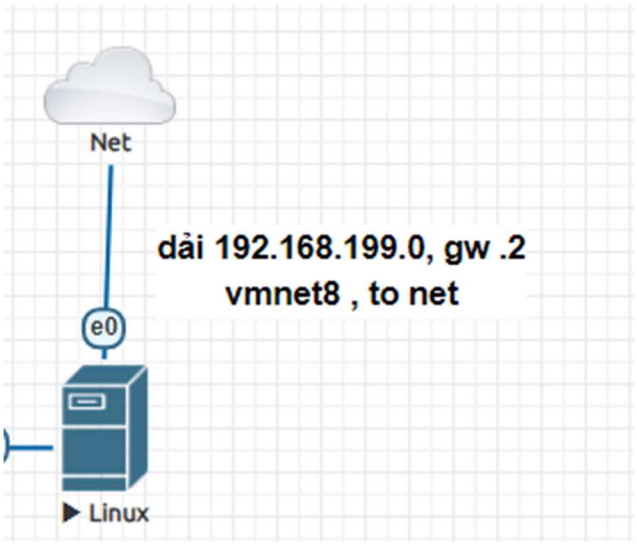
LAB DC SPINE-LEAF + K8S BGP

Lưu ý tài liệu này chưa phù hợp cho các CCNA-er,CCNP-er truyền thống;

Ae ôn phỏng vấn vào Datancenter thì tham khảo thêm))

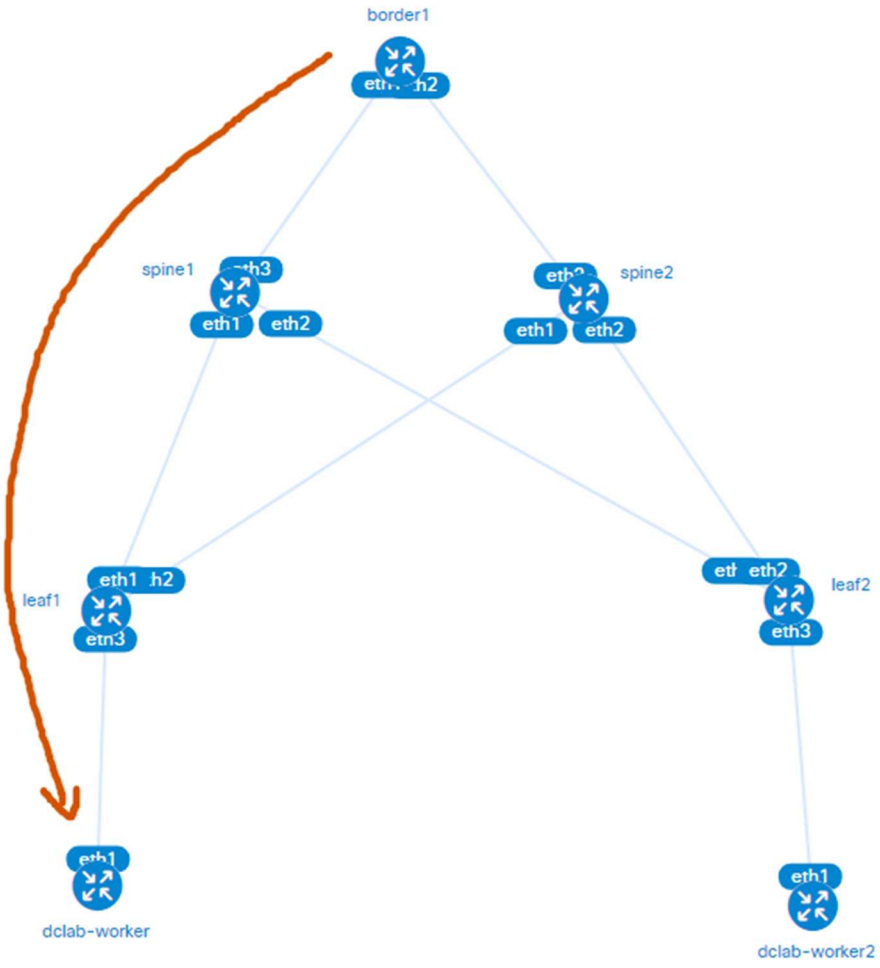
Môi trường xài cho lab này: Ubuntu 22.04, RAM tối thiểu 8GB, 4 CPU

