

2022.12.07 CKA(Certified Kubernetes Administrator)

2022.06.25 (PSI Secure Browser)

;

URL :

<https://training.linuxfoundation.org/bridge-migration-2021/>

— 30



CKA Schedule 30
TAKE EXAM .

URL
<https://trainingportal.linuxfoundation.org/learn/course/certified-kubernetes-administrator-cka/exam/exam>

TAKE EXAM PSI
Cotana, (zoon, bluejean) , VPN 가

CHAT

-
-
- ()
(, HDMI)
CHAT)
- , (/)
-

가



14

17 가 2 , 2-3가

2~4
6~12

Flag

URL :

<https://docs.linuxfoundation.org/tc-docs/certification/tips-cka-and-ckad#adjusting-font-and-windows-in-the-examui>

&

가 Firefox
Kubernetes.io Document

YAML

- Ctrl+Shift+C , V 가
Copy&Paste

&

PSI 가 Firefox

CHAT 가
가

```

100      66 (2/3)      Pass      .
24
          24      가      .
Fail      Reschedule      .
          Flag
          .
          가      .
/

```

Kubernetes 1.24 + cri-docker Installation (kubeadm)

Kubernetes

18.04

Controll Node (node01) , Worker Node (node02 ,node03) 3EA

root 가 user(worker) sudo

(Pre-Option)

```

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 가
sudo swapoff /swap.img
sudo sed -i -e '/swap.img/d' /etc/fstab
```

() Container Runtime Install

k8s 1.24 (2022/05)

```
k8s dockershim
cri-docker 가 k8s
```

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/latest|grep tag_name | cut -d '"' -f 4|sed 's/v//g')
echo $VER
wget https://github.com/Mirantis/cri-dockerd/releases/download/v${VER}/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
sudo apt-get update
```

```
# k8s
sudo apt-get install -y kubelet kubeadm kubectl
```

```
#
kubectl version --short
```

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

```
#
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
IP.
```

```
#          join Command
kubeadm   join      211.115.207.136:6443      --token
r8gfco.6s7f60dns4vgwcc0 \
          --discovery-token-ca-cert-hash
sha256:7b0f82be076748e67f8615eab0b86a61317bac397f94b2921810231
ab14afdcc
```

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

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

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

```
# Worker Node      kubectl          admin.conf
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          가          .
init
```

```
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:b10158bcea37aae0e92ed6b68b4dd1e8213623cc7d406e77eef55fe
6196fe346 --cri-socket /var/run/cri-dockerd.sock
```

```
# Join
sudo kubeadm join [Controller IP]:6443 --token
r8gfco.6s7f60dns4vgwcc0 \
          --discovery-token-ca-cert-hash
```

```
sha256:7b0f82be076748e67f8615eab0b86a61317bac397f94b2921810231
ab14afdcc --cri-socket /var/run/cri-dockerd.sock
```

```
# Check
systemctl status kubelet
```

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

```
가 node          kubectl get nodes -o wide
```

Pod Network Install - Controller Node (Node01)

```
#          CNI Plugin          CoreDNS 가 Pending
```

```
kube-system-proxy
```

```
# CNI Mode
```

```
# Check - kube-proxy          . kube-proxy Node
```

```
3
```

```
kubectl get pod -A
```

```
kubectl logs -f pod/kube-proxy-[name] -n kube-system
```

```
# Log          - Linux          iptables Mode
```

```
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
```

```
,          CKA
```

```
Callico
```

```
Plugin
```

```
Calico Network IPIP , VXLAN
```

```
IPVS
```

```
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 Taint (
) calico-controller-pod가 Pending

## 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 Status
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)

```
# 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

```
# 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
```

[Container Runtime] CRI-O 가

CRI-O 가

CRI-O ?

```
# CRI-O OCI (CRI) .
가 Docker Kubernetes 1.24
```

Docker CRI() shim
(Docker Engine) CRI-0 가 .

(VM)

Controller Server : 1EA

Worker Server : 1EA

OS

Ubuntu 20.04 Server Minimal

<https://github.com/cri-o/cri-o/blob/main/install.md#supported-versions> :

<https://tech.hostway.co.kr/2022/02/06/418/>

Traffic Setup

```
# .conf
```

```
# Controller / Worker
```

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

```
sudo modprobe overlay  
sudo modprobe br_netfilter
```

```
# sysctl , .
```

```
# Controller / Worker
```

```
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf  
net.bridge.bridge-nf-call-iptables = 1  
net.ipv4.ip_forward = 1  
net.bridge.bridge-nf-call-ip6tables = 1  
EOF
```

```
sudo sysctl --system
```

cgroup driver

```
# CRI-O                                systemd cgroup                                .

cgroupfs cgroup                          , /etc/crio/crio.conf
      /etc/crio/crio.conf.d/02-cgroup-manager.conf      -
      (drop-in)                                       .

[crio.runtime]
common_cgroup = "pod"
cgroup_manager = "cgroupfs"
```

CRI-O

```
# Controller / Worker
```

```
sudo -i
export OS=xUbuntu_20.04 # OS
export VERSION=1.19     # cri-o

echo                                "deb
https://download.opensuse.org/repositories/devel:kubic:libco
ntainers:/stable/$OS/
/etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list
echo                                "deb
http://download.opensuse.org/repositories/devel:kubic:libcon
tainers:/stable:/cri-o:/$VERSION/$OS/
/etc/apt/sources.list.d/devel:kubic:libcontainers:stable:cri-
o:$VERSION.list

curl                                -L
https://download.opensuse.org/repositories/devel:kubic:libcont
ainers:stable:cri-o:$VERSION/$OS/Release.key | apt-key add -
curl                                -L
https://download.opensuse.org/repositories/devel:kubic:libco
ntainers:/stable/$OS/Release.key | apt-key add -

sudo apt-get update

sudo apt-get install cri-o cri-o-runc
```

```
sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl status crio
```

Controller Node Initialize

```
# docker                                1                                CRI-
0

