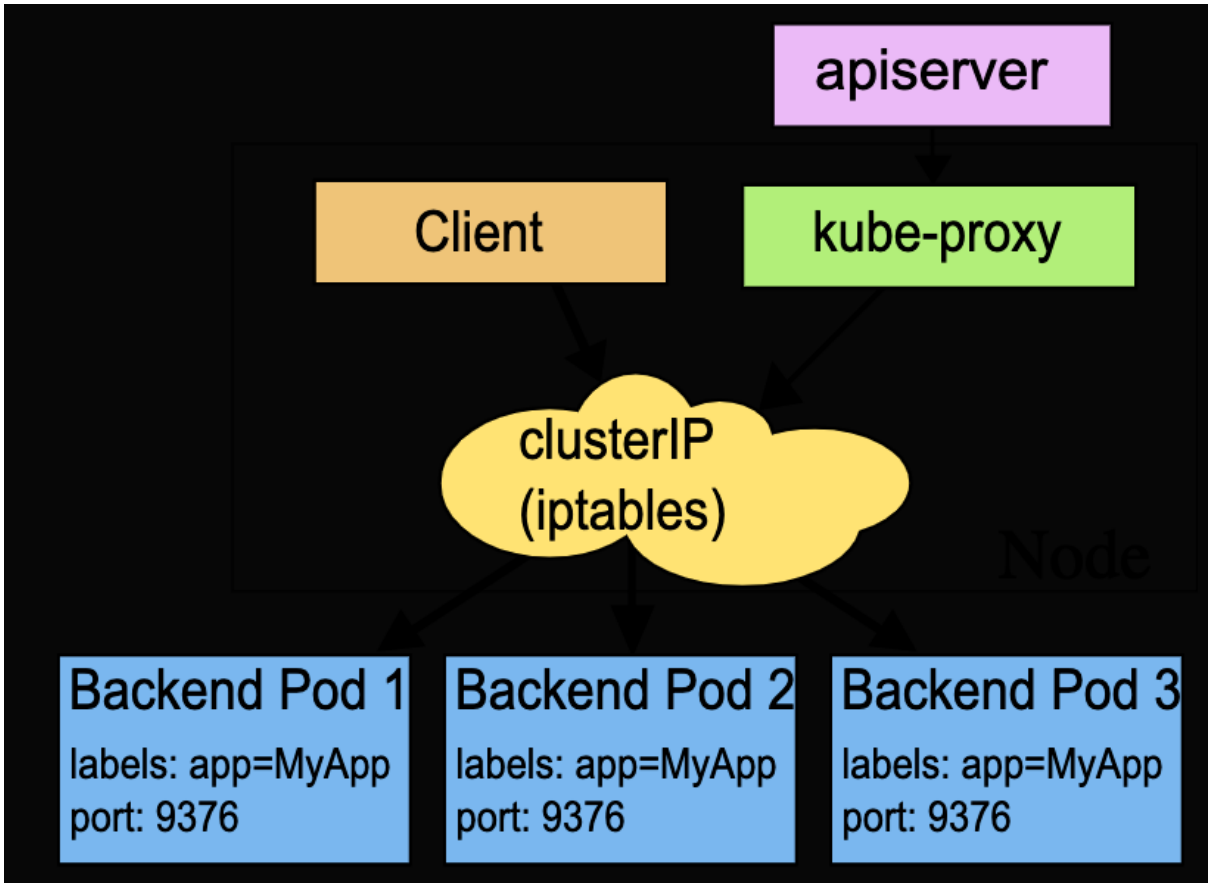
Wireguard在kubernetes中的應用

作者：解決方案架構部-梁帥

需求

Kubernetes容器管理平台現階段已經成為了主流應用交付底層平台，利用kubernetes默認提供的網絡服務模型可以直接對客戶端提供對應的服務（暴露端口），基本原理模型如圖：



备注：

- backend pod 獲取kubernetes分配的pod CIDR地址段
- clusterIP通過kube-proxy暴露backend POD提供的服務端口，默認情況下clusterIP僅僅為kubernetes內部可以訪問，外部客戶端訪問不了

需求

- 那麼如何在不採用第三方CNI插件/NodePort/LoadBalancer提供的能力基礎上能完成對外暴露backend pod和clusterIP呢?

解決方案

使用Wireguard VPN可以很好的解決暴露kubernetes內部IP地址的需求，Wireguard本身是Layer3加密的VPN隧道技術，目前已經完全集成在了linux kernel中，版本號為5.6，其主要特點為速度快，配置簡潔，對於管理員維護成本低，適配各種主流OS操作系統，用戶僅僅需要簡單的配置即可完成VPN的鏈接，本文將介紹如何搭建wireguard server和在kubernetes中的應用。

測試場景介紹

- 互聯網客戶端對kubernetes內部的IP地址直接訪問鏈接

- Multi-cluster kubernetes之間的內部IP地址透傳

測試環境搭建

- 準備獨立的2個kubernetes環境
- 終端設備

環境介紹

Aa kubernetes version	OS	version
kubernetes version	ubuntu20.04	v1.20.2
OS	ubuntu20.04	5.6(默認為5.4)
client	macos ubuntu20.04 windows	windows10 19042 macos BigSur 11.1 ubuntu desktop 20.04

IP地址規劃

Aa 節點名稱	IP地址	pod/cluster IP CIDR
kubernetes slave2	GW:10.211.55.1 Nic:10.211.55.7/24	pod: 10.255.0.0/16 clusterIP: 10.22.22.0/24
kubernetes slave3	GW:10.211.55.1 Nic:10.211.55.8/24	pod: 10.233.0.0/16 clusterIP: 10.33.33.0/24

升級Linux內核

```
sudo -i #使用root權限
mkdir kernel #創建kernel文件夾
cd kernel #進入kernel文件夾
wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.6/linux-headers-5.6.0-0
50600_5.6.0-050600.202003292333_all.deb;

wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.6/linux-headers-5.6.0-0
50600-generic_5.6.0-050600.202003292333_amd64.deb;

wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.6/linux-image-unsigned-
5.6.0-050600-generic_5.6.0-050600.202003292333_amd64.deb;

wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.6/linux-modules-5.6.0-0
50600-generic_5.6.0-050600.202003292333_amd64.deb;
#下載5.6版本內核
```

```
dpkg -i *.deb #安裝
reboot #重啟系統
rm -rfv /lib/modules/5.4* #刪除老版本內核
```

