Kubernetes 1.24 + cri-docker Installation (kubeadm)

우분투에서 Kubernetes 최신 버전 설치하는 방법입니다.
18.04 에서 시험되었으나 상위버전에서도 차이는 없습니다.

Controll Node (node01) , Worker Node (node02 ,node03) 3EA 구성입니다.
모든 설치는 root 가 아닌 `user(worker)`을 사용하며 `sudo` 를 사용합니다

(Pre-Option) 설치 패키지 캐쉬 초기화

```
sudo kill -9 $(lsof -t /var/lib/dpkg/lock)
sudo kill -9 $(lsof -t /var/lib/apt/lists/lock)
sudo kill -9 $(lsof -t /var/cache/apt/archives/lock)
sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
sudo rm /var/lib/dpkg/lock
sudo dpkg --configure -a
```

(Require) SWAPOFF

```
# SWAP-On 시 퍼포먼스 이슈가 있어 일반적으로 제거
sudo swapoff /swap.img
sudo sed -i -e '/swap.img/d' /etc/fstab
```

(유의사항) Container Runtime Install

k8s 1.24 (2022/05) 이후
k8s 에서 기본적으로 내부 연결 지원해주던 `dockershim` 이 제거되어
cri-docker 추가 설치하여 도커를 k8s에 연결하는 작업이 필요

Pre-Setting – All node

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo systemctl enable --now docker && sudo systemctl status docker --no-pager
sudo usermod -aG docker worker
```

```
sudo docker container ls

# cri-docker Install
VER=$(curl                                                    -s
https://api.github.com/repos/Mirantis/cri-dockerd/releases/lat
est|grep tag_name | cut -d '"' -f 4|sed 's/v//g')
echo $VER
wget
https://github.com/Mirantis/cri-dockerd/releases/download/v${V
ER}/cri-dockerd-${VER}.amd64.tgz
tar xvf cri-dockerd-${VER}.amd64.tgz
sudo mv cri-dockerd/cri-dockerd /usr/local/bin/

# cri-docker Version Check
cri-dockerd --version

wget
https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/
packaging/systemd/cri-docker.service
wget
https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/
packaging/systemd/cri-docker.socket
sudo    mv    cri-docker.socket    cri-docker.service
/etc/systemd/system/
sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-
dockerd,' /etc/systemd/system/cri-docker.service

sudo systemctl daemon-reload
sudo systemctl enable cri-docker.service
sudo systemctl enable --now cri-docker.socket

# cri-docker Active Check
sudo systemctl restart docker && sudo systemctl restart cri-
docker
sudo systemctl status cri-docker.socket --no-pager

# Docker cgroup Change Require to Systemd
sudo mkdir /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
```

```
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

sudo systemctl restart docker && sudo systemctl restart cri-docker
sudo docker info | grep Cgroup

# Kernel Forwarding
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sudo sysctl --system
```

Packages Install – All node

```
sudo apt-get update
sudo apt-get 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

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

# Update 해야 패키지 인식함
sudo apt-get update

# k8s 설치
```

```
sudo apt-get install -y kubelet kubeadm kubectl

# 버전 확인하기
kubectl version --short

Client Version: v1.24.3
Kustomize Version: v4.5.4

# 버전 고정하기
sudo apt-mark hold kubelet kubeadm kubectl
```

k8s Init – Controller Node (Node01)

```
# Controller Node
sudo kubeadm config images pull --cri-socket unix:///run/cri-dockerd.sock

sudo kubeadm init --ignore-preflight-errors=all --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=211.115.207.136 --cri-socket /var/run/cri-dockerd.sock
## --apiserver-advertise-address=203.248.23.161 -> Controller 서버 IP.

# 설치 시 출력되는 join Command 를 복사해둔다.
kubeadm join 211.115.207.136:6443 --token r8gfco.6s7f60dns4vgwcc0 \
                        --discovery-token-ca-cert-hash sha256:7b0f82be076748e67f8615eab0b86a61317bac397f94b2921810231ab14afdcc

# kubeadm 을 root 처럼 사용하기 위한 추가 설정
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Token 기억이 안 날 경우
kubeadm token create --print-join-command

# 서비스 확인 ( Ready , Running )
kubectl get nodes -o wide
kubectl get pod -A
```

```
# Worker Node에서 kubectl 을 사용하려면 admin.conf 를 복사한다
sudo scp /etc/kubernetes/admin.conf worker@node02:/home/worker/admin.conf
sudo scp /etc/kubernetes/admin.conf worker@node03:/home/worker/admin.conf
```

Node Join – Worker Node (Node02 , Node03)

