阿里云 × CLOUD NATIVE COMPUTING FOUNDATION

云原生技术公开课

关注“阿里巴巴云原生”公众号
获取第一手技术资料

第 13 讲

Kubernetes网络概念及策略控制

叶磊 阿里巴巴高级技术专家

1 Kubernetes 基本网络模型

2 Netns探秘

3 主流网络方案简介

4 Network Policy 的用处

5 思考时间

基本法： 约法三章 + 四大目标

Kubernetes 对于 Pod 间的网络没有任何限制，只需满足如下「三个基本条件」：

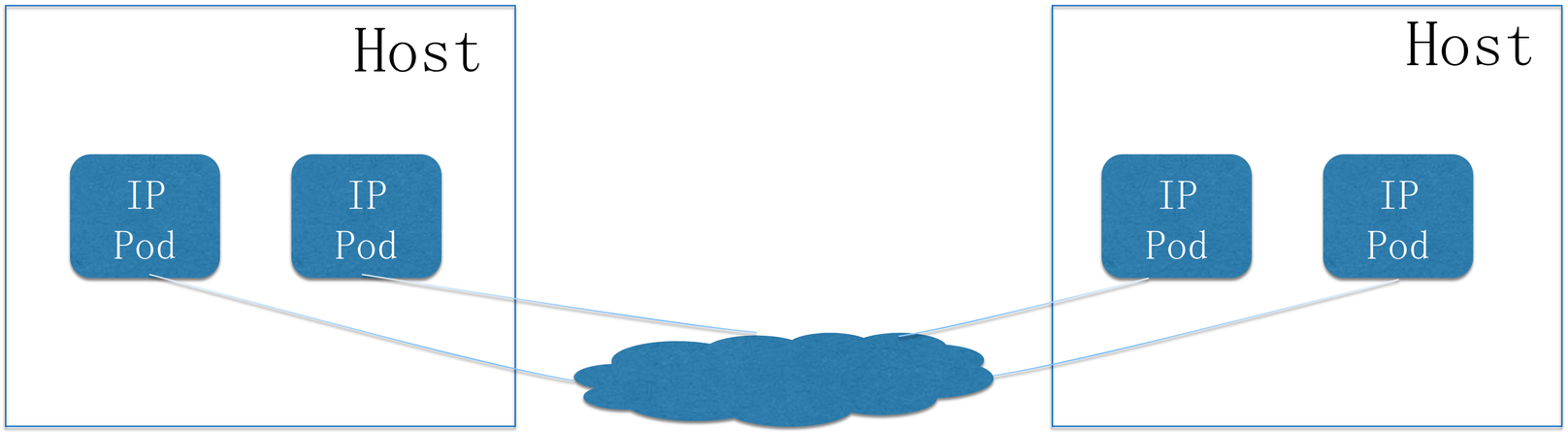
- 所有 Pod 可以与其他 Pod 直接通信，无需显式使用 NAT
- 所有 Node 可以与所有 Pod 直接通信，无需显式使用 NAT
- Pod 可见的 IP 地址确为其他 Pod 与其通信时所用，无需显式转换

基于以上准入条件，我们在审视一个网络方案的时候，需要考虑如下「四大目标」：

- 容器与容器间的通信
- Pod 与 Pod 之间的通信
- Pod 与 Service 间的通信
- 外部世界与 Service 间的通信

对基本约束的解释

容器与其宿主存在寄生关系，从而在实现上，容器网络方案可分为 **Underlay/Overlay** 两大派别，其主要的差异在于是否与 **Host** 网络同层，这样对于微服务发现及治理，容器可访问方式都造成很大的差异，所以社区的同学以 **perPodperIP** 这种简单武断的模型，摈弃了显示端口映射等 **NAT** 配置，统一了容器网络对外服务的视角。



