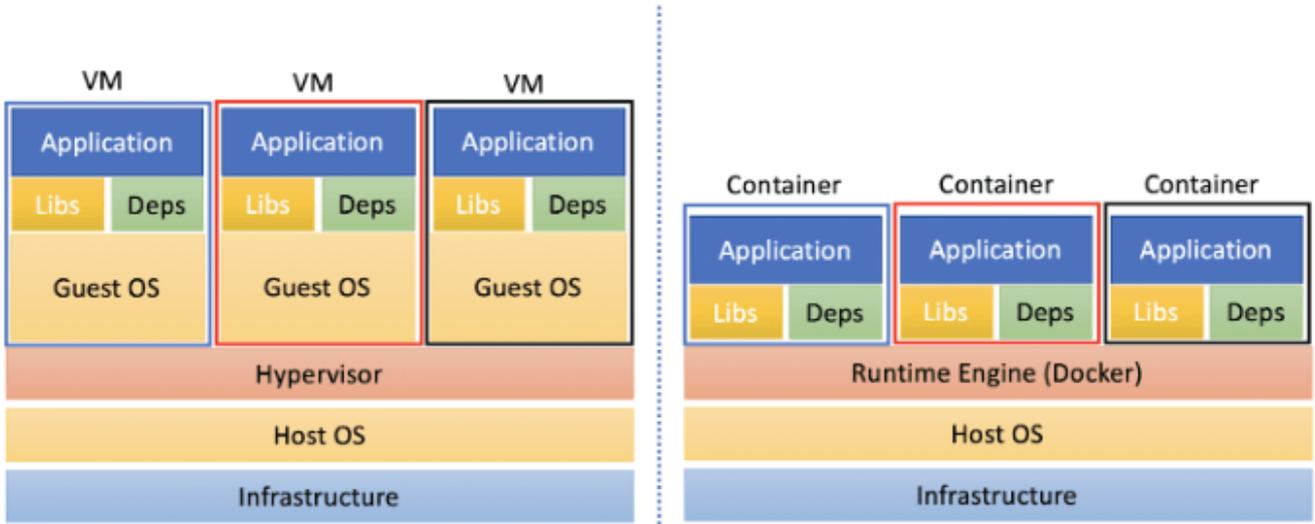


[] VM Container



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.

Kubenerstes

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
```