Kube-OVN：一个非著名的 K8s CNI 网络插件

目录

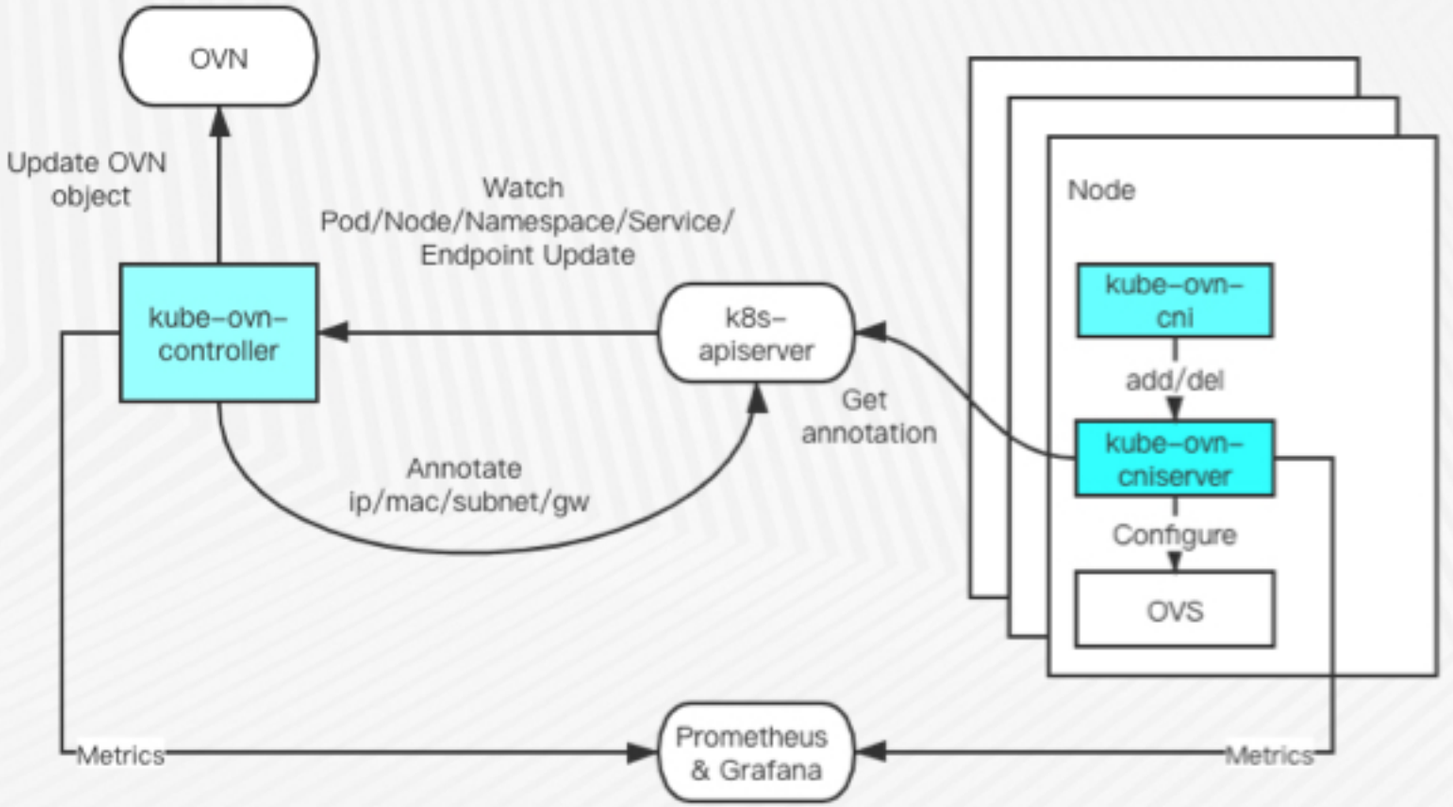
- 为何要 Kube-OVN?
- 什么是 Kube-OVN?
- 怎么用 Kube-OVN?

对 Kubernetes 网络:

- 如何兼容传统网络架构，容器网络需和已有基础实施互通互联
- 传统应用对固定IP的需求
- 容器多网络平面，多网卡的管理
- 多租户，VPC类型容器网络需求
- 多集群网络互通，跨云容器网络成为难题