1 Kubernetes 基本网络模型

2 Netns探秘

3 主流网络方案简介

4 Network Policy 的用处

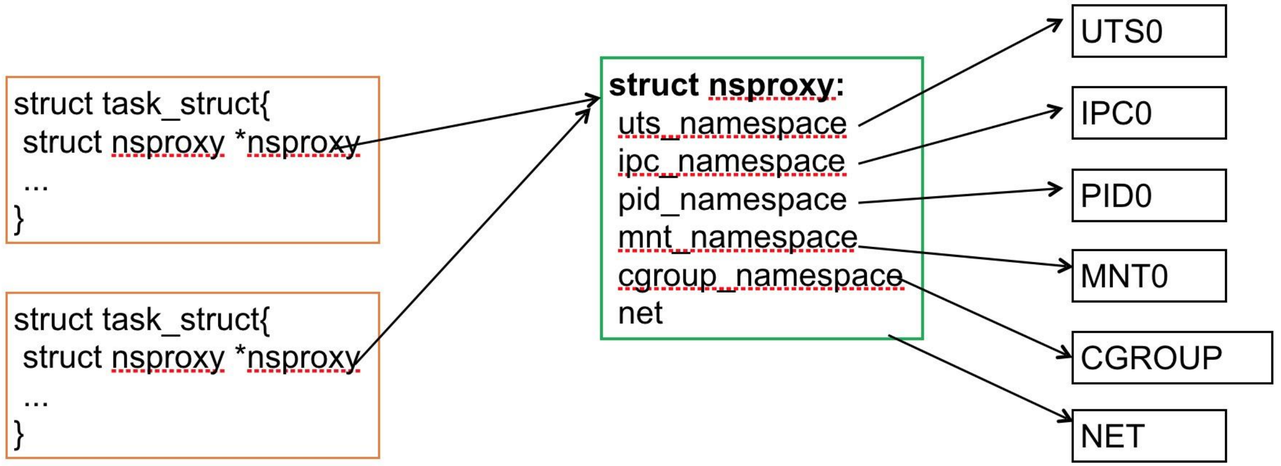
5 思考时间

Netns 究竟实现了什么

Network namespace 是实现网络虚拟化的内核基础，创建了**隔离的网络空间**

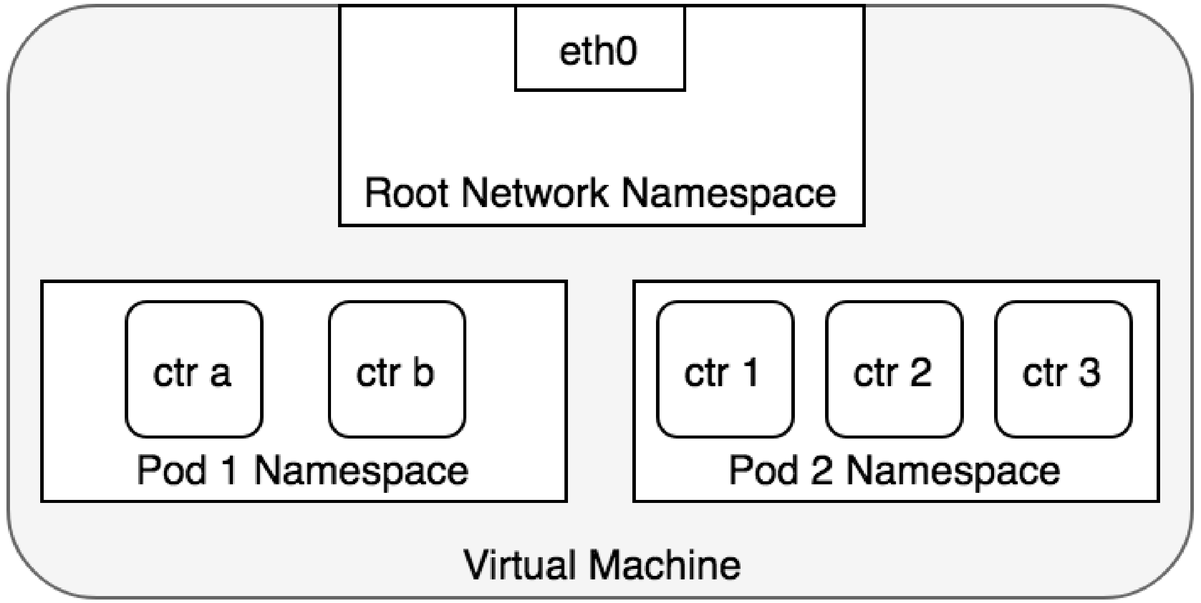
- 拥有独立的附属网络设备（**lo**、**veth** 等虚设备/物理网卡)
- 独立的协议栈，**IP** 地址和路由表
- **iptables** 规则
- **ipvs** 等

nsproxy相当于运行环境



Pod 与 Netns 的关系

每个 **Pod** 拥有独立的 **Netns** 空间，**Pod** 内的 **Container** 共享该空间，可通过 **Loopback** 接口实现通信，或通过共享的 **Pod-IP** 对外提供服务。别忘记，宿主上还有个 **Root Netns**，可以看做一个特殊的容器空间。



