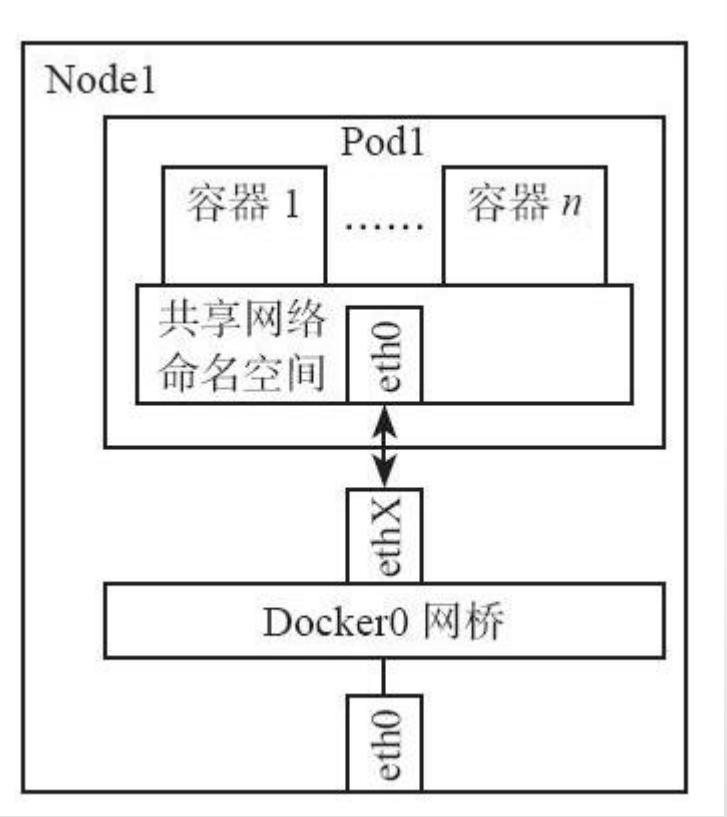
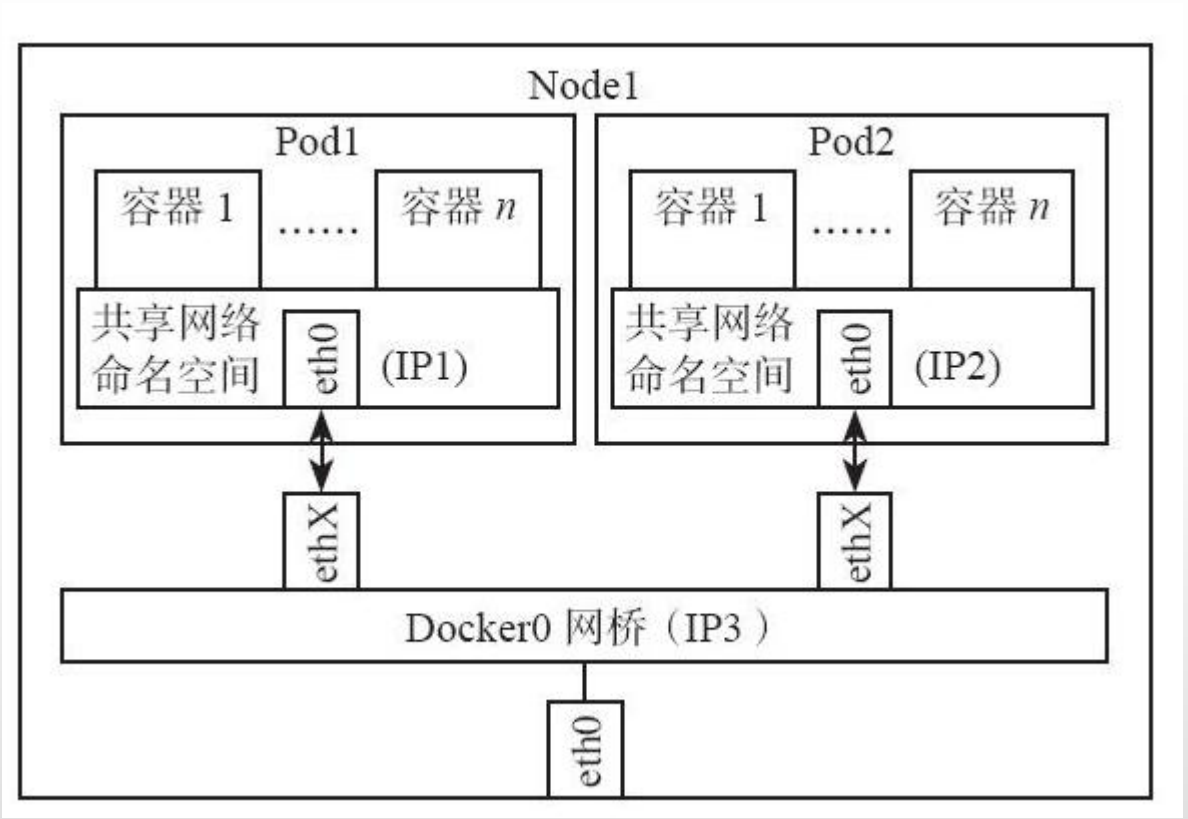
CKA-网络