- 利用社区最为成熟的 OVS 作为网络底座
- 基于 Kubernetes 架构原生设计
- 结合灵雀云企业端多年实践打造功能
- 复用 OVS 社区的生态

- 基于标准 K8s API，通过Operator模式进行开发
- 所有功能和配置可通过Annotation，CRD 完成
- 第三方可根据自己需求进行界面调整



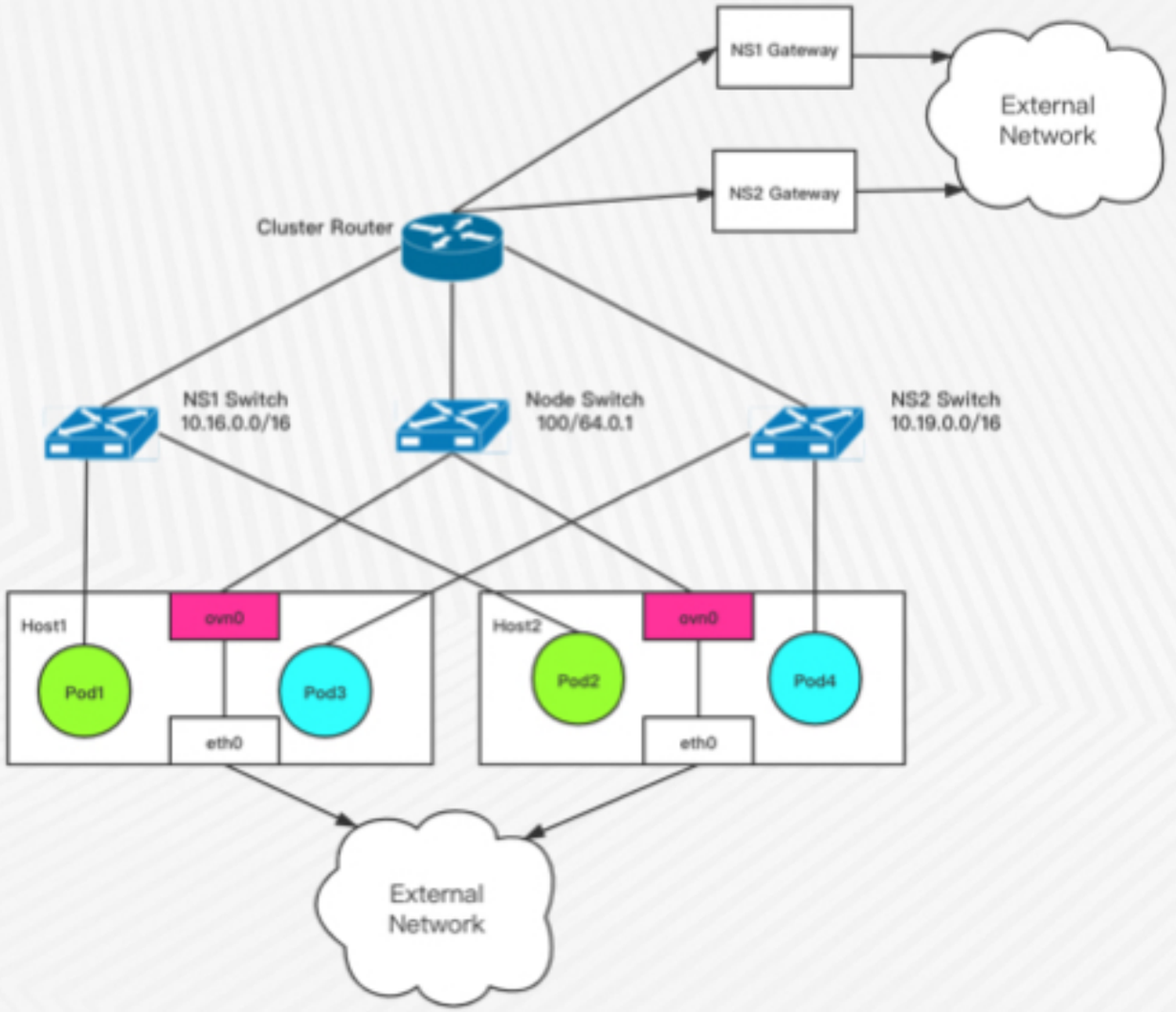