部署kubernetes測試環境

```
#!bin/bash
sudo -i

echo "turn off swap" #關閉虛擬內存
swapoff -a;
sed -i '/ swap / s/^/#/' /etc/fstab;
sleep 3

echo "apt update+install curl" #apt軟件包更新+安裝curl
apt-get update && apt-get install -y apt-transport-https curl;
apt install -y gnupg2;
apt install -y jq;

echo "add apt-key" #添加apt-key
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -;

echo "add source-list" #添加kubernetes更新源
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF

echo deb https://apt.kubernetes.io/ kubernetes-xenial main > /etc/apt/sources.lis
t.d/kubernetes.list

echo "docker installation" #安裝docker
apt-get update;
apt-get install -y docker.io;
#修改systemd去維護cgroup進程
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
# 開啟docker進程&安裝kubernetes
mkdir -p /etc/systemd/system/docker.service.d;
systemctl start docker;
systemctl enable docker;
apt-get install -y kubelet kubeadm kubectl;
```

kubeadm初始化（地址分配請參考IP地址規劃表格）

```
##slave2
kubeadm init --apiserver-advertise-address 10.211.55.7 --pod-network-cidr 10.255.
0.0/16 --service-cidr 10.22.22.0/24; --ignore-preflight-errors=all
##slave3
kubeadm init --apiserver-advertise-address 10.211.55.8 --pod-network-cidr 10.233.
0.0/16 --service-cidr 10.33.33.0/24; --ignore-preflight-errors=all
```

kubectl初始化

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

kubernetes master節點訂閱（默認情況下kubernetes不允許用戶在master節點創建pod，本次測試將修改這個默認行為，所有測試pod都在master節點上存在）

```
kubectl taint nodes --all node-role.kubernetes.io/master-;
```

部署CNI插件（如果沒有CNI插件的情況下，所有的kubernetes node狀態為not ready，後期部署的測試pod均為pending狀態）本環境使用calico 3.14版本CNI插件，建議將yaml配置下載到本地，以方便後期維護

```
#下載calico.yaml配置文件
wget https://docs.projectcalico.org/archive/v3.14/manifests/calico.yaml
#安裝calico CNI插件
kubectl apply -f https://docs.projectcalico.org/archive/v3.14/manifests/calico.ya
ml
```

kube-system pod狀態檢查

```
watch kubectl get pods --all-namespaces
NAMESPACE    NAME                                       READY   STATUS    RESTARTS
AGE
kube-system  calico-kube-controllers-6ff88bf6d4-tgtzb   1/1     Running   0
2m45s
kube-system  calico-node-24h85                          1/1     Running   0
2m43s
kube-system  coredns-846jhw23g9-9af73                   1/1     Running   0
4m5s
kube-system  etcd-jbaker-1                              1/1     Running   0
6m22s
kube-system  kube-apiserver-jbaker-1                    1/1     Running   0
6m12s
kube-system  kube-controller-manager-jbaker-1           1/1     Running   0
```

```
6m16s
kube-system  kube-proxy-8fzp2                          1/1    Running  0
5m16s
kube-system  kube-scheduler-jbaker-1                   1/1    Running  0
5m41s
```

檢查kubectl與kubernetes API server鏈接(node節點狀態為Ready)

```
##cluster1
kubectl get nodes -owide
NAME         STATUS   ROLES                  AGE     VERSION   INTERNAL-IP   EXTE
RNAL-IP   OS-IMAGE             KERNEL-VERSION         CONTAINER-RUNTIME
k8s-slave2   Ready    control-plane,master   3d15h   v1.20.2   10.211.55.7   <non
e>        Ubuntu 20.04.1 LTS   5.6.0-050600-generic   containerd://1.3.3-0ubuntu
2.2
##cluster2
kubectl get node -o wide
NAME         STATUS   ROLES                  AGE    VERSION   INTERNAL-IP   EXTER
NAL-IP   OS-IMAGE             KERNEL-VERSION         CONTAINER-RUNTIME
k8s-slave3   Ready    control-plane,master   2d2h   v1.20.2   10.211.55.8   <none
>        Ubuntu 20.04.1 LTS   5.6.0-050600-generic   docker://19.3.8
```

在kubernetes中創建測試pod和svc

```yaml
##busybox pod
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dnsbox-deployment-slave2
  labels:
    app: dnsbox
spec:
  replicas: 1
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: dnsbox
  template:
    metadata:
      labels:
        app: dnsbox
    spec:
      containers:
      - name: dnsbox
        image: gcr.io/kubernetes-e2e-test-images/dnsutils:1.3
        imagePullPolicy: IfNotPresent

        command: ['sh', '-c', 'echo Container 1 is Running ; sleep 3600']
```

```yaml
##nginx pod
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx3
  name: nginx3
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx3
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx3
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

配置nginx服務（clusterIP，暴露端口80）

```yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: nginx3
  name: nginx3
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx3
status:
  loadBalancer: {}
```

檢查創建情況

```
$ kubectl get pod -o wide
NAME                                  READY   STATUS    RESTARTS   AGE    IP
NODE         NOMINATED NODE   READINESS GATES
dnsbox-k8s-slave3-5f56984b58-hm5c4    1/1     Running   0          46h    10.233.
```

```
158.208   k8s-slave3   <none>           <none>
nginx3-5465449d99-j46zn               1/1     Running   1          2d2h   10.233.
158.205   k8s-slave3   <none>           <none>
$ kubectl get svc -o wide
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE    SELECTOR
kubernetes   ClusterIP   10.33.33.1    <none>        443/TCP   2d2h   <none>
nginx3       ClusterIP   10.33.33.53   <none>        80/TCP    2d2h   app=nginx3
```

部署wireguard服務器流程

1. 創建namespace: wireguard（可選）
2. 創建pv存儲（用來保存wireguard生成的客戶端配置文件）
3. 創建pvc存儲資源申請
4. 創建configmap（讓wireguard服務器讀取配置）
5. 創建wireguard pod服務器節點
6. 通過NodePort/Loadbalancer暴露VPN服務端口，用於客戶端進行呼叫鏈接（本文採用NodePort模式進行演示）

創建namespace

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: wireguard
  labels:
    name: wireguard
```

創建pv (3G硬盤空間，wireguard生成客戶端配置存儲路徑：掛載到宿主機ubuntu: /mnt/data)

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  namespace: wireguard
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 3Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

創建pvc

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-claim-wireguard
  namespace: wireguard
  labels:
    type: local
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

創建configmap

PUID為當前ubuntu用戶的id (非root用戶默認為1000)

peers: 告訴wireguard服務器生成多少個客戶端的配置，本實例為3個

peerdns為kubernetes的kube-dns地址

serverport為nodeport指定的服務暴露端口

allowedip為告訴終端哪些流量經過wireguard VPN封裝，如下配置為所有流量都指向到wireguard服務器)

檢查當前用戶的PUID&PGID

```
hitler@k8s-slave3:[~]:
$id
uid=1000(hitler) gid=1000(hitler) groups=1000(hitler),4(adm),24(cdrom),27(sudo),3
0(dip),46(plugdev),116(lxd),118(docker)
hitler@k8s-slave3:[~]:
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: wireguard-configmap
  namespace: wireguard
data:
  PUID: "1000"
  PGID: "1000"
  TZ: "CH/BJ"
  SERVERPORT: "31820"
  PEERS: "3"
```

```
  PEERDNS: "10.22.22.10"
  ALLOWEDIPS: "0.0.0.0/0, ::/0 "
  INTERNAL_SUBNET: "192.168.1.0"
```