Lấy con ubuntu như này trong eve, rồi cài như dưới ra dc sơ đồ mạng :



Link ubuntu mình dùng https://hainguyenit.edubit.vn/blog/ubuntu-server-v22-de-test-network-trong-eve-ng

Mục tiêu: Từ Border Router ping thẳng vào Pod IP (10.244.x.x) xuyên qua Spine-Leaf BGP



PHẦN 1: CÀI ĐẶT CONTAINERLAB (Lỗi thì search AI thêm)

```
# Cài Docker
curl -fsSL https://get.docker.com | sh
systemctl enable --now docker

# Cài Containerlab
bash -c "$(curl -sL https://get.containerlab.dev)"

# Kiểm tra
clab version
docker --version
```

PHẦN 2: CÀI K8S (KinD) + CALICO BGP

2.1 Cài KinD

```
curl -Lo /usr/local/bin/kind https://kind.sigs.k8s.io/dl/v0.20.0/kind-linux-amd64
chmod +x /usr/local/bin/kind

# Cài kubectl
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

2.2 Tạo cluster config

```
cat > kind-config.yaml << 'EOF'
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
name: dclab
networking:
  disableDefaultCNI: true
  podSubnet: "10.244.0.0/16"
nodes:
- role: control-plane
- role: worker
- role: worker
EOF
```

2.3 Tạo cluster

```
kind create cluster --config kind-config.yaml
kubectl get nodes
# Status = NotReady (chưa cài CNI, bình thường)
```

2.4 Cài Calico (để chạy BGP)

```
# Tigera Operator
kubectl create -f
https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/tigera-operator.yaml

# Calico config
cat > calico-install.yaml << 'EOF'
apiVersion: operator.tigera.io/v1
kind: Installation
metadata:
  name: default
spec:
  calicoNetwork:
    bgp: Enabled
    ipPools:
    - cidr: 10.244.0.0/16
      encapsulation: None
      natOutgoing: Enabled
      nodeSelector: all()
EOF

kubectl apply -f calico-install.yaml

# Đợi ~2 phút
watch kubectl get pods -n calico-system
# Tất cả STATUS = Running thì OK
```

2.5 Cài calicoctl

```
curl -L https://github.com/projectcalico/calico/releases/download/v3.27.0/calicoctl-linux-amd64 -o /usr/local/bin/calicoctl
chmod +x /usr/local/bin/calicoctl
```

2.6 Kiểm tra

```
kubectl get nodes       # Tất cả Ready
kubectl get pods -A     # Tất cả Running
```

PHẦN 3: TẠO SƠ ĐỒ MẠNG CONTAINERLAB

3.1 File daemons FRR

