

Install Kubernetes on CentOS

1. Prerequisites

OS : CentOS v7.6

Arch : x86_64

2. k8s클러스터는 1마스터 2노드로 구성 (혹은 1Master 1Node)

Master : 4cpu, ram16G

Node : 2cpu, ram8G

==>> Master : 2cpu, ram4G

Node : 2cpu, ram4G

설치가 완료되면 root 계정으로 로그인을 함

3. Steps

(M,N) : Master와 Node 전부 적용

3.1 Set Hostname on Nodes (M,N)

각 Master, Node들마다 자신의 hostname을 결정

\$ hostnamectl set-hostname {호스트이름 }

Master

#hostnamectl set-hostname k8s-master

Node

#hostnamectl set-hostname k8s-node1

#hostnamectl set-hostname k8s-node2

#hostname

결정한 이름을 각 노드의 hosts파일에 ip와 작성

vim /etc/hosts

kubernetes cluster (주석)

192.168.0.23 kube-master

192.168.0.21 kube-node1

192.168.0.22 kube-node2

scp를 이용해서 kube-node1로 복사/ 직접 추가

#ssh-keygen -t rsa (Master와 Node에서 동일하게 명령 실행)

#cd /root/.ssh

#cp id_rsa.pub authorized_keys

#cd

#ssh localhost

#exit

#scp /root/.ssh/authorized_keys 192.168.0.21:/root/

비밀번호 입력

===>> Node에서

```
#cat authorized_keys >> /root/.ssh/authorized_keys
```

```
#scp /root/.ssh/authorized_keys 192.168.0.23:/root
```

===>> Master에서

```
#cat authorized_keys >> /root/.ssh/authorized_keys
```

3.2 Install Docker (M,N)

k8s 클러스터를 구성하기 위해서는 도커가 필요. 전체 노드에 도커를 설치.

도커가 이미 깔려있는 상태라면 다음단계로 진행

Docker 설치

가능하시면 root로 로그인하시면 됩니다.

```
# yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
# yum-config-manager --add-repo
```

```
https://download.docker.com/linux/centos/docker-ce.repo
```

```
# yum install docker-ce -y
```

```
# systemctl start docker && systemctl enable docker
```

(재부팅시 재시작하지 않고자 한다면 systemctl disable docker)

도커 최신 버전 문제로 인해 kubeadm join이 구동이 되지 않으면 (node 머신에서 진행시)

```
#yum install docker-ce-19.03.12 docker-ce-cli-19.03.12 containerd.io 로 설치
```

3.3 Configure Firewall (M,N)

방화벽을 끄지 않고 마스터와 노드의 포트를 설정 가능하지만 방화벽 OFF

firewalld 비활성화

```
$ systemctl stop firewalld
```

```
$ systemctl disable firewalld
```

```
$ systemctl status firewalld
```

```
[root@kubem ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

방화벽을 끄지 않고 직접 설정을 할 경우

```
#firewall-cmd --permanent --add-port=6443/tcp
```

```
#firewall-cmd --permanent --add-port=10250/tcp
```

포트를 추가해서 지정을 하시고 아래 명령을 실행

```
#firewall-cmd --reload
```

```
#iptables -L
```

```
#iptables -F
```

```
#iptables-save
```

3.4 Update iptables Settings (M,N)

iptables 설정

```
$ cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables=1
net.bridge.bridge-nf-call-iptables=1
EOF
```

cat /proc/sys/net/bridge/bridge-nf-call-iptables 값이 1로 변경되어있으면 되는데, 변경을 하려면 아래 명령어를 실행

혹은 /proc/sys/net/ipv4/ip_forward의 값을 1로 하셔도 됩니다.