# Controller

kubeadm init --cri-socket=/var/run/crio/crio.sock --ignore-
preflight-errors=all --pod-network-cidr=192.168.0.0/16 --
apiserver-advertise-address=203.248.23.192

# Worker

kubeadm      join      203.248.23.192:6443      --token
9oy7rn.qtw04nfd8ga417p4      --discovery-token-ca-cert-hash
sha256:26218e0c80320b7c23735916c130fc48f644b26314212d917969553
ec0651256
```

[Network] K8S Overlay Network (IPIP -> VXLAN)

K8S Overlay Network

IPIP -> VXLAN

```
)
                                POD가
( pod                            가 )
```

Calico IP-IP Network VXLAN

Node : Controller / Worker01 / Worker02

```
## Controller
# Mode            IPIPMode
calicoctl get ippool -o wide
NAME                      CIDR                      NAT            IPIPMode
VXLANMode    Disabled    DisableBGPExport    Selector
default-ipv4-ippool    192.168.0.0/16    true            Always        Never
false        false
all()
```

```
#        Manifest YAML
kubectl delete -f calico.yaml
```

```
## Controller / Worker
# 가                      tunl0                      가
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 reboot
```

```
## Controller
# Manifest. calico.yaml                      VXLAN

livenessProbe:
  exec:
    command:
      - /bin/calico-node
      - -felix-live
      # - -bird-live                      // VXLAN        bird(BGP)

  periodSeconds: 10
```

```
    initialDelaySeconds: 10
    failureThreshold: 6
    timeoutSeconds: 10
  readinessProbe:
    exec:
      command:
        - /bin/calico-node
        - -felix-ready
        # - -bird-ready    //

# Enable IPIP
- name: CALICO_IPV4POOL_IPIP
  value: "Never"          // Always --> Never

# Enable or Disable VXLAN on the default IP pool.
- name: CALICO_IPV4POOL_VXLAN
  value: "Always"        // Never --> Always
```

```
kind: ConfigMap
```

```
apiVersion: v1
```

```
metadata:
```

```
  name: calico-config
```

```
  namespace: kube-system
```

```
data:
```

```
  # Typha is disabled.
```

```
  typha_service_name: "none"
```

```
  # Configure the backend to use.
```

```
  calico_backend: "vxlan"          // "bird" --> "vxlan"
```

```
.
```

```
#
```

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

```
# Calico Node      .      Ready      .      .
```

```
kubectl get nodes -o wide -A
```

```
# Calico Pod      . kube-system PoD      가      .
```

```
kubectl get pod -o wide -A
```

```
# Calico Type . BIRD
sudo calicoctl node status
Calico process is running.
The BGP backend process (BIRD) is not running.
```

```
# Network VXLANMODE 가 .
calicoctl get ippool -o wide
NAME CIDR NAT IPIPMODE
VXLANMODE DISABLED DISABLEBGPEXPORT SELECTOR
default-ipv4-ippool 192.168.0.0/16 true Never
Always false false
all()
```

```
# tunl0 가 vxlan 가 .
# vxlan 가 .
```

```
hostway@controller:~$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric
Ref Use Iface
0.0.0.0 10.10.10.1 0.0.0.0 UG 0 0
0 ens18
10.10.10.0 0.0.0.0 255.255.255.0 U 0 0
0 ens18 // External (SNAT)
172.17.0.0 0.0.0.0 255.255.0.0 U 0 0
0 docker0 // Container Runtime Bridge
192.168.5.0 192.168.5.0 255.255.255.192 UG 0 0
0 vxlan.calico // Worker01
192.168.30.64 192.168.30.64 255.255.255.192 UG 0 0
0 vxlan.calico // Worker02
192.168.49.0 0.0.0.0 255.255.255.192 U 0 0
0 * // Controller vxlan
192.168.49.1 0.0.0.0 255.255.255.255 UH 0 0
0 cali09ae4a7064b // Node(Worker01)가 GW
192.168.49.2 0.0.0.0 255.255.255.255 UH 0 0
0 cali1fdac863dc5 // Node(Worker02)가 GW
```

```
# Worker
hostway@controller:~$ ip nei | grep vxlan
192.168.5.0 dev vxlan.calico lladdr 66:8c:33:86:44:ce
PERMANENT
192.168.30.64 dev vxlan.calico lladdr 66:fb:72:20:22:a1
```


PERMANENT

```
# VXLAN Traffic Port UDP .  
udp 0 0 0.0.0.0:4789 0.0.0.0:*
```

PoD

```
hostway@controller:~$ kubectl create deployment sampleos --  
image=gcr.io/google-samples/kubernetes-bootcamp:v1 --  
replicas=3
```

```
deployment.apps/sampleos created
```

```
hostway@controller:~$ kubectl get pod -o wide
```

NAME	READY	STATUS	RESTARTS	AGE
IP	NODE	NOMINATED	NODE	READINESS GATES
sampleos-646dc9654b-8xjw9	1/1	Running	0	45s
192.168.5.11	worker01	<none>	<none>	
sampleos-646dc9654b-gxn75	1/1	Running	0	45s
192.168.5.10	worker01	<none>	<none>	
sampleos-646dc9654b-snkxg	1/1	Running	0	45s
192.168.30.75	worker02	<none>	<none>	

VXLAN

// Controller

1) worker01 worker02 POD Ping .