Kubernetes CNI的基本组网

Geneve Overlay 网络，可灵活添加网络功能(VPC, ACL, etc.)，并和现有物理网络保持独立

子网和主机无关，容器 IP 地址可以在整个集群漂移

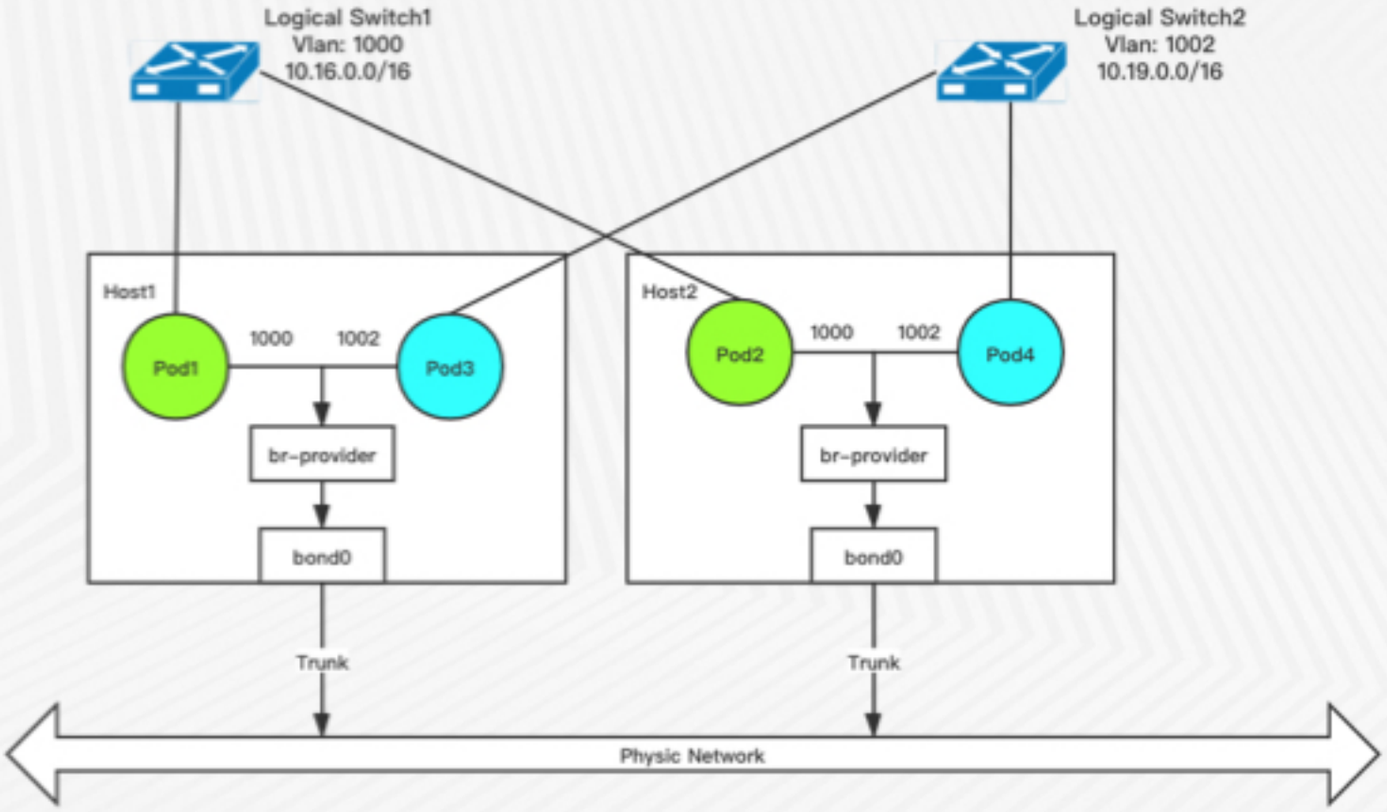
子网绑定 Namespace，方便多租户地址空间管理，以及地址和项目，应用进行关联

