**### 권장 실습 환경 ###**

**01. Windows 권장**

**02. 한글 계정명 사용 금지**

**03. C드라이브 여유공간 50GB 이상**

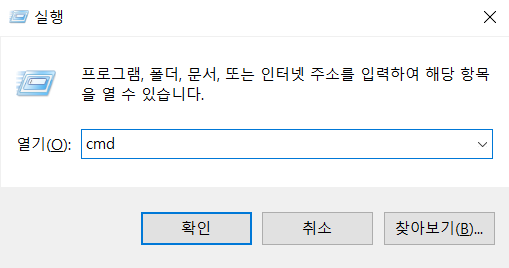
**04. 메모리 16GB 이상 권장**

**05. 안정적인 네트워크 환경** (가급적 유선연결)

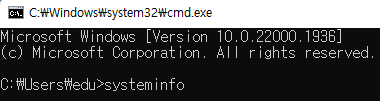
**### Hyper-V 설정 확인 ###**

**\* 실습용 Windows 시스템에 Hyper-V 기능이 활성화되어 있으면 실습 불가**

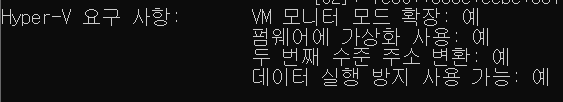
**01. Windows Key + r** >>> **실행** >>> **cmd** (입력 후 엔터)



**02.** C:\Users\edu> **systeminfo** (입력 후 엔터)



**03. 출력 결과 확인 (정상)**



**04. 추가 확인 필요**



>>> 제어판 >>> 프로그램 >>> Windows 기능 켜기/끄기 >>> Windows 하이퍼바이저 플랫폼 (확인)

**05.** C:\Users\edu> **exit** (입력 후 엔터)

**### VirtualBox 설치 및 master VM Import ###**

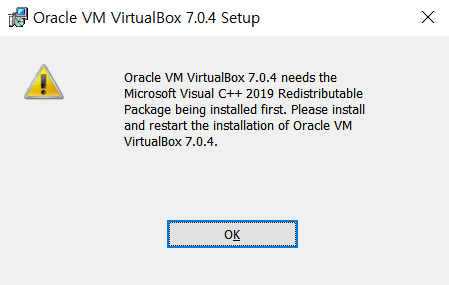
**01. VirtualBox 다운로드 및 설치 (VMware, Hyper-V 제거 필요)**

- [**https://www.virtualbox.org**](https://www.virtualbox.org)

>>> **VirtualBox-7.0.6-155176-Win.exe**

- 설치파일 다운로드 후 **관리자 권한**으로 실행 (**한글 계정명 사용 금지**)

**\* Microsoft Visual C++ 2019 Redistributable Package** **Error**



**\* 설치 후 진행**

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

**>>> VC\_redist.x64.exe**

**- VirtualBox 7.0.6 platform packages:**

>>> **VirtualBox-7.0.6-155176-Win.exe**

>>> **관리자 권한으로 실행** (**마우스 오른쪽 버튼 클릭**)

>>> **Next** >>> **Next**

>>> **Yes** >>> **Yes** >>> **Install**

>>> **Finish**

**02. VirtualBox 호스트키 조합 설정**

>>> **파일** >>> **환경 설정** >>> **입력** >>> **가상 머신**

>>> **호스트 키 조합** >>> **Shift + Ctrl + Alt** (동시에 누르기)

>>> **확인**

**03. Ubuntu VM Import - (Ubuntu Linux Server 20.04)**

>>> **파일** >>> **가상 시스템 가져오기** >>> **파일(^)**

>>> **Ubuntu.ova** (선택) >>> **열기** >>> **다음**

>>> **MAC 주소 정책** >>> **모든 네트워크 어댑터의 새 MAC 주소 생성** (**반드시 선택하여 변경**)

>>> **Finish**

**\* VM Import 장애발생 시 사용자에 생성된 폴더 삭제 후 재실행**

>>> C:\Users\**edu**\VirtualBox VMs\**Ubuntu**

>>> **폴더 삭제 후 다시 진행**

**04. VirtualBox - NatNetwork 설정(192.168.137.0/24)**

>>> **도구** >>> **네트워크**

>>> **NAT Networks** (선택) >>> 만들기**(+)** (클릭) >>> NatNetwork (생성됨)

