GOTC

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE , OPEN WORLD

「开源云原生计算时代论坛」专场

实现基于 Macvlan 的高性能容器网络

张智博 2021年08月1日

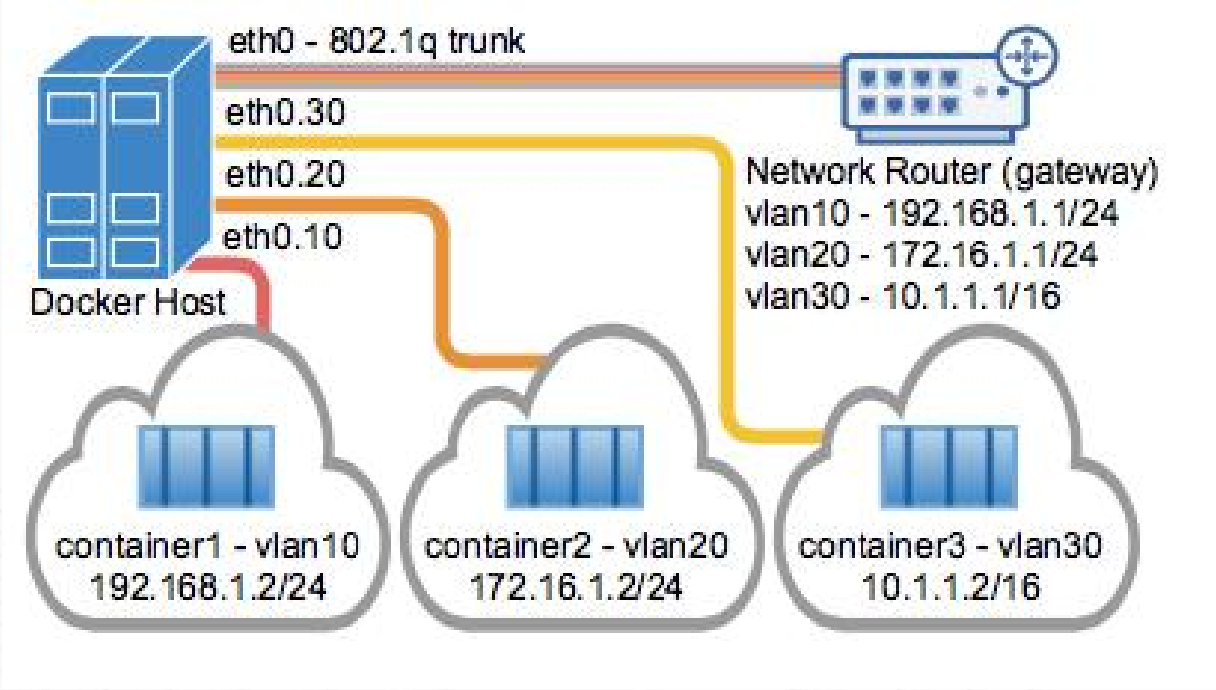
About Me

GOTC

- 张智博（牛小腩），深耕云计算多年，Openstack---->Docker---->Kubernetes
- Rancher -----> SUSE
  - Rancher中国研发总监
  - SUSE中国研发总监
- 长期聚焦在Linux+Container开源技术栈
  - Rancher 1.x ----> Rancher 2.x
  - RancherOS ----> K3OS ----> SUSE China Linux
- Base在沈阳

容器网络千千万，为何钟情Macvlan?