```
# vi /etc/sysctl.d/ip.conf
net.ipv4.ip_forward=1
```

적용

```
$ sysctl --system
```

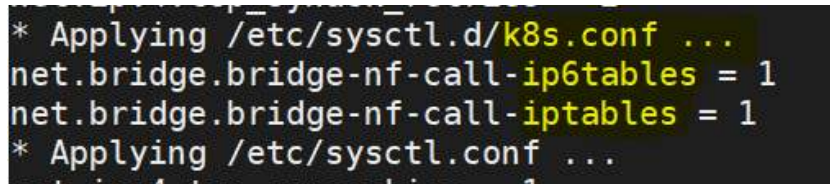
net.bridge.bridge-nf-call-iptables=1 의 의미 :

centOS와 같은 리눅스 배포판은 net.bridge.bridge-nf-call-iptables의 default값이 0이다.

이는 bridge 네트워크를 통해 송수신 되는 패킷이 **iptables** 설정을 우회한다는 의미.

컨테이너의 네트워크 패킷이 호스트머신의 **iptables** 설정에 따라 제어되도록 하는 것이

바람직하며, 이를 위해서는 값을 1로 설정



```
* Applying /etc/sysctl.d/k8s.conf ...
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
* Applying /etc/sysctl.conf ...
```

3.5 Disable SELinux (M,N)

SELinux는 Linux의 보안을 강화해주는 **보안 커널**이며, application의 취약점으로 인한 해킹을 방지해주는 핵심 구성 요소.

총 세가지 모드

enforcing (Default) : SELinux의 rule에 어긋나는 operation은 거부

permissive : rule에 어긋나는 동작이 있을 경우 audit log를 남기고 **operation**은 허용

disabled : 제한없음 (#vi /etc/sysconfig/selinux의 SELINUX=disabled 변경)

production 서버일 경우 SELinux를 끄기보다는 서비스가 SELinux하에서 잘 동작하도록 하는것이 바람직하며, 개발서버일 경우 Permissive모드로 진행하다가 추후에 enforce모드로 전환 필요. /etc/sysconfig/selinux 파일을 직접 수정

```
$ getenfoce
```

```
$ setenforce 0 (메모리에서만 변경되며 재부팅을 하면 원래대로 바뀐다)
```

```
$ sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

```
$ getenfoce
```

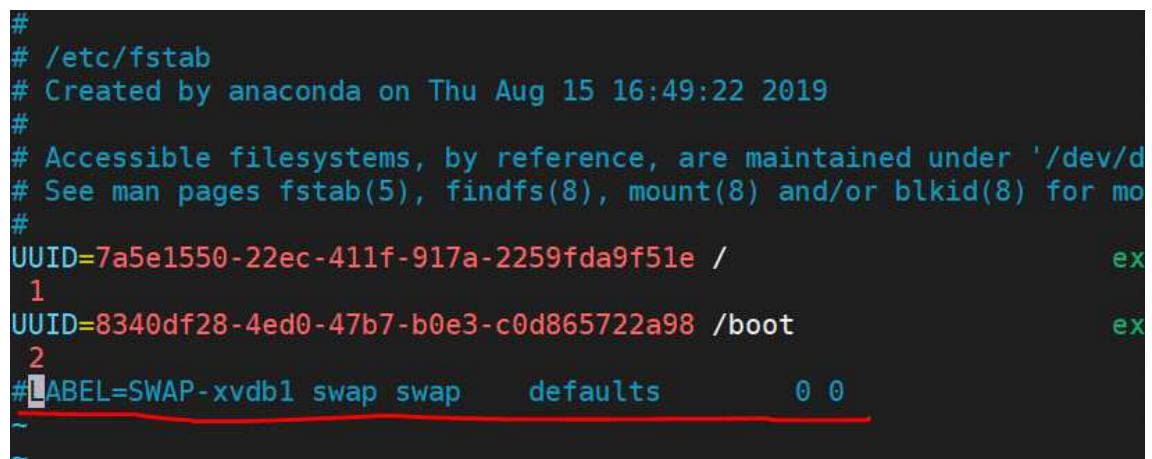
혹은

```
#vi /etc/sysconfig/selinux
SELINUX=permissive
```

3.6 Disable SWAP (M,N)

다음으로, kubelet이 제대로 동작하기 위해 SWAP을 disable

```
$ swapoff -a
$ vim /etc/fstab
# 아래 부분을 찾아서 주석처리 해줍니다.
# LABEL=SWAP-xvdb1 swap swap defaults 0 0
$ reboot
$ free -h
```



```
#
# /etc/fstab
# Created by anaconda on Thu Aug 15 16:49:22 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/d
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for mo
#
UUID=7a5e1550-22ec-411f-917a-2259fda9f51e /                                ex
1
UUID=8340df28-4ed0-47b7-b0e3-c0d865722a98 /boot                          ex
2
# LABEL=SWAP-xvdb1 swap swap defaults 0 0
~
~
```