```
cat > daemons << 'EOF'
zebra=yes
bgpd=yes
ospfd=no
ospf6d=no
ripd=no
ripngd=no
isisd=no
pimd=no
ldpd=no
nhrpd=no
eigrpd=no
babeld=no
sharpd=no
staticd=no
pbrd=no
bfdd=no
fabricd=no
vrrpd=no
EOF
```

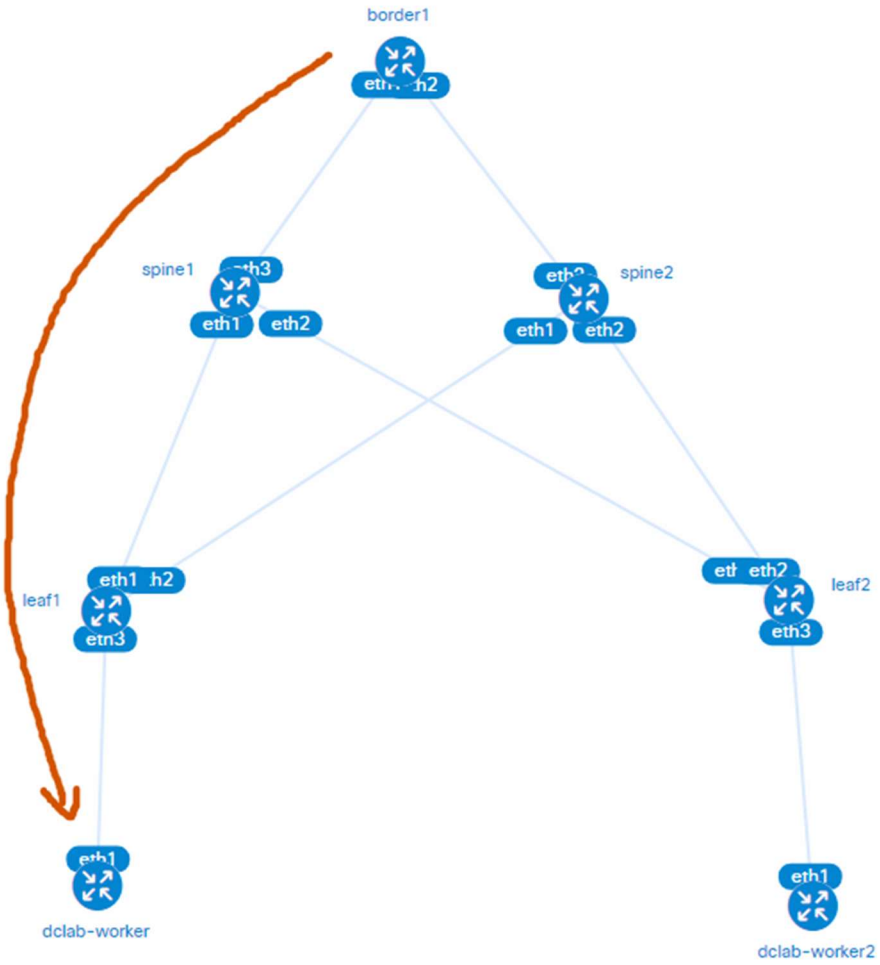
3.2 File topology

```
cat > frr-evpn-k8s.yml << 'EOF'
name: evpn-k8s

topology:
  kinds:
    linux:
      image: quay.io/frrouting/frr:9.1.0

  nodes:
    border1:
      kind: linux
      binds:
        - daemons:/etc/frr/daemons
    spine1:
      kind: linux
      binds:
        - daemons:/etc/frr/daemons
    spine2:
      kind: linux
      binds:
        - daemons:/etc/frr/daemons
    leaf1:
      kind: linux
      binds:
        - daemons:/etc/frr/daemons
    leaf2:
      kind: linux
      binds:
        - daemons:/etc/frr/daemons
    dclab-worker:
      kind: ext-container
    dclab-worker2:
      kind: ext-container

  links:
    # Border <-> Spines
    - endpoints: ["border1:eth1", "spine1:eth3"]
```

```
    - endpoints: ["border1:eth2", "spine2:eth3"]
    # Spine1 <-> Leafs
    - endpoints: ["spine1:eth1", "leaf1:eth1"]
    - endpoints: ["spine1:eth2", "leaf2:eth1"]
    # Spine2 <-> Leafs
    - endpoints: ["spine2:eth1", "leaf1:eth2"]
    - endpoints: ["spine2:eth2", "leaf2:eth2"]
    # Leaf <-> K8s Workers
    - endpoints: ["leaf1:eth3", "dclab-worker:eth1"]
    - endpoints: ["leaf2:eth3", "dclab-worker2:eth1"]
EOF
```

3.3 Deploy + Graph

```
clab deploy -t frr-evpn-k8s.yml
nohup clab graph -t frr-evpn-k8s.yml &
# Truy cập http://<IP_UBUNTU>:50080
```



PHẦN 4: GÁN IP