纯粹的技术视角，Macvlan简单到爆炸

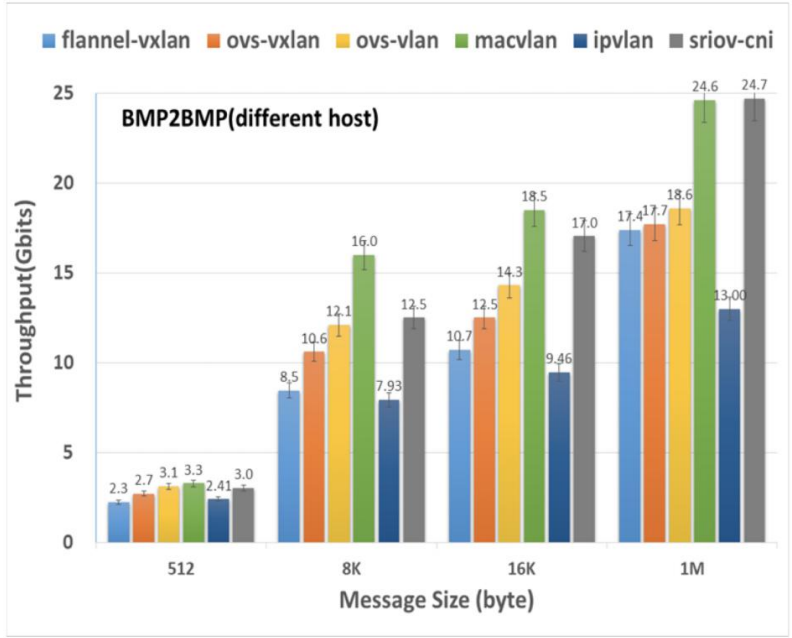


1. Docker时代就已“深入人心”
2. 对内核几乎没有特殊要求，主流OS版本均默认支持
3. 无需复杂协议配置或者高端硬件需求，交换机二层网络划分即可
4. 运维人员容易管理，容器IP直达

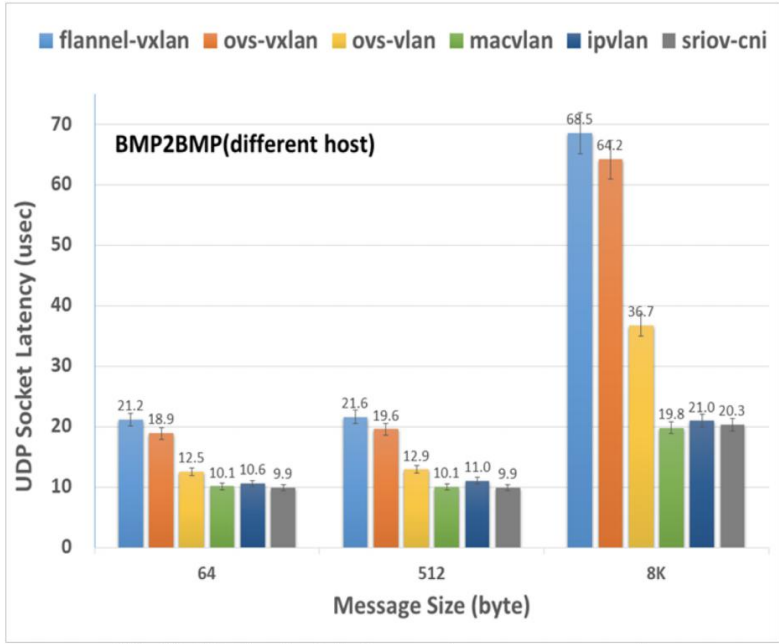
Kubernetes中使用Macvlan给用户带来什么?

GOTC

1. 最大程度兼容虚拟化向容器迁移时，一些旧有的使用习惯
2. 较少的CPU消耗，极致的网络性能
3. 较小的管理代价，简单易懂

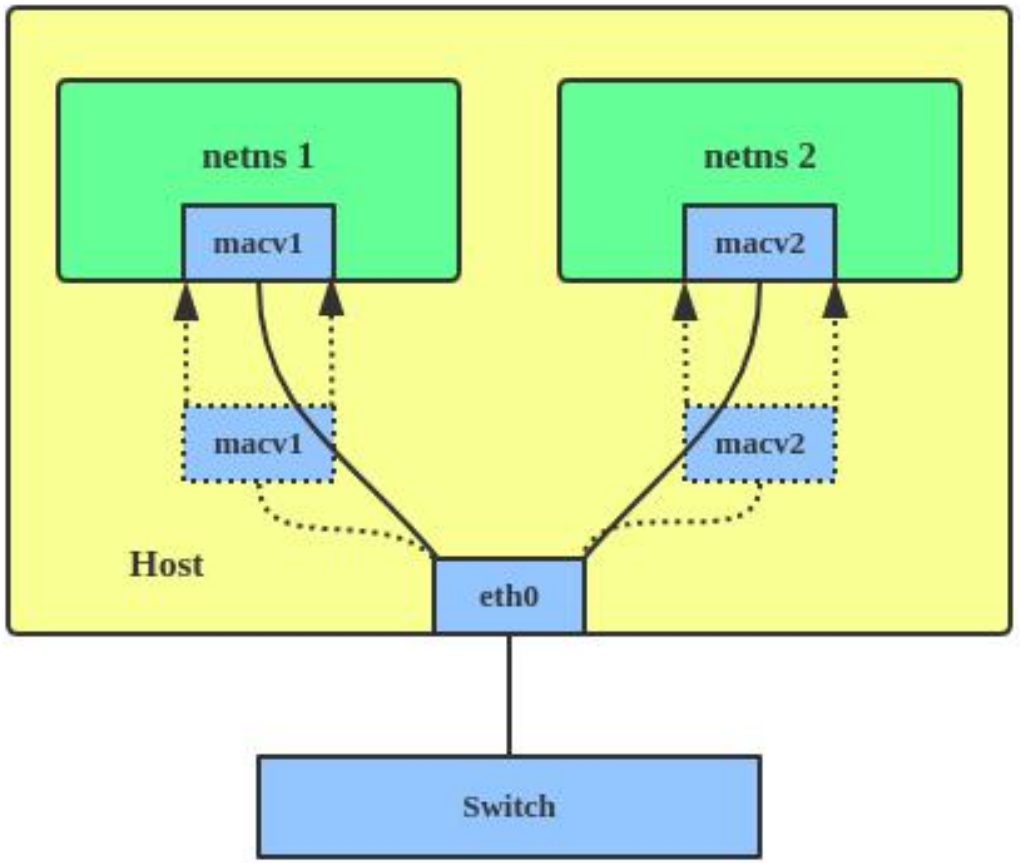


(a) Network traffic throughput of TCP by message size



(b) UDP Socket latency by message size

世上仅存不多的，尚能手动“操作”的网络插件




```
# add the namespaces
ip netns add ns1
ip netns add ns2

# create the macvlan link attaching it to the parent host eno1
ip link add mv1 link eno1 type macvlan mode bridge
ip link add mv2 link eno1 type macvlan mode bridge

# move the new interface mv1/mv2 to the new namespace
ip link set mv1 netns ns1
ip link set mv2 netns ns2

# bring the two interfaces up
ip netns exec ns1 ip link set dev mv1 up
ip netns exec ns2 ip link set dev mv2 up

# set ip addresses
ip netns exec ns1 ifconfig mv1 192.168.1.50/24 up
ip netns exec ns2 ifconfig mv2 192.168.1.60/24 up

# show interface detail
ip netns exec ns1 ip a
ip netns exec ns2 ip a

# ping from one ns to another
ip netns exec ns1 ping -c 4 192.168.1.60

# cleanup
ip netns del ns1
ip netns del ns2
```