```
hostway@controller:~$ kubectl exec -it
```

```
sampleos-646dc9654b-8xjw9 -- ping 192.168.30.75
```

```
PING 192.168.30.75: 56 data bytes
```

```
64 bytes from 192.168.30.75: icmp_seq=0 ttl=115 time=92.124 ms
```

```
64 bytes from 192.168.30.75: icmp_seq=1 ttl=115 time=79.735 ms
```

```
64 bytes from 192.168.30.75: icmp_seq=2 ttl=115 time=79.233 ms
```

2) tcpdump

```
sudo tcpdump -i ens18 -w vxlan.pcap
```

3) Wireshark . UDP .



[] CentOS 7 Kubernetes Install

CentOS 7 Kubernetes

OS : CentOS 7.6.1810 Minimal
Account : root
- SNAT IP
Controller : 10.10.10.237 SSH:4223
Worker-01 : 10.10.10.204 SSH:4224
Worker-02 : 10.10.10.190 SSH:4225

```
# root . sudo
useradd -d /home/username username
echo "password" | passwd username --stdin
```

```
# su
chmod 700 /usr/bin/su
```

```
# sudoer wheel 가
sed -ie '/wheel/s/$/\:username/' /etc/group
```

```
# Timezone
sudo timedatectl set-timezone Asia/Seoul
```

```
# SWAP OFF
sudo swapoff -a
sudo sed -i -e '/swap/d' /etc/fstab
```

```
# firewalld off
sudo systemctl stop firewalld && sudo systemctl disable
firewalld
```

```
# Selinux
setenforce 0
sudo sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/selinux/config

# Hostname
sudo hostnamectl set-hostname controller
sudo hostnamectl set-hostname worker-01
sudo hostnamectl set-hostname worker-02

## Controller / Worker
#curl -s https://get.docker.com | sudo sh
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

## Check
sudo docker -v
sudo docker ps -a

## Controller / Worker
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

## Docker enable && restart
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

## Packages Repo
sudo cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
```

```
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes
-el7-x86_64
enabled=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

```
## Install
sudo yum install -y kubelet kubeadm kubectl --
disableexcludes=kubernetes
```

Controller Init

```
# Controller. IP API
(Advertise)
sudo kubeadm init --ignore-preflight-errors=all --pod-network-
cidr=192.168.0.0/16 --apiserver-advertise-address=10.10.10.237
```

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

```
# Network Plugin Setting ( Calico )
curl
https://projectcalico.docs.tigera.io/manifests/calico.yaml -O
kubectl apply -f calico.yaml
```

```
# System Namespace ( kube-system ) check. CoreDNS 가
kubectl get pods -o wide -A
```

NAMESPACE	NAME	READY		
STATUS	RESTARTS	AGE	IP	NODE
NOMINATED	NODE	READINESS	GATES	
kube-system	calico-kube-controllers-7c845d499-p85pm	1/1		
Running	0	3m6s	192.168.49.3	controller
<none>	<none>			
kube-system	calico-node-fnm2q	1/1		
Running	0	3m6s	10.10.10.237	controller
<none>	<none>			

```

kube-system    coredns-64897985d-cgvml                1/1
Running       0                5m41s    192.168.49.2    controller
<none>       <none>
kube-system    coredns-64897985d-vdckf                1/1
Running       0                5m42s    192.168.49.1    controller
<none>       <none>
kube-system    etcd-controller                         1/1
Running       0                5m54s    10.10.10.237    controller
<none>       <none>
kube-system    kube-apiserver-controller                1/1
Running       0                5m54s    10.10.10.237    controller
<none>       <none>
kube-system    kube-controller-manager-controller       1/1
Running       0                6m       10.10.10.237    controller
<none>       <none>
kube-system    kube-proxy-nn5zn                        1/1
Running       0                5m42s    10.10.10.237    controller
<none>       <none>
kube-system    kube-scheduler-controller                1/1
Running       0                5m54s    10.10.10.237    controller
<none>       <none>

```

```

# ( ) Multi NIC 가 INTERNAL-IP
가 K8S NIC IP 가
INTERNAL-IP
INTERNAL-IP Init
kubeadm --apiserver-advertise-address IP

```

```

cat << EOF | sudo tee /etc/default/kubelet
KUBELET_EXTRA_ARGS='--node-ip $(hostname -I | cut -d ' ' -f2)'
EOF
sudo systemctl daemon-reload
sudo systemctl restart kubelet
kubectl cluster-info

```

Worker Join

```
# Worker-01 Woker-02 Node User Privileges
```

```
sudo scp /etc/kubernetes/admin.conf
username@10.10.10.204:/home/username/admin.conf
sudo scp /etc/kubernetes/admin.conf
username@10.10.10.190:/home/username/admin.conf
```

```
# Worker
```

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

```
# Worker kubeadm Join .
sudo kubeadm join 10.10.10.237:6443 --token
jgocer.fu65ql39kdod5qi0 \
--discovery-token-ca-cert-hash
sha256:3cb85267e89913d7865d219922daaa8fc6e788dd2be0e2f80fae271
76e2dfe3b
```

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

```
# Check
```

```
kubectl get nodes -o wide
NAME STATUS ROLES AGE VERSION
INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-
VERSION CONTAINER-RUNTIME
controller Ready control-plane,master 16m v1.23.5
10.10.10.237 <none> CentOS Linux 7 (Core)
3.10.0-1062.el7.x86_64 docker://20.10.14
worker-01 Ready <none> 55s v1.23.5
10.10.10.204 <none> CentOS Linux 7 (Core)
3.10.0-1062.el7.x86_64 docker://20.10.14
worker-02 NotReady <none> 38s v1.23.5
10.10.10.190 <none> CentOS Linux 7 (Core)
3.10.0-1062.el7.x86_64 docker://20.10.14
```

```
# Check Pod Create
```

```
kubectl run hello --image=nginx --dry-run=client -o yaml |
kubectl apply -f-
pod/hello created
```

```
[myungin.baek@controller ~]$ kubectl get pods -o wide
NAME READY STATUS RESTARTS AGE IP
NODE NOMINATED NODE READINESS GATES
```

```
hello      1/1      Running    0           42s      192.168.171.1
worker-01  <none>          <none>
```

[] CNI - Calico Plugin

: CNI Calico Network

#1 (controller , worker)

CNI (Container Network Interface)

CNCF
Kubernetes Plugin Kubenet CNI Network

Calico Network?

vRouter (L3)
Kubernetes Network Plugin CNI Network

Document URL :
<https://projectcalico.docs.tigera.io/reference/>

Non-overlay

Direct
- BGP(Border Gateway Protocol) BIRD

Pod Pod 가 .
Node Calico Pod BGP Peer .