讲师： 老段 RHCE/RHCA/COA/CKA

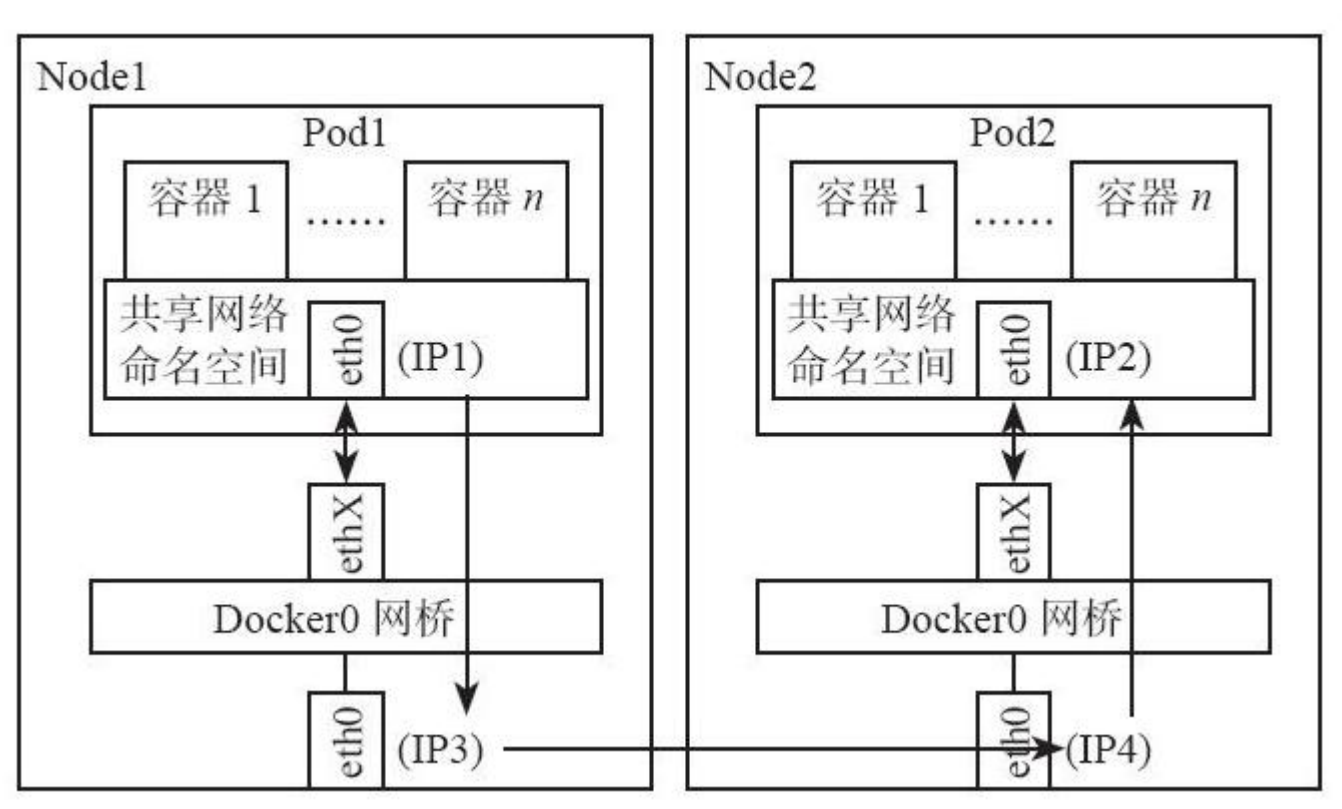
pod间的通信





图片摘自《基于kubernetes的容器云平台实战》

pod间的通信



图片摘自《基于kubernetes的容器云平台实战》

有的基于overlay网络实现

有的基于vxlan实现

有的基于openvswitch实现

有的基于BGP实现

网络解决方案

CNI(container network interface) CNCF下的一个项目，由coreOS提出

通过插件的方式统一配置

flannel---基于overlay 不支持网络策略

calico---基于BGP 支持网络策略

canal 支持网络策略

各种解决方案的对比

特性 \ 方案	Flannel	Calico	macvlan	Open vSwitch	直接路由
方案特性	通过虚拟设备 flannel0 实现对 docker0 的管理	基于 BGP 协议的纯三层的网络方案	基于 Linux Kernel 的 macvlan 技术	基于隧道的虚拟路由器技术	基于 Linux Kerne 的 vRouter 技术
对底层网络的要求	三层互通	三层互通	二层互通	三层互通	二层互通
配置难易度	简单 基于 etcd	简单 基于 etcd	简单 直接使用宿主机网络，需要仔细规划 IP 地址范围	复杂 需手工配置各节点的 bridge	简单 使用宿主机 vRoute 功能，需要仔细规划每个 Node 的 IP 地址范围

续表

特性 \ 方案	Flannel	Calico	macvlan	Open vSwitch	直接路由
网络性能	host-gw > VxLAN > UDP	BGP 模式性能损失小 IPIP 模式较小	性能损失可忽略	性能损失较小	性能损失小
网络联通性限制	无	在不支持 BGP 协议的网络环境下无法使用	基于 macvlan 的容器无法与宿主机网络通信	无	在无法实现大二层互通的网络环境下无法使用

不管哪种解决方案

每个pod都有独立IP

可以直接通信

只是性能及配置的难易

及是否支持网络策略

图片摘自《kubernetes权威指南》

配置canal网络

--在maser上执行
kubeadm init --kubernetes-version=v1.11.1 --pod-network-cidr=10.244.0.0/16

kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/installation/hosted/canal/rbac.yaml

kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/installation/hosted/canal/canal.yaml

网络策略

```
apiVersion:
networking.k8s.io/v1
kind: NetworkPolicy
metadata:
 name: test-network-
policy
 namespace: default
spec:
 podSelector:
  matchLabels:
   aa: aa1
 policyTypes:
 - Ingress
 - Egress
 ingress:
 - from:
  - podSelector:
