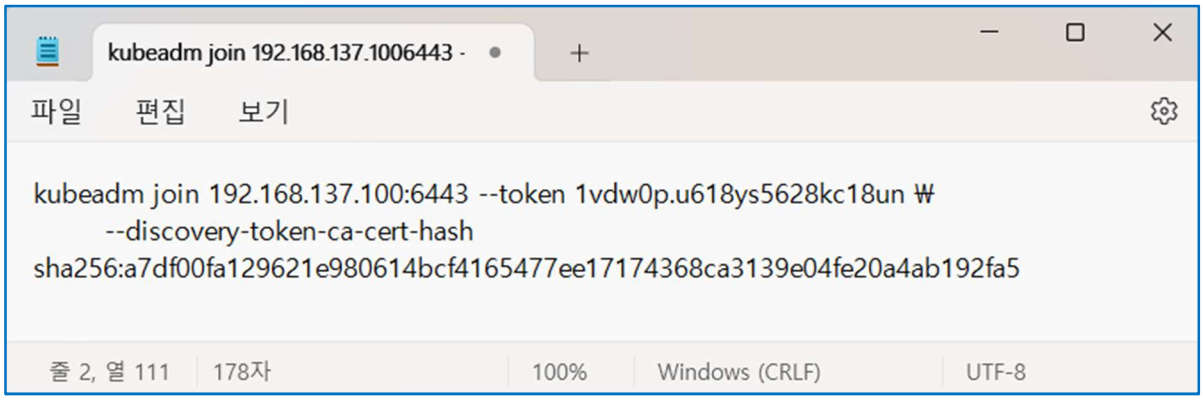
② 다음과 같이 Cluster Join Token의 값을 확인할 있다.

kubeadm token create --print-join-command

```
root@master:~# kubeadm token create --print-join-command
kubeadm join 192.168.137.100:6443 --token htsvo8.vc4mkqwobzq0npvc --discovery-token-ca-cert-hash sha256:a7df00fa129621e980614bcf4165477ee17174368ca3139e04fe20a4ab192fa5
root@master:~#
```

③ Cluster Join Token 복사 및 저장

- 메모장을 열어서 다음과 같이 Join Token을 복사 후 붙여넣기 하고 join_token.txt로 저장한다.



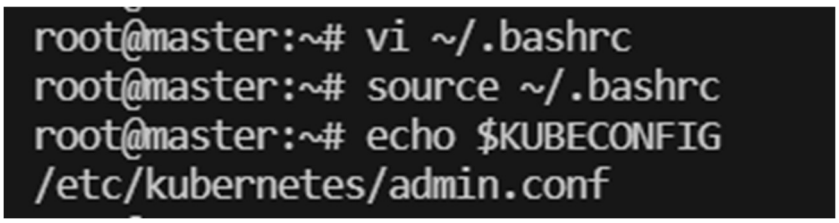
④ kubeadm init 마무리

- kubeadm init을 성공적으로 마친 후 2가지 중 하나를 할 수 있다.
- root 사용자 클러스터의 API 접근 인증 설정은 다음과 같이 수행한다.
- kubeconfig 파일의 위치를 root 사용자의 KUBECONFIG 셸 환경변수에 추가하는 작업이다.
- root 사용자의 .bashrc 파일을 열고 제일 마지막 줄에 KUBECONFIG를 추가한다.

```
# vi ~/.bashrc
export KUBECONFIG=/etc/kubernetes/admin.conf
```

- 파일을 저장 후 다음의 명령을 마저 수행하여 설정 내용을 확인한다.

```
# source ~/.bashrc
# echo $KUBECONFIG
```