創建wireguard pod, wireguard自身需要調用底層OS kernel Function，所以需要privilege=ture，capability: NET_ADMIN/SYS_MODULE

```
apiVersion: v1
kind: Pod
metadata:
  name: wireguard
  namespace: wireguard
  labels:
    app: wireguard
spec:
  containers:
  - name: wireguard
    image: ghcr.io/linuxserver/wireguard
    envFrom:
    - configMapRef:
        name: wireguard-configmap
    securityContext:
      capabilities:
        add:
          - NET_ADMIN
          - SYS_MODULE
      privileged: true
    volumeMounts:
      - name: wg-config
        mountPath: /config
      - name: host-volumes
        mountPath: /lib/modules
    ports:
    - containerPort: 51820
      protocol: UDP
  volumes:
    - name: wg-config
      persistentVolumeClaim:
        claimName: pv-claim-wireguard
    - name: host-volumes
      hostPath:
        path: /lib/modules
        type: Directory
```

創建wireguard的服務端口暴露（NodePort）模式，目的為讓客戶端能成功撥入，wireguard默認服務端口為51820，映射到nodeIP:31820（這也是客戶端撥入的地址與端口）

```
kind: Service
apiVersion: v1
metadata:
```

```
  labels:
    k8s-app: wireguard
  name: wireguard-service
  namespace: wireguard
spec:
  type: NodePort
  ports:
  - port: 51820
    nodePort: 31820
    protocol: UDP
    targetPort: 51820
  selector:
    app: wireguard
```

檢查wireguard狀態

```
mkdir wg
cd wg
ls
00-wg-ns.yaml  01.pv.yaml  02-wg-pvc.yaml  03-wg-configmap.yaml  04-wg-pod.yaml
 05-wg-svc-nordport.yaml
kubectl apply -f .
namespace/wireguard created
persistentvolume/task-pv-volume created
persistentvolumeclaim/pv-claim-wireguard created
configmap/wireguard-configmap created
pod/wireguard created
service/wireguard-service created
hitler@k8s-slave2:[~/wg]:
#
kubectl get svc -n wireguard
NAME                TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)           AGE
wireguard-service   NodePort   10.22.22.128   <none>        51820:31820/UDP   9s
hitler@k8s-slave2:[~/wg]:
#
hitler@k8s-slave2:[~/wg]:
kubectl get all -n wireguard
NAME            READY   STATUS    RESTARTS   AGE
pod/wireguard   1/1     Running   0          21s

NAME                        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
service/wireguard-service   NodePort   10.22.22.128   <none>        51820:31820/U
DP   21s
```

開啟wireguard pod中的net.ipv4.ip_forward=1，默認為0

```
kubectl exec -ti -n wireguard wireguard bash
sysctl net.ipv4.ip_forward=1
sysctl -p
sysctl --system
sysctl -a | grep net.ipv4.ip_forward
```

wireguard log檢查

```
kubectl -n wireguard logs wireguard
[s6-init] making user provided files available at /var/run/s6/etc...exited 0.
[s6-init] ensuring user provided files have correct perms...exited 0.
[fix-attrs.d] applying ownership & permissions fixes...
[fix-attrs.d] done.
[cont-init.d] executing container initialization scripts...
[cont-init.d] 01-envfile: executing...
[cont-init.d] 01-envfile: exited 0.
[cont-init.d] 10-adduser: executing...

-------------------------------------
          _         ()
         | |  ___   _    __
         | | / __| | |  /  \
         | | \__ \ | | | () |
         |_| |___/ |_|  \__/


Brought to you by linuxserver.io
-------------------------------------

To support the app dev(s) visit:
WireGuard: https://www.wireguard.com/donations/

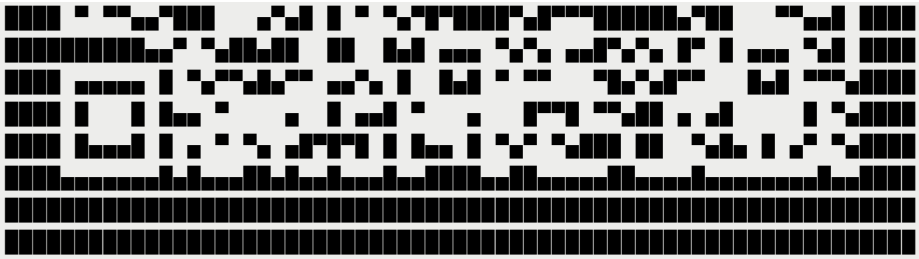
To support LSIO projects visit:
https://www.linuxserver.io/donate/
-------------------------------------
GID/UID
-------------------------------------

User uid:    1000
User gid:    1000
-------------------------------------

[cont-init.d] 10-adduser: exited 0.
[cont-init.d] 30-config: executing...
Uname info: Linux wireguard 5.6.0-050600-generic #202003292333 SMP Sun Mar 29 23:
35:58 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
**** It seems the wireguard module is already active. Skipping kernel header inst
all and module compilation. ****
**** Server mode is selected ****
**** SERVERURL var is either not set or is set to "auto", setting external IP to
 auto detected value of 45.117.99.230 ****
**** External server port is set to 31820. Make sure that port is properly forwar
ded to port 51820 inside this container ****
**** Internal subnet is set to 10.255.0.0 ****
**** AllowedIPs for peers 0.0.0.0/0, ::/0 ****
**** Peer DNS servers will be set to 10.22.22.10 ****
**** No wg0.conf found (maybe an initial install), generating 1 server and 2 pee
r/client confs ****
grep: /config/peer*/*.conf: No such file or directory
PEER 1 QR code:

```

```
PEER 2 QR code:
```



```
[cont-init.d] 30-config: exited 0.
[cont-init.d] 99-custom-scripts: executing...
[custom-init] no custom files found exiting...
[cont-init.d] 99-custom-scripts: exited 0.
[cont-init.d] done.
[services.d] starting services
[services.d] done.
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
.:53
CoreDNS-1.8.1
linux/amd64, go1.15.7, 95622f4
[#] ip -4 address add 10.255.0.1 dev wg0
[#] ip link set mtu 1360 up dev wg0
[#] ip -4 route add 10.255.0.3/32 dev wg0
[#] ip -4 route add 10.255.0.2/32 dev wg0
[#] iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; i
ptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
hitler@k8s-slave2:[~/wg]:
```

當wireguard運行完成後將自動生成客戶端連接所需要的配置文件，路徑為宿主host路徑/mnt/data/

```
hitler@k8s-slave3:/mnt/data$ ls
coredns  peer1  peer2  peer3  server  templates  wg0.conf
hitler@k8s-slave3:/mnt/data$
```

