Kubernetes 클러스터 구축

- Centos 7 vm 3대
  - CPU : 2 Core
  - 메모리 : 10240MB
- Kubernetes v1.27.1

K8S 클러스터 구성

HOSTNAME	IP ADDRESS	ROLE
k8s-master	192.168.56.141	control-plane
k8s-node1	192.168.56.142	worker-node
k8s-node2	192.168.56.143	worker-node

1. VM 세팅

공통적인 서버 세팅을 끝낸 다음 CLONE 하는 방법을 택하였다. 그리고 쿠버네티스 설치를 하다 보면 여러가지 에러가 발생할 가능성이 높은데 한번 설정이 꼬이기 시작하면 바로 잡는 것이 쉽지 않기 때문에 CLONE까지 하고 난 다음 각 서버마다 Snapshot 생성하는 것을 추천한다.

/etc/hosts 설정

```
192.168.56.141 k8s-master
192.168.56.142 k8s-node1
192.168.56.143 k8s-node2
```

hostname 변경

```
[root@k8s-master ~]# hostnamectl set-hostname k8s-master
```

hostname을 변경한 후 ssh 재접속하면 hostname이 정상적으로 변경된다.

SELinux

쿠버네티스가 Pod 네트워크에 필요한 호스트 파일시스템에 접근할 수 있도록 기존 SELINUX=permissive를 SELINUX=permissive로 변경한다. 이후 reboot 하여 적용.

```
[root@k8s-master ~]# vi /etc/sysconfig/selinux

# This file controls the state of SELinux on the system.
```

```
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are
protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted

[root@k8s-master ~]# reboot
```

방화벽 해제

```
[root@k8s-master ~]# systemctl stop firewalld && systemctl disable firewalld
```

NetworkManager 비활성화

```
[root@k8s-master ~]# systemctl stop NetworkManager && systemctl disable
NetworkManager
```

swap 비활성화

```
[root@k8s-master ~]# swapoff -a
[root@k8s-master ~]# sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

/etc/sysctl.d/k8s.conf 설정

CNI(Container network interface) 플러그인을 사용하기 위해 아래와 같은 설정이 필요하다.

```
[root@k8s-master ~]# vi /etc/sysctl.d/k8s.conf

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

K8S YUM Repository 설정

```
[root@k8s-master ~]# 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
```

Centos Package Update

```
[root@k8s-master ~]# yum update -y
```

2. Docker & K8S 설치

Docker 설치 전 사전 세팅

```
[root@k8s-master ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
```

Docker 레포지토리 설정

```
[root@k8s-master ~]# yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
```

Docker 패키지 설치

```
[root@k8s-master ~]# yum update && yum install docker-ce
```

쿠버네티스 설치 & 확인

```
[root@k8s-master ~]# yum install -y --disableexcludes=kubernetes kubeadm kubectl
kubelet
[root@k8s-master ~]# kubelet --version
Kubernetes v1.27.1
```

쿠버네티스 Container Runtime 전환

쿠버네티스 1.24 이후 버전에서 더이상 Docker를 컨테이터 런타임으로 지원하지 않기 때문에 컨테이터 런타임을 **Docker에서 containerd로 변경**해야 한다.

/etc/containerd/config.toml 파일의 `disabled_plugins` 라인을 주석 처리한다.

```
[root@k8s-master ~]# vi /etc/containerd/config.toml

#disabled_plugins = ["cri"]
```

containerd 재시작.

```
[root@k8s-master ~]# systemctl restart containerd
```

이후 마스터노드에서 쿠버네티스 init을 실행하니 아래와 같은 에러들이 발생하였다. 차근차근 정리해보자.

1) Docker cgroup 드라이버 변경

```
[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver.
The recommended driver is "systemd".
```

Docker cgroup 드라이버가 cgroupfs로 설정되어 있고 이를 **권장 드라이버인 systemd로 변경**해주어야 한다.

```
[root@k8s-master ~]# systemctl start docker
[root@k8s-master ~]# docker info | grep -i cgroup
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
[root@k8s-master ~]# vi /etc/docker/daemon.json

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

[root@k8s-master ~]# systemctl restart docker
[root@k8s-master ~]# docker info | grep -i cgroup
 Cgroup Driver: systemd
 Cgroup Version: 1
```

2) NumCPU 에러

```
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR NumCPU]: the number of available CPUs 1 is less than the required
2
```

kubeadm으로 init 할 때 시스템 상태의 유효성을 검사하기 위해 일련의 검사를 실행한다. 해당 에러는 호스트에 최소 2개 이상의 CPU가 필요하기 때문에 발생하는 에러이다. `kubeadm init --ignore-preflight-errors=NumCPU` 명령어로 해당 에러를 무시할 수도 있지만 **VM 서버의 CPU를 2개로 늘려주어 해결**하였다.