```
# Border
docker exec clab-evpn-k8s-border1 ip addr add 10.0.1.0/31 dev eth1
docker exec clab-evpn-k8s-border1 ip addr add 10.0.2.0/31 dev eth2
docker exec clab-evpn-k8s-border1 ip link set eth1 up
docker exec clab-evpn-k8s-border1 ip link set eth2 up

# Spine1
docker exec clab-evpn-k8s-spine1 ip addr add 10.0.1.1/31 dev eth3
docker exec clab-evpn-k8s-spine1 ip link set eth3 up

# Spine2
docker exec clab-evpn-k8s-spine2 ip addr add 10.0.2.1/31 dev eth3
docker exec clab-evpn-k8s-spine2 ip link set eth3 up

# Leaf1
docker exec clab-evpn-k8s-leaf1 ip addr add 10.1.1.1/30 dev eth3
docker exec clab-evpn-k8s-leaf1 ip link set eth3 up

# Leaf2
docker exec clab-evpn-k8s-leaf2 ip addr add 10.1.2.1/30 dev eth3
docker exec clab-evpn-k8s-leaf2 ip link set eth3 up

# Worker1
```

```
docker exec dclab-worker ip addr add 10.1.1.2/30 dev eth1
docker exec dclab-worker ip link set eth1 up
docker exec dclab-worker sysctl -w net.ipv4.ip_forward=1

# Worker2
docker exec dclab-worker2 ip addr add 10.1.2.2/30 dev eth1
docker exec dclab-worker2 ip link set eth1 up
docker exec dclab-worker2 sysctl -w net.ipv4.ip_forward=1
```

PHẦN 5: CẤU HÌNH BGP

Router Border (AS 65500)

```
docker exec clab-evpn-k8s-border1 vtysh << 'VTYSH'
conf t
interface lo
 ip address 10.0.0.100/32
exit
route-map RM_ALLOW permit 10
exit
router bgp 65500
 bgp router-id 10.0.0.100
 neighbor 10.0.1.1 remote-as 65000
 neighbor 10.0.2.1 remote-as 65000
 address-family ipv4 unicast
  redistribute connected
  neighbor 10.0.1.1 route-map RM_ALLOW in
  neighbor 10.0.1.1 route-map RM_ALLOW out
  neighbor 10.0.2.1 route-map RM_ALLOW in
  neighbor 10.0.2.1 route-map RM_ALLOW out
 exit-address-family
exit
end
write memory
VTYSH
```

Router Spine1 (AS 65000)

```
docker exec clab-evpn-k8s-spine1 vtysh << 'VTYSH'
conf t
interface lo
 ip address 10.0.0.1/32
exit
interface eth1
 ipv6 nd ra-interval 5
 no ipv6 nd suppress-ra
exit
interface eth2
 ipv6 nd ra-interval 5
 no ipv6 nd suppress-ra
exit
route-map RM_ALLOW permit 10
exit
router bgp 65000
 bgp router-id 10.0.0.1
 neighbor 10.0.1.0 remote-as 65500
 neighbor eth1 interface remote-as 65011
 neighbor eth2 interface remote-as 65012
 address-family ipv4 unicast
  redistribute connected
  neighbor 10.0.1.0 route-map RM_ALLOW in
  neighbor 10.0.1.0 route-map RM_ALLOW out
  neighbor eth1 route-map RM_ALLOW in
  neighbor eth1 route-map RM_ALLOW out
  neighbor eth2 route-map RM_ALLOW in
  neighbor eth2 route-map RM_ALLOW out
 exit-address-family
exit
end
write memory
VTYSH
```

Router Spine2 (AS 65000)