1 Kubernetes 基本网络模型

2 Netns探秘

3 主流网络方案简介

4 Network Policy 的用处

5 思考时间

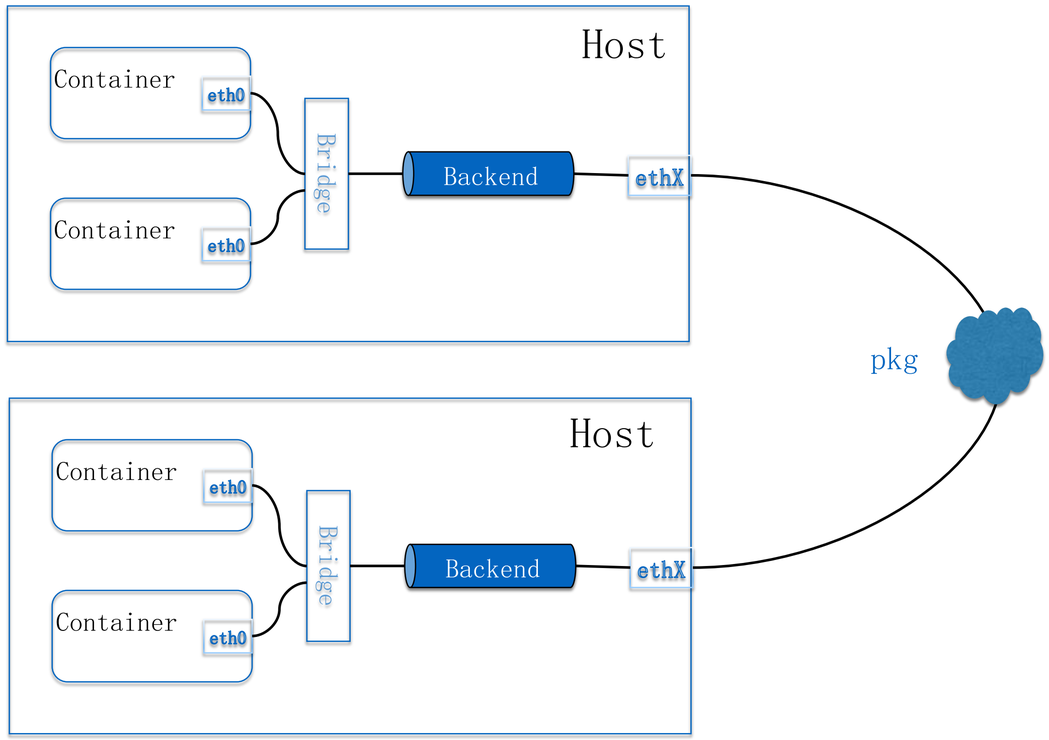
典型容器网络实现方案

容器网络可能是 **Kubernetes** 领域最为百花齐放的一个领域，依照 **IaaS** 层的配置、外部物理网络的设备、性能 **or** 灵活优先，可以有不同的实现：

- **Flannel**，最为普遍的实现，提供多种网络 **backend** 实现，覆盖多种场景
- **Calico**，采用 **BGP** 提供网络直连，功能丰富，对底层网络有要求
- **Canal (Flannel for network + Calico for firewalling)**，嫁接型创新项目
- **Cilium**，基于 **eBPF** 和 **XDP** 的高性能 **Overlay** 网络方案
- **Kube-router**，同样采用 **BGP** 提供网络直连，集成基于 **LVS** 的负载均衡能力
- **Romana**，采用 **BGP or OSPF** 提供网络直连能力的方案
- **WeaveNet**，采用 **UDP** 封装实现 **L2 Overlay**，支持用户态（慢，可加密）/内核态（快，不能加密）两种实现

Flannel 方案

Flannel 是目前使用最为普遍的方案，通过将 **backend** 机制独立，它目前已经支持多种数据路径，也可以适用于 **overlay/underlay** 等多种场景，封装可以选用用户态 **udp**（纯用户态实现），内核 **Vxlan**（性能好），如集群规模不大，处于同一二层域，也可以选择 **host-gw** 方式。



1 Kubernetes 基本网络模型

2 Netns探秘

3 主流网络方案简介

4 Network Policy 的用处

5 思考时间

Network Policy 基本概念

Network Policy 提供了基于策略的网络控制，用于隔离应用并减少攻击面。它使用标签选择器模拟传统的分段网络，并通过策略控制它们之间的流量以及来自外部的流量。

在使用 Network Policy 之前，需要注意：

- apiserver 开启 extensions/v1beta1/networkpolicies
- 网络插件要支持 Network Policy，如 Calico、 Romana、 Weave Net 和 trireme 等

配置实例

通过使用标签选择器（包括 **namespaceSelector** 和**podSelector**）来控制 **Pod** 之间的流量。

如右图配置：

- 允许 **default namespace** 中带有 **role=frontend** 标签的**Pod** 访问 **default namespace** 中带有 **role=db** 标签 **Pod** 的 **6379** 端口
- 允许带有 **project=myprojects** 标签的 **namespace** 中所有 **Pod** 访问 **default namespace** 中带有 **role=db** 标签**Pod** 的 **6379** 端口

```yaml
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: tcp
      port: 6379
```

1 Kubernetes 基本网络模型

2 Netns探秘

3 主流网络方案简介

4 Network Policy 的用处

5 思考时间

小节总结

- **Pod** 在容器网络中的核心概念是 **IP**，每个 **Pod** 必须有内外视角一致的独立 **IP** 地址
- 影响容器网络性能的关键是拓扑设计，也就数据包端到端的路径设计
- 牢记 **Overlay/Underlay** 下各种网络方案的设计选择，如果不知道，可以这样选：普适性最强 —— **Flannel-VxLan**，2层可直连 —— **Calico/Flannel-Hostgw**
- **Network Policy** 是个强大的工具，可以实现 **ingress** 的流量精确控制，关键是选择好 **Pod Selector**

思考一下

- 为什么网络接口标准化了（CNI），而网络方案没有标准化？
- 为什么 Network Policy 没有交给一个标准 Controller 来实现，而是交给方案提供方？
- 能不能完全不用 Net-dev 型的设备，实现一个容器网络？
- 网络问题排查，值不值得做一个开源工具实现？

关注“阿里巴巴云原生”公众号
获取第一手技术资料

谢谢观看

THANK YOU