클러스터 생성 및 설정 (Kubeadm)

본 **"Vanilla Kubernetes"**는 광범위한 사용 사례에 적합한 쿠버네티스의 표준 버전을 의미하다. 이 버전은 쿠버네티스의 핵심 기능들을 모두 포함하고, 이번 데모에서는 한 물리 서버

*VMWare Workstation*에서 실습 진행

- Master Node: 192.168.1.10
- Worker Node 1: 192.168.1.20
- Worker Node 2: 192.168.1.30

> 쿠버네티스 클러스터 설정 시 Swap Memory 비활성화를 권장 한다. Swap Memory는 하드 디스크를 이용하여 RAM을 보충하는 가상 메모리이지만, K8s는
>
> 메모리를 비활성화해야 한다. 더 자세한 내용은 여기 블로그

CentOS 7에서 K8s 클러스터

Docker Installation

시작 전, 이전 버전들을 삭제 한다

```
sudo dnf remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-engine
```

yum-utils 설치와 로컬 yum repository (저장소) 지정

```
sudo dnf install -y yum-utils
```

```
sudo dnf-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

도커 최신 버전 설치

```
sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

도커 서비스 시작

```
sudo systemctl start docker
```

도커 버전 확인

```
docker version
```

Kubernetes Installation

☑ *Master Node*과 *Worker Node*에서 동일 하게 작업

Swap Memory 비활성화

```
swapoff -a
```

daemon.json 파일 생성 후 cgroupdriver을 systemd로 설정

```
vi /etc/docker/daemon.json


{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
```

daemon 재시작

```
systemctl daemon-reload
systemctl restart docker
```

쿠버네티스 로컬 저장소 지정 및 설치

```
vi /etc/yum.repos.d/kubernetes.repo

[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
       https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

# 쿠버네티스 설치
dnf install -y kubelet-1.19.16-0.x86_64 kubectl-1.19.16-0.x86_64 kubeadm-1.19.16-0.x86_64
```

쿠버네티스 설치 완료 확인

```
rpm -qa | grep kube
```

쿠버네티스 클러스터 요소들 통신을 위해 방화벽 포트 추가

```
firewall-cmd --permanent --add-port=80/tcp
firewall-cmd --permanent --add-port=443/tcp
firewall-cmd --permanent --add-port=2376/tcp
firewall-cmd --permanent --add-port=2379/tcp
firewall-cmd --permanent --add-port=2380/tcp
firewall-cmd --permanent --add-port=6443/tcp
firewall-cmd --permanent --add-port=8472/udp
```

```
firewall-cmd --permanent --add-port=9099/tcp
firewall-cmd --permanent --add-port=10250/tcp
firewall-cmd --permanent --add-port=10251/tcp
firewall-cmd --permanent --add-port=10252/tcp
firewall-cmd --permanent --add-port=10254/tcp
firewall-cmd --permanent --add-port=10255/tcp
firewall-cmd --permanent --add-port=30000-32767/tcp
firewall-cmd --permanent --add-port=30000-32767/udp
firewall-cmd --permanent --add-masquerade
firewall-cmd --reload
```

마스터 노드에서만 진행

쿠버네티스 클러스터 생성

```
kubeadm init --apiserver-advertise-address=192.168.1.10 --pod-network-cidr=10.244.0.0/16

# K8s control plane 생성 완료 화면
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 192.168.1.10:6443 --token 172vji.r0u77jcmcnccm6no \
```

```
    --discovery-token-ca-cert-hash
sha256:72b9648c647f724ab52471847cb06c47b23097375f2e67633b745fc69db16e8d
```

kubectl 활성화 위해 admin.conf 복사 작업

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

네트워크 flannel 다운로드 (

[CNI(Container Network Interface)는 컨테이너를 위한 네트워킹을 제어할 수 있는 플러그인을 만들기 위한 표준이라고 한다. 간단하게 설명 하면 CNI플러그인을 쿠버

더 자세한 내용은 링크 참고

```
curl -O -L https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

kube-flannel.yml 파일 안에 --iface=(가상 인터페이스 이름) 지정

```
vi kube-flannel.yml

args:
        - --ip-masq
        - --kube-subnet-mgr
        - --iface=ens160
```

네트워크 플러그인 YAML 파일 적용 후 kubelet 재시작

```
kubectl apply -f kube-flannel.yml
systemctl restart kubelet
```

*Worker Node*에서 작업

Master Node에서 받은 Discovery Token 값을 join 명령어로 통해 Master과 Worker 노드 묶기

```
kubeadm join 192.168.1.10:6443 --token 172vji.r0u77jcmcnccm6no \
    --discovery-token-ca-cert-hash
sha256:72b9648c647f724ab52471847cb06c47b23097375f2e67633b745fc69db16e8d