⑤ Pod 통신을 위한 Cilium 기반 Pod 네트워크 추가

- 다음과 같이 Cilium을 다운로드 후 압축을 풀고 다운받은 파일을 삭제한다.

```
# curl -LO https://github.com/cilium/cilium-cli/releases/latest/download/cilium-linux-amd64.tar.gz

# tar xzvfC cilium-linux-amd64.tar.gz /usr/local/bin

# rm cilium-linux-amd64.tar.gz
```

```
root@master:~# curl -LO https://github.com/cilium/cilium-cli/releases/latest/download/cilium-linux-amd64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0
100 42.2M  100 42.2M    0     0  9407k      0  0:00:04  0:00:04 --:--:-- 18.5M
root@master:~# tar xzvfC cilium-linux-amd64.tar.gz /usr/local/bin
cilium
root@master:~# rm cilium-linux-amd64.tar.gz
```

⑥ Cilium을 설치한다.

```
# cilium install --set ipam.operator.clusterPoolIPv4PodCIDRList=10.10.0.0/16
```

```
root@master:~# cilium install --set ipam.operator.clusterPoolIPv4PodCIDRList=10.10.0.0/16
ℹ  Using Cilium version 1.15.5
🔮 Auto-detected cluster name: kubernetes
🔮 Auto-detected kube-proxy has been installed
root@master:~# cilium status
    /¯¯\
 /¯¯\__/¯¯\    Cilium:             1 errors, 1 warnings
 \__/¯¯\__/    Operator:           1 errors, 1 warnings
 /¯¯\__/¯¯\    Envoy DaemonSet:    disabled (using embedded mode)
 \__/¯¯\__/    Hubble Relay:       disabled
    \__/       ClusterMesh:        disabled

Deployment             cilium-operator    Desired: 1, Unavailable: 1/1
DaemonSet              cilium             Desired: 1, Unavailable: 1/1
Containers:            cilium             Pending: 1
                       cilium-operator    Pending: 1
Cluster Pods:          0/2 managed by Cilium
Helm chart version:
Image versions         cilium-operator    quay.io/cilium/operator-generic:v1.15.5@sha256:f5d3d19754074ca052be6aac5d1ffb1de1eb5f2d947222b5f10f6d97ad4383e8: 1
                       cilium             quay.io/cilium/cilium:v1.15.5@sha256:4ce1666a73815101ec9a4d360af6c5b7f1193ab00d89b7124f8505dee147ca40: 1
Errors:                cilium-operator    cilium-operator                  1 pods of Deployment cilium-operator are not ready
                       cilium             cilium                           1 pods of DaemonSet cilium are not ready
Warnings:              cilium             cilium-tbz5z                     pod is pending
                       cilium-operator    cilium-operator-65bcfcf69c-4fxgv    pod is pending
root@master:~#
```

- 아직 초기화하는 과정이어서 error가 보인다.
- 다음의 명령으로 현재까지의 node를 읽어온다.

```
# kubectl get nodes
```

```
root@master:~# kubectl get nodes
NAME     STATUS   ROLES           AGE   VERSION
master   Ready    control-plane   12m   v1.29.5
root@master:~#
```

- 아직 worker1, worker2 노드가 Cluster에 추가하지 않았기 때문에 master node만 보인다.

7. worker1 노드에서 Cluster 조인하기

① 다음의 명령으로 worker1 노드를 클러스터에 조인한다.

kubeadm join 192.168.137.100:6443 –token {Cluster Join Token 값}

```
root@worker1:~#
root@worker1:~# kubeadm join 192.168.137.100:6443 --token 1vdw0p.u618ys5628kc18un \
        --discovery-token-ca-cert-hash sha256:a7df00fa129621e980614bcf4165477ee17174368ca3139e04fe20a4ab192fa5
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@worker1:~#
```

8. worker2 노드에서 Cluster 조인하기

① 다음의 명령으로 worker1 노드를 클러스터에 조인한다.

kubeadm join 192.168.137.100:6443 –token {Cluster Join Token 값}

```
root@worker2:~#
root@worker2:~# kubeadm join 192.168.137.100:6443 --token 1vdw0p.u618ys5628kc18un \
        --discovery-token-ca-cert-hash sha256:a7df00fa129621e980614bcf4165477ee17174368ca3139e04fe20a4ab192fa5
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@worker2:~#
```

