KUBERNETES COMMUNITY DAYS BEIJING 2025

KubeVrit 网络深度探索

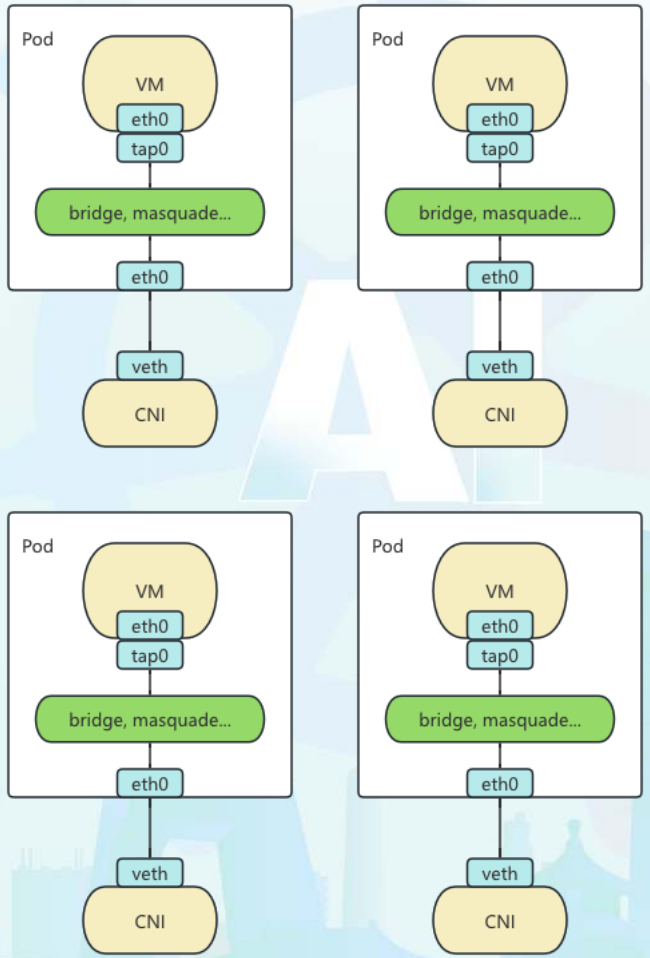
灵雀云 —— 刘梦馨

Content

目录

KubeVirt 网络概述

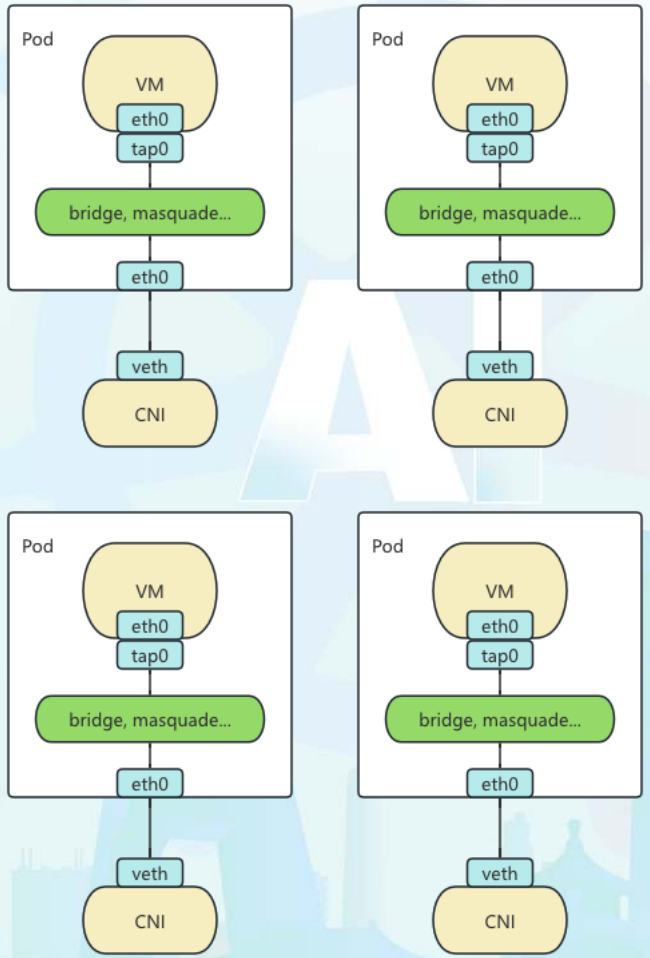
- KubeVirt 采用 Pod 运行 VM，复用 CNI 网络
- 网络分为两部分：Pod 网络（CNI 提供）与 VM 网络（Domain 网络）
- 如何连接这两者？
  - Bridge、Masquerade、SR-IOV、Passt、Slirp