. (ex:

)

Overlay Network

Workload IP(ex:)

(Encaptulation) (L2)

Node IP 가 , POD

IP 가 .

IP in IP (Default)

- 가 Direct
IP tunl0(tunneling)

가 Direct 가 BGP (BIRD)

Node (IPVS)

Calico Routing .

VXLAN

- 가

IP in IP

. (ex: Azure)

Calico

BGP

가

VXLAN Node

L2 UDP

IP in IP 가 .

Cross-subnet

가 (가)

가

(/)

()

WireGuard

Calico

가 .

가

Calicoctl

```
Controller      Calico Network
Host            kubectl  plugin
```

Host

```
$ cd /usr/local/bin
```

```
$ sudo curl -L
https://github.com/projectcalico/calico/releases/download/v3.2
2.1/calicoctl-linux-amd64 -o calicoctl
$ sudo chmod +x calicoctl
```

Check

```
Calico 가      Network  Pool  Block
$ sudo calicoctl ipam show --show-blocks
```

```
+-----+-----+-----+-----+
-----+
| GROUPING |          CIDR          | IPS TOTAL | IPS IN USE |
IPS FREE  |
+-----+-----+-----+-----+
-----+
| IP Pool  | 192.168.0.0/16        |    65536 | 5 (0%)     |
65531 (100%) |
| Block   | 192.168.136.0/26     |    64   | 4 (6%)     | 60
(94%)      |
| Block   | 192.168.153.192/26   |    64   | 1 (2%)     | 63
(98%)      |
+-----+-----+-----+-----+
-----+
```

BGP

```
$ sudo calicoctl node status
```

```
Calico process is running.
```

```
IPv4 BGP status
```

```
+-----+-----+-----+-----+
-----+
| PEER ADDRESS | PEER TYPE | STATE | SINCE |
INFO      |
+-----+-----+-----+-----+
```

```

-----+
| 203.248.23.215 | node-to-node mesh | up      | 05:27:05 |
Established |
+-----+-----+-----+-----+-----+
-----+

```

Block

```
$ route -n | egrep "tun|cali|\\"*
```

```

192.168.136.0    0.0.0.0          255.255.255.192 U        0
0              0 *
192.168.136.1    0.0.0.0          255.255.255.255 UH       0        0
0 calibc6c3028870
192.168.136.2    0.0.0.0          255.255.255.255 UH       0        0
0 calid6edae09645
192.168.136.3    0.0.0.0          255.255.255.255 UH       0        0
0 calic6bfd11bfbe
192.168.153.192 203.248.23.215  255.255.255.192 UG       0        0
0 tunl0

```

Pod가 calicxxxxxx

System(default) Namespace -A 가 .

```
$ calicoctl get workloadendpoint -A
```

NAMESPACE	WORKLOAD	INTERFACE	NODE
kube-system	calico-kube-controllers-56fcbf9d6b-nlqg2	calid6edae09645	user-
kube-system	coredns-64897985d-jgj5s	calic6bfd11bfbe	user-
kube-system	coredns-64897985d-vbpn4	calibc6c3028870	user-

Calico Veth type(Pair) .

```

$ ip -br -c link show type veth
calibc6c3028870@if3 UP ee:ee:ee:ee:ee:ee
<BROADCAST,MULTICAST,UP,LOWER_UP>
calid6edae09645@if4 UP ee:ee:ee:ee:ee:ee
<BROADCAST,MULTICAST,UP,LOWER_UP>
calic6bfd11bfbe@if4 UP ee:ee:ee:ee:ee:ee
<BROADCAST,MULTICAST,UP,LOWER_UP>

```

Calico Management Pod

Daemon Pod
Controller Worker Node Pod 가

```
$ kubectl get pods -o wide -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE
calico-kube-controllers-56fcbf9d6b-nlqg2  1/1    Running   0          30m   192.168.136.2  user-controller
calico-node-8cts6                        1/1    Running   0          30m   10.0.2.15     user-controller
calico-node-mb9n6                        1/1    Running   0          29m   10.0.2.15     user-worker
```

Calico DB etcd datastore

```
$ kubectl get pods -o wide -n kube-system | grep -i etcd
etcd-user-controller                1/1    Running   0          39m   10.0.2.15     user-controller
```

Calico Felix

Pod kube-proxy
etcd Pod Network
kube-proxy 가 iptables / ipvs Mode
iptables ipvs

□ IPVS = Hash

```
$ sudo iptables -t nat -S | grep -i cali
$ sudo iptables -t filter -S | grep -i cali
```

Networking

IP in IP Networking

Controller Node Worker Node Pod

❌
1) Controller 192.168.136.2 Pod Worker 192.168.153.193
Pod

```

2) Controller Pod (veth) Pair Host calico
(veth) ARP
3) Host Calico Worker Pod
ARP
4) Calico link-local ( )
HOST 가 BIRD Worker
5) Controller Calico vRouter ARP_Proxy
Worker ARP
6) BIRD Tunl0 --> Host Pod
가
7)

Felix SNAT ( MASQUERADE ) tunl0
HOST ens33 .

```

Packet Check

```

# ( Controllor POD <---> Worker POD ) Ping

$ kubectl get pod -o wide
NAME READY STATUS RESTARTS AGE IP
NODE NOMINATED NODE READINESS GATES
hello-776c774f98-894tt 1/1 Running 0 13d
192.168.153.193 user-worker <none> <none>
hi 1/1 Running 0 13d
192.168.136.5 user-controller <none> <none>

# Worker POD --> Container POD. Ping Pod
Host PID .
$ sudo nsenter -t 225201 -n ping 192.168.136.5
64 bytes from 192.168.136.5: icmp_seq=627 ttl=62 time=0.709 ms
64 bytes from 192.168.136.5: icmp_seq=628 ttl=62 time=0.675 ms
64 bytes from 192.168.136.5: icmp_seq=629 ttl=62 time=0.727 ms
64 bytes from 192.168.136.5: icmp_seq=630 ttl=62 time=0.797 ms
64 bytes from 192.168.136.5: icmp_seq=631 ttl=62 time=0.887 ms

# Controller . IPIP API
, API .
$ sudo tcpdump -i enp0s8 -nn proto 4 -w test.pcap

# Wireshark .