```
docker exec clab-evpn-k8s-spine2 vtysh << 'VTYSH'
conf t
interface lo
 ip address 10.0.0.2/32
exit
interface eth1
 ipv6 nd ra-interval 5
 no ipv6 nd suppress-ra
exit
interface eth2
 ipv6 nd ra-interval 5
 no ipv6 nd suppress-ra
exit
route-map RM_ALLOW permit 10
exit
router bgp 65000
 bgp router-id 10.0.0.2
 neighbor 10.0.2.0 remote-as 65500

##Lưu ý ở đây, là 2 spine ko cần peer trực tiếp với nhau như cisco truyền thống##
 neighbor eth1 interface remote-as 65011
 neighbor eth2 interface remote-as 65012
 address-family ipv4 unicast
  redistribute connected
  neighbor 10.0.2.0 route-map RM_ALLOW in
  neighbor 10.0.2.0 route-map RM_ALLOW out
  neighbor eth1 route-map RM_ALLOW in
  neighbor eth1 route-map RM_ALLOW out
  neighbor eth2 route-map RM_ALLOW in
  neighbor eth2 route-map RM_ALLOW out
 exit-address-family
exit
end
write memory
VTYSH
```

Router Leaf1 (AS 65011)

```
docker exec clab-evpn-k8s-leaf1 vtysh << 'VTYSH'
conf t
interface lo
 ip address 10.0.0.11/32
exit
interface eth1
 ipv6 nd ra-interval 5
 no ipv6 nd suppress-ra
exit
interface eth2
 ipv6 nd ra-interval 5
 no ipv6 nd suppress-ra
exit
route-map RM_ALLOW permit 10
exit
router bgp 65011
 bgp router-id 10.0.0.11
 neighbor eth1 interface remote-as 65000
 neighbor eth2 interface remote-as 65000
 neighbor 10.1.1.2 remote-as 65100

##Lưu ý ở đây, là 2 leaf ko cần peer trực tiếp với nhau như cisco truyền thống##
 address-family ipv4 unicast
  redistribute connected
  neighbor eth1 route-map RM_ALLOW in
```

```
  neighbor eth1 route-map RM_ALLOW out
  neighbor eth2 route-map RM_ALLOW in
  neighbor eth2 route-map RM_ALLOW out
  neighbor 10.1.1.2 route-map RM_ALLOW in
  neighbor 10.1.1.2 route-map RM_ALLOW out
 exit-address-family
exit
end
write memory
VTYSH
```

Router Leaf2 (AS 65012)

```
docker exec clab-evpn-k8s-leaf2 vtysh << 'VTYSH'
conf t
interface lo
 ip address 10.0.0.12/32
exit
interface eth1
 ipv6 nd ra-interval 5
 no ipv6 nd suppress-ra
exit
interface eth2
 ipv6 nd ra-interval 5
 no ipv6 nd suppress-ra
exit
route-map RM_ALLOW permit 10 ##Luôn cần route-map allow thì nó mới cho quảng bá route, Cisco
ko cần
exit
router bgp 65012
 bgp router-id 10.0.0.12
 neighbor eth1 interface remote-as 65000
 neighbor eth2 interface remote-as 65000
 neighbor 10.1.2.2 remote-as 65100 ##đây là peer với con node k8s, cũng chạy BGP luôn
 address-family ipv4 unicast
  redistribute connected
  neighbor eth1 route-map RM_ALLOW in
  neighbor eth1 route-map RM_ALLOW out
  neighbor eth2 route-map RM_ALLOW in
  neighbor eth2 route-map RM_ALLOW out
  neighbor 10.1.2.2 route-map RM_ALLOW in
  neighbor 10.1.2.2 route-map RM_ALLOW out
 exit-address-family
exit
end
write memory
VTYSH
```

PHẦN 6: CẤU HÌNH BGP TRÊN NODE K8S