欣赏Ta的优点，也要包容Ta的缺点
（面向Kubernetes生产环境）

与Kubernetes期望的网络模型并不兼容

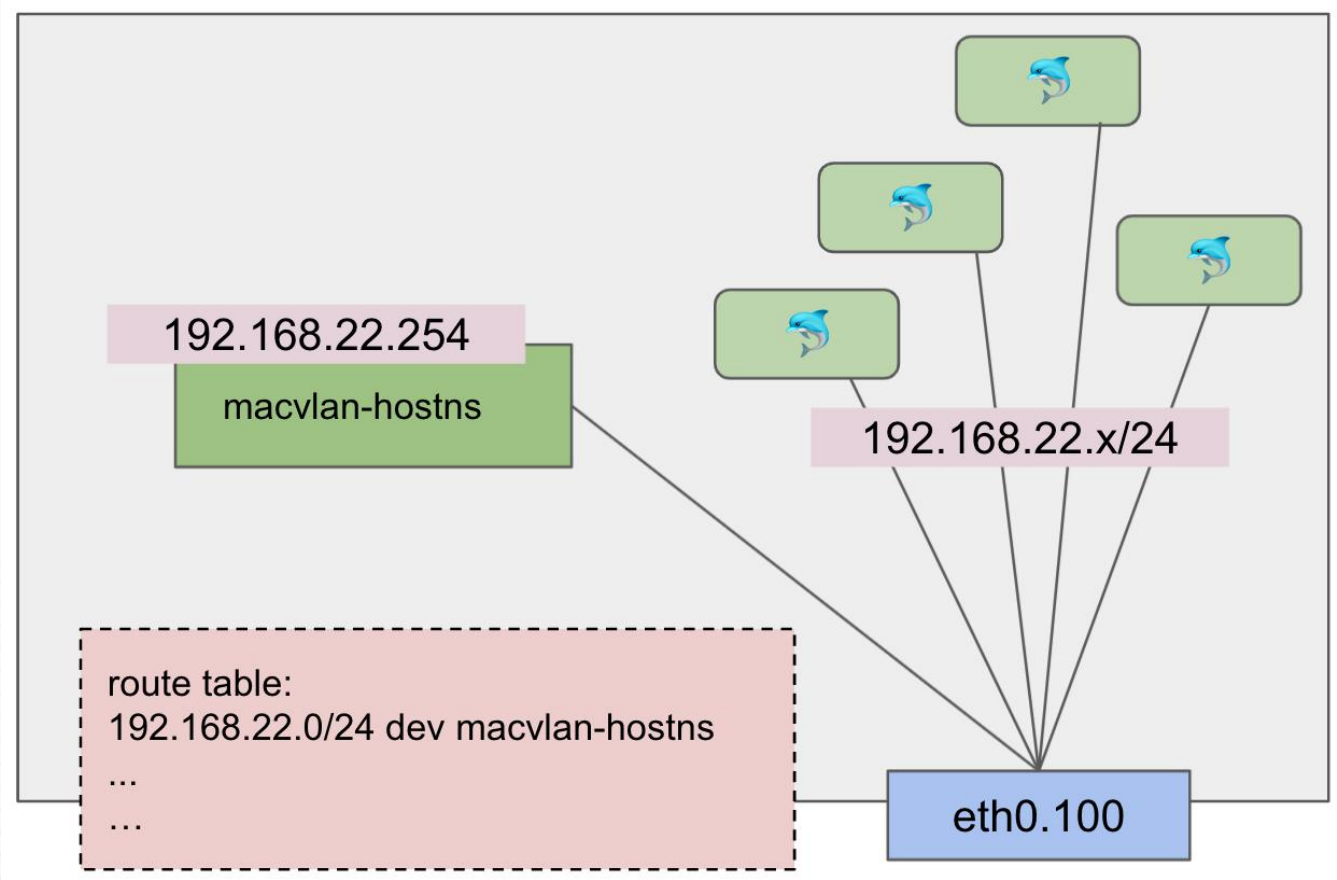
Kubernetes网络模型

- Container-to-container
- Pod-to-pod
- Pod-to-service
- Internet-to-service
  - Loadbalancer
  - Nodeport

Macvlan使用限制的影响

- 默认情况下，本机无法访问本地macvlan IP
  - Service Cluster IP受限
  - Pod healthcheck受限
- IP可直达访问
  - Nodeport/Loadbalancer模式不兼容
  - 保持IP可固定