3) CNI 선택

CNI는 컨테이너 간의 네트워킹을 제어할 수 있는 플러그인을 만들기 위한 표준이다. 쿠버네티스에서는 Pod 간의 통신을 위해서 CNI를 사용한다. 쿠버네티스는 기본적으로 `kubenet`이라는 자체적인 CNI 플러그인을 제공하지만 네트워크 기능이 매우 제한적인 단점이 있다. 그 단점을 보안하기 위해 3rd-party 플러그인을 사용하는데 대표적으로 Calico, Flannel, Weavenet 등이 존재한다.

3. VM CLONE & Snapshot

Docker와 K8S까지 설치가 완료되면 해당 서버를 Clone하여 Woker-node 세팅을 진행한다. 이후 설치 과정에서 에러가 발생할 수 있기 때문에 현 시점의 VM Snapshot을 생성하는 것을 추천한다.

hostname 변경

```
-- 각 서버에 맞게 설정(node1, node2)
[root@k8s-master ~]# hostnamectl set-hostname k8s-node1
```

hostname을 변경한 후 ssh 재접속하면 hostname이 정상적으로 변경된다.

이후 호스트명과 ip 주소를 입력해준다.

```
[root@k8s-node1 ~]# vi /etc/hosts
192.168.56.141 k8s-master
192.168.56.142 k8s-node1
192.168.56.143 k8s-node2
```

4. Master node 설정

Docker & 쿠버네티스 실행

```
[root@k8s-master ~]# systemctl daemon-reload
[root@k8s-master ~]# systemctl enable --now docker
[root@k8s-master ~]# systemctl enable --now kubelet
```

쿠버네티스 클러스터 구축(초기화)

```
[root@k8s-master ~]# kubeadm init --apiserver-advertise-address=192.168.56.141 --pod-network-cidr=10.244.0.0/16
```

- apiserver-advertise-address : Master node의 ip 주소
- pod-network-cidr : Flannel 에서 권장하는 네트워크 대역

초기화가 정상적으로 진행되면 "kubeadm join xxx.xxx.xxx.xxx" 으로 시작하는 메세지가 출력되는데 이는 Worker node에서 클러스터 구축에 필요한 명령어이기 때문에 잘 복사해준다.

환경변수 설정

```
[root@k8s-master ~]# mkdir -p $HOME/.kube
[root@k8s-master ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master ~]# chown $(id -u):$(id -g) $HOME/.kube/config
[root@k8s-master ~]# vi /etc/profile
export KUBECONFIG=/etc/kubernetes/admin.conf
[root@k8s-master ~]# source /etc/profile
```

5. Worker node 설정

Kubernetes Worker node가 되기 위해서는 kubeadm join 명령어를 실행해서 Master node에 등록한다. (node1, node2에서 각각 실행)

Docker & 쿠버네티스 실행

```
[root@k8s-node1 ~]# systemctl daemon-reload
[root@k8s-node1 ~]# systemctl enable --now docker
[root@k8s-node1 ~]# systemctl enable --now kubelet
```

Node 연결

Master node에서 kubeadm init 명령어로 나왔던 명령어를 실행한다.

```
[root@k8s-node1 ~]# kubeadm join xxx...
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
```

```
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.
```

본인은 해당 과정에 있어 여러가지 문제가 발생하였고 이를 대비하여 기존에 생성했던 Snapshot을 활용하여 복구하는 방식으로 진행하였다. Master node에서 초기화를 새롭게 진행하고 Worker node에서 위의 작업을 실행하는데 아래와 같은 에러가 발생하였다.

```
certificate has expired or is not yet valid: current time 2023-05-
09T14:24:30+09:00 is before 2023-05-09T06:25:11Z"
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR ImagePull]: failed to pull image registry.k8s.io/kube-
apiserver:v1.27.1: output: E0509 14:24:30.667354    6185 remote_image.go:171]
"PullImage from image service failed" err="rpc error: code = Unknown desc =
failed to pull and unpack image \"registry.k8s.io/kube-apiserver:v1.27.1\":
failed to resolve reference \"registry.k8s.io/kube-apiserver:v1.27.1\": failed to
do request: Head \"https://registry.k8s.io/v2/kube-apiserver/manifests/v1.27.1\":
x509: certificate has expired or is not yet valid: current time 2023-05-
09T14:24:30+09:00 is before 2023-05-09T06:25:11Z" image="registry.k8s.io/kube-
apiserver:v1.27.1"
time="2023-05-09T14:24:30+09:00" level=fatal msg="pulling image: rpc error: code
= Unknown desc = failed to pull and unpack image \"registry.k8s.io/kube-
apiserver:v1.27.1\": failed to resolve reference \"registry.k8s.io/kube-
apiserver:v1.27.1\": failed to do request: Head
\"https://registry.k8s.io/v2/kube-apiserver/manifests/v1.27.1\": x509:
certificate has expired or is not yet valid: current time 2023-05-
09T14:24:30+09:00 is before 2023-05-09T06:25:11Z"
```