```

1) POD IP ICMP



2) MAC Controller Worker Node API IP

Controller

```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    link/ether 08:00:27:39:ce:bd brd ff:ff:ff:ff:ff:ff
    inet 203.248.23.214/25 brd 203.248.23.255 scope global
enp0s8
```

Worker

```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    link/ether 08:00:27:bc:85:3a brd ff:ff:ff:ff:ff:ff
    inet 203.248.23.215/25 brd 203.248.23.255 scope global
enp0s8
```

3) IPv4 Protocol 2

```
#      Outer IP      POD      Inner IP 2
```



Outer IP 가 InnerIP IP-IP Protocol



4) Messages 가



Network Overlay : <https://ikcoo.tistory.com/117>

[CKA] #2. Pod - 1

[CKA] #2. Pod (1)

CKA 가 URL .

Pod ?

Kubernetes

Pod Network , Pod Container
Pod 가 /pause
Pod Network , /pause 가

Pod Create

yaml dry run

```
$ kubectl run hello --image=nginx --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: hello
  name: hello
spec:
  containers:
  - image: nginx
    name: hello
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

```
# | ( ) (apply) -
```

```
kubectl run hello2 --image=nginx --dry-run=client -o yaml |  
kubectl apply -f -  
kubectl get pods -o wide
```

```
# nodeName 가 pod 가 .  
Taint .  
yaml nodeName 가 .  
$ kubectl run hi --image=nginx --dry-run=client -o yaml >  
hi.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
  creationTimestamp: null  
  labels:  
    run: hi  
  name: hi  
spec:  
  containers:  
  - image: nginx  
    name: hi  
    resources: {}  
  dnsPolicy: ClusterFirst  
  restartPolicy: Always  
  nodeName: user-controller ## Node  
status: {}
```

```
# YAML Pod  
$ kubectl create -f hi.yaml  
$ kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE
IP		NOMINATED	NODE	READINESS
GATES				
hello-776c774f98-894tt	1/1	Running	0	18h
192.168.153.193		<none>		<none>
hi	1/1	Running	0	3m10s
192.168.136.5		<none>		<none>

Pod status

```
$ kubectl describe pod hi
```

```
# Pod . -- kubectl arg --
```

```
$ kubectl exec -it hi -- /bin/bash  
root@hi:/#
```

Pod

```
# --replicas . deployment
```

```
$ kubectl create deployment web --image=nginx --replicas=3  
deployment.apps/web created
```

```
$ kubectl get pods -o wide | grep -i web
```

web-76b56fd968-c2pk9	1/1	Running	0	11s
192.168.153.217	user-worker	<none>		<none>
web-76b56fd968-chr4w	1/1	Running	0	11s
192.168.136.6	user-controller	<none>		<none>
web-76b56fd968-mmdfn	1/1	Running	0	11s
192.168.153.218	user-worker	<none>		<none>

Pod log

```
# Pod info  
kubectl describe pod hi
```

```
# Pod log  
kubectl logs hi
```

```
# journal Log(kubelet)  
sudo journalctl -u kubelet
```

```
# Log .  
hi POD Container .  
/pause 가 Pod . nginx 가
```

```
$ sudo docker ps -a | grep -i hi
```



```
d507d8b298c3    nginx    "/docker-
entrypoint..."    26 hours ago    Up 26 hours
k8s_hi_hi_default_6a1464a1-0fea-4ff8-a5c6-426afe281173_0
d8fb1a992247    k8s.gcr.io/pause:3.6    "/pause"
26 hours ago    Up 26 hours
k8s_POD_hi_default_6a1464a1-0fea-4ff8-a5c6-426afe281173_0
```

```
# nginx Service Container info
$ sudo docker inspect d507d8b298c3
$ sudo docker logs d507d8b298c3
```

```
# Pod Network info
$ sudo docker inspect d8fb1a992247
$ sudo docker logs d8fb1a992247
```

```
# Container . 가
$ sudo docker exec -it d507d8b298c3 ls
bin  docker-entrypoint.d  home  media  proc  sbin  tmp
boot docker-entrypoint.sh  lib   mnt    root  srv   usr
dev  etc                  lib64 opt    run   sys   var
```

```
$ sudo docker exec -it d507d8b298c3 /bin/bash
root@hi:/#
```

```
# Net 가 nsenter .
ip
$ $ sudo docker exec -it d507d8b298c3 ip addr
OCI runtime exec failed: exec failed: container_linux.go:380:
starting container process caused: exec: "ip": executable file
not found in $PATH: unknown
```

```
PID
$ sudo docker inspect --format '{{ .State.Pid }}' d507d8b298c3
1244489
```

```
nsenter PID (pod namespace가
) 가 .
$ sudo nsenter -t 1244489 -n ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

```
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group
default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
4: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1480 qdisc
noqueue state UP group default
    link/ether c6:3d:04:5d:80:82 brd ff:ff:ff:ff:ff:ff link-
netnsid 0
    inet 192.168.136.5/32 scope global eth0
        valid_lft forever preferred_lft forever
```

Pod delete

```
# Pod
$ kubectl delete hi
```

```
# Pod Delete. replicas 가 .
$ kubectl delete deployment web
deployment.apps "web" deleted
$ kubectl get pods -o wide | grep -i web
-
```

Pod : <https://kubernetes.io/docs/concepts/workloads/pods/>
Pod Networking :
<https://www.digitalocean.com/community/tutorials/how-to-inspect-kubernetes-networking>

[] Network Namespace

: CentOS 7.6.1810

: root

Network Namespace

가

Network Space() , , IP

Host

Default Network Namespace Check