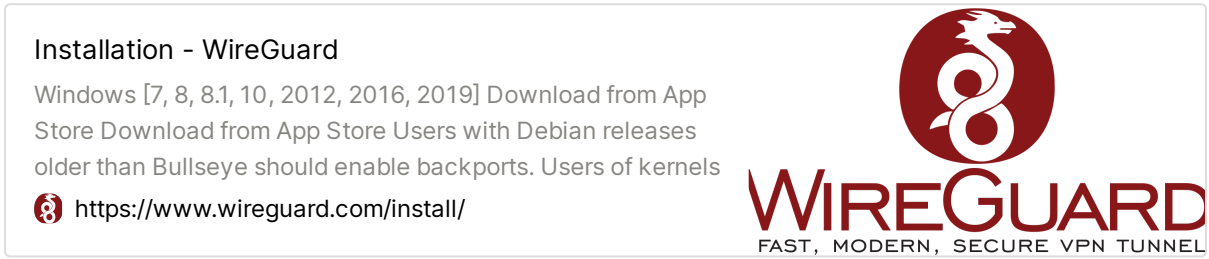
將peer配置文件拷貝到測試終端上，並且打開wireguard客戶端軟體導入配置

peer配置如下

```
$ cat peer1.conf
[Interface]
Address = 192.168.1.2
PrivateKey = oFriYU6NRukv5Wep3ljpKeiVT5M9ukOiPlI59tNXRVk=
ListenPort = 51820
DNS = 10.22.22.10

[Peer]
PublicKey = C6ut8ZYgiLHjA5WRDPtIc9rs215K8AyXvyJ7SNtlr1s=
Endpoint = 10.211.55.7:31820 （k8s通過nodeport方式暴露出的NodeIP:Port）
AllowedIPs = 10.255.0.0/16, 10.22.22.0/24 （VPN加密的流量，可以按需修改，10.255.0.0/16=
podIP，10.22.22.0=clusterIP）
```

Wireguard客戶端下載地址

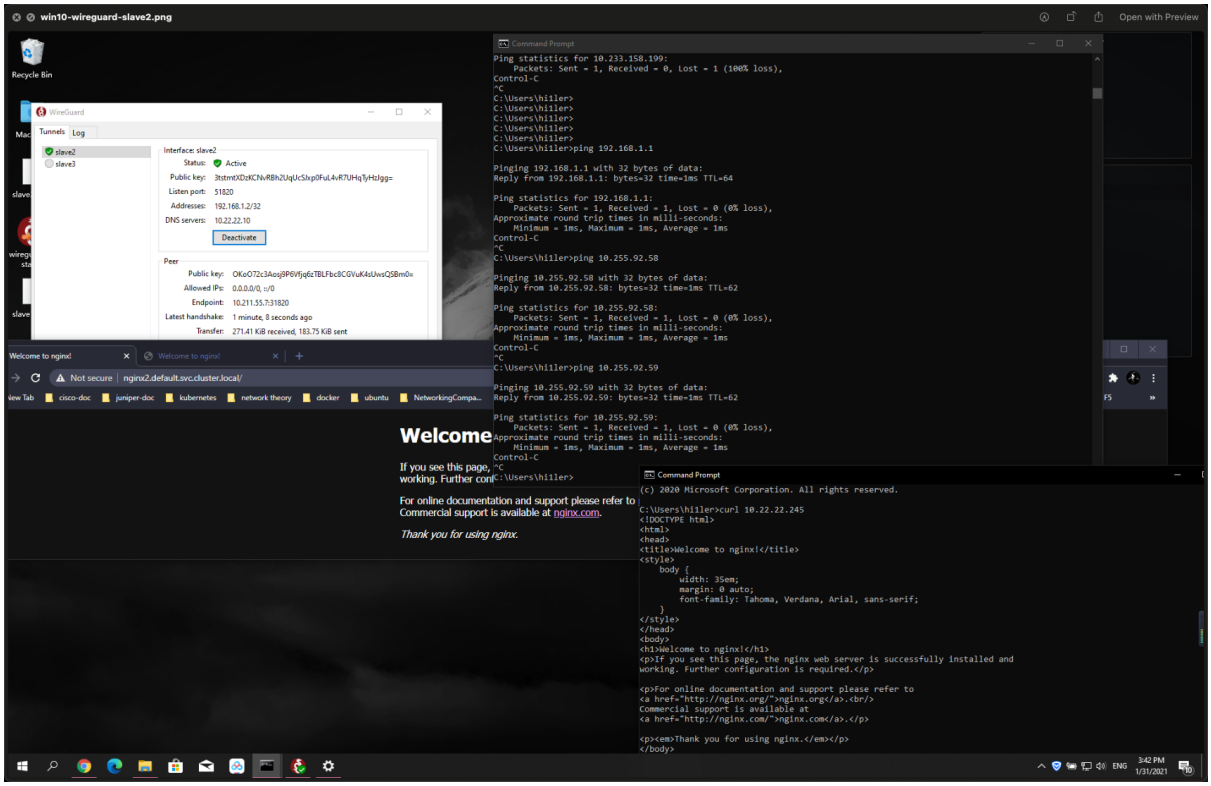


windows10測試截圖

ping podIP 測試通過

curl nginx svc測試通過

瀏覽器 http://nginx2.default.cluster.local (通過kubernetes內部的kube-dns服務查詢)