    matchLabels:
     xx: xx
- ipBlock:
    cidr: 192.168.26.0/24
  ports:
  - protocol: TCP
   port: 80
  - protocol: TCP
   port: 31234
 egress:
 - to:
  - ipBlock:
    cidr: 192.168.26.0/24
  ports:
  - protocol: TCP
   port: 80
```

podSelect：设置对哪些pod生效

ingress:

设置来自哪些客户端 from

可以访问我的哪个端口ports

默认策略

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
 name: default-deny
spec:
 podSelector: {}
 policyTypes:
 - Ingress
```

测试

kubectl run nginx --image=nginx --replicas=1 --labels=aa=aa1

kubectl run nginx2 --image=nginx --replicas=1 --labels=bb=bb2

kubectl expose deployment nginx --port=80 --type=NodePort

kubectl expose deployment nginx2 --port=80 --type=NodePort

kubectl run busybox --rm -it --image=busybox sh

在没创建网络策略的情况下
wget nginx
rm -rf index.html
wget nginx2
说明可以正常访问

也能ping通

测试2

kubectl run nginx --image=nginx --replicas=1 --labels=aa=aa1

kubectl run nginx2 --image=nginx --replicas=1 --labels=bb=bb2

kubectl expose deployment nginx --port=80 --type=NodePort

kubectl expose deployment nginx2 --port=80 --type=NodePort

在没有任何网络策略的情况下
kubectl run busybox --rm -it --image=busybox sh
wget nginx
rm -rf index.html
wget nginx2
说明可以正常访问

例子

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
 name: test-network-policy
 namespace: default
spec:
 podSelector:
  matchLabels:
   aa: aa1
 policyTypes:
 - Ingress
 ingress:
 - from:
  - podSelector:
    matchLabels:
     xx: xx
  - ipBlock:
    cidr: 0.0.0.0/0
  ports:
  - protocol: TCP
   port: 80
```