로그를 살펴보니 **서버의 시간과 현재 시간이 맞지 않아서 생기는 문제**인 것으로 보였다. 서버의 시간을 `date` 명령어로 확인해보니 역시나 현재 시간과 맞지 않았다. VM은 Snapshot을 생성한 그 시점의 모든 설정을 저장하기 때문에 시간도 **Snapshot을 생성한 시점의 시간으로 정지되어 있는 것으로 보여진다**. 따라서 서버 시간을 동기화 해줌으로써(ntp) 에러를 해결하였고 `kubeadm join` 명령어도 정상적으로 실행되었다.

CNI 플러그인 설치(Flannel)

```
[root@k8s-master ~]# kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
flannel.yml
```

Worker node 2대에서 `kubeadm join` 명령어를 통해 정상적으로 Master node에 등록이 되었다면 Master node에서 `kubectl get nodes` 명령어로 클러스터 상태를 확인한다.

```
[root@k8s-master ~]# kubectl get nodes
NAME         STATUS     ROLES           AGE     VERSION
k8s-master   Ready      control-plane   6d20h   v1.27.1
k8s-node1    NotReady   <none>          6d20h   v1.27.1
k8s-node2    NotReady   <none>          6d20h   v1.27.1
```

현재 STATUS 를 보면 NotReady로 표시되어 있는 것을 볼 수 있다. 이는 클러스터 구축은 정상적으로 완료되었지만 각 pod간의 통신 문제가 발생한 것으로 보여진다.

```
[root@k8s-master ~]# kubectl get pods --all-namespaces
NAMESPACE      NAME                                 READY   STATUS
RESTARTS        AGE
kube-flannel   kube-flannel-ds-pphsj                0/1     Init:0/2
 0              6d20h
kube-flannel   kube-flannel-ds-wrmph                1/1     Running
1 (2d23h ago)   6d20h
kube-flannel   kube-flannel-ds-x6dx2                0/1     Init:0/2
 0              6d20h
kube-system    coredns-5d78c9869d-68wwv             1/1     Running
1 (2d23h ago)   6d20h
kube-system    coredns-5d78c9869d-72f7x             1/1     Running
1 (2d23h ago)   6d20h
kube-system    etcd-k8s-master                      1/1     Running
5 (2d23h ago)   6d20h
kube-system    kube-apiserver-k8s-master            1/1     Running
5 (2d23h ago)   6d20h
kube-system    kube-controller-manager-k8s-master   1/1     Running
7 (2d23h ago)   6d20h
kube-system    kube-proxy-fjw2n                     0/1     ContainerCreating
0               6d20h
kube-system    kube-proxy-rgxj7                     1/1     Running
1 (2d23h ago)   6d20h
kube-system    kube-proxy-tffsz                     0/1     ContainerCreating
0               6d20h
kube-system    kube-scheduler-k8s-master            1/1     Running
7 (2d23h ago)   6d20h
```

모든 네임스페이드 대상 pods 리스트를 보면 CNI 네트워크 설정이 정상적으로 되지 않은 것으로 보여진다.

각 노드의 상태를 확인해보자. 그 중 k8s-node1의 내용만 가져왔다.

```
[root@k8s-master ~]# kubectl describe node
.
.
Name:               k8s-node1
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=k8s-node1
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket:
unix:///var/run/containerd/containerd.sock
```

```
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Thu, 11 May 2023 17:37:04 +0900
Taints:             node.kubernetes.io/not-ready:NoExecute
                    node.kubernetes.io/not-ready:NoSchedule
Unschedulable:      false
Lease:
  HolderIdentity:  k8s-node1
  AcquireTime:     <unset>
  RenewTime:       Thu, 18 May 2023 14:13:29 +0900
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime
            Reason                       Message
  ----             ------  -----------------                 ------------------
              ------                       -------
  MemoryPressure   False   Thu, 18 May 2023 14:08:42 +0900   Thu, 11 May 2023
17:37:04 +0900   KubeletHasSufficientMemory   kubelet has sufficient memory
available
  DiskPressure     False   Thu, 18 May 2023 14:08:42 +0900   Thu, 11 May 2023
17:37:04 +0900   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Thu, 18 May 2023 14:08:42 +0900   Thu, 11 May 2023
17:37:04 +0900   KubeletHasSufficientPID      kubelet has sufficient PID
available
  Ready            False   Thu, 18 May 2023 14:08:42 +0900   Thu, 11 May 2023
17:37:04 +0900   KubeletNotReady              container runtime network not
ready: NetworkReady=false reason:NetworkPluginNotReady message:Network plugin
returns error: cni plugin not initialized
```

