Kube-OVN Document

v1.17.0

Kube-OVN Team

2025 Kube-OVN Team

Table of contents

1. Kube-OVN 5
   1.1 What is Kube-OVN? 5
   1.2 Why Kube-OVN? 5
   1.3 CNI □□□□ 5
   1.4 □□□□□OVN/ovn-kubernetes/Kube-OVN 6
   1.5 □□ 7
2. □□□□ 8
   2.1 □□□□ 8
   2.2 □□□□ 10
   2.3 Underlay □□□□ 12
   2.4 □□ CNI □□ 18
   2.5 Talos □□ 23
   2.6 □□□□□ 24
   2.7 □□ 26
3. □□□□ 27
   3.1 □□□□ 27
   3.2 □□□□□□ 35
   3.3 □□□□ 37
   3.4 IP □□□ 40
   3.5 □□□□□ 42
   3.6 EIP □ SNAT □□ 43
   3.7 □□□□ QoS □□ 46
   3.8 □□□□ 49
   3.9 Webhook □□ 51
   3.10 □□□□ 54
   3.11 LoadBalancer □□ Service 60
   3.12 □□□□□□□ 65
   3.13 □□ 69
4. KubeVirt 82
   4.1 VM □□ IP 82
   4.2 □□□□□ 85
   4.3 □□□□ 87
   4.4 □□□ 89
   4.5 □□□□□ 94
   4.6 DHCP 97

2025 Kube-OVN Team

5. VPC ▯▯ 99
5.1 VPC ▯▯ 99
5.2 VPC Egress Gateway 111
5.3 VPC Egress Gateway ▯▯▯▯ 126
5.4 Egress Gateway BGP ▯ EVPN ▯▯ 129
5.5 VPC NAT ▯▯▯▯ Service 133
5.6 VPC QoS 135
5.7 ▯▯▯ VPC ▯▯▯▯▯▯ 138
5.8 ▯▯▯ VPC ▯▯ DNS 143
5.9 SecurityGroup ▯▯ 146
5.10 OVN EIP FIP SNAT DNAT ▯▯ 150
5.11 VPC ▯▯ 159
6. ▯▯▯▯ 161
6.1 kubectl-ko ▯▯ 161
6.2 ▯▯▯▯▯▯▯▯ 168
6.3 ▯▯▯▯▯▯ 169
6.4 ▯▯ ovn-central ▯▯ 171
6.5 OVN ▯▯▯▯▯▯▯▯ 175
6.6 ▯▯▯▯ CIDR 178
6.7 ▯▯ Join ▯▯ CIDR 179
6.8 ▯▯▯▯▯▯ 180
6.9 ▯▯▯▯▯▯ 181
7. ▯▯▯▯ 185
7.1 ▯▯▯▯▯ 185
7.2 ▯▯▯▯ 192
7.3 ▯▯▯▯ FastPath ▯▯ 197
7.4 ▯▯ eBPF ▯▯▯▯▯ TCP ▯▯ 199
7.5 ▯▯ OVN-IC ▯▯▯▯▯▯▯ 203
7.6 ▯▯ Submariner ▯▯▯▯▯▯▯ 209
7.7 Overlay ▯▯▯▯▯▯▯▯▯ 211
7.8 Overlay ▯▯▯▯▯▯▯▯ 213
7.9 BGP ▯▯ 216
7.10 ▯▯ MetalLB ▯ Kube-OVN Underlay ▯▯ 223
7.11 Cilium ▯▯ 227
7.12 Cilium NetworkPolicy ▯▯ 230
7.13 Cilium ▯▯▯▯▯▯ 234
7.14 ▯▯▯▯▯▯ 239
7.15 VIP ▯▯ IP 240

7.16 Mellanox □□ Offload □□ 243
7.17 □□□□□ Offload □□ 252
7.18 □□□□□□ Offload □□ 256
7.19 □□□□□□ Offload □□ 260
7.20 DPDK □□ 263
7.21 OpenStack □□ 267
7.22 □□ IPsec □□□□□□□ 271
7.23 OVN □□□□ 273
7.24 □□ OVN Central 276
7.25 □□□□ DNS □□□ Kube-OVN □□ 278
7.26 □□ VPC NAT □□□□ 280
8. □□□□ 281
8.1 □□□□ 281
8.2 What's Next 284
8.3 □□□□□□□ 287
8.4 □□□□ 297
8.5 Underlay □□□□ 299
8.6 □□ Netfilter □□ 306
8.7 □□□□□□□ 312
8.8 □□□ OVS/OVN □□ 314
8.9 □□□□□□ 316
8.10 Kube-OVN □□□□ 318
8.11 Kube-OVN □□□□ 324
8.12 Annotation □□□□ 355
8.13 □□□□ 360
9. □□□□ 363
9.1 □□□□ 363
9.2 □□□□ 363
9.3 □□ 363
9.4 □□ 363

2025 Kube-OVN Team

1. Kube-OVN

1.1 What is Kube-OVN?

Kube-OVN [illegible] CNCF [illegible] SDN [illegible]

Kube-OVN [illegible] Open Virtual Network[illegible]OVN[illegible] Open vSwitch [illegible] Kubernetes [illegible] OVN [illegible] OVS [illegible] Kubernetes [illegible] SDN [illegible]Kube-OVN [illegible] Kubernetes [illegible] Kubernetes [illegible]

1.2 Why Kube-OVN?

[illegible] Kubernetes [illegible]OVN [illegible] OVS [illegible]

[illegible] Kubernetes [illegible] KubeVirt, [illegible] Kube-OVN [illegible]Kube-OVN [illegible] SDN [illegible]

[illegible]

[illegible] SDN [illegible] Kube-OVN [illegible]

[illegible] OVS/OVN [illegible] SDN [illegible]Kube-OVN [illegible] IP [illegible]/[illegible]Underlay/Overlay [illegible] VPC [illegible]VPC EIP/FIP/SNAT[illegible]VPC [illegible] KubeVirt [illegible]Non-Primary CNI [illegible]BGP [illegible] OVN IPSec[illegible]QoS [illegible] ACL [illegible] ARM/x86 [illegible]

[illegible]

[illegible] Kube-OVN [illegible]

[illegible] eBPF[illegible]DPDK[illegible] Kube-OVN [illegible] OVN [illegible]Kube-OVN [illegible] Pod [illegible]

[illegible] Kube-OVN [illegible] CPU [illegible]

[illegible]

[illegible]Kube-OVN [illegible]

Kube-OVN [illegible] Grafana [illegible] Cilium [illegible] eBPF [illegible] NPM [illegible]

1.3 CNI [illegible]

Kubernetes [illegible] CNI [illegible]

1. [illegible]
2. [illegible]

[illegible] Kube-OVN [illegible] CNI [illegible] Kube-OVN [illegible] Kube-OVN [illegible]

1.3.1 你需要 eBPF 的方案

选择 Cilium 或 Calico eBPF。

Kube-OVN 使用 Open vSwitch 作为数据平面，这是一项相对古老的网络虚拟化技术方案。

1.3.2 你需要 CNI, Ingress, Service Mesh 和 Observability All in One 的方案

选择 Cilium。

Kube-OVN 主要聚焦在 CNI 层面的网络能力，你需要通过组合生态内其他的项目来完成这些功能。

1.3.3 你需要在 OpenShift 上运行

选择 ovn-kubernetes。

在 OpenShift 上使用第三方 CNI 需要适配 Cluster Network Operator 规范，Kube-OVN 目前没有相关计划。此外，第三方网络插件无法获得 RedHat 官方技术支持，而网络作为 Kubernetes 的重要组件，后续的方案设计和故障排查都需要在多个供应商间协调。

1.3.4 在公有云提供的 Kubernetes（EKS/AKS/GKE 等）上运行

选择对应 Kubernetes 提供商的默认 CNI。原因同上。

1.3.5 运行 AI 训练和推理任务

选择 Hostnetwork 或者 host-device 将物理设备直接分配给容器。

AI 任务对网络延迟要求很高，任何额外的容器网络操作都是没有必要的。

1.4 ▯▯▯▯▯OVN/ovn-kubernetes/Kube-OVN

由于以上几个单词的相似性，以及一些简称，在交流沟通中会导致上述几个单词的混淆，在这里做简要澄清：

1.4.1 OVN

OVN 是 Open vSwitch 社区维护的一个虚拟网络控制器，提供了虚拟网络的基础抽象。其本身平台无关，可为 OpenStack, Kubernetes 等多个 CMS 提供网络服务。ovn-kubernetes 和 Kube-OVN 都是依靠 OVN 的网络能力给 Kubernetes 提供网络功能。

1.4.2 ovn-kubernetes

ovn-kubernetes 最早是由 OVN 维护者发起的使用 OVN 为 Kubernetes 提供 CNI 网络能力的项目。目前是 OpenShift 上的默认网络，在 OpenShift 的环境中有着广泛使用，并提供了 UDN, Multihoming, Hardware Acceleration 等高级功能。

1.4.3 Kube-OVN

Kube-OVN 最早诞生是为了解决静态 IP 地址分配，按照 Namespace 分配地址空间，集中式网关等问题而基于 OVN 研发的项目。项目早期借鉴了大量 ovn-kubernetes 的经验和架构设计，例如：使用 annotation 传递 Pod 网络信息、使用 join 网络打通容器网络和主机网络等。目前已经在社区的协助下演化出 Underlay，VPC，KubeVirt 支持等丰富的能力。

PDF Slack Support

2025 Kube-OVN Team

2026⬚10⬚20⬚

2022⬚5⬚20⬚

GitHub  



1.5 ⬚⬚

2. □□□□

2.1 □□□□

Kube-OVN □□□□□ CNI □□□□□□□□□□□□□□□ Kubernetes □□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

2.1.1 □□□□

- Kubernetes >= 1.29□
- Docker >= 1.12.6, Containerd >= 1.3.4□
- □□□□: CentOS 7/8, Ubuntu 16.04/18.04/20.04□
- □□ Linux □□□□□□□□□□□□□□□□□□□ `geneve`, `openvswitch`, `ip_tables` □ `iptable_nat`□Kube-OVN □□□□□□□□□□□

□□□□□

1. □□□□□□□ 3.10.0-862□□□ `netfilter` □□□□ bug □□□ Kube-OVN □□□□□□□□□□□□□□□□□□□□□□□□□ CentOS □□□□□□□□□□□□□□□□□□□□□□□ bug □□ Floating IPs broken after kernel upgrade to Centos/RHEL 7.5 - DNAT not working□
2. Rocky Linux 8.6 □□□ 4.18.0-372.9.1.el8.x86_64 □□ TCP □□□□ TCP connection failed in Rocky Linux 8.6□□□□□□□ 4.18.0-372.13.1.el8_6.x86_64 □□□□□□
3. □□□□□□□ 4.4□□□□□□□ `openvswitch` □□□□□□□□□□□□□□□□□ `openvswitch` □□□□□□□□□□□
4. Geneve □□□□□□□□□ IPv6□□□□ `cat /proc/cmdline` □□□□□□□□□ □□□□ bug □□□ Geneve tunnels don't work when ipv6 is disabled□

2.1.2 □□□□

- □□□□□□□□ IPv6□□□□□□□□□□□ `ipv6.disable=1` □□□□□□□ 0□
- `kube-proxy` □□□□□Kube-OVN □□□□ Service ClusterIP □□□ `kube-apiserver`□
- □□ kubelet □□□□□□□ CNI□□□□□□□□□□□, kubelet □□□□□□□□□□ `--network-plugin=cni --cni-bin-dir=/opt/cni/bin --cni-conf-dir=/etc/cni/net.d`□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□ `/etc/cni/net.d/` □□□

2.1.3 □□□□

□□	□□	□□
ovn-central	6641/tcp	ovn nb db server □□□□
ovn-central	6642/tcp	ovn sb db server □□□□
ovn-central	6643/tcp	ovn northd server □□□□
ovn-central	6644/tcp	ovn raft server □□□□
ovn-ic	6645/tcp	ovn ic nb db server □□□□
ovn-ic	6646/tcp	ovn ic sb db server □□□□
ovs-ovn	Geneve 6081/udp, STT 7471/tcp, Vxlan 4789/udp	□□□□
kube-ovn-controller	10660/tcp	□□□□□□
kube-ovn-daemon	10665/tcp	□□□□□□
kube-ovn-monitor	10661/tcp	□□□□□□

□□□□□□□□ firewalld□□□□□□□□ Packet Forwarding □□ Masquerade□

```
#    Packet Forwarding
firewall-cmd --add-forward --permanent
```

```
#    IPv4 Masquerade
firewall-cmd --add-masquerade --permanent
#   Kube-OVN IPv6/       Masquerade
firewall-cmd --permanent --add-rich-rule 'rule family="ipv6" source address="fd00:10:16::/112" masquerade'

firewall-cmd --reload
```

PDF Slack Support

2025⬚11⬚20⬚

2022⬚5⬚20⬚

GitHub



2.1.4 ⬚⬚

2.2 [illegible]

Kube-OVN 提供了一键安装脚本和 Charts 仓库，可以帮助你快速安装一个高可用，生产就绪的 Kube-OVN 容器网络，默认部署为 Overlay 类型网络。

如果默认网络需要搭建 Underlay/Vlan 网络，请参考 Underlay 网络支持，更多安装配置请参考 安装和配置选项。

安装前请参考准备工作确认环境配置正确。如果想完全删除 Kube-OVN 请参考卸载。

2.2.1 [illegible]

下载安装脚本

我们推荐在生产环境使用稳定的 release 版本，请使用下面的命令下载稳定版本安装脚本：

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/refs/tags/v1.17.0/dist/images/install.sh
```

如果对 master 分支的最新功能感兴趣，请使用下面的命令下载开发版本部署脚本：

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/images/install.sh
```

修改配置参数

使用编辑器打开脚本，并修改下列变量为预期值：

```
REGISTRY="docker.io/kubeovn"           #
VERSION="v1.17.0"                      #     /Tag
POD_CIDR="10.16.0.0/16"                #      CIDR     SVC/NODE/JOIN CIDR
SVC_CIDR="10.96.0.0/12"                #     apiserver   service-cluster-ip-range
JOIN_CIDR="100.64.0.0/16"              # Pod         CIDR     SVC/NODE/POD CIDR
LABEL="node-role.kubernetes.io/control-plane" #    OVN DB
IFACE=""                               #                       Kubernetes    Node IP
TUNNEL_TYPE="geneve"                   #          geneve, vxlan   stt stt        ovs
```

可使用正则表达式来匹配网卡名，例如 `IFACE=enp6s0f0,eth.*` 。

[illegible]

> 脚本执行需要有 root 权限

`bash install.sh`

等待安装完成。

升级

当使用该脚本进行 Kube-OVN 升级时需要注意以下几点：

1. 脚本中的 `[Step 4/6]` 会重启所有容器网络 Pod。在升级过程中，应跳过此步骤或在脚本中将其注释掉，以避免不必要的重启。
2. 重要提示：如果在 Kube-OVN 运行过程中对参数进行过调整，务必在升级前将这些变更更新到脚本中。否则，之前的参数调整将被还原。

2.2.2 Helm Chart [illegible]

由于 Kube-OVN 的安装需要设置一些参数，因此使用 Helm 安装 Kube-OVN 需要按照以下步骤执行。

查看节点 IP 地址

```
# kubectl get node -o wide
NAME                   STATUS     ROLES           AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
kube-ovn-control-plane   NotReady   control-plane   20h   v1.26.0   172.18.0.3    <none>        Ubuntu 22.04.1 LTS   5.10.104-linuxkit   containerd://1.6.9
kube-ovn-worker          NotReady   <none>          20h   v1.26.0   172.18.0.2    <none>        Ubuntu 22.04.1 LTS   5.10.104-linuxkit   containerd://1.6.9
```


2025 Kube-OVN Team

⬚⬚⬚⬚ label

```
# kubectl label node -lbeta.kubernetes.io/os=linux kubernetes.io/os=linux --overwrite
node/kube-ovn-control-plane not labeled
node/kube-ovn-worker not labeled

# kubectl label node -lnode-role.kubernetes.io/control-plane kube-ovn/role=master --overwrite
node/kube-ovn-control-plane labeled

#    label    dpdk        dpdk
# kubectl label node -lovn.kubernetes.io/ovs_dp_type!=userspace ovn.kubernetes.io/ovs_dp_type=kernel --overwrite
node/kube-ovn-control-plane labeled
node/kube-ovn-worker labeled
```

⬚⬚ Helm Repo ⬚⬚

```
# helm repo add kubeovn https://kubeovn.github.io/kube-ovn/
"kubeovn" has been added to your repositories

# helm repo list
NAME            URL
kubeovn         https://kubeovn.github.io/kube-ovn/

# helm repo update kubeovn
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "kubeovn" chart repository
Update Complete. ⎈Happy Helming!⎈

# helm search repo kubeovn
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
kubeovn/kube-ovn        v1.17.0         v1.17.0         Helm chart for Kube-OVN
```

⬚⬚ helm install ⬚⬚ Kube-OVN

Chart ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `values.yaml` ⬚⬚⬚⬚⬚⬚⬚⬚

```
# helm install kube-ovn kubeovn/kube-ovn --wait -n kube-system --version v1.17.0
NAME: kube-ovn
LAST DEPLOYED: Thu Apr 24 08:30:13 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

⬚⬚

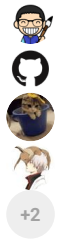
⬚⬚⬚⬚⬚ ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ **Helm** ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `values.yaml` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
helm upgrade -f values.yaml kube-ovn kubeovn/kube-ovn --wait -n kube-system --version v1.17.0
```

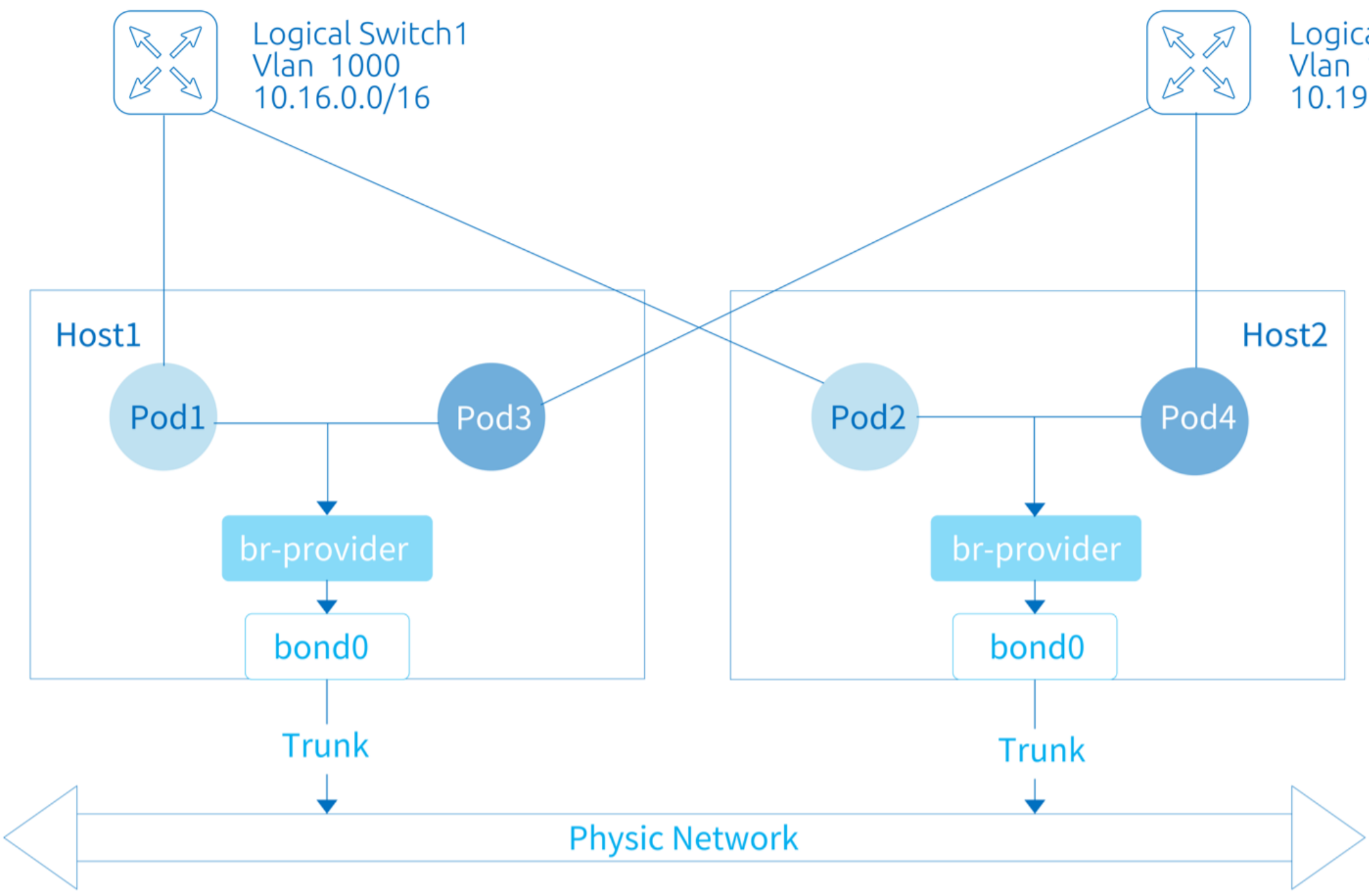
PDF Slack Support

2026⬚4⬚28⬚

2022⬚5⬚20⬚

GitHub 

2025 Kube-OVN Team



2.3 Underlay ⍰⍰⍰⍰

默认情况下 Kube-OVN 的默认子网使用 Geneve 对跨主机流量进行封装，在基础设施之上抽象出一层虚拟的 Overlay 网络。

对于希望容器网络直接使用物理网络地址段情况，可以将 Kube-OVN 的默认子网工作在 Underlay 模式，可以直接给容器分配物理网络中的地址资源，达到更好的性能以及和物理网络的连通性。

2.3.1 ⍰⍰⍰⍰

由于该模式下容器网络直接使用物理网络进行二层包转发，Overlay 模式下的 SNAT/EIP，分布式网关/集中式网关等 L3 功能无法使用，VPC 级别的隔离也无法对 Underlay 子网生效。

2.3.2 ⍰ Macvlan ⍰⍰

Kube-OVN 的 Underlay 模式和 Macvlan 工作模式十分类似，在功能和性能上主要有以下几个区别:

1. 由于 Macvlan 的内核路径更短，并且不需要 OVS 对数据包进行处理，Macvlan 在吞吐量和延迟性能指标上表现会更好。
2. Kube-OVN 通过流表提供了 arp-proxy 功能，可以缓解大规模网络下的 arp 广播风暴风险。
3. 由于 Macvlan 工作在内核底层，会绕过宿主机的 netfilter、Service 和 NetworkPolicy 功能需要额外开发。Kube-OVN 通过 OVS 流表提供了 Service 和 NetworkPolicy 的能力。
4. Kube-OVN 的 Underlay 模式相比 Macvlan 额外提供了地址管理，固定 IP 和 QoS 等功能。

2.3.3 ⍰⍰⍰⍰

在 Underlay 模式下，OVS 将会桥接一个节点网卡到 OVS 网桥，并将数据包直接通过该节点网卡对外发送，L2/L3 层面的转发能力需要依赖底层网络设备。需要预先在底层网络设备配置对应的网关、Vlan 和安全策略等配置。

1. 对于 OpenStack 的 VM 环境，需要将对应网络端口的 `PortSecurity` 关闭。
2. 对于 VMware 的 vSwitch 网络，需要将 `MAC Address Changes`，`Forged Transmits` 和 `Promiscuous Mode Operation` 设置为 `allow`。
3. 对于 VMware 的 NSX-T 网络，无法支持 Underlay 网络模式。
4. 对于 Hyper-V 虚拟化，需要开启虚拟机网卡高级功能中的 `MAC Address Spoofing`。
5. 公有云，例如 AWS、GCE、阿里云等由于不支持用户自定义 Mac 无法支持 Underlay 模式网络，在这种场景下如果想使用 Underlay 推荐使用对应公有云厂商提供的 VPC-CNI。

2025 Kube-OVN Team

6. [illegible] Linux Bridge [illegible]

[illegible] Kube-OVN Underlay [illegible] Underlay [illegible]

[illegible] Underlay [illegible]

[illegible]Kube-OVN [illegible] Mac [illegible]IP [illegible] MTU [illegible] OVS Bridge[illegible] Underlay [illegible]OVS Bridge [illegible] `br-PROVIDER_NAME` [illegible] `PROVIDER_NAME` [illegible] `ProviderNetwork` [illegible] provider[illegible]

1. [illegible] OVS Bridge [illegible] NetworkManager [illegible] DHCP [illegible]
2. [illegible] OVS Bridge[illegible] OVS [illegible] Kube-OVN [illegible] OVS [illegible] Underlay [illegible]

2.3.4 [illegible]

[illegible] Underlay [illegible] Pod [illegible] Underlay [illegible]

[illegible]

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/dist/images/install.sh
```

[illegible]

```
ENABLE_ARP_DETECT_IP_CONFLICT #            vlan    arp
NETWORK_TYPE                  #     vlan
VLAN_INTERFACE_NAME           #                    eth1
VLAN_ID                       #        VLAN Tag      0     VLAN
POD_CIDR                      #        CIDR     192.168.1.0/24
POD_GATEWAY                   #             192.168.1.1
EXCLUDE_IPS                   #                  IP       192.168.1.1..192.168.1.100
ENABLE_LB                     #    Underlay        Service       true
EXCHANGE_LINK_NAME            #        provider-network   OVS               false
LS_DNAT_MOD_DL_DST            # DNAT      MAC            Service         true
```

[illegible]

```
bash install.sh
```

2.3.5 [illegible] CRD [illegible] Underlay [illegible]

[illegible] Underlay [illegible] Pod [illegible] `ProviderNetwork` [illegible] `Vlan` [illegible] `Subnet` [illegible]

[illegible] ProviderNetwork

ProviderNetwork [illegible] Underlay [illegible]

[illegible] ProviderNetwork [illegible]:

```yaml
apiVersion: kubeovn.io/v1
kind: ProviderNetwork
metadata:
  name: net1
spec:
  defaultInterface: eth1
  customInterfaces:
    - interface: eth2
      nodes:
        - node1
  nodeSelector:
    matchLabels:
      kubernetes.io/arch: amd64
      network-type: underlay
    matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
          - node1
          - node2
```


2025 Kube-OVN Team

注意：**ProviderNetwork** 资源名称长度不得超过 **12** 个字符。

- `defaultInterface`：为默认使用的节点网卡名称。该网卡可以是物理网卡也可以是 Bond 网卡或者 Vlan 子接口。ProviderNetwork 创建成功后，各节点（除 excludeNodes 外）中会创建名为 br-net1（格式为 `br-NAME`）的 OVS 网桥，并将指定的节点网卡桥接至此网桥。
- `customInterfaces`：为可选项，可针对特定节点指定需要使用的网卡。
- `nodeSelector`：可选项，用于基于节点标签选择需要创建 OVS 网桥的节点。支持 `matchLabels` 和 `matchExpressions` 两种选择方式。
- `excludeNodes`：可选项，用于指定不桥接网卡的节点。该列表中的节点会被添加 `net1.provider-network.ovn.kubernetes.io/exclude=true` 标签。注意：一旦使用了 **`nodeSelector`**，**`excludeNodes`** 将不再生效，推荐只使用 **`nodeSelector`**。
- `autoCreateVlanSubinterfaces`：可选项，默认为 `false`。设置为 `true` 时，系统会自动在指定的物理接口上创建 VLAN 子接口，支持 `<interface>.<vlanid>` 格式的接口命名。

其它节点会被添加如下标签：

Key	Value	描述
net1.provider-network.ovn.kubernetes.io/ready	true	节点中的桥接工作已完成，ProviderNetwork 在节点中可用
net1.provider-network.ovn.kubernetes.io/interface	eth1	节点中被桥接的网卡的名称
net1.provider-network.ovn.kubernetes.io/mtu	1500	节点中被桥接的网卡的 MTU

> 如果节点网卡上已经配置了 IP，则 IP 地址和网卡上的路由会被转移至对应的 OVS 网桥。

自动创建 VLAN 子接口

当 ProviderNetwork 中启用 `autoCreateVlanSubinterfaces: true` 时，系统会自动创建 VLAN 子接口，简化多 VLAN 环境下的网络配置。

示例

```yaml
apiVersion: kubeovn.io/v1
kind: ProviderNetwork
metadata:
  name: vlan-provider
spec:
  defaultInterface: eth0.341
  autoCreateVlanSubinterfaces: true
  customInterfaces:
    - interface: eth2.341
      nodes:
        - worker-node-1
        - worker-node-2
```

说明

系统会自动创建 `<interface>.<vlanid>` 格式的 VLAN 子接口，无需手动配置。适用于多 VLAN 环境和大规模部署，可以显著简化网络管理。删除 ProviderNetwork 时会自动清理相关接口。

保留已有 VLAN 接口

当节点上已经有预配置的 VLAN 子接口时，可以通过 `preserveVlanInterfaces` 和 `vlanInterfaces` 来避免 OVS 网桥接管过程中丢失这些接口。

- `preserveVlanInterfaces`：启用后，系统会自动检测节点上属于 `defaultInterface` 的 VLAN 子接口（格式 `<interface>.<vlanid>`），并将其保留在 OVS 网桥之外。
- `vlanInterfaces`：显式指定要保留的 VLAN 接口名称列表（如 `eth0.10`、`bond0.20`）。

```yaml
apiVersion: kubeovn.io/v1
kind: ProviderNetwork
metadata:
  name: net1
spec:
  defaultInterface: eth1
  preserveVlanInterfaces: true
  vlanInterfaces:
  - eth1.100
  - eth1.200
```

创建 VLAN

Vlan 提供了将 Vlan Tag 和 ProviderNetwork 进行绑定的能力。

2025 Kube-OVN Team

创建如下 VLAN 并应用:

```
apiVersion: kubeovn.io/v1
kind: Vlan
metadata:
  name: vlan1
spec:
  id: 0
  provider: net1
```

- `id`：为 VLAN ID/Tag，Kube-OVN 会对对该 Vlan 下的流量增加 Vlan 标签，为 0 时不增加任何标签，vlan 标签作用在一个 localnet 端口。
- `provider`：为需要使用的 ProviderNetwork 资源的名称。多个 VLAN 可以引用同一个 ProviderNetwork。

创建 **Subnet**

将 Vlan 和一个子网绑定，如下所示:

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: subnet1
spec:
  protocol: IPv4
  cidrBlock: 172.17.0.0/16
  gateway: 172.17.0.1
  vlan: vlan1
  disableGatewayCheck: false
```

- `vlan`：需要使用的 VLAN 名称，多个 Subnet 可以引用同一个 VLAN。
- `disableGatewayCheck`：若所在 Underlay 网络网关不存在，可将该字段设置为 `true`，关闭网关探测。

2.3.6 容器创建

可按正常容器创建方式进行创建，查看容器 IP 是否在规定范围内，以及容器是否可以和物理网络互通。

如有固定 IP 需求，可参考 Pod 固定 IP 和 Mac

2.3.7 使[illegible]逻辑网[illegible]

对于物理网络不存在网关的情况，Kube-OVN 支持在 Underlay 模式的子网中配置使用逻辑网关。若要使用此功能，设置子网的 `spec.logicalGateway` 为 `true` 即可:

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: subnet1
spec:
  protocol: IPv4
  cidrBlock: 172.17.0.0/16
  gateway: 172.17.0.1
  vlan: vlan1
  logicalGateway: true
```

开启此功能后，Pod 不使用外部网关，而是使用 Kube-OVN 创建的逻辑路由器（Logical Router）对于跨网段通信进行转发。

2.3.8 Underlay 和 Overlay 网络互通

如果一个集群同时存在 Underlay 和 Overlay 子网，默认情况下 Overlay 子网下的 Pod 可以通过网关以 NAT 的方式访问 Underlay 子网下的 Pod IP。在 Underlay 子网的 Pod 看来 Overlay 子网的地址是一个外部的地址，需要通过底层物理设备去转发，但底层物理设备并不清楚 Overlay 子网的地址无法进行转发。因此 Underlay 子网下的 Pod 无法通过 Pod IP 直接访问 Overlay 子网的 Pod。

如果需要 Underlay 和 Overlay 互通需要将子网的 `u2oInterconnection` 设置为 `true`，在这个情况下 Kube-OVN 会额外使用一个 Underlay IP 将 Underlay 子网和 `ovn-cluster` 逻辑路由器连接，并设置对应的路由规则实现互通。和逻辑网关不同，该方案只会连接 Kube-OVN 内部的 Underlay 和 Overlay 子网，其他访问外网的流量还是会通过物理网关进行转发。

> **Warning**
>
> 若多个不同 Vlan 下的 Underlay 存在子网地址重叠，同时接入后会导致逻辑路由器路由异常，因此当 Underlay 子网地址重叠时不能同时开启 u2o 功能。

2025 Kube-OVN Team

指定逻辑网关 IP

开启互通功能后，会随机从 subnet 内的取一个 IP 作为逻辑网关，如果需要指定 Underlay Subnet 的逻辑网关可以指定字段 `u2oInterconnectionIP`。

指定 Underlay Subnet 连接的自定义 VPC

默认情况下 Underlay Subnet 会和默认 VPC 上的 Overlay Subnet 互通，如果要指定和某个 VPC 互通，在 `u2oInterconnection` 设置为 `true` 后，指定 `subnet.spec.vpc` 字段为该 VPC 名字即可。

2.3.9 ⬚⬚⬚⬚

如果您使用的节点网卡上配置有 IP 地址，且操作系统通过 Netplan 配置网络（如 Ubuntu），建议您将 Netplan 的 renderer 设置为 NetworkManager，并为节点网卡配置静态 IP 地址（关闭 DHCP）：

```
network:
  renderer: NetworkManager
  ethernets:
    eth0:
      dhcp4: no
      addresses:
        - 172.16.143.129/24
  version: 2
```

如果您要修改网卡的 IP 或路由配置，需要在修改 netplan 配置后执行以下命令：

```
netplan generate

nmcli connection reload netplan-eth0
nmcli device set eth0 managed yes
```

执行以上命令后，Kube-OVN 会将网卡上的 IP 及路由重新转移至 OVS 网桥。

如果您使用的操作系统通过 NetworkManager 管理网络（如 CentOS），在修改网卡配置后需要执行以下命令：

```
nmcli connection reload eth0
nmcli device set eth0 managed yes
nmcli -t -f GENERAL.STATE device show eth0 | grep -qw unmanaged || nmcli device reapply eth0
```

注意：节点网卡配置的动态修改仅支持 IP 和路由，不支持 MAC 地址的修改。

2.3.10 ⬚⬚⬚⬚

物理网络开启 hairpin 时 Pod 网络异常

当物理网络开启 hairpin 或类似行为时，可能出现创建 Pod 时网关检查失败、Pod 网络通信异常等问题。这是因为 OVS 网桥默认的 MAC 学习功能不支持这种网络环境。

要解决此问题，需要关闭 hairpin（或修改物理网络的相关配置），或更新 Kube-OVN 版本。

Pod 数量较多时新建 Pod 网关检查失败

若同一个节点上运行的 Pod 数量较多（大于 300），可能会出现 ARP 广播包的 OVS 流表 resubmit 次数超过上限导致丢包的现象：

```
2022-11-13T08:43:46.782Z|00222|ofproto_dpif_upcall(handler5)|WARN|Flow:
arp,in_port=331,vlan_tci=0x0000,dl_src=00:00:00:25:eb:39,dl_dst=ff:ff:ff:ff:ff:ff,arp_spa=10.213.131.240,arp_tpa=10.213.159.254,arp_op=1,arp_sha=00:00:00:25:e
b:39,arp_tha=ff:ff:ff:ff:ff:ff

bridge("br-int")
----------------
 0. No match.
     >>>> received packet on unknown port 331 <<<<
    drop