```
# SSH vào Ubuntu chủ (hoặc mở console EVE-NG)
# Rồi gõ ngay tại shell root@ubuntu22-server:~#

   cat > bgp-config.yaml << 'EOF'
   apiVersion: projectcalico.org/v3
   kind: BGPConfiguration
   metadata:
     name: default
   spec:
     nodeToNodeMeshEnabled: true
     asNumber: 65100
   ---
   apiVersion: projectcalico.org/v3
   kind: BGPPeer
   metadata:
     name: leaf1-peer
   spec:
     node: dclab-worker
     peerIP: 10.1.1.1
     asNumber: 65011
   ---
   apiVersion: projectcalico.org/v3
   kind: BGPPeer
   metadata:
     name: leaf2-peer
   spec:
     node: dclab-worker2
     peerIP: 10.1.2.1
     asNumber: 65012
   EOF

   calicoctl apply -f bgp-config.yaml
```

Fix RPF + iptables trên Workers

```
   # Tắt RPF
   docker exec dclab-worker sysctl -w net.ipv4.conf.eth1.rp_filter=0
   docker exec dclab-worker sysctl -w net.ipv4.conf.all.rp_filter=0
   docker exec dclab-worker2 sysctl -w net.ipv4.conf.eth1.rp_filter=0
   docker exec dclab-worker2 sysctl -w net.ipv4.conf.all.rp_filter=0

   # Allow forwarding qua eth1
   docker exec dclab-worker iptables -I FORWARD -i eth1 -j ACCEPT
   docker exec dclab-worker iptables -I FORWARD -o eth1 -j ACCEPT
   docker exec dclab-worker2 iptables -I FORWARD -i eth1 -j ACCEPT
   docker exec dclab-worker2 iptables -I FORWARD -o eth1 -j ACCEPT
```

Lưu ý phần K8s này khá loằng ngoằng cho ae ta, (chỉ quen làm mạng); nếu lỗi thì hỏi thêm AI

PHẦN 7: VERIFY

7.1 Tạo Pod test

```
kubectl create deployment web-server --image=nginx
kubectl run net-tester --image=nicolaka/netshoot --command -- sleep 3600
kubectl get pods -o wide
```

7.2 Kiểm tra BGP

```
# BGP trên tất cả router
for r in border1 spine1 spine2 leaf1 leaf2; do
  echo "=== $r ==="
  docker exec clab-evpn-k8s-$r vtysh -c "show bgp ipv4 unicast summary"
done

# Calico BGP
calicoctl node status
```

7.3 Border phải thấy Pod subnet

```
docker exec clab-evpn-k8s-border1 vtysh -c "show ip route" | grep 10.244
# B>* 10.244.165.128/26 ...
# B>* 10.244.181.128/26 ...
```

```
  *                       via 10.0.2.1, eth2, weight 1, 00:01:39
B>* 10.244.165.128/26 [20/0] via 10.0.1.1, eth1, weight 1, 00:00:58
  *                          via 10.0.2.1, eth2, weight 1, 00:00:58
B>* 10.244.181.128/26 [20/0] via 10.0.1.1, eth1, weight 1, 00:00:58
  *                          via 10.0.2.1, eth2, weight 1, 00:00:58
C>* 172.20.20.0/24 is directly connected, eth0, 00:14:46
border1#
```

```
border1# show ip bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 10.0.0.100, local AS number 65500 vrf-id 0
BGP table version 12
RIB entries 23, using 2208 bytes of memory
Peers 2, using 26 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
10.0.1.1        4      65000        41        32       12    0    0 00:03:44            9
10.0.2.1        4      65000        41        39       12    0    0 00:03:44            9

Total number of neighbors 2
border1#
```

7.4 Ping từ Border → Pod

root@ubuntu22-server:~# docker exec -it clab-evpn-k8s-border1 vtysh

```
border1# ping 10.244.181.135
PING 10.244.181.135 (10.244.181.135): 56 data bytes
64 bytes from 10.244.181.135: seq=0 ttl=61 time=0.794 ms
64 bytes from 10.244.181.135: seq=1 ttl=61 time=0.170 ms
64 bytes from 10.244.181.135: seq=2 ttl=61 time=0.431 ms
64 bytes from 10.244.181.135: seq=3 ttl=61 time=0.123 ms
```