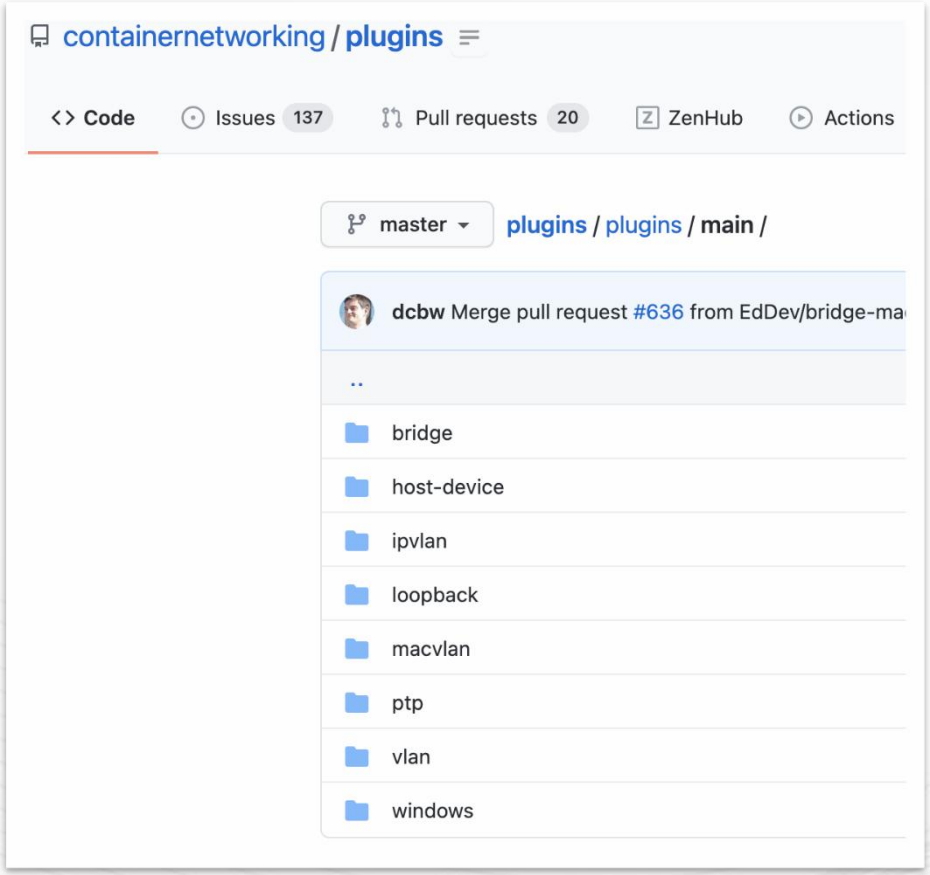
打通本机与本地容器的macvlan网络

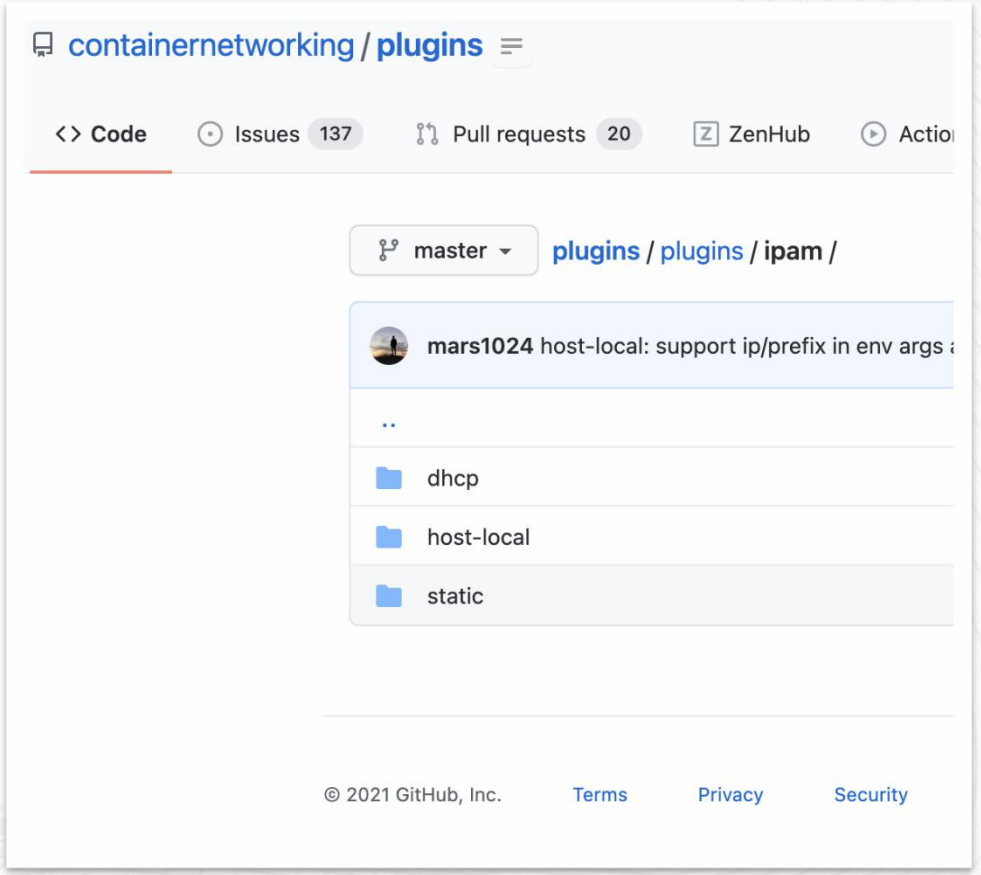


1. 在hostns中创建同网段的macvlan设备
2. 通过hostns路由方式转发访问
3. Kubernetes集群host较多时，需要保留大量IP，比较浪费IP资源