分布式网关和集中式网关两种出网方式，每个命名空间可以有独立的网络控制



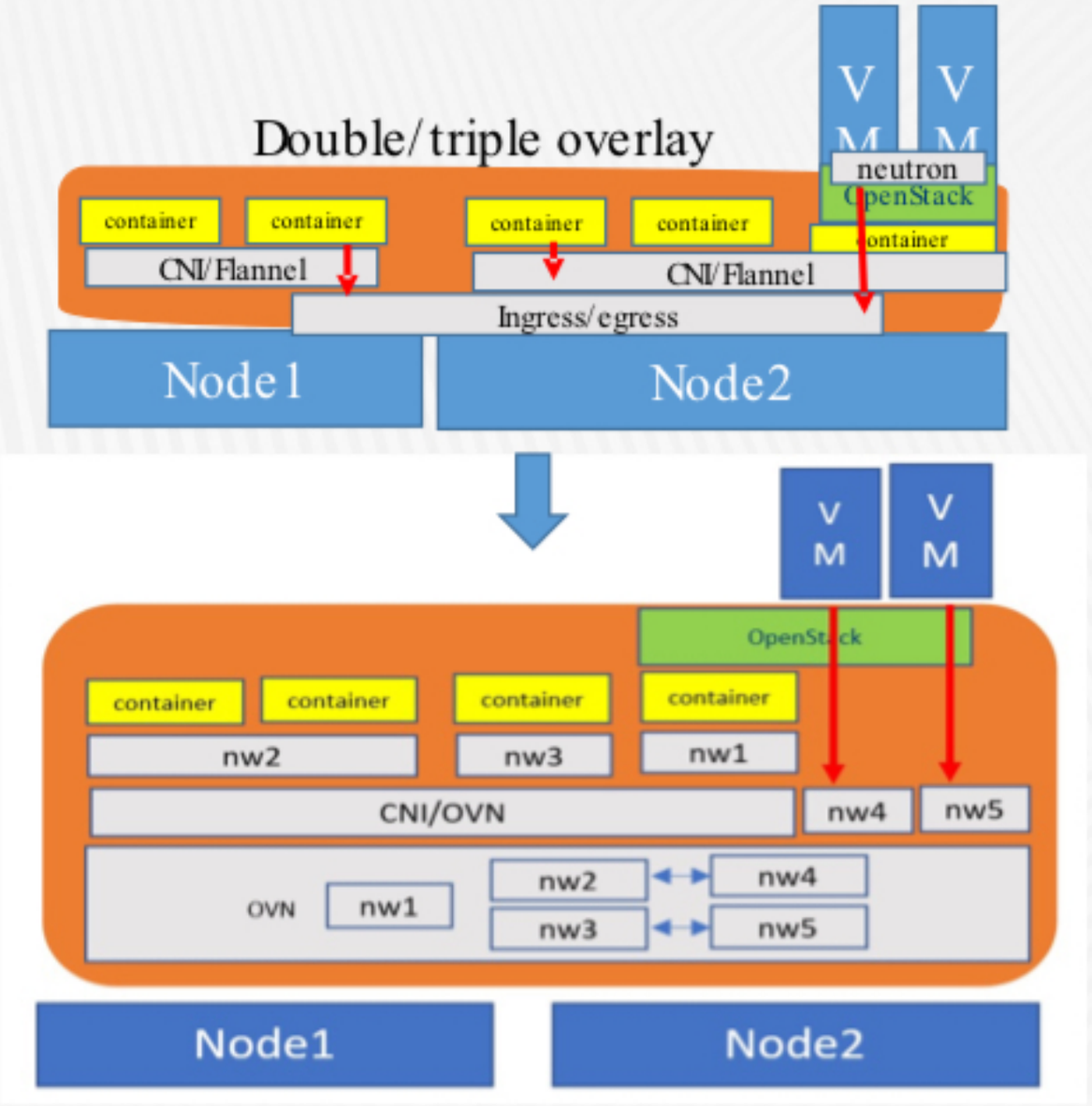
Kubernetes CNI的基本组网

- Underlay 网络，和物理网络直接打通，提供直连底层结构的高性能网络.
- 支持 Pod 运行在不同 Vlan 网络中，可通过 Vlan 实现隔离
- 提供固定 IP/Mac 的能力
- Underlay 与 Overlay 可共存互通



