

2022.12.07 CKA(Certified Kubernetes Administrator)

2022.06.25 (PSI Secure Browser)

;

URL :

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

—

30

Exam

Exam Preparation Checklist

- ✓ Agree to Global Candidate Agreement [Read Now](#)
- ✓ Verify Name Status: Done
- ✓ Select Platform Platform: Ubuntu 20.04
- ✓ Schedule an Exam Exam Date: December 07, 2022 - 01:30PM
Asia/Seoul
- ✓ Check System Requirements Status: System Requirements Checked
- ✓ Get Candidate Handbook [Read Now](#)
- ✓ Read the Important Instructions [Read the Important Instructions](#)
- ✓ Take Exam

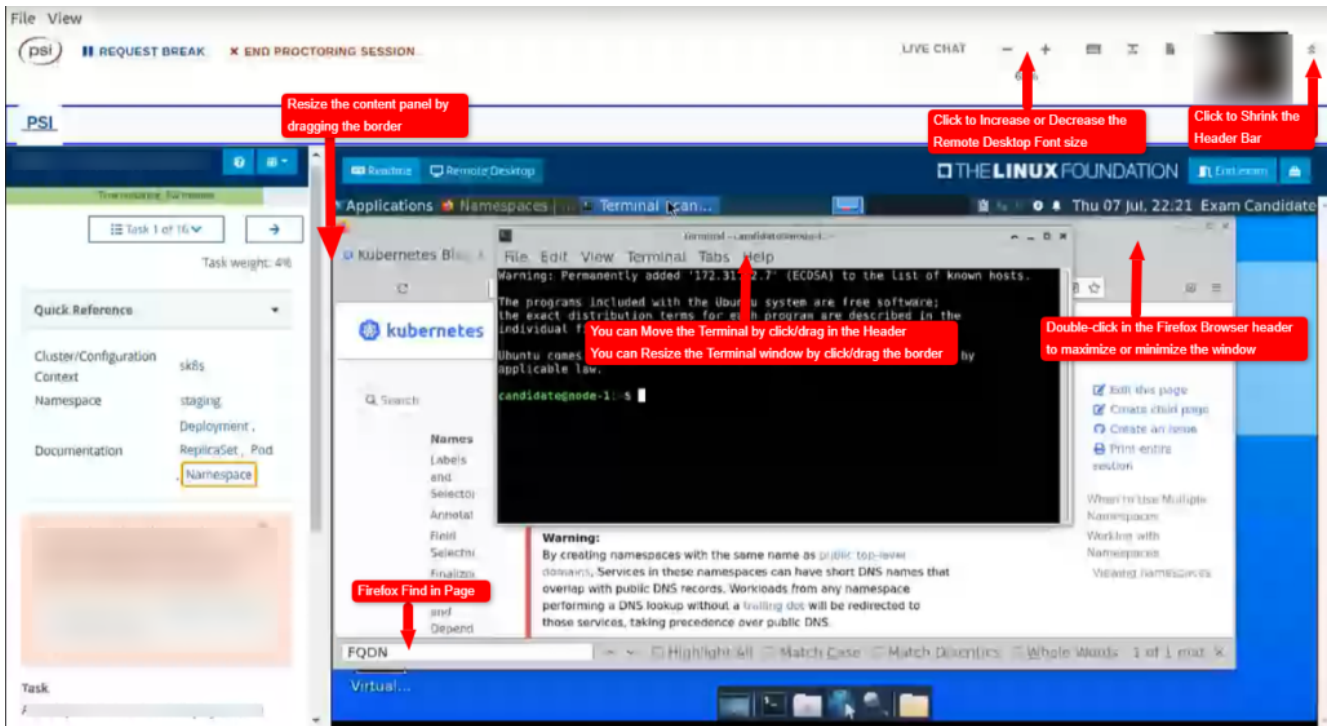
TAKE EXAM

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
가

100 66 (2/3)
24

Pass

24 가

Fail Reschedule

Flag

가

/

[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	DISABLEBGPEXP	SELECTOR
default-ipv4-ippool	192.168.0.0/16	true	Always
false	false		Never

all()