从实践经验看，这种模式仅适合小集群，面向通用产品化反而增加管理负担，容易出错。

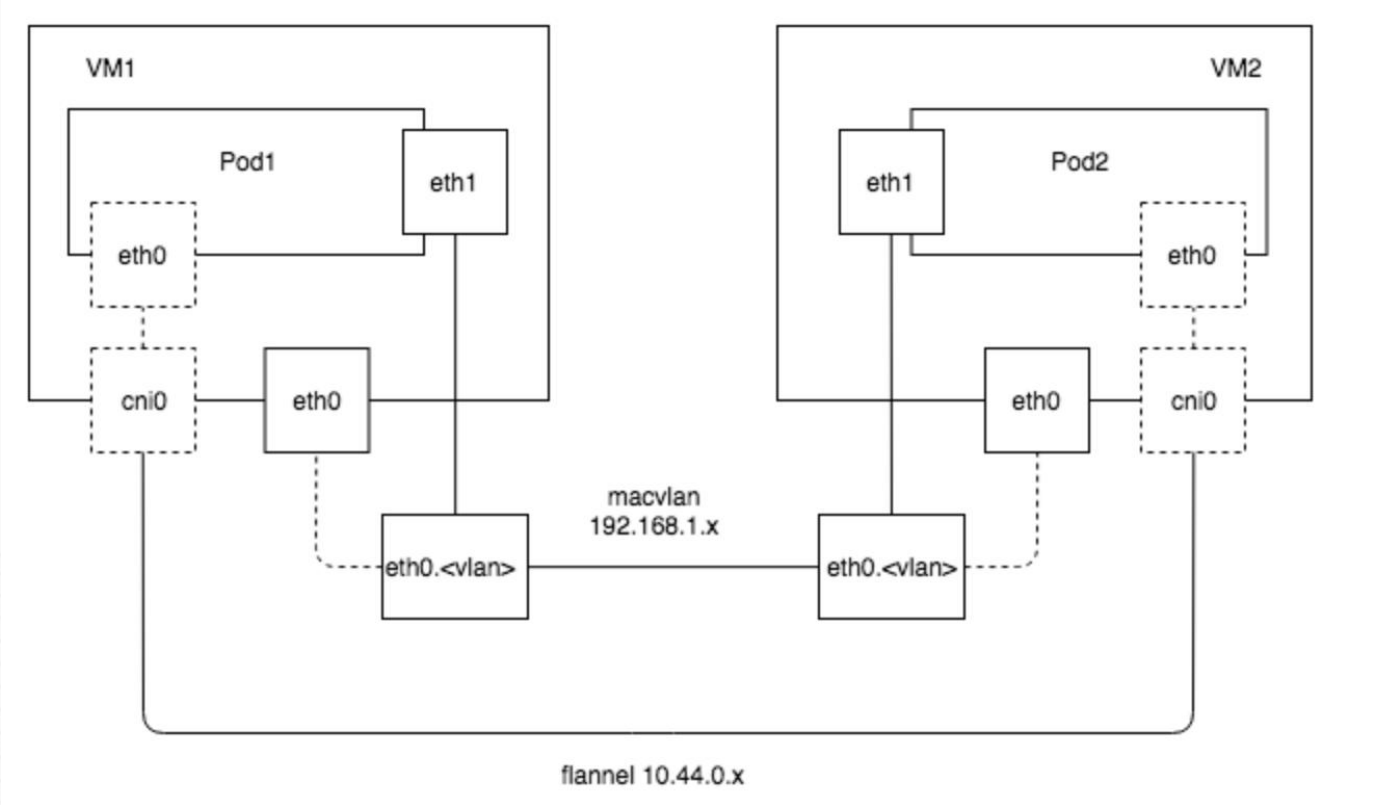
开源社区原生的Macvlan CNI功能较弱





产品化解决之道

使用Multus CNI实现双网卡机制



1. 基于Multus CNI可以支持多CNI的原理
2. 主网卡使用与Kubernetes兼容性好的CNI插件，如Flannel/Canal/等
3. 扩展网卡使用Macvlan，使用Controller&CRD控制Pod IP分配
4. 容器网络既能满足Kubernetes网络模型需求，又能兼具高性能与IP直达能力