```
# Worker Node Join. --cri-docker sorket 수동 지정해서 가입해야 한다.
init 시 나온 토큰값을 참고한다. 기억이 안 날 경우 커맨드로 확인한다

sudo kubeadm join --token <token> <controlplane-host>:<controlplane-port> --ignore-preflight-errors=all --cri-socket unix:///var/run/cri-dockerd.sock
# sudo kubeadm join [Controller IP]:6443 --token 3nvrgw.33k750dq9klm5omi --discovery-token-ca-cert-hash sha256:b10158bcea37aae0e92ed6b68b4dd1e8213623cc7d406e77eef55fe6196fe346 --cri-socket /var/run/cri-dockerd.sock

# Join
sudo kubeadm join [Controller IP]:6443 --token r8gfco.6s7f60dns4vgwcc0 \
                        --discovery-token-ca-cert-hash sha256:7b0f82be076748e67f8615eab0b86a61317bac397f94b2921810231ab14afdcc --cri-socket /var/run/cri-dockerd.sock

# Check
systemctl status kubelet

# worker 계정에 kubectl 사용 환경변수 추가
cd ~
mkdir -p $HOME/.kube
sudo cp -i ./admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

추가 후 node 에서도 kubectl get nodes -o wide 등을 사용할 수 있게 됨
```

Pod Network Install – Controller Node (Node01)

```
# 예전 버전에는 CNI Plugin 을 셋업하지 않으면 CoreDNS 가 Pending 이었는데
지금은 기본 kube-system-proxy 로도 구성이 잘 되는 것 같다.
```

```
# CNI Mode 확인하기
# Check - kube-proxy 이름 확인한다. kube-proxy 는 Node 마다 존재하므로 3개일 것이다.
kubectl get pod -A
kubectl logs -f pod/kube-proxy-[name] -n kube-system

# Log 로 확인 시 - Linux 커널의 iptables Mode 인 것으로 확인되었다.
kubectl logs -f pod/kube-proxy-6dqp8 -n kube-system | grep mode
I0805 06:32:48.444077       1 server_others.go:578] "Unknown proxy mode, assuming iptables proxy" proxyMode=""
I0805 06:32:48.469028       1 server_others.go:213] "kube-proxy running in dual-stack mod
" ipFamily=IPv4

## Pod Network Plugin Install
다양한 네트워크 플러그인이 있으며, 해당 CKA 과정 글에서는 Callico 의 Plugin 을 사용한다.
Calico Network는 IPIP , VXLAN 등을 지원하는 IPVS 모델이다

kubectl create -f https://projectcalico.docs.tigera.io/manifests/tigera-operator.yaml
curl https://projectcalico.docs.tigera.io/manifests/custom-resources.yaml -O
kubectl create -f custom-resources.yaml
kubectl get pods -A

## Control-Plane ( Controller ) Node 일 경우 Taint (해당 노드 생성 금지) 영향으로 calico-controller-pod가 Pending 걸릴 수 있는데 다음과 같이 실행

## Taint Check
kubectl describe node node01 | grep Taints

kubectl taint nodes --all node-role.kubernetes.io/master-
kubectl taint nodes node01 node-role.kubernetes.io/control-plane:NoSchedule-

## PoD check
kubectl get pod -A
```

```
#kubectl describe pods/calico-kube-controllers-657d56796-xxxxx
-n calico-system

## Calicoctl 로 Status 확인하기
cd /usr/local/bin
sudo                          curl                          -L
https://github.com/projectcalico/calico/releases/download/v3.2
3.3/calicoctl-linux-amd64 -o calicoctl
sudo chmod +x calicoctl

## CNI Type Check
calicoctl get ippool -o wide

## Block Check
sudo calicoctl ipam show --show-blocks

## BGP Protocol Check
sudo calicoctl node status

## Node Endpoint Check
calicoctl get workloadendpoint -A
```

Rejoin or Reset

(Trouble) 설치 중 에러 발생시 클린 삭제 후 재 설치를 진행한다

```
# All Node
sudo systemctl stop kubelet
sudo kubeadm reset -f --cri-socket /var/run/cri-dockerd.sock

sudo rm -rf ~/.kube
sudo rm -rf /root/.kube
sudo rm -rf /var/lib/etcd
sudo rm -rf /etc/kubernetes
```

Calico Network 만 삭제하고 싶다면

```
# ALL Node
kubectl delete -f custom-resources.yaml
kubectl delete -f tigera-operator.yaml
sudo rm -rf /var/run/calico/
sudo rm -rf /var/lib/calico/
```

```
sudo rm -rf /etc/cni/net.d/
sudo rm -rf /var/lib/cni/
sudo rm -rf /opt/cni
sudo reboot
```