Kuberenetes Master (Control-Plane) - Setup (k8s_v1.28 / RHEL8.4)

📌 Info.

- VMware ESXi 환경에서 VM으로 구성
- OS : RedHat Enterprise Linux 8.4
- CPU : 4 core
- Memory : 8 GB
- Disk : 100 GB
- Kuberetes ver : v1.28.15
- Container Runtime : Containerd 사용 (RHEL 8.4 버전)
- Container Network Interface : Calico v3.19

1. local.repo 구성

```
lsblk
fdisk -l

mkdir /iso
mkdir /mnt/cdrom

mount /dev/sr0 /mnt/cdrom

cp -rvf /mnt/cdrom/* /iso
```

```
umount /mnt/cdrom
```

```
vi /etc/yum.repo.d/local.repo
---
[BaseOS]
name=BaseOS
baseurl=file:///iso/BaseOS
enabled=1
gpgcheck=0

[AppStream]
name=AppStream
baseurl=file:///iso/AppStream
enabled=1
gpgcheck=0
---
```

```
yum clean all

yum repolist

yum list
```

- 필요 pkg 확인

```
yum list | grep -i <PKG_NAME>

# 예시
yum list | grep -i network-scripts
yum list | grep -i chrony
```

2. firewalld, NetworkManager Disable/Stop

- 네트워크 설정 및 편의성에 따라 진행

```
# firewalld
systemctl status firewalld

systemctl stop firewalld

systemctl disable firewalld

# NetworkManager
systemctl status NetworkManager

systemctl stop NetworkManager

systemctl disable NetworkManager
```

3. selinux 비활성화

```
vi /etc/selinux/config
---
...
SELINUX=disabled  # enforcing → disabled 로 변경
...
---
```

- 확인

```
cat /etc/selinux/config | grep 'SELINUX=disabled'
```

4. Network Install/Start/Enable

```
yum install -y network-scripts

systemctl status network

systemctl start network

systemctl enable network

systemctl is-enabled network

systemctl status network
```

5. swap 기능 비활성화

```
swaopoff -a

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

# 확인
# swaopon --show
swapon -s

# swaop 영역 확인
free -h
```

6. `network-scripts` 설정

- 네트워크 인터페이스, IP 등 세부 설정은 상황에 맞게 변경

```
vi /etc/sysconfig/network-scripts/ifcfg-ens192
---
TYPE=Ethernet
#PROXY_METHOD=none
#BROWSER_ONLY=no
BOOTPROTO=static
#DEFROUTE=yes
#IPV4_FAILURE_FATAL=no
#IPV6INIT=yes
#IPV6_AUTOCONF=yes
#IPV6_DEFROUTE=yes
#IPV6_FAILURE_FATAL=no
NAME=ens192
DEVICE=ens192
ONBOOT=yes
IPADDR=192.168.0.75    # 실제 설정에 맞게 변경
NETMASK=255.255.255.0  # 실제 설정에 맞게 변경
GATEWAY=192.168.0.254  # 실제 설정에 맞게 변경
---
```

```
systemctl restart network
```

- 확인

```
ping -c 2 192.168.0.254

PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=0.374 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=0.351 ms

--- 192.168.0.254 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1049ms
rtt min/avg/max/mdev = 0.351/0.362/0.374/0.022 ms
```

7. hostname, /etc/hosts, /etc/resolv.conf 설정

```
hostnamectl set-hostname k8s-master01

exec bash

[root@k8s-master01 ~]
```

```
vi /etc/hosts
---
# 추가
192.168.0.70 k8s-master01
192.168.0.75 k8s-worker01
---
```

7-1. (인터넷 사용 가능 및 필요 시)(인터페이스 내 설정) DNS 서버 설정

```
cat /etc/sysconfig/network-scripts/ifcfg-ens192
---
TYPE=Ethernet
#PROXY_METHOD=none
#BROWSER_ONLY=no
BOOTPROTO=static
#DEFROUTE=yes
#IPV4_FAILURE_FATAL=no
#IPV6INIT=yes
#IPV6_AUTOCONF=yes
#IPV6_DEFROUTE=yes
#IPV6_FAILURE_FATAL=no
NAME=ens192
DEVICE=ens192
ONBOOT=yes
```

```
IPADDR=192.168.0.75
NETMASK=255.255.255.0
GATEWAY=192.168.0.254
DNS1=8.8.8.8          # 추가
---
```

7-2. `/etc/resolv.conf` 설정

- 추후, `kubeadm init` 으로 클러스터 생성 시, `/etc/reslov.conf` 파일의 존재 여부를 점검합니다.
  오류 방지를 위해 생성 및 설정합니다.

```
vi /etc/resolv.conf
---
# 추가
nameserver 8.8.8.8
---
```