使用CRD来持久化IP与Subnet数据

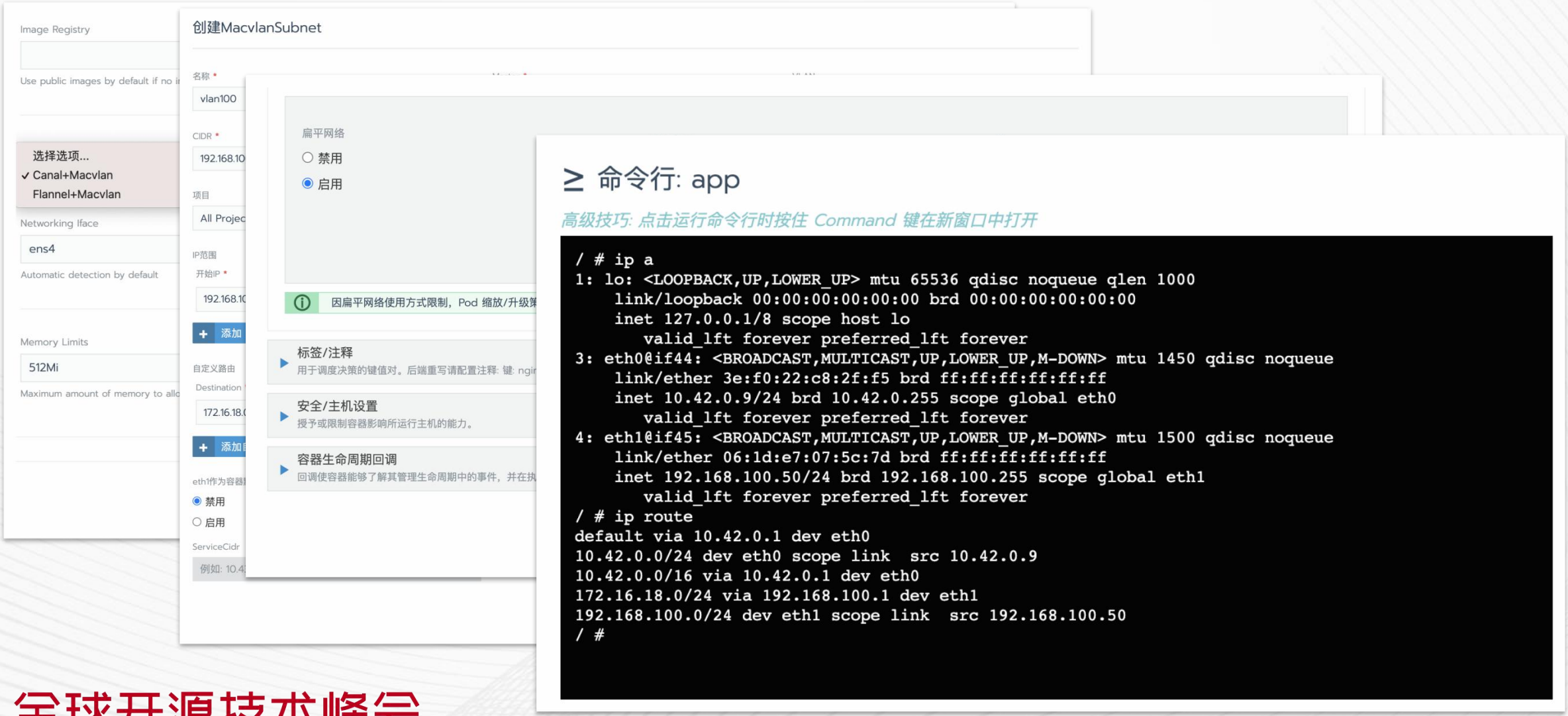
```
>
> kubectl get macvlanips -A
NAMESPACE   NAME                      SUBNET    PODID                                  CIDR                MAC
default     app-577bd97656-596lw      vlan100   fdab9c7b-e094-42e4-8d42-245392725bdc   192.168.100.50/24   06:1d:e7:07:5c:7d
default     app2-765cbf568d-7mhh4     vlan100   93cd6a2e-26aa-4fda-81cd-6bff3089891c   192.168.100.10/24   e6:56:1c:96:94:09
default     app2-765cbf568d-mhh2j     vlan100   f9e0bbe5-0fa7-42a8-9f58-fe142115edb3   192.168.100.11/24   1a:7a:15:0d:7e:18
default     app3-74d7dcdbdc-hlzvn     vlan200   98293a73-929c-434f-9d21-08fb8cdf47c1   192.168.200.2/24    2a:25:83:87:bb:b0
default     app4-54794f4f46-pt9nm     vlan10    919b3359-9dc7-4870-84c4-b07848073ead   192.168.10.2/24     de:55:a1:53:2e:80
default     app4-54794f4f46-tjlpf     vlan10    9948cc32-1071-4866-ab89-82868b69b26b   192.168.10.3/24     4e:00:75:94:ee:ef
>
>
> kubectl get macvlansubnets -A
NAMESPACE     NAME      MASTER   VLAN   CIDR              MODE     GATEWAY
kube-system   vlan10    ens4     10     192.168.10.0/24   bridge   192.168.10.1
kube-system   vlan100   ens4     100    192.168.100.0/24  bridge   192.168.100.1
kube-system   vlan200   ens4     200    192.168.200.0/24  bridge   192.168.200.1
>
>
> kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
app-577bd97656-596lw      1/1     Running   0          8m8s
app2-765cbf568d-7mhh4     1/1     Running   0          87s
app2-765cbf568d-mhh2j     1/1     Running   0          87s
app3-74d7dcdbdc-hlzvn     1/1     Running   0          71s
app4-54794f4f46-pt9nm     1/1     Running   0          57s
app4-54794f4f46-tjlpf     1/1     Running   0          56s
>
```

为什么自己管理子网和IP分配?