```
# Host Network Namespace
$ lsns -t net -o pid,uid,user,command
PID  UID USER  COMMAND
  1    0 root  /sbin/init maybe-ubiquity
```

```
Host PID 1 ( Init )
가 nic( : eth0) lo 가
```

Create Network Namespace

lo 가

```
# test 가 Namespace
$ ip netns add test
```

```
$ ip netns
test
```

Check

PID 가 lsns

```

$ lsns -t net
PID USER TYPE COMMAND
  1 root net /usr/lib/systemd/systemd --switched-root --
system --deserialize 22

```

Namespace Network 1 - 가



```

가 Network Namespace 가
veth
veth HOST <--->

# HOST 가 가 veth type peer pair
$ ip link add veth0 type veth peer name veth1

# HOST veth0/veth1 2 가 가
$ ip -br -c addr
lo UNKNOWN 127.0.0.1/8
ens33 UP 211.239.150.48/23
ens36 UP 192.168.0.2/24
veth1@veth0 DOWN
veth0@veth1 DOWN

```

Namespace Network 2 - 가



```

가 , test

# veth0 test Namespace Set
$ ip link set veth0 netns test

# HOST veth0 test namespace
$ ip -br -c addr
lo UNKNOWN 127.0.0.1/8

```

```

ens33          UP          211.239.150.48/23
ens36          UP          192.168.0.2/24
veth1@if5     DOWN

```

```

test namespace 가 veth0

```

```

# test namespace netns exec
$ ip netns exec test ip -br addr
lo          DOWN
veth0@if4   DOWN

```

Namespace Network 3 - bridge



```

HOST test namespace veth0 veth1 가
DOWN
가 IP
HOST (bridge)
가

```

```

# Check
$ ( 가 ) yum install -y bridge-utils-1.5-9.el7.x86_64
$ brctl show
bridge name      bridge id          STP enabled
interfaces
가 . br0          HOST

```

```

# Bridge Create && Check
$ ip link add br0 type bridge

```

```

$ brctl show
bridge name      bridge id          STP enabled
interfaces
br0              8000.000000000000 no

```

```

# 가
$ ip -br -c addr

```

```
lo                UNKNOWN          127.0.0.1/8
ens33             UP                211.239.150.48/23
ens36             UP                192.168.0.2/24
veth1@if5        DOWN
br0              DOWN
```

```
br0 vethx
```

```
# HOST          veth1  Host  br0
$ ip link set veth1 master br0
```

```
# check bridge veth1 가
$ brctl show
bridge name      bridge id                STP enabled
interfaces
br0              8000.46df623e69e4       no                veth1
```

```
가 , , IP
ifconfig          net-util                ip
```

```
# netns exec          test Namespace  veth0          IP
UP
$ ip netns exec test ip addr add 10.10.10.2/24 dev veth0
$ ip netns exec test ip link set veth0 up
```

```
# host veth1          bridge          up
$ ip link set br0 up
$ ip link set veth1 up
```

```
# UP check          가          UP
$ ip -br -c addr
lo                UNKNOWN          127.0.0.1/8
ens33             UP                211.239.150.48/23
ens36             UP                192.168.0.2/24
veth1@if5        UP
br0              UP
```

```
# test namespace          UP
$ ip netns exec test ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
5: veth0@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP mode DEFAULT group default qlen 1000
    link/ether f2:1c:09:d4:47:fc brd ff:ff:ff:ff:ff:ff link-
netnsid 0
```

```
# lo          가 DOWN          가
. UP          UNKNOWN
$ ip netns exec test ip link set dev lo up
$ ip netns exec test ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
```

```
# Check
$ ip netns exec test ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.063 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.058 ms
```

```
# Check 2
Host
Gateway          IP
Routing          가          ,          ip
```

```
#          IP
```

```
$ ip addr add 10.10.10.200/24 dev br0
```

```
# test          veth0          Ping
$ ping 10.10.10.2
ping 10.10.10.2 -c 2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.073 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.071 ms
```

```
--- 10.10.10.2 ping statistics ---
```

2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.071/0.072/0.073/0.001 ms

Namespace Network 4 -



```
test namespace 가
.
가 , 가 .
# test2 namespace 가 beth0/beth1
# IP test <---> test2 Ping
.
ip netns add test2
ip link add beth0 type veth peer name beth1
ip link set beth0 netns test2
ip link set beth1 master br0
ip netns exec test2 ip addr add 10.10.10.3/24 dev beth0
ip netns exec test2 ip link set beth0 up
ip netns exec test2 ip link set dev lo up
ip link set beth1 up

# test2 namespace
$ ip netns
test2 (id: 1)
test (id: 0)

$ ip -br -c addr
lo UNKNOWN 127.0.0.1/8
ens33 UP 211.239.150.48/23
ens36 UP 192.168.0.2/24
veth1@if5 UP
br0 UP
beth1@if8 UP

$ brctl show
bridge name bridge id STP enabled
interfaces
```



```
br0                8000.2e0e64ccb0e5        no                beth1
                                                           veth1
```

```
# test namespace veth0(10.10.10.2) Ping
ip netns exec test2 ping 10.10.10.2 -c 2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.112 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.076 ms

--- 10.10.10.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.076/0.094/0.112/0.018 ms
```