Bridge 网络模式

- Bridge 模式将 VM 的 tap0 和 CNI 的 eth0 接入同一个 bridge
- 将容器网络的 eth0 地址转移给 VM 的 eth0
- virt-handler 运行 DHCP Server 拦截 DHCP 请求
- 性能较好，且和容器网络共享 IP 地址，管理较为方便
- 配合 Underlay 网络可以直接实现 VM 和外部网络互通

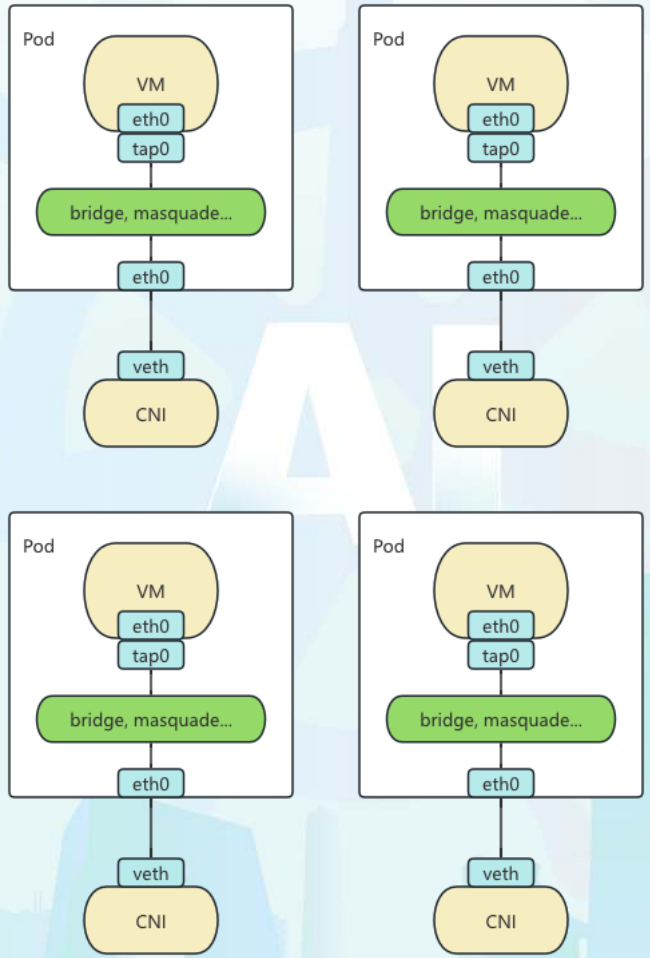
- 局限性
  - virt-handler DHCP 功能有限，不支持 IPv6 RA 不支持高级功能
  - KubeVirt 默认不支持 Bridge 模式网络热迁移
  - 存在一层 bridge 有额外的性能开销



Masquerade 网络模式

- 通过 nftables 将 VM 网络流量 NAT 到容器网络
- VM 对外暴露的为 Pod IP，内部为独立的 IP
- 支持 VM IP 不变
- 支持 live migration