8. NTP 설치 및 시간 동기화

- 클러스터를 구성하는 노드별 시간이 동일 및 동기화 돼야 합니다.

```
yum list | grep -i chrony

yum install -y chrony

systemctl status chronyd

systemctl start chronyd

systemctl enable chronyd

systemctl is-enabled chronyd
```

```
systemctl status chronyd
```

```
# NTP 확인
chronyc tracking

chronyc sources
```

9. containerd 설치 및 설정

9-1. 기존 docker 관련 패키지 삭제 (필수)

- 설치 과정에서 오류 발생 가능하므로, 기존에 설치된 pkg들을 삭제

```
dnf remove docker \\
docker-client \\
docker-client-latest \\
docker-common \\
docker-latest \\
docker-latest-logrotate \\
docker-logrotate \\
docker-engine \\
podman \\
runc
```

9-2. 패키지 다운로드 및 설치

- RHEL 8.4 version 기준

```
mkdir pkg-containerd

cd pkg-containerd
```

```
curl -fsSLO https://download.docker.com/linux/rhel/8/x86_64/stable/Packages/containerd.io-1.7.27-3.1.el8.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/8/x86_64/stable/Packages/docker-buildx-plugin-0.23.0-1.el8.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/8/x86_64/stable/Packages/docker-ce-28.1.1-1.el8.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/8/x86_64/stable/Packages/docker-ce-cli-28.1.1-1.el8.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/8/x86_64/stable/Packages/docker-ce-rootless-extras-28.1.1-1.el8.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/8/x86_64/stable/Packages/docker-compose-plugin-2.35.1-1.el8.x86_64.rpm
```

```
# 확인
containerd.io-1.7.27-3.1.el8.x86_64.rpm
docker-buildx-plugin-0.23.0-1.el8.x86_64.rpm
docker-ce-28.1.1-1.el8.x86_64.rpm
docker-ce-cli-28.1.1-1.el8.x86_64.rpm
docker-ce-rootless-extras-28.1.1-1.el8.x86_64.rpm
docker-compose-plugin-2.35.1-1.el8.x86_64.rpm


# 설치
yum install ./*
```

9-3. containerd 확인 및 start/enable

```
systemctl status docker

systemctl start docker
```

```
systemctl enable docker

systemctl status docker
```

```
systemctl status containerd

systemctl enable containerd
```

9-4. containerd 설정

```
containerd config default | tee /etc/containerd/config.toml

sed -i 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/
config.toml

# 확인
grep 'SystemdCgroup' /etc/containerd/config.toml
            SystemdCgroup = true

systemctl restart containerd
```

10. 커널 모듈 및 파라미터 설정

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF
```

- 확인

```
cat /etc/modules-load.d/containerd.conf
```

- 모듈 명시적으로 로드

```
modprobe overlay
modprobe br_netfilter
```

- 확인

```
[root@k8s-master01 ~]# lsmod | grep overlay
overlay              135168  0

[root@k8s-master01 ~]# lsmod | grep br_netfilter
br_netfilter           24576  0
bridge               192512  1 br_netfilter
```

- IPv4를 포워딩하여 iptables가 브리지된 트래픽을 보게 하기
    - 재부팅 후에도 설정 유지

```
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
```

```
# 재부팅하지 않고, sysctl 파라미터 적용
sysctl --system
```

11. 포트 포워딩 활성화

```
cat /proc/sys/net/ipv4/ip_forward
0 # 결과

echo '1' > /proc/sys/net/ipv4/ip_forward

cat /proc/sys/net/ipv4/ip_forward
1 # 변경 확인
```

12. kubernetes repo 설정

- OS, 커널 버전, cgroup 버전 등에 따라 호환되는 쿠버네티스 버전이 상이하니,
  공식 홈페이지에서 적절한 버전 확인 후 설치해야 합니다.
- 여기서는 v1.28로 진행했습니다.

```
cat <<EOF | sudo tee /etc/yum.repos.d/kuberentes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

- 확인

```
cat /etc/yum.repos.d/kuberentes.repo
```

13. kubelet, kubeadm, kubectl 설치

```
yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
```

- `-disableexcludes=kubernetes`
  - `yum`이 `kubernetes` 에 대해 설정된 모든 제외(excludes) 규칙을 무시하게 합니다.