DHCP方式依赖交换机设置，很多用户实际使用时并不灵活。
通过Controller&CRD管理更加方便，比如可调整子网的IP range、控制IP释放，固定IP等。

简洁部署 简化使用

GOTC



重点问题：Pod的服务发现

GOTC

```
> kubectl get deploy app2
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
app2    2/2     2            2           21m
>
> kubectl get svc app2-macvlan
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
app2-macvlan   ClusterIP   None         <none>        43/TCP    21m
>
> kubectl get ep app2-macvlan
NAME           ENDPOINTS                                  AGE
app2-macvlan   192.168.100.10:43,192.168.100.11:43        21m
>
>
```

1. 使用headless service来给Macvlan网卡做服务发现
2. 生产实践经验看，固定IP更彻底更直接，避免DNS缓存的干扰

```
root@app-565db8494f-frtxk:/# nslookup app2.default.svc.cluster.local
Server:         10.43.0.10
Address:        10.43.0.10#53

Name:   app2.default.svc.cluster.local
Address: 10.42.0.10
Name:   app2.default.svc.cluster.local
Address: 10.42.0.11

root@app-565db8494f-frtxk:/# nslookup app2-macvlan.default.svc.cluster.local
Server:         10.43.0.10
Address:        10.43.0.10#53

Name:   app2-macvlan.default.svc.cluster.local
Address: 192.168.100.10
Name:   app2-macvlan.default.svc.cluster.local
Address: 192.168.100.11

root@app-565db8494f-frtxk:/#
```

重点问题：ARP缓存表同步

GOTC

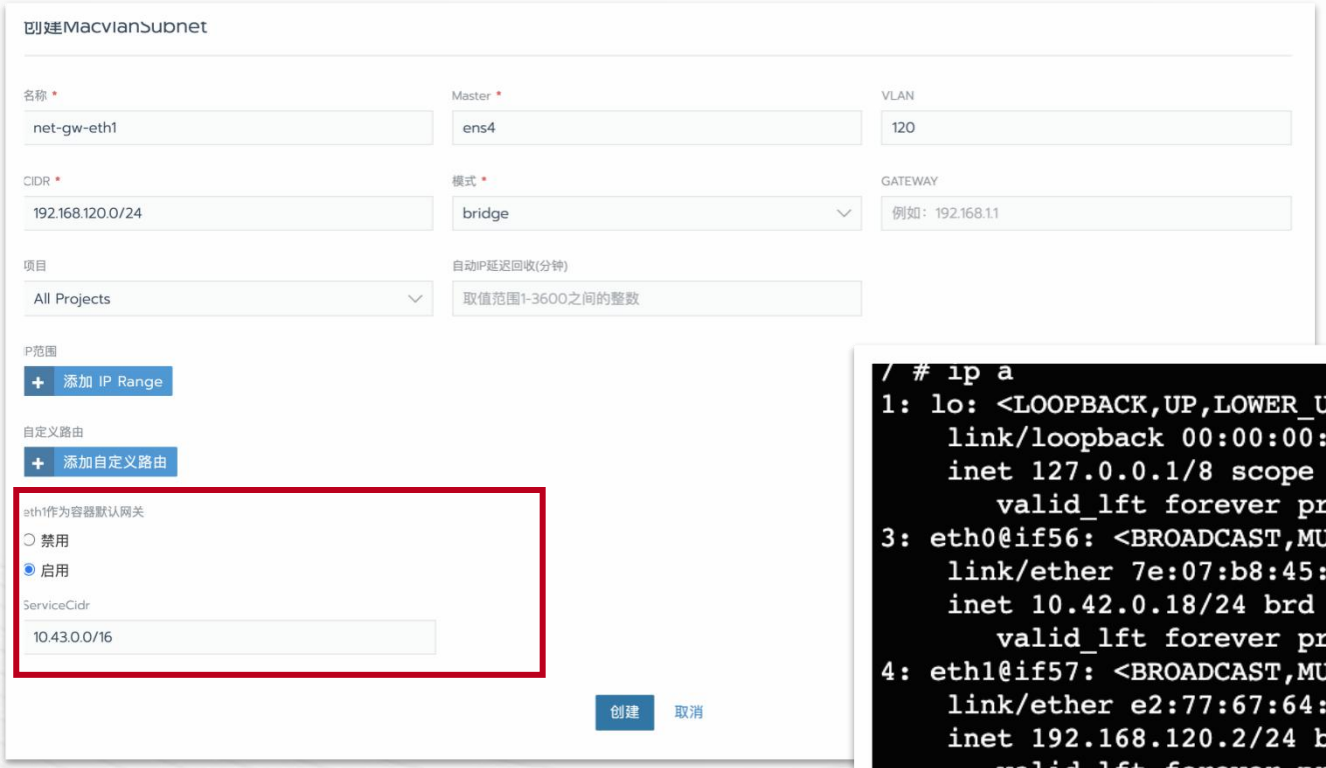
```
err = netns.Do(func(_ ns.NetNS) error {
        if err := ipam.ConfigureIface(args.IfName, result); err != nil {
                return err
        }

        contVeth, err := net.InterfaceByName(args.IfName)
        if err != nil {
                return fmt.Errorf("failed to look up %q: %v", args.IfName, err)
        }

        for _, ipc := range result.IPs {
                if ipc.Address.IP.To4() != nil {
                        _ = arping.GratuitousArpOverIface(ipc.Address.IP, *contVeth)
                }
        }
        return nil
})
```