Final flow: unchanged
Megaflow: recirc_id=0,eth,arp,in_port=331,dl_src=00:00:00:25:eb:39
Datapath actions: drop
2022-11-13T08:44:34.077Z|00224|ofproto_dpif_xlate(handler5)|WARN|over 4096 resubmit actions on bridge br-int while processing
arp,in_port=13483,vlan_tci=0x0000,dl_src=00:00:00:59:ef:13,dl_dst=ff:ff:ff:ff:ff:ff,arp_spa=10.213.152.3,arp_tpa=10.213.159.254,arp_op=1,arp_sha=00:00:00:59:e
f:13,arp_tha=ff:ff:ff:ff:ff:ff
```

要解决此问题，可修改 OVN NB 选项 `bcast_arp_req_flood` 为 `false`：

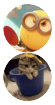
```
kubectl ko nbctl set NB_Global . options:bcast_arp_req_flood=false
```


2025 Kube-OVN Team

PDF Slack Support

2026⊠11⊠5⊠

2022⊠5⊠20⊠



GitHub



+2

2.3.11 ⊠⊠

2025 Kube-OVN Team

2.4 ▯▯ CNI ▯▯

本文档介绍如何将 Kube-OVN 作为附属 CNI 与其他主 CNI 插件（如 Cilium、Calico 或 Flannel）一起使用。

2.4.1 ▯▯

在非主 CNI 模式下，Kube-OVN 作为辅助 CNI 插件，为 Pod 提供额外的网络接口，而另一个 CNI 处理主网络接口（eth0）。这样可以：

- 在现有 CNI 基础上使用 Kube-OVN 的高级网络功能（VPC、子网、安全组）
- 实现网络分段和多租户
- 为 Pod 提供多个网络接口以实现不同用途
- 利用 Kube-OVN 的 VPC NAT 网关功能为次要网络提供外部连接

2.4.2 ▯▯▯▯

必需组件

1. 主 **CNI**：任何与 Kubernetes 兼容的 CNI（Cilium、Calico、Flannel 等）
2. **Multus CNI**：多网络支持所必需
3. **Kube-OVN**：配置为非主模式

安装顺序

1. 安装并配置主 CNI
2. 安装 Multus CNI
3. 以非主模式安装 Kube-OVN

2.4.3 ▯▯

使用 **Helm Chart v2**

```
# values.yaml
cni:
  nonPrimaryCNI: true
```

使用 Helm 安装：

```
helm install kube-ovn ./charts/kube-ovn-v2 \
  --namespace kube-system \
  --set cni.nonPrimaryCNI=true
```

使用 **Helm Chart v1**

```
# values.yaml
cni_conf:
  NON_PRIMARY_CNI: true
```

使用 Helm 安装：

```
helm install kube-ovn ./charts/kube-ovn \
  --namespace kube-system \
  --set cni_conf.NON_PRIMARY_CNI=true
```

手动安装

在 `kube-ovn-controller` Deployment 与 `kube-ovn-cni` DaemonSet 中均需添加以下标志（两者都需打开，否则 Pod 网络分配仍会按主网络模式执行）：

2025 Kube-OVN Team

```
# kube-ovn-controller Deployment
containers:
- name: kube-ovn-controller
  args:
  - --non-primary-cni-mode=true
```

```
# kube-ovn-cni DaemonSet
containers:
- name: kube-ovn-cni
  args:
  - --non-primary-cni-mode=true
```

2.4.4 ▯▯

▯▯▯▯▯▯▯**NADs**▯

▯▯ NAD ▯▯▯▯▯▯▯▯▯▯▯▯

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: kube-ovn-vpc-network
  namespace: default
spec:
  config: |
    {
      "cniVersion": "0.3.1",
      "type": "kube-ovn",
      "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
      "provider": "kube-ovn-vpc-network.default.ovn"
    }
```

VPC ▯▯▯▯▯

▯▯▯▯▯▯▯ VPC ▯▯▯▯▯▯

```
apiVersion: kubeovn.io/v1
kind: Vpc
metadata:
  name: vpc-secondary
spec:
  namespaces:
  - default
---
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: subnet-secondary
spec:
  vpc: vpc-secondary
  cidr: "10.100.0.0/16"
  gateway: "10.100.0.1"
  provider: kube-ovn-vpc-network.default.ovn
```

Pod ▯▯

▯ Pod ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-network-pod
  annotations:
    k8s.v1.cni.cncf.io/networks: default/kube-ovn-vpc-network
spec:
  containers:
  - name: app
    image: nginx
```

2.4.5 ▯▯▯▯▯▯

▯▯▯▯**eth0**▯

- ▯▯ CNI▯Cilium▯Calico ▯▯▯▯
- ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

2025 Kube-OVN Team

- ███ Kubernetes ██████████

██████**net1**、**net2...**、

- ██ Kube-OVN ██
- ██████████
- ████████ VPC ███
- ██ Kube-OVN ████████QoS ██ NAT

2.4.6 ████

1. ████

████████████████████████

```
#    Pod
apiVersion: v1
kind: Pod
metadata:
  name: frontend
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
        {"name": "default/public-network"},
        {"name": "default/internal-network"}
      ]
```

2. ███

███████████

```
#    A Pod
apiVersion: v1
kind: Pod
metadata:
  name: tenant-a-app
  annotations:
    k8s.v1.cni.cncf.io/networks: default/tenant-a-network

---
#    B Pod
apiVersion: v1
kind: Pod
metadata:
  name: tenant-b-app
  annotations:
    k8s.v1.cni.cncf.io/networks: default/tenant-b-network
```

3. VPC NAT ██

██████████████

```
apiVersion: kubeovn.io/v1
kind: VpcNatGateway
metadata:
  name: vpc-nat-gw
spec:
  vpc: vpc-secondary
  subnet: subnet-secondary
  lanIp: "10.100.0.254"
```

2.4.7 ████

████████

Pod ████████████████████

```
metadata:
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
```


2025 Kube-OVN Team

```
        {"name": "default/kube-ovn-network-1", "interface": "net1"},
        {"name": "default/kube-ovn-network-2", "interface": "net2"}
      ]
```

IP [illegible]

[illegible] IP [illegible]

```
#    IP
metadata:
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [{
        "name": "default/kube-ovn-network",
        "ips": ["10.100.0.100/16"]
      }]
```

[illegible]**QoS**[illegible]

[illegible] QoS [illegible]

```
metadata:
  annotations:
    ovn.kubernetes.io/ingress_rate: "1000"
    ovn.kubernetes.io/egress_rate: "1000"
```

2.4.8 [illegible]

[illegible]

1. **Pod** [illegible] - [illegible] NAD [illegible] - [illegible] kube-ovn-cni [illegible] - [illegible] Multus [illegible]
2. [illegible] - [illegible] VPC [illegible] - [illegible] Pod [illegible] - [illegible]
3. **IP** [illegible] - [illegible] CIDR [illegible] - [illegible] IP [illegible] - [illegible] IPAM [illegible]

[illegible]

```
#    Pod
kubectl get pod <pod-name> -o yaml | grep -A 10 "networks-status"

#    Pod
kubectl exec <pod-name> -- ip addr show

#
kubectl exec <pod-name> -- ip route show

#    kube-ovn-cni
kubectl logs -n kube-system daemonset/kube-ovn-cni
```

2.4.9 [illegible]

1. [illegible] Kubernetes [illegible]
2. [illegible]
3. **DNS**[illegible]Pod DNS [illegible]

2.4.10 [illegible]

1. [illegible] IP [illegible]
2. [illegible]
3. [illegible]
4. [illegible]
5. [illegible]

2025 Kube-OVN Team

2.4.11 ▯▯

- Multus CNI ▯▯
- ▯▯▯▯▯▯▯▯▯
- Kube-OVN ▯▯

PDF | Slack | Support

2026▯4▯28▯

2025▯10▯30▯

GitHub 

2.4.12 ▯▯

2.5 Talos ⬚⬚

Talos Linux ⬚⬚ Kubernetes ⬚⬚⬚⬚⬚ Linux ⬚⬚⬚⬚

2.5.1 ⬚⬚ Helm Chart ⬚⬚ Kube-OVN

⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Talos Linux ⬚⬚⬚⬚⬚ Kube-OVN⬚

```
helm install kube-ovn kubeovn/kube-ovn --wait \
    -n kube-system \
    --version v1.17.0 \
    --set OVN_DIR=/var/lib/ovn \
    --set OPENVSWITCH_DIR=/var/lib/openvswitch \
    --set DISABLE_MODULES_MANAGEMENT=true \
    --set cni_conf.MOUNT_LOCAL_BIN_DIR=false
```

⬚⬚⬚⬚⬚⬚⬚ Underlay ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Helm ⬚⬚⬚⬚⬚⬚⬚ Chart ⬚⬚⬚⬚⬚⬚⬚⬚

```
helm install kubeovn kubeovn/kube-ovn --wait \
    -n kube-system \
    --version v1.17.0 \
    --set OVN_DIR=/var/lib/ovn \
    --set OPENVSWITCH_DIR=/var/lib/openvswitch \
    --set DISABLE_MODULES_MANAGEMENT=true \
    --set cni_conf.MOUNT_LOCAL_BIN_DIR=false \
    --set networking.NETWORK_TYPE=vlan \
    --set networking.vlan.VLAN_INTERFACE_NAME=enp0s5f1 \
    --set networking.vlan.VLAN_ID=0 \    # VLAN_ID=0      VLAN tag    untagged      chart      100
    --set networking.NET_STACK=ipv4 \
    --set-json networking.EXCLUDE_IPS='"172.99.99.11..172.99.99.99"' \
    --set-json ipv4.POD_CIDR='"172.99.99.0/24"' \
    --set-json ipv4.POD_GATEWAY='"172.99.99.1"'
```

Note

VLAN⬚Bond⬚Bridge ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Underlay ⬚⬚⬚⬚⬚⬚⬚⬚Underlay ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Talos ⬚⬚⬚⬚⬚⬚ `ignore=true` ⬚⬚⬚⬚⬚⬚

```
machine:
  network:
    interfaces:
      - interface: enp0s5f1
        ignore: true
```

PDF Slack Support

2026⬚4⬚28⬚

2025⬚4⬚16⬚

GitHub



2.5.2 ⬚⬚

2025 Kube-OVN Team

2.6 [illegible]

Kube-OVN 是否生产就绪？如果你问出这个问题，那就说明你尚未生产就绪。有的人可以轻松将 pre-v0.0.1 版本运行到上千节点的关键业务生产环境还觉得没什么压力，有的人只是在自己的开发环境部署最新版本就已经焦头烂额。生产就绪更多的是对使用者的要求而不是软件本身的标准。不要轻信各种市场营销术语，也不要相信别人的案例，他们都不会来解决你生产环境中的问题，你自己要准备好面对这一切。这篇文档将会介绍我们认为在上生产环境前的一些必要准备工作，帮助你在生产环境中更有信心且自如地使用 Kube-OVN。

2.6.1 下载完整制品

开源软件可能会突然停止版本发布、限制使用方式、删除代码仓库、删除镜像或删除文档：

- Linkerd 停止发行稳定版本
- Elasticsearch 停止开源镜像发布
- EMQX 社区用户生产环境仅允单节点部署
- KubeSphere 停止开源下载

你需要一份离线完整的代码，镜像和文档，并保证你能够在自己环境里重新构建完整的镜像。请使用备份脚本下载对应版本的代码、镜像和文档。并参考开发和贡献指导学习如何从代码构建镜像。

2.6.2 OpenVswitch/OVN 学习

Kube-OVN 底层的数据平面大部分由 OpenVswitch 和 OVN 实现，了解底层原理会让你对复杂问题更有信心。

- OVN 参考文档
- OpenVswitch 参考文档

其中我们推荐 ovn-architecture 文档必读，其他文档只需了解大概框架，可在实际使用中再去了解细节内容。

2.6.3 Kube-OVN 文档学习

我们建议详细阅读技术参考和运维指南部分下的全部文档。技术参考部分可以帮助你建立对 Kube-OVN 整体架构，技术选型的一些概念，运维指南里包含了我们认为日常可能会使用到的复杂操作和一系列工具的使用。

其中运维指南里的操作可在开发测试环境提前进行演练，包括灾备，节点切换，地址变更等复杂操作，平日的操作可以增强你在生产环境使用的信心，也更加熟悉和了解 Kube-OVN 的原理。其中 kubectl 插件使用 bash 编写了大量简化日常操作的命令，但我们强烈推荐了解每个命令背后的具体执行过程，这可以帮助你对 Kube-OVN 的整体架构有更深刻的理解。

2.6.4 了解具体功能

Kube-OVN 中包含了大量功能，并且还在快速迭代。针对你使用的功能我们建议你在了解如何使用时还要了解其局限性和背后的原理，我们也会在每个功能的文档中不断完善相关内容。

同时我们建议你构建一个自己所使用到功能的测试集合，在 Kube-OVN 新版本发布后可以快速验证新版本对自己的影响。如果你的功能使用场景目前没有包含在 Kube-OVN 的 E2E 测试代码中，我们欢迎你能贡献相关的测试，帮助项目更好的成长。

2.6.5 建设监控体系

大部分应用侧的故障都长得像网络问题，实际上只有很少部分是真正的网络问题，但是很多查不出来的问题都被归结为网络问题，这会导致网络侧承担不必要的压力。我们建议和应用侧共同建设监控体系，将应用侧和网络侧的监控、日志、告警等数据打通，当故障发生时可以联合所有的数据进行故障分析，确定责任边界。在 Kube-OVN 社区中我们接触过使用 OpenTelemetry 打通应用侧链路，使用 DeepFlow 打通底层网络链路的案例，也推荐你根据自己的情况探索合适的监控体系。

2.6.6 正确反馈问题

Kube-OVN 社区的维护者无法 7*24 小时实时响应问题，我们更倾向于使用 Github Issue 的方式进行异步沟通。为了提升效率，请尽可能详细的提供问题出现的环境、复现步骤以及你的分析，这可以帮助我们更好的了解问题，避免多轮对话迭代。

我们在 Github Issue 中提供了 AI 机器人会对问题进行初步的分析，你也可以使用自己熟悉的 AI 工具对代码仓库提问，有时可能会得到超出预期的结果。

PDF Slack Support

2025 Kube-OVN Team

2025⽉9⽉22⽇

2025⽉8⽉29⽇

GitHub 

2.6.7 ⽹⽹

2.7 ⬚⬚

⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚ OVS ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ ⬚⬚⬚⬚ issue ⬚⬚⬚⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

2.7.1 ⬚⬚⬚ Kubernetes ⬚⬚⬚⬚⬚⬚

⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

Script Uninstall **Helm Uninstall**

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/dist/images/cleanup.sh
bash cleanup.sh

helm uninstall kube-ovn -n kube-system
```

2.7.2 ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ ovsdb ⬚⬚ openvswitch ⬚⬚⬚⬚⬚⬚

```
rm -rf /var/run/openvswitch
rm -rf /var/run/ovn
rm -rf /etc/origin/openvswitch/
rm -rf /etc/origin/ovn/
rm -rf /etc/cni/net.d/01-kube-ovn.conflist
#       v1.10         00-kube-ovn.conflist
rm -rf /etc/cni/net.d/00-kube-ovn.conflist
rm -rf /var/log/openvswitch
rm -rf /var/log/ovn
rm -fr /var/log/kube-ovn
```

2.7.3 ⬚⬚⬚⬚

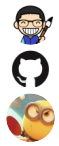
⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚iptable/ipset ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
reboot
```

PDF Slack Support

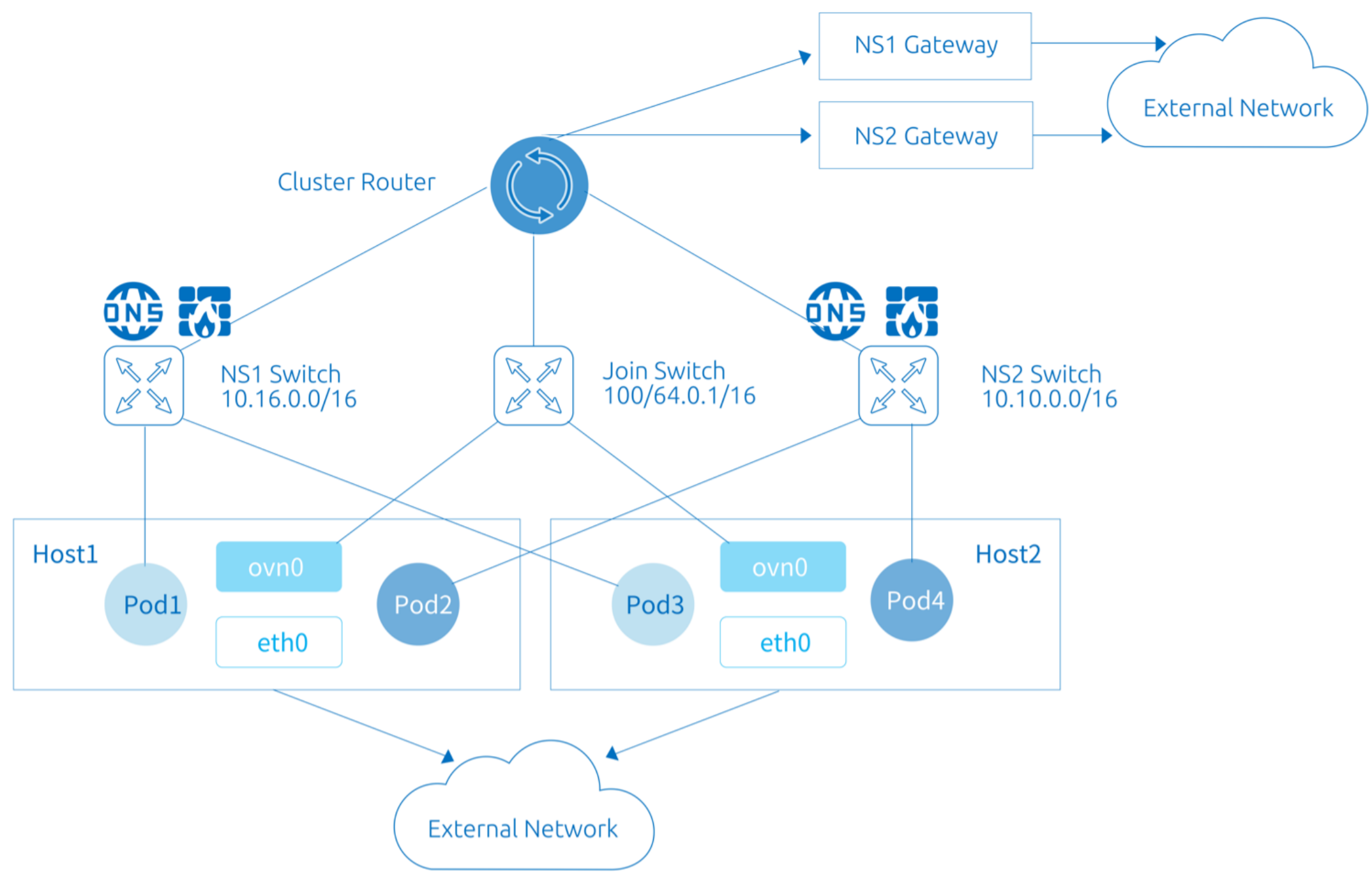
2026⬚4⬚28⬚

2022⬚5⬚24⬚

GitHub 

2.7.4 ⬚⬚

2025 Kube-OVN Team



3. [illegible]

3.1 [illegible]

子网是 Kube-OVN 中的一个核心概念和基本使用单元，Kube-OVN 会以子网来组织 IP 和网络配置，每个 Namespace 可以归属于特定的子网， Namespace 下的 Pod 会自动从所属的子网中获取 IP 并共享子网的网络配置（CIDR，网关类型，访问控制，NAT 控制等）。

和其他 CNI 的每个节点绑定一个子网的实现不同，在 Kube-OVN 中子网为一个全局的虚拟网络配置，同一个子网的地址可以分布在任意一个节点上。

注意：同一 VPC 下的不同子网不能包含同样的 IP，以及基于 VPC peering 或者 VPN 互相打通的不同子网不能包含同样的 IP。

Overlay 和 Underlay 的子网在使用和配置上存在一些差异，本文档将会介绍不同类型子网的一些共同配置和差异化功能。

3.1.1 [illegible]

为了方便用户的快速上手使用，Kube-OVN 内置了一个默认子网，所有未显式声明子网归属的 Namespace 会自动从默认子网中分配 IP，并使用默认子网的网络信息。该子网的配置为安装时指定，可以参考内置网络设置，如果要在安装后修改默认网络的 CIDR 请参考修改默认网络。

在 Overlay 模式下，默认子网使用了分布式网关并对出网流量进行 NAT 转换，其行为和 Flannel 的默认行为基本一致，用户无需额外的配置即可使用到大部分的网络功能。

在 Underlay 模式下，默认子网使用物理网关作为出网网关，并开启 arping 检查网络连通性。

查看默认子网

默认子网 spec 中的 default 字段为 true，一个集群下只有一个默认子网，默认名为 `ovn-default`。

查看默认子网：

```
# kubectl get subnet ovn-default -o yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  creationTimestamp: "2019-08-06T09:33:43Z"
  generation: 1
  name: ovn-default
  resourceVersion: "1571334"
```


2025 Kube-OVN Team

```
    selfLink: /apis/kubeovn.io/v1/subnets/ovn-default
    uid: 7e2451f8-fb44-4f7f-b3e0-cfd27f6fd5d6
  spec:
    cidrBlock: 10.16.0.0/16
    default: true
    excludeIps:
    - 10.16.0.1
    gateway: 10.16.0.1
    gatewayType: distributed
    natOutgoing: true
    private: false
    protocol: IPv4
```

3.1.2 Join □□

□ Kubernetes □□□□□□□□□ Node □□□□□□ Pod □□□□□ □□□ Overlay □□□□□□□□□□□□ Kube-OVN □□□□□ `join` □□□ □□□□ Node □□□□□□□□□□□ ovn0 □□ `join` □□□□□□□□□□□ □ Pod □□□□□□□□□

□□ Pods □ Nodes □□□□□□□□□□□ ovn0 □□□□□Node □□ Pod □□□ ovn0 □□□□□□□□□□□□□□□ ovn0 □□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ join □□□ CIDR □□□□□ Join □□

□□ Join □□

> □□□□□□□□□□□ Pod □□□□ `hostport` □□□□□□ `externalTrafficPolicy: Local` □ NodePort □□ Service □□□□□

□□□□□□□ `join` □□□□□□□□ CIDR □□□□□□□□□□□□□□

```
# kubectl get subnet join -o yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  creationTimestamp: "2019-08-06T09:33:43Z"
  generation: 1
  name: join
  resourceVersion: "1571333"
  selfLink: /apis/kubeovn.io/v1/subnets/join
  uid: 9c744810-c678-4d50-8a7d-b8ec12ef91b8
spec:
  cidrBlock: 100.64.0.0/16
  default: false
  excludeIps:
  - 100.64.0.1
  gateway: 100.64.0.1
  gatewayNode: ""
  gatewayType: ""
  natOutgoing: false
  private: false
  protocol: IPv4
```

□ Node □□□□ ovn0 □□□

```
# ifconfig ovn0
ovn0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1420
        inet 100.64.0.4  netmask 255.255.0.0  broadcast 100.64.255.255
        inet6 fe80::800:ff:fe40:5  prefixlen 64  scopeid 0x20<link>
        ether 0a:00:00:40:00:05  txqueuelen 1000  (Ethernet)
        RX packets 18  bytes 1428 (1.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 1810 (1.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3.1.3 □□□□□□□□

□□□□□□□□□□□□□□□□□□□ Namespace □□□□□□□□□□□□□□□□□□□□□□□□

□□□□

```
cat <<EOF | kubectl create -f -
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: subnet1
spec:
  protocol: IPv4
  cidrBlock: 10.66.0.0/16
  excludeIps:
```


2025 Kube-OVN Team

```
  - 10.66.0.1..10.66.0.10
  - 10.66.0.101..10.66.0.151
  gateway: 10.66.0.1
  gatewayType: distributed
  natOutgoing: true
  routeTable: ""
  namespaces:
  - ns1
  - ns2
EOF
```

- `cidrBlock` : [illegible] CIDR [illegible] VPC [illegible] Subnet CIDR [illegible]
- `excludeIps` : [illegible] IP [illegible] Underlay [illegible]
- `gateway` [illegible]Overlay [illegible] Kube-OVN [illegible]Underlay [illegible]
- `namespaces` : [illegible] Namespace [illegible] Namespace [illegible] Pod [illegible]
- `routeTable` : [illegible]

[illegible]

```
# kubectl create ns ns1
namespace/ns1 created

# kubectl run nginx --image=docker.io/library/nginx:alpine -n ns1
deployment.apps/nginx created

# kubectl get pod -n ns1 -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP           NODE    NOMINATED NODE   READINESS GATES
nginx-74d5899f46-n8wtg 1/1     Running   0          10s   10.66.0.11   node1   <none>           <none>
```

Workload [illegible]

[illegible] Pod [illegible] Namespace [illegible] IP [illegible] Namespace [illegible] Workload [illegible] Pod [illegible] Annotation `ovn.kubernetes.io/logical_switch` [illegible]

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: another-subnet
  annotations:
    ovn.kubernetes.io/logical_switch: subnet1
spec:
  containers:
  - name: another-subnet
    image: docker.io/library/nginx:alpine
```

[illegible] Workload [illegible] Deployment[illegible]StatefulSet [illegible] `ovn.kubernetes.io/logical_switch` Annotation [illegible] `spec.template.metadata.annotations` [illegible]

3.1.4 Overlay [illegible]

> [illegible] VPC [illegible] Overlay [illegible] Underlay [illegible] VPC [illegible]

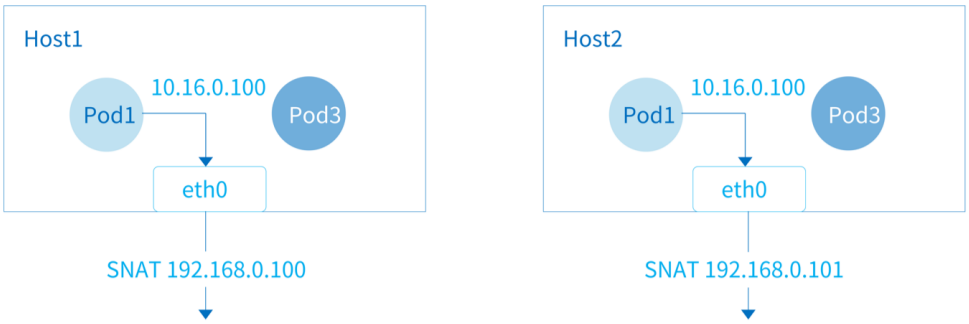
Overlay [illegible] Pod [illegible]Kube-OVN [illegible]

[illegible] `natOutgoing` [illegible] Pod [illegible] snat[illegible]

[illegible]

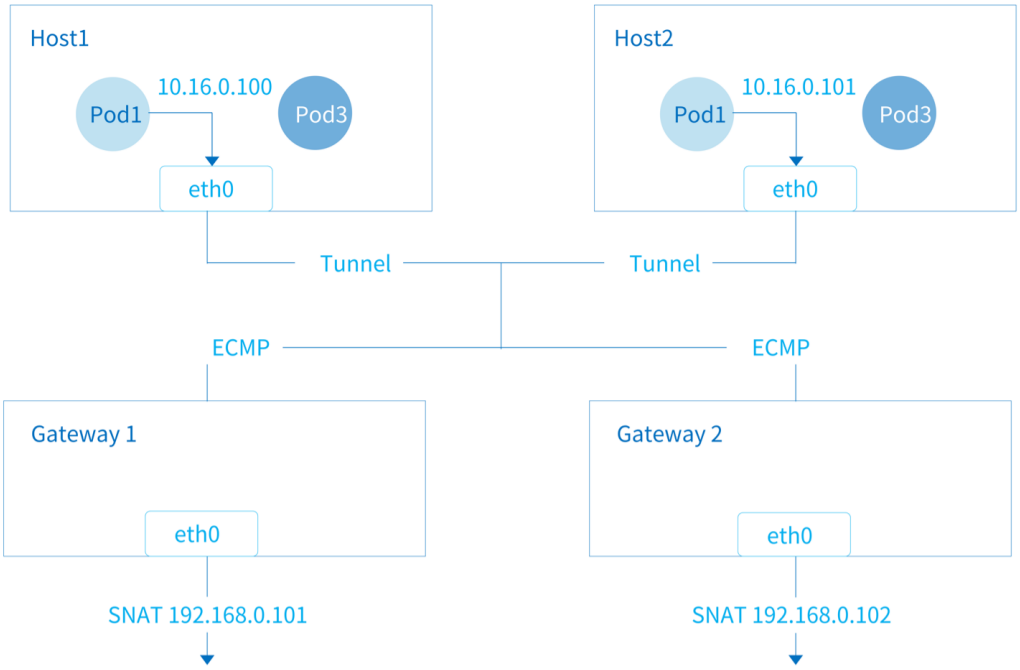
[illegible] Node [illegible] Node [illegible] Pod [illegible] `ovn0` [illegible] `natOutgoing` [illegible] `true` [illegible]Pod [illegible] IP[illegible]

2025 Kube-OVN Team



gatewayType distributed

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: distributed
spec:
  protocol: IPv4
  cidrBlock: 10.166.0.0/16
  default: false
  excludeIps:
  - 10.166.0.1
  gateway: 10.166.0.1
  gatewayType: distributed
  natOutgoing: true
```



IP Pod ovn0 natOutgoing true Pod IP

gatewayType centralized gatewayNode Kubernetes NodeName gatewayNode

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: centralized
spec:
  protocol: IPv4
  cidrBlock: 10.166.0.0/16
  default: false
  excludeIps:
  - 10.166.0.1
  gateway: 10.166.0.1
  gatewayType: centralized
  gatewayNode: "node1,node2"
  natOutgoing: true
```

- gatewayNode kube-ovn-worker:172.18.0.2, kube-ovn-control-plane:172.18.0.3
- ECMP ECMP
- Kube-OVN v1.12.0 subnet crd spec enableEcmp ECMP ECMP kube-ovn-controller Deployment enable-ecmp v1.12.0

>

ECMP kube-ovn-controller ping 5s 5s-10s

Node Ready

gatewayNodeSelectors

2025 Kube-OVN Team

gatewayNodeSelectors 使用说明

- 如果 `gatewayNode` 不为空，则优先使用 `gatewayNode`，`gatewayNodeSelectors` 不会生效。
- 多个选择器之间是 OR 的关系，节点匹配任意一个选择器即可成为网关节点。
- 当节点标签发生变化时，系统会自动更新网关节点列表。

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: centralized-selector
spec:
  protocol: IPv4
  cidrBlock: 10.166.0.0/16
  default: false
  excludeIps:
  - 10.166.0.1
  gateway: 10.166.0.1
  gatewayType: centralized
  gatewayNodeSelectors:
    - matchLabels:
        role: gateway
    - matchExpressions:
        - key: node-type
          operator: In
          values: ["gateway", "egress"]
  natOutgoing: true
```

3.1.5 ACL

Warning

Kube-OVN 同时支持 NetworkPolicy、Network Policy API、Subnet ACL 和 Security Group 四种类型的访问控制，其底层都是通过 OVN ACL 来实现。其中 NetworkPolicy 和 NetworkPolicy API 在设计时考虑了规则分层，可以做到优先级不冲突，其他类型的访问控制方式混用都可能导致优先级冲突。我们推荐不要同时使用多种访问控制规则，以避免优先级冲突带来的规则混乱。

对于有细粒度 ACL 控制的场景，Kube-OVN 的 Subnet 提供了 ACL 规则的设置，可以实现网络规则的精细控制。

Subnet 中的 ACL 规则和 OVN 的 ACL 规则一致，相关字段内容可以参考 ovn-nb ACL Table，`match` 字段支持的字段可参考 ovn-sb Logical Flow Table。

允许 IP 地址为 `10.10.0.2` 的 Pod 访问所有地址，但不允许其他地址主动访问自己的 ACL 规则示例如下：

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: acl
spec:
  allowEWTraffic: false
  acls:
    - action: drop
      direction: to-lport
      match: ip4.dst == 10.10.0.2 && ip
      priority: 1002
    - action: allow-related
      direction: from-lport
      match: ip4.src == 10.10.0.2 && ip
      priority: 1002
  cidrBlock: 10.10.0.0/24
```

某些场景下用户希望配置了 ACL 规则的子网内部网络通信不受影响，可以通过配置 `allowEWTraffic: true` 来实现。

3.1.6

> 子网 ACL 的功能可以覆盖子网隔离的功能，并有更好的灵活性，我们推荐使用子网 ACL 来做相应的配置。

默认情况下 Kube-OVN 创建的子网之间可以相互通信，Pod 也可以通过网关访问外部网络。

如需对子网间的访问进行控制，可以在子网 CRD 中将 `private` 设置为 true，则该子网将和其他子网以及外部网络隔离，只能进行子网内部的通信。如需开白名单，可以通过 `allowSubnets` 设置。`allowSubnets` 内的网段和该子网可以双向互访。

开启访问控制的子网示例

2025 Kube-OVN Team

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: private
spec:
  protocol: IPv4
  default: false
  namespaces:
  - ns1
  - ns2
  cidrBlock: 10.69.0.0/16
  private: true
  allowSubnets:
  - 10.16.0.0/16
  - 10.18.0.0/16
```

3.1.7 Underlay ⬚⬚⬚⬚

> 该部分功能只对 Underlay 类型子网生效。

- `vlan`：如果使用 Underlay 网络，该字段用来控制该 Subnet 和哪个 Vlan CR 进行绑定。该选项默认为空字符串，即不使用 Underlay 网络。
- `logicalGateway`：一些 Underlay 环境为纯二层网络，不存在物理的三层网关。在这种情况下可以借助 OVN 本身的能力设置一个虚拟网关，将 Underlay 和 Overlay 网络打通。默认值为：`false`。

仅分配 **MAC** 的 **Underlay** 子网（外部 **DHCP**）

对于由外部 DHCP 或虚拟机客户机自行获取 IP 的网络，可以创建一个不配置 CIDR 的 Underlay 子网。该模式要求 `vlan` 非空且 `cidrBlock` 为空，Kube-OVN 会将子网协议标记为 `Mac`（`ProtocolMac`），为每个 Pod 网卡通过 IPAM 分配并追踪 MAC，不分配 IP。客户机随后从外部 DHCP 获取 IP。

MAC-only 子网不能作为普通 Pod 的主网卡：主网卡没有 IP 时，CRI 无法完成 Pod 创建，Pod 会保持 Pending。请将该模式用于 Multus 附属网卡，或用于由客户机自行配置网络的 KubeVirt VM。

由于没有地址池，`gateway` 和 `excludeIps` 必须为空；该子网不应配置依赖 CIDR 的网关、路由或 NAT。CNI 在拿到 MAC 后会跳过 IP、路由和网关连通性等待，接口只配置 MAC，避免把外部 DHCP 尚未完成误判为地址分配失败。

`vlan` 必须引用已配置到目标 `ProviderNetwork` 的 `Vlan`。例如：

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: dhcp-underlay
spec:
  vlan: vlan-dhcp
  provider: dhcp-net.default.ovn
  cidrBlock: ""
  gateway: ""
  excludeIps: []
```

Pod 可以使用 provider 级 MAC annotation 指定固定 MAC。主网卡使用 `ovn.kubernetes.io/mac_address`，OVN 附属网卡使用 `<nadName>.<nadNamespace>.ovn.kubernetes.io/mac_address`；当同一个 NAD 挂载多个接口时，可进一步使用 `<nadName>.<nadNamespace>.kubernetes.io/mac_address.<interfaceName>`，按接口指定 MAC。按接口 annotation 优先于 provider 级 annotation，未指定时由 IPAM 随机分配并在重试中保持不变。

```yaml
metadata:
  annotations:
    k8s.v1.cni.cncf.io/networks: '[{"name":"dhcp-net","namespace":"default","interface":"net1"}]'
    dhcp-net.default.kubernetes.io/mac_address.net1: "02:00:00:00:10:01"
```

MAC-only 子网的 IP CR 会记录 MAC 和空 IP；删除 Pod 或释放网卡时，IPAM 会同时移除该 NIC 的 MAC 映射。已有 CIDR 的子网不能通过更新清空 `cidrBlock`，MAC-only 子网也不能再添加 CIDR；需要变更网络模式时请创建新的 Subnet。

3.1.8 ⬚⬚⬚⬚⬚⬚

默认情况下 `kube-ovn-cni` 在启动 Pod 后会使用 ICMP 或 ARP 协议请求网关并等待返回，以验证网络工作正常，在部分 Underlay 环境网关无法响应 ICMP 请求，或无需网络外部联通的场景可以关闭网关检查。

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: disable-gw-check
spec:
  disableGatewayCheck: true
```


2025 Kube-OVN Team

3.1.9 Multicast-Snoop ⬚⬚

⬚⬚⬚⬚ subnet ⬚⬚ Pod ⬚⬚⬚⬚⬚⬚⬚⬚OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Pod⬚⬚⬚⬚⬚ subnet ⬚ multicast snoop ⬚⬚⬚OVN ⬚⬚⬚ `South Database` ⬚⬚⬚⬚⬚ `Multicast_Group` ⬚⬚ ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: sample1
spec:
  enableMulticastSnoop: true
```

3.1.10 Subnet MTU ⬚⬚

⬚⬚ Subnet ⬚ Pod ⬚ MTU⬚⬚⬚⬚⬚⬚⬚⬚ Subnet ⬚⬚ Pod ⬚⬚⬚

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: sample1
spec:
  mtu: 1300
```

3.1.11 ⬚⬚⬚⬚⬚⬚⬚

- IP ⬚⬚⬚
- ⬚⬚ VPC NAT ⬚⬚⬚⬚
- QoS ⬚⬚
- ⬚⬚⬚⬚⬚⬚
- DHCP ⬚⬚
- ⬚⬚⬚⬚⬚⬚⬚
- ⬚⬚⬚⬚⬚⬚⬚
- ⬚⬚ IP ⬚⬚

PDF Slack Support

2026⬚10⬚28⬚

2022⬚5⬚24⬚

GitHub



+5

3.1.12 ⬚⬚

3.2 [illegible]

[illegible] `Subnet` [illegible] `spec.routed: true` [illegible] Pod [illegible]IPv4 [illegible] `/32` [illegible]IPv6 [illegible] `/128` [illegible] Pod [illegible] OVN [illegible]

3.2.1 [illegible]

[illegible] Overlay IPv4 [illegible]

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: routed-subnet
spec:
  cidrBlock: 10.0.0.0/16
  gateway: 10.0.0.1
  protocol: IPv4
  provider: ovn
  routed: true
```

`spec.cidrBlock` [illegible] IPAM [illegible] Pod [illegible] ACL[illegible] OVN [illegible] `10.0.0.5` [illegible] Pod [illegible]CNI [illegible]

```
ip addr add 10.0.0.5/32 dev eth0
ip route add 10.0.0.1/32 dev eth0
ip route add default via 10.0.0.1
```

IPv6 Pod [illegible] `/128` [illegible] IPv6 [illegible]

3.2.2 [illegible]

[illegible] OVN [illegible] hairpin[illegible] OVN ACL [illegible]

- [illegible] ARP/ND[illegible]
- [illegible] ACL [illegible] MAC [illegible] IP [illegible]
- [illegible] MAC [illegible] IP [illegible] `from-lport` [illegible] Pod [illegible] MAC[illegible]
- [illegible] IP[illegible]ARP [illegible] ND [illegible]

[illegible] `private: true` [illegible] hairpin[illegible] Join CIDR [illegible] `allowSubnets` [illegible]

3.2.3 [illegible]

[illegible] DPDK [illegible] Provider [illegible] `ovn.kubernetes.io/routes` [illegible] U2O [illegible] U2O [illegible] IP[illegible] CIDR[illegible] CNI ADD [illegible]

[illegible] Pod [illegible] `ovn.kubernetes.io/routed` [illegible] Pod [illegible] OVN [illegible] Pod [illegible]

3.2.4 [illegible]

- [illegible] OVN Provider [illegible]Overlay [illegible]Underlay [illegible] `logicalGateway` [illegible] `u2oInterconnection` [illegible] OVN [illegible]
- [illegible] `enableDHCP` [illegible] `enableIPv6RA` [illegible] `enableMulticastSnoop` [illegible]mac-only/BYO-DHCP [illegible] `spec.acls` [illegible]
- [illegible] DPDK/vhost-user [illegible]
- [illegible] `routed` [illegible] Pod[illegible] Pod[illegible]

PDF | Slack | Support

2025 Kube-OVN Team

2026⊠10⊠28⊠

2026⊠10⊠28⊠

GitHub 

3.2.5 ⊠⊠

3.3 [illegible]

Kube-OVN 默认会根据 Pod 所在 Namespace 所属的子网中随机分配 IP 和 Mac。针对工作负载需要固定地址的情况，Kube-OVN 根据不同的场景，提供了多种固定地址的方法：

- 单个 Pod 固定 IP/Mac。
- 同一交换机下多网卡分别固定 IP/MAC（多网卡接入同一子网时按网卡指定）。
- Workload 通用 IP Pool 方式指定固定地址范围。
- StatefulSet 固定地址。
- KubeVirt VM 固定地址。
- 使用 Multus 给附属网卡固定地址。

3.3.1 单个 Pod 固定 IP 和 Mac

可以在创建 Pod 时通过 annotation 来指定 Pod 运行时所需的 IP/Mac，`kube-ovn-controller` 运行时将会跳过地址随机分配阶段，经过冲突检测后直接使用指定地址，如下所示：

```
apiVersion: v1
kind: Pod
metadata:
  name: static-ip
  annotations:
    ovn.kubernetes.io/ip_address: 10.16.0.15   #           10.16.0.15,fd00:10:16::15
    ovn.kubernetes.io/mac_address: 00:00:00:53:6B:B6
spec:
  containers:
  - name: static-ip
    image: docker.io/library/nginx:alpine
```

在使用 annotation 定义单个 Pod IP/Mac 时需要注意以下几点：

1. 所使用的 IP/Mac 不能和已有的 IP/Mac 冲突。
2. IP 必须在所属子网的 CIDR 内。
3. 可以只指定 IP 或 Mac，只指定一个时，另一个会随机分配。

3.3.2 同一交换机下多网卡分别固定 IP/MAC

当 Pod 或 VM 通过 Multus 挂载了多个网卡且这些网卡接入同一逻辑交换机时，可为每个网卡分别指定静态 IP/MAC。需在 Multus 的 `k8s.v1.cni.cncf.io/networks` 中为每个挂载指定不同的 `interface` 名称，再使用按网卡区分的 annotation：

- `<nadName>.<nadNamespace>.kubernetes.io/ip_address.<interfaceName>`：指定该网卡的静态 IP
- `<nadName>.<nadNamespace>.kubernetes.io/mac_address.<interfaceName>`：指定该网卡的静态 MAC

其中 `<nadName>`、`<nadNamespace>` 为 NetworkAttachmentDefinition 的名称与命名空间，`<interfaceName>` 须与 Multus 中该挂载的 `interface` 一致。未使用上述按网卡 annotation 时，仍可使用扁平 annotation `ovn.kubernetes.io/ip_address` / `ovn.kubernetes.io/mac_address` 作为回退（作用于主网卡或单网卡场景）。

示例：同一 NAD 挂载两次到同一子网，两个网卡分别固定 IP/MAC：

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-if-static
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: '[{"name": "attachnet", "namespace": "default", "interface": "net1"}, {"name": "attachnet", "namespace": "default",
"interface": "net2"}]'
    attachnet.default.kubernetes.io/ip_address.net1: 172.17.0.100
    attachnet.default.kubernetes.io/mac_address.net1: 00:00:00:53:6B:B1
    attachnet.default.kubernetes.io/ip_address.net2: 172.17.0.101
    attachnet.default.kubernetes.io/mac_address.net2: 00:00:00:53:6B:B2
spec:
  containers:
  - name: multi-if-static
    image: docker.io/library/nginx:alpine
```


2025 Kube-OVN Team

3.3.3 Workload 绑定 IP Pool 的使用

Kube-OVN 支持通过 annotation `ovn.kubernetes.io/ip_pool` 给 Workload（Deployment/StatefulSet/DaemonSet/Job/CronJob）设置固定 IP。`kube-ovn-controller` 会自动选择 `ovn.kubernetes.io/ip_pool` 中指定的 IP 并进行冲突检测。

IP Pool 的 Annotation 需要加在 `template` 内的 `annotation` 字段，除了 Kubernetes 内置的 Workload 类型，其他用户自定义的 Workload 也可以使用同样的方式进行固定地址分配。

Deployment 固定 IP 示例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ippool
  labels:
    app: ippool
spec:
  replicas: 2
  selector:
    matchLabels:
      app: ippool
  template:
    metadata:
      labels:
        app: ippool
      annotations:
        ovn.kubernetes.io/ip_pool: 10.16.0.15,10.16.0.16,10.16.0.17 #
10.16.0.15,fd00:10:16::000E;10.16.0.16,fd00:10:16::000F;10.16.0.17,fd00:10:16::0010
    spec:
      containers:
      - name: ippool
        image: docker.io/library/nginx:alpine
```

对 Workload 使用固定 IP 需要注意以下几点：

1. `ovn.kubernetes.io/ip_pool` 中的 IP 应该属于所在子网的 CIDR 内。
2. `ovn.kubernetes.io/ip_pool` 中的 IP 不能和已使用的 IP 冲突。
3. 当 `ovn.kubernetes.io/ip_pool` 中的 IP 数量小于 replicas 数量时，多出的 Pod 将无法创建。你需要根据 Workload 的更新策略以及扩容规划调整 `ovn.kubernetes.io/ip_pool` 中 IP 的数量。

3.3.4 StatefulSet 的使用

StatefulSet 默认支持固定 IP，而且和其他 Workload 相同，可以使用 `ovn.kubernetes.io/ip_pool` 来指定 Pod 使用的 IP 范围。

由于 StatefulSet 多用于有状态服务，对网络标示的固定有更高的要求，Kube-OVN 做了特殊的强化：

1. Pod 会按顺序分配 `ovn.kubernetes.io/ip_pool` 中的 IP。例如 StatefulSet 的名字为 web，则 web-0 会使用 `ovn.kubernetes.io/ip_pool` 中的第一个 IP，web-1 会使用第二个 IP，以此类推。
2. StatefulSet Pod 在更新或删除的过程中 OVN 中的 logical_switch_port 不会删除，新生成的 Pod 直接复用旧的 interface 信息。因此 Pod 可以复用 IP/Mac 及其他网络信息，达到和 StatefulSet Volume 类似的状态保留功能。
3. 基于 2 的能力，对于没有 `ovn.kubernetes.io/ip_pool` 注解的 StatefulSet，Pod 第一次生成时会随机分配 IP/Mac，之后在整个 StatefulSet 的生命周期内，网络信息都会保持固定。

StatefulSet 示例

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: docker.io/library/nginx:alpine
        ports:
        - containerPort: 80
          name: web
```


2025 Kube-OVN Team

可以尝试删除 StatefulSet 下 Pod 观察 Pod IP 变化信息。

更新 StatefulSet Pod IP

由于 StatefulSet 下的 IP 和 Pod Name 生命周期绑定，直接更新 Statefulset 的 `ovn.kubernetes.io/ip_pool` Annotation 无法更新 Pod 的 IP。

如果遇到需要更新 StatefulSet Pod IP 的场景请先将 StatefulSet 的副本数 scale 到 0，之后再更新 Annotation 并恢复 StatefulSet 副本数。

3.3.5 KubeVirt VM 固定地址

针对 KubeVirt 创建的 VM 实例，`kube-ovn-controller` 可以按照类似 StatefulSet Pod 的方式进行 IP 地址分配和管理。以达到 VM 实例在生命周期内启停，升级，迁移等操作过程中地址固定不变，更符合虚拟化用户的实际使用体验。具体操作请参考 VM 固定 IP。

3.3.6 使用 Multus 给附属网卡固定地址

在使用 Multus 给 Pod 配置多网卡的情况下，针对 Kube-OVN 类型的网卡可以通过特定的 annotation 来配置固定地址，对于非 Kube-OVN 的其他 CNI，Kube-OVN 也可以单独提供 IPAM 能力来使得其他 CNI 也具备固定地址的能力。具体操作请参考多网卡管理。

PDF Slack Support

2026年7月4日

2022年5月20日

GitHub +3

3.3.7 评

3.4 IP ■■■

IP ■■IPPool■■■■■■Subnet■■■■■■ IPAM ■■■■■■■■■■ IP ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■Namespace■■■ Workload ■■■

3.4.1 ■■■■

■■■■■

```
apiVersion: kubeovn.io/v1
kind: IPPool
metadata:
  name: pool-1
spec:
  subnet: ovn-default
  ips:
  - "10.16.0.201"
  - "10.16.0.210/30"
  - "10.16.0.220..10.16.0.230"
  namespaces:
  - ns-1
  enableAddressSet: true
```

■■■■■ Workload■

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ippool
  labels:
    app: ippool
spec:
  replicas: 2
  selector:
    matchLabels:
      app: ippool
  template:
    metadata:
      labels:
        app: ippool
      annotations:
        ovn.kubernetes.io/ip_pool: pool-1
    spec:
      containers:
      - name: ippool
        image: docker.io/library/nginx:alpine
```

3.4.2 ■■■■ IP ■

■ `ovn.kubernetes.io/ip_pool` ■■■ IPPool ■■■■■ `pool-1` ■■■■■■■■ Pod ■■■■■ `ovn.kubernetes.io/logical_switch` ■■■■■■■ `pool-1.spec.subnet` ■■ Pod ■■■■ ■■■ Pod ■ `logical_switch` ■ `cidr` ■ `gateway` ■■■

```
metadata:
  annotations:
    ovn.kubernetes.io/ip_pool: pool-1
```

■■■■■■ `ovn.kubernetes.io/logical_switch` ■■■■■■ `IPPool.spec.subnet` ■■■ ■■■■■■■■■■■■■

■■■■■ Pod■■■■■■ `<provider>.kubernetes.io/ip_pool` ■■■ provider ■■■■ IP ■■provider ■■■■■■ IPPool ■■■■■■■■■■■■■■■■■■■ `<provider>.kubernetes.io/logical_switch` ■■■■■■■■■■■■■■■■■■■■■ IP■ ■■■■■ IP ■■■■■■■ IPv4■IPv6 ■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■■■■■ IP ■■■■■■■■■■ IPPool ■■■■■■

`spec.namespaces` ■■■ IPPool ■■■■■■■■■■■■■■■■■■■■■■■■■ IPPool ■ Pod■ ■■■■■■■■■■■■■■■■■■ Pod ■■■■■■■■■■■■■IPPool ■■■■■■■■■■■ StatefulSet ■ VM ■ Pod ■■■■■■ ■■■■■■■ Workload ■■■■■

2025 Kube-OVN Team

[illegible]

[illegible]	[illegible]	[illegible]
subnet	[illegible]	[illegible]
ips	[illegible] IP [illegible]	[illegible] .. [illegible] IPv6[illegible] ipCount [illegible]
ipCount	[illegible]	[illegible] 1-32[illegible] IPv4 [illegible] IPv6 [illegible] status.ips [illegible] ips [illegible]
namespaces	[illegible]	[illegible] IPPool [illegible] Pod[illegible]
enableAddressSet	[illegible] AddressSet	[illegible] false[illegible] true [illegible] ACL [illegible] AddressSet [illegible]

[illegible] **ipCount** [illegible]

[illegible] IP [illegible] `ipCount` [illegible]

```yaml
apiVersion: kubeovn.io/v1
kind: IPPool
metadata:
  name: pool-auto
spec:
  subnet: ovn-default
  ipCount: 5
  namespaces:
  - ns-1
```

[illegible] 5 [illegible] `status.ips` [illegible] `ipCount` [illegible] IPPool [illegible]

3.4.3 [illegible]

1. [illegible] Workload [illegible] IP Pool [illegible] [illegible]IP [illegible] IP [illegible]
2. IP [illegible] `.spec.ips` [illegible] IP [illegible] IP [illegible] `.spec.ips` [illegible] CIDR [illegible]
3. [illegible] IP [illegible]IP [illegible]
4. IP [illegible] `.spec.ips` [illegible]
5. IP [illegible] IP[illegible] IP [illegible] IP [illegible] IP [illegible] IP[illegible]
6. [illegible] IP [illegible] IP [illegible]
7. [illegible] IP [illegible] Namespace[illegible]
8. IP [illegible] `.spec.enableAddressSet` [illegible] `false` [illegible] `true` [illegible] IP [illegible] OVN NB [illegible] AddressSet [illegible] IP [illegible] IP [illegible] AddressSet [illegible] NetworkPolicy [illegible] VPC [illegible] AddressSet [illegible]

PDF Slack Support

2026[illegible]11[illegible]5[illegible]

2023[illegible]7[illegible]10[illegible]

GitHub



3.4.4 [illegible]

2025 Kube-OVN Team

3.5 □□□□□

□□□□ Pod □□□ Annotations □□□□□□□□□□□□□□□□□□

```
apiVersion: v1
kind: Pod
metadata:
  name: custom-routes
  annotations:
    ovn.kubernetes.io/routes: |
      [{
        "dst": "192.168.0.101/24",
        "gw": "10.16.0.254"
      }, {
        "gw": "10.16.0.254"
      }]
spec:
  containers:
  - name: nginx
    image: docker.io/library/nginx:alpine
```

> `dst` □□□□□□□□□□□□□□□

□□□□□□□ Deployment□DaemonSet □ StatefulSet□□□□ Annotation □□□□□□□□□ `.spec.template.metadata.annotations` □□□□□□□□

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: custom-routes
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        ovn.kubernetes.io/routes: |
          [{
            "dst": "192.168.0.101/24",
            "gw": "10.16.0.254"
          }, {
            "gw": "10.16.0.254"
          }]
    spec:
      containers:
      - name: nginx
        image: docker.io/library/nginx:alpine
```

PDF Slack Support

2023□9□26□

2023□2□16□

GitHub



3.5.1 □□

3.6 EIP 和 SNAT 配置

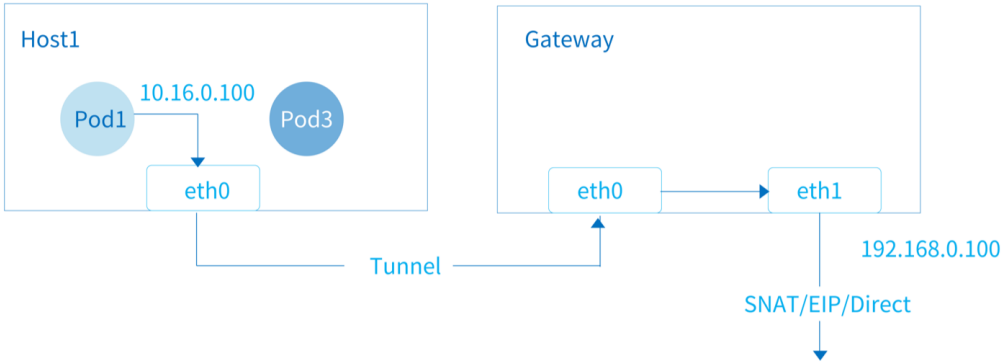
> 介绍如何在 VPC 网络中使用 VPC 内部的 NAT网关。

- VPC Iptables NAT 网关
- VPC OVN NAT 网关

> 在 VPC 网络中使用浮动公网IP, 实现公网访问

- 支持为指定 VPC 内的指定 OVN NAT 网关添加 pod annotation 来申请浮动公网IP
- 支持为 VPC 内的指定 OVN NAT 网关创建 provider-network、vlan、subnet CRD 来对接外部网络，创建 ovn-eip、ovn-dnat、ovn-fip、ovn-snat CRD 来进行 EIP 和 NAT的管理。VPC OVN NAT 网关
- 支持为 VPC 内的指定 Iptables NAT 网关对接VPC Iptables NAT 网关

Kube-OVN 支持 OVN 的 L3 Gateway 功能为 Pod 提供 SNAT 和 EIP 等多种 SNAT功能。Pod 可以使用 IP 地址申请浮动 EIP 并配置 Pod 的网络 IP 地址为指定的 EIP 地址，Pod 可以使用 EIP 访问外网。



3.6.1 使用场景

- 支持为 OVN 的 L3 Gateway 配置多种不同的 OVS 流表以支持 Overlay 和 Underlay 子网的多种不同网络出口
- 支持为 NAT 网关配置浮动 Underlay 网络的公网地址以访问外网
- 支持 EIP 和 SNAT 的高可用以及流量的负载均衡

3.6.2 配置说明

在 `kube-system` 命名空间下创建 ConfigMap `ovn-external-gw-config` 如下：

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: ovn-external-gw-config
  namespace: kube-system
data:
  type: "centralized"
  enable-external-gw: "true"
  external-gw-nodes: "kube-ovn-worker"
  external-gw-nic: "eth1"
  external-gw-addr: "172.56.0.1/16"
  nic-ip: "172.56.0.254/16"
  nic-mac: "16:52:f3:13:6a:25"
```

- `enable-external-gw`：是否启用 SNAT 和 EIP 功能
- `type`：`centralized` 或 `distributed`，表示 `centralized` 或者 `distributed` 两种不同的网关模式
- `external-gw-nodes`：`centralized` 模式下的网关节点

2025 Kube-OVN Team

- `external-gw-nic` : ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚
- `external-gw-addr` : ⬚⬚⬚⬚⬚⬚⬚ IP ⬚⬚⬚⬚
- `nic-ip`,`nic-mac` : ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ IP ⬚ Mac⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ IP ⬚ Mac⬚

3.6.3 ⬚⬚ OVN ⬚ OVS ⬚⬚⬚⬚⬚⬚⬚⬚

⬚⬚ OVN-NB ⬚⬚⬚⬚⬚ `ovn-external` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `ovn-cluster-ovn-external` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ chassis⬚

```
# kubectl ko nbctl show
switch 3de4cea7-1a71-43f3-8b62-435a57ef16a6 (external)
    port localnet.external
        type: localnet
        addresses: ["unknown"]
    port external-ovn-cluster
        type: router
        router-port: ovn-cluster-external
router e1eb83ad-34be-4ed5-9a02-fcc8b1d357c4 (ovn-cluster)
    port ovn-cluster-external
        mac: "ac:1f:6b:2d:33:f1"
        networks: ["172.56.0.100/16"]
        gateway chassis: [a5682814-2e2c-46dd-9c1c-6803ef0dab66]
```

⬚⬚ OVS ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `br-external` ⬚⬚⬚

```
# kubectl ko vsctl ${gateway node name} show
e7d81150-7743-4d6e-9e6f-5c688232e130
    Bridge br-external
        Port br-external
            Interface br-external
                type: internal
        Port eth1
            Interface eth1
        Port patch-localnet.external-to-br-int
            Interface patch-localnet.external-to-br-int
                type: patch
                options: {peer=patch-br-int-to-localnet.external}
```

3.6.4 Pod ⬚⬚ EIP ⬚ SNAT

⬚⬚⬚⬚ Pod ⬚⬚⬚ `ovn.kubernetes.io/snat` ⬚ `ovn.kubernetes.io/eip` annotation ⬚⬚⬚⬚⬚ SNAT ⬚ EIP⬚

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-snat
  annotations:
    ovn.kubernetes.io/snat: 172.56.0.200
spec:
  containers:
  - name: pod-snat
    image: docker.io/library/nginx:alpine
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-eip
  annotations:
    ovn.kubernetes.io/eip: 172.56.0.233
spec:
  containers:
  - name: pod-eip
    image: docker.io/library/nginx:alpine
```

⬚⬚⬚ kubectl ⬚⬚⬚⬚⬚⬚⬚⬚ Pod ⬚⬚⬚⬚ EIP ⬚ SNAT ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `ovn.kubernetes.io/routed` annotation ⬚⬚⬚⬚⬚⬚⬚⬚

```
kubectl annotate pod pod-eip ovn.kubernetes.io/eip=172.56.0.221 --overwrite
kubectl annotate pod pod-eip ovn.kubernetes.io/routed-
```

⬚ EIP ⬚ SNAT ⬚⬚⬚⬚⬚⬚ `ovn.kubernetes.io/routed` annotation ⬚⬚⬚⬚⬚⬚⬚⬚

3.6.5 ⬚⬚⬚⬚

`kube-ovn-controller` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ SNAT ⬚ EIP ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

- `--external-gateway-config-ns` : Configmap `ovn-external-gw-config` ⬚⬚ Namespace⬚⬚⬚⬚⬚ `kube-system` ⬚

2025 Kube-OVN Team



- `--external-gateway-net`：物理网卡所桥接的网桥名，默认为 `external`。
- `--external-gateway-vlanid`：物理网络 Vlan Tag 号，默认为 0，即不使用 Vlan。

PDF Slack Support

2026年4月28日

2022年5月24日

GitHub

3.6.6 ⛓

2025 Kube-OVN Team

3.7 ▯▯▯▯ QoS ▯▯

Kube-OVN ▯▯▯▯▯▯ Pod ▯▯▯▯▯▯▯▯ QoS▯

- ▯▯▯▯▯▯ QoS▯
- `linux-netem` ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ QoS▯▯▯▯▯▯▯▯▯▯

> ▯▯▯▯▯ Pod ▯▯ QoS ▯▯▯ Namespace ▯ Subnet ▯▯▯ QoS ▯▯▯

3.7.1 ▯▯▯▯▯▯▯▯▯▯ QoS

▯▯▯▯ QoS ▯▯▯▯ Pod annotation ▯▯▯▯▯▯▯▯▯▯▯ Pod ▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯ `Mbit/s` ▯

▯▯▯ `ovn.kubernetes.io/ingress_burst` ▯ `ovn.kubernetes.io/egress_burst` ▯▯▯▯▯▯▯▯▯▯▯▯▯▯ burst▯▯▯▯ `Mbit` ▯▯▯▯▯▯▯burst ▯▯▯▯▯▯▯▯▯ 80% ▯▯▯▯▯▯▯ `0` ▯▯▯ burst▯burst ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ Pod ▯▯▯▯▯▯ `<provider>.kubernetes.io/ingress_burst` ▯ `<provider>.kubernetes.io/egress_burst` ▯▯▯ provider ▯▯ ▯▯▯ burst▯

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: qos
  namespace: ls1
  annotations:
    ovn.kubernetes.io/ingress_rate: "3"
    ovn.kubernetes.io/egress_rate: "1"
    ovn.kubernetes.io/ingress_burst: "2"
    ovn.kubernetes.io/egress_burst: "0"
spec:
  containers:
  - name: qos
    image: docker.io/library/nginx:alpine
```

▯▯ annotation ▯▯▯▯ QoS▯

```
kubectl annotate --overwrite  pod nginx-74d5899f46-d7qkn ovn.kubernetes.io/ingress_rate=3
#             burst    Mbit
kubectl annotate --overwrite pod nginx-74d5899f46-d7qkn ovn.kubernetes.io/ingress_burst=2 ovn.kubernetes.io/egress_burst=0
```

▯▯ **QoS** ▯▯

▯▯▯▯▯▯▯▯▯▯▯▯▯

```yaml
kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: perf
  namespace: ls1
  labels:
    app: perf
spec:
  selector:
    matchLabels:
      app: perf
  template:
    metadata:
      labels:
        app: perf
    spec:
      containers:
      - name: nginx
        image: docker.io/kubeovn/perf
```

▯▯▯▯▯▯ Pod ▯▯▯ iperf3 server▯

```
# kubectl exec -it perf-4n4gt -n ls1 sh
# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

▯▯▯▯▯ Pod ▯▯▯▯▯ Pod▯

2025 Kube-OVN Team

```
# kubectl exec -it perf-d4mqc -n ls1 sh
# iperf3 -c 10.66.0.12
Connecting to host 10.66.0.12, port 5201
[  4] local 10.66.0.14 port 51544 connected to 10.66.0.12 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  86.4 MBytes   725 Mbits/sec    3    350 KBytes
[  4]   1.00-2.00   sec  89.9 MBytes   754 Mbits/sec  118    473 KBytes
[  4]   2.00-3.00   sec   101 MBytes   848 Mbits/sec  184    586 KBytes
[  4]   3.00-4.00   sec   104 MBytes   875 Mbits/sec  217    671 KBytes
[  4]   4.00-5.00   sec   111 MBytes   935 Mbits/sec  175    772 KBytes
[  4]   5.00-6.00   sec   100 MBytes   840 Mbits/sec  658    598 KBytes
[  4]   6.00-7.00   sec   106 MBytes   890 Mbits/sec  742    668 KBytes
[  4]   7.00-8.00   sec   102 MBytes   857 Mbits/sec  764    724 KBytes
[  4]   8.00-9.00   sec  97.4 MBytes   817 Mbits/sec 1175    764 KBytes
[  4]   9.00-10.00  sec   111 MBytes   934 Mbits/sec 1083    838 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1010 MBytes   848 Mbits/sec  5119             sender
[  4]   0.00-10.00  sec  1008 MBytes   846 Mbits/sec                  receiver

iperf Done.
```

□□□□□ Pod □□□□□ QoS□

```
kubectl annotate --overwrite  pod perf-4n4gt -n ls1 ovn.kubernetes.io/ingress_rate=30
```

□□□□□□ Pod □□□□□ Pod □□□

```
# iperf3 -c 10.66.0.12
Connecting to host 10.66.0.12, port 5201
[  4] local 10.66.0.14 port 52372 connected to 10.66.0.12 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  3.66 MBytes  30.7 Mbits/sec    2   76.1 KBytes
[  4]   1.00-2.00   sec  3.43 MBytes  28.8 Mbits/sec    0    104 KBytes
[  4]   2.00-3.00   sec  3.50 MBytes  29.4 Mbits/sec    0    126 KBytes
[  4]   3.00-4.00   sec  3.50 MBytes  29.3 Mbits/sec    0    144 KBytes
[  4]   4.00-5.00   sec  3.43 MBytes  28.8 Mbits/sec    0    160 KBytes
[  4]   5.00-6.00   sec  3.43 MBytes  28.8 Mbits/sec    0    175 KBytes
[  4]   6.00-7.00   sec  3.50 MBytes  29.3 Mbits/sec    0    212 KBytes
[  4]   7.00-8.00   sec  3.68 MBytes  30.9 Mbits/sec    0    294 KBytes
[  4]   8.00-9.00   sec  3.74 MBytes  31.4 Mbits/sec    0    398 KBytes
[  4]   9.00-10.00  sec  3.80 MBytes  31.9 Mbits/sec    0    526 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  35.7 MBytes  29.9 Mbits/sec    2             sender
[  4]   0.00-10.00  sec  34.5 MBytes  29.0 Mbits/sec                  receiver

iperf Done.
```

3.7.2 linux-netem QoS

> RHEL □□□□□□□□□□ netem □□□□□ yum install -y kernel-modules-extra && modprobe sch_netem

Pod □□□□□□ annotation □□ `linux-netem` □□ QoS□□□ netem □□□□□□□□□□ QoS □□□ Pod □ Ingress □□□□□□□

- `ovn.kubernetes.io/latency` □□□ Pod □□□□□□□□□□□□□□□ ms□
- `ovn.kubernetes.io/jitter` □□□ Pod □□□□□□□□□□□□□□□□□□ ms□
- `ovn.kubernetes.io/limit` □ □ `qdisc` □□□□□□□□□□□□□□□□□□□□□□□□□□□ 1000□
- `ovn.kubernetes.io/loss` □ □□□□□□□□□□□□□□□ float □□□□□□□□ 20□□□□□ 20% □□□□□□

PDF Slack Support

2026□10□29□

2022□5□24□

GitHub 

3.7.3 ▯▯

3.8 [illegible]

Kube-OVN 中不同的子网可以支持不同的协议，一个集群内可以同时存在 IPv4，IPv6 和双栈类型的子网。建议在一个集群内使用统一的协议类型以简化使用和维护。

为了支持双栈，需要主机网络满足双栈要求，同时需要对 Kubernetes 相关参数做调整，请参考 Kubernetes 的双栈官方指导。

3.8.1 [illegible]栈子[illegible]

在配置双栈时，只需要设置对应子网 CIDR 格式为 `cidr=<IPv4 CIDR>,<IPv6 CIDR>` 即可。CIDR 顺序要求 IPv4 在前，IPv6 在后，如下所示：

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: ovn-test
spec:
  cidrBlock: 10.16.0.0/16,fd00:10:16::/64
  excludeIps:
  - 10.16.0.1
  - fd00:10:16::1
  gateway: 10.16.0.1,fd00:10:16::1
```

如果需要在安装时默认子网使用双栈，需要在安装脚本中修改如下参数（`install.sh` 双栈下的默认掩码为 `/112`，更易避开常见 ULA 段；如希望使用 `/64` 也允许，但要确保不与节点/Service CIDR 冲突）：

```
POD_CIDR="10.16.0.0/16,fd00:10:16::/112"
JOIN_CIDR="100.64.0.0/16,fd00:100:64::/112"
```

3.8.2 查看 Pod [illegible]

配置双栈网络的 Pod 将会从该子网同时分配 IPv4 和 IPv6 的地址，分配结果会显示在 Pod 的 annotation 中：

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    ovn.kubernetes.io/allocated: "true"
    ovn.kubernetes.io/cidr: 10.16.0.0/16,fd00:10:16::/64
    ovn.kubernetes.io/gateway: 10.16.0.1,fd00:10:16::1
    ovn.kubernetes.io/ip_address: 10.16.0.9,fd00:10:16::9
    ovn.kubernetes.io/logical_switch: ovn-default
    ovn.kubernetes.io/mac_address: 00:00:00:14:88:09
    ovn.kubernetes.io/routed: "true"
...
podIP: 10.16.0.9
  podIPs:
  - ip: 10.16.0.9
  - ip: fd00:10:16::9
```

3.8.3 为单个 Pod 或网卡选择 IP 协议族

在双栈子网中，如果某个 Pod 或某块网卡只需要分配 IPv4 或 IPv6 地址，可以在创建 Pod 时使用 `ip_family` annotation。未设置该 annotation 时，Kube-OVN 仍保持默认双栈分配行为。以下示例假设目标 Pod 或网卡使用的 Subnet 为双栈子网。

默认网络使用 `ovn.kubernetes.io/ip_family`：

```
apiVersion: v1
kind: Pod
metadata:
  name: ipv4-only
  annotations:
    ovn.kubernetes.io/ip_family: ipv4
spec:
  containers:
  - name: nginx
    image: docker.io/library/nginx:alpine
```

支持的取值为 `ipv4` 和 `ipv6`。该功能仅用于从双栈子网中选择单个协议族，不会改变子网自身的协议类型。如果在 IPv4 单栈子网中请求 `ipv6`，或在 IPv6 单栈子网中请求 `ipv4`，Pod 将无法分配地址。

对于通过 Multus 添加的附属网卡，使用 `<provider>.kubernetes.io/ip_family`，`<provider>` 需要和对应 NetworkAttachmentDefinition 或 Subnet 中的 provider 一致：

```
apiVersion: v1
kind: Pod
metadata:
```


2025 Kube-OVN Team

```
  name: attach-ipv6-only
  annotations:
    k8s.v1.cni.cncf.io/networks: default/attachnet
    attachnet.default.ovn.kubernetes.io/ip_family: ipv6
spec:
  containers:
  - name: nginx
    image: docker.io/library/nginx:alpine
```

▯▯▯▯ NetworkAttachmentDefinition ▯▯▯▯▯▯▯▯▯▯▯▯▯ `interface` ▯▯▯▯Kube-OVN ▯▯▯▯▯▯▯▯ provider ▯▯ annotation▯ ▯▯ `net1` ▯▯▯ IPv4▯ `net2` ▯▯▯ IPv6▯

```
apiVersion: v1
kind: Pod
metadata:
  name: attach-mixed-family
  annotations:
    k8s.v1.cni.cncf.io/networks: '[{"name": "attachnet", "namespace": "default", "interface": "net1"}, {"name": "attachnet", "namespace": "default",
"interface": "net2"}]'
    attachnet.default.ovn.net1.kubernetes.io/ip_family: ipv4
    attachnet.default.ovn.net2.kubernetes.io/ip_family: ipv6
spec:
  containers:
  - name: nginx
    image: docker.io/library/nginx:alpine
```

▯▯▯▯▯▯▯▯▯ IP▯▯▯ `ovn.kubernetes.io/ip_address` ▯ `<provider>.kubernetes.io/ip_address` ▯▯▯▯ NetworkAttachmentDefinition ▯▯▯▯▯▯▯ `<nadName>.<nadNamespace>.kubernetes.io/ip_address.<interfaceName>` ▯▯▯ IP ▯▯▯▯▯▯▯ `ip_family` ▯▯▯

PDF Slack Support

2026▯7▯3▯

2022▯5▯24▯



GitHub 



3.8.4 ▯▯

2025 Kube-OVN Team

3.9 Webhook □□

□□ Webhook □□□ Kube-OVN □□ CRD □□□□□□□□□ Webhook □□□□ □□ IP □□□□□□□ Subnet CIDR □□□□□□□□□□□□□□□□□□□□□□□

□□ Webhook □□□□□□ Subnet □ Pod □□□□□□□□□□□□ Kube-OVN □□□ Webhook □□□□□□ Pod□

3.9.1 Cert-Manager □□

Webhook □□□□□□□□□□□□□□□□ cert-manager □□□□□□□□□□□□□□ Webhook □□□□ cert-manager□

□□□□□□□□□□□ cert-manager:

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.8.0/cert-manager.yaml
```

□□ cert-manager □□□□□ cert-manager □□□□

3.9.2 □□ Webhook

□□ Webhook □□□ yaml □□□□:

```
# kubectl apply -f https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/yamls/webhook.yaml
deployment.apps/kube-ovn-webhook created
service/kube-ovn-webhook created
validatingwebhookconfiguration.admissionregistration.k8s.io/kube-ovn-webhook created
certificate.cert-manager.io/kube-ovn-webhook-serving-cert created
issuer.cert-manager.io/kube-ovn-webhook-selfsigned-issuer created
```

3.9.3 □□ v2 Helm Chart □□□□

□□□□□□□ `validatingWebhook.certificate.create` □□□□ `kube-ovn-v2` Helm Chart□ □□□□□□□□□□□□□□□

□ `validatingWebhook.enabled` □ `true` □□Chart □□□□ cert-manager Certificate □□□□ Issuer□□□□□□ `validatingWebhook.certificate.create` □□□□□ `true` □ □□□□□ `false` □□□□ Chart □□□□ Certificate□□□□□ Webhook □ Deployment□Service □ ValidatingWebhookConfiguration□

```yaml
validatingWebhook:
  enabled: true
  certificate:
    create: false
```

□ `validatingWebhook.enabled` □ `false` □□□□ `validatingWebhook.certificate.create` □□□□□□□□□□□□ Webhook □□□

□□□□ Certificate

□□ Webhook □□□□□ cert-manager □□ CA □□□□□□□□□□□□□□ □□□□□□□□□ cert-manager □□□□ □□ Certificate □□□□□□□□□ Issuer□□□□□□□ ClusterIssuer□

□□□ Chart □ GitOps □□□□□□□□□□ Certificate□ Certificate □ Secret □□□□□□ Chart □□□□□□□□□

□□	□□□□	□□
Certificate	`kube-ovn-webhook-serving-cert`	□ Webhook □ `cert-manager.io/inject-ca-from` □□□□□
Secret	`kube-ovn-webhook-secret`	□ Webhook Deployment □□□□□□□□□□□□□□

□□□□□□□□□ Chart □ `namespace` □□□□□□□□□□□ □□□□□□□□□ `kube-system` □□ `clusterDomain` □□□□ `cluster.local` □□ □□□□□□□□ `webhook-certificate.yaml` □

```yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: kube-ovn-webhook-serving-cert
  namespace: kube-system
spec:
  secretName: kube-ovn-webhook-secret
  dnsNames:
    - kube-ovn-webhook.kube-system
    - kube-ovn-webhook.kube-system.svc
```

```
    - kube-ovn-webhook.kube-system.svc.cluster.local
  issuerRef:
    group: cert-manager.io
    kind: Issuer
    name: existing-webhook-issuer
```

▯ `existing-webhook-issuer` ▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯ ClusterIssuer▯▯▯▯ `issuerRef.kind` ▯▯ `ClusterIssuer` ▯ ▯▯ Chart ▯▯▯▯▯▯▯▯▯ DNS ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ▯ ▯▯▯▯▯▯▯▯▯▯▯▯

▯▯▯▯▯▯▯▯▯ `<chart-namespace>` ▯▯▯ Chart ▯▯▯ `namespace` ▯▯ ▯▯▯▯▯▯ `webhook-certificate.yaml` ▯▯ `metadata.namespace` ▯▯▯

▯▯▯▯▯▯▯▯▯▯▯ Certificate ▯▯▯▯▯▯▯▯▯▯▯▯ `validatingWebhook.certificate.create: false` ▯▯ Webhook▯

```
kubectl apply -f webhook-certificate.yaml
kubectl wait --for=condition=Ready certificate/kube-ovn-webhook-serving-cert \
  -n "<chart-namespace>" --timeout=120s
```

Webhook ▯▯ `kube-ovn-webhook-secret` ▯▯ `tls.crt` ▯ `tls.key` ▯▯▯▯ HTTPS ▯▯▯▯▯▯▯▯▯▯▯▯ `ca.crt` ▯

cert-manager ▯ CA ▯▯▯▯▯▯ Webhook ▯ `cert-manager.io/inject-ca-from` ▯▯▯▯▯▯▯ `kube-ovn-webhook-serving-cert` ▯ Certificate▯ ▯▯▯▯ Certificate ▯▯▯ Secret ▯▯ `ca.crt` ▯▯▯ Webhook ▯ `caBundle` ▯ API server ▯▯▯ CA ▯▯▯▯▯ Webhook ▯▯▯▯▯▯▯

▯▯ Webhook Deployment ▯▯▯▯▯ CA ▯▯▯▯▯▯▯

```
kubectl rollout status deployment/kube-ovn-webhook -n "<chart-namespace>"
kubectl get validatingwebhookconfiguration kube-ovn-webhook \
  -o jsonpath='{.webhooks[0].clientConfig.caBundle}'
```

▯▯▯▯▯▯

▯▯ Chart ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ Certificate ▯▯▯▯▯▯▯▯ Helm ▯▯▯▯▯▯▯▯▯▯▯▯▯▯ Certificate ▯▯▯▯ Issuer▯ ▯▯▯▯▯▯▯ Certificate ▯▯▯▯▯▯▯▯▯▯ Secret ▯▯▯▯▯▯▯▯▯▯ cert-manager ▯ Certificate ▯▯▯▯▯▯▯▯▯ ▯▯ Helm ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯ Certificate ▯▯▯▯▯▯▯

3.9.4 ▯▯ Webhook ▯▯

▯▯▯▯▯ Pod▯▯▯ Pod IP `10.16.0.15` ▯

```
# kubectl get pod -o wide
NAME                      READY   STATUS    RESTARTS   AGE     IP           NODE              NOMINATED NODE   READINESS GATES
static-7584848b74-fw9dm   1/1     Running   0          2d13h   10.16.0.15   kube-ovn-worker   <none>
```

▯▯ yaml ▯▯▯▯ IP ▯ Pod▯

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    ovn.kubernetes.io/ip_address: 10.16.0.15
    ovn.kubernetes.io/mac_address: 00:00:00:53:6B:B6
  labels:
    app: static
  managedFields:
  name: staticip-pod
  namespace: default
spec:
  containers:
  - image: docker.io/library/nginx:alpine
    imagePullPolicy: IfNotPresent
    name: qatest
```

▯▯▯▯ yaml ▯▯▯▯▯▯ Pod ▯▯▯▯▯▯ IP ▯▯▯▯▯

```
# kubectl apply -f pod-static.yaml
Error from server (annotation static-ip 10.16.0.15 is conflict with ip crd static-7584848b74-fw9dm.default, ip 10.16.0.15): error when creating "pod-
static.yaml": admission webhook "pod-ip-validating.kube-ovn.io" denied the request: annotation static-ip 10.16.0.15 is conflict with ip crd static-7584848b74-
fw9dm.default, ip 10.16.0.15
```

PDF Slack Support

2025 Kube-OVN Team

2026▯10▯25▯

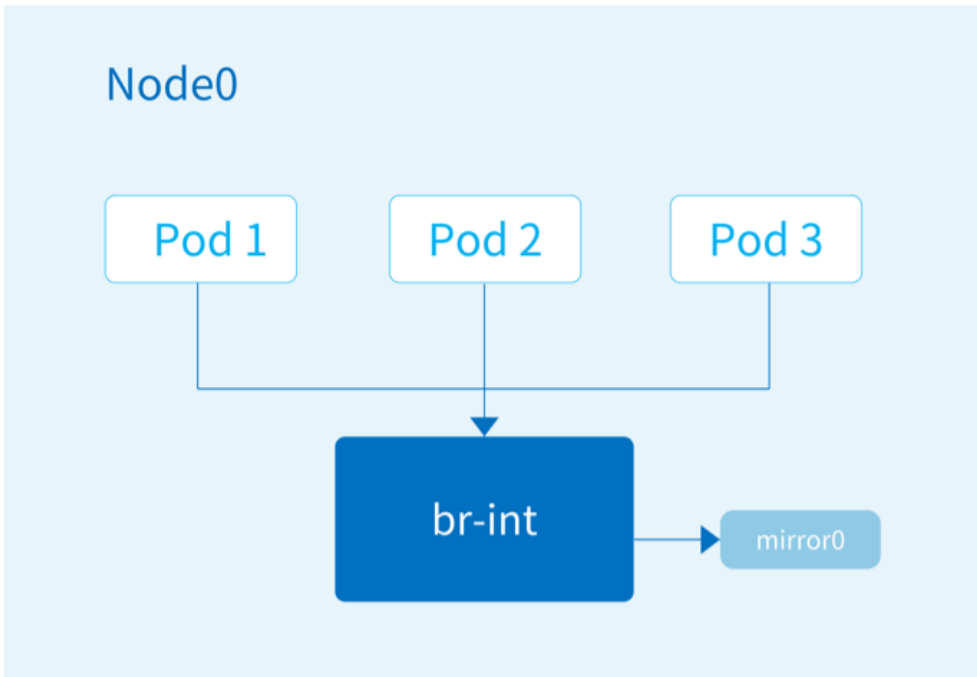
2022▯5▯24▯



GitHub 



3.9.5 ▯▯



3.10 [illegible]

流量镜像功能可以将进出容器网络的数据包进行复制到主机的特定网卡。管理员或开发者 可以通过监听这块网卡获得完整的容器网络流量来进一步进行分析，监控，安全审计等操作。也可和传统的 NPM 对接获取更细粒度的流量监控。

流量镜像功能会带来一定的性能损失，根据 CPU 性能以及流量的特征，会有 5%~10% 的 额外 CPU 消耗。

3.10.1 全局流量镜像配置

流量镜像功能默认为关闭状态，如果需要开启请修改 `kube-ovn-cni` DaemonSet 的启动参数:

- `--enable-mirror=true`：是否开启流量镜像。
- `--mirror-iface=mirror0`：流量镜像所复制到的网卡名。该网卡可为主机上已存在的一块物理网卡，此时该网卡会被桥接进 br-int 网桥，镜像流量会直接接入底层交换机。若网卡名不存在，Kube-OVN 会自动 创建一块同名的虚拟网卡，管理员或开发者可以在宿主机上通过该网卡获取当前节点所有流量。默认为 `mirror0` 。

接下来可以用 tcpdump 或其他流量分析工具监听 `mirror0` 上的流量:

```
tcpdump -ni mirror0
```

3.10.2 Pod 级别流量镜像配置

如果只需对部分 Pod 流量进行镜像，则需要关闭全局的流量镜像功能，然后在特定 Pod 上增加 `ovn.kubernetes.io/mirror` annotation 来开启 Pod 级别流量镜像。

```
apiVersion: v1
kind: Pod
metadata:
  name: mirror-pod
  namespace: ls1
  annotations:
    ovn.kubernetes.io/mirror: "true"
spec:
  containers:
  - name: mirror-pod
    image: docker.io/library/nginx:alpine
```


2025 Kube-OVN Team

3.10.3 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

1. Pod to Pod in the same Nodes

□□□□□□□

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	12.7 us	289 Mbits/sec	12.6 us	(1.8%)	77.9 Mbits/sec
128	15.5 us	517 Mbits/sec	12.7 us	(0%)	155 Mbits/sec
512	12.2 us	1.64 Gbits/sec	12.4 us	(0%)	624 Mbits/sec
1k	13 us	2.96 Gbits/sec	11.4 us	(0.53%)	1.22 Gbits/sec
4k	18 us	7.67 Gbits/sec	25.7 us	(0.41%)	1.50 Gbits/sec

□□□□□□□

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	11.9 us	324 Mbits/sec	12.2 us	(0.22%)	102 Mbits/sec
128	10.5 us	582 Mbits/sec	9.5 us	(0.21%)	198 Mbits/sec
512	11.6 us	1.84 Gbits/sec	9.32 us	(0.091%)	827 Mbits/sec
1k	10.5 us	3.44 Gbits/sec	10 us	(1.2%)	1.52 Gbits/sec
4k	16.7 us	8.52 Gbits/sec	18.2 us	(1.3%)	2.42 Gbits/sec

2. Pod to Pod in the different Nodes

□□□□□□□

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	258 us	143 Mbits/sec	237 us	(61%)	28.5 Mbits/sec
128	240 us	252 Mbits/sec	231 us	(64%)	54.9 Mbits/sec
512	236 us	763 Mbits/sec	256 us	(68%)	194 Mbits/sec
1k	242 us	969 Mbits/sec	225 us	(62%)	449 Mbits/sec
4k	352 us	1.12 Gbits/sec	382 us	(0.71%)	21.4 Mbits/sec

□□□□□□□

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	278 us	140 Mbits/sec	227 us	(24%)	59.6 Mbits/sec
128	249 us	265 Mbits/sec	265 us	(23%)	114 Mbits/sec
512	233 us	914 Mbits/sec	235 us	(21%)	468 Mbits/sec
1k	238 us	1.14 Gbits/sec	240 us	(15%)	891 Mbits/sec
4k	370 us	1.25 Gbits/sec	361 us	(0.43%)	7.54 Mbits/sec

2025 Kube-OVN Team

3. Node to Node

□□□□□□□

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	205 us	162 Mbits/sec	183 us	(11%)	74.2 Mbits/sec
128	222 us	280 Mbits/sec	206 us	(6.3%)	155 Mbits/sec
512	220 us	1.04 Gbits/sec	177 us	(20%)	503 Mbits/sec
1k	213 us	2.06 Gbits/sec	201 us	(8.6%)	1.14 Gbits/sec
4k	280 us	5.01 Gbits/sec	315 us	(37%)	1.20 Gbits/sec

□□□□□□□

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	204 us	157 Mbits/sec	204 us	(8.8%)	81.9 Mbits/sec
128	213 us	262 Mbits/sec	225 us	(19%)	136 Mbits/sec
512	220 us	1.02 Gbits/sec	227 us	(21%)	486 Mbits/sec
1k	217 us	1.79 Gbits/sec	218 us	(29%)	845 Mbits/sec
4k	275 us	5.27 Gbits/sec	336 us	(34%)	1.21 Gbits/sec

4. Pod to the Node where the Pod is located

□□□□□□□

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	12.2 us	295 Mbits/sec	12.7 us	(0.27%)	74.1 Mbits/sec
128	14.1 us	549 Mbits/sec	10.6 us	(0.41%)	153 Mbits/sec
512	13.5 us	1.83 Gbits/sec	12.7 us	(0.23%)	586 Mbits/sec
1k	12 us	2.69 Gbits/sec	13 us	(1%)	1.16 Gbits/sec
4k	18.9 us	4.51 Gbits/sec	21.8 us	(0.42%)	1.81 Gbits/sec

□□□□□□□

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	10.4 us	335 Mbits/sec	12.2 us	(0.75%)	95.4 Mbits/sec
128	12.1 us	561 Mbits/sec	11.3 us	(0.25%)	194 Mbits/sec
512	11.6 us	1.87 Gbits/sec	10.7 us	(0.66%)	745 Mbits/sec
1k	12.7 us	3.12 Gbits/sec	10.9 us	(1.2%)	1.46 Gbits/sec
4k	16.5 us	8.23 Gbits/sec	17.9 us	(1.5%)	2.51 Gbits/sec

2025 Kube-OVN Team

5. Pod to the Node where the Pod is not located

▯▯▯▯▯▯▯

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	234 us	153 Mbits/sec	232 us	(63%)	29.4 Mbits/sec
128	237 us	261 Mbits/sec	238 us	(49%)	76.1 Mbits/sec
512	231 us	701 Mbits/sec	238 us	(57%)	279 Mbits/sec
1k	256 us	1.05 Gbits/sec	228 us	(56%)	524 Mbits/sec
4k	330 us	1.08 Gbits/sec	359 us	(1.5%)	35.7 Mbits/sec

▯▯▯▯▯▯▯

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	283 us	141 Mbits/sec	230 us	(26%)	55.8 Mbits/sec
128	234 us	255 Mbits/sec	234 us	(25%)	113 Mbits/sec
512	246 us	760 Mbits/sec	234 us	(22%)	458 Mbits/sec
1k	268 us	1.23 Gbits/sec	242 us	(20%)	879 Mbits/sec
4k	326 us	1.20 Gbits/sec	369 us	(0.5%)	7.87 Mbits/sec

6. Pod to the cluster ip service

▯▯▯▯▯▯▯

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	237 us	133 Mbits/sec	213 us	(65%)	25.5 Mbits/sec
128	232 us	271 Mbits/sec	222 us	(62%)	54.8 Mbits/sec
512	266 us	800 Mbits/sec	234 us	(60%)	232 Mbits/sec
1k	248 us	986 Mbits/sec	239 us	(50%)	511 Mbits/sec
4k	314 us	1.03 Gbits/sec	367 us	(0.6%)	13.2 Mbits/sec

TCP-Conn-Number	QPS	Avg-Resp-Time	Stdev-Resp-Time	Max-Resp-Time
10	14305.17	0.87ms	1.48ms	24.46ms
100	29082.07	3.87ms	4.35ms	102.85ms

▯▯▯▯▯▯▯

Size	TCP Latency	TCP Bandwidth	UDP Latency	UDP Lost Rate	UDP Bandwidth
64	241 us	145 Mbits/sec	225 us	(19%)	60.2 Mbits/sec
128	245 us	261 Mbits/sec	212 us	(15%)	123 Mbits/sec
512	252 us	821 Mbits/sec	219 us	(14%)	499 Mbits/sec
1k	253 us	1.08 Gbits/sec	242 us	(16%)	852 Mbits/sec
4k	320 us	1.32 Gbits/sec	360 us	(0.47%)	6.70 Mbits/sec

TCP-Conn-Number	QPS	Avg-Resp-Time	Stdev-Resp-Time	Max-Resp-Time
10	13634.07	0.96ms	1.72ms	30.07ms
100	30215.23	3.59ms	3.20ms	77.56ms

7. Host to the Node port service where the Pod is not located on the target Node

▯▯▯▯▯▯▯

TCP-Conn-Number	QPS	Avg-Resp-Time	Stdev-Resp-Time	Max-Resp-Time
10	14802.73	0.88ms	1.66ms	31.49ms
100	29809.58	3.78ms	4.12ms	105.34ms

▯▯▯▯▯▯▯

TCP-Conn-Number	QPS	Avg-Resp-Time	Stdev-Resp-Time	Max-Resp-Time
10	14273.33	0.90ms	1.60ms	37.16ms
100	30757.81	3.62ms	3.41ms	59.78ms

8. Host to the Node port service where the Pod is located on the target Node

▯▯▯▯▯▯▯

TCP-Conn-Number	QPS	Avg-Resp-Time	Stdev-Resp-Time	Max-Resp-Time
10	15402.39	802.50us	1.42ms	30.91ms
100	29424.66	4.05ms	4.31ms	90.60ms

▯▯▯▯▯▯▯

TCP-Conn-Number	QPS	Avg-Resp-Time	Stdev-Resp-Time	Max-Resp-Time
10	14649.21	0.91ms	1.72ms	43.92ms
100	32143.61	3.66ms	3.76ms	67.02ms

PDF Slack Support

2025□11□20□

2022□5□24□

GitHub 

3.10.4 □□

3.11 LoadBalancer [illegible] Service

Kube-OVN [illegible] VPC [illegible] VPC [illegible] VPC [illegible]

[illegible] VPC [illegible] VPC [illegible] VPC [illegible] `LoadBalancer` [illegible] Service[illegible] LoadBalancerIP [illegible] VPC [illegible] Service[illegible]

[illegible]

1. [illegible] `multus-cni` [illegible] `macvlan cni` [illegible]
2. LoadBalancer Service [illegible] VPC [illegible] `vpc-nat-gw` [illegible] macvlan [illegible]
3. [illegible] `VPC` [illegible] VPC [illegible] LoadBalancer [illegible] VPC [illegible] VPC [illegible]

3.11.1 [illegible] VPC LoadBalancer Service [illegible]

[illegible]

[illegible] kube-system namespace [illegible] deployment `kube-ovn-controller` [illegible] `args` [illegible] `--enable-lb-svc=true` [illegible] false[illegible]

```
containers:
- args:
  - /kube-ovn/start-controller.sh
  - --default-cidr=10.16.0.0/16
  - --default-gateway=10.16.0.1
  - --default-gateway-check=true
  - --enable-lb-svc=true                  #       true
```

[illegible] NetworkAttachmentDefinition CRD [illegible]

[illegible] yaml[illegible] `net-attach-def` [illegible]

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: lb-svc-attachment
  namespace: kube-system
spec:
  # master                    eth0 / ens192 / bond0
  config: '{
      "cniVersion": "0.3.0",
      "type": "macvlan",
      "master": "eth0",
      "mode": "bridge"
    }'
```

[illegible] `eth0` [illegible] `master` [illegible]

[illegible] Subnet

[illegible] Subnet[illegible] LoadBalancer Service [illegible] LoadBalancerIP[illegible] Underlay Subnet [illegible]

[illegible] yaml[illegible]

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: attach-subnet
spec:
  protocol: IPv4
  provider: lb-svc-attachment.kube-system    # provider             net-attach-def     Name.Namespace
  cidrBlock: 172.18.0.0/16
  gateway: 172.18.0.1
  excludeIps:
  - 172.18.0.0..172.18.0.10
```

Subnet [illegible] `provider` [illegible] `ovn` [illegible] `.ovn` [illegible] Kube-OVN [illegible] `logical switch` [illegible]

`provider` [illegible] `ovn` [illegible] `.ovn` [illegible] Kube-OVN [illegible] IPAM [illegible] IP [illegible]

2025 Kube-OVN Team

□□ LoadBalancer Service

□□□□ yaml□□□ LoadBalancer Service□

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    lb-svc-attachment.kube-system.kubernetes.io/logical_switch: attach-subnet   #
    ovn.kubernetes.io/attachmentprovider: lb-svc-attachment.kube-system          #
  labels:
    app: dynamic
  name: test-service
  namespace: default
spec:
  loadBalancerIP: 172.18.0.18                                                    #
  ports:
    - name: test
      protocol: TCP
      port: 80
      targetPort: 80
  selector:
    app: dynamic
  sessionAffinity: None
  type: LoadBalancer
```

□ yaml □□annotation `ovn.kubernetes.io/attachmentprovider` □□□□□□□□□□□□□□□ `net-attach-def` □□□ Name.Namespace □□□□ annotation □□□□□ Pod □□ □□ `net-attach-def` □□□

□□□□ annotation □□□□□□□□□□□□□□□□annotation key □□□ net-attach-def □□□ `Name.Namespace.kubernetes.io/logical_switch` □□□□□ □□ □□□□□□□□ LoadBalancerIP □□□□□□□□□□□□□□□□□□□□□□□□□ LoadBalancerIP □□□

□□□□□□□□ LoadBalancerIP □□□□□□□□ `spec.loadBalancerIP` □□□□□□□□□□□□□□□□□□□□□□□

□□□ yaml □□ Service □□□ Service □ Namespace □□□□□□ Pod □□□□□

```
# kubectl get pod
NAME                                    READY   STATUS    RESTARTS   AGE
lb-svc-test-service-6869d98dd8-cjvll    1/1     Running   0          107m
# kubectl get svc
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
test-service     LoadBalancer   10.109.201.193   172.18.0.18   80:30056/TCP   107m
```

□□ `service.spec.loadBalancerIP` □□□□□□□□□□□□□□□ service external-ip □□□□□□□□□□□□□□□□□□□□□□□□

□□□□ Pod □ yaml □□□□□□□□□□□□□□□□□□□

```
# kubectl get pod -o yaml lb-svc-test-service-6869d98dd8-cjvll
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.v1.cni.cncf.io/network-status: |-
      [{
          "name": "kube-ovn",
          "ips": [
              "10.16.0.2"
          ],
          "default": true,
          "dns": {}
      },{
          "name": "default/test-service",
          "interface": "net1",
          "mac": "ba:85:f7:02:9f:42",
          "dns": {}
      }]
    k8s.v1.cni.cncf.io/networks: default/test-service
    k8s.v1.cni.cncf.io/networks-status: |-
      [{
          "name": "kube-ovn",
          "ips": [
              "10.16.0.2"
          ],
          "default": true,
          "dns": {}
      },{
          "name": "default/test-service",
          "interface": "net1",
          "mac": "ba:85:f7:02:9f:42",
          "dns": {}
      }]
    ovn.kubernetes.io/allocated: "true"
    ovn.kubernetes.io/cidr: 10.16.0.0/16
    ovn.kubernetes.io/gateway: 10.16.0.1
```

```
    ovn.kubernetes.io/ip_address: 10.16.0.2
    ovn.kubernetes.io/logical_router: ovn-cluster
    ovn.kubernetes.io/logical_switch: ovn-default
    ovn.kubernetes.io/mac_address: 00:00:00:45:F4:29
    ovn.kubernetes.io/pod_nic_type: veth-pair
    ovn.kubernetes.io/routed: "true"
    test-service.default.kubernetes.io/allocated: "true"
    test-service.default.kubernetes.io/cidr: 172.18.0.0/16
    test-service.default.kubernetes.io/gateway: 172.18.0.1
    test-service.default.kubernetes.io/ip_address: 172.18.0.18
    test-service.default.kubernetes.io/logical_switch: attach-subnet
    test-service.default.kubernetes.io/mac_address: 00:00:00:AF:AA:BF
    test-service.default.kubernetes.io/pod_nic_type: veth-pair
```

□□ Service □□□□

```
# kubectl get svc -o yaml test-service
apiVersion: v1
kind: Service
metadata:
  annotations:
    ovn.kubernetes.io/vpc: ovn-cluster
    test-service.default.kubernetes.io/logical_switch: attach-subnet
  creationTimestamp: "2022-06-15T09:01:58Z"
  labels:
    app: dynamic
  name: test-service
  namespace: default
  resourceVersion: "38485"
  uid: 161edee1-7f6e-40f5-9e09-5a52c44267d0
spec:
  allocateLoadBalancerNodePorts: true
  clusterIP: 10.109.201.193
  clusterIPs:
  - 10.109.201.193
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: test
    nodePort: 30056
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: dynamic
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 172.18.0.18
```

3.11.2 □□ LoadBalancerIP □□

□□□□ yaml, □□□□ Pod□□□ Service □ Endpoints □□□□□

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: dynamic
  name: dynamic
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: dynamic
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: dynamic
    spec:
      containers:
      - image: docker.io/library/nginx:alpine
        imagePullPolicy: IfNotPresent
        name: nginx
      dnsPolicy: ClusterFirst
      restartPolicy: Always
```

Service LoadBalancerIP:Port

```
# curl 172.18.0.18:80
<html>
<head>
        <title>Hello World!</title>
        <link href='//fonts.googleapis.com/css?family=Open+Sans:400,700' rel='stylesheet' type='text/css'>
        <style>
        body {
                background-color: white;
                text-align: center;
                padding: 50px;
                font-family: "Open Sans","Helvetica Neue",Helvetica,Arial,sans-serif;
        }
        #logo {
                margin-bottom: 40px;
        }
        </style>
</head>
<body>
                <h1>Hello World!</h1>
                                <h3>Links found</h3>
        <h3>I am on  dynamic-7d8d7874f5-hsgc4</h3>
        <h3>Cookie                  =</h3>
                                    <b>KUBERNETES</b> listening in 443 available at tcp://10.96.0.1:443<br />
                                        <h3>my name is hanhouchao!</h3>
                        <h3> RequestURI='/'</h3>
</body>
</html>
```

Service Pod

```
# ip a
4: net1@if62: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether ba:85:f7:02:9f:42 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.18/16 scope global net1
       valid_lft forever preferred_lft forever
    inet6 fe80::b885:f7ff:fe02:9f42/64 scope link
       valid_lft forever preferred_lft forever
36: eth0@if37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP group default
    link/ether 00:00:00:45:f4:29 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.16.0.2/16 brd 10.16.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe45:f429/64 scope link
       valid_lft forever preferred_lft forever

# ip rule
0: from all lookup local
32764: from all iif eth0 lookup 100
32765: from all iif net1 lookup 100
32766: from all lookup main
32767: from all lookup default

# ip route show table 100
default via 172.18.0.1 dev net1
10.109.201.193 via 10.16.0.1 dev eth0
172.18.0.0/16 dev net1 scope link

# iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            172.18.0.18          tcp dpt:80 to:10.109.201.193:80

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      *       0.0.0.0/0            10.109.201.193
```

lb service Pod nodeSelector

ovn-vpc-nat-config ConfigMap nodeSelector LoadBalancer service Pod

```
apiVersion: v1
data:
  image: docker.io/kubeovn/vpc-nat-gateway:v1.17.0
  nodeSelector: |
    kubernetes.io/hostname: kube-ovn-control-plane
    kubernetes.io/os: linux
kind: ConfigMap
metadata:
  name: ovn-vpc-nat-config
  namespace: kube-system
```


2025 Kube-OVN Team

PDF Slack Support

2026⬚11⬚3⬚

2022⬚7⬚19⬚



GitHub 

+3

3.11.3 ⬚⬚

3.12 ████████

Kube-OVN ████████████████████████████████ Prometheus ████████████

████ kube-prometheus ████ CRD ███████ Prometheus ██████ ████████ kube-prometheus ███████ CRD▯Kube-OVN ████████████████ Kube-OVN ████▯

3.12.1 ▯▯ ServiceMonitor

Kube-OVN ▯▯ `ServiceMonitor` CRD ████████

```
#
kubectl apply -f https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/pinger-monitor.yaml
# kube-ovn-controller
kubectl apply -f https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/controller-monitor.yaml
# kube-ovn-cni
kubectl apply -f https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/cni-monitor.yaml
# ovn
kubectl apply -f https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/ovn-monitor.yaml
```

Prometheus ████████████ 15s████████████ yaml ▯▯ `interval` ███

3.12.2 ▯▯ kube-ovn-v2 Helm Chart ███████

`kube-ovn-v2` Helm Chart ████████ `PrometheusRule` ▯ Grafana Dashboard ConfigMaps▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ Prometheus Operator CRDs▯

▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `monitoring` ▯▯▯▯▯

```
prometheusRule:
  enabled: true
  namespace: monitoring
  labels:
    release: kube-prometheus-stack
grafanaDashboards:
  enabled: true
  namespace: monitoring
  datasource: Prometheus
```

`prometheusRule.namespace` ▯ `grafanaDashboards.namespace` ▯▯▯▯▯▯▯▯ `.Values.namespace` ▯▯▯▯▯ `monitoring` ▯▯▯▯▯▯▯▯▯▯▯▯ `helm --namespace kube-system` ▯▯▯ Helm release ▯▯▯▯▯▯▯▯▯▯▯▯▯ override ▯▯▯▯▯▯▯▯▯▯▯▯ `ServiceMonitor` ▯▯▯▯▯▯▯ Chart ▯▯▯▯▯▯ `.Values.namespace` ▯▯▯▯▯▯

▯▯▯▯▯▯▯▯▯▯▯▯▯

```
helm upgrade --install kube-ovn oci://ghcr.io/kubeovn/charts/kube-ovn-v2 \
  --namespace kube-system \
  --values monitoring-values.yaml
```

`prometheusRule` ▯▯ controller▯pinger▯subnet ▯ OVSDB ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ controller▯OVN monitor▯pinger ▯ agent ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ServiceMonitor ▯ ▯▯ `controller.serviceMonitor.enabled` ▯ `monitor.serviceMonitor.enabled` ▯ `pinger.serviceMonitor.enabled` ▯ `agent.serviceMonitor.enabled` ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ job ▯ metric labels▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

`KubeOvnControllerAbsent` ▯▯ `absent(up{job="kube-ovn-controller"})` ▯▯▯▯▯▯▯▯ target▯▯▯▯▯▯▯▯ 15m ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ Alertmanager ▯ silence/inhibit ▯▯▯▯▯▯▯▯ `prometheusRule.enabled` ▯▯▯▯▯▯▯ `prometheusRule.additionalGroups` ▯▯▯▯▯ `PrometheusRule.spec.groups[]` ▯▯▯▯▯▯▯▯▯▯

`grafanaDashboards.datasource` ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ Prometheus ▯▯▯▯▯▯▯▯▯▯▯ `Prometheus` ▯Chart ▯▯▯▯▯ Dashboard ▯▯▯▯▯ `${DS_PROMETHEUS}` ▯▯▯▯▯▯ ▯▯▯▯▯▯

▯▯ `grafanaDashboards.enabled` ▯▯Helm ▯▯▯▯▯▯ `grafana_dashboard: "1"` ▯▯▯ Dashboard ConfigMaps▯▯▯▯ Grafana sidecar ▯▯▯▯▯▯▯▯▯▯ sidecar ▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯ Dashboard ▯▯▯

3.12.3 ▯▯ Grafana ▯▯

Kube-OVN ▯▯▯▯▯▯▯▯▯▯▯ Grafana Dashboard ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

▯▯▯▯ Dashboard ▯▯▯

2025 Kube-OVN Team

```
#
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/pinger-grafana.json
# kube-ovn-controller
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/controller-grafana.json
# kube-ovn-cni
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/cni-grafana.json
# ovn
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/ovn-grafana.json
# ovs
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/master/dist/monitoring/ovs-grafana.json
```

▒ Grafana ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ Prometheus ▒▒▒▒▒▒▒ Dashboard▒

`kube-ovn-controller` ▒▒▒▒▒▒▒▒▒▒



`kube-ovn-pinger`



`kube-ovn-cni`



PDF Slack Support

2025 Kube-OVN Team

2026▯10▯29▯

2022▯5▯23▯

GitHub 

3.12.4 ▯▯

3.13 ⬚⬚

3.13.1 NetworkPolicy ⬚⬚

NetworkPolicy 是 Kubernetes 提供的一种网络策略接口，用于控制 Pod 之间以及 Pod 与其他网络端点之间的网络流量。Kube-OVN 通过 OVN 的 ACL 机制来实现了 Kubernetes NetworkPolicy 的规范，在这之上提供了两种不同严格程度的规则模式来适应不同场景以及 NetworkPolicy 日志功能来排查规则问题。

Kube-OVN 中的实现机制

Kube-OVN 通过 OVN（Open Virtual Network）的原生能力来实现 NetworkPolicy，主要使用了三个 OVN 组件：

PORT GROUP（端口组）

当创建一个 NetworkPolicy 时，Kube-OVN 会为该策略创建一个 Port Group，用于将所有匹配 `podSelector` 的 Pod 的逻辑端口组织在一起。Port Group 的名称格式为 `<  >.<  >`，其中 `-` 会被替换为 `.`。

ADDRESS SET（地址集）

Address Set 用于存储 NetworkPolicy 规则中定义的允许或拒绝访问的 IP 地址集合。对于每个 NetworkPolicy 规则，Kube-OVN 会根据 `podSelector`、`namespaceSelector` 或 `ipBlock` 计算出对应的 IP 地址，并存储在 Address Set 中。

为每个 NetworkPolicy 的 Ingress 和 Egress 规则分别创建 Allow 和 Except 两类 Address Set：

- `<  >.<  >.ingress.allow`：允许的入向 IP 地址
- `<  >.<  >.ingress.except`：排除的入向 IP 地址
- `<  >.<  >.egress.allow`：允许的出向 IP 地址
- `<  >.<  >.egress.except`：排除的出向 IP 地址

ACL（访问控制列表）

ACL 是 OVN 中实际执行流量控制的组件。Kube-OVN 将 NetworkPolicy 规则转换为 OVN ACL 规则，并关联到对应的 Port Group 上。ACL 规则包含匹配条件、优先级和动作（允许或拒绝）。

Kube-OVN 为 NetworkPolicy 使用的 ACL 优先级范围：

- Ingress Allow 规则：优先级 2001
- Egress Allow 规则：优先级 2001
- Default Deny 规则：优先级 2000

通过这种方式，Kube-OVN 能够高效地实现 Kubernetes NetworkPolicy 的语义，并充分利用 OVN 的分布式 ACL 能力。

注意事项

与其他访问控制机制的关系

Kube-OVN 同时支持多种网络访问控制机制：

- **NetworkPolicy**：Kubernetes 标准的网络策略
- **Network Policy API**：AdminNetworkPolicy 和 BaselineAdminNetworkPolicy
- **Subnet ACL**：子网级别的访问控制
- **Security Group**：安全组

这些机制在底层都是通过 OVN ACL 实现的。虽然 NetworkPolicy 和 Network Policy API 在设计时考虑了规则分层以避免优先级冲突，但同时使用多种访问控制机制可能导致规则管理复杂和优先级冲突。建议不要同时使用多种访问控制规则。

NAMED PORT

NetworkPolicy 规范支持使用 Named Port 来指定端口，例如：

```
ports:
- protocol: TCP
  port: http
```


2025 Kube-OVN Team

Kube-OVN 根据每个目标 Pod 和规则指定的协议解析 Named Port。例如，两个被选中的 Pod 可以分别将 `http` 声明为 TCP 8080 和 TCP 8081，规则会匹配各自的端口。同一个名称也可以用于不同协议的端口。Ingress 使用策略选中的目标 Pod；Egress 使用规则选中的目标 Pod，包括其他命名空间中的 Pod。如果目标没有声明匹配名称和协议的端口，Named Port 规则不会匹配该目标。仅有 `ipBlock` 的 Egress 规则无法确定用于解析 Named Port 的目标 Pod。

AdminNetworkPolicy 和 BaselineAdminNetworkPolicy 在 Ingress 和 Egress 中使用 `ports[].namedPort`，ClusterNetworkPolicy 使用 `protocols[].destinationNamedPort`。这些策略同样针对被选中的目标 Pod 解析 Named Port。CNP 会匹配该名称对应的所有协议；`destinationNamedPort` 不能与 TCP、UDP 或 SCTP 协议字段组合。Node、Network 和域名类型的 Peer 不选择目标 Pod，因此无法解析 Named Port。

IPBLOCK EXCEPT 规则的性能影响

NetworkPolicy 的 `ipBlock` 规则支持 `except` 字段来排除特定的 IP 地址段：

```
egress:
- to:
  - ipBlock:
      cidr: 10.0.0.0/8
      except:
      - 10.0.1.0/24
      - 10.0.2.0/24
```

在 OVN 的流表实现中，`except` 规则会导致流表的显著膨胀。每个 `except` 子网都需要额外的 ACL 规则来实现，这会增加 OVN 数据库的大小和流表处理的复杂度，对网络性能产生负面影响。

建议尽可能避免使用 **`except`** 规则。如果必须排除某些 IP 地址段，考虑以下替代方案：

- 将 CIDR 拆分为多个不重叠的较小网段，直接指定允许的网段
- 使用更精确的 `podSelector` 或 `namespaceSelector` 来替代 IP 地址过滤

NetworkPolicy 日志

Kube-OVN 提供了 NetworkPolicy 日志功能，可以帮助管理员快速定位网络策略规则是否生效，以及排查网络连通性问题。

> **Warning**
>
> NetworkPolicy 日志功能一旦开启，对每个命中规则的数据包都需要打印日志，会带来额外的性能开销。在恶意攻击场景下，短时间大量日志可能会耗尽 CPU 资源。
>
> 建议在生产环境中默认关闭日志功能，仅在需要排查问题时动态开启。
>
> 可以通过 `ovn.kubernetes.io/acl_log_meter_rate` annotation 来限制 ACL 日志生成速度，避免日志过多导致的性能问题。

启用日志记录

在需要开启日志记录的 NetworkPolicy 中增加 annotation `ovn.kubernetes.io/enable_log`：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-ingress
  namespace: default
  annotations:
    ovn.kubernetes.io/enable_log: "true"
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

启用日志后，默认只会记录被拒绝（Drop）的流量日志。

查看日志

日志会记录在 Pod 所在节点的 `/var/log/ovn/ovn-controller.log` 文件中：

```
# tail -f /var/log/ovn/ovn-controller.log
2022-07-20T05:55:03.229Z|00394|acl_log(ovn_pinctrl0)|INFO|name="np/default-deny-ingress.default/IPv4/0", verdict=drop, severity=warning, direction=to-lport:
udp,vlan_tci=0x0000,dl_src=00:00:00:21:b7:d1,dl_dst=00:00:00:8d:0b:86,nw_src=10.16.0.10,nw_dst=10.16.0.7,nw_tos=0,nw_ecn=0,nw_ttl=63,tp_src=54343,tp_dst=53
```

日志中包含了详细的五元组信息（源 IP、目的 IP、协议、源端口、目的端口），方便进行问题排查。

2025 Kube-OVN Team

[illegible]

[illegible] Kube-OVN v1.13.0 [illegible] `ovn.kubernetes.io/log_acl_actions` annotation [illegible]Allow[illegible]

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-client
  namespace: default
  annotations:
    ovn.kubernetes.io/enable_log: "true"
    ovn.kubernetes.io/log_acl_actions: "allow"
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: client
```

`ovn.kubernetes.io/log_acl_actions` [illegible]

- `drop` [illegible]
- `allow` [illegible]
- `allow,drop` [illegible]

[illegible]

```
# tail -f /var/log/ovn/ovn-controller.log
2024-08-14T09:27:49.590Z|00004|acl_log(ovn_pinctrl0)|INFO|name="np/allow-from-client.default/ingress/IPv4/0", verdict=allow, severity=info, direction=to-lport: icmp,vlan_tci=0x0000,dl_src=96:7b:b0:2f:a0:1a,dl_dst=a6:e5:1b:c2:1b:f8,nw_src=10.16.0.7,nw_dst=10.16.0.12,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icm
```

[illegible]

[illegible] `ovn.kubernetes.io/acl_log_meter_rate` annotation [illegible]

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-client
  namespace: default
  annotations:
    ovn.kubernetes.io/enable_log: "true"
    ovn.kubernetes.io/log_acl_actions: "allow"
    ovn.kubernetes.io/acl_log_meter_rate: "100"
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: client
```

`ovn.kubernetes.io/acl_log_meter_rate` [illegible]/[illegible] `100` [illegible] 100 [illegible]

[illegible]

[illegible] annotation `ovn.kubernetes.io/enable_log` [illegible] `false` [illegible]

```
kubectl annotate networkpolicy -n default allow-from-client ovn.kubernetes.io/enable_log=false --overwrite
```

[illegible]

Kube-OVN [illegible]

- **standard**[illegible] NetworkPolicy [illegible] IP [illegible]
- **lax**[illegible] TCP/UDP/SCTP [illegible] ICMP [illegible] L4 [illegible] IP [illegible] DHCP [illegible] UDP [illegible]

2025 Kube-OVN Team

⬚⬚⬚⬚⬚ NetworkPolicy ⬚⬚⬚ annotation ⬚⬚⬚⬚⬚⬚⬚

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: example-policy
  namespace: default
  annotations:
    ovn.kubernetes.io/network_policy_enforcement: "lax"
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚ `--np-enforcement` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

⬚⬚⬚⬚

- ACL ⬚⬚
- ⬚⬚⬚ NetworkPolicy

PDF Slack Support

2026⬚11⬚3⬚

2025⬚10⬚23⬚

GitHub

+1

⬚⬚

3.13.2 Network Policy API

Kube-OVN 支持 `network-policy-api v0.3.0` 的 `v1beta1` 版本的 `ClusterNetworkPolicy` 和CNP和API。支持 ANP/CNP 中的 DNS 名称匹配 和[illegible]DNS resolver 。

支持 `protocols` 字段

CNP 规则中可以使用 `protocols` 来限制流量协议：

```
protocols:
  - tcp:
      destinationPort:
        number: 443
  - udp:
      destinationPort:
        range:
          start: 8000
          end: 9000
```

支持 TCP、UDP 和 SCTP 协议，可以使用 `destinationNamedPort` 匹配命名端口。Pod 规则支持匹配 Pod 本身、Pod 所在的 Namespace 或所在Node的 Network 地址作为 peer 来匹配 Pod。

支持 `protocols` 字段的规则可以匹配 peer 和 DNS resolver，但不能使用 `Accept` 。

支持 **Tier**

`spec.priority` 字段可以设置 `0` 到 `399` 之间的整数来指定 Tier 中的 CNP 规则的 priority，controller 会按照 ACL中ANP/BANP 的 priority 顺序将 Network Policy API 和 原生的 CNP 中的 priority 统一排序。

其他变更

升级 `network-policy-api v0.3.0` 后 controller 和 CRD 会将原有 CNP 中的 `ports` 和 `protocols` 字段合并到 `ports` 字段。

v0.3.0 CRD 中的 `v1alpha2` 和 `v1beta1` 中的 `v1beta1` 版本与 v0.3.0 的 `protocols` schema 一致，conversion webhook不会修改 schema 。`v1alpha2` 版本的 `v1beta1` 版本的 CRD 中的 `ports` 字段和 `protocols` 在v1.16 版本会合并 `ports` 。CNP 中的 `apiVersion: v1alpha2` 将会被废弃。

ANP 和 BANP 的 `v1alpha1` CRD 和 `network-policy-api v0.3.0` 中的 CNP conformance profile中的 ANP/BANP 测试用例

PDF Slack Support

2026年11月7日

2026年11月7日

GitHub

2025 Kube-OVN Team

3.13.3 □□□ NetworkPolicy ACL □□

Kube-OVN 可以采样为 `networking.k8s.io/v1` `NetworkPolicy` 生成的 OVN ACL 决策。此功能处于实验阶段，默认关闭，适用于短期故障排查和可观测性场景，不应替代执行日志或通用报文采集器。

初始实现区分两类决策：

- `allow`：针对某条入向或出向 NetworkPolicy 规则生成的有状态 `allow-related` ACL；
- `default-deny`：没有允许规则匹配时生效的生成式丢弃 ACL。

`default-deny` 事件可能会记录拥有被采样 ACL 的 NetworkPolicy，但这种归属并不具有排他性。当多个策略选择同一个 Pod 时，采样到的 default-deny ACL 不能证明是某一条策略导致了拒绝。因此 default-deny 事件不包含规则索引。

要求和限制

控制器侧 ACL 采样要求 OVN 24.09 或更高版本。节点本地传输还要求满足以下全部条件：

- OVS 3.4 或更高版本；
- 使用 Linux 内核 datapath，而不是 userspace 或 DPDK datapath；
- 当前 OVS datapath 能力中存在 `psample=true`；
- OVS schema 包含 `Flow_Sample_Collector_Set.local_group_id`；
- Linux 内核提供 `psample` generic-netlink family。

只有为 Kubernetes NetworkPolicy 规则允许和 NetworkPolicy 默认拒绝生成的 Kube-OVN ACL 可以采样。不包含 DHCP 例外、网关和节点 ACL、SecurityGroup、AdminNetworkPolicy、BaselineAdminNetworkPolicy、ClusterNetworkPolicy 以及其他 ACL 生成方。

采样不会刷新 conntrack。启用此功能后，已有连接仍遵循 OVN 当前的 conntrack 状态，可能不会产生新的会话采样。

启用功能

两个 Helm chart 提供相同的 `aclSampling` 配置。推荐的 v2 chart 可以使用以下示例：

```
aclSampling:
  enabled: true
  setID: 142
  localGroupID: 142
  appIDNew: 102
  appIDEstablished: 103
  collectorIDAllow: 1
  collectorIDDefaultDeny: 2
  allowProbabilityPercent: 1
  defaultDenyProbabilityPercent: 100
```

```
helm upgrade --install kube-ovn ./charts/kube-ovn-v2 \
  --set aclSampling.enabled=true
```

如需配置 classic chart，请使用 `./charts/kube-ovn`。在拆分部署或独立管理的部署中，控制器和 CNI agent 的配置必须保持一致。

对于基于 manifest 的安装器，请设置 ACL 采样环境变量后运行安装命令。安装器默认值与上面的 Helm 示例一致：

```
ENABLE_ACL_SAMPLING=true \
ACL_SAMPLING_SET_ID=142 \
ACL_SAMPLING_LOCAL_GROUP_ID=142 \
ACL_SAMPLING_APP_ID_NEW=102 \
ACL_SAMPLING_APP_ID_ESTABLISHED=103 \
ACL_SAMPLING_COLLECTOR_ID_ALLOW=1 \
ACL_SAMPLING_COLLECTOR_ID_DEFAULT_DENY=2 \
ACL_SAMPLING_ALLOW_PROBABILITY_PERCENT=1 \
ACL_SAMPLING_DEFAULT_DENY_PROBABILITY_PERCENT=100 \
bash dist/images/install.sh
```

这些变量是 `ENABLE_ACL_SAMPLING`、`ACL_SAMPLING_SET_ID`、`ACL_SAMPLING_LOCAL_GROUP_ID`、`ACL_SAMPLING_APP_ID_NEW`、`ACL_SAMPLING_APP_ID_ESTABLISHED`、`ACL_SAMPLING_COLLECTOR_ID_ALLOW`、`ACL_SAMPLING_COLLECTOR_ID_DEFAULT_DENY`、`ACL_SAMPLING_ALLOW_PROBABILITY_PERCENT` 和 `ACL_SAMPLING_DEFAULT_DENY_PROBABILITY_PERCENT`。需要使用其他 collector set 或采样速率时可以覆盖这些值。

2025 Kube-OVN Team

配置字段的语义如下：

字段	约束	用途
`enabled`	布尔值	启用控制器对象、ACL 挂载和节点本地传输。
`setID`	非零无符号 32 位整数	共享 OVN collector set 和 OVS collector-set ID。
`localGroupID`	无符号 32 位整数，允许为零	OVS collector set 携带的 Linux psample group。
`appIDNew`	1 至 255 的唯一整数	`acl-new` 采样的 application 字节。
`appIDEstablished`	1 至 255 的唯一整数	`acl-est` 采样的 application 字节。
`collectorIDAllow`	1 至 255 的唯一整数	allow 决策使用的 collector。
`collectorIDDefaultDeny`	1 至 255 的唯一整数	default-deny 决策使用的 collector。
`allowProbabilityPercent`	0 至 100 的数字	allow 决策的采样概率。
`defaultDenyProbabilityPercent`	0 至 100 的数字	default-deny 决策的采样概率。

Kube-OVN 使用 `round(percent * 65535 / 100)` 将百分比转换为 OVN 的 16 位概率。概率为零时不会为对应决策挂载采样。default-deny 默认概率为 100%，在扫描或攻击期间可能产生大量事件；长时间使用时应降低该值。

配置的 ID 不得与其他应用拥有的对象冲突。Kube-OVN 可以复用匹配的未拥有 `Sampling_App`，但不会修改或删除它。未拥有的 application、collector 或节点 collector set 发生冲突时，采样不可用，并且这些对象保持不变。

解码和监听

镜像包含独立的 `/kube-ovn/kube-ovn-acl-sample` binary，提供两个子命令。该 binary 与 `kube-ovn` CNI plugin 分开构建。`kubectl ko acl` 会在选定的 Pod 中调用此 binary；执行 `decode` 时，wrapper 会自动补充 OVN northbound socket 地址。

解码十进制或带 `0x` 前缀的十六进制采样 metadata 值或 OVS user cookie：

```
kubectl ko acl decode 0x640abcde000000c8
```

在一个节点上监听，并通过 OVN northbound leader 解码收到的每个事件：

```
kubectl ko acl listen --node worker-1
```

监听器会读取已启用的 CNI DaemonSet 中的 `localGroupID`，在指定节点选择一个运行中的 `kube-ovn-cni` Pod，并为每个解码事件输出一个 YAML 文档。如果某个采样无法再从 NBDB 解析，监听器会将错误写入 standard error，但不会停止。

OVN 本地采样 cookie 的编码如下：

```
63                          32 31                           0
+-----------------------------+-----------------------------+
| observation domain          | Sample.metadata             |
+-----------------------------+-----------------------------+

observation domain = application ID (8 bits) | datapath key (24 bits)
```

仅有 metadata 的输入可以识别 ACL 和策略，但无法识别事件来自 `acl-new` 还是 `acl-est`，也不包含逻辑 datapath key。

解码后的 YAML 对 allow 事件包含策略 UID、namespace、名称、方向和规则索引；还包含 OVN ACL UUID、action、方向、优先级、tier、match hash 以及采样 application 和 metadata。default-deny 输出使用 `policyOwner`，设置 `reason: network-policy-default-deny` 和 `attribution: non-exclusive`，并省略规则索引。

安全性

Linux psample 消息包含捕获的报文字节。当前 debug 命令只输出 cookie 和解码后的策略事件，但节点本地进程仍会从内核接收报文数据。因此，访问 `kubectl ko acl listen` 就具备报文观察能力，应限制为授权运维人员。使用尽可能低的采样概率，将命令输出作为敏感数据保护，并在排查完成后停止监听器。

故障隔离和监控

采样采用尽力而为方式。Kube-OVN 会先提交 NetworkPolicy 执行规则，再在单独的事务中挂载采样。不支持的 schema、ID 冲突、metadata 分配失败、节点能力不满足或挂载失败都不能阻止有效的 NetworkPolicy 生效。

2025 Kube-OVN Team

controller/daemon warning

`acl_sampling_controller_available`	reconcile `1` `0`
`acl_sampling_controller_failures_total{operation}`	`reconcile` `prepare` `attach`
`acl_sampling_node_available{node_name}`	`1` `0`
`acl_sampling_node_failures_total{node_name}`	collector-set reconcile

controller agent `aclSampling.enabled` `false` Kube-OVN NetworkPolicy ACL sample OVN `Sample` collector application collector set

PDF Slack Support

2026-10-28

2026-10-28

GitHub

2025 Kube-OVN Team

3.13.4 ▯▯▯ NetworkPolicy

概述

默认情况下，Kube-OVN 会将 NetworkPolicy 应用到 `podSelector` 选中的 Pod 上的所有 OVN 接口。

对于多网卡 Pod，可以通过以下注解将策略限定到指定 Provider：

- `ovn.kubernetes.io/network_policy_for`

这样可以让一条策略只作用于指定接口，而不是 Pod 的全部接口。

注解格式

注解值为逗号分隔：

```
metadata:
  annotations:
    ovn.kubernetes.io/network_policy_for: "ovn,default/net-a,default/net-b"
```

支持的条目格式：

- `ovn`（默认 OVN Provider）
- `<namespace>/<net-attach-def>`

示例：

- `ovn`
- `default/net-a`
- `ovn,default/net-a`

Provider 匹配行为

- 不设置注解：保持原有行为，策略作用于所有 OVN Provider。
- 非法条目会被忽略并记录日志。
- 如果所有条目都非法，则不会选中任何 Provider，策略不会选中端口。
- 重复条目会自动去重。

`<namespace>/<net-attach-def>` 在 Kube-OVN 内部会映射为 Provider 名称格式：

- `<nad-name>.<nad-namespace>.ovn`

示例

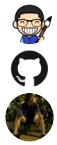
假设 Pod 有以下接口：

- 默认 OVN Provider（`ovn`）
- `default/net-a`
- `default/net-b`

那么：

- 不设置 `network_policy_for`：
- 策略作用于 `ovn`、`net-a`、`net-b`
- `network_policy_for: default/net-a`：
- 策略仅作用于 `net-a`
- `network_policy_for: ovn,default/net-b`：

2025 Kube-OVN Team



- 策略作用于 `ovn` 和 `net-b`

说明

- 该注解只用于限定策略作用的 Provider/接口范围，不改变 Kubernetes NetworkPolicy 本身语义。
- 建议显式配置注解值，避免策略范围过大或过小。

PDF Slack Support

2026年4月28日

2026年2月24日

GitHub

评论

3.13.5

Kubernetes NetworkPolicy L3 L4 AdminNetworkPolicy (ANP) Pod DNSNameResolver CoreDNS

NetworkPolicy OVN AddressSet IP IP OVN AddressSet DNS

1. kube-ovn-controller AdminNetworkPolicy DNSNameResolver CR
2. CoreDNS DNSNameResolver CR IP DNSNameResolver status
3. kube-ovn-controller DNSNameResolver CR status AddressSet

IP Deny Allow Allow Deny

ANP BANP CRD

AdminNetworkPolicy CRD

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/network-policy-api/refs/heads/main/config/crd/experimental/
policy.networking.k8s.io_adminnetworkpolicies.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/network-policy-api/refs/heads/main/config/crd/experimental/
policy.networking.k8s.io_baselineadminnetworkpolicies.yaml
```

DNSNAMERESOLVER

DNSNameResolver

```
kubectl apply -f https://raw.githubusercontent.com/kubeovn/dnsnameresolver/refs/heads/main/manifest/crd.yaml
kubectl apply -f https://raw.githubusercontent.com/kubeovn/dnsnameresolver/refs/heads/main/manifest/rbac.yaml
kubectl apply -f https://raw.githubusercontent.com/kubeovn/dnsnameresolver/refs/heads/main/manifest/cm.yaml
```

COREDNS

DNSNameResolver CoreDNS tag `dev` kubeovn/dnsnameresolver releases

```
kubectl set image deployment/coredns coredns=kubeovn/dnsnameresolver:<release-tag> -n kube-system
```

CoreDNS

```
kubectl get pod -n kube-system -l k8s-app=kube-dns
```

ANP

kube-ovn-controller

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kube-ovn-controller
spec:
  template:
    spec:
      containers:
      - name: kube-ovn-controller
        args:
        - --enable-anp=true
        - --enable-dns-name-resolver=true
        # ...
```

[illegible]

[illegible]

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: deny-external-domains
spec:
  priority: 55
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: kube-system
  egress:
  - action: Deny
    name: deny-baidu-google
    to:
    - domainNames:
      - '*.baidu.com.'
      - '*.google.com.'
```

[illegible]

[illegible]	[illegible]
`priority`	[illegible]
`subject`	[illegible]Pod [illegible]
`egress`	[illegible]
`action`	[illegible] `Allow` / `Deny` / `Pass`
`domainNames`	[illegible] `.` [illegible]

[illegible]

[illegible] kube-ovn-pinger [illegible]

```
#
kubectl exec -it -n kube-system kube-ovn-pinger-xxxxx -- ping baidu.com
```

> [illegible] DNS [illegible] ACL [illegible]

[illegible] DNSNameResolver [illegible]

```
# kubectl get dnsnameresolver
NAME                                DNS NAME        RESOLVED IPS
anp-deny-external-domains-88dc32ab  *.google.com.
anp-deny-external-domains-fb3029ce  *.baidu.com.    220.181.7.203
```

PDF Slack Support

2026-4-28

2025-9-17

GitHub 

[illegible]

3.13.6 ████

██████████ Pod ████████████████████Pod ████ Kube-OVN IPAM ███ MAC ███ IP ██

██████████

- ████ Pod ███ IP ██████
- ██ Pod ██ MAC ████ ARP ██
- ████████████████

████

██████ OVN ▮ Port Security ███████ Pod ████████Kube-OVN ██ OVN ████████████████████

- ████████████████ MAC ████ IP ████
- OVN ██████████████████ MAC ████ IP ██
- ██████ IPAM ████████████████
- ████████ OVN ████

████ OVN ████████████████████████████████████

████

████████████ Pod ███ `ovn.kubernetes.io/port_security` annotation ██████████

██████

```
apiVersion: v1
kind: Pod
metadata:
  name: secure-pod
  annotations:
    ovn.kubernetes.io/port_security: "true"
spec:
  containers:
  - name: nginx
    image: docker.io/library/nginx:alpine
```

PDF Slack Support

2025▮10▮24▮

2025▮10▮24▮

GitHub 

▮▮

4. KubeVirt

4.1 VM IP

在容器场景下，容器的 IP 地址通常是动态分配的，并且会在容器重启后发生变化。但是对于 VM 用户来说，他们希望 VM 的 IP 地址是固定的，来进行后续的管理和操作。

但是大部分常见 CNI 存在下面的局限性：

- 无法将 IP 地址与 VM 生命周期绑定，VM 重启或迁移后 IP 会发生变化。
- IP 地址和 Node 绑定，VM 迁移到新的节点后无法复用之前的 IP。
- 无法支持自定义 IP 地址配置，用户无法指定 VM 的 IP 地址。

因此通常会使用 KubeVirt 的 `masquerade` 网络模式，通过 iptables 将 VM 的流量转发到宿主机网卡上，从而实现 VM 的 IP 固定，但 `masquerade` 相比 `bridge` 存在以下问题：

- Pod IP 和 VM IP 不一致，管理方面存在复杂度，且重启和热迁移后 Pod IP 发生变化，外部访问来看地址依然不固定。
- `masquerade` 使用 iptables 进行流量转发，性能相比 `bridge` 模式要差。
- `masquerade` 只支持三层流量转发，一些二层网络功能无法实现。
- `masquerade` 流量通过 conntrack 记录转发状态，热迁移期间存在流量中断可能。

Kube-OVN 支持为 KubeVirt 下 `bridge` 和 `managedTap` 网络模式下的 IP 和 VM 生命周期绑定，该 IP 地址在 VM 重启，热迁移等操作后仍然保持不变。同时也支持通过添加 annotation 的方式为 VM 配置固定 IP 地址。对于附属网卡，Kube-OVN 可通过和 Multus 配合达到相同的固定地址的效果，具体操作请参考多网卡管理。

4.1.1 IP VM

对于只希望 VM 生命周期内 IP 地址固定，但不需要指定 IP 地址的场景，用户只需按原先方式创建 VM 即可。Kube-OVN 内部的 IPAM 会自动记录 VM 生命周期，保证 VM 在重启和迁移后使用相同的 IP 地址。

下面以 `bridge` 网络模式为例，创建一个 VM，进行重启，热迁移等操作，并观察 IP 地址的变化。

1. 创建 VM

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: testvm
spec:
  runStrategy: Always
  template:
    metadata:
      labels:
        kubevirt.io/size: small
        kubevirt.io/domain: testvm
      annotations:
        kubevirt.io/allow-pod-bridge-network-live-migration: "true"
    spec:
      domain:
        devices:
          disks:
            - name: containerdisk
              disk:
                bus: virtio
            - name: cloudinitdisk
              disk:
                bus: virtio
          interfaces:
          - name: default
            bridge: {}
        resources:
          requests:
            memory: 64M
      networks:
      - name: default
        pod: {}
      volumes:
        - name: containerdisk
          containerDisk:
            image: quay.io/kubevirt/cirros-container-disk-demo
        - name: cloudinitdisk
```


2025 Kube-OVN Team

```
        cloudInitNoCloud:
          userDataBase64: SGkuXG4=
```

1. □□ VM □□

```
kubectl get vmi testvm
```

1. □□ VM

```
virtctl restart testvm
```

1. □□□ VM

```
virtctl migrate testvm
```

□□□□□ bridge □□□ VM □□□□□□□IP □□□□□□□

4.1.2 □□ IP/Mac □□

□□□□□□ VM □ IP/Mac □□□□□□□□□□□□□□□ VM □□□ annotation □□□□ VM □□ IP □□□□□□□□□□□□□ KubeVirt □□□□ VM □□□□□□□□□□□□□□□□□□□□□□□□□□□□ IP/MAC□□□ `<nadName>.<nadNamespace>.kubernetes.io/ip_address.<interfaceName>` □ `<nadName>.<nadNamespace>.kubernetes.io/mac_address.<interfaceName>` □□□□□□□ - □□□□□□□□□□□□□ IP/MAC□

```yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: testvm
spec:
  runStrategy: Always
  template:
    metadata:
      labels:
        kubevirt.io/size: small
        kubevirt.io/domain: testvm
      annotations:
        ovn.kubernetes.io/ip_address: 10.16.0.15 #(1)
        ovn.kubernetes.io/mac_address: 00:00:00:53:6B:B6 #(2)
        kubevirt.io/allow-pod-bridge-network-live-migration: "true"
    spec:
      domain:
        devices:
          disks:
            - name: containerdisk
              disk:
                bus: virtio
            - name: cloudinitdisk
              disk:
                bus: virtio
          interfaces:
          - name: default
            bridge: {}
        resources:
          requests:
            memory: 64M
      networks:
      - name: default
        pod: {}
      volumes:
        - name: containerdisk
          containerDisk:
            image: quay.io/kubevirt/cirros-container-disk-demo
        - name: cloudinitdisk
          cloudInitNoCloud:
            userDataBase64: SGkuXG4=
```

1. 🙋 □□□□□ IP □□□□□□□□□□□□□
2. 🙋 □□□□□ Mac □□□□□□□□□□□□□

> ⚠ **Warning**
>
> KubeVirt VM API □□ `spec.template.spec.domain.devices.interfaces.macAddress` □□□□□□□□□□□ Mac □□□□□□□□□□□□□□□□□ Mac □□□ Kube-OVN □□□□□□□□□□□□□□□□□□□□□□□□□□□□ annotation □□□□□□□□□□□□ Mac □□□□ Mac □□□□□□□□□□□□□□□□□□□

2025 Kube-OVN Team

4.1.3 VM □□ IP

Kube-OVN □□□□□□□□□ VM IP□□□□ IP □□□□□ VM □□□□□□

□□□□□□□□□ VM IP□

1. □□ VM Annotation□□□□□□□□ IP □□□

2. □□ `virtctl restart <vm name>` □□ VM□□□□ IP □□□□□

4.1.4 □□□□

□□□□□ VM □□□□□□□ IP□□□□□□□□ VM □□□□□□□□□□□□ `ovn.kubernetes.io/logical_switch` □□□□

```yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: testvm
spec:
  runStrategy: Always
  template:
    metadata:
      labels:
        kubevirt.io/size: small
        kubevirt.io/domain: testvm
      annotations:
        ovn.kubernetes.io/logical_switch: subnet1 #(1)
        kubevirt.io/allow-pod-bridge-network-live-migration: "true"
    spec:
      domain:
        devices:
          disks:
            - name: containerdisk
              disk:
                bus: virtio
            - name: cloudinitdisk
              disk:
                bus: virtio
          interfaces:
          - name: default
            bridge: {}
        resources:
          requests:
            memory: 64M
      networks:
      - name: default
        pod: {}
      volumes:
        - name: containerdisk
          containerDisk:
            image: quay.io/kubevirt/cirros-container-disk-demo
        - name: cloudinitdisk
          cloudInitNoCloud:
            userDataBase64: SGkuXG4=
```

1. 🙋 □□□□□ VM □□□□□

PDF Slack Support

2026□2□11□

2025□3□3□

GitHub

4.1.5 □□

4.2 □□□□□

KubeVirt □ Clone □□□□□□□□□□□□□□□□□□□□□□□□ Label □ Annotation□□□□□□□□□□ Kube-OVN □ Annotation □□□ IP□MAC □□□

4.2.1 □□ Annotation

□ `VirtualMachineClone` □□□□□ Kube-OVN □□□ Annotation□□□□□□□

```yaml
kind: VirtualMachineClone
apiVersion: "clone.kubevirt.io/v1beta1"
metadata:
  name: testclone
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-source
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-target
  template:
    annotationFilters:
      - "ovn.kubernetes.io/*"
```

4.2.2 □□□□□□□□□□□

> **Note**
>
> `patches` □□□ KubeVirt 1.6 □□□□□□□

□□□□□□□□□□□□□□□□□□□ IP □□□

```yaml
kind: VirtualMachineClone
apiVersion: "clone.kubevirt.io/v1beta1"
metadata:
  name: testclone
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-source
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-target
  patches:
  - '{"op": "replace", "path": "/spec/template/metadata/annotations/ovn.kubernetes.io~1ip_address", "value": "10.16.0.15"}'
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```yaml
kind: VirtualMachineClone
apiVersion: "clone.kubevirt.io/v1beta1"
metadata:
  name: testclone
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-source
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-target
  patches:
  - '{"op": "remove", "path": "/spec/template/metadata/annotations/ovn.kubernetes.io~1ip_address"}'
```

□□□□□□□□□□□□□ Annotation □□□□□□□□□□□□□□ KubeVirt Clone API□

PDF Slack Support

2025 Kube-OVN Team

2025□9□10□

2025□8□13□



GitHub

4.2.3 □□

4.3 ▒▒▒▒

▒ KubeVirt ▒ Bridge ▒▒▒▒▒▒DHCP ▒▒▒ virt-launcher ▒▒▒▒▒▒▒▒ KubeVirt ▒▒▒▒ IPv4 ▒▒▒ DHCP▒▒▒ Bridge ▒▒▒ KubeVirt VM ▒▒▒▒ RA ▒▒▒▒▒▒ IPv6 ▒▒▒▒▒ Kube-OVN ▒▒▒ DHCP ▒ RA ▒▒▒▒▒▒ KubeVirt ▒▒▒▒▒ DHCP/RA ▒▒▒▒▒▒▒▒▒▒▒▒▒

▒ KubeVirt 1.4.0 ▒▒▒▒▒▒▒▒ Network Binding Plugin ▒▒▒▒▒ Bridge ▒▒▒▒▒ `managedTap` ▒▒▒▒▒ KubeVirt ▒▒▒ DHCP ▒▒▒▒▒▒▒▒▒▒ `managedTap` ▒▒▒ Kube-OVN ▒ DHCP/RA ▒▒▒▒▒▒▒▒ VM ▒▒▒▒▒▒▒▒▒▒▒▒▒

4.3.1 ▒▒▒▒ DHCP

▒ Kube-OVN ▒ Subnet ▒▒▒ DHCP ▒ IPv6 RA ▒▒▒▒▒▒▒▒ YAML ▒▒▒▒▒

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: dual-stack-subnet
spec:
  cidrBlock: "10.244.0.0/16,fd00:10:244::/64"
  enableDHCP: true
  enableIPv6RA: true
```

4.3.2 ▒▒ `managedTap` ▒▒▒▒

▒ KubeVirt ▒▒▒ `managedTap` ▒ Network Binding Plugin:

```
# kubectl patch kubevirts -n kubevirt kubevirt --type=json -p=\
'[{"op": "add", "path": "/spec/configuration/network",    "value": {
    "binding": {
        "managedtap": {
            "domainAttachmentType": "managedTap"
        }
    }
}}]'
```

4.3.3 ▒▒▒▒▒▒▒▒▒▒ `managedTap` ▒▒▒▒

```yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: dual-stack-vm
  namespace: default
spec:
  running: false
  template:
    spec:
      domain:
        devices:
          interfaces:
            - name: default
              binding:
                name: managedtap
      networks:
      - name: default
        pod: {}
```

▒▒▒▒▒▒▒▒▒▒▒▒ VM ▒▒ DHCP ▒ IPv6 RA ▒▒▒▒▒▒▒▒▒ IPv4/IPv6 ▒▒▒

PDF Slack Support

2025▒9▒10▒

2025▒3▒3▒

GitHub



4.3.4 ⬚⬚

4.4 ...

...

KubeVirt ...

- KubeVirt ... Bridge ...
- KubeVirt ...
- ... IP ...
- ...

Kube-OVN ... 0.5 ... TCP ...

4.4.1 ...

... VM Spec ... `kubevirt.io/allow-pod-bridge-network-live-migration: "true"` ... annotation...Kube-OVN ...

1. ... VM

```
kubectl apply -f - <<EOF
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: testvm
spec:
  runStrategy: Always
  template:
    metadata:
      labels:
        kubevirt.io/size: small
        kubevirt.io/domain: testvm
      annotations:
        kubevirt.io/allow-pod-bridge-network-live-migration: "true"
    spec:
      domain:
        devices:
          disks:
            - name: containerdisk
              disk:
                bus: virtio
            - name: cloudinitdisk
              disk:
                bus: virtio
          interfaces:
          - name: default
            bridge: {}
        resources:
          requests:
            memory: 64M
      networks:
      - name: default
        pod: {}
      volumes:
        - name: containerdisk
          containerDisk:
            image: quay.io/kubevirt/cirros-container-disk-demo
        - name: cloudinitdisk
          cloudInitNoCloud:
            userDataBase64: SGkuXG4=
EOF
```

1. SSH ...

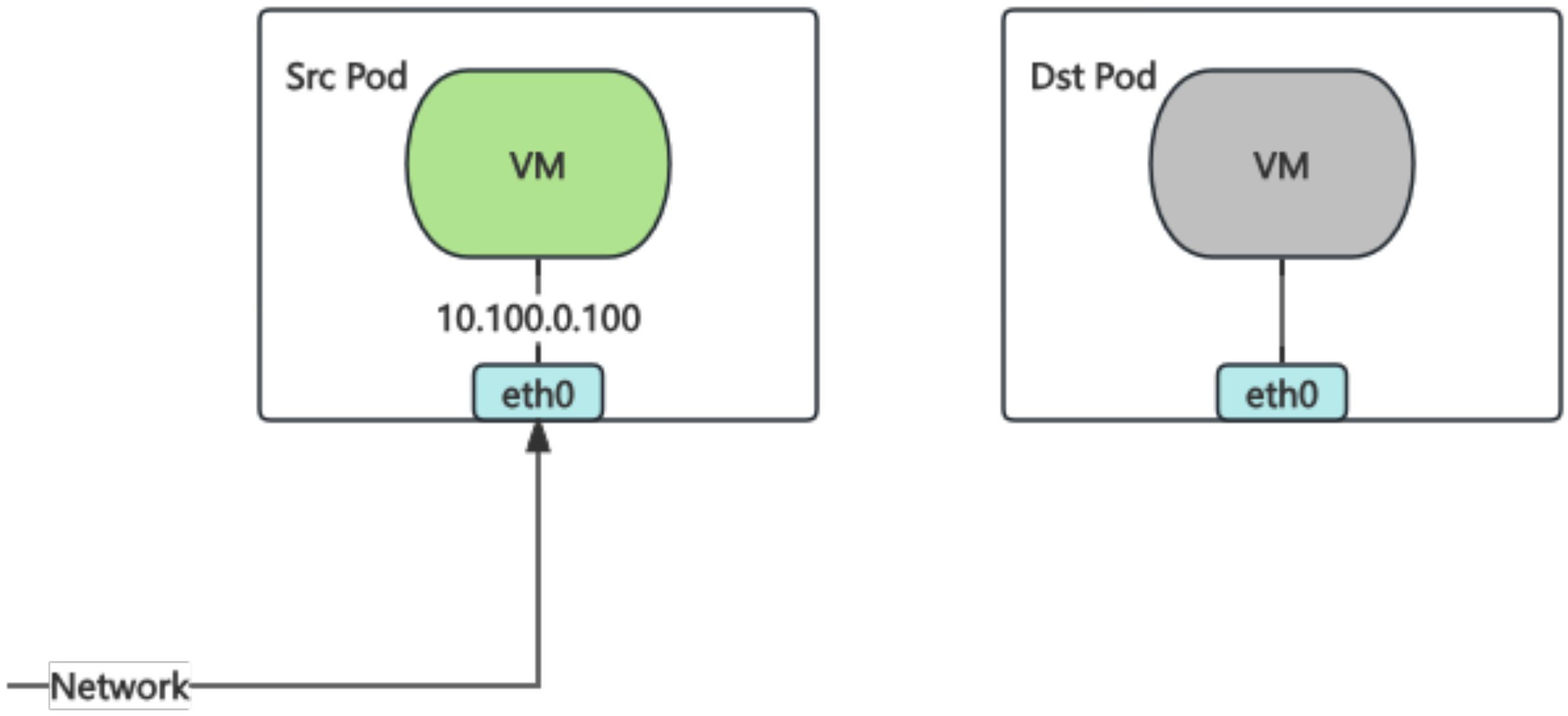
```
# password: gocubsgo
virtctl ssh cirros@testvm
ping 8.8.8.8
```

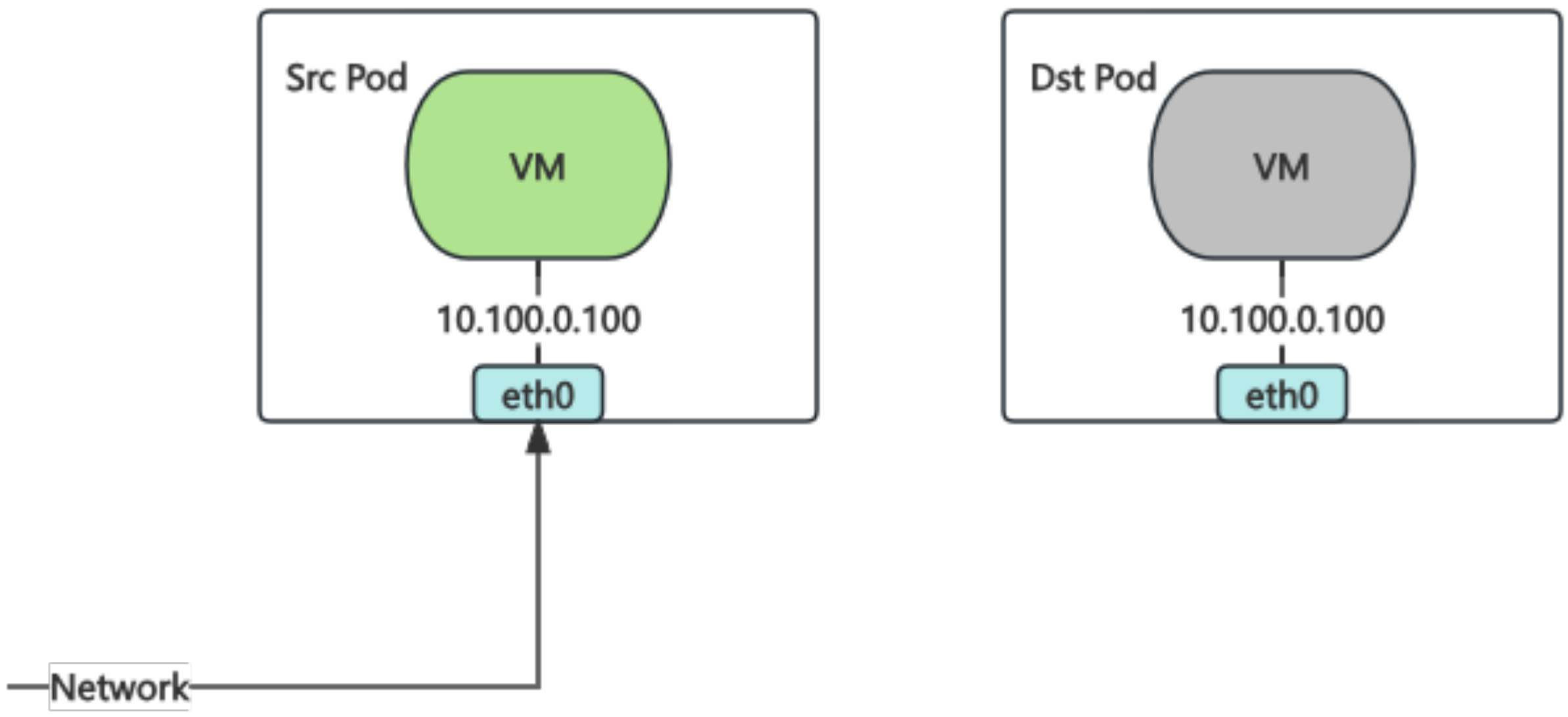
1. ...

```
virtctl migrate testvm
```

... VM ... SSH ... ping ...

2025 Kube-OVN Team





4.4.2 □□□□□

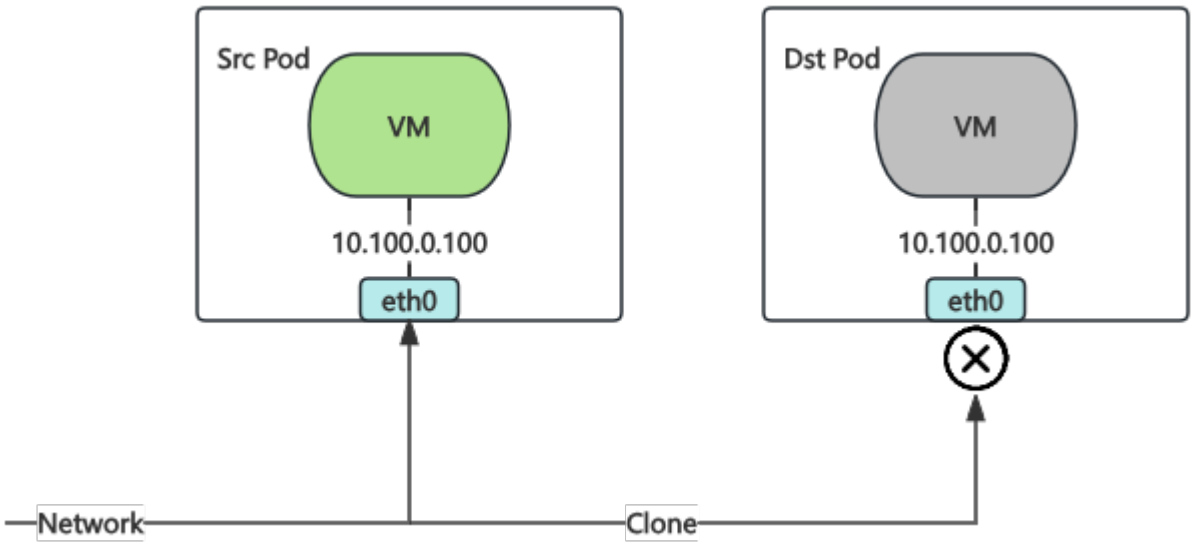
Kube-OVN 在热迁移过程中的实现借鉴了红帽团队的Live migration - Reducing downtime with multchassis port bindings。

在热迁移过程中为了保证源虚拟机和目标虚拟机的网络一致，在迁移过程中网络中会同时存在两个相同的 IP 地址，在这个过程中需要处理网络冲突和流量混乱。具体步骤如下：

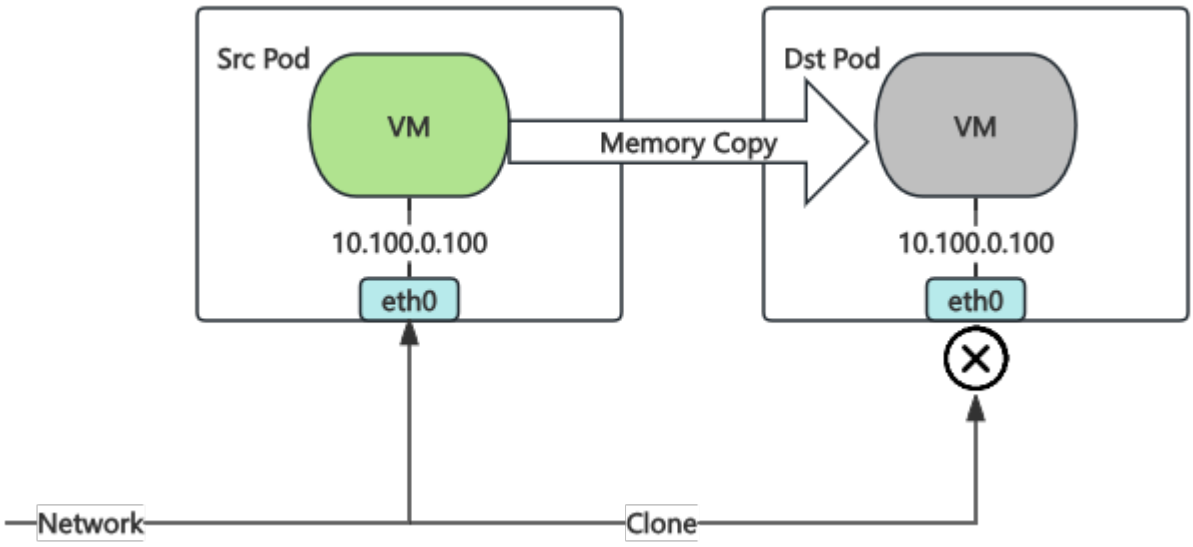
1. KubeVirt 发起迁移，并在目标机器创建对应的 Pod。

1. Kube-OVN 检测到该 Pod 为热迁移的目标 Pod，复用源 Pod 的网络端口信息。

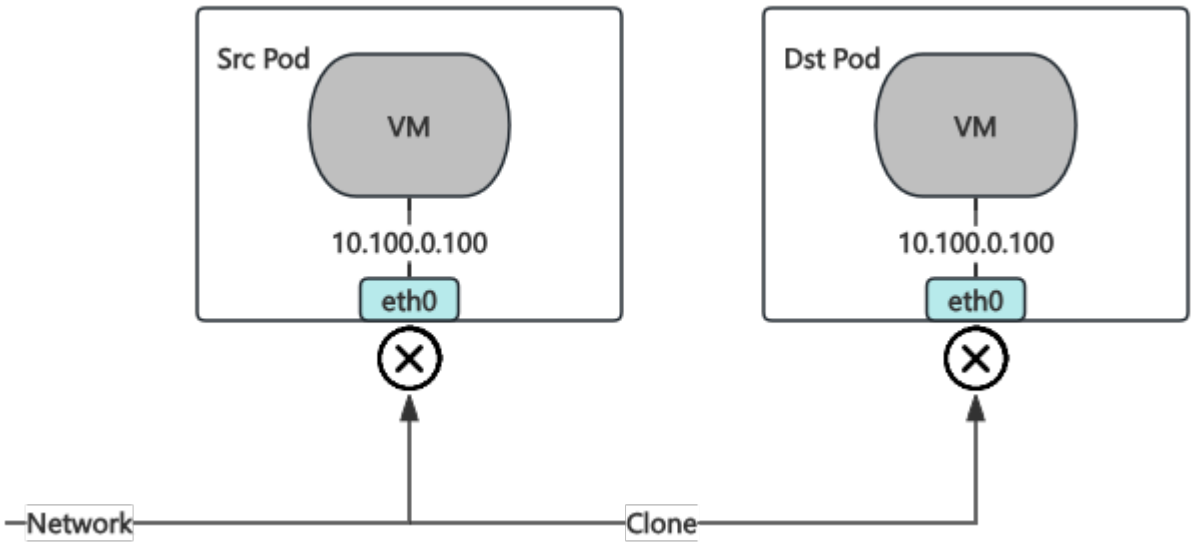
1. Kube-OVN 设置流量复制，此时网络流量会同时复制到源 Pod 和目标 Pod，以减少网络切换时控制平面切换带来的中断时间，同时将目标 Pod 的网络端口暂时停用，因此目标 Pod 不会真正接受到复制的流量，避免流量混乱。

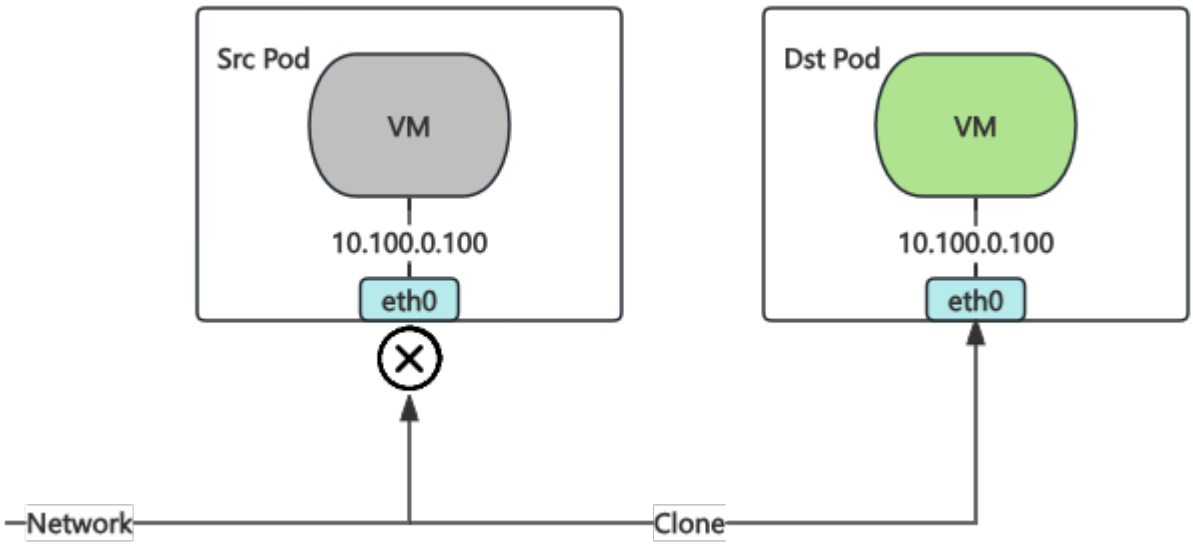


1. KubeVirt □□ VM □□□□□



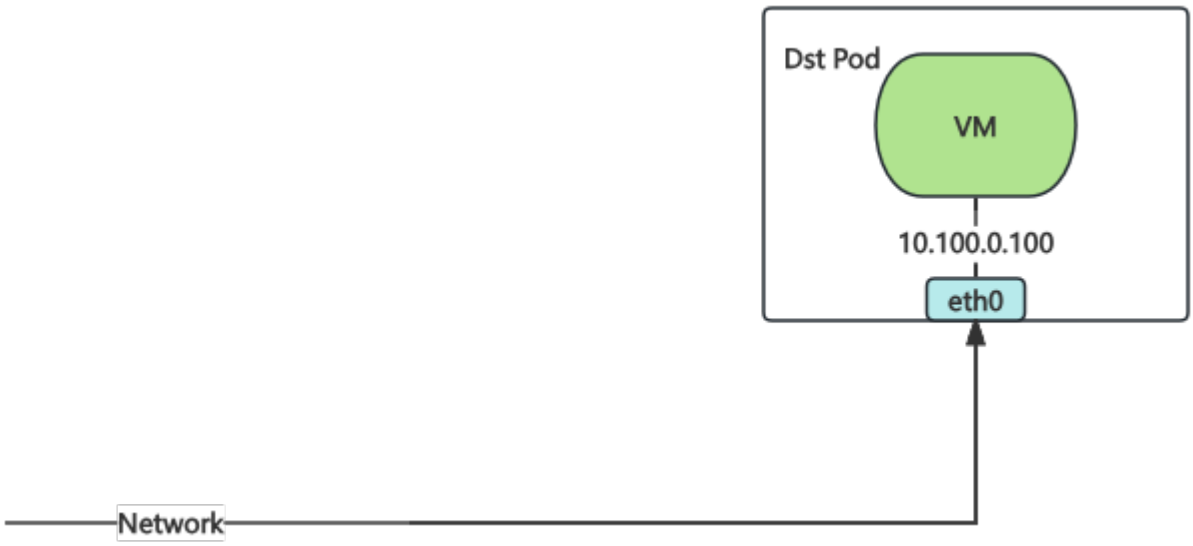
1. KubeVirt □□□□□□□□□□□ Pod□□□□ Pod □□□□□□□□□□□





1. KubeVirt 激活目标 Pod，此时 libvirt 会发送 RARP 从而激活目标 Pod 的网络端口，目标 Pod 开始流量处理。

1. KubeVirt 删除源 Pod，完成热迁移，Kube-OVN 通过 Watch Migration CR 来监听迁移完成事件，迁移完成后关闭流量复制。



⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ 5 ⬚⬚⬚ 6 ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ libvirt ⬚ RARP ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ 0.5 ⬚⬚⬚⬚⬚ TCP ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

PDF Slack Support

2025⬚9⬚10⬚

2025⬚3⬚3⬚

GitHub

4.4.3 ⬚⬚

2025 Kube-OVN Team

4.5 [illegible]

Kube-OVN [illegible] Multus Dynamic Networks Controller [illegible] KubeVirt [illegible] v1.4.0 [illegible] VM [illegible]

4.5.1 [illegible]

[illegible] Thick [illegible] Multus[illegible]

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/multus-cni/refs/heads/master/deployments/multus-daemonset-thick.yml
```

[illegible] Multus Dynamic Networks Controller[illegible]

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/multus-dynamic-networks-controller/refs/heads/main/manifests/dynamic-networks-
controller.yaml
```

4.5.2 [illegible]

[illegible] NetworkAttachmentDefinition

[illegible] `provider` [illegible] `ovn` [illegible]

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: attachnet
  namespace: default
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "kube-ovn",
      "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
      "provider": "attachnet.default.ovn"
    }'
```

- `spec.config.type` : [illegible] `kube-ovn` [illegible] CNI [illegible] Kube-OVN [illegible]
- `server_socket` : Kube-OVN [illegible] socket [illegible] `/run/openvswitch/kube-ovn-daemon.sock` [illegible]
- `provider` : [illegible] NetworkAttachmentDefinition [illegible] `<name>.<namespace>.ovn` , Kube-OVN [illegible] Subnet [illegible] ovn[illegible]

[illegible] Kube-OVN Subnet

[illegible] Kube-OVN [illegible] `provider` [illegible] NetworkAttachmentDefinition [illegible] `<name>.<namespace>.ovn` [illegible] `ovn` [illegible] Kube-OVN [illegible] Subnet [illegible]

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: attachnet
spec:
  protocol: IPv4
  provider: attachnet.default.ovn
  cidrBlock: 172.17.0.0/16
  gateway: 172.17.0.1
  excludeIps:
  - 172.17.0.0..172.17.0.10
```

4.5.3 [illegible] VM [illegible]

[illegible] yaml [illegible] VM[illegible]

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: vm-fedora
spec:
  runStrategy: Always
  template:
    spec:
      domain:
```

```
        devices:
          disks:
          - disk:
              bus: virtio
            name: containerdisk
          interfaces:
          - masquerade: {}
            name: defaultnetwork
          rng: {}
        resources:
          requests:
            memory: 1024M
      networks:
      - name: defaultnetwork
        pod: {}
      terminationGracePeriodSeconds: 0
      volumes:
      - containerDisk:
          image: quay.io/kubevirt/fedora-with-test-tooling-container-disk:devel
        name: containerdisk
```

▯▯▯▯

▯▯ VM ▯ Spec ▯▯▯▯▯▯▯▯▯

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: vm-fedora
template:
  spec:
    domain:
      devices:
        interfaces:
        - name: defaultnetwork
          masquerade: {}
          # new interface
        - name: dyniface1
          bridge: {}
    networks:
    - name: defaultnetwork
      pod: {}
      # new network
    - name: dyniface1
      multus:
        networkName: attachnet
```

▯▯▯▯

▯▯▯ `interface` ▯ `state` ▯▯▯ `absent` ▯▯▯▯▯▯▯▯▯

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: vm-fedora
template:
  spec:
    domain:
      devices:
        interfaces:
          - name: defaultnetwork
            masquerade: {}
          # set the interface state to absent
          - name: dyniface1
            state: absent
            bridge: {}
    networks:
      - name: defaultnetwork
        pod: {}
      - name: dyniface1
        multus:
          networkName: attachnet
```

PDF Slack Support

2025 Kube-OVN Team

2025⬚11⬚18⬚

2025⬚11⬚18⬚

GitHub 

4.5.4 ⬚⬚

4.6 DHCP

在使用 `managedTap`、`SR-IOV` 或 `DPDK` 类型网络时，KubeVirt 内置的 DHCP 无法在该网络模式下工作。Kube-OVN 可以利用 OVN 的 DHCP 能力在子网级别或 Pod 级别设置 DHCP 选项，从而帮助该网络类型的 KubeVirt 虚机正常使用 DHCP 获得分配的 IP 地址。同时 Kube-OVN 的 DHCP 还提供了 DHCPv6, IPv6RA, DNS, TFTP 等 DHCP 高级选项，用户可以根据自己的需求定义 DHCP 服务的具体行为。

> **Warning**
>
> 对于 `bridge` 类型网络，KubeVirt 的 DHCP 会先于 Kube-OVN 拦截并响应 DHCP 请求，因此 Kube-OVN 设置的 DHCP 功能无法生效。如果要使用 Kube-OVN 提供的高级 DHCP 能力，我们推荐将 `managedTap` 类型网络替换 `bridge` 类型网络。`managedTap` 类型网络的配置请参阅 managedTap 类型网络。

4.6.1 [illegible] DHCP [illegible]

子网 DHCP 的配置如下:

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: sn-dual
spec:
  cidrBlock: "10.0.0.0/24,240e::a00/120"
  default: false
  disableGatewayCheck: true
  disableInterConnection: false
  excludeIps:
    - 10.0.0.1
    - 240e::a01
  gateway: 10.0.0.1,240e::a01
  gatewayNode: ''
  gatewayType: distributed
  natOutgoing: false
  private: false
  protocol: Dual
  provider: ovn
  vpc: vpc-test
  enableDHCP: true
  dhcpV4Options: "lease_time=3600,router=10.0.0.1,server_id=169.254.0.254,server_mac=00:00:00:2E:2F:B8"
  dhcpV6Options: "server_id=00:00:00:2E:2F:C5"
  enableIPv6RA: true
  ipv6RAConfigs: "address_mode=dhcpv6_stateful,max_interval=30,min_interval=5,send_periodic=true"
```

- `enableDHCP`：是否开启子网的 DHCP 功能。
- `dhcpV4Options`, `dhcpV6Options`：该字段直接暴露 ovn-nb 内 DHCP 相关选项，请参考 DHCP Options。默认值分别为 `"lease_time=3600, router=$ipv4_gateway, server_id=169.254.0.254, server_mac=$random_mac"` 和 `server_id=$random_mac`。
- `enableIPv6RA`：是否开启 DHCPv6 的路由广播功能。
- `ipv6RAConfigs`：该字段直接暴露 ovn-nb 内 Logical_Router_Port 相关选项，请参考 Logical Router Port。默认值为 `address_mode=dhcpv6_stateful, max_interval=30, min_interval=5, send_periodic=true`。

4.6.2 Pod [illegible] DHCP [illegible]

除了子网级别的 DHCP 配置外，Kube-OVN 还支持通过 Pod 注解为每个 Pod 的网络接口单独配置 DHCP 选项。Pod 级别的 DHCP 配置具有最高优先级，会覆盖子网级别的 DHCP 设置，并且不受子网 `enableDHCP` 设置的影响。

注解格式

```
#     provider    "ovn"
ovn.kubernetes.io/dhcp-v4-options: "lease_time=3600,router=10.0.0.1,dns_server=8.8.8.8"
ovn.kubernetes.io/dhcp-v6-options: "server_id=00:00:00:00:00:01"

#      provider    "net1.ns1.ovn"
net1.ns1.ovn.kubernetes.io/dhcp-v4-options: "lease_time=7200"
```

注解的 key 格式为 `<provider>.kubernetes.io/dhcp-v4-options` 和 `<provider>.kubernetes.io/dhcp-v6-options`，其中 `<provider>` 为网络提供者名称。对于默认网络，provider 为 `ovn`；对于通过 Multus 添加的附加网络，provider 格式为 `<net-attach-def-name>.<namespace>.ovn`。

2025 Kube-OVN Team

▯▯▯▯

▯▯▯▯▯ Pod ▯▯▯▯▯▯ DHCPv4 ▯▯▯

```
apiVersion: v1
kind: Pod
metadata:
  name: dhcp-pod
  annotations:
    ovn.kubernetes.io/dhcp-v4-options: "lease_time=3600,router=10.0.0.1,dns_server=8.8.8.8"
    ovn.kubernetes.io/dhcp-v6-options: "server_id=00:00:00:00:00:01"
spec:
  containers:
    - name: test
      image: docker.io/library/nginx:alpine
```

▯▯ Multus ▯▯▯▯▯▯▯▯▯ provider ▯▯▯▯▯▯▯▯▯▯▯▯ DHCP ▯▯▯

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-nic-dhcp-pod
  annotations:
    k8s.v1.cni.cncf.io/networks: '[{"name": "net1", "namespace": "ns1"}]'
    ovn.kubernetes.io/dhcp-v4-options: "lease_time=3600,router=10.0.0.1"
    net1.ns1.ovn.kubernetes.io/dhcp-v4-options: "lease_time=7200,router=10.0.1.1"
spec:
  containers:
    - name: test
      image: docker.io/library/nginx:alpine
```

▯▯▯▯▯

DHCP ▯▯▯▯▯▯▯▯▯▯▯▯▯

1. **Pod** ▯▯▯▯▯ `<provider>.kubernetes.io/dhcp-v4-options` ▯ `<provider>.kubernetes.io/dhcp-v6-options` ▯▯▯ Pod ▯▯ DHCP ▯▯▯
2. ▯▯▯▯▯▯▯ Subnet CRD ▯ `dhcpV4Options` ▯ `dhcpV6Options` ▯▯▯▯▯▯▯▯▯ DHCP ▯▯▯

▯ Pod ▯▯▯ DHCP ▯▯▯▯▯▯▯▯▯ Pod ▯▯▯▯ DHCP ▯▯▯▯▯▯▯▯▯▯▯▯▯

> **Note**
>
> ▯▯▯▯▯▯▯ Pod ▯▯ DHCP ▯▯▯▯▯▯▯▯ DHCP ▯▯▯▯▯▯▯ Pod ▯▯▯▯▯

PDF Slack Support

2026▯3▯30▯

2022▯5▯24▯

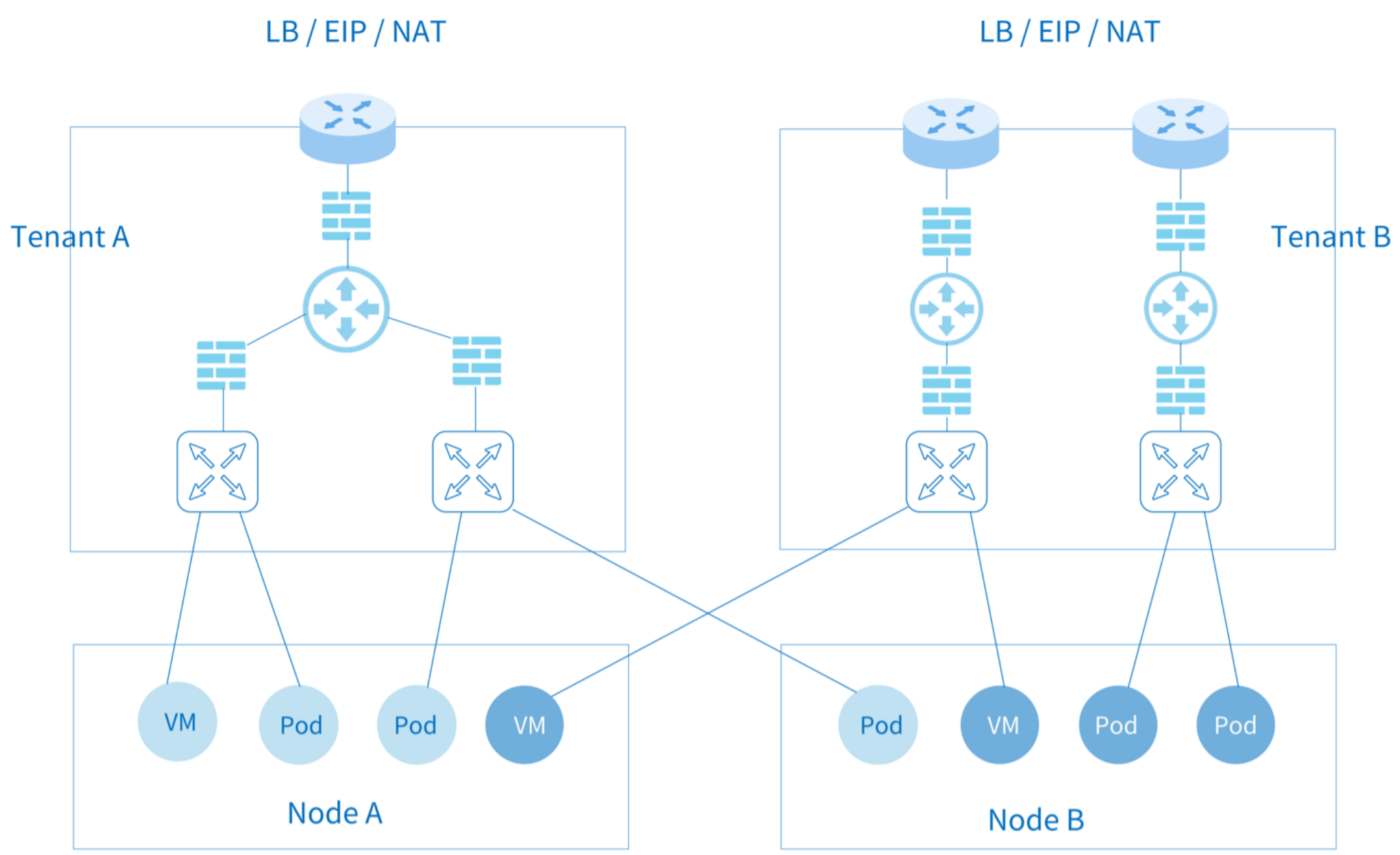
GitHub

5. VPC ⊠⊠

5.1 VPC ⊠⊠

Kube-OVN ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ Subnet ⊠⊠⊠ ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠EIP ⊠⊠⊠⊠

> VPC ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ Kubernetes ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ ⊠⊠⊠⊠⊠ Pod ⊠⊠⊠NodePort ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ DNS ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ ⊠⊠⊠⊠⊠⊠ Kubernetes ⊠⊠⊠⊠⊠⊠ Kube-OVN ⊠⊠ VPC ⊠⊠⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠ Subnet ⊠⊠⊠⊠ Kubernetes ⊠⊠⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠EIP ⊠ NAT ⊠⊠⊠⊠⊠⊠ ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠⊠⊠⊠⊠ ACL ⊠⊠⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠ ⊠ Underlay ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠VPC ⊠⊠⊠ Underlay ⊠⊠⊠⊠⊠⊠⊠



5.1.1 ⊠⊠⊠⊠

⊠ Kube-OVN ⊠⊠⊠⊠ VPC ⊠⊠⊠⊠ OVN ⊠⊠ IP ⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ OVN ⊠⊠⊠⊠⊠⊠ Datapath ID ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ Datapath ID ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ IP ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ VPC ⊠⊠⊠⊠⊠⊠⊠⊠ IP ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠OVN Architecture Design Decisions⊠

5.1.2 ⊠⊠⊠⊠⊠ VPC

⊠⊠⊠⊠ VPC⊠

```yaml
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: test-vpc-1
spec:
  namespaces:
  - ns1
---
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: test-vpc-2
spec:
  namespaces:
  - ns2
```

- `namespaces` ⊠⊠⊠⊠⊠⊠⊠⊠ Namespace ⊠⊠⊠⊠⊠⊠ VPC⊠⊠⊠⊠⊠⊠⊠⊠⊠

2025 Kube-OVN Team

创建两个子网，分属两个不同的 VPC 并有相同的 CIDR:

```
kind: Subnet
apiVersion: kubeovn.io/v1
metadata:
  name: net1
spec:
  vpc: test-vpc-1
  cidrBlock: 10.0.1.0/24
  protocol: IPv4
  namespaces:
    - ns1
---
kind: Subnet
apiVersion: kubeovn.io/v1
metadata:
  name: net2
spec:
  vpc: test-vpc-2
  cidrBlock: 10.0.1.0/24
  protocol: IPv4
  namespaces:
    - ns2
```

分别在两个 Namespace 下创建 Pod:

```
apiVersion: v1
kind: Pod
metadata:
  namespace: ns1
  name: vpc1-pod
spec:
  containers:
    - name: vpc1-pod
      image: docker.io/library/nginx:alpine
---
apiVersion: v1
kind: Pod
metadata:
  namespace: ns2
  name: vpc2-pod
spec:
  containers:
    - name: vpc2-pod
      image: docker.io/library/nginx:alpine
```