node1의 상태를 살펴보면 `container runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady message:Network plugin returns error: cni plugin not initialized` 로 역시 **CNI 네트워크 문제**인 것으로 보인다. 아직 해당 에러를 자세히 살펴보지 못했지만 추가적으로 확인하여 해결할 예정이다.

6. CNI 네트워크 연결 에러

CNI 플러그인 종류를 여러가지 변경해서 클러스터를 초기화 해보았지만 여전히 Worker node에서 CNI 플러그인 연결이 안되는 문제가 발생하였다. 자세히 살펴보니 CLONE으로 생성한 vm에서 외부 네트워크 연결이 안되고 있었다. 당연히 CLONE을 생성 할 때 모든 네트워크 설정을 새롭게 생성하였고 VM 3대 모두 `/etc/sysconfig/network-scripts/ifcfg-enp0s8` 파일에서 ip 주소를 개별적으로 생성해주었는데 외부 ping이 나가지 않는 문제가 발생하였다. (기존 외부 네트워크와 연결이 되었던 Master node에서도 CLONE을 생성해주면 외부 ping이 나가지 않는 문제)

Worker node에서 join을 하고 **CNI 플러그인을 Master node에서 배포할 때 Worker node가 외부와의 연결이 되지 않기 때문에 CNI 네트워크 연결에 필요한 설정들을 다운로드 하는 데 문제가 생긴 것이라는 판단**이 들었고 이미 쿠버네티스 클러스터 구축하는데 많은 시간을 소요했고 더이상 지체할 수 없었기 때문에 VM 3대를 CLONE하는 방식이 아닌 VM을 각각 새롭게 올리는 방식을 택하였다.

VM 3대 모두 위에서 했던 설정 그대로 진행하였으며 기존에 클러스터를 구축하기 위해 여러 번 시도 했었기 때문에 빠른 시간 안에 쿠버네티스 구축에 성공할 수 있었다. 그럼 Master node에서 init 하는 과정부터 다시 살펴보자.

7. Master node 설정

Docker & 쿠버네티스 실행

```
[root@k8s-master ~]# systemctl daemon-reload
[root@k8s-master ~]# systemctl enable --now docker
[root@k8s-master ~]# systemctl enable --now kubelet
```

쿠버네티스 클러스터 구축(초기화)

```
[root@k8s-master ~]# kubeadm init --apiserver-advertise-address=192.168.56.141 --pod-network-cidr=10.244.0.0/16
```

- apiserver-advertise-address : Master node의 ip 주소
- pod-network-cidr : Flannel 에서 권장하는 네트워크 대역

초기화가 정상적으로 진행되면 "kubeadm join xxx.xxx.xxx.xxx" 으로 시작하는 메세지가 출력되는데 이는 Worker node에서 클러스터 구축에 필요한 명령어이기 때문에 잘 복사해준다.

환경변수 설정

```
[root@k8s-master ~]# mkdir -p $HOME/.kube
[root@k8s-master ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master ~]# chown $(id -u):$(id -g) $HOME/.kube/config
[root@k8s-master ~]# vi /etc/profile
export KUBECONFIG=/etc/kubernetes/admin.conf
[root@k8s-master ~]# source /etc/profile
```

8. Worker node 설정

Kubernetes Worker node가 되기 위해서는 kubeadm join 명령어를 실행해서 Master node에 등록한다. (node1, node2에서 각각 실행)

Docker & 쿠버네티스 실행

```
[root@k8s-node1 ~]# systemctl daemon-reload
[root@k8s-node1 ~]# systemctl enable --now docker
[root@k8s-node1 ~]# systemctl enable --now kubelet
```

Node 연결

Master node에서 kubeadm init 명령어로 나왔던 명령어를 실행한다.

```
[root@k8s-node1 ~]# kubeadm join xxx...
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system
get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

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.
```

CNI 플러그인 설치(Flannel)

```
[root@k8s-master ~]# kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
flannel.yml
```