macos測試截圖

注意事項：macos bigsur使用wireguard GUI工作不了，許下載命令行bin執行wireguard連接

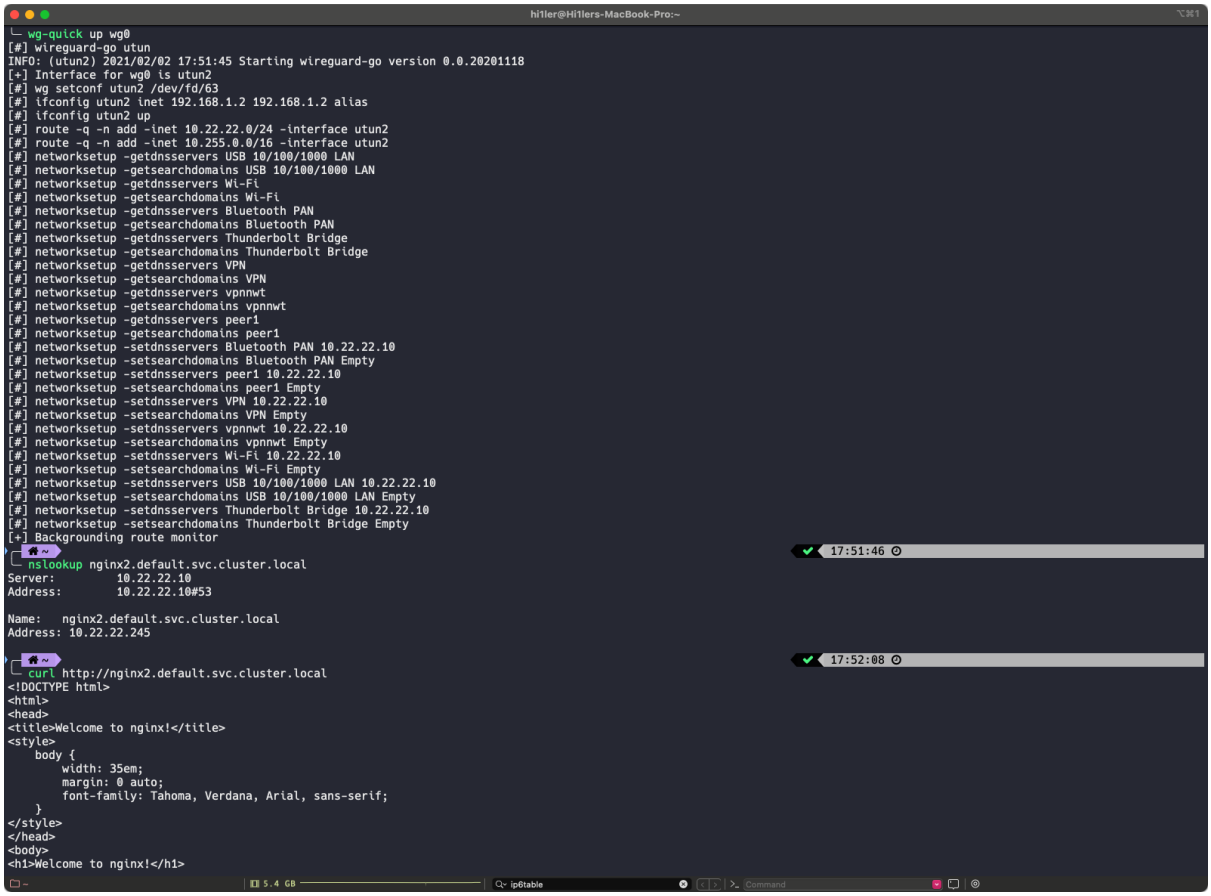
brew install wireguard-go

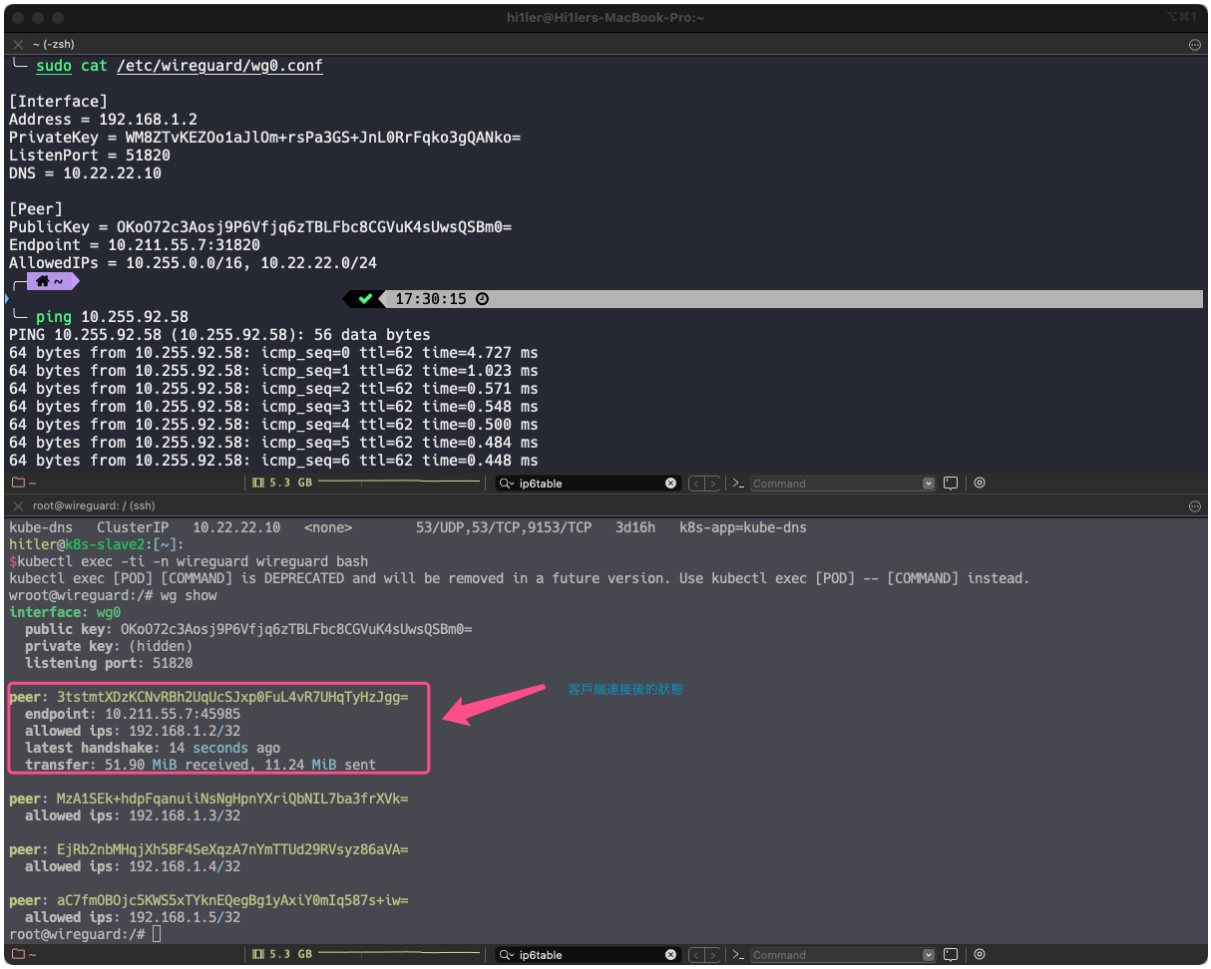
brew install wireguard-tools

創建/etc/wireguard/ 文件夾

將peer配置文件拷貝進來

執行wg-quick up wg0





ubuntu desktop測試日誌

```
##導入客戶端配置
sudo nmcli connection import type wireguard file peer1.conf
##連接
sudo nmcli connection up peer1
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkMan
ager/ActiveConnection/4)
hitler@k8s-slave1:/etc/netplan$ cd ..

$ ping 10.255.92.15 # podIP
PING 10.255.92.15 (10.255.92.15) 56(84) bytes of data.
64 bytes from 10.255.92.15: icmp_seq=1 ttl=62 time=13.0 ms
^C
--- 10.255.92.15 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 13.004/13.004/13.004/0.000 ms

$ curl <nginx-svc-ip>
<!DOCTYPE html>
<html>
<head>_
<title>Welcome to nginx!</title>
<style>
    body {
```

```
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
hitler@k8s-slave1:/etc/NetworkManager$
```

Multi-cluster kubernetes Pod、clusterIP透傳

需求： 多cluster之間的podIP/clusterIP互聯互通

注意：Multi-cluster內部的IP地址分配不要重疊

```
## kernel version check
```sh
#slave2
$uname -a
Linux k8s-slave2 5.6.0-050600-generic #202003292333 SMP Sun Mar 29 23:35:58 UTC 2
020 x86_64 x86_64 x86_64 GNU/Linux

#slave3
$ uname -a
Linux k8s-slave3 5.6.0-050600-generic #202003292333 SMP Sun Mar 29 23:35:58 UTC 2
020 x86_64 x86_64 x86_64 GNU/Linux
```
multi-master check
```sh
#slave2
$kubectl get nodes
NAME         STATUS   ROLES                  AGE   VERSION
k8s-slave2   Ready    control-plane,master   39h   v1.20.2