Manifest YAML

kubectl delete -f calico.yml

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.yml 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     DISABLEBGPEXPOR  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	NOMINATED	NODE	READINESS GATES	
sampleos-646dc9654b-8xjw9	1/1	Running	0	45s
192.168.5.11	<none>	worker01	<none>	
sampleos-646dc9654b-gxn75	1/1	Running	0	45s
192.168.5.10	<none>	worker01	<none>	
sampleos-646dc9654b-snkxg	1/1	Running	0	45s
192.168.30.75	<none>	worker02	<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 .
```


vxlan.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.5.11	192.168.49.2	DNS	158	Standard query 0x475e A storage.googleapis.com.default.svc.cluster.local
2	0.000001	192.168.5.11	192.168.49.2	DNS	158	Standard query 0xdf0e AAAA storage.googleapis.com.default.svc.cluster.local
3	0.002360	192.168.49.2	192.168.5.11	DNS	251	Standard query response 0xdf0e No such name AAAA storage.googleapis.com.default.svc.cluster.local
4	0.002958	192.168.49.2	192.168.5.11	DNS	251	Standard query response 0x475e No such name A storage.googleapis.com.default.svc.cluster.local
5	0.003747	192.168.5.11	192.168.49.2	DNS	150	Standard query 0xebca A storage.googleapis.com.svc.cluster.local
6	0.003939	192.168.5.11	192.168.49.2	DNS	150	Standard query 0x7f90 AAAA storage.googleapis.com.svc.cluster.local

> Frame 2: 158 bytes on wire (1264 bits), 158 bytes captured (1264 bits)

> Ethernet II, Src: 76:2d:1c:43:96:bd (76:2d:1c:43:96:bd), Dst: 56:44:d0:06:59:33 (56:44:d0:06:59:33)

> Internet Protocol Version 4, Src: 10.10.10.25, Dst: 10.10.10.26 Worker01 --> Worker02 물리 IP

> User Datagram Protocol, Src Port: 48384, Dst Port: 4789

Source Port: 48384

Destination Port: 4789 VXLAN Port (UDP)

Length: 124

Checksum: 0xab5 [unverified]

[Checksum Status: Unverified]

[Stream index: 0]

> [Timestamps]

UDP payload (116 bytes)

> Virtual eXtensible Local Area Network

> Flags: 0x0800, VXLAN Network ID (VNI)

Group Policy ID: 0

VXLAN Network Identifier (VNI): 4096 VNI 식별

Reserved: 0

> Ethernet II, Src: 66:8c:33:86:44:cc (66:8c:33:86:44:cc), Dst: 66:f7:9a:22:22:c3 (66:f7:9a:22:22:c3)

> Internet Protocol Version 4, Src: 192.168.5.11, Dst: 192.168.49.2 Calico VXLAN Interface

> User Datagram Protocol, Src Port: 47490, Dst Port: 53

> Domain Name System (query)

[] Network Namespace

: CentOS 7.6.1810
: root

Network Namespace

가

Network Space() , IP

Host

Default Network Namespace Check

Local Host



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

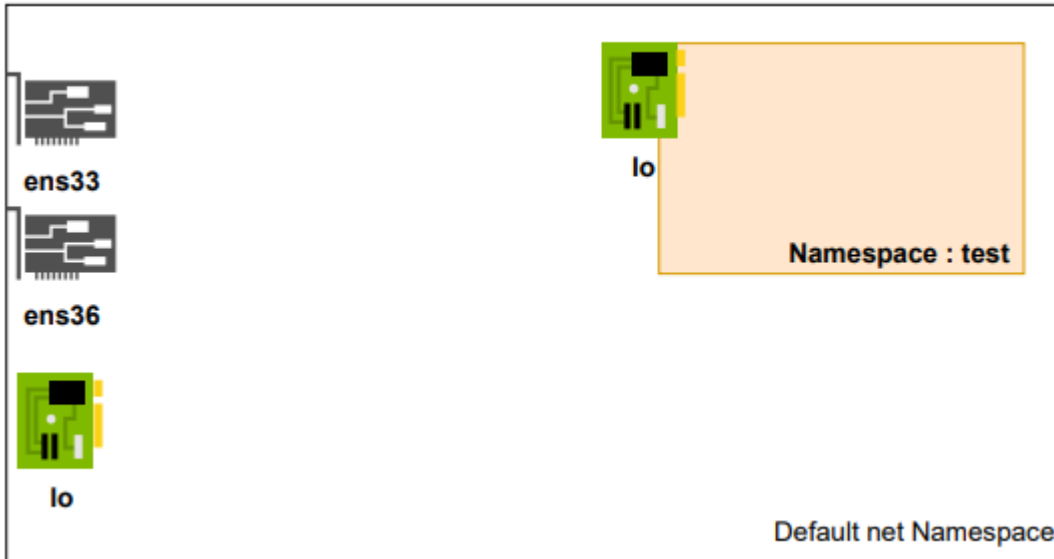
```
Host          PID 1 ( Init )
.
가   nic( : eth0)  lo   가
.
```

Create

Network

Namespace

Local Host



lo 가