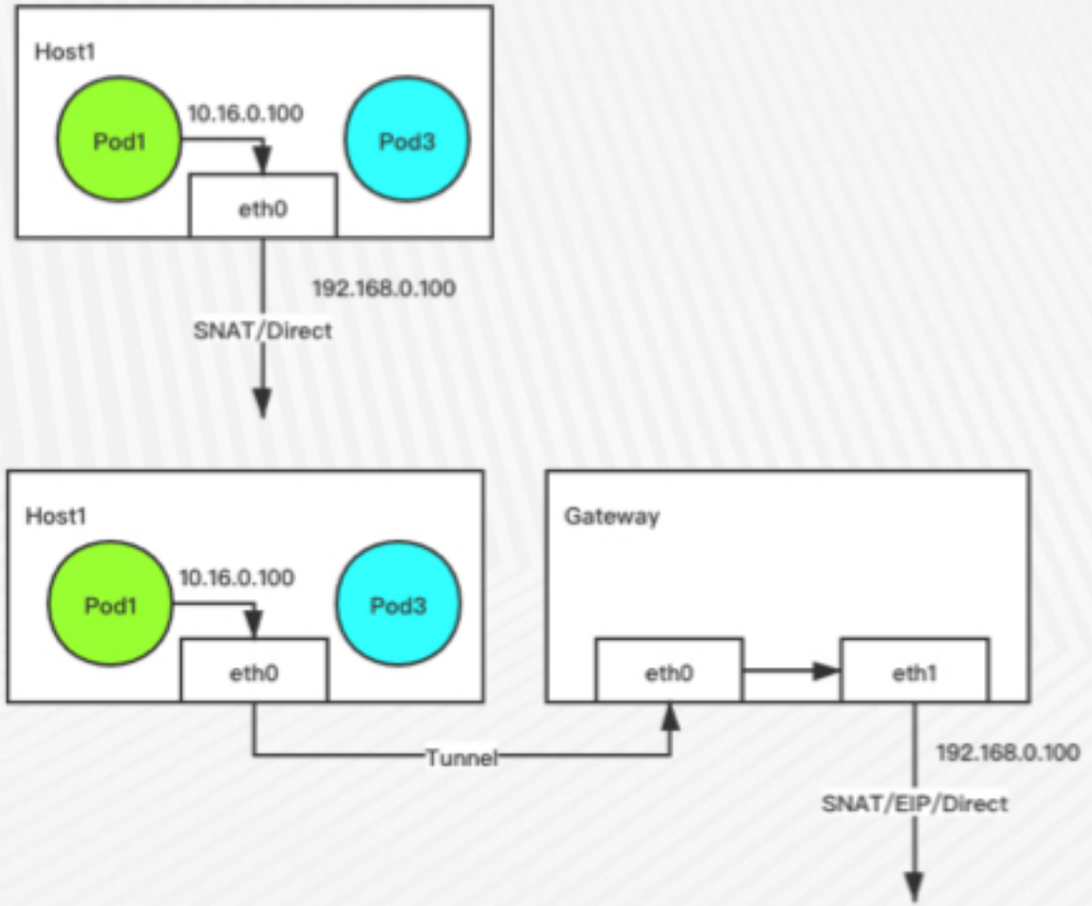
Kubernetes 和 OpenStack 互联互通

- OVN-IC 方式实现 OpenStack 内 VPC 和 Kubernetes 内 VPC 对等连接
- Kubernetes 和 OpenStack 共享底层 OVN 基础设施
- Kubernetes 内导入 OpenStack VPC，直接在 Kubernetes 内创建 OpenStack VPC 网络下 Pod

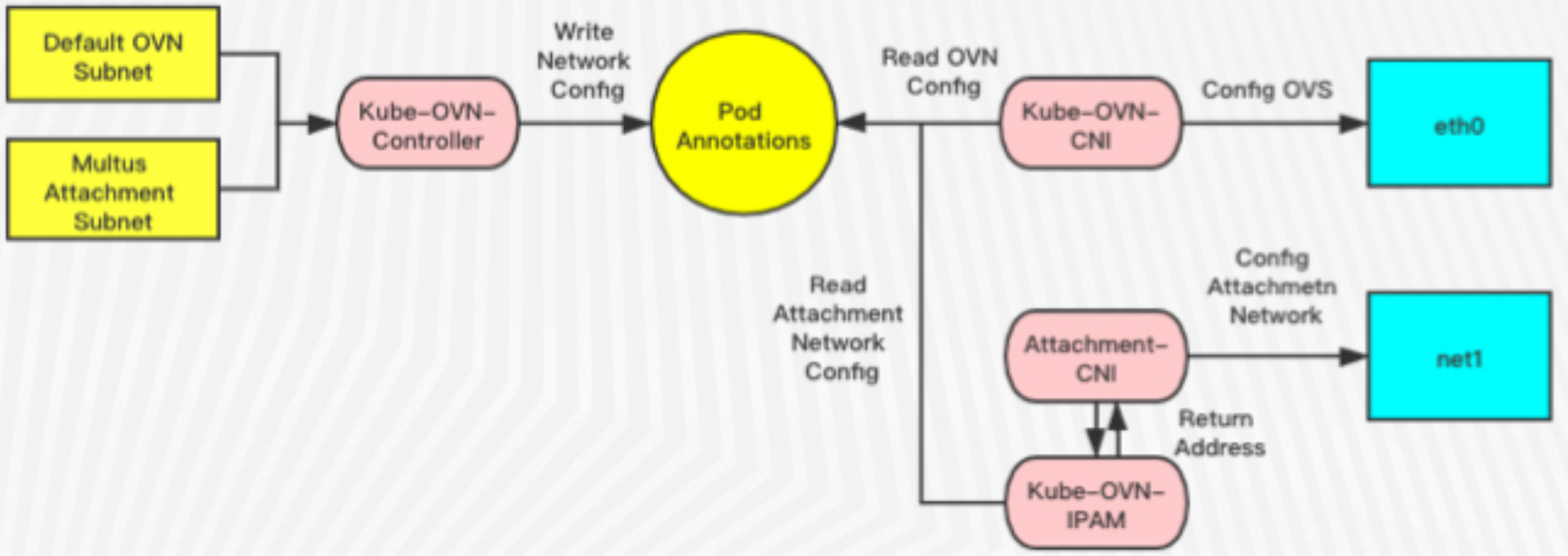


Kubernetes 对外打通

- Namespace 级别出网控制
  - 分布式网关
  - 集中式网关
  - NAT 控制
- Pod 级别出网控制
  - SNAT
  - EIP
- Underlay 方案
  - Pod 接入底层 Vlan，利用物理能力打通
- Kubernetes clusterset 打通
  - 可提供 pod ip 直连方案
  - 正在做基于 svc ip 的访问方案