#slave3
$kubectl get nodes
NAME         STATUS   ROLES                  AGE    VERSION
k8s-slave3   Ready    control-plane,master   163m   v1.20.2
```
wireguard cli tools installation
```sh
```
```

```
sudo apt install -y wireguard
wg genkey | tee privatekey | wg pubkey > publickey
ip link add wg0 type wireguard
ip addr add 172.16.255.2/24 dev wg0
wg set wg0 private-key privatekey
ip link set wg0 up

wg genkey | tee privatekey | wg pubkey > publickey
ip link add wg0 type wireguard
ip addr add 172.16.255.3/24 dev wg0
wg set wg0 private-key privatekey
ip link set wg0 up


interface: wg0
  public key: E0s6lo/7yI5PCidPd08lhKyByN1VfG6CP6vC6cV+lis=
  private key: (hidden)
  listening port: 38401
root@k8s-slave2:[/etc/wireguard]:

interface: wg0
  public key: guKzxfkfoRwro53sWG7ajFyNNE4kyKz4BKdStvG+Hgg=
  private key: (hidden)
  listening port: 42799
root@k8s-slave3:/etc/wireguard#

#slave2 -gw set wg0 peer <peer-public-key>
wg set wg0 listen-port 51820
wg set wg0 peer guKzxfkfoRwro53sWG7ajFyNNE4kyKz4BKdStvG+Hgg= allowed-ips 172.16.2
55.3/32 endpoint 10.211.55.8:51820

#slave3
wg set wg0 listen-port 51820
wg set wg0 peer E0s6lo/7yI5PCidPd08lhKyByN1VfG6CP6vC6cV+lis= allowed-ips 172.16.2
55.2/32 endpoint 10.211.55.7:51820

## port-listen
root@k8s-slave2:[/etc/wireguard]:
ss -nlu
State           Recv-Q           Send-Q                     Local Address:Port
Peer Address:Port          Process
UNCONN          0                0                          127.0.0.53%lo:53
0.0.0.0:*
UNCONN          0                0                                0.0.0.0:31820
0.0.0.0:*
UNCONN          0                0                                0.0.0.0:51820
0.0.0.0:*
UNCONN          0                0                                   [::]:51820
[::]:*
root@k8s-slave2:[/etc/wireguard]:

root@k8s-slave3:/etc/wireguard# ss -nlu
State           Recv-Q           Send-Q                     Local Address:Port
Peer Address:Port          Process
UNCONN          0                0                          127.0.0.53%lo:53
0.0.0.0:*
UNCONN          0                0                                0.0.0.0:31820
0.0.0.0:*
```

```
UNCONN          0                0                                    0.0.0.0:51820
0.0.0.0:*
UNCONN          0                0                                       [::]:51820
[::]:*
root@k8s-slave3:/etc/wireguard#

## wg0-conf generation
cd /etc/wireguard/
wg showconf wg0 >> wg0.conf
#開啟wireguard
$wg-quick up wg0
#關閉wireguard
$wg-quick down wg0

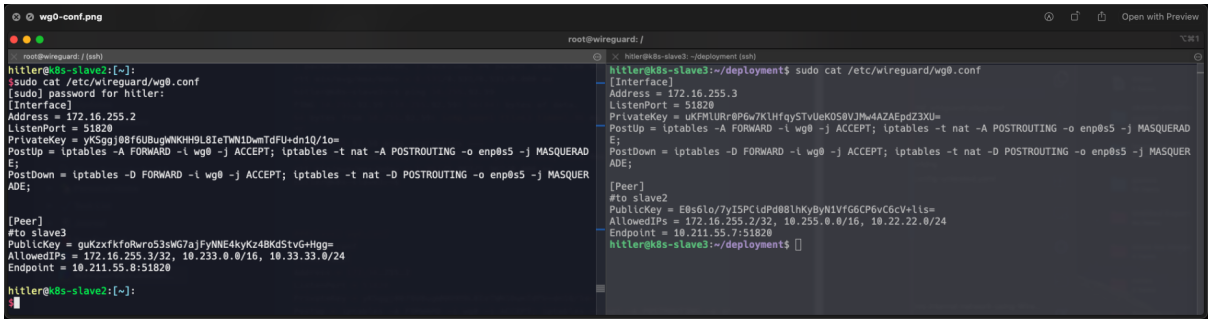
###slave2 wg0.conf
cat wg0.conf
[Interface]
Address = 172.16.255.2
ListenPort = 51820
PrivateKey = yKSggj08f6UBugWNKHH9L8IeTWN1DwmTdFU+dn1Q/1o=
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o
 enp0s5 -j MASQUERADE;
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -
o enp0s5 -j MASQUERADE;


[Peer]
#to slave3
PublicKey = guKzxfkfoRwro53sWG7ajFyNNE4kyKz4BKdStvG+Hgg=
AllowedIPs = 172.16.255.3/32, 10.233.0.0/16, 10.33.33.0/24
Endpoint = 10.211.55.8:51820

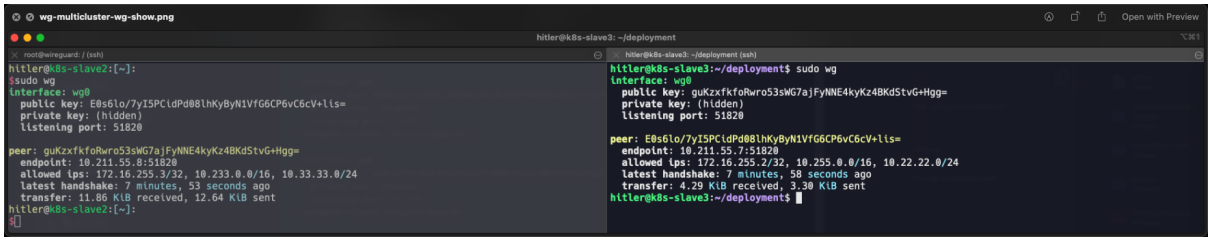
###slave3 wg0.conf
root@k8s-slave3:/etc/wireguard# cat wg0.conf
[Interface]
Address = 172.16.255.3
ListenPort = 51820
PrivateKey = uKFMlURr0P6w7KlHfqySTvUeKOS0VJMw4AZAEpdZ3XU=
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o
 enp0s5 -j MASQUERADE;
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -
o enp0s5 -j MASQUERADE;

[Peer]
#to slave2
PublicKey = E0s6lo/7yI5PCidPd08lhKyByN1VfG6CP6vC6cV+lis=
AllowedIPs = 172.16.255.2/32, 10.255.0.0/16, 10.22.22.0/24
Endpoint = 10.211.55.7:51820
```