```
# test 가 Namespace
```

```
$ ip netns add test
```

```
$ ip netns  
test
```

```
# Check
```

PID 가 lsns

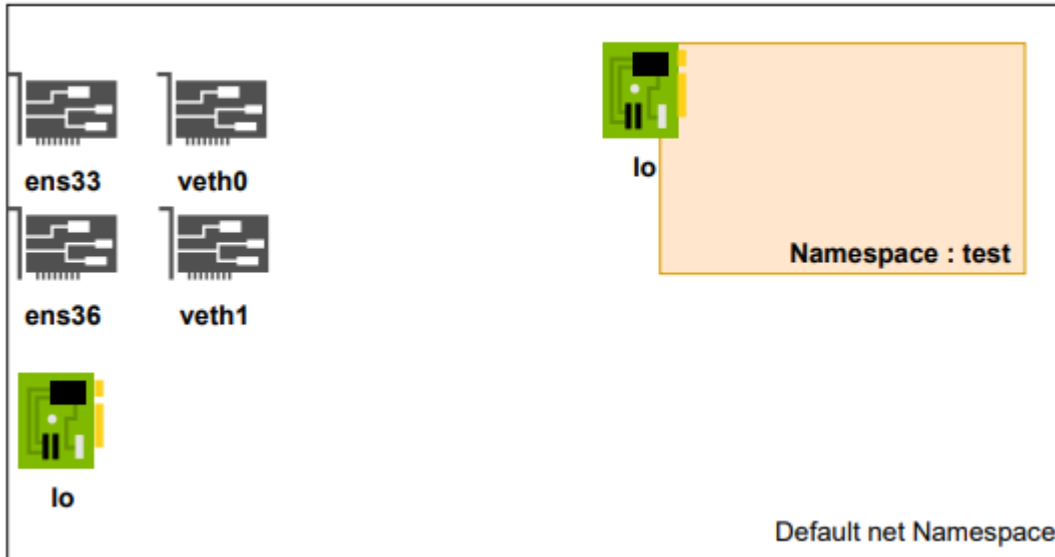
```
$ lsns -t net
```

```
PID USER TYPE COMMAND
```

```
1 root net /usr/lib/systemd/systemd --switched-root --  
system --deserialize 22
```

Namespace Network 1 - 가

Local Host

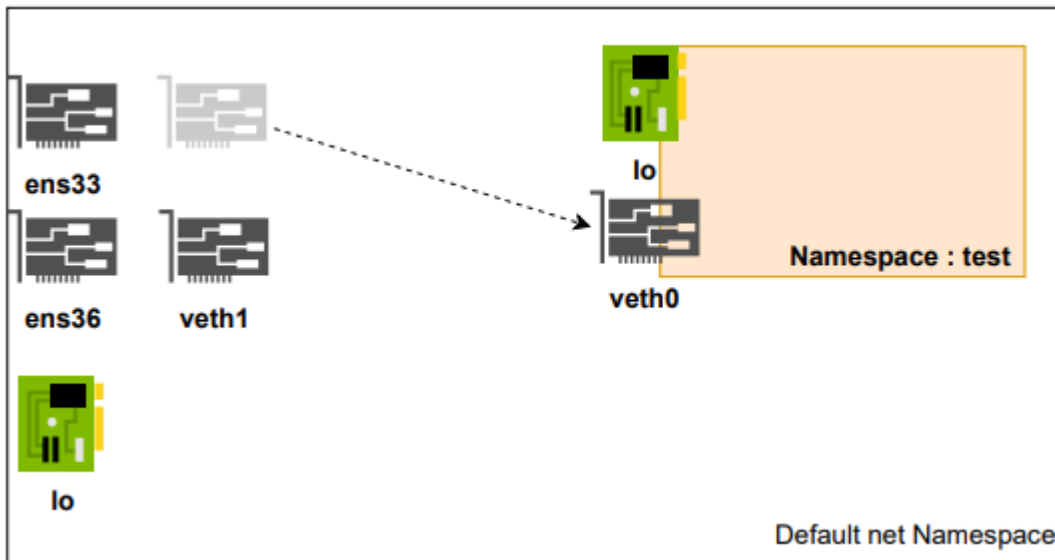


```

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
  
```

Local Host



가 , test

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