只允许

来自 标签为aa=xx的pod访问

目的地 标签为aa1的pod 的端口80

kubectl run busybox --rm -it --image=busybox sh

此时访问不了

wget nginx 没法下载

ping nginx1_pod_ip 也是ping不通的

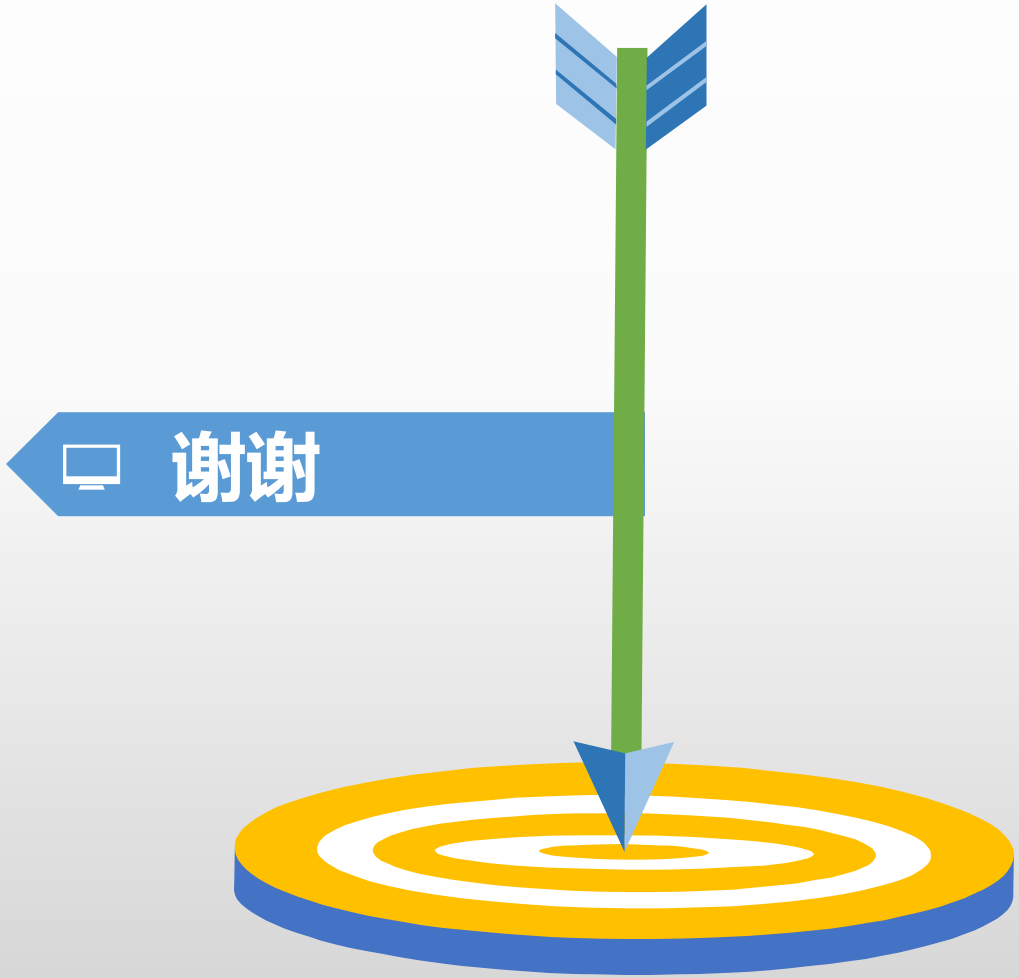
访问nginx2没问题

kubectl run busybox --rm -it --labels=xx=xx --image=busybox sh

此时
wget nginx 可以下载，即能访问端口80

ping nginx1_pod_ip 是ping不通的，只能访问端口80

访问nginx2没问题


谢谢