>>> **General Options** >>> **IP4 Prefix: 192.168.137.0/24** (**아이피주소 변경**)

>>> **적용** (클릭)

>>> **포트 포워딩** (선택) >>> **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**

>>> **적용**

****

- **Ubuntu**

>>> 설정 >>> **네트워크**

>>> 다음에 연결됨: **NAT 네트워크** (변경)

>>> 이름: **NatNetwork** >>> **확인**

**05. Ubuntu VM (헤드리스) 시작** >>> **‘Windows Defender 방화벽~’ (액세스 허용)**

>>> ID: **root**

>>> PW: **ubuntu**

**# ip a**

**# ifconfig**

**# ping www.google.com**

>>> [Ctrl + C]

**# exit**

**### PuTTY Login ###**

[https://www.putty.org](https://www.putty.org/)

**01. PuTTY Login Session 정보 설정**

>>> Host Name (or IP address): **192.168.56.1** (입력)

>>> Port: **100** (**수정**)

>>> **Category**

>>> **Terminal** >>> **Keyboard** >>> **The Backspace key** >>> **Control-H** (선택)

>>> **Window** >>> **Appearance** >>> **Font settings** >>> **Change…** (클릭 후 변경)

>>> **Colours** >>> **Default Foreground/글자색 & Default Background/배경색** >>> **Modify** (클릭 후 변경)

- **Saved Sessions**

>>> **Session** (클릭) >>> **Ubuntu** (입력) >>> **Save** >>> **Open**

>>> 최초 접속 시 PuTTY Security Alert 발생 >>> **Accept** (선택)

>>> **login as:** root

>>> **root@192.168.56.1’s password:** ubuntu

- ID: **root**

- PW: **ubuntu**

**02. IP 정보 확인**(**192.168.137.100/24**)

>>> 명령어 **복사[Ctrl + C]**/**붙여넣기[마우스 오른쪽 버튼 클릭]** 가능

**# ip a**

**# ifconfig**

**# ping www.google.com**

>>> [Ctrl + C]

**### Docker Package Installation ###**

**01. Host 장비 인터넷 연결 상태 확인**

# ping www.google.com

>>> [Ctrl + C]

**02. Host 운영체제 커널 버전(3.8 이상) 및 배포판 리눅스 버전 확인**

# uname -r

# cat /etc/os-release

**03. 레파지토리 정보 갱신**

# apt update

**04. https 기반 레파지토리 사용 지원 패키지 설치**

# apt install -y apt-transport-https ca-certificates curl software-properties-common

**05. Docker 공식 레파지토리에서 패키지 다운 시 위변조 확인을 위한 GPG 키 추가**

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

**06. docker.com GPG키 등록 확인**

# apt-key fingerprint

**07. Docker 공식 레파지토리 등록**

# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

**08. 저장소 등록정보 확인**

# grep docker /etc/apt/sources.list

**09. 레파지토리 정보 갱신**

# apt update

**10. Docker Container Engine 설치**

# apt install -y docker-ce

**11. 설치 확인**

# docker --version

# docker version

**12. 상세 정보 확인**

# docker info

**13. 사용량 확인**

# docker system df

**14. 실습교재 Github**

>>> <https://github.com/bjpublic/core_kubernetes>

**15. 실습 Github Clone**

# git clone https://github.com/bjpublic/core\_kubernetes

**16. etc**

# docker container run -d **--name test** nginx

# docker container run -it **--name centos** centos

# docker container **exec -it** centos /bin/bash

# docker container run -d **--name web** -p 8080:80 nginx

# curl localhost:8080



>>> chrome >>> 192.168.56.1:8080

**### Docker Registry ###**

**Docker Hub**

**01. Docker Hub Site 계정 생성**

>>> http://hub.docker.com

**02. 가입 시 실제 수신 가능한 이메일 사용**

>>> 수신된 메일 확인 후 계정 생성 완료

**### Docker 실습 내용 삭제 ###**

# docker container rm -f $(docker ps -a -q)

# docker container ps -a

# docker image rm -f $(docker image ls -q)

# docker image ls

# **poweroff**

**### 가상머신 Snapshot ###**

**01. Ubuntu 가상머신 이름 변경**

>>> **Ubuntu** >>> 일반 >>> 이름: **master** (변경)