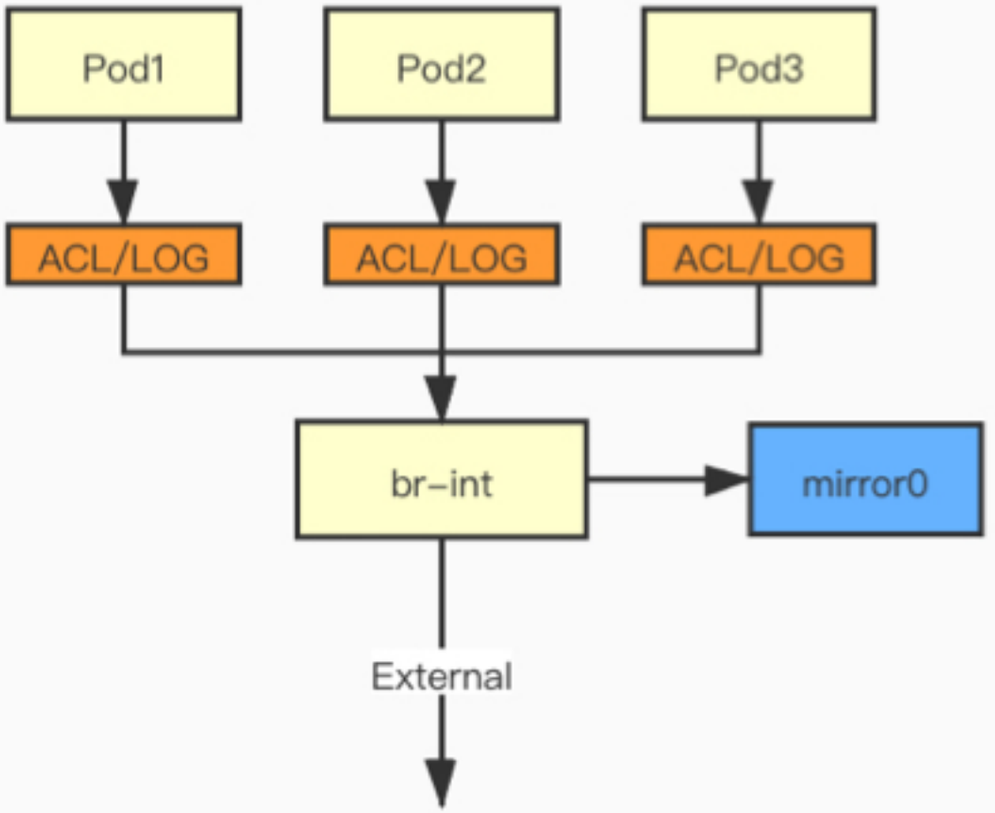


多网卡管理



- 特殊业务需要多网络：网关，VNF
- 统一管理主网络和附属网络
- 为附属网络提供子网，固定 IP 等功能

- 标准 NetworkPolicy 支持
- 子网 ACL 控制
- ACL 日志记录规则拦截数据
- PortSecurity 支持
- 容器流量全量镜像，便于安全审计，流量分析
- 动态 QoS 限制双向带宽



简单易用的工具

- OVN/OVS 的完整监控
- 集群网络质量实时监控
  - Node/Pod/Service/DNS 连通性
  - Node/Pod/Service/DNS 网络

- Tracing/Tcpdump 工具支持，性能测试工具支持.
- 容器网络流量完整镜像
- Prometheus/Grafana 集成
- Cilium 的集成