3.7 Kubernetes Repo 추가 (M,N)

```
# ping google.com
# cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

```
#vi /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-$basearch
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
exclude=kubelet kubeadm kubectl
```

3.8 kubelet, kubeadm, kubectl 설치 (M,N)

```
# yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
# yum install -y kubelet-1.20.2-0 kubeadm-1.20.2-0 kubectl-1.20.2-0
--disableexcludes=kubernetes
# systemctl enable kubelet && systemctl start kubelet
```

위 명령이 실행되고 정상 동작하지 않는다면 잠시 아래 내용을 확인하고 다시 실행

** kubelet과 docker의 Cgroup Driver 일치

kubelet과 docker의 cgroupdriver가 다를 경우 kubelet이 실행되지 않은 경우가 있다.

kubelet은 default로 cgroupdriver를 systemd를 사용하는 반면 docker는 cgroupfs를 사용한다.

에러 발생시

```
#vi /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
```

혹은

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

```
# systemctl restart docker
```

```
# systemctl restart kubelet
```

혹은

#vi /lib/systemd/system/docker.service

--exec-opt native.cgroupdriver=systemd 추가

```
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --exec-opt native.cgroupdriver=systemd
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
```

- kubelet cgroup driver 설정 파일 : /var/lib/kubelet/config.yaml

/*

kubeadm init

혹은

kubeadm join

*/

=====>>> vm image를 복제

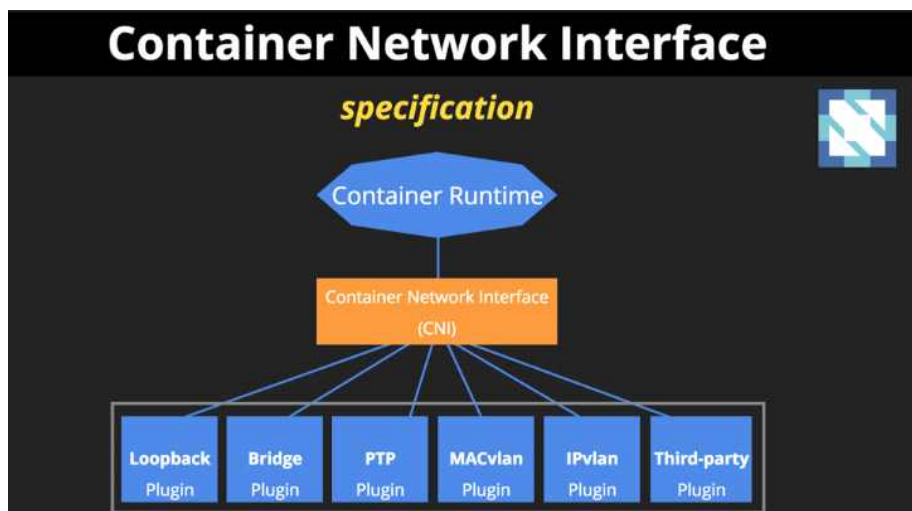
3.9 Master 초기화 (M)

다음 명령어를 사용하여 마스터를 초기화

--pod-network-cidr 옵션은 Container Network Interface로 어떤 걸 사용할지 결정

Container Network Interface(CNI)?

컨테이너 런타임과 컨테이너 네트워크 사이의 표준 API



CNI로 calico를 사용하기로 하고, 다른 CNI들을 사용하려면 다음 링크를 참조하여 작성

Kubernetes : [Installing a pod network add-on](#)

CNI는 calico사용 (192.168.0.0/16)

CNI는 flannel 사용시 (10.244.0.0/16)

calico CNI 사용

\$ kubeadm init --pod-network-cidr=192.168.0.0/16

--apiserver-advertise-address=192.168.0.23 (호스트 서버 IP) ==>> ifconfig 혹은 ip addr
혹은 flannel CNI 사용

```
$ kubeadm init --pod-network-cidr=10.244.0.0/16
```

--apiserver-advertise-address=192.168.0.23 (호스트 서버 IP)

위 명령어를 실행해 마스터를 초기화시키고 나면 성공적으로 초기화시켰다는 로그와 함께
노드가 마스터에 join할 수 있게 하는 커맨드

다음 커맨드와 유사한 메시지가 뜰테니 어딘가에 복사 필요 ==>> worker node에서 실행