Worker node 2대에서 `kubeadm join` 명령어를 통해 정상적으로 Master node에 등록이 되었다면 Master node에서 `kubectl get nodes` 명령어로 클러스터 상태를 확인한다.

```
[root@k8s-master ~]# kubectl get nodes
NAME         STATUS   ROLES           AGE     VERSION
k8s-master   Ready    control-plane   4m32s   v1.27.2
k8s-node1    Ready    <none>          3m16s   v1.27.2
k8s-node2    Ready    <none>          3m13s   v1.27.2
```

역시 네트워크 문제였다. vm 3대 모두 외부와의 연결이 가능한 지금은 모든 노드의 상태가 `Ready`로 변경된 것을 볼 수 있다.

```
[root@k8s-master ~]# kubectl get pods --all-namespaces
NAMESPACE      NAME                                READY   STATUS    RESTARTS
AGE
kube-flannel   kube-flannel-ds-2wphh               1/1     Running   0
75m
kube-flannel   kube-flannel-ds-pnw8f               1/1     Running   0
75m
```

```
kube-flannel   kube-flannel-ds-r9bx8                1/1     Running   0
75m
kube-system    coredns-5d78c9869d-ppj4v             1/1     Running   0
77m
kube-system    coredns-5d78c9869d-x6gqw             1/1     Running   0
77m
kube-system    etcd-k8s-master                      1/1     Running   0
77m
kube-system    kube-apiserver-k8s-master            1/1     Running   0
77m
kube-system    kube-controller-manager-k8s-master   1/1     Running   1
77m
kube-system    kube-proxy-49t48                     1/1     Running   0
76m
kube-system    kube-proxy-fzdmt                     1/1     Running   0
77m
kube-system    kube-proxy-vm2qd                     1/1     Running   0
76m
kube-system    kube-scheduler-k8s-master            1/1     Running   1
77m
```

위처럼 coredns와 kube-flannel이 정상적으로 `Running` 상태가 되어야 한다.

9. 쿠버네티스 클러스터 재구성

기존 `kubeadm init` 명령어를 통해 클러스터를 구축했다면 다음과 같이 쿠버네티스 클러스터를 초기화 할 수 있다. 이는 클러스터를 다른 CNI 네트워크로 묶는다던지 클러스터 구축 할 때 에러가 발생하여 노드 초기화가 필요할 때 사용한다.

Master node 초기화

```
[root@k8s-master ~]# kubeadm reset
>>>
[reset] Reading configuration from the cluster...
[reset] FYI: You can look at this config file with 'kubectl -n kube-system get
cm kubeadm-config -o yaml'
[reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join'
will be reverted.
[reset] Are you sure you want to proceed? [y/N]: y

[preflight] Running pre-flight checks
The 'update-cluster-status' phase is deprecated and will be removed in a future
release. Currently it performs no operation
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of config directories: [/etc/kubernetes/manifests
/etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/kubelet.conf
/etc/kubernetes/bootstrap-kubelet.conf /etc/kubernetes/controller-manager.conf
/etc/kubernetes/scheduler.conf]
```

```
[reset] Deleting contents of stateful directories: [/var/lib/etcd
/var/lib/kubelet /var/lib/dockershim /var/run/kubernetes /var/lib/cni]

The reset process does not clean CNI configuration. To do so, you must remove
/etc/cni/net.d

The reset process does not reset or clean up iptables rules or IPVS tables.
If you wish to reset iptables, you must do so manually by using the "iptables"
command.

If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)
to reset your system's IPVS tables.

The reset process does not clean your kubeconfig files and you must remove them
manually.
Please, check the contents of the $HOME/.kube/config file.
```

위의 내용을 참조하여 CNI 설정 및 쿠버네티스 관련 설정을 삭제한다.

```
[root@k8s-master ~]# rm -r /etc/cni/net.d/*
[root@k8s-master ~]# rm -r ~/.kube/config
```

`kubelet` 서비스 재시작

```
[root@k8s-master ~]# systemctl restart kubelet
```

Worker node 초기화

Master node에서 했던 작업과 추후 다른 CNI를 이용하여 클러스터 구축하는 것을 대비해 CNI 설정파일도 삭제한다.

```
[root@k8s-node1 ~]# kubeadm reset

>>>
[reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join'
will be reverted.
[reset] Are you sure you want to proceed? [y/N]:y
```

CNI, 클러스터 관련 파일 삭제

```
[root@k8s-node1 ~]# rm -r /etc/cni/net.d/*
[root@k8s-node1 ~]# rm -r /etc/kubernetes/*
```