- https://github.com/alauda/kube-ovn/blob/master/docs/ovn-ovs-monitor.md

简单易用的工具

Trace

```
[root@liumengxin-ovn1-192 ~]# kubectl ko trace default/web-0 114.114.114.114 udp 53
Defaulting container name to ovn-central.
Use 'kubectl describe pod/ovn-central-748d766476-rnzkk -n kube-system' to see all of the
+ kubectl exec ovn-central-748d766476-rnzkk -n kube-system -- ovn-trace --ct=new ovn-def
== 114.114.114.114 && udp.src == 10000 && udp.dst == 53'
Defaulting container name to ovn-central.
Use 'kubectl describe pod/ovn-central-748d766476-rnzkk -n kube-system' to see all of the
# udp,reg14=0x2d,vlan_tci=0x0000,dl_src=00:00:00:38:06:e7,dl_dst=00:00:00:8d:3a:fc,nw_sr

ingress(dp="ovn-default", inport="web-0.default")
-------------------------------------------------
 0. ls_in_port_sec_l2 (ovn-northd.c:4769): inport == "web-0.default", priority 50, uuid
    next;
20. ls_in_l2_lkup (ovn-northd.c:7542): eth.dst == 00:00:00:8d:3a:fc, priority 50, uuid d
    outport = "ovn-default-ovn-cluster";
    output;

egress(dp="ovn-default", inport="web-0.default", outport="ovn-default-ovn-cluster")
-----------------------------------------------------------------------------------
 1. ls_out_pre_lb (ovn-northd.c:5118): ip, priority 100, uuid f1507423
    reg0[0] = 1;
    next;
 3. ls_out_pre_stateful (ovn-northd.c:5135): reg0[0] == 1, priority 100, uuid 054a7b02
    ct_next;

ct_next(ct_state=new|trk)
-------------------------
10. ls_out_port_sec_l2 (ovn-northd.c:4835): outport == "ovn-default-ovn-cluster", priori
    output;
    /* output to "ovn-default-ovn-cluster", type "patch" */

ingress(dp="ovn-cluster", inport="ovn-cluster-ovn-default")
----------------------------------------------------------
```

Tcpdump

```
[root@liumengxin-ovn1-192 ~]# kubectl ko tcpdump kube-system/kube-ovn-pinger-dpqt2
+ kubectl exec -it kube-ovn-cni-6zkfc -n kube-system -- tcpdump -nn -i 6454a509c784_h
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on 6454a509c784_h, link-type EN10MB (Ethernet), capture size 262144 bytes
06:05:10.374863 IP 10.16.0.2 > 192.168.16.44: ICMP echo request, id 14683, seq 2, length 24
06:05:10.375198 IP 192.168.16.44 > 10.16.0.2: ICMP echo reply, id 14683, seq 2, length 24
06:05:10.475656 IP 10.16.0.2 > 192.168.16.45: ICMP echo request, id 28537, seq 0, length 24
06:05:10.475991 IP 192.168.16.45 > 10.16.0.2: ICMP echo reply, id 28537, seq 0, length 24
06:05:10.575899 IP 10.16.0.2 > 192.168.16.45: ICMP echo request, id 28537, seq 1, length 24
06:05:10.576217 IP 192.168.16.45 > 10.16.0.2: ICMP echo reply, id 28537, seq 1, length 24
06:05:10.675879 IP 10.16.0.2 > 192.168.16.45: ICMP echo request, id 28537, seq 2, length 24
06:05:10.676252 IP 192.168.16.45 > 10.16.0.2: ICMP echo reply, id 28537, seq 2, length 24
06:05:10.776784 IP 10.16.0.2 > 192.168.16.46: ICMP echo request, id 3032, seq 0, length 24
06:05:10.776856 IP 192.168.16.46 > 10.16.0.2: ICMP echo reply, id 3032, seq 0, length 24
06:05:10.877009 IP 10.16.0.2 > 192.168.16.46: ICMP echo request, id 3032, seq 1, length 24
06:05:10.877063 IP 192.168.16.46 > 10.16.0.2: ICMP echo reply, id 3032, seq 1, length 24
06:05:10.977129 IP 10.16.0.2 > 192.168.16.46: ICMP echo request, id 3032, seq 2, length 24
06:05:10.977221 IP 192.168.16.46 > 10.16.0.2: ICMP echo reply, id 3032, seq 2, length 24
06:05:11.077668 IP 10.16.0.2.41252 > 10.96.0.1.443: Flags [P.], seq 2749951050:2749951089, ac
06:05:11.077924 IP 10.96.0.1.443 > 10.16.0.2.41252: Flags [.], ack 39, win 59, options [nop,n
06:05:11.081087 IP 10.96.0.1.443 > 10.16.0.2.41252: Flags [P.], seq 1:58, ack 39, win 59, opt
06:05:11.081108 IP 10.96.0.1.443 > 10.16.0.2.41252: Flags [.], seq 58:1406, ack 39, win 59, o
```