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.
```

Join 작업이 완료 후 마스터 노드에서 쿠버네티스 클러스터 확인

```
kubectl get nodes
NAME     STATUS   ROLES    AGE    VERSION
master   Ready    master   107m   v1.19.16
node-1   Ready    <none>   91s    v1.19.16
node-2   Ready    <none>   48s    v1.19.16
```

Rocky Linux 9.1에서 K8s 클러스터

Docker Installation

업데이트 확인 및 업데이트 있으면 실행

```
sudo dnf check-update
sudo dnf update
```

도커 repository (저장소) 추가

```
sudo dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

도커 설치

```
sudo dnf install docker-ce docker-ce-cli containerd.io
```

도커 시작 및 확인

```
sudo systemctl start docker
sudo systemctl enable docker

sudo docker version
```

'sudo' 생략 하고 docker 만 쓰고 싶으면

```
sudo usermod -aG docker $(whoami)
```

Kubernetes Installation

☑*Master Node*과 *Worker Node*에서 동일 하게 작업

Swap Memory 비활성화

```
sudo swapoff -a
```

daemon.json 파일 생성 후 cgroupdriver을 systemd로 설정

```
sudo vi /etc/docker/daemon.json

{
   "exec-opts": ["native.cgroupdriver=systemd"]
}
```

daemon 재시작

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

쿠버네티스 로컬 저장소 지정 및 설치

```
sudo vi /etc/yum.repos.d/kubernetes.repo

[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
       https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

# 쿠버네티스 설치
sudo dnf -y kubelet-1.19.16-0.x86_64 kubectl-1.19.16-0.x86_64 kubeadm-1.19.16-0.x86_64
```

쿠버네티스 설치 완료 확인

```
sudo rpm -qa | grep kube
```

쿠버네티스 클러스터 요소들 통신을 위해 방화벽 포트 추가

```
sudo firewall-cmd --permanent --add-port=80/tcp
sudo firewall-cmd --permanent --add-port=443/tcp
sudo firewall-cmd --permanent --add-port=2376/tcp
sudo firewall-cmd --permanent --add-port=2379/tcp
sudo firewall-cmd --permanent --add-port=2380/tcp
sudo firewall-cmd --permanent --add-port=6443/tcp
sudo firewall-cmd --permanent --add-port=8472/udp
```

```
sudo firewall-cmd --permanent --add-port=9099/tcp
sudo firewall-cmd --permanent --add-port=10250/tcp
sudo firewall-cmd --permanent --add-port=10251/tcp
sudo firewall-cmd --permanent --add-port=10252/tcp
sudo firewall-cmd --permanent --add-port=10254/tcp
sudo firewall-cmd --permanent --add-port=10255/tcp
sudo firewall-cmd --permanent --add-port=30000-32767/tcp
sudo firewall-cmd --permanent --add-port=30000-32767/udp
sudo firewall-cmd --permanent --add-masquerade
sudo firewall-cmd --reload
```

마스터 노드에서만 진행

슈퍼 유저 (root 계정)으로 전환

```
sudo su
```

쿠버네티스 클러스터 생성

```
kubeadm init --apiserver-advertise-address=192.168.1.10 --pod-network-cidr=10.244.0.0/16

# K8s control plane 생성 완료 화면
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:

sudo kubeadm join 192.168.1.10:6443 --token 1x7qb2.dww3u7mjsrq2hoxt --discovery-token-ca-cert-hash
sha256:10a6803e9a45bb029af4ad3c1d0d894dfaee9980d2318c495739296daeffb9eb
```

kubectl 활성화를 위해 admin.conf 복사 작업 (꼭 root계정으로 해야됨!)

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

네트워크 플러그인 다운로드

```
curl -O -L https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

네트워크 플러그인 YAML 파일 적용 후 kubelet 재시작 (꼭 root계정으로 해야됨!)

```
kubectl apply -f kube-flannel.yml
systemctl restart kubelet
```

*Worker Node*에서 작업

Master Node에서 받은 Discovery Token 값을 join 명령어로 통해 Master과 Worker 노드 묶기

```
sudo kubeadm join 192.168.1.10:6443 --token 1x7qb2.dww3u7mjsrq2hoxt \
--discovery-token-ca-cert-hash
sha256:10a6803e9a45bb029af4ad3c1d0d894dfaee9980d2318c495739296daeffb9eb

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.
```

Join 작업이 완료 후 마스터 노드에서 쿠버네티스 클러스터 확인

```
sudo kubectl get nodes
NAME      STATUS   ROLES    AGE    VERSION
master    Ready    master   33m    v1.19.16
worker-1  Ready    <none>   91s    v1.19.16
worker-2  Ready    <none>   48s    v1.19.16
```

Ubuntu 20.04.6에서 K8s 클러스터

Docker Installation

업데이트 확인 및 실행