9. master node에서 클러스터 조인 확인

kubectl get nodes

```
root@master:~# kubectl get nodes
NAME      STATUS   ROLES           AGE    VERSION
master    Ready    control-plane   17m    v1.29.5
worker1   Ready    <none>          3m2s   v1.29.5
worker2   Ready    <none>          66s    v1.29.5
root@master:~#
```

kubectl get pods --all-namespaces -o wide

```
root@master:~# kubectl get pods --all-namespaces -o wide
NAMESPACE     NAME                                READY   STATUS    RESTARTS      AGE   IP               NODE      NOMINATED NODE   READINESS GATES
kube-system   cilium-7h7sx                        1/1     Running   0             29m   192.168.137.101  worker1   <none>           <none>
kube-system   cilium-operator-65bcfcf69c-4fxgv    1/1     Running   0             32m   192.168.137.100  master    <none>           <none>
kube-system   cilium-rw6sc                        1/1     Running   0             27m   192.168.137.102  worker2   <none>           <none>
kube-system   cilium-tbz5z                        1/1     Running   0             32m   192.168.137.100  master    <none>           <none>
kube-system   coredns-76f75df574-j8vfj            1/1     Running   0             42m   10.10.0.155      master    <none>           <none>
kube-system   coredns-76f75df574-m8wq8            1/1     Running   0             42m   10.10.0.117      master    <none>           <none>
kube-system   etcd-master                         1/1     Running   0             43m   192.168.137.100  master    <none>           <none>
kube-system   kube-apiserver-master               1/1     Running   0             43m   192.168.137.100  master    <none>           <none>
kube-system   kube-controller-manager-master      1/1     Running   1 (31m ago)   43m   192.168.137.100  master    <none>           <none>
kube-system   kube-proxy-chgkn                    1/1     Running   0             29m   192.168.137.101  worker1   <none>           <none>
kube-system   kube-proxy-dgqht                    1/1     Running   0             42m   192.168.137.100  master    <none>           <none>
kube-system   kube-proxy-fxlth                    1/1     Running   0             27m   192.168.137.102  worker2   <none>           <none>
kube-system   kube-scheduler-master               1/1     Running   1 (31m ago)   43m   192.168.137.100  master    <none>           <none>
root@master:~#
```

kubectl cluster-info

```
root@master:~# kubectl cluster-info
Kubernetes control plane is running at https://192.168.137.100:6443
CoreDNS is running at https://192.168.137.100:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
root@master:~#
```

10. 다시 Cilium 상태 확인하기

cilium status

```
root@master:~# cilium status
    /¯¯\
 /¯¯\__/¯¯\    Cilium:             OK
 \__/¯¯\__/    Operator:           OK
 /¯¯\__/¯¯\    Envoy DaemonSet:    disabled (using embedded mode)
 \__/¯¯\__/    Hubble Relay:       disabled
    \__/       ClusterMesh:        disabled

Deployment             cilium-operator    Desired: 1, Ready: 1/1, Available: 1/1
DaemonSet              cilium             Desired: 3, Ready: 3/3, Available: 3/3
Containers:            cilium             Running: 3
                       cilium-operator    Running: 1
Cluster Pods:          2/2 managed by Cilium
Helm chart version:
Image versions         cilium             quay.io/cilium/cilium:v1.15.5@sha256:4ce1666a73815101ec9a4d360af6c5b7f1193ab00d89b7124f8505dee147ca40: 3
                       cilium-operator    quay.io/cilium/operator-generic:v1.15.5@sha256:f5d3d19754074ca052be6aac5d1ffb1de1eb5f2d947222b5f10f6d97ad4383e8: 1
root@master:~#
```

설치 중 장애 발생 시 참고

1. master node에서

   # **kubeadm reset**

   # **rm -rf /var/lib/cni/**

   # **systemctl daemon-reload**

2. worker1 node에서

   # **kubeadm reset**

3. worker2 node에서

   # **kubeadm reset**