基于annotation的自定义参数

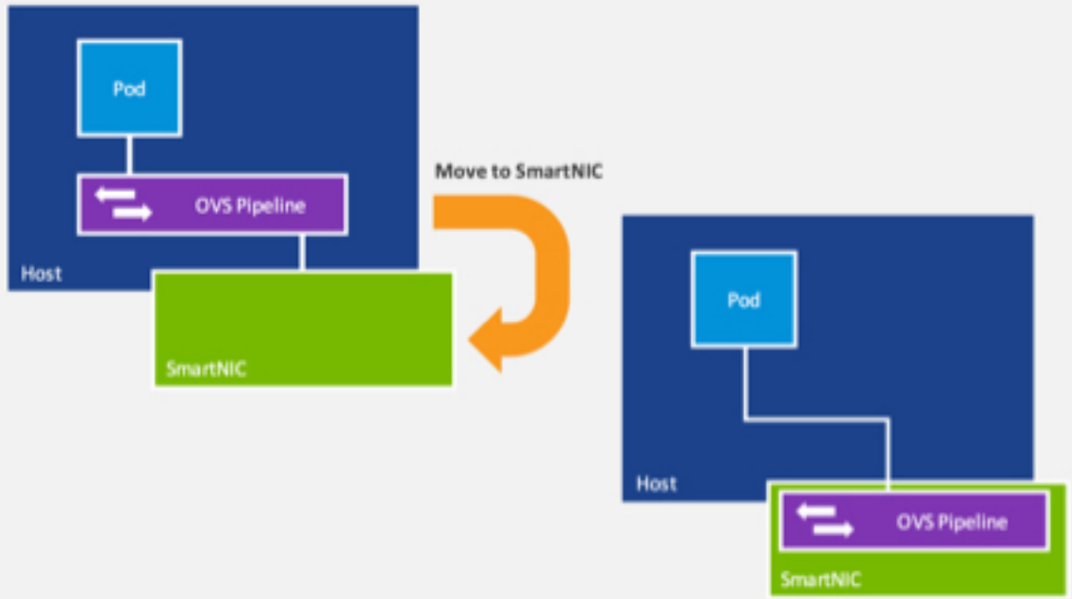
工作负载固定地址方法(例):

- Pod 可通过 annotation 指定固定 IP/MAC
- Workload 可通过 ippool annotation 指定一组 IP，达到固定 IP 效果
- StatefulSet 特殊优化，Pod 名和 IP 一一对应
- IP 地址可在整个集群内漂移

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-ip
  namespace: ls1
  annotations:
    ovn.kubernetes.io/ip_address: 10.16.0.15
    ovn.kubernetes.io/mac_address: 00:00:00:53:6B:B6
spec:
  containers:
  - name: static-ip
    image: nginx:alpine
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: ls1
  name: starter-backend
  labels:
    app: starter-backend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: starter-backend
  template:
    metadata:
      labels:
        app: starter-backend
      annotations:
        ovn.kubernetes.io/ip_pool: 10.16.0.15,10.16.0.16,10.16.0.17
```

- OVN调优，单集群支撑 200节点 1w+ Pod
- 流表实现 Service 避免 Iptables 性能损耗
- 硬件加速方案，卸载所有流表匹配至智能网卡
- fastpath 内核模块,OVS 编译优化,OVN LB 流表裁剪,STT 隧道支持
  https://github.com/kubeovn/kube-ovn/blob/master/docs/performance-tuning.md



2022年功能展望

- 裸金属支持
  - 智能网卡支持，完整 offload ovs/ovn 到网卡
  - 支持 SDN 交换机，纳管基础设施网络设备
- 性能持续优化
  - OVS/OVN 功能裁剪
  - 流表优化
- 数据可视化
  - ebpf 相关监控引入
  - L7 协议栈流量分析
- 多集群互联，支持 MCS-API
  - (12.10 12:10~12:45) 用 Kube-OVN 创建一个跨 Kubernetes 的统一网络平面 - Cheng Chen, PingCAP
- Windows 节点支持

- 即将迎来新的一年，社区不断完善运营方式，为了帮助更多关注Kube-OVN项目和容器网络技术的社区伙伴们，更好地掌握项目动态，方便大家参与到Kube-OVN社区中来，【Kube-OVN社区双周例会】将在每个月的【1日】和【15日】与大家线上碰面！

  届时，Kube-OVN Maintainer与核心开发工程师将会与大家同步项目的Roadmap，最新研发进度，并且针对近期高发问题与大家互动！

运维支持:

- 由灵雀云安排工程师远程或到现场进行实施、安装或升级等，解决客户问题；
- 服务完成后协助客户验证并确保业务的正常运行，同时出具相关的优化报告及建议；
- 发现 Kube-OVN 有严重安全漏洞时，第一时间通知客户；

技术支持:

- 灵雀云专家对问题进行排查、判断，为客户做出解答；
- 遵循客户要求排查故障，并出具故障报告；
- 对于 Kube-OVN 自身导致的问题，为客户提供优化方案；
- 依据客户要求提供Kube-OVN的培训；

联系我们

社区:

官网: https://www.kube-ovn.io/

微信交流群: 备注 “ Kube-OVN”

公众号: kube-ovn

成就一亿技术人

成为技术人交流和成长的家园

用户为本 | 求真求是 | 协作共赢 | 极客精神 | 结果导向

CSDN