```
#                ?
HOST            lo (          )                , Host
```

```
    NAT
ip4            FORWARD    HOST
```

```
#    HOST iptables FORWARD    ACCEPT
$ iptables -nL | grep -i forward
Chain FORWARD (policy DROP)
```

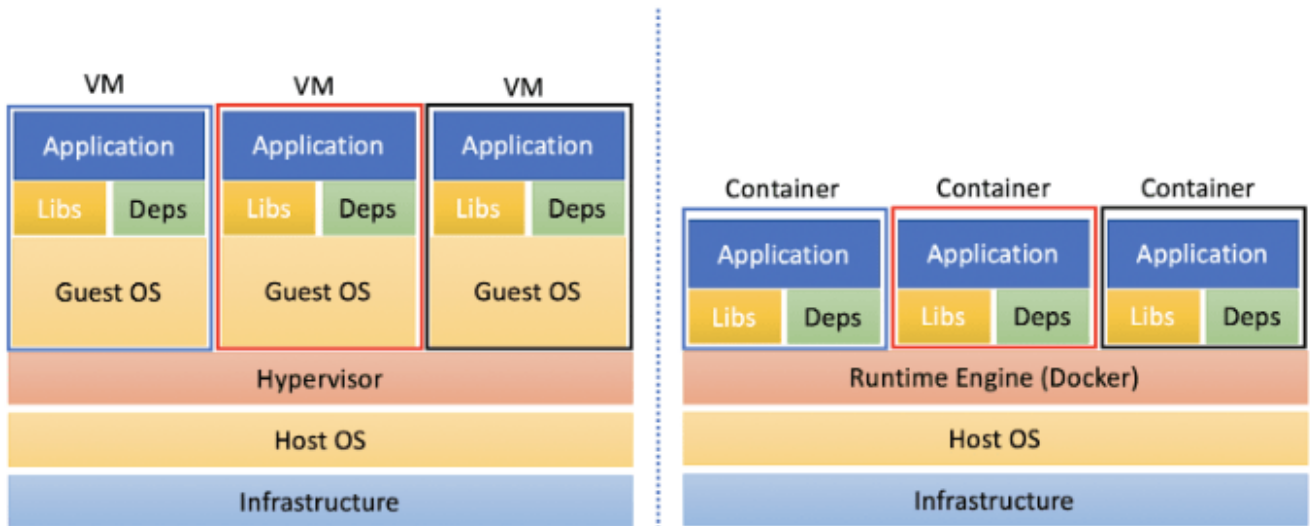
```
#
$ iptables --policy FORWARD ACCEPT
$ iptables -nL | grep -i forward
Chain FORWARD (policy ACCEPT)
```

```
$ service iptables save (                OS                )
```

```
#                ip4v.forward
echo 1 > /proc/sys/net/ipv4/ip_forward
sysctl --system
```

```
# check
#
```

[] VM Container



Container

- (LXC) LXC (:
 Docker) HOST

- Namespace
 cgroup

namespace Host 가
 HOST Linux 가 . VM 가

pid
 user
 uts
 ipc
 mnt
 net

#cgroup Host

Memory
CPU
Network
Device
I/O

- Host ,
Windows OS .
- Container Host 가

VM

- VM Host Hypervisor 가
OS
- Host , 가
Linux/Windows/Other Guest OS
OS

[CKA] #1.

: [CKA] #1.

Kubeneretes

가 CKA
kubeadm .

(VM)

Controller Server : 1EA

Worker Server : 1EA

OS

Ubuntu 20.04 Server Minimal

```

# SWAP
sudo swapoff /swap.img
sudo sed -i -e '/swap.img/d' /etc/fstab

# (regular user) sudo

sudo hostnamectl set-hostname controller
sudo hostnamectl set-hostname worker

```

Traffic Setup

```

# ( : Docker), kube-proxy
iptables

## Container / Worker
netfilter(iptables)

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

```

Container Runtime

```

# CKA POD 가
가 Docker /

## Controller / Worker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

## Check
sudo docker -v
sudo docker ps -a

```

cgroup

```
# cgroup
    OS   cgroup          systemd , docker, kubelet
cgroupfs   가                systemd
```

```
## Controller / Worker
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
```

```
## Docker enable && restart
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

```
## Docker   cgroup driver   ,   cgroupfs   systemd
```

```
sudo docker info | grep -i cgroup
```

```
Cgroup Driver: systemd
Cgroup Version: 1
```

```
#           kebe           /
```

```
## Controller / Worker
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
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

```

Kube Initialize.

```

# Controller Node          init
.
.
--cri-socket :          kubeadm socket
.
.
--pod-network-cidr : pod          network
CoreDNS Service
--apiserver-advertise-address=<ip-address> :
Controller          API
.
## Controller.          IP          API
(Advertise)
sudo kubeadm init --ignore-preflight-errors=all --pod-network-
cidr=192.168.0.0/16          --apiserver-advertise-
address=203.248.23.192

```

```

#          init          가
.
1)          ,          (regular user) + sudo          cluster

```

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

```

```
## Check
```

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE
user1-controller	NotReady	control-plane,master	6m28s

```
v1.23.5
```

```
2) pod network Network Plugin .
```

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/>

```
## Pod Network CoreDNS 가
```

```
(Pending)
```

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

NAMESPACE	NAME	STATUS	RESTARTS	AGE	READY
kube-system	coredns-64897985d-9sj9j	Pending	0	12m	0/1
kube-system	coredns-64897985d-zfl8q	Pending	0	12m	0/1
kube-system	etcd-user1-controller	Running	0	12m	1/1
kube-system	kube-apiserver-user1-controller	Running	0	12m	1/1
kube-system	kube-controller-manager-user1-controller	Running	0	12m	1/1
kube-system	kube-proxy-g5xdv	Running	0	12m	1/1
kube-system	kube-scheduler-user1-controller	Running	0	12m	1/1

```
## Pod Network Plugin Install
```

```
, CKA
```

```
Callico
```

```
Plugin
```

```
curl
```

```
https://projectcalico.docs.tigera.io/manifests/calico.yaml -O
```

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

```
kubectl get nodes
```

```
## Check
```

```
, coredns status 가 Running
```

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

NAMESPACE	NAME	STATUS	RESTARTS	AGE	READY
kube-system	calico-kube-controllers-56fcbf9d6b-bnxz5	Pending	0	20s	0/1
kube-system	calico-node-khp2h	Init:2/3	0	20s	0/1
kube-system	coredns-64897985d-9sj9j	Pending	0	22m	0/1
kube-system	coredns-64897985d-zfl8q	Pending	0	22m	0/1
kube-system	etcd-user1-controller	Running	0	22m	1/1

Multi NIC 가 INTERNAL-IP