>>> **확인**

**02. master 가상머신 스냅샷**

>>> **master** >>> **스냅샷** >>> **찍기(+)**

>>> 스냅샷 이름: **Docker** >>> 스냅샷 설명: **Kubernetes 설치 전**

>>> **확인**

**### Kubernetes Installation ###**

**>>> master**

**01. 쿠버네티스 클러스터 구성 전 요구사항 확인**

- 쿠버네티스가 지원하는 배포판 여부 확인 (대부분 **Debian**, Redhat 계열 배포판 지원)

- **2GB 이상 메모리**

>>> # free

- **2 CPUs or more**

>>> # lscpu

- 클러스터내의 모든 노드 간의 통신 상태 확인 (public or private network is fine)

- 모든 노드의 고유 hostname, MAC address, and product\_uuid 정보 확인

**02. master 노드 호스트네임 변경(호스트네임: ubuntu >>> master)**

# hostname

# hostnamectl set-hostname master

# hostname

# vi /etc/hosts

-------------------------------------------------------------------------------

192.168.137.100 master

192.168.137.101 worker1

192.168.137.102 worker2

-------------------------------------------------------------------------------

**03. IP 주소 및 UUID 확인**

# ip a

# ifconfig

# cat /sys/class/dmi/id/product\_uuid

**04. kubelet 적절한 동작을 위해서 swap 사용 중지 설정**

# swapon

# swapoff -a

# swapon

# tail -1 /etc/fstab

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

# tail -1 /etc/fstab

**05. SELinux와 방화벽 해제**

# setenforce 0

# ufw disable

# ufw status

**06. iptables가 bridged traffic을 확인하고 제어 할 수 있도록 설정**

>>> **br\_netfilter 모듈 Load**

----------------------------------------------------------------------- **마지막줄 EOF 뒤쪽에 공백이 없도록 주의!!!**

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

br\_netfilter

EOF

-----------------------------------------------------------------------

>>> **네트워크 파라미터 설정**

----------------------------------------------------------------------- **마지막줄 EOF 뒤쪽에 공백이 없도록 주의!!!**

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

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

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

EOF

-----------------------------------------------------------------------

>>> **시스템에 설정 적용**

# sysctl --system

**07. Installing kubeadm, kubelet and kubectl (Debian-based distributions)**

**1) https 기반 레파지토리 사용 지원 패키지 설치**

# apt update

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

**2) 구글 클라우드의 공개 사이닝 키 다운로드**

# mkdir /etc/apt/keyrings

# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.27/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

**3) 쿠버네티스 apt 레파지토리 추가**

# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.27/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

**4) kubelet, kubeadm, kubectl 설치 후 버전을 고정**

# apt update

# apt **install** -y **kubelet=1.27.12-1.1** **kubeadm=1.27.12-1.1** **kubectl=1.27.12-1.1**

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

**5) kubeadm 설치 확인**

# kubeadm version

**08. cri-dockerd 설치 (https://github.com/Mirantis/cri-dockerd)**

# cd ~

# git clone https://github.com/Mirantis/cri-dockerd.git

# ls -l cri-dockerd

# wget https://github.com/Mirantis/cri-dockerd/releases/download/v0.3.3/cri-dockerd-0.3.3.amd64.tgz

# tar -zxvf cri-dockerd-0.3.3.amd64.tgz

>>> 스크립트 생성

# vi install.sh

------------------------------------------------------------------------------------------

# Run these commands as root

cd cri-dockerd

mkdir -p /usr/local/bin

install -o root -g root -m 0755 cri-dockerd /usr/local/bin/cri-dockerd

install packaging/systemd/\* /etc/systemd/system

sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service

systemctl daemon-reload

systemctl enable cri-docker.service

systemctl enable --now cri-docker.socket

------------------------------------------------------------------------------------------

>>> 스크립트 실행

# chmod +x install.sh

# ./install.sh

>>> 시스템 적용

# systemctl start cri-docker.service

# systemctl status cri-docker

**08. Configuring the kubelet cgroup driver**

# ls -ld /etc/docker

>>> Docker가 설치된 상태에서 기존 디렉토리가 있을 경우 다시 생성하지 않음

# mkdir /etc/docker

>>> cGroups이 컨테이너 런타임과 kubelet에 의해 제어될 수 있도록 설정

----------------------------------------------------------------------- **마지막줄 EOF 뒤쪽에 공백이 없도록 주의!!!**

