Kubernetes Worker(Data-Plane) - Setup (k8s_v1.28 / RHEL 9.4)

📌 Info.

- VMware ESXi 환경에서 VM으로 구성
- OS : RedHat Enterprise Linux 9.4
- CPU : 4 core
- Memory : 16 GB
- Disk : 200 GB
- Kuberetes ver : v1.28.15
- Container Runtime : Containerd 사용 (RHEL 9.4 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 Disable/Stop

```
# firewalld
systemctl status firewalld

systemctl stop firewalld

systemctl disable firewalld
```

3. selinux 비활성화

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

- 확인

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

4. swap 기능 비활성화

```
swaopoff -a

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

```
# 확인
# swaopon --show
swapon -s

# swaop 영역 확인
free -h
```

5. `NetworkManager` 설정

- RHEL 9부터는 `iso` 에 `network-scripts` 패키지가 없습니다.
  따라서, 기본으로 사용되는 NetworkManager를 통해서 네트워크를 설정합니다.

```
nmcli con show
nmcli dev show <NET_INTERFACE>

nmcli con down <CON_NAME>

nmcli con mod <CON_NAME> ip4 <IP_ADDR> gw4 <GATEWAY_ADDR> ipv
4.method "manual" connection.autoconnect yes

systemctl restart NetworkManager

nmcli con up <CON_NAME>

nmcli con show
nmcli dev show <NET_INTERFACE>
```

- 확인

```
[root@k8s-worker01 ~] 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
```

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

```
hostnamectl set-hostname k8s-worker01

exec bash

[root@k8s-worker01 ~]
```

```
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

```
nmcli con show
nmcli dev show <NET_INTERFACE>

nmcli con down <CON_NAME>

nmcli con mod <CON_NAME> ipv4.dns "8.8.8.8" # 실제 설정에 맞게 수정

nmcli con up <CON_NAME>
```

```
# 확인
nmcli con show
nmcli dev show <NET_INTERFACE>
```

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.
- RHEL 9 버전에서는 기본으로 설치 및 활성화 돼있습니다. 따라서, 확인만 수행합니다.
  만약, 설치 및 활성화가 안돼있을 시, 직접 설치 및 활성화합니다.

```
# 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 9.4 version 기준

```
mkdir pkg-containerd

cd pkg-containerd


curl -fsSLO https://download.docker.com/linux/rhel/9/x86_64/stable/Packages/containerd.io-1.7.27-3.1.el9.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/9/x86_64/stable/Packages/docker-buildx-plugin-0.23.0-1.el9.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/9/x86_64/stable/Packages/docker-ce-28.1.1-1.el9.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/9/x86_64/stable/Packages/docker-ce-cli-28.1.1-1.el9.x86_64.rpm
```

```
curl -fsSLO https://download.docker.com/linux/rhel/9/x86_64/stable/Packages/docker-ce-rootless-extras-28.1.1-1.el9.x86_64.rpm

curl -fsSLO https://download.docker.com/linux/rhel/9/x86_64/stable/Packages/docker-compose-plugin-2.35.1-1.el9.x86_64.rpm
```

```
# 확인
containerd.io-1.7.27-3.1.el9.x86_64.rpm
docker-buildx-plugin-0.23.0-1.el9.x86_64.rpm
docker-ce-28.1.1-1.el9.x86_64.rpm
docker-ce-cli-28.1.1-1.el9.x86_64.rpm
docker-ce-rootless-extras-28.1.1-1.el9.x86_64.rpm
docker-compose-plugin-2.35.1-1.el9.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. (Worker Node) 클러스터 Join

- Master Node에서 `kubeadm init` 시 생성된 클러스터 join 명령을 복사하여, Worker Node에서 실행합니다.

```
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 ⇒ worker 노드에서도 `kubectl` 사용이 가능해 집니다.

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

- Master에서 클러스터 확인

```
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
```