- 局限性
  - VM Pod IP 不一致增加管理复杂度
  - NAT 会有较高性能损失
  - 对 Underlay 网络不友好

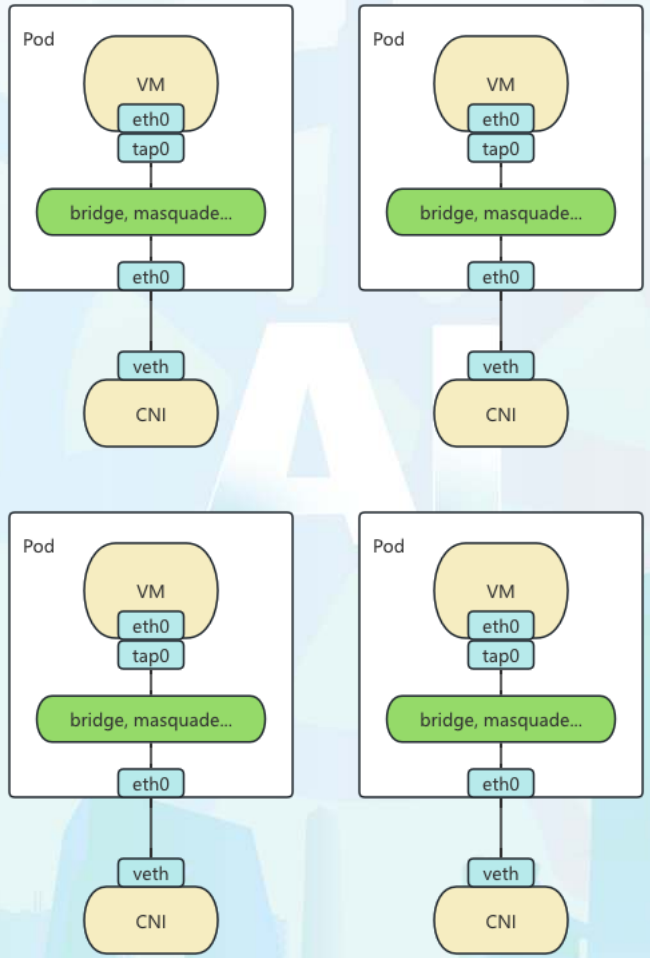


其他网络方案

- Passt, Slirp, macvtap, SR-IOV ...
- 每个网络方案都解决了特定场景的问题，但是也有各自的局限性

- 1.4.0 之前所有网络方案代码都集成在 kubevirt
- 如何对已有的网络方案进行调整？
- 如何增加新的网络方案？

- 所有网络方案都是把 CNI 和 domain 网络连接，是不是有新的抽象机制？



Hook Sidecar

- 允许在 VM 启动前通过自定义镜像或脚本修改 libvirt 启动参数
- 类似 CNI 的接口，可以在调用 libvirt 前执行两个特定的二进制文件
  - 通过 `/usr/bin/onDefineDomain` 修改 libvirt XML
  - 通过 `/usr/bin/preCloudInitIso` 修改 cloud-init 配置
- 具备强大的灵活性
  - libvirt 的所有参数都可以按需调整
  - 可以执行任意代码，甚至可以覆盖 KubeVirt 的默认行为

Network Binding Plugin

- 与 Hook Sidecar 类似，但改用 gRPC 机制
- 支持 `onDefineDomain` 和 `preCloudInitIso` 方法
- 内置 plugin: bridge, masquerade, managedTap
- passt， slirp 被移出 kubevirt 主线代码
- 允许外部插件扩展 KubeVirt 网络，不需修改 KubeVirt 核心代码

What's Next

- 为什么 KubeVirt 网络模式这里会有这么多奇奇怪怪的机制？
  - 必须兼容 Pod 网络，容器网络设计并未考虑过还要再将网络接入 VM
- 传统虚拟化如何做？
  - 直接将 tap 设备接入 OVS，Linux Bridge ...
- KubeVirt 模式下容器网络是否还合适？
- 既然 Network Binding Plugin 提供了足够的灵活性我们就玩点大的

Kube-OVN and Network Binding Plugin

- 使用 webhook 将 KubeVirt 的 Pod 之间创建成主机网络模式
  - 跳过 CNI 的处理
  - 虚拟化本身的隔离性更强，无需单独的 network namespace 做隔离
- 自定义 Network Binding Plugin 管理网络
  - 在宿主机创建 tap 设备接入 OVS
  - 还原虚拟化场景的传统用法

测试结果

TCP 延迟测试结果 (单位: us)

测试场景	64k	32k	16k	8k	4k	2k	1k	512	256	128	64
场景一：自定义虚机直通tap	135	87.4	71.2	59.5	57.8	55.3	48.9	49.5	47.3	49.0	48.5
场景二：自定义虚机 tap+bridge+veth	133	92.1	74.6	68.3	65.8	60.2	55.1	54.9	54.2	54.5	55.7
场景三：自定义虚机 tc bypass	134	94.6	74.7	61.1	60.7	57.1	51.3	50.5	49.3	51.3	49.9
场景四：kubevirt 虚机	130	86.0	70.8	60.1	58.8	55.3	49.4	49.0	49.3	48.7	47.7
场景五：kubevirt 虚机 tc bypass	136	85.8	70.9	59.6	58.6	54.0	47.6	48.2	48.9	48.7	48.0

UDP 延迟测试结果 (单位: us)

测试场景	64k	32k	16k	8k	4k	2k	1k	512	256	128	64
场景一：自定义虚机直通tap	0	250	138	85.0	63.1	58.1	44.9	43.6	43.0	42.7	43.2
场景二：自定义虚机 tap+bridge+veth	458	291	157	102	74.8	65.2	49.1	49.8	49.7	49.4	48.5
场景三：自定义虚机 tc bypass	0	268	151	90.4	68.8	60.6	46.6	45.7	45.4	45.0	44.5
场景四：kubevirt 虚机	0	269	163	93.8	67.5	60.7	46.1	44.8	44.6	44.8	43.0
场景五：kubevirt 虚机 tc bypass	0	251	142	89.2	65.5	56.2	44.3	43.4	43.1	43.0	42.2

Thanks.