# cat << EOF | tee /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

-----------------------------------------------------------------------

# cat /etc/docker/daemon.json

# systemctl enable docker

# systemctl daemon-reload

# systemctl restart docker

**10. master 노드 종료**

# poweroff

**### worker1 복제 및 환경설정 ###**

**01. worker1 복제**

>>> **master** (마우스 오른쪽 버튼 클릭) >>> **복제**

>>> 이름: **worker1**

>>> MAC 주소 정책: **모든 네트워크 어댑터의 새 MAC 주소 생성** (**반드시 선택하여 변경**)

>>> 추가 옵션: **체크안함**

>>> **다음** >>> **완전한 복제** >>> **다음** >>> **현재 머신 상태** >>> **Finish**

**02. worker1 환경설정**

>>> **worker1 (헤드리스) 시작**

>>> **로그인** >>> **root**/**ubuntu**

>>> **hostname 변경**

# hostname

# hostnamectl set-hostname worker1

# hostname

>>> **IP Address 변경**

# ip a

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

------------------------------------------------------------------------------------------------------

addresses: [192.168.137.10**0**/24] >>> 10**0**을 10**1**로 변경

------------------------------------------------------------------------------------------------------

# cat /etc/netplan/00-installer-config.yaml

>>> 변경될 IP 주소 확인

# netplan apply

>>> IP 주소 변경 적용

# ip a

**03. 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** >>> **확인** >>> **확인**

**04. worker1 PuTTY Login Session 정보 설정**

>>> **PuTTY 실행** >>> **master** (선택) >>> **Load**

>>> HostName: **192.168.56.1**

>>> Port: **101** (수정)

>>> **Category**

>>> **Colours** >>> **Default Background/배경색** >>> **Modify** (클릭 후 변경)

- **Saved Sessions**

>>> **Session** (클릭) >>> **worker1** (입력) >>> **Save** >>> **Open**

>>> 최초 접속 시 PuTTY Security Alert 발생 >>> **Accept** (선택)

>>> **login as:** root

>>> **root@192.168.56.1’s password:** ubuntu

- ID: **root**

- PW: **ubuntu**

# ip a

# ping www.google.com

>>> [Ctrl + C]

# ping worker1

>>> [Ctrl + C]

**### worker2 복제 및 환경설정 ###**

**01. worker2 복제**

>>> **master** (마우스 오른쪽 버튼 클릭) >>> **복제**

>>> 이름: **worker2**

>>> MAC 주소 정책: **모든 네트워크 어댑터의 새 MAC 주소 생성** (**반드시 선택하여 변경**)

>>> 추가 옵션: **체크안함**

>>> **다음** >>> **완전한 복제** >>> **다음** >>> **현재 머신 상태** >>> **Finish**

**02. worker2 환경설정**

>>> **worker2 (헤드리스) 시작**

>>> **로그인** >>> **root**/**ubuntu**

>>> **hostname 변경**

# hostname

# hostnamectl set-hostname worker2

# hostname

>>> **IP Address 변경**

# ip a

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

------------------------------------------------------------------------------------------------------

addresses: [192.168.137.10**0**/24] >>> 10**0**을 10**2**로 변경

------------------------------------------------------------------------------------------------------

# cat /etc/netplan/00-installer-config.yaml

>>> 변경될 IP 주소 확인

# netplan apply

>>> IP주소 변경 적용

# ip a

**03. 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** >>> **확인** >>> **확인**

**04. worker2 PuTTY Login Session 정보 설정**

>>> **PuTTY 실행** >>> **master** (선택) >>> **Load**

>>> HostName: **192.168.56.1**

>>> Port: **102** (수정)

>>> **Category**

>>> **Colours** >>> **Default Background/배경색** >>> **Modify** (클릭 후 변경)

- **Saved Sessions**

>>> **Session** (클릭) >>> **worker2** (입력) >>> **Save** >>> **Open**

>>> 최초 접속 시 PuTTY Security Alert 발생 >>> **Accept** (선택)

>>> **login as:** root

>>> **root@192.168.56.1’s password:** ubuntu

- ID: **root**

- PW: **ubuntu**

# ip a

# ping www.google.com

>>> [Ctrl + C]