```
# HOST veth0 test namespace
```

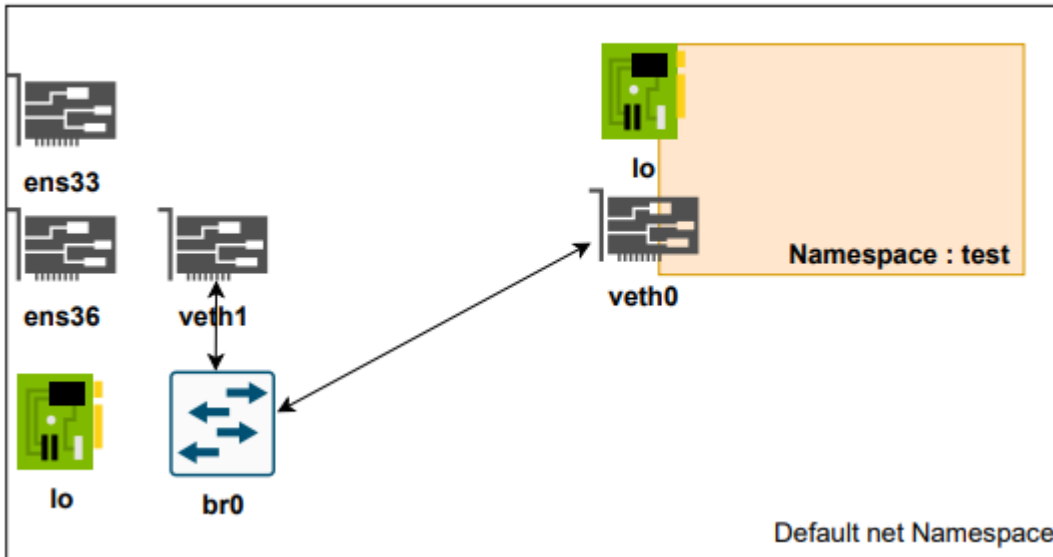
```
$ ip -br -c addr
lo UNKNOWN 127.0.0.1/8
ens33 UP 211.239.150.48/23
ens36 UP 192.168.0.2/24
veth1@if5 DOWN
```

```
test namespace 가 veth0
```

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

Namespace Network 3 - bridge

Local Host



```
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
```

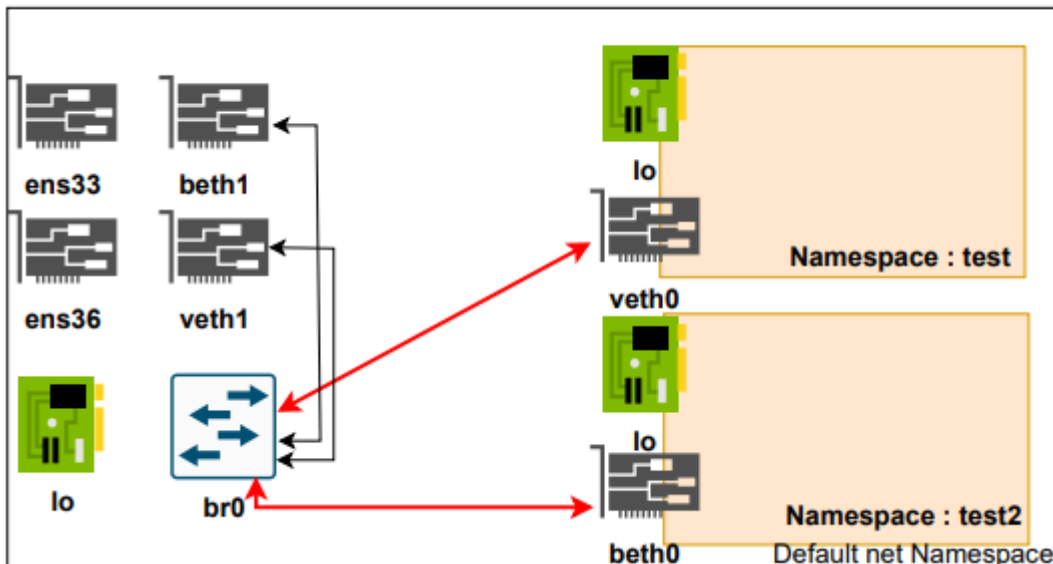
```
--- 10.10.10.2 ping statistics ---
```