```

가 K8S NIC IP 가
INTERNAL-IP
INTERNAL-IP Init
kubeadm --apiserver-advertise-address IP

```

```
# INTERNAL-IP 가 10.0.2.15 ( Calico Network Default )
```

```
$ kubectl get nodes -o wide
```

NAME	STATUS	ROLES	AGE
VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE
KERNEL-VERSION	CONTAINER-RUNTIME		
user-controller	Ready	control-plane,master	44h
v1.23.5	10.0.2.15	<none>	Ubuntu 20.04.1 LTS
5.4.0-64-generic	docker://20.10.14		
user-worker	Ready	<none>	44h
v1.23.5	10.0.2.15	<none>	Ubuntu 20.04.1 LTS
5.4.0-64-generic	docker://20.10.14		

```
# Controller.
```

```
cat << EOF | sudo tee /etc/default/kubelet
```

```
KUBELET_EXTRA_ARGS='--node-ip $(hostname -I | cut -d ' ' -f2)'
```



```

EOF
sudo systemctl daemon-reload
sudo systemctl restart kubelet
kubectl cluster-info

# Worker.
cat << EOF | sudo tee /etc/default/kubelet
KUBELET_EXTRA_ARGS='--node-ip $(hostname -I | cut -d ' ' -f2)'
EOF
sudo systemctl daemon-reload
sudo systemctl restart kubelet

```

```

# Check Internal-IP 가 advertise
$ kubectl get nodes -o wide
NAME                STATUS    ROLES    AGE
VERSION            INTERNAL-IP    EXTERNAL-IP    OS-IMAGE
KERNEL-VERSION     CONTAINER-RUNTIME
user-controller    Ready      control-plane,master    45h
v1.23.5            203.248.23.214    <none>          Ubuntu 20.04.1 LTS
5.4.0-64-generic  docker://20.10.14
user-worker        Ready      <none>        44h
v1.23.5            203.248.23.215    <none>          Ubuntu 20.04.1 LTS
5.4.0-64-generic  docker://20.10.14

```

Worker Controller Join

Then you can join any number of worker nodes by running the following on each as root:

```

root
      Worker      kebeadm      Controller
/etc/kebenertes/admin.conf      Worker

```

```

# Controller
sudo      scp      /etc/kubernetes//admin.conf
vagrant@203.248.23.193:/home/vagrant/admin.conf

```

```

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

```

```
kubeadm join 203.248.23.192:6443 --token
wy1lvq.bk2rze7g9lilg2d9 \
--discovery-token-ca-cert-hash
sha256:f7bc17bb974c804821b21427d500cb96615f66c1fd88cb53c023d8b
2c598d3f7
```

```
가 ignore 가
sudo kubeadm join 203.248.23.192:6443 --token
wy1lvq.bk2rze7g9lilg2d9 --ignore-preflight-errors=all --
discovery-token-ca-cert-hash
sha256:f7bc17bb974c804821b21427d500cb96615f66c1fd88cb53c023d8b
2c598d3f7
```

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.

Check

```
Worker pod 가
kubectl get nodes
NAME STATUS ROLES AGE
VERSION
user1-controller Ready control-plane,master 33m
v1.23.5
user1-worker Ready <none> 84s
v1.23.5
```

```
kubectl get pods --all-namespaces
NAMESPACE NAME READY
STATUS RESTARTS AGE
kube-system calico-kube-controllers-56fcbf9d6b-bnxz5 1/1
Running 0 11m
kube-system calico-node-khp2h 1/1
Running 0 11m
kube-system calico-node-skdl 1/1
Running 0 2m3s
```

```

kube-system    coredns-64897985d-9sj9j    1/1
Running 0      33m
kube-system    coredns-64897985d-zfl8q    1/1
Running 0      33m
kube-system    etcd-user1-controller      1/1
Running 0      33m
kube-system    kube-apiserver-user1-controller 1/1
Running 0      33m
kube-system    kube-controller-manager-user1-controller 1/1
Running 0      33m
kube-system    kube-proxy-g5xdv           1/1
Running 0      33m
kube-system    kube-proxy-m6ztf           1/1
Running 0      2m3s
kube-system    kube-scheduler-user1-controller 1/1
Running 0      33m

```

(Trouble)

All Node

```

sudo systemctl stop kubelet
sudo kubeadm reset -f

```

```

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

```

Network Plugin Status

```

Pod Network      -      Calico
,      Status
      (calicoctl)   가      , Kubectl
.

```

Host

```

$ cd /usr/local/bin
$ sudo curl -L
https://github.com/projectcalico/calico/releases/download/v3.2
2.1/calicoctl-linux-amd64 -o calicoctl
$ sudo chmod +x calicoctl

```

```

# Check
$ calicoctl ipam show --show-blocks
+-----+-----+-----+-----+-----+
-----+
| GROUPING |          CIDR          | IPS TOTAL | IPS IN USE |
IPS FREE  |
+-----+-----+-----+-----+-----+
-----+
| IP Pool  | 192.168.0.0/16        | 65536 | 8 (0%)      |
65528 (100%) |
| Block   | 192.168.136.0/26     | 64 | 3 (5%)      | 61
(95%)      |
| Block   | 192.168.153.192/26  | 64 | 5 (8%)      | 59
(92%)      |
+-----+-----+-----+-----+-----+
-----+

```

Kubernetes Auto Complation

```

#          alias  Tab
echo '' >> ~/.bashrc
echo 'source <(kubectl completion bash)' >> ~/.bashrc
echo 'alias k=kubectl' >> ~/.bashrc
echo 'complete -F __start_kubectl k' >> ~/.bashrc

. ~/.bashrc

```

```

# Check
## Tab
k get nodes -o wide
kubectl get nodes -o wide

```