- 확인

```
kubelet --version

kubeadm version

kubectl version
```

14. kubelet 실행 및 확인

```
systemctl start kubelet

systemctl enable kubelet

systemctl is-enabled kubelet

systemctl status kubelet
```

15. (Master Node) 클러스터 생성

```
kubeadm init --pod-network-cidr=192.168.158.0/24
```

- 성공 메세지
  - 아래의 성공 메세지가 출력 안된다면, 오류 메세지를 읽고 트러블 슈팅이 필요합니다.
  - 아래 join에 사용되는 키 값은 매번 달라지니, 참고만 하시길 바랍니다.

```
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 192.168.0.70:6443 --token gq2w9s.ebkkcq3um7926wwi \\
        --discovery-token-ca-cert-hash sha256:149134142b21de6122e6f4552
970422045d99227d494994c47614b58b7312db3
```

- config 설정
  - 출력 결과에 나온대로 config 설정

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

- 출력 결과 중 다음의 내용은 추후 worker node를 클러스터에 join시킬 때 필요하므로, 따로 저장합니다.

```
kubeadm join 192.168.0.70:6443 --token gq2w9s.ebkkcq3um7926wwi \\
        --discovery-token-ca-cert-hash sha256:149134142b21de6122e6f4552
970422045d99227d494994c47614b58b7312db3
```

- 만약 저장하지 않고 넘어갔다면, 다음의 명령을 마스터 노드에서 실행함으로써 다시 얻을 수 있습니다. 기존 클러스터 노드의 운영에는 영향을 끼치지 않습니다.

```
kubeadm token create --print-join-command
```

15. (Master Node) 클러스터 확인

- 클러스터 확인 시, `STATUS` 가 `NotReady` 일텐데, CNI가 설치 및 설정이 안돼있기 때문입니다.
  정상인 상황이니 다음으로 넘어갑니다.

```
kubectl get nodes

# 출력 결과
NAME          STATUS     ROLES          AGE     VERSION
k8s-master01   NotReady   control-plane   3m53s   v1.28.15
```

16. CNI 설치

해당 사항은 클러스터를 구성하려는 Master node와 Worker node를 join 시킨 후, Master Node에서 수행해야 합니다. 즉, 다음과 같은 상태여야 합니다.

```
kubectl get nodes

# 출력 결과
NAME          STATUS   ROLES          AGE     VERSION
k8s-master01   NotReady   control-plane   6m8s    v1.28.15
k8s-worker01   NotReady   <none>         4m11s   v1.28.15
```

- 쿠버네티스 클러스터에서 구성 요소 간 통신을 가능하게 하는 CNI를 설치합니다.
- 여러 CNI가 존재하지만, 여기서는 `Calico CNI`로 구성하겠습니다.
- `Calico CNI`도 쿠버네티스 버전에 따라 호환되는 버전이 상이합니다.
  반드시 공식 홈페이지에서 호환성을 확인 후, 적절한 버전으로 설치합니다.
- 여기서는 `Kubernetes v1.28`을 호환하는 `calico v3.19`로 구성하겠습니다.

18-1. calico manifest 다운

```
curl -fsSLO https://docs.projectcalico.org/archive/v3.19/manifests/calico.yaml
```

```
ls

# 출력 결과 (현재 위치에 calico.yaml 파일이 다운 받아집니다.)
calico.yaml
```

18-2. calico.yaml 수정

- Pod CIDR를 기본값인 `192.168.0.0/16`로 사용하는 경우, 해당 절차를 생략하고 바로 `apply` 합니다.

- 하지만, 여기서는 `kubeadm init` 으로 클러스터 생성 시, Pod CIDR를 `192.168.158.0/24` 으로 설정 했습니다. 따라서, `calico.yaml` 에서 다음의 내용을 수정해야 합니다.

```
vi calico.yaml
---
# /CALICO_IPV4POOL_CIDR 를 통해 패턴을 찾은 후, 주석 제거 후 변경

...

## 변경 전
            # - name: CALICO_IPV4POOL_CIDR
            #   value: "192.168.0.0/16"

## 변경 후
            - name: CALICO_IPV4POOL_CIDR
              value: "192.168.158.0/24"

...
---
```

- 다음의 내용도 수정해야 합니다.
    - Kubernetes 1.21 버전부터 `policy/v1` 이 정식 API 버전으로 도입되어, Kubernetes v1.25 버전부터 `policy/v1beta1` API 버전의 `PodDisruptionBudget` 이 더 이상 제공되지 않기 때문입니다.

```
vi calico.yaml
---
...

apiVersion: policy/v1 # 기존 v1beta1 을 수정
kind: PodDisruptionBudget
metadata:
  name: calico-kube-controllers
  namespace: kube-system
  labels:
    k8s-app: calico-kube-controllers
spec:
  maxUnavailable: 1
```

```
  selector:
    matchLabels:
      k8s-app: calico-kube-controllers

...
---
```

18-3. calico.yaml 적용

```
kubectl apply -f calico.yaml
```