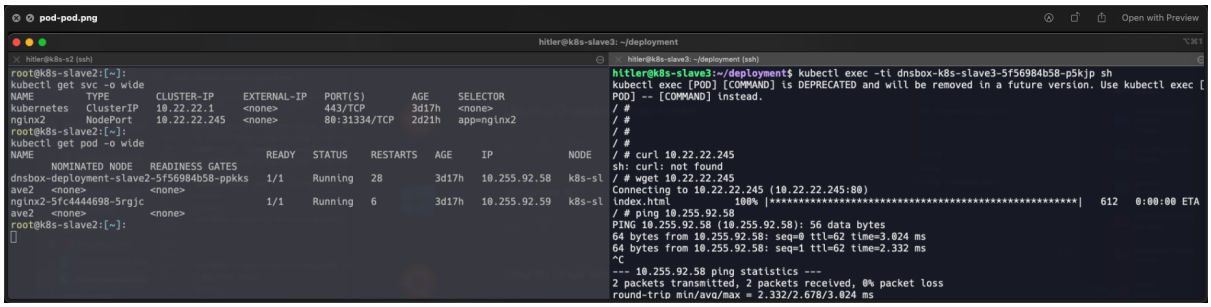
wg0.conf



wg show狀態檢查



multi-cluster pod之間連通性測試



Multi-cluster重叠地址互访

准备两个kubernetes环境，地址均为10.233.0.0/16，利用wireguard搭配iptables完成地址转换实现对接需求

overlapping subnet k8s

slave3 podIP 10.233.0.0/16 clusterip 10.33.33.0/24

mapping 192.33.0.0/16(dnat)

slave4 podIP 10.233.0.0/16 clusterip 10.33.33.0/24

mapping 192.44.0.0/16(dnat)

```
### slave3-添加iptables DNAT策略，访问到192.33.0.0翻译为真实的podIP地址段
iptables -t nat -A PREROUTING -d 192.33.0.0/16 -j NETMAP --to 10.233.0.0/16

$sudo cat /etc/wireguard/wg0.conf
[Interface]
Address = 172.16.255.3
ListenPort = 51820
PrivateKey = uKFMlURr0P6w7KlHfqySTvUeKOS0VJMw4AZAEpdZ3XU=
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o
 enp0s5 -j MASQUERADE;
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -
o enp0s5 -j MASQUERADE;

[Peer]
#to slave2
PublicKey = E0s6lo/7yI5PCidPd08lhKyByN1VfG6CP6vC6cV+lis=
AllowedIPs = 172.16.255.2/32, 10.255.0.0/16, 10.22.22.0/24
Endpoint = 10.211.55.7:51820

[Peer]
#to slave4
PublicKey = 9iq4e3AsBnxP3W+lU0bgY1qhjxqhfIqE1UAU2ysYiHo=
AllowedIPs = 172.16.255.4/32, 192.44.0.0/16
Endpoint = 10.211.55.9:51820
hitler@k8s-slave3:[~]:


### slave4-添加iptables DNAT策略，访问到192.44.0.0翻译为真实的podIP地址段
iptables -t nat -A PREROUTING -d 192.44.0.0/16 -j NETMAP --to 10.233.0.0/16

$sudo cat /etc/wireguard/wg0.conf
[Interface]
Address = 172.16.255.4
ListenPort = 51820
PrivateKey = YEqZrBPaZbhsvvTvdNNCIjZMFQ4tYPsi/AfKEfGwwWo=
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o
 enp0s5 -j MASQUERADE;
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -
o enp0s5 -j MASQUERADE;

[Peer]
#to slave3
PublicKey = guKzxfkfoRwro53sWG7ajFyNNE4kyKz4BKdStvG+Hgg=
AllowedIPs = 172.16.255.3/32, 192.33.0.0/16
Endpoint = 10.211.55.8:51820

## slave3/4 wg-quick up wg0

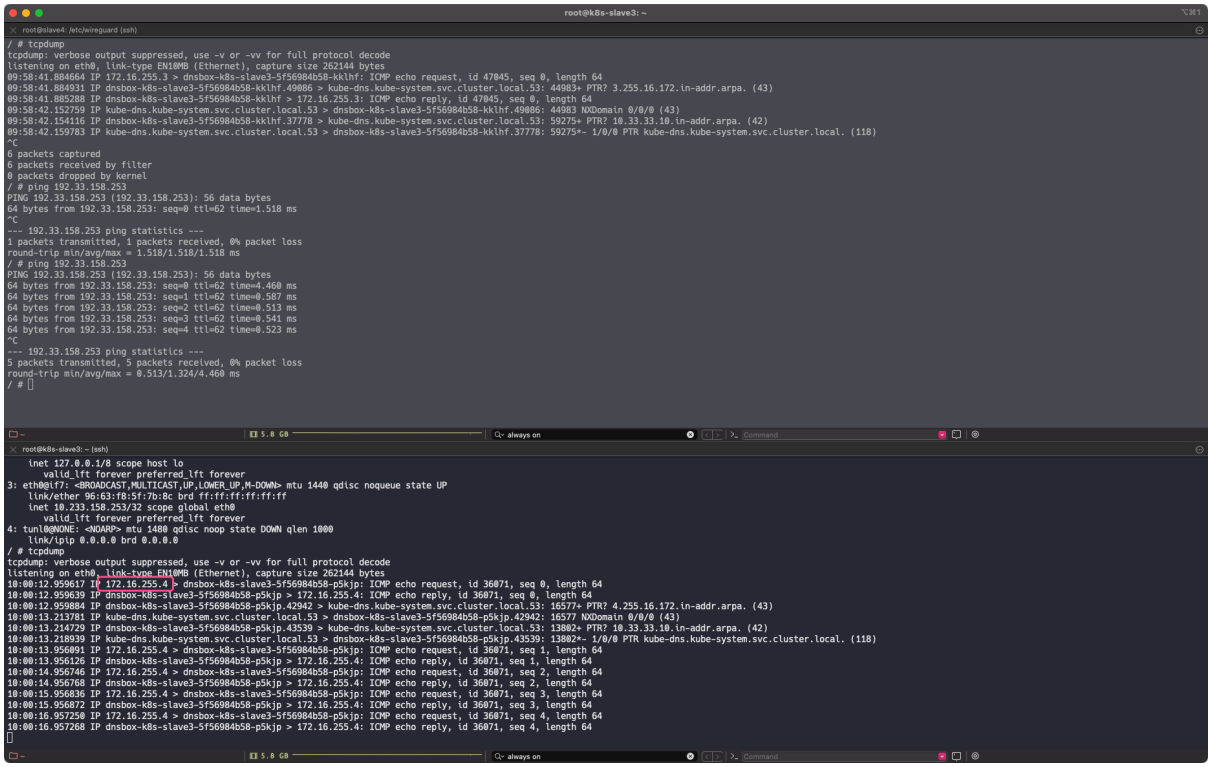
### 3ping4
hitler@k8s-slave3:[~]:
$kubectl exec -ti dnsbox-k8s-slave3-5f56984b58-p5kjp sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future versio
n. Use kubectl exec [POD] -- [COMMAND] instead.
/ #
/ #
/ # ping 192.44.206.198
```

```
PING 192.44.206.198 (192.44.206.198): 56 data bytes
64 bytes from 192.44.206.198: seq=0 ttl=62 time=1.588 ms
64 bytes from 192.44.206.198: seq=1 ttl=62 time=0.564 ms
^C
--- 192.44.206.198 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.564/1.076/1.588 ms
/ #
/ #
/ # ping 192.44.206.198
PING 192.44.206.198 (192.44.206.198): 56 data bytes
64 bytes from 192.44.206.198: seq=0 ttl=62 time=1.672 ms
64 bytes from 192.44.206.198: seq=1 ttl=62 time=0.553 ms
64 bytes from 192.44.206.198: seq=2 ttl=62 time=0.472 ms
64 bytes from 192.44.206.198: seq=3 ttl=62 time=0.776 ms
64 bytes from 192.44.206.198: seq=4 ttl=62 time=1.353 ms
^C
--- 192.44.206.198 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.472/0.965/1.672 ms
/ #

## 4ping3
hitler@slave4:[~/deployment]:
$kubectl exec -ti dnsbox-k8s-slave3-5f56984b58-kklhf sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future versio
n. Use kubectl exec [POD] -- [COMMAND] instead.
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN 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 qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
4: eth0@if21: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1440 qdisc noqueue sta
te UP
    link/ether 82:cb:4d:78:b3:0e brd ff:ff:ff:ff:ff:ff
    inet 10.233.206.196/32 scope global eth0
       valid_lft forever preferred_lft forever
/ # ping 192.33.158.253
PING 192.33.158.253 (192.33.158.253): 56 data bytes
64 bytes from 192.33.158.253: seq=0 ttl=62 time=3.564 ms
^C
--- 192.33.158.253 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 3.564/3.564/3.564 ms
/ #
## slave3 dnat rule
root@k8s-slave3:~# iptables -L -t nat | grep NETMAP
NETMAP     all  --  anywhere             192.33.0.0/16        to:10.233.0.0/16

## slave4 dnat rule
root@slave4:~# iptables -L -t nat | grep NETMAP
NETMAP     all  --  anywhere             192.44.0.0/16        to:10.233.0.0/16
```