1. Pod网卡启动后自动发送Gratuitous ARP，通知交换机和其他Pod更新ARP缓存表
2. Pod容器启动时，应用镜像自动向网关地址发送ICMP亦可取得同样效果

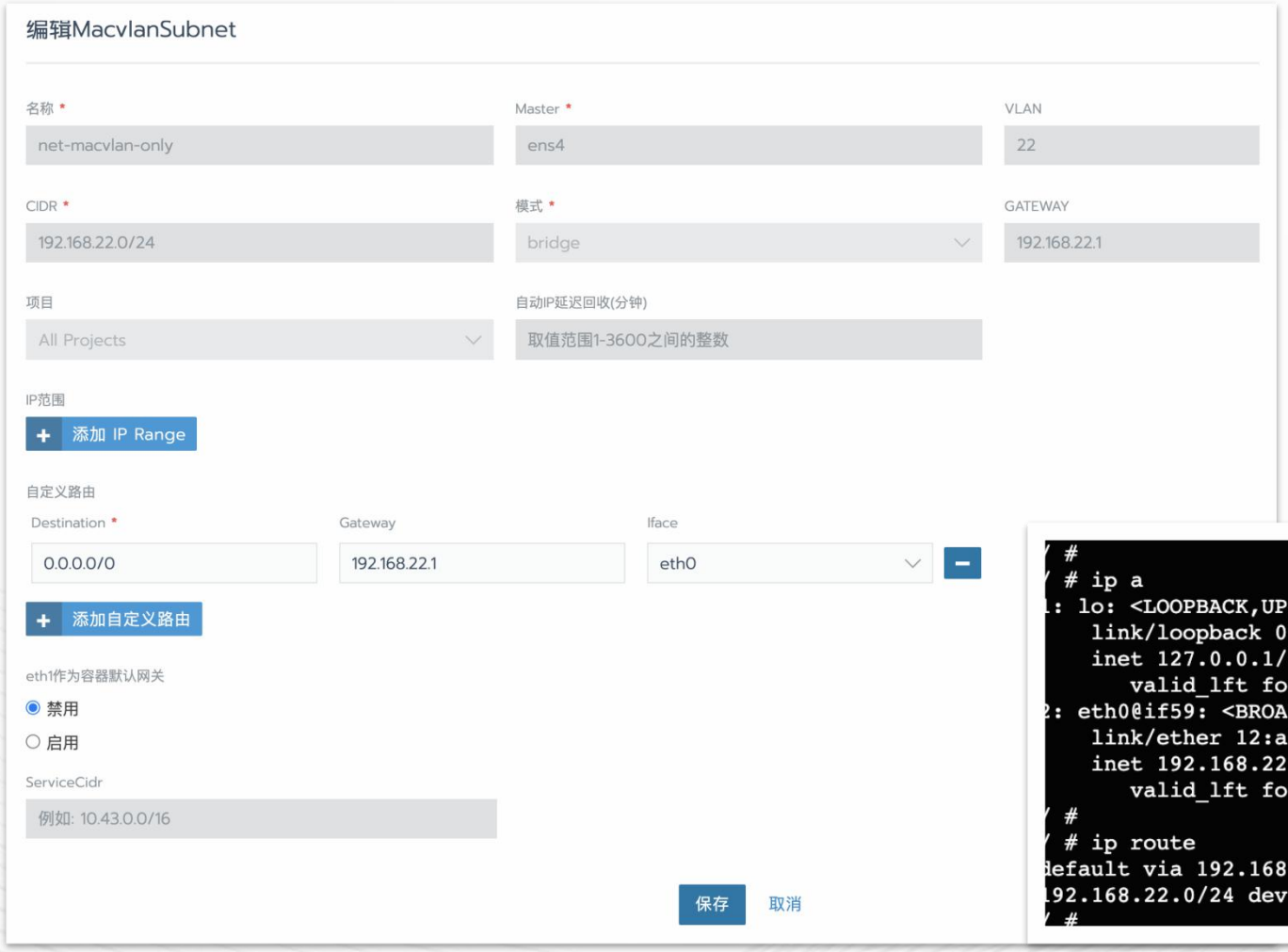
重点问题：避免主网卡的SNAT干扰



1. 默认情况下，对其他非macvlan网段的访问走eth0(Flannel/Canal等)，SNAT成host IP
2. 以Macvlan网卡作为默认网关，外部网关打通其他网段，避免SNAT导致Source IP变动

```
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue 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
3: eth0@if56: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1450 qdisc noqueue
    link/ether 7e:07:b8:45:d7:d7 brd ff:ff:ff:ff:ff:ff
    inet 10.42.0.18/24 brd 10.42.0.255 scope global eth0
       valid_lft forever preferred_lft forever
4: eth1@if57: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether e2:77:67:64:c2:da brd ff:ff:ff:ff:ff:ff
    inet 192.168.120.2/24 brd 192.168.120.255 scope global eth1
       valid_lft forever preferred_lft forever
/ #
/ # ip route
default via 192.168.120.1 dev eth1
10.42.0.0/24 dev eth0 scope link  src 10.42.0.18
10.42.0.0/16 via 10.42.0.1 dev eth0
10.43.0.0/16 via 10.42.0.1 dev eth0
192.168.120.0/24 dev eth1 scope link  src 192.168.120.2
/ #
```

重点问题：纯粹Macvlan单网卡



1. Pod容器中只有一个Macvlan网卡，无法兼容Kubernetes网络模型，各种svc服务发现、健康检查等机制均失效
2. 纯粹把容器作为虚拟机场景使用（过渡性方案）
3. 适合向Kubernetes平滑过渡的方案，外部自建服务发现

```
/ #
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue 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: eth0@if59: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 12:a8:1c:59:13:83 brd ff:ff:ff:ff:ff:ff
    inet 192.168.22.3/24 brd 192.168.22.255 scope global eth0
       valid_lft forever preferred_lft forever
/ #
/ # ip route
default via 192.168.22.1 dev eth0
192.168.22.0/24 dev eth0 scope link  src 192.168.22.3
/ #
```

GOTC

THANKS

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE