

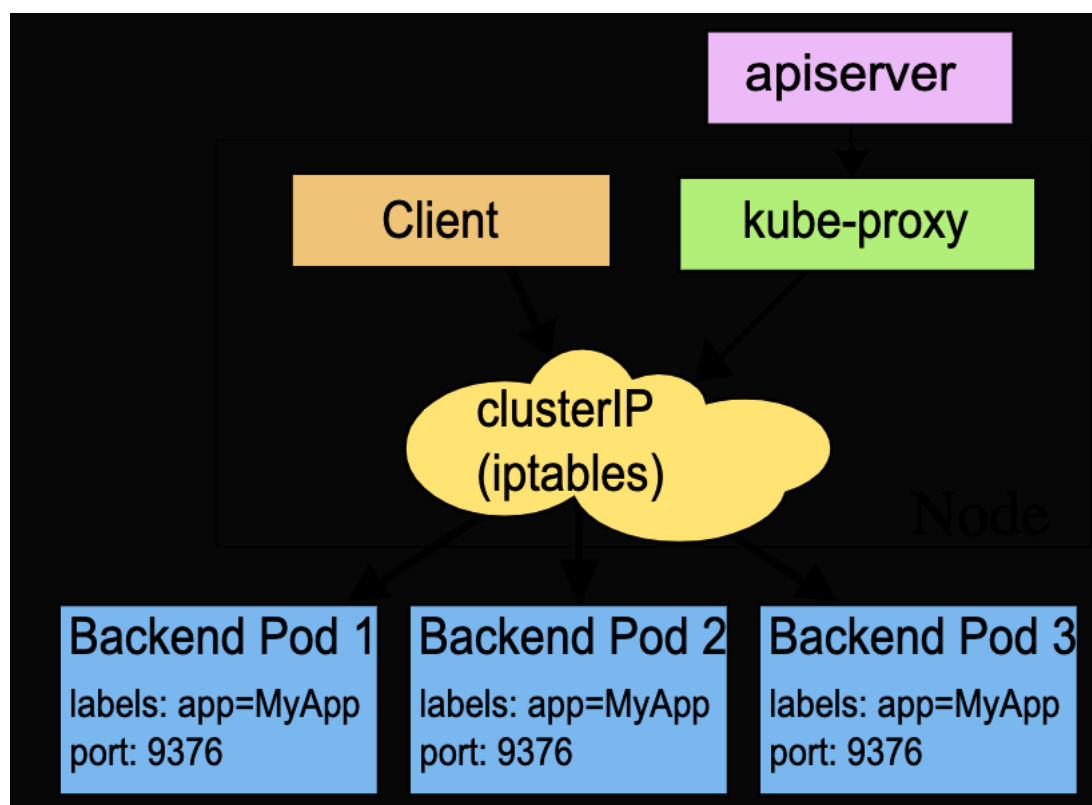


Wireguard在kubernetes中的應用

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

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

IP地址規劃

 節點名稱	 IP地址	 pod/cluster CIDR
<u>kubernetes cluster-1</u>	GW:10.211.55.1 Nic:10.211.55.7/24	pod: 10.255.0.0/16 clusterIP: 10.22.22.0/24
<u>kubernetes cluster-2</u>	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-5.6.0-050600_5.6.0-050600.202003292333_all.deb;

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

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.

wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.6/linux-modules-5.6.0-050600-generic_5.6.0-050600.202003292333_amd64.
#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"
swapoff -a;
sed -i 's/ swap / s/^/#/' /etc/fstab;
sleep 3

echo "apt update+install curl"
apt-get update && apt-get install -y apt-transport-https curl;
apt install -y gnupg2;
apt install -y jq;

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

```

```

echo "add source-list"

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.list.d/kubernetes.list

echo "docker installation"
apt-get update;
apt-get install -y docker.io;

cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
# enable docker& install 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）

```

##cluster1
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-
##cluster2
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-

```

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

```

kubectl apply -f https://docs.projectcalico.org/archive/v3.14/manifests/calico.yaml

```

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   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION
k8s-slave2          Ready     control-plane,master   3d15h   v1.20.2   10.211.55.7   <none>         Ubuntu 20.04.1 LTS   5.6.0-050600-gen
##cluster2
kubectl get node -o wide
NAME                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION
k8s-slave3          Ready     control-plane,master   2d2h   v1.20.2   10.211.55.8   <none>         Ubuntu 20.04.1 LTS   5.6.0-050600-gen
```

在kubernetes中創建測試pod和svc

```
##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']
```

```
##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）

```
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

```

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/mnt/data)

```

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)

```
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

```
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:
```



```

[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 GNU/Linux
**** It seems the wireguard module is already active. Skipping kernel header install 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 forwarded 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 peer/client configs ****
grep: /config/peer*/*.conf: No such file or directory
PEER 1 QR code:
[QR Code for Peer 1]
PEER 2 QR code:
[QR Code for Peer 2]
[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; iptables -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 = oFr1YU6NRukv5Wep3ljpKeiVT5M9uk0iP1I59tNXRVk=
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

Installation - WireGuard

Windows [7, 8, 8.1, 10, 2012, 2016, 2019] Download from App Store Download from App Store Users with Debian releases older than Bullseye should enable backports. Users of kernels < 5.6 may also choose wireguard-libs or wireguard-dkms+ linux-headers, depending on which kernel is used. The  <https://www.wireguard.com/install/>

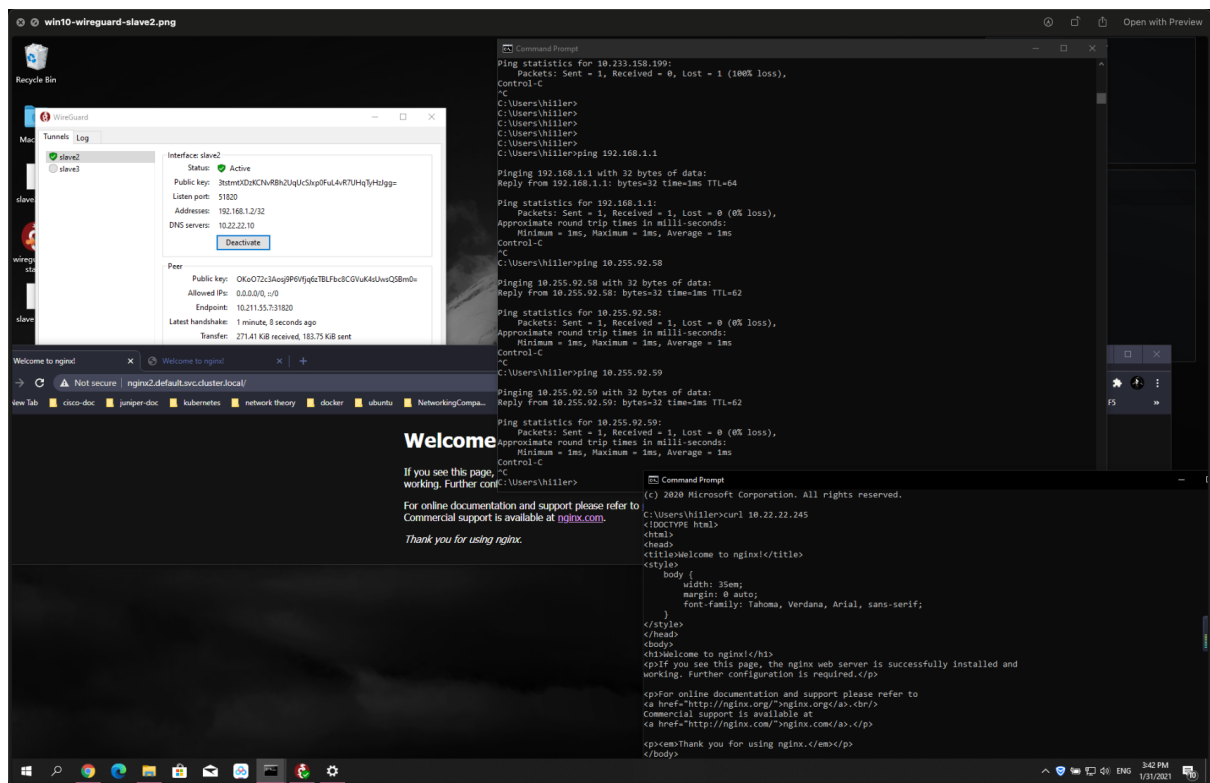


windows10測試截圖

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

```
hitler@Hitlers-MacBook-Pro:~  
└─ wg-quick up wg0  
[#] wireguard-go utun  
INFO: (utun2) 2021/02/02 17:51:45 Starting wireguard-go version 0.0.20201118  
[+] Interface for wg0 is utun2  
[#] wg setconf utun2 /dev/fd/63  
[#] ifconfig utun2 inet 192.168.1.2 192.168.1.2 alias  
[#] ifconfig utun2 up  
[#] route -q -n add -inet 10.22.22.0/24 -interface utun2  
[#] route -q -n add -inet 10.255.0.0/16 -interface utun2  
[#] networksetup -getdnsservers USB 10/100/1000 LAN  
[#] networksetup -getsearchdomains USB 10/100/1000 LAN  
[#] networksetup -getdnsservers Wi-Fi  
[#] networksetup -getsearchdomains Wi-Fi  
[#] networksetup -getdnsservers Bluetooth PAN  
[#] networksetup -getsearchdomains Bluetooth PAN  
[#] networksetup -getdnsservers Thunderbolt Bridge  
[#] networksetup -getsearchdomains Thunderbolt Bridge  
[#] networksetup -getdnsservers VPN  
[#] networksetup -getsearchdomains VPN  
[#] networksetup -getdnsservers vpnwt  
[#] networksetup -getsearchdomains vpnwt  
[#] networksetup -getdnsservers peer1  
[#] networksetup -getsearchdomains peer1  
[#] networksetup -setdnsservers Bluetooth PAN 10.22.22.10  
[#] networksetup -setsearchdomains Bluetooth PAN Empty  
[#] networksetup -setdnsservers peer1 10.22.22.10  
[#] networksetup -setsearchdomains peer1 Empty  
[#] networksetup -setdnsservers VPN 10.22.22.10  
[#] networksetup -setsearchdomains VPN Empty  
[#] networksetup -setdnsservers vpnwt 10.22.22.10  
[#] networksetup -setsearchdomains vpnwt Empty  
[#] networksetup -setdnsservers Wi-Fi 10.22.22.10  
[#] networksetup -setsearchdomains Wi-Fi Empty  
[#] networksetup -setdnsservers USB 10/100/1000 LAN 10.22.22.10  
[#] networksetup -setsearchdomains USB 10/100/1000 LAN Empty  
[#] networksetup -setdnsservers Thunderbolt Bridge 10.22.22.10  
[#] networksetup -setsearchdomains Thunderbolt Bridge Empty  
[+] Backgrounding route monitor  
└─ nslookup nginx2.default.svc.cluster.local  
Server:      10.22.22.10  
Address:     10.22.22.10#53  
  
Name:   nginx2.default.svc.cluster.local  
Address: 10.22.22.245  
  
└─ curl http://nginx2.default.svc.cluster.local  
<!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>
```

```
h1tler@H1tlers-MacBook-Pro:~  
~ (-zsh)  
sudo cat /etc/wireguard/wg0.conf  
[Interface]  
Address = 192.168.1.2  
PrivateKey = WM8ZTVKEZ0o1aJl0m+rsPa3GS+JnL0RrFqko3gQANko=  
ListenPort = 51820  
DNS = 10.22.22.10  
[Peer]  
PublicKey = 0Ko072c3Aosj9P6Vfjq6zTBLFbc8CGVuK4sUwsQSBm0=  
Endpoint = 10.211.55.7:31820  
AllowedIPs = 10.255.0.0/16, 10.22.22.0/24  
~  
ping 10.255.92.58  
PING 10.255.92.58 (10.255.92.58): 56 data bytes  
64 bytes from 10.255.92.58: icmp_seq=0 ttl=62 time=4.727 ms  
64 bytes from 10.255.92.58: icmp_seq=1 ttl=62 time=1.023 ms  
64 bytes from 10.255.92.58: icmp_seq=2 ttl=62 time=0.571 ms  
64 bytes from 10.255.92.58: icmp_seq=3 ttl=62 time=0.548 ms  
64 bytes from 10.255.92.58: icmp_seq=4 ttl=62 time=0.500 ms  
64 bytes from 10.255.92.58: icmp_seq=5 ttl=62 time=0.484 ms  
64 bytes from 10.255.92.58: icmp_seq=6 ttl=62 time=0.448 ms  
~  
root@wireguard: / (ssh)  
kube-dns ClusterIP 10.22.22.10 <none> 53/UDP,53/TCP,9153/TCP 3d16h k8s-app=kube-dns  
h1tler@k8s-slave2: [~]:  
$ kubectl exec -ti -n wireguard wireguard bash  
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.  
wroot@wireguard:/# wg show  
interface: wg0  
public key: 0Ko072c3Aosj9P6Vfjq6zTBLFbc8CGVuK4sUwsQSBm0=  
private key: (hidden)  
listening port: 51820  
peer: 3tstmtXDzKCNvRBh2UqUcSJxp0FuL4vR7UHqTyHzJgg=  
endpoint: 10.211.55.7:45985  
allowed ips: 192.168.1.2/32  
latest handshake: 14 seconds ago  
transfer: 51.90 MiB received, 11.24 MiB sent  
peer: MzA1SEk+hdPqanuiiNsNgHpnYXriQbN1L7ba3frXVvk=  
allowed ips: 192.168.1.3/32  
peer: EjRb2nbMHqjXh5BF4SeXqzA7nYmTTUd29Rvsyz86aVA=  
allowed ips: 192.168.1.4/32  
peer: aC7fm0B0jc5KwS5xTYknEQegBg1yAxiY0mIq587s+iw=  
allowed ips: 192.168.1.5/32  
root@wireguard:/#
```

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/NetworkManager/ActiveConnection/4)  
h1tler@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 2020 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 2020 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: guKzxkf0Rwro53sWG7ajFyNNE4kyKz4BKdStvG+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 guKzxkf0Rwro53sWG7ajFyNNE4kyKz4BKdStvG+Hgg= allowed-ips 172.16.255.3/32 endpoint 10.211.55.8:51820

#slave3
wg set wg0 listen-port 51820
wg set wg0 peer E0s6lo/7yI5PCidPd08lhKyByN1VfG6CP6vC6cV+lis= allowed-ips 172.16.255.2/32 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
UNCONN 0 0 127.0.0.53:53 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
```

```

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;

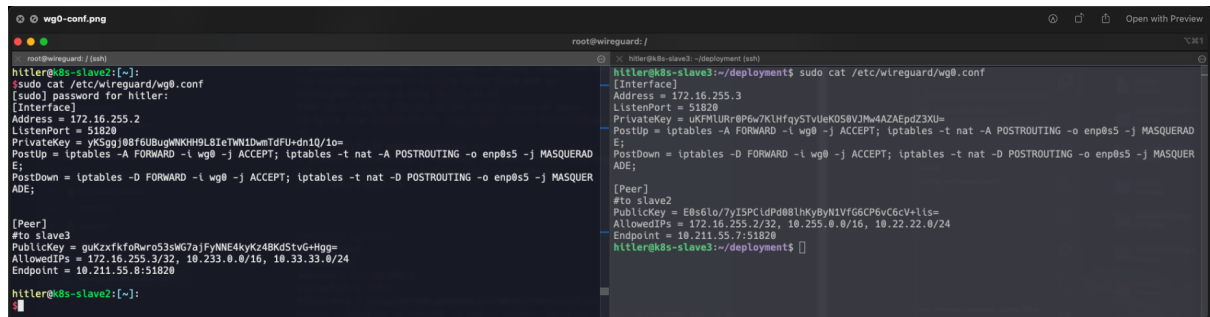
[Peer]
#to slave3
PublicKey = guKzxkf0Rwro53sWG7ajFyNNE4kyKz4BKdStvG+Hgg=
AllowedIPs = 172.16.255.3/32, 10.233.0.0/16, 10.33.33.0/24
Endpoint = 10.211.55.8:51820

###slave3 wg0.conf
root@k8s-slave3:/etc/wireguard# cat wg0.conf
[Interface]
Address = 172.16.255.3
ListenPort = 51820
PrivateKey = uKFMlURr0P6w7KlHfqyStvUeK0S0VJMw4AZAEpdZ3XU=
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;

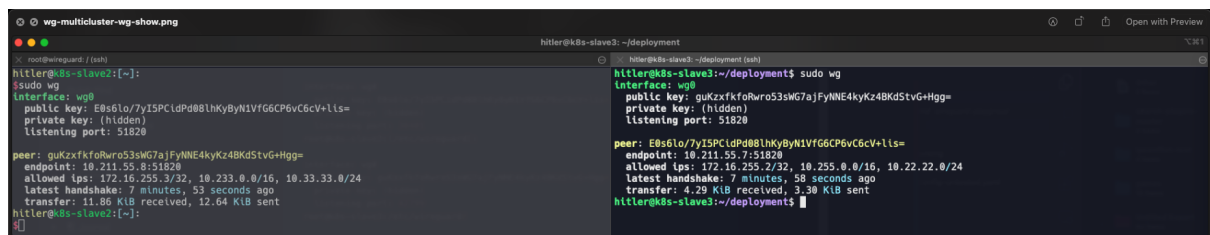
[Peer]
#to slave2
PublicKey = E0s6lo/7yI5PCldPd08lhKyByN1VfG6CP6vC6cV+lis=
AllowedIPs = 172.16.255.2/32, 10.255.0.0/16, 10.22.22.0/24
Endpoint = 10.211.55.7:51820

```

## wg0.conf



## wg show狀態檢查



## multi-cluster pod之間連通性測試

```
pod-pod.png
hitler@k8s-slave3: ~/deployment
root@k8s-slave2:[~]:
kubectll get svc -o wide
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR
kubernetes ClusterIP 10.22.22.1 <none> 443/TCP 3d17h <none>
nginx2 NodePort 10.22.22.245 <none> 80:31334/TCP 2d21h app=nginx2
root@k8s-slave2:[~]:
kubectll get pod -o wide
NAME Nominated Node Readiness Gates READY STATUS RESTARTS AGE IP NODE
dnsbox-deployment-slave2-5f56984b58-ppkks 1/1 Running 28 3d17h 10.255.92.58 k8s-sl
ave2 <none>
nginx2-5fc4444b98-5rgjc 1/1 Running 6 3d17h 10.255.92.59 k8s-sl
ave2 <none>
root@k8s-slave2:[~]:
[]

hitler@k8s-slave3:~/deployment$ kubectll exec -tl dnsbox-k8s-slave3-5f56984b58-pskjp sh
kubectll exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectll exec [
POD] -- [COMMAND] instead.
/ #
/ #
/ #
/ #
/ # curl 10.22.22.245
sh: curl: not found
/ # wget 10.22.22.245
Connecting to 10.22.22.245 (10.22.22.245:80)
index.html 100% |*****| 612 0:00:00 ETA
/ # ping 10.255.92.58
PING 10.255.92.58 (10.255.92.58): 56 data bytes
64 bytes from 10.255.92.58: seq=0 ttl=62 time=3.024 ms
64 bytes from 10.255.92.58: seq=1 ttl=62 time=2.332 ms
^C
--- 10.255.92.58 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 2.332/2.678/3.024 ms
```

完結