```
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 -

Local Host



```
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
```

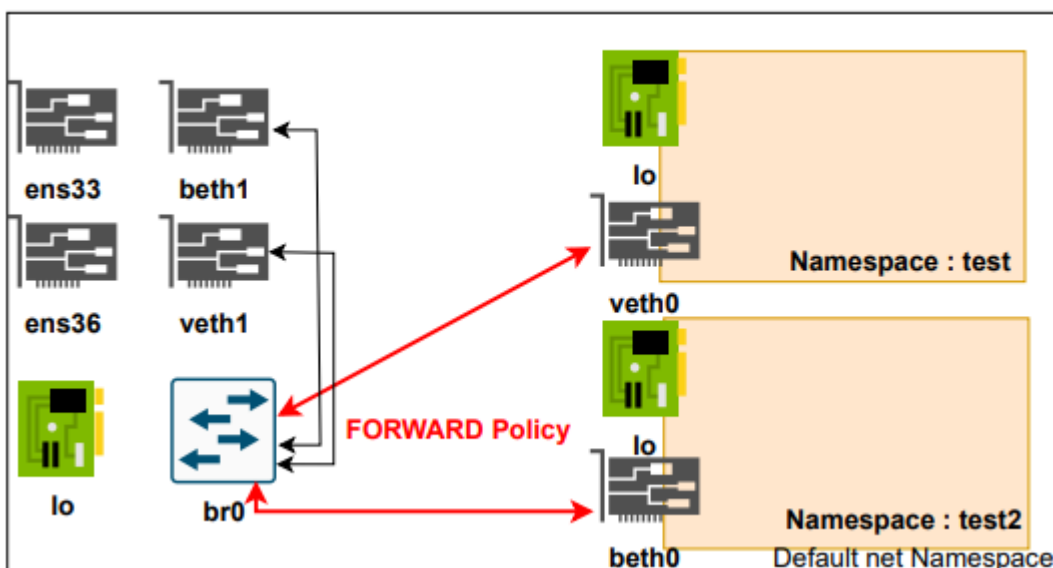
```
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.076 ms
```

```
--- 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
```

Local Host



```
#  
HOST
```

```
?  
lo ( )
```

```
, Host
```

```

NAT
ip4 FORWARD HOST

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

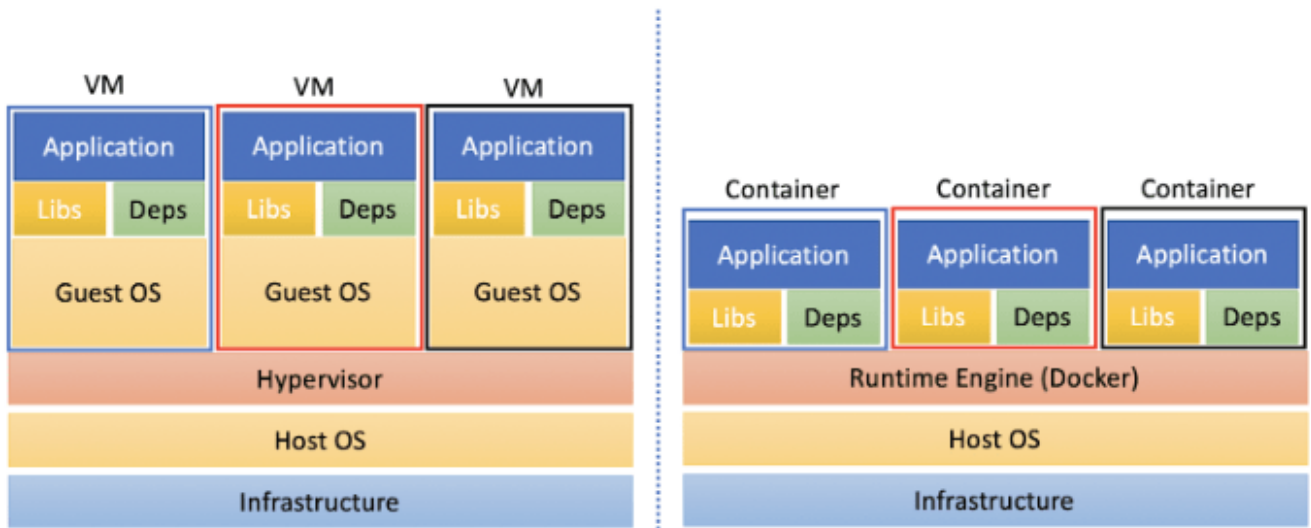
#
$ iptables --policy FORWARD ACCEPT
$ iptables -nL | grep -i forward
Chain FORWARD (policy ACCEPT)

$ 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
Docker / 가

## 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          .
                                   .
                                   가

--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
VERSION

```



```
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 -0
```

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

```
kubectl get nodes
```

```
## Check
```

```
, coredns status 가 Running
```

```

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

```

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/kubernetes/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
wy11vq.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

```