```
kubeadm join 192.168.0.23:6443 --token b0em0n.b8v9b760cmz8nmtY W
```

```
--discovery-token-ca-cert-hash
```

```
sha256:dd5ae0bb6c4f6e7aff5f8ac34d63383d50e1f374a36c9530cec756c406bec4e7
```

환경변수 설정

그 다음 root계정으로 kubectl명령어를 사용하기 위해 환경변수 설정

```
# export KUBECONFIG=/etc/kubernetes/admin.conf
```

```
# vi ~/.bash_profile 파일에 위 명령어 작성
```

```
$ source ~/.bash_profile
```

```
mkdir -p /root/.kube
```

```
cp /etc/kubernetes/admin.conf /root/.kube/config
```

config 파일이나 admin.conf 파일의 내용을 보시면 IP와 포트 부분이 있습니다. 그 부분을
반드시 확인하셔야 합니다.

CNI 설치

마스터 노드를 초기화할때 calico로 설치하는 것으로 설정을 했으니, 다음 명령어를 통해
설치.(50개 노드 이하에서 사용)

```
$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

calico 공식홈페이지 : [Install Calico networking and network policy for on-premises deployments](https://docs.projectcalico.org/manifests/calico.yaml)

혹은

```
$ kubectl apply -f
```

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

```
$ kubectl get pods --all-namespaces
```

```
[root@kube_m ~]# kubectl get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	calico-kube-controllers-55754f75c-zgn69	1/1	Running	0	78s
kube-system	calico-node-46pmc	1/1	Running	0	78s
kube-system	coredns-5644d7b6d9-4z46r	1/1	Running	0	18m
kube-system	coredns-5644d7b6d9-p2gmm	1/1	Running	0	18m
kube-system	etcd-kube-m	1/1	Running	0	17m
kube-system	kube-apiserver-kube-m	1/1	Running	0	17m
kube-system	kube-controller-manager-kube-m	1/1	Running	0	17m
kube-system	kube-proxy-6fdqk	1/1	Running	0	18m
kube-system	kube-scheduler-kube-m	1/1	Running	0	17m

3.10 Node join (N)

위에서 Master를 초기화시킬 때 복사해둔 kubeadm join~을 사용할 차례
복사한 명령어를 각 노드로 가서 입력

```
#kubeadm join 192.168.0.23:6443 --token b0em0n.b8v9b760cmz8nmtY W  
--discovery-token-ca-cert-hash  
sha256:dd5ae0bb6c4f6e7aff5f8ac34d63383d50e1f374a36c9530cec756c406bec4e7
```

.....

This node has joined the cluster:

- * Certificate signing request was sent to apiservert 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.
성공적으로 join했다는 메시지가 뜨면 마스터로 가서 확인

Master 에서

```
$ kubectl get nodes
```

```
[root@kube_m ~]# kubectl get nodes  
NAME          STATUS    ROLES    AGE   VERSION  
kube-m        Ready     master   28m   v1.16.2  
kube-n01      Ready     <none>   34s   v1.16.2  
kube-n02      Ready     <none>   28s   v1.16.2
```

마스터 하나에 노드 두개로 클러스터가 구성된 것을 확인

혹시 Master의 토큰 값이 이상하거나 분실되었을 경우 Master 아래 명령을 실행하시면 토큰 값을 재할당 합니다.

Master에서 실행

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

명령 실행후 나온 명령을 Node에서 실행하시면 join 됩니다.

에러 발생시 (preflight 등 에러)

마스터 노드: #kubeadm reset

```
#kubeadm reset cleanup-node 명령(파일까지 지워짐)
```

```
#systemctl restart kubelet
```

워커노드: #kubeadm reset

```
#kubeadm join
```

port 해제

```
#fuser -k -n tcp 8000
```



```
#fuser -k -n tcp 10250
```

Unable to connect to the server:x509:certificate signed by unknown authority 에러

```
#export KUBECONFIG=/etc/kubernetes/admin.conf
```

4. Cluster test (nginx)

클러스터가 구성이 되었으니 간단한 예제를 하나 실행

nginx 앱을 클러스터에 올려서 외부에서 접속해보는 테스트를 진행

4.1 Docker image

먼저 쿠버네티스 클러스터에 올릴 nginx 이미지를 docker hub에서 pull

```
$ docker image pull nginx
```

4.2 pod 실행

받은 이미지를 pod로 실행

nginx-test는 임의의 이름

```
$ kubectl run nginx-test --image=nginx --port=8080
```

--generator =run-pod/v1 옵션은 실행이 안될수 있음.