测试截图


root@k8s-slave3: ~
root@slave4: /etc/wireguard (ssh)
/ # tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:58:41.884664 IP 172.16.255.3 > dnsbox-k8s-slave3-5f56984b58-kklhf: ICMP echo request, id 47045, seq 0, length 64
09:58:41.884931 IP dnsbox-k8s-slave3-5f56984b58-kklhf.49086 > kube-dns.kube-system.svc.cluster.local.53: 44983+ PTR? 3.255.16.172.in-addr.arpa. (43)
09:58:41.885288 IP dnsbox-k8s-slave3-5f56984b58-kklhf > 172.16.255.3: ICMP echo reply, id 47045, seq 0, length 64
09:58:42.152759 IP kube-dns.kube-system.svc.cluster.local.53 > dnsbox-k8s-slave3-5f56984b58-kklhf.49086: 44983 NXDomain 0/0/0 (43)
09:58:42.154116 IP dnsbox-k8s-slave3-5f56984b58-kklhf.37778 > kube-dns.kube-system.svc.cluster.local.53: 59275+ PTR? 10.33.33.10.in-addr.arpa. (42)
09:58:42.159783 IP kube-dns.kube-system.svc.cluster.local.53 > dnsbox-k8s-slave3-5f56984b58-kklhf.37778: 59275*- 1/0/0 PTR kube-dns.kube-system.svc.cluster.local. (118)
^C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
/ # ping 192.33.158.253
PING 192.33.158.253 (192.33.158.253): 56 data bytes
64 bytes from 192.33.158.253: seq=0 ttl=62 time=1.518 ms
^C
--- 192.33.158.253 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.518/1.518/1.518 ms
/ # ping 192.33.158.253
PING 192.33.158.253 (192.33.158.253): 56 data bytes
64 bytes from 192.33.158.253: seq=0 ttl=62 time=4.460 ms
64 bytes from 192.33.158.253: seq=1 ttl=62 time=0.587 ms
64 bytes from 192.33.158.253: seq=2 ttl=62 time=0.513 ms
64 bytes from 192.33.158.253: seq=3 ttl=62 time=0.541 ms
64 bytes from 192.33.158.253: seq=4 ttl=62 time=0.523 ms
^C
--- 192.33.158.253 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.513/1.324/4.460 ms
/ #
5.0 GB
always on
Command
root@k8s-slave3: ~ (ssh)
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
3: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1440 qdisc noqueue state UP
link/ether 96:63:f8:5f:7b:8c brd ff:ff:ff:ff:ff:ff
inet 10.233.158.253/32 scope global eth0
valid_lft forever preferred_lft forever
4: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN qlen 1000
link/ipip 0.0.0.0 brd 0.0.0.0
/ # tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:00:12.959617 IP 172.16.255.4 > dnsbox-k8s-slave3-5f56984b58-p5kjp: ICMP echo request, id 36071, seq 0, length 64
10:00:12.959639 IP dnsbox-k8s-slave3-5f56984b58-p5kjp > 172.16.255.4: ICMP echo reply, id 36071, seq 0, length 64
10:00:12.959884 IP dnsbox-k8s-slave3-5f56984b58-p5kjp.42942 > kube-dns.kube-system.svc.cluster.local.53: 16577+ PTR? 4.255.16.172.in-addr.arpa. (43)
10:00:13.213781 IP kube-dns.kube-system.svc.cluster.local.53 > dnsbox-k8s-slave3-5f56984b58-p5kjp.42942: 16577 NXDomain 0/0/0 (43)
10:00:13.214729 IP dnsbox-k8s-slave3-5f56984b58-p5kjp.43539 > kube-dns.kube-system.svc.cluster.local.53: 13802+ PTR? 10.33.33.10.in-addr.arpa. (42)
10:00:13.218939 IP kube-dns.kube-system.svc.cluster.local.53 > dnsbox-k8s-slave3-5f56984b58-p5kjp.43539: 13802*- 1/0/0 PTR kube-dns.kube-system.svc.cluster.local. (118)
10:00:13.956091 IP 172.16.255.4 > dnsbox-k8s-slave3-5f56984b58-p5kjp: ICMP echo request, id 36071, seq 1, length 64
10:00:13.956126 IP dnsbox-k8s-slave3-5f56984b58-p5kjp > 172.16.255.4: ICMP echo reply, id 36071, seq 1, length 64
10:00:14.956746 IP 172.16.255.4 > dnsbox-k8s-slave3-5f56984b58-p5kjp: ICMP echo request, id 36071, seq 2, length 64
10:00:14.956768 IP dnsbox-k8s-slave3-5f56984b58-p5kjp > 172.16.255.4: ICMP echo reply, id 36071, seq 2, length 64
10:00:15.956836 IP 172.16.255.4 > dnsbox-k8s-slave3-5f56984b58-p5kjp: ICMP echo request, id 36071, seq 3, length 64
10:00:15.956872 IP dnsbox-k8s-slave3-5f56984b58-p5kjp > 172.16.255.4: ICMP echo reply, id 36071, seq 3, length 64
10:00:16.957250 IP 172.16.255.4 > dnsbox-k8s-slave3-5f56984b58-p5kjp: ICMP echo request, id 36071, seq 4, length 64
10:00:16.957268 IP dnsbox-k8s-slave3-5f56984b58-p5kjp > 172.16.255.4: ICMP echo reply, id 36071, seq 4, length 64
5.0 GB
always on
Command

完結