```
sudo apt update
```

필수 패키지 설치

```
sudo apt install ca-certificates curl gnupg lsb-release
```

도커 GPG 키 추가

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

도커 Repository 등록

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

도커 엔진 설치 및 확인

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io

sudo docker version
sudo systemctl status docker

## 서버 재시작 시 docker 가 자동으로 시작되도록 설정 ##
sudo systemctl enable docker
```

Kubernetes Installation

*Master Node*과 *Worker Node*에서 동일 하게 작업

Swap Memory 비활성화

```
sudo swapoff -a
```

노드간 통신을 위한 iptables에 브릿지 관련 설정 추가

```
sudo vi /etc/modules-load.d/k8s.conf

br_netfilter

sudo vi /etc/sysctl.d/k8s.conf

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

```
# sysctl.conf 파일 맨 아래에도 동일 하게 두 줄 추가
sudo vi /etc/sysctl.conf

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

# sysctl 설정 파일들의 변동을 저장 하기 위해
sudo sysctl --system
sudo sysctl -p
```

daemon.json 파일 생성 후 cgroupdriver을 systemd로 설정

```
sudo vi /etc/docker/daemon.json


{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
```

daemon 재시작

```
sudo systemctl daemon-reload
sudo systemctl restart docker
sudo systemctl enable docker
```

apt 업데이트 및 ca 관련 패키지 다운로드 및 설

```
sudo apt update
sudo apt install -y apt-transport-https ca-certificates curl
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

도커 gpg 및 소스 리스트 내용 추가 및 apt 업데이

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/
kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt update
```

kubeadm, kubelet 그리고 kubectl 설치 및 활성화

```
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

sudo systemctl start kubelet && systemctl enable kubelet
```

마스터 노드에서만 진행

쿠버네티스 클러스터 생성을 위해 kubeadm init 실행

```
sudo kubeadm init --apiserver-advertise-address=192.168.1.10 --pod-network-cidr=10.244.0.0/16

# K8s control plane 생성 완료 화면
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 192.168.1.10:6443 --token mu72kx.sn06xzcg3lde7mha \
--discovery-token-ca-cert-hash
sha256:d134654458b474c796667776ab8175adbc0e1dc4bc21a712a19a7bb405ee9273
```

kubectl 활성화를 위해 admin.conf 복사 작업

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

네트워크 플러그인 다운로드

```
curl -O -L https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

네트워크 플러그인 YAML 파일 적용 후 kubelet 재시작

```
kubectl apply -f kube-flannel.yml
sudo systemctl restart kubelet
```

*Worker Node*에서 진행

Master Node에서 받은 Discovery Token 값을 join 명령어로 통해 Master과 Worker 노드 묶기

```
sudo kubeadm join 192.168.1.10:6443 --token mu72kx.sn06xzcg3lde7mha \
--discovery-token-ca-cert-hash
sha256:d134654458b474c796667776ab8175adbc0e1dc4bc21a712a19a7bb405ee9273

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.
```

Join 작업이 완료 후 마스터 노드에서 쿠버네티스 클러스터 확인

```
kubectl get nodes
NAME       STATUS   ROLES    AGE    VERSION
master     Ready    master   33m    v1.19.16
worker-1   Ready    <none>   91s    v1.19.16
worker-2   Ready    <none>   48s    v1.19.16
```

Error Troubleshooting

[ERROR CRI]: container runtime is not running [Issue Encountered]

이 문제는 사용되는 CRI가 Containerd인 경우 마스터 노드에서 kubeadm init 명령을 실행할 때 자주 발생한다. 대부분의 경우 config.toml 파일에 문제가 있다.

해결

```
# config.toml 파일을 삭제 한다
sudo rm /etc/containerd/config.toml

# containerd 서비스 재시작
sudo systemctl restart containerd
```

The connection to the server <IP 주소>:6443 was refused - did you specif the right host or port?

이 에러는 보통 API 서버가 실행 중이지 않거나 네트워킹 문제가 있는 경우 발생할 수 있다. 하지만 이 애러가 마스터 노드 재시작 후 발생 한다면 Swap 메모리 문제일 가능성이 높다. 그런 경우

해결

/etc/fstab 설정 파일안에서 swap을 주석 처리 해준다

```
sudo vi /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>
# / was on /dev/sda3 during curtin installation
/dev/disk/by-uuid/05bb7e29-bd2d-4ffb-86c6-8868e48548f4 / ext4 defaults 0 1
# /boot/efi was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/ebbd40d6-5f57-4214-806f-8cf8e929b23d /boot/efi ext4 defaults 0 1
# /swap.img     none    swap    sw      0       0
```

Revision #9
Created 29 May 2023 05:19:17 by □□
Updated 31 May 2023 00:46:39 by □□