```
$ kubectl run nginx-test --image=nginx --port=8080 --generator=run-pod/v1
```

pod/nginx-test created

4.3 service 실행

pod만 띄우면 끝나는 것이 아니라 pod가 포함된 컨테이너에 연결할 수 있도록 서비스를 생성

```
$ kubectl expose pod nginx-test
```

service/nginx-test exposed

get service를 통해 생성된 서비스들을 확인

```
$ kubectl get service
```

```
[root@kube_m ~]# kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	80m
nginx-test	ClusterIP	10.108.74.127	<none>	8080/TCP	15s

현재는 쿠버네티스와 방금전에 올린 nginx-test 서비스만 실행

서비스 nginx-test의 TYPE은 ClusterIP인데 이는 내부 클러스터에만 ip가 열려있는 상태

즉 외부에서 접근하지 못한다

외부에서 접근할 수 있게 하려면 여러가지 방법이 있지만, 이번엔 Type만 바꾸는 방법

4.4 Service Type 변경

kubectl edit를 통해 현재 떠있는 서비스를 수정할 수 있습니다.

```
$ kubectl edit services nginx-test
```

```
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2019-11-05T08:32:52Z"
  labels:
    run: nginx-test
  name: nginx-test
  namespace: default
  resourceVersion: "7278"
  selfLink: /api/v1/namespaces/default/services/nginx-test
  uid: 0872c517-d2c8-44c4-9e2c-4cfd40111257
spec:
  clusterIP: 10.108.74.127
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    run: nginx-test
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
```

빨강게 밑줄쳐져있는 부분을 ClusterIP에서 NodePort로 변경
 변경 후, get service하면

```
[root@kube_m ~]# kubectl get service
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1     <none>         443/TCP          87m
nginx-test   NodePort    10.108.74.127 <none>         8080:30657/TCP   7m52s
```

외부포트가 지정된 것을 확인

30657로 접근하면 nginx-test서비스가 바라보고있는 nginx앱의 8080으로 연결

4.5 결과 확인

IP:port로 접속하면 아래와 같은 화면.

이때 ip는 마스터의 ip이든 노드의 ip든 상관없다.

Master IP 혹은 Node IP:30657

192.168.0.23:30657

192.168.0.21:30657

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

5. Kubernetes 종료

쿠버네티스 클러스터를 삭제하는 방법

Master에서 :

```
$ kubectl drain {노드이름} --delete-local-data --force --ignore-daemonsets
```

```
$ kubectl delete node {노드이름}
```

```
$ kubeadm reset
```

Node에서 :

```
$ kubeadm reset
```

cluster 삭제하고나서 재시작이 안되는 경우:

```
$ iptables -F
```

```
$ iptables -t nat -F
```

에러처리

1. core dns 의 STATUS가 ContainerCreating

```
#kubectl edit cm coredns -n kube-system
```

loop 부분 주석처리

2. 설치 과정중

```
https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64/repodata/repomd.xml: [Errno 14] curl#60 - "Peer's certificate issuer has been marked as not trusted by the user"
```

sslverify=0을 /etc/yum.repos.d/kubernetes.repo 파일에 추가

3. kubeadm reset 실행 후

```
$ kubeadm init --pod-network-cidr=10.244.0.0/16
```

```
--apiserver-advertise-address=192.168.0.23 (호스트 서버 IP)
```

4. 명령 오류

```
#docker run -d --restart=always -e DOMAIN=cluster --name nginx-app -p 80:80 nginx
#kubectl expose deployment nginx-app --port=80 --name=nginx-http
```

5. 파일 오류

```
config net found /etc/kubernetes/admin.conf
```

```
#cp /etc/kubernetes/kubelet.conf /etc/kubernetes/admin.conf
```

혹은

```
#kubectl get nodes --kubeconfig /etc/kubernetes/kubelet.conf
```

```
#kubectl get pods --all-namespaces
```

```
#kubectl delete pod nginx-test2
```

```
#kubectl delete pod,service baz foo
```

```
#kubectl delete pods,services -l name=myLabel
```