# ping worker1

>>> [Ctrl + C]

# ping worker2

>>> [Ctrl + C]

**>>> master**

**- master VM Power-On**

# ping worker1

# ping worker2

# ping www.google.com

**01.** (**마스터 노드에서만 실행**) **kubeadm init 명령을 통해서 클러스터를 생성 >>> 터미널 가로 최대화 후 진행**

>>>최대 10분 정도 소요. **작업 종료 후 터미널 조작 금지!!!!!**

# **kubeadm init --cri-socket=unix:///var/run/cri-dockerd.sock**

----------------------------------------------------------------------------------------------------------------------------------------------

......

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 192.168.137.101:6443 --token k73e80.otf7kbdnbxvc9qvk \**

**--discovery-token-ca-cert-hash sha256:76507fe082ede68174b6e324f3c72dd0bfd70b78a186c7b46efa26fbf893fd74**

----------------------------------------------------------------------------------------------------------------------------------------------

**02. 클러스터 조인 토큰 복사 및 저장**

>>> **01번 실행결과의 마지막 내용** (join token)

>>> Windows + r >>> notepad (실행) >>> join\_token.txt

**03. root 사용자 클러스터의 API 접근 인증 설정**

>>> kubeconfig 파일의 위치를 root 사용자의 KUBECONFIG 쉘 환경변수에 추가

# vi ~/.bashrc

-------------------------------------------------------------------------------

export KUBECONFIG=/etc/kubernetes/admin.conf

-------------------------------------------------------------------------------

>>> **붉은색 내용 마지막줄에 추가**

# source ~/.bashrc

# echo $KUBECONFIG

**04. Pod 통신을 위한 CNI(Container Network Interface) 기반 Pod 네트워크 추가**

>>> 설치되기 전에는 클러스터 DNS (CoreDNS)가 시작되지 않음

>>> https://docs.projectcalico.org/getting-started/kubernetes/self-managed-onprem/onpremises

# wget https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml

# kubectl apply -f calico.yaml

**05. 클러스터 구성 상태 확인**

>>> **master에서만 가능**

# kubectl get nodes

# kubectl get pods --all-namespaces

>>> **STATUS가 모두 Running이 될때까지 확인**

**>>> worker1**

**01. worker1를 클러스터에 조인**

(**worker1**) # **kubeadm join 192.168.137.100:6443 --token 77uebw.yt3nrsfacvsybjmf \**

**--discovery-token-ca-cert-hash sha256:51964ee4461dc5d1227be04b51825829aa09dee7fc252e1634e741c4dc03d23d \**

**--cri-socket=unix:///var/run/cri-dockerd.sock**

**>>> worker2**

**01. worker2를 클러스터에 조인**

(**worker2**) # **kubeadm join 192.168.137.100:6443 --token 77uebw.yt3nrsfacvsybjmf \**

**--discovery-token-ca-cert-hash sha256:51964ee4461dc5d1227be04b51825829aa09dee7fc252e1634e741c4dc03d23d \**

**--cri-socket=unix:///var/run/cri-dockerd.sock**

**>>> master**

**01. worker 클러스터 조인 확인**

# kubectl get nodes

NAME STATUS ROLES AGE VERSION

master **Ready** control-plane,master 13m v1.21.0

worker1 **Ready** <none> 104s v1.21.0

worker2 **Ready** <none> 104s v1.21.0

# kubectl get pods --all-namespaces -o wide

# kubectl cluster-info

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**>>> 설치 중 장애 발생 시 참고**

(**master**) # kubeadm reset

(**master**) # rm -rf /var/lib/cni/

(**master**) # systemctl daemon-reload

(**worker1**) # kubeadm reset

(**worker2**) # kubeadm reset

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**### 템플릿 YAML ###**

>>> **apiVersion 확인**

# kubectl api-resources

# kubectl api-resources | grep -i -e deploy -e APIVERSION

**### Monitoring ###**

>>> **k8s 상태 모니터링**

# watch -n 1 kubectl get pods -o wide --show-labels

**### Static Pod ###**

**01. master 노드 부팅 시 kubelet 데몬이 실행하는 yaml 파일**

>>> /etc/kubernetes/manifests/\*.yaml

# ls -l /etc/kubernetes/manifests

* **수고하셨습니다 -**