运行成功后可观察两个 Pod 地址属于同一个 CIDR，但由于运行在不同的租户 VPC，两个 Pod 无法相互访问。

5.1.3 ▯▯ VPC ▯▯

VPC 内容器访问外部网络需要通过 VPC 网关，VPC 网关可以打通物理网络和租户网络，并提供 浮动 IP，SNAT 和 DNAT 功能。

VPC 网关功能依赖 Multus-CNI 的多网卡功能，安装请参考 multus-cni。

> **Note**
>
> 自定义 VPC 下的子网不支持默认 VPC 下的分布式网关和集中式网关。
>
> 目前自定义 VPC 支持三种和外网互通的方案: VPC NAT 网关，OVN 网关和 Egress Gateway。其中 VPC NAT 网关为 Kube-OVN 最早支持的出网方式，为每个 VPC NAT 网关创建一个多网卡 Pod，一个网卡接入自定义 VPC 网络，另一个网卡通过 Macvlan 接入底层物理网络，通过 Pod 内的 iptables 实现各类出网和入网操作。该方式目前支持最多的功能，被使用的时间也最长，但也存在单点故障和使用复杂的缺点。
>
> OVN 网关使用了 OVN 内部原生支持的各类 NAT 能力实现出网和入网，可通过硬件加速提升性能，通过 OVN 内置的 BFD 实现故障切换，由于暴露的是 OVN 原生的概念，需要用户对 OVN 的使用较为熟悉。
>
> Egress Gateway 是针对 VPC NAT 网关的单点问题的改进，实现了水平扩展和故障快速切换，但是目前只实现了出网能力，没有入网能力。

配置外部网络

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: ovn-vpc-external-network
spec:
  protocol: IPv4
  provider: ovn-vpc-external-network.kube-system
  cidrBlock: 192.168.0.0/24
  gateway: 192.168.0.1  # IP address of the physical gateway
  excludeIps:
```

```
  - 192.168.0.1..192.168.0.10
---
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ovn-vpc-external-network
  namespace: kube-system
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "macvlan",
      "master": "eth1",
      "mode": "bridge",
      "ipam": {
        "type": "kube-ovn",
        "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
        "provider": "ovn-vpc-external-network.kube-system"
      }
    }'

         `ipam`        VPC     net1     Kube-OVN    IP
---
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ovn-vpc-external-network
  namespace: kube-system
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "macvlan",
      "master": "eth1",
      "mode": "bridge"
    }'
```

- 该 Subnet 用来管理可用的外部地址，网段内的地址将会通过 Macvlan 分配给 VPC 网关，请和网络管理员沟通给出可用的物理段 IP。
- VPC 网关使用 Macvlan 做物理网络配置，`NetworkAttachmentDefinition` 的 `master` 需为对应物理网络网卡的网卡名。
- `name` 外部网络名称。

在 Macvlan 模式下，附属网卡会将数据包直接通过该节点网卡对外发送，L2/L3 层面的转发能力需要依赖底层网络设备。需要预先在底层网络设备配置对应的网关、Vlan 和安全策略等配置。

1. 对于 OpenStack 的 VM 环境，需要将对应网络端口的 `PortSecurity` 关闭。
2. 对于 VMware 的 vSwitch 网络，需要将 `MAC Address Changes`，`Forged Transmits` 和 `Promiscuous Mode Operation` 设置为 `allow`。
3. 对于 Hyper-V 虚拟化，需要开启虚拟机网卡高级功能中的 `MAC Address Spoofing`。
4. 公有云，例如 AWS、GCE、阿里云等由于不支持用户自定义 Mac 无法支持 Macvlan 模式网络。
5. 由于 Macvlan 本身的限制，Macvlan 子接口无法访问父接口地址，也就意味着无法在 VpcNATGateway Pod 所在的宿主机上通过网络访问该 Pod。
6. 如果物理网卡对应交换机接口为 Trunk 模式，需要在该网卡上创建子接口再提供给 Macvlan 使用。

开启 **VPC** 网关功能

VPC 网关功能需要通过 `kube-system` 命名空间下的 `ovn-vpc-nat-gw-config` 配置开启。同时，`ovn-vpc-nat-config` ConfigMap 中可以配置 `image`（网关 Pod 所使用的镜像）、`imagePullSecret`（私有镜像仓库凭据）以及 `nodeSelector`（注意：此 `nodeSelector` 仅用于 `LoadBalancer` 类型 Service 的 lb-svc 部署，VpcNATGateway Pod 的调度由 `VpcNatGateway.spec.selector` 字段决定）。如果网关镜像位于私有仓库，需要先在 NAT 网关 workload 所在的命名空间创建 Docker registry Secret，再在该 ConfigMap 中填写 Secret 名称。默认 workload 命名空间为 `kube-system`；如果设置了 `spec.namespace`，则需要使用对应的命名空间：

```
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: ovn-vpc-nat-config
  namespace: kube-system
data:
  image: 'docker.io/kubeovn/vpc-nat-gateway:v1.17.0'
  imagePullSecret: vpc-nat-registry
  nodeSelector: |
    kubernetes.io/hostname: kube-ovn-control-plane
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: ovn-vpc-nat-gw-config
  namespace: kube-system
data:
  enable-vpc-nat-gw: 'true'
```

- `image`：网关 Pod 所使用的镜像。

2025 Kube-OVN Team

- `imagePullSecret`：可选的 Docker registry Secret 名称。Secret 必须已存在于 NAT 网关 workload 所在的命名空间，网关 StatefulSet/Deployment 会将其用于拉取私有镜像。
- `enable-vpc-nat-gw`：控制是否启用 VPC 网关功能。

例如，使用私有仓库时可以执行以下命令创建凭据（请替换为实际仓库信息）：

```
kubectl create secret docker-registry vpc-nat-registry \
  --namespace kube-system \
  --docker-server=registry.example.com \
  --docker-username='<username>' \
  --docker-password='<password>'
```

创建 **VPC** 网关

```
kind: VpcNatGateway
apiVersion: kubeovn.io/v1
metadata:
  name: gw1
spec:
  vpc: test-vpc-1
  subnet: net1
  lanIp: 10.0.1.254
  selector:
    - "kubernetes.io/hostname: kube-ovn-worker"
    - "kubernetes.io/os: linux"
  externalSubnets:
    - ovn-vpc-external-network
  noDefaultEIP: false
```

- `vpc`：该 VpcNatGateway 所属的 VPC。
- `subnet`：为 VPC 内某个 Subnet 名，VPC 网关 Pod 会在该子网下用 `lanIp` 来连接租户网络。
- `namespace`：可选，VPC 网关工作负载（Deployment 或 StatefulSet）及其 Pod 所在的 Namespace。未设置时使用 kube-ovn-controller 所在的 Namespace（通常为 `kube-system`）。`VpcNatGateway` 创建后该字段不可修改。`VpcNatGateway` 本身是集群级资源，因此该字段位于 `spec` 而不是 `metadata.namespace`。
- `lanIp`：`subnet` 内某个未被使用的 IP，VPC 网关 Pod 最终会使用该 IP。当 VPC 配置路由需要指向当前 VpcNatGateway 时 `nextHopIP` 需要设置为这个 `lanIp`。
- `selector`：VpcNatGateway Pod 的节点选择器，格式和 Kubernetes 中的 NodeSelector 格式相同。
- `externalSubnets`：VPC 网关使用的外部网络，如果不配置则默认使用 `ovn-vpc-external-network`，当前版本只支持配置一个外部网络。
- `noDefaultEIP`：VPC 网关的附属网卡是否绑定默认 EIP，默认为 `false` 兼容 v1.15 前行为模式，在非 BGP 网关模式下推荐设置为 `true` 来节省 Underlay 地址。

其他可配参数：

- `tolerations`：为 VPC 网关配置容忍度，具体配置参考 污点和容忍度。
- `affinity`：为 VPC 网关 Pod 或节点配置亲和性，具体设置参考 亲和性与反亲和性。

需要将 VPC 网关 Pod 放入自定义 Namespace 时，在 `spec.namespace` 中指定已存在的 Namespace：

```
kind: VpcNatGateway
apiVersion: kubeovn.io/v1
metadata:
  name: gw1
spec:
  namespace: tenant-network
  vpc: test-vpc-1
  subnet: net1
  externalSubnets:
    - ovn-vpc-external-network
```

在使用 VPC-NAT-GW 时需要注意：

1. nat gw pod 创建之后，net1 arp 是关闭的，ping 不通物理网关，创建 eip 后会自动打开 arp，可以 ping 通。

▯▯ LAN IP ▯▯▯▯

`spec.lanIp` 可以留空。单副本模式（`replicas` 为 1 或未设置）下，controller 会从子网动态分配 LAN IP，将实际地址写回 `spec.lanIp`；写回后该字段不可变。单副本模式使用 StatefulSet。

当 `spec.replicas` 大于 1 时，网关使用 Deployment 运行高可用副本，每个 Pod 动态分配自己的 LAN IP，此时 `spec.lanIp` 不用于固定地址。controller 会在副本替换后重新指向 OVN 策略路由和 BFD 会话。`status.lanIp` 记录实际观察到的地址：单副本为单个地址，高可用模式为按排序排列、逗号分隔的地址列表；`status.replicas` 反映配置的 `spec.replicas` 值，`status.ready` 单独表示网关 workload 是否就绪。未设置 `spec.replicas` 时 workload 实际使用 1 个副本，但 `status.replicas` 可能仍为 0。

LAN IP

```
spec:
  replicas: 2
  # lanIp      controller
```

BFD

VPC NAT BFDBidirectional Forwarding DetectionVPC `spec.bfd.enabled` true

```
spec:
  vpc: test-vpc-1
  subnet: net1
  replicas: 2
  bfd:
    enabled: true
    minRX: 500
    minTX: 500
    multiplier: 3
```

`bfd.enabled`	boolean	false	BFD
`bfd.minRX`	int32	1000	BFDms
`bfd.minTX`	int32	1000	BFDms
`bfd.multiplier`	int32	3	BFD = minRX × multiplier

`internalSubnets` `internalCIDRs` CIDR NAT

```
spec:
  vpc: test-vpc-1
  subnet: net1
  internalSubnets:
  - subnet-a
  - subnet-b
  internalCIDRs:
  - 10.0.2.0/24
```

EIP

EIP IP VPC DNATSNAT IP

EIP

```
kind: IptablesEIP
apiVersion: kubeovn.io/v1
metadata:
  name: eip-random
spec:
  natGwDp: gw1
```

EIP

```
kind: IptablesEIP
apiVersion: kubeovn.io/v1
metadata:
  name: eip-static
spec:
  natGwDp: gw1
  v4ip: 192.168.0.100
```

EIP

```
kind: IptablesEIP
apiVersion: kubeovn.io/v1
metadata:
  name: eip-random
spec:
```

```
  natGwDp: gw1
  externalSubnet: ovn-vpc-external-network
```

- `externalSubnet`：EIP 所在外部网络名称，如果不指定则默认为 `ovn-vpc-external-network`。IptablesEIP 的外部网络和 VpcNatGateway 的 `externalSubnets` 是完全独立的，EIP 可以使用任意有效的外部子网，无需和网关的外部网络保持一致。
- `namespace`：可选，EIP 所引用的 NAT 网关工作负载及 Pod 所在 Namespace。未设置时从 `natGwDp` 引用的 VpcNatGateway 推导（网关的 `spec.namespace` 为空时使用 controller 所在的 Namespace）；显式设置时必须与该网关的有效 Namespace 一致。`IptablesEIP` 是集群级资源，该字段位于 `spec` 而不是 `metadata.namespace`。

需要显式指定 EIP 的 NAT 网关工作负载 Namespace 时，在 `spec.namespace` 中填写与所引用 VpcNatGateway 相同的 Namespace：

```
kind: IptablesEIP
apiVersion: kubeovn.io/v1
metadata:
  name: eip-tenant
spec:
  natGwDp: gw1
  namespace: tenant-network
```

创建 DNAT 规则

通过 DNAT 规则，外部可以通过一个 EIP 加端口的方式来访问 VPC 内的一个 IP 和端口。

```
kind: IptablesEIP
apiVersion: kubeovn.io/v1
metadata:
  name: eipd01
spec:
  natGwDp: gw1

---
kind: IptablesDnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: dnat01
spec:
  eip: eipd01
  externalPort: '8888'
  internalIp: 10.0.1.10
  internalPort: '80'
  protocol: tcp
```

DNAT 共享负载均衡（SHARE 类型）

`type: share` 用于 **VPC NAT 网关接管 Kubernetes Service** 场景：网关将流量负载均衡分发到多个后端 Pod。

share 类型的 **IptablesDnatRule** 由控制器自动管理，不支持手动创建。使用方式见下方「VPC NAT 网关接管 Kubernetes Service」章节。

字段说明：

- `type`：`exclusive`（默认）为独占模式，一个 EIP:Port 只能被一个 DNAT 规则使用；`share` 由控制器在 Service 接管场景下自动生成，允许多个后端共享同一个 EIP:Port。
- `sessionAffinity`：仅 `share` 类型支持。设置为 `ClientIP` 后，同一客户端 IP 的连接会被固定到同一个后端，默认超时 10800 秒（3 小时），可通过 `sessionAffinityTimeoutSeconds` 指定。

创建 SNAT 规则

通过 SNAT 规则，VPC 内的 Pod 访问外部的地址时将会通过对应 EIP 进行 SNAT。

```
---
kind: IptablesEIP
apiVersion: kubeovn.io/v1
metadata:
  name: eips01
spec:
  natGwDp: gw1
---
kind: IptablesSnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: snat01
spec:
  eip: eips01
  internalCIDR: 10.0.1.0/24
```

□□□□ **IP**

□□□□ IP □□□VPC □□□□ IP □□ EIP □□□□□□□□□□□□□□□□ EIP □□ VPC □□ IP□VPC □□□□ IP □□□□□□□□□□ SNAT □□□ EIP□

```
---
kind: IptablesEIP
apiVersion: kubeovn.io/v1
metadata:
  name: eipf01
spec:
  natGwDp: gw1

---
kind: IptablesFIPRule
apiVersion: kubeovn.io/v1
metadata:
  name: fip01
spec:
  eip: eipf01
  internalIp: 10.0.1.5
```

VPC NAT □□□□ Kubernetes Service

□□□□□□□□□□□□□□□□□□

□□	□□□□	□□
`--enable-lb`	true	OVN □□□□□□□□ ClusterIP / NodePort / LoadBalancer □□ Service □□□□□
`--enable-gw-nftable-lb-svc`	false	VPC NAT □□□□ LoadBalancer □□ Service□□□□□ nftables □□□□□□□□□□ `ovn.kubernetes.io/eip` annotation
`--enable-gw-nftable-svc-cluster-ip`	false	□ Service ClusterIP □□□□□ nftables□□□□□□ ClusterIP □□□□
`--enable-lb-svc`	false	Pod □□ LoadBalancer□□□ LoadBalancer Service □□□□ Pod□DNAT EIP □ ClusterIP□□ `--enable-gw-nftable-lb-svc` □□□□□□ `--enable-lb` □ `--enable-gw-nftable-svc-cluster-ip` □□ ClusterIP □□□

□□□□□□□□

□□	□□□□
□ `--enable-lb` □□□□	OVN □□□□□□□□□ Service □□
`--enable-lb` + `--enable-gw-nftable-lb-svc`	OVN □□ ClusterIP□□□□□ LoadBalancer □ EIP □□□□
`--enable-gw-nftable-lb-svc` + `--enable-gw-nftable-svc-cluster-ip`	□□□□□□ EIP □ ClusterIP□OVN LB □□□
□ `--enable-gw-nftable-svc-cluster-ip`	□□□□□ ClusterIP□□ EIP□□□□ `--enable-lb` □□
`--enable-lb-svc` + `--enable-lb`	Pod □□□□ EIP□OVN □□□□□□□ ClusterIP
`--enable-lb-svc` + `--enable-gw-nftable-svc-cluster-ip`	Pod □□□□ EIP□□□□□ ClusterIP

□□□□□

- `--enable-lb-svc` □ `--enable-gw-nftable-lb-svc` □□□□□□ LoadBalancer Service □ EIP □□□□
- □□ Service □□□□□□ `ovn.kubernetes.io/vpc-nat-gw` □ `ovn.kubernetes.io/service_external_ip_from_subnet` annotation□webhook □□□□□□□□□□□□□□ LoadBalancer ingress IP□

□□□□□ □ Service □□ annotation `ovn.kubernetes.io/vpc-nat-gw` □□□□□□□□LoadBalancer □□□ Service □□□□ `ovn.kubernetes.io/eip` □□ EIP □□□□□□□□

```
apiVersion: v1
kind: Service
metadata:
  name: my-app
  annotations:
    ovn.kubernetes.io/vpc-nat-gw: gw1
    ovn.kubernetes.io/eip: eipd01
```

```
spec:
  type: LoadBalancer
  selector:
    app: my-app
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
```

控制器会自动为每个 Service 端口和后端组合创建对应的 `IptablesDnatRule`（`type: share`），并在网关 Pod 上编程 nftables 规则。Service 删除时自动清理。

特性说明：

- 同时支持 `LoadBalancer` 和 `ClusterIP` 类型的 Service
- 集群内流量（ClusterIP）和外部流量（EIP）均由网关负载均衡
- 只支持 `ExternalTrafficPolicy: Cluster`，不支持 Local 模式
- share DNAT 仅支持 IPv4
- 后端变化通过 EndpointSlice 自动感知

5.1.4 [illegible]路由

在自定义 VPC 内，用户可以自定义网络内部的路由规则，结合网关实现更灵活的转发。 Kube-OVN 支持静态路由和更为灵活的策略路由。

静态路由

```
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: test-vpc-1
spec:
  staticRoutes:
    - cidr: 0.0.0.0/0
      nextHopIP: 10.0.1.254
      policy: policyDst
    - cidr: 172.31.0.0/24
      nextHopIP: 10.0.1.253
      policy: policySrc
      routeTable: "rtb1"
```

- `policy`：支持目的地址路由 `policyDst` 和源地址路由 `policySrc`。
- 当路由规则存在重叠时，CIDR 掩码较长的规则优先级更高，若掩码长度相同则目的地址路由优先于源地址路由。
- `routeTable`：可指定静态路由所在的路由表，默认在主路由表。子网关联路由表请参考创建子网

策略路由

针对静态路由匹配的流量，可通过策略路由进行更细粒度的控制。策略路由提供了更精确的匹配规则，优先级控制 和更多的转发动作。该功能为 OVN 内部逻辑路由器策略功能的一个对外暴露，更多使用信息请参考 Logical Router Policy。

简单示例如下：

```
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: test-vpc-1
spec:
  policyRoutes:
    - action: drop
      match: ip4.src==10.0.1.0/24 && ip4.dst==10.0.1.250
      priority: 11
    - action: reroute
      match: ip4.src==10.0.1.0/24
      nextHopIP: 10.0.1.252
      priority: 10
```

5.1.5 自定义内部负载均衡规则

Kubernetes 本身提供的 Service 能力可以完成内部负载均衡的功能，但是受限于 Kubernetes 实现， Service 的 IP 地址是全局分配且不能重复。对于 VPC 的使用场景，用户希望能自定义内部负载均衡的地址 范围，不同 VPC 下的负载均衡地址可能重叠，Kubernetes 内置的 Service 功能无法完全满足。

针对这类场景，Kube-OVN 提供了 `SwitchLBRule` 资源，用户可以自定义内部负载均衡的地址范围。

2025 Kube-OVN Team

... `SwitchLBRule` ...

```
apiVersion: kubeovn.io/v1
kind: SwitchLBRule
metadata:
  name:  cjh-slr-nginx
spec:
  vip: 1.1.1.1
  sessionAffinity: ClientIP
  namespace: default
  selector:
    - app: nginx
  ports:
  - name: dns
    port: 8888
    targetPort: 80
    protocol: TCP
```

- `vip` ...
- `namespace` ... Pod ... Namespace...
- `sessionAffinity` ... Service ... `sessionAffinity` ...
- `selector` ... Service ... `selector` ...
- `ports` ... Service ... `port` ...

...

```
# kubectl get slr
NAME               VIP          PORT(S)                  SERVICE                              AGE
vpc-dns-test-cjh2  10.96.0.3    53/UDP,53/TCP,9153/TCP   kube-system/slr-vpc-dns-test-cjh2    88m
```

5.1.6 ... vpc-dns

... VPC ... VPC ...VPC ... Pod ... coredns ... ... VPC ... CoreDNS ... Service ... Kube-OVN ... vpc-dns ...

...

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ovn-nad
  namespace: default
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "kube-ovn",
      "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
      "provider": "ovn-nad.default.ovn"
    }'
```

... **vpc-dns ... ConfigMap**

... kube-system ... configmap... vpc-dns ... vpc-dns ...

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: vpc-dns-config
  namespace: kube-system
data:
  coredns-vip: 10.96.0.3
  enable-vpc-dns: "true"
  nad-name: ovn-nad
  nad-provider: ovn-nad.default.ovn
```

- `enable-vpc-dns` ... `true` ... `false` ... `true` ...
- `coredns-image` ...dns ... coredns ...
- `coredns-vip` ... coredns ... lb ... vip...
- `nad-name` ... `network-attachment-definitions` ...
- `nad-provider` ... provider ...

2025 Kube-OVN Team

- `k8s-service-host` □□□□□□□□ coredns □□ k8s apiserver □□□ ip□
- `k8s-service-port` □□□□□□□□ coredns □□ k8s apiserver □□□ port□

□□ **vpc-dns** □□□□

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:vpc-dns
rules:
  - apiGroups:
    - ""
    resources:
    - endpoints
    - services
    - pods
    - namespaces
    verbs:
    - list
    - watch
  - apiGroups:
    - discovery.k8s.io
    resources:
    - endpointslices
    verbs:
    - list
    - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: vpc-dns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:vpc-dns
subjects:
- kind: ServiceAccount
  name: vpc-dns
  namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vpc-dns
  namespace: kube-system
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: vpc-dns-corefile
  namespace: kube-system
data:
  Corefile: |
    .:53 {
        errors
        health {
          lameduck 5s
        }
        ready
        kubernetes cluster.local in-addr.arpa ip6.arpa {
          pods insecure
          fallthrough in-addr.arpa ip6.arpa
        }
        prometheus :9153
        forward . /etc/resolv.conf {
          prefer_udp
        }
        cache 30
        loop
        reload
        loadbalance
    }
```

□□ **vpc-dns**

```yaml
kind: VpcDns
apiVersion: kubeovn.io/v1
metadata:
  name: test-cjh1
spec:
```

```
  vpc: cjh-vpc-1
  subnet: cjh-subnet-1
```

- `vpc`：用于部署 dns 组件的 vpc 名称。
- `subnet`：用于部署 dns 组件的子网名称。

查看资源信息：

```
[root@hci-dev-mst-1 kubeovn]# kubectl get vpc-dns
NAME        ACTIVE   VPC         SUBNET
test-cjh1   false    cjh-vpc-1   cjh-subnet-1
test-cjh2   true     cjh-vpc-1   cjh-subnet-2
```

- `ACTIVE`：`true` 成功部署了自定义 dns 组件，`false` 无部署

限制

- 一个 VPC 下只会部署一个自定义 DNS 组件；
- 当一个 VPC 下配置多个 vpc-dns 资源（即同一个 VPC 不同的 subnet），只有一个 vpc-dns 资源状态为 `true`，其他为 `false`；
- 当 `true` 的 vpc-dns 被删除掉，会获取其他 `false` 的 vpc-dns 进行部署。

5.1.7 ⊠⊠⊠⊠

如果自定义 VPC 下存在多个子网，用户可以指定其中一个作为 VPC 下的默认子网。

```
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: test-vpc-1
spec:
  namespaces:
  - ns1
  defaultSubnet: test
```

- `defaultSubnet`：VPC 下默认子网的名称。

默认子网会给所有未指定子网的 Namespace 增加 `ovn.kubernetes.io/logical_switch` 注解，所有没有显式使用 `ovn.kubernetes.io/logical_switch` 注解的 Pod 都会自动从默认子网分配地址。

自定义 VPC Pod 支持 livenessProbe 和 readinessProbe

由于常规配置下自定义 VPC 下的 Pod 和节点的网络之间并不互通，所以 kubelet 发送的探测报文无法到达自定 VPC 内的 Pod。Kube-OVN 通过 TProxy 将 kubelet 发送的探测报文重定向到自定义 VPC 内的 Pod，从而实现这一功能。

配置方法如下，在 DaemonSet `kube-ovn-cni` 中增加参数 `--enable-tproxy=true`：

```
spec:
  template:
    spec:
      containers:
      - args:
        - --enable-tproxy=true
```

该功能限制条件：

1. 当同一个节点下出现不同 VPC 下的 Pod 具有相同的 IP，探测功能失效。
2. 目前暂时只支持 `tcpSocket` 和 `httpGet` 两种探测方式。

PDF Slack Support

2025 Kube-OVN Team

2026□11□7□

2022□5□24□

GitHub 

5.1.8 □□

5.2 VPC Egress Gateway

> **Note**
>
> ### VPC VPC
>
> VPC VPCNAT OVN Egress Gateway VPC NAT Kube-OVN VPC NAT Pod VPC Macvlan Pod iptables
>
> OVN OVN NAT OVN BFD OVN OVN OVN
>
> Egress Gateway VPC NAT

VPC Egress Gateway VPCVPC Pod

- ECMP Active-Active
- BFD <1s
- IPv6
- Namespace Pod
- Node Egress Gateway

VPC Egress Gateway BGP EVPN/VXLAN Egress Gateway BGP/EVPN

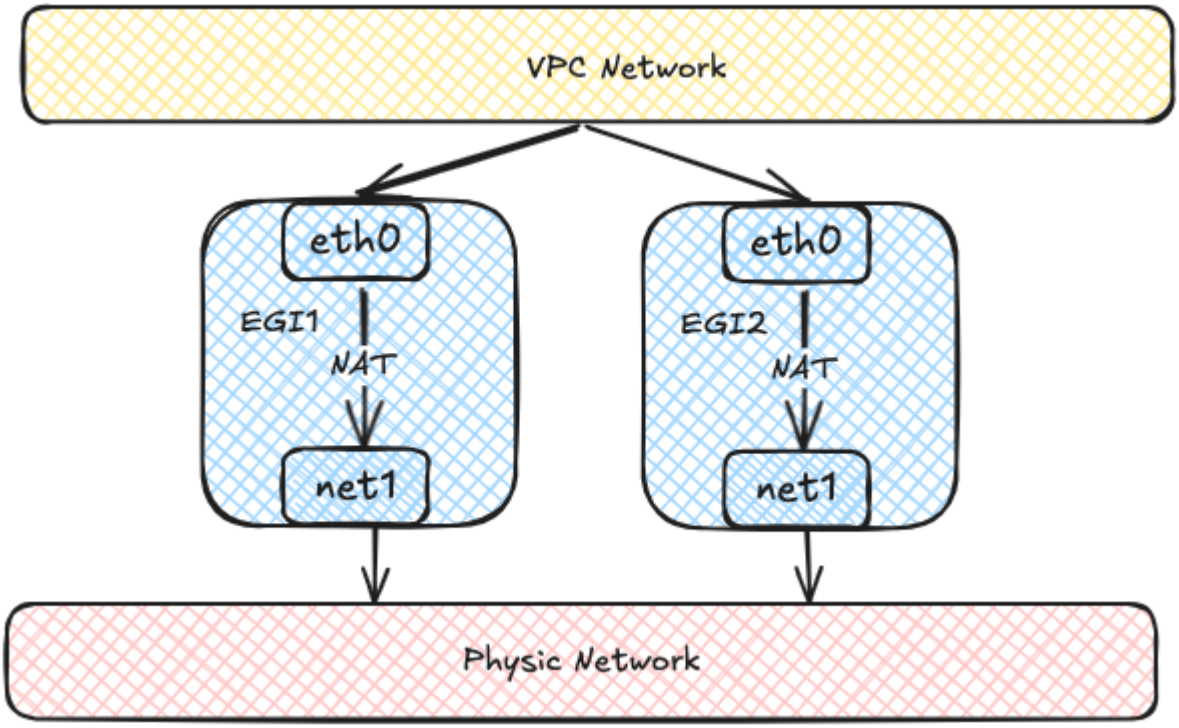
conntrack NAT JSON VPC Egress Gateway

VPC Egress Gateway

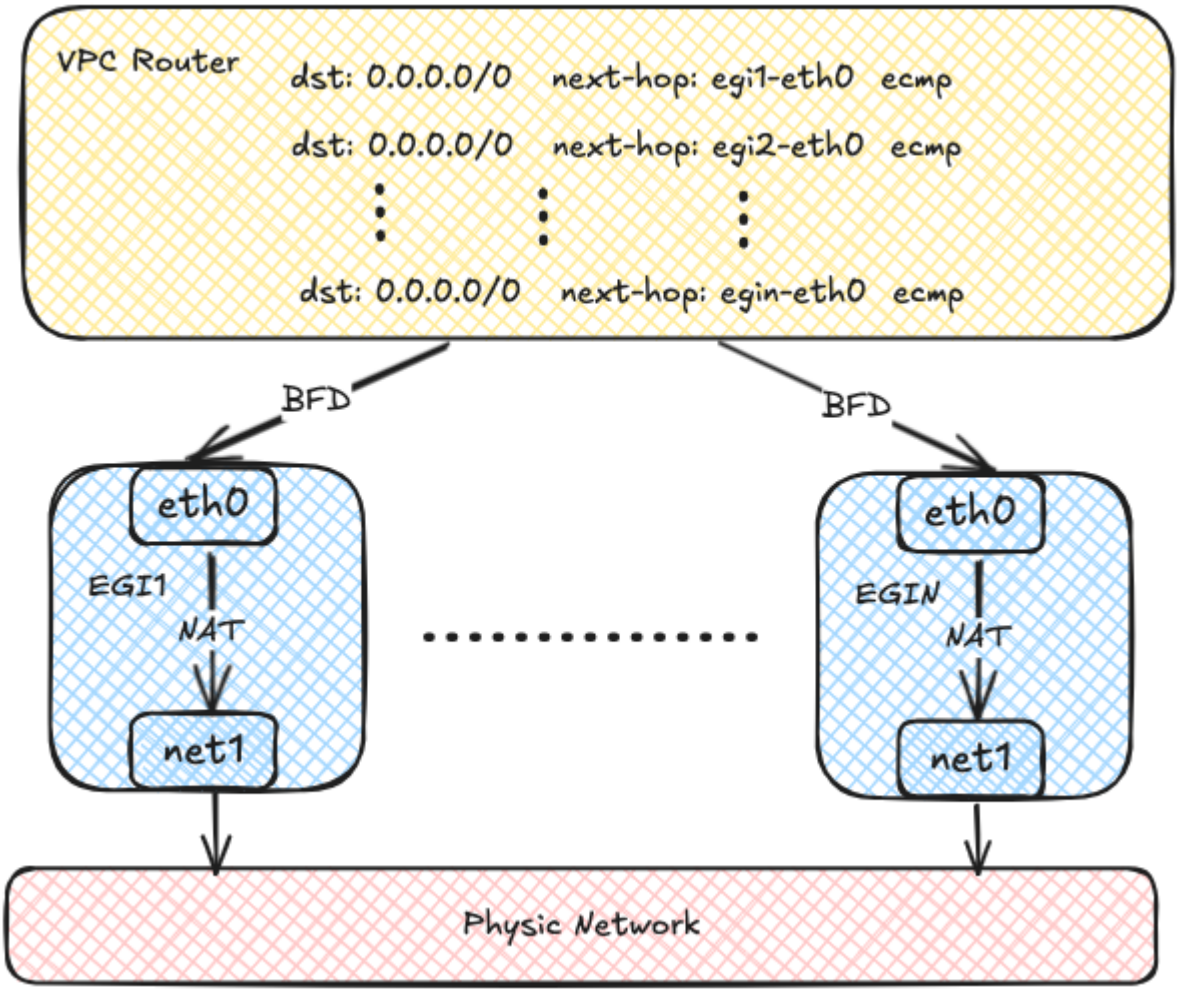
- Macvlan Underlay
- Gateway Egress IP
- SNAT EIP DNAT
-

5.2.1

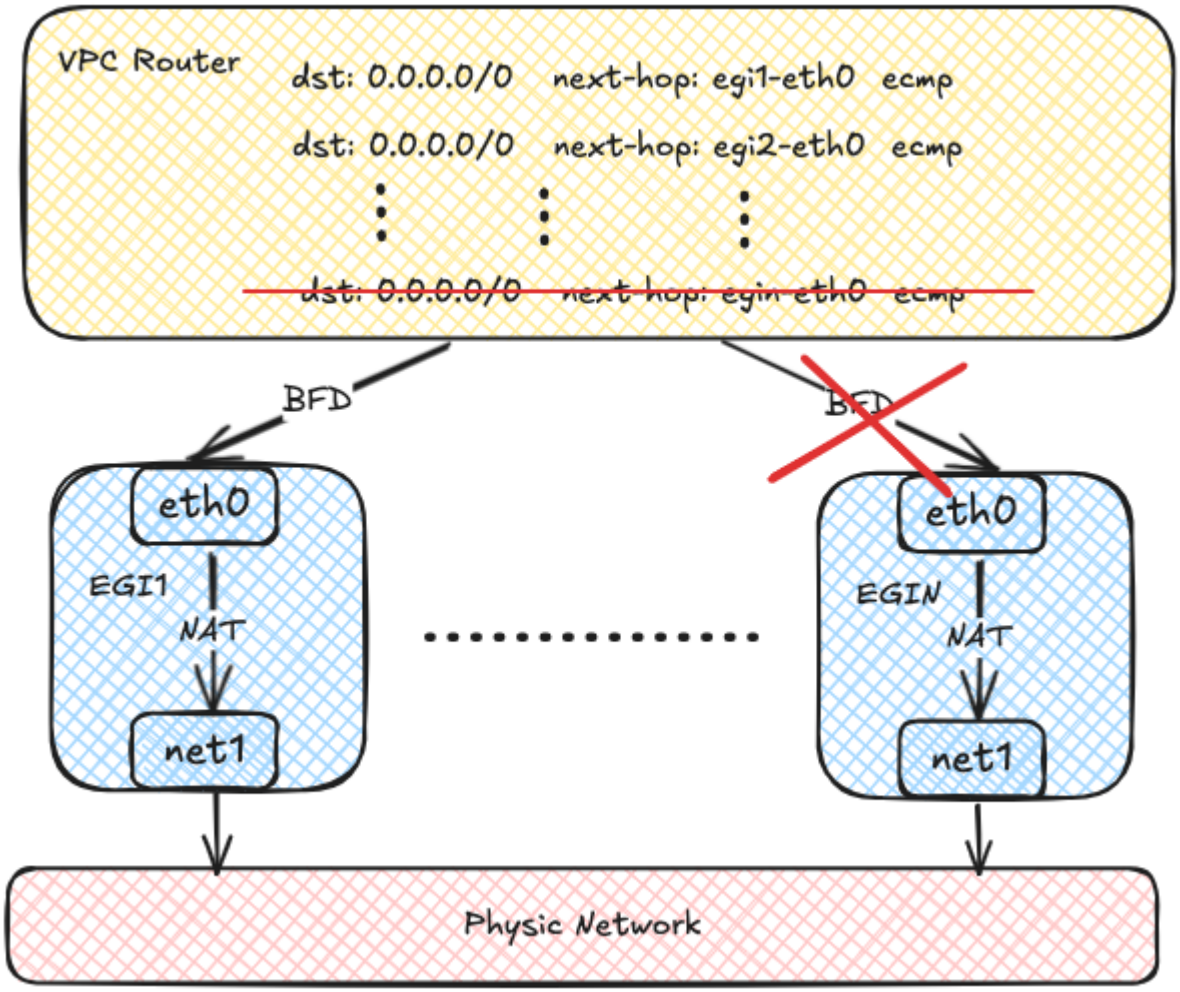
Egress Gateway Pod Pod VPC Macvlan Egress Gateway NAT



每个 Egress Gateway 实例会将自己的地址注册到 OVN 路由表内，当 VPC 内 Pod 需要访问外部网络时，OVN 会使用源地址哈希将流量转发到多个 Egress Gateway 实例地址，从而实现负载均衡，随着 Egress Gateway 实例数量的增加，吞吐量也可以实现横向扩展。



OVN ■■ BFD ■■■■■ Egress Gateway ■■■■■■■■■■ Egress Gateway ■■■■■■OVN ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■



5.2.2 ▯▯▯▯

VPC Egress Gateway ▯ VPC NAT Gateway ▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯ Multus-CNI▯

> ▯▯ VPC Egress Gateway ▯▯▯▯▯▯ ConfigMap▯

▯ Kube-OVN ▯▯▯▯▯ CNI ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ NetworkAttachmentDefinition ▯▯▯ Kube-OVN ▯▯▯▯▯▯▯▯ provider ▯▯▯▯ `<nad-name>.<nad-namespace>.ovn` ▯▯▯▯▯▯ ▯▯ NetworkAttachmentDefinition ▯▯▯▯▯▯▯▯▯▯

```yaml
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: veg-internal
  namespace: default
spec:
  config: |
    {
      "cniVersion": "0.3.1",
      "type": "kube-ovn",
      "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
      "provider": "veg-internal.default.ovn"
    }
---
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: veg-internal-subnet
spec:
  vpc: vpc1
  provider: veg-internal.default.ovn
  protocol: IPv4
  cidrBlock: 192.168.0.0/24
  gateway: 192.168.0.1
```

█ VPC Egress Gateway ███ `internalSubnet` ████████████████Multus ████ NetworkAttachmentDefinition ██ Pod ██████████ Pod ██████████ `eth0` █ `externalSubnet` ██████ NetworkAttachmentDefinition ██ `net1` ████████████████ CNI `eth0` ██████████ Pod██████████ `policies` ████████████ CNI ███████ `selectors` ███ provider ████████ NetworkAttachmentDefinition████████████ reconcile█

5.2.3 ████

██ NetworkAttachmentDefinition

VPC Egress Gateway ██████████ VPC ████████████ NetworkAttachmentDefinition ████████████ `macvlan` ████ Kube-OVN ██ IPAM ███████

```yaml
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: eth1
  namespace: default
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "macvlan",
      "master": "eth1",
      "mode": "bridge",
      "ipam": {
        "type": "kube-ovn",
        "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
        "provider": "eth1.default"
      }
    }'
---
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: macvlan1
spec:
  protocol: IPv4
  provider: eth1.default
  cidrBlock: 172.17.0.0/16
  gateway: 172.17.0.1
  excludeIps:
    - 172.17.0.0..172.17.0.10
```

> ███████ CNI ████ NetworkAttachmentDefinition █ VPC Egress Gateway ██████████

██████████████████ ██████

██ VPC Egress Gateway

██ VPC Egress Gateway ████████

```yaml
apiVersion: kubeovn.io/v1
kind: VpcEgressGateway
metadata:
  name: gateway1
  namespace: default
spec:
  vpc: ovn-cluster
  replicas: 1
  externalSubnet: macvlan1
  policies:
    - snat: true
      subnets:
        - ovn-default
```

███████ default ███████ VPC `ovn-cluster` ███████ gateway1 ████ VPC Egress Gateway█ `ovn-cluster` █ `ovn-default` ███10.16.0.0/16██████ Pod ███ `macvlan1` ███ SNAT ███████████

████████ CNI ████████████████████████████████████

```yaml
apiVersion: kubeovn.io/v1
kind: VpcEgressGateway
metadata:
  name: gateway-nonprimary
  namespace: default
spec:
  vpc: vpc1
  replicas: 1
  internalSubnet: veg-internal-subnet
  externalSubnet: macvlan1
  policies:
    - snat: true
```


2025 Kube-OVN Team

```
      subnets:
        - veg-internal-subnet
```

... VPC Egress Gateway

```
$ kubectl get veg gateway1
NAME       VPC           REPLICAS   BFD ENABLED   EXTERNAL SUBNET   PHASE       READY   AGE
gateway1   ovn-cluster   1          false         macvlan1          Completed   true    13s
```

...

```
kubectl get veg gateway1 -o wide
NAME       VPC           REPLICAS   BFD ENABLED   EXTERNAL SUBNET   PHASE       READY   INTERNAL IPS     EXTERNAL IPS      WORKING NODES         AGE
gateway1   ovn-cluster   1          false         macvlan1          Completed   true    ["10.16.0.12"]   ["172.17.0.11"]   ["kube-ovn-worker"]   82s
```

...

```
$ kubectl get deployment -l ovn.kubernetes.io/vpc-egress-gateway=gateway1
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
gateway1   1/1     1            1           4m40s

$ kubectl get pod -l ovn.kubernetes.io/vpc-egress-gateway=gateway1 -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP           NODE              NOMINATED NODE   READINESS GATES
gateway1-b9f8b4448-76lhm   1/1     Running   0          4m48s   10.16.0.12   kube-ovn-worker   <none>           <none>
```

.. Pod .. IP.... iptables ...

```
$ kubectl exec gateway1-b9f8b4448-76lhm -c gateway -- ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: net1@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 62:d8:71:90:7b:86 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.11/16 brd 172.17.255.255 scope global net1
       valid_lft forever preferred_lft forever
    inet6 fe80::60d8:71ff:fe90:7b86/64 scope link
       valid_lft forever preferred_lft forever
17: eth0@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP group default
    link/ether 36:7c:6b:c7:82:6b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.16.0.12/16 brd 10.16.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::347c:6bff:fec7:826b/64 scope link
       valid_lft forever preferred_lft forever

$ kubectl exec gateway1-b9f8b4448-76lhm -c gateway -- ip route show
default via 172.17.0.1 dev net1
10.16.0.0/16 dev eth0 proto kernel scope link src 10.16.0.12
172.17.0.0/16 dev net1 proto kernel scope link src 172.17.0.11

$ kubectl exec gateway1-b9f8b4448-76lhm -c gateway -- iptables -t nat -S
-P PREROUTING ACCEPT
-P INPUT ACCEPT
-P OUTPUT ACCEPT
-P POSTROUTING ACCEPT
-A POSTROUTING -s 10.16.0.0/16 -j MASQUERADE --random-fully
```

. Gateway Pod ...

```
$ kubectl exec -ti gateway1-b9f8b4448-76lhm -c gateway -- bash
nobody@gateway1-b9f8b4448-76lhm:/kube-ovn$ tcpdump -i any -nnve icmp and host 172.17.0.1
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
06:50:58.936528 eth0  In  ifindex 17 92:26:b8:9e:f2:1c ethertype IPv4 (0x0800), length 104: (tos 0x0, ttl 63, id 30481, offset 0, flags [DF], proto ICMP (1),
length 84)
    10.16.0.9 > 172.17.0.1: ICMP echo request, id 37989, seq 0, length 64
06:50:58.936574 net1  Out ifindex 2 62:d8:71:90:7b:86 ethertype IPv4 (0x0800), length 104: (tos 0x0, ttl 62, id 30481, offset 0, flags [DF], proto ICMP (1),
length 84)
    172.17.0.11 > 172.17.0.1: ICMP echo request, id 39449, seq 0, length 64
06:50:58.936613 net1  In  ifindex 2 02:42:39:79:7f:08 ethertype IPv4 (0x0800), length 104: (tos 0x0, ttl 64, id 26701, offset 0, flags [none], proto ICMP
(1), length 84)
    172.17.0.1 > 172.17.0.11: ICMP echo reply, id 39449, seq 0, length 64
06:50:58.936621 eth0  Out ifindex 17 36:7c:6b:c7:82:6b ethertype IPv4 (0x0800), length 104: (tos 0x0, ttl 63, id 26701, offset 0, flags [none], proto ICMP
(1), length 84)
    172.17.0.1 > 10.16.0.9: ICMP echo reply, id 37989, seq 0, length 64
```

OVN Logical Router ...

```
$ kubectl ko nbctl lr-policy-list ovn-cluster
Routing Policies
     31000                           ip4.dst == 10.16.0.0/16           allow
```

```
     31000                          ip4.dst == 100.64.0.0/16           allow
     30000                             ip4.dst == 172.18.0.2         reroute               100.64.0.3
     30000                             ip4.dst == 172.18.0.3         reroute               100.64.0.2
     30000                             ip4.dst == 172.18.0.4         reroute               100.64.0.4
     29100                           ip4.src == 10.16.0.0/16         reroute               10.16.0.12
     29000 ip4.src == $ovn.default.kube.ovn.control.plane_ip4         reroute               100.64.0.2
     29000       ip4.src == $ovn.default.kube.ovn.worker2_ip4         reroute               100.64.0.4
     29000        ip4.src == $ovn.default.kube.ovn.worker_ip4         reroute               100.64.0.3
```

... `.spec.replicas` ...

```
$ kubectl scale veg gateway1 --replicas=2
vpcegressgateway.kubeovn.io/gateway1 scaled

$ kubectl get veg gateway1
NAME       VPC           REPLICAS   BFD ENABLED   EXTERNAL SUBNET   PHASE       READY   AGE
gateway1   ovn-cluster   2          false         macvlan           Completed   true    39m

$ kubectl get pod -l ovn.kubernetes.io/vpc-egress-gateway=gateway1 -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP           NODE               NOMINATED NODE   READINESS GATES
gateway1-b9f8b4448-76lhm   1/1     Running   0          40m   10.16.0.12   kube-ovn-worker    <none>           <none>
gateway1-b9f8b4448-zd4dl   1/1     Running   0          64s   10.16.0.13   kube-ovn-worker2   <none>           <none>

$ kubectl ko nbctl lr-policy-list ovn-cluster
Routing Policies
     31000                            ip4.dst == 10.16.0.0/16           allow
     31000                          ip4.dst == 100.64.0.0/16            allow
     30000                             ip4.dst == 172.18.0.2         reroute               100.64.0.3
     30000                             ip4.dst == 172.18.0.3         reroute               100.64.0.2
     30000                             ip4.dst == 172.18.0.4         reroute               100.64.0.4
     29100                           ip4.src == 10.16.0.0/16         reroute               10.16.0.12, 10.16.0.13
     29000 ip4.src == $ovn.default.kube.ovn.control.plane_ip4         reroute               100.64.0.2
     29000       ip4.src == $ovn.default.kube.ovn.worker2_ip4         reroute               100.64.0.4
     29000        ip4.src == $ovn.default.kube.ovn.worker_ip4         reroute               100.64.0.3
```

... **PrimaryCNI** ...

... CNI ... Underlay... `PrimaryCNI` ... Kube-OVN v1.17 ... CNI ...v1 chart ... `cni_conf.NON_PRIMARY_CNI=true` ...v2 chart ... `cni.nonPrimaryCNI=true` ... Pod ... `eth0` ... Multus attachment ... VPC ... Kube-OVN ... attachment ... NetworkAttachmentDefinition ... Kube-OVN ... attachment ... VPC ... CNI ...

```yaml
apiVersion: kubeovn.io/v1
kind: VpcEgressGateway
metadata:
  name: gateway-primary-cni
  namespace: default
spec:
  vpc: ovn-cluster
  replicas: 1
  externalNetworkMode: PrimaryCNI
  internalSubnet: gateway-internal
  policies:
    - subnets:
        - workload-subnet
      snat: true
```

PrimaryCNI ... CNI ... NAT ... raw ...OVN ... attachment ... IP ... `status.internalIPs` ... CNI Pod IP ... `status.externalIPs` ...IPv4 SNAT ... IPv6 NAT66 ... attachment ... `eth0` ...

... `snat` ... `false` ... VPC ...PrimaryCNI ... `externalSubnet` ... `externalIPs` ... `externalIPPool` ...BGP...EVPN...selectors... bandwidth ... observability ... `externalNetworkMode` ... `Attachment` ... `externalSubnet` ...

... **Egress Gateway IP** ...

... `externalIPs` ... `nodeSelector` ... Egress Gateway Pods ... Egress IP ...

```yaml
apiVersion: kubeovn.io/v1
kind: VpcEgressGateway
metadata:
  name: gateway1
  namespace: default
spec:
  vpc: ovn-cluster
  replicas: 2
  externalSubnet: macvlan1
  policies:
    - snat: true
      subnets:
        - ovn-default
  externalIPs:
    - 172.17.0.10
```

```
    - 172.17.0.11
  nodeSelector:
    - matchLabels:
        kubernetes.io/hostname: kube-ovn-worker
```

... IP ... `internalIPPool` ... `externalIPPool` ... `IPPool` ...IPPool ... `internalSubnet` ... `externalSubnet` ... `internalIPs` ... `externalIPs` ...

```
spec:
  internalIPPool: veg-internal
  externalIPPool: veg-external
```

... **BFD** ...

BFD ... VPC ... BFD LRP ... VPC ... BFD Port...

```
apiVersion: kubeovn.io/v1
kind: Vpc
metadata:
  name: vpc1
spec:
  bfdPort:
    enabled: true
    ip: 10.255.255.255
---
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: subnet1
spec:
  vpc: vpc1
  protocol: IPv4
  cidrBlock: 192.168.0.0/24
```

... BFD Port ... OVN LR ... BFD ... LRP...

```
$ kubectl ko nbctl show vpc1
router 0c1d1e8f-4c86-4d96-88b2-c4171c7ff824 (vpc1)
    port bfd@vpc1
        mac: "8e:51:4b:16:3c:90"
        networks: ["10.255.255.255"]
    port vpc1-subnet1
        mac: "de:c9:5c:38:7a:61"
        networks: ["192.168.0.1/24"]
```

... VPC Egress Gateway ... `.spec.bfd.enabled` ... `true` ...

```
apiVersion: kubeovn.io/v1
kind: VpcEgressGateway
metadata:
  name: gateway2
  namespace: default
spec:
  vpc: vpc1
  replicas: 2
  internalSubnet: subnet1
  externalSubnet: macvlan
  bfd:
    enabled: true
  policies:
    - snat: true
      ipBlocks:
        - 192.168.0.0/24
```

... VPC Egress Gateway ...

```
$ kubectl get veg gateway2 -o wide
NAME       VPC    REPLICAS   BFD ENABLED   EXTERNAL SUBNET   PHASE       READY   INTERNAL IPS                    EXTERNAL IPS                    WORKING
NODES                        AGE
gateway2   vpc1   2          true          macvlan           Completed   true    ["192.168.0.2","192.168.0.3"]   ["172.17.0.13","172.17.0.14"]   ["kube-ovn-
worker","kube-ovn-worker2"]   58s

$ kubectl get pod -l ovn.kubernetes.io/vpc-egress-gateway=gateway2 -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP            NODE               NOMINATED NODE   READINESS GATES
gateway2-fcc6b8b87-8lgvx    1/1     Running   0          2m18s   192.168.0.3   kube-ovn-worker2   <none>           <none>
gateway2-fcc6b8b87-wmww6    1/1     Running   0          2m18s   192.168.0.2   kube-ovn-worker    <none>           <none>

$ kubectl ko nbctl lr-policy-list vpc1
Routing Policies
     29100                     ip4.src == 192.168.0.0/24         reroute               192.168.0.2, 192.168.0.3
     29090                     ip4.src == 192.168.0.0/24            drop
```


2025 Kube-OVN Team

```
$ kubectl ko nbctl list bfd
_uuid               : 223ede10-9169-4c7d-9524-a546e24bfab5
detect_mult         : 3
dst_ip              : "192.168.0.2"
external_ids        : {af="4", vendor=kube-ovn, vpc-egress-gateway="default/gateway2"}
logical_port        : "bfd@vpc1"
min_rx              : 1000
min_tx              : 1000
options             : {}
status              : up

_uuid               : b050c75e-2462-470b-b89c-7bd38889b758
detect_mult         : 3
dst_ip              : "192.168.0.3"
external_ids        : {af="4", vendor=kube-ovn, vpc-egress-gateway="default/gateway2"}
logical_port        : "bfd@vpc1"
min_rx              : 1000
min_tx              : 1000
options             : {}
status              : up
```

██ Pod ██ BFD ███

```
$ kubectl exec gateway2-fcc6b8b87-8lgvx -c bfdd -- bfdd-control status
There are 1 sessions:
Session 1
 id=1 local=192.168.0.3 (p) remote=10.255.255.255 state=Up

$ kubectl exec gateway2-fcc6b8b87-wmww6 -c bfdd -- bfdd-control status
There are 1 sessions:
Session 1
 id=1 local=192.168.0.2 (p) remote=10.255.255.255 state=Up
```

████

VPC BFD PORT

██	██	██	███	██	██
`enabled`	`boolean`	▯	`false`	████ BFD Port▯	`true`
`ip`	`string`	▯	-	BFD Port ███ IP ███ ██████████████ IPv6 ▯ ███	`169.255.255.2` `fdff::1` / `169.255.255.2`
`nodeSelector`	`object`	▯	-	███████ BFD Port ███ ████████BFD Port ▯ ████████████████ OVN HA Chassis Group███ Active/ Backup ███████ Active ██████████ nodeSelector▯ Kube-OVN ████████ ██████████ `kubectl ko nbctl list ha_chassis_group` ███████ OVN HA Chassis Group ███	-
`nodeSelector.matchLabels`	`dict/map`	▯	-	██████████████	-
`nodeSelector.matchExpressions`	`object array`	▯	-	██████████████	-

2025 Kube-OVN Team

VPC EGRESS GATEWAY

Spec▯

...	...	...	...	...	...
`vpc`	`string`	...	... VPC ...ovn-cluster...	VPC ...	`vpc1`
`replicas`	`integer/int32`	...	`1`	...	`2`
`prefix`	`string`	...	-	... Deployment ...	`veg-`
`image`	`string`	...	-	... Deployment ...	`docker.io/kubeovn/kube-ovn:v1.14.0-debug`
`externalNetworkMode`	`string`	...	`Attachment`	... `Attachment` ... `externalSubnet` ... `PrimaryCNI` ... Pod ... CNI ... `eth0` ... Multus attachment ... Kube-OVN ...	`PrimaryCNI`
`internalSubnet`	`string`	...	VPC ...	... VPC ...	`subnet1`
`externalSubnet`	`string`	...	-	...	`ext1`
`externalNetworkMode`	`string`	...	`Attachment`	... `Attachment` ... externalSubnet ... `PrimaryCNI` ... Pod ...PrimaryCNI ... externalIP/externalIPPool/BGP/EVPN/selectors/bandwidth/observability... CNI ...	`PrimaryCNI`
`internalIPs`	`string array`	...	-	... VPC ... IP ... IPv6 ... IP ... `<replicas> + 1` ... Pod ...	`10.16.0.101` / `fd00::11` / `10.16.0.101,fd00::11`
`externalIPs`	`string array`	...	-	... IP ... IPv6 ... IP ... `<replicas> + 1` ... Pod ...	`10.16.0.101` / `fd00::11` / `10.16.0.101,fd00::11`
`internalIPPool`	`string`	...	-	... `IPPool` ...IPPool ... `internalSubnet` ...	`veg-internal`

2025 Kube-OVN Team

...	...	...	...	...	...
				`internalIPs` ...	
`externalIPPool`	`string`	...	-	... `IPPool` ...IPPool ... `externalSubnet` ... `externalIPs` ...	`veg-external`
`bfd`	`object`	...	-	BFD ...	-
`policies`	`object array`	...	-	Egress ... `selectors` ...	-
`selectors`	`object array`	...	-	... Namespace Selector ... Pod Selector ... Egress ... Pod ... SNAT/ MASQUERADE... `policies` ...	-
`nodeSelector`	`object array`	...	-	...Deployment/ Pod...	-
`trafficPolicy`	`string`	...	`Cluster`	... `Cluster` / `Local` ... **BFD** ... ... `Local` ... Egress ... VPC Egress Gateway ... VPC Egress Gateway ... Egress ...	`Local`
`podAntiAffinity`	`string`	...	`Required`	Pod ... `Required` ... ... `Preferred` ... ... `Preferred` ... `Required` ... Pod ...	`Preferred`
`priorityClassName`	`string`	...	-	... Pod ... PriorityClass ... ... Pod ...	`high-priority`
`bandwidth`	`object`	...	-	...	-
`bandwidth.ingress`	`integer` `string`	...	-	... VPC ...	`1024` / `1Gi`
`bandwidth.egress`	`integer` `string`	...	-	... VPC ...	`1024` / `1Gi`

...

...... Mbps ... `M` ... `Mi` ... `G` ... `Gi` ... quantity ... bit/s..... Mbps... `100M` ... `100Mi` ... `1G` ... `1Gi` ... 100...105...1000 ... 1074 Mbps...

2025 Kube-OVN Team

████████████████ 9223372036854 Mbps██ `ingress` ▯ `egress` ████████████████████████████

```
spec:
  bandwidth:
    ingress: 1024
    egress: 1Gi
```

> ⚠ ████████

> ████0 ▯ 9223372036854 ████████████████ Mbps ███████ Kube-OVN ████████▯ `vpc-egress-gateways.kubeovn.io` CRD ██████████ `x-kubernetes-int-or-string: true` ███████████████
>
> Helm ████ Chart ▯ `crds/` ████ CRD▯▯▯ `kube-ovn-v2` Chart ███████ `KUBE_OVN_VERSION` ██████████ Kube-OVN ████████████████████████ CRD bundle▯
>
> ```
> kubectl apply -f "https://raw.githubusercontent.com/kubeovn/kube-ovn/${KUBE_OVN_VERSION}/charts/kube-ovn-v2/crds/kube-ovn-crd.yaml"
> ```
>
> ████████████████████████████████ quantity ████████████ Mbps ▯▯

BFD ▯▯▯

▯▯	▯▯	▯▯	▯▯▯	▯▯	▯▯
`enabled`	`boolean`	▯	`false`	████ BFD▯	`true`
`minRX`	`integer/int32`	▯	`1000`	BFD minRX ▯ ▯ms▯▯	`500`
`minTX`	`integer/int32`	▯	`1000`	BFD minTX ▯ ▯ms▯▯	`500`
`multiplier`	`integer/int32`	▯	`3`	BFD multiplier ▯▯	`1`

Egress ▯▯▯

▯▯	▯▯	▯▯	▯▯▯	▯▯	▯▯
`snat`	`boolean`	▯	`false`	████ SNAT/ MASQUERADE▯	`true`
`ipBlocks`	`string array`	▯	-	████ Gateway ▯ IP ███████ IPv6▯	`192.168.0.1` / `192.168.0.0/24`
`subnets`	`string array`	▯	-	████ Gateway ▯ VPC ████████ IPv6 ████████	`subnet1`

2025 Kube-OVN Team

Selectors：

██	██	██	███	██	██
`namespaceSelector`	`object`	：	-	Namespace ████ ████ Namespace：	-
`namespaceSelector.matchLabels`	`dict/map`	：	-	████████	-
`namespaceSelector.matchExpressions`	`object array`	：	-	████████	-
`podSelector`	`object`	：	-	Pod ████ Pod：	-
`podSelector.matchLabels`	`dict/map`	：	-	████████	-
`podSelector.matchExpressions`	`object array`	：	-	████████	-

██████

██	██	██	███	██	██
`matchLabels`	`dict/map`	：	-	████████	-
`matchExpressions`	`object array`	：	-	████████	-
`matchFields`	`object array`	：	-	████████	-

Status：

██	██	██	██
`ready`	`boolean`	Gateway ████	`true`
`phase`	`string`	Gateway ████	`Pending` / `Processing` / `Completed`
`internalIPs`	`string array`	██ VPC ████ IP ██	-
`externalIPs`	`string array`	████████ IP ██	-
`workload`	`object`	██████	-
`workload.apiVersion`	`string`	████ API ██	`apps/v1`
`workload.kind`	`string`	██████	`Deployment`
`workload.name`	`string`	██████	`gateway1`
`workload.nodes`	`string array`	████████	-
`conditions`	`object array`	-	-

PDF Slack Support

2025 Kube-OVN Team

2026⬚11⬚7⬚

2024⬚12⬚12⬚

GitHub



+1

5.2.4 ⬚⬚

5.3 VPC Egress Gateway 管理

VPC Egress Gateway 可以在每个网关 Pod 中运行一个原生可观测性 sidecar。该 sidecar 可以暴露内部和外部网卡计数器、低基数 conntrack NAT 指标，并将版本化 NAT 流日志写入容器标准输出。这三项功能默认均禁用，不影响网关数据平面就绪状态。

5.3.1 使用要求和升级说明

- observer 使用 restartable init container，因此需要 Kubernetes 1.29 或更高版本，并启用 `SidecarContainers` 特性。控制器会对一个零副本 Deployment 执行服务端 dry-run，并验证 API Server 是否保留 restartable init 策略。如果版本过低或未启用该特性，控制器不会注入 observer，并设置 `ObservabilityConfigured=False`；网关数据平面仍会正常协调。控制器会重试短暂的能力探测错误；无法验证能力时，已存在的 observer 保持最后一个已知良好配置。
- 创建使用 `spec.observability` 的资源前，需要显式升级 `vpc-egress-gateways.kubeovn.io` CRD。Helm 不会自动升级已安装的 CRD。请使用常规 CRD 升级流程，应用与 Kube-OVN 版本匹配的 CRD。
- conntrack 采集需要网关 Pod 网络命名空间中的 `NET_ADMIN`。官方 observer binary 仅带有 `CAP_NET_ADMIN` file capability，生成的容器 security context 仅允许 `NET_ADMIN` 进入 capability bounding set。`allowPrivilegeEscalation` 已启用，使该可信 file capability 在非 root launcher 执行 `exec` 后仍然生效；observer 仍以 UID 和 GID 65534 运行，丢弃其他所有 capability，并使用只读根文件系统。
- Prometheus Operator 是可选依赖。缺少 ServiceMonitor CRD 时，网关仍可正常工作。

5.3.2 启用可观测性

以下示例启用两个指标采集器和 JSON 流日志：

```yaml
apiVersion: kubeovn.io/v1
kind: VpcEgressGateway
metadata:
  name: gateway1
  namespace: default
spec:
  vpc: ovn-cluster
  replicas: 2
  externalSubnet: macvlan1
  policies:
    - snat: true
      subnets:
        - ovn-default
  observability:
    resources:
      requests:
        cpu: 20m
        memory: 64Mi
      limits:
        cpu: 200m
        memory: 256Mi
    interfaceMetrics:
      enabled: true
    conntrack:
      metrics:
        enabled: true
      log:
        enabled: true
        events: [start, end]
        rateLimit:
          recordsPerSecond: 100
          burst: 1000
        filters:
          include: []
          exclude: []
    serviceMonitor:
      labels:
        monitoring: platform
      annotations: {}
```

`resources` 为空或省略时，observer 的 request 为 `20m` CPU 和 `64Mi` 内存，limit 为 `200m` CPU 和 `256Mi` 内存。interface-only 模式会持续测试，以确保稳态常驻内存不超过 `20MiB`；更大的默认 request 为 conntrack cache 和日志队列增长保留了空间。

控制器将运行时配置存储在每个网关的 ConfigMap 中。采集器开关、流日志事件、filter 和 rate limit 可在不替换网关 Pod 的情况下热加载。以下更改会更新 Deployment 并替换 Pod：

- 首次启用可观测性。
- 禁用所有可观测性功能。
- 更改 `observability.resources`。
- 更改网关工作负载镜像。

2025 Kube-OVN Team

□□□□□□ConfigMap □ Service □□□□□□□□□□□□□□□□□ condition□□□□□□ observer □□□volume □□□ Pod□

□□□□□□□□□□□□□□□□ `/kube-ovn/vpc-egress-gateway-observer` □launcher □□□□□ `sleep infinity` □□□□ binary □□□□□□□□□□□ observer □□□□□□□□□□□ `CAP_NET_ADMIN` file capability□□□□□□□□□□□□ conntrack □□□□□□□□□□□□□□□□

5.3.3 □□

observer □ TCP □□ `10666` □□□ `/metrics` □ `/healthz` □observer □□□□ Prometheus registry□□□□ `go_*` □ `process_*` □ `promhttp_*` □□□□□□□□□□ `namespace` □ `name` □ `pod` □ `node` □□□□□

□□□□□□liveness probe □□□□□□□ observer binary□□□□ loopback □□ `/healthz` □□□ observer □□□□□□□□ Pod □□□□□□□□□□□□□□□□□□□□ observer binary □□□ `sleep infinity` □□□□□□□□□□

□□□□

□□ scrape □□□ `/proc/net/dev` □□□□□□observer □□□□□ scrape □□□ Pod `network-status` annotation □□□□□□□□ Multus □□□□□□□□□□□□□□□□□□□□

□□	□□□□	□□
`kube_ovn_vpc_egress_gateway_interface_rx_bytes_total`	`interface` □ `type`	□□□□□□
`kube_ovn_vpc_egress_gateway_interface_tx_bytes_total`	`interface` □ `type`	□□□□□□
`kube_ovn_vpc_egress_gateway_interface_rx_packets_total`	`interface` □ `type`	□□□□□□
`kube_ovn_vpc_egress_gateway_interface_tx_packets_total`	`interface` □ `type`	□□□□□□
`kube_ovn_vpc_egress_gateway_interface_drops_total`	`interface` □ `type` □ `direction`	□□□□□□
`kube_ovn_vpc_egress_gateway_interface_errors_total`	`interface` □ `type` □ `direction`	□□□□□□

`type` □□□ `internal` □ `external` □ `direction` □ `rx` □ `tx` □

Conntrack □□

Conntrack □□□□□ IP □□□□□□□□□□□□□□□□□□□□

□□	□□□□	□□
`kube_ovn_vpc_egress_gateway_conntrack_nat_flows_active`	`address_family` □ `protocol` □ `nat_type`	□□ NAT □□□□ conntrack table dump □□□□□□□□□□□
`kube_ovn_vpc_egress_gateway_conntrack_nat_flows_started_total`	`address_family` □ `protocol` □ `nat_type`	observer □□□□□□□ NAT □□□□□□
`kube_ovn_vpc_egress_gateway_conntrack_nat_flows_ended_total`	`address_family` □ `protocol` □ `nat_type`	□□□□ NAT □□□□□□□□□
`kube_ovn_vpc_egress_gateway_conntrack_nat_packets_total`	`address_family` □ `protocol` □ `nat_type` □ `direction`	conntrack accounting □□□□□□□□□□□
`kube_ovn_vpc_egress_gateway_conntrack_nat_bytes_total`	`address_family` □ `protocol` □ `nat_type` □ `direction`	conntrack accounting □□□□□□□□□□□

`address_family` □ `ipv4` □ `ipv6` □□□ `protocol` □□ `tcp` □ `udp` □ `sctp` □ `icmp` □ `icmpv6` □□□□□□□□□□□ `other` □ `nat_type` □ `snat` □ `dnat` □ `snat_dnat` □ `direction` □ `original` □ `reply` □

□□□□□□ `kube_ovn_vpc_egress_gateway_observability_*` □□□□□□□□□□□□□□□□□□□□□□□□conntrack □□□□□□cache □□□□□□□□□□□□□□□□□□□□□□□□ conntrack accounting □□□□□□□□□□□□□□□□ `/metrics` □□□□□□□□□□□□□□□□□□□

2025 Kube-OVN Team

5.3.4 ServiceMonitor □□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `publishNotReadyAddresses: true` □ headless Service□□□□ `targetPort` □ `10666` □□□□□□□□□□□□□□□□□□ ServiceMonitor□ `observability.serviceMonitor` □□□□ label □ annotation □□□□ ServiceMonitor □□□□□□□□□□□□□□□□ selector label□

□□ ServiceMonitor CRD □□□□□□□□□□□ `ServiceMonitorReady=False` □□□□□□CRD □□□□□□□□□□□□□ ServiceMonitor□□ condition □□□□□ `Ready` condition □□□□□

5.3.5 JSON □□□

□□□□ JSON Lines □□□□□ `observability` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
kubectl logs -n default <gateway-pod> -c observability
```

`v1` schema □□ `start` □ `end` □□□□□□□□□□□□□ conntrack update □□□□□□□□□□□

```
{"schemaVersion":"v1","timestamp":"2026-08-05T03:40:00Z","event":"end","conntrackID":2471802592,"zone":0,"namespace":"default","name":"gateway1","pod":"gateway1
r9x8m","node":"worker-1","addressFamily":"ipv4","protocol":"tcp","protocolNumber":6,"natType":["snat"],"original":
{"sourceIP":"10.16.0.25","sourcePort":42136,"destinationIP":"203.0.113.10","destinationPort":443},"translated":
{"sourceIP":"172.17.0.11","sourcePort":42136,"destinationIP":"203.0.113.10","destinationPort":443},"counters":
{"originalPackets":8,"originalBytes":624,"replyPackets":7,"replyBytes":591}}
```

□□□□□ `other` □□□□ `protocolNumber` □□□□ IP □□□□ `natType` □□□□□□□□□□□□□□□ `dnat` □ `snat` □conntrack accounting □□□□□□ `counters` □□□observer □□□□ `nf_conntrack_acct` □

observer □ dump □□ conntrack table □□□ conntrack □□□□□ NAT □□□□□ active-flow □□□ cache□□□□□□□□□ `start` □□□cache □□□ 65,536 □□□□netlink □□ channel □ □□□□□□□□□□ 4,096 □□□□□□□□□ consumer □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ conntrack □□□

□□□□□□□□□□□□ NAT □□□□□□□□□□□□□□ Pod □□□□□ 100 □□□□□□□□□□□□□ 1,000□□□□□□□□□□ best-effort □□□□□□□□□□□□□□□□□□□

5.3.6 □□□ Filter

□□ include □ exclude □□□□□□□□□□□□□NAT □□□□□□□□□□□ tuple □□□□□□□□□□□□□ OR □□□□□□□□□□□□□□□□□ AND □□□□□□□□□□□□□ OR □□□exclude □□□□□ include □□□□ include □ □□□□□ NAT □□

□□□□□□□□□□□□□□□□□□□□□□□□□□ `10.16.0.0/16` □□□□□ 80 □ 443 □ TCP SNAT □□□□□□□□□□□□□□□□□ `192.0.2.0/24` □□□

```yaml
spec:
  observability:
    conntrack:
      log:
        enabled: true
        filters:
          include:
            - protocols: [tcp]
              natTypes: [snat]
              original:
                sourceCIDRs: [10.16.0.0/16]
                destinationPorts:
                  - start: 80
                    end: 443
          exclude:
            - translated:
                destinationCIDRs: [192.0.2.0/24]
```

PDF Slack Support

2026□8□10□

2026□8□10□

GitHub 

5.3.7 □□

5.4 Egress Gateway BGP 和 EVPN 支持

实验目标

BGP 和 EVPN 支持为租户网络提供了与外部网络进行动态路由交换的能力。

学习内容：

- 理解 L3VPN和以太网 L2VPN
- 掌握 FRR 在容器环境中的应用
- 掌握 BGP BFD

5.4.1 概述

VPC Egress Gateway 支持通过 BGP 和 EVPN/VXLAN 两种方式将流量从 Egress Gateway Pod 中的 FRR（Free Range Routing）转发出去。BGP 和 EVPN 两种模式均支持 Gateway 的 init 容器自动生成 BgpConf 和 EvpnConf 并启动 FRR 进程。

在 BGP 模式下，Egress Gateway 通过 BGP 协议将其所在节点作为下一跳发布给外部路由器。EVPN 模式下，init 容器会为 Pod 所在的 Linux VRF、bridge 和 VXLAN 设备以及 L2VPN EVPN 所需的路由信息配置好 VXLAN 隧道，使 Egress Gateway Pod 可以通过Kubernetes 集群内部署的 Geneve 隧道

> 本实验将使用 BGP 模式，通过 kube-ovn-speaker 将 Pod/子网/Service 路由发布

5.4.2 实验拓扑

在 BGP/EVPN 模式下，通过 VPC Egress Gateway 可以使用 Multus-CNI 网络

本实验将使用 BGP协议与 EVPN协议与外部网络进行 BGP 交互

5.4.3 实验步骤

步骤 BgpConf 配置

BgpConf 用于配置与外部网络的 BGP 邻居关系

```
apiVersion: kubeovn.io/v1
kind: BgpConf
metadata:
  name: bgp-conf-6502
spec:
  localASN: 65002
  peerASN: 65001
  neighbours:
    - 10.0.1.1
  holdTime: 90s
  keepaliveTime: 30s
  connectTime: 10s
  ebgpMultiHop: true
```

本实验将使用 AS 号为 65002的 AS 号为 65001 的 BGP 邻居，邻居地址为 10.0.1.1，并启用 EBGP Multi-Hop。

步骤 EvpnConf 配置

EvpnConf 用于配置与外部网络的 EVPN 信息。EvpnConf 需要与 BgpConf 配合使用

```
apiVersion: kubeovn.io/v1
kind: EvpnConf
metadata:
  name: evpn-conf-1016
spec:
  vni: 1016
  routeTargets:
    - "65000:1016"
```


2025 Kube-OVN Team

■■■■■■■■■■ VNI ■ 1016 ■ EVPN ■■■Route Target ■ `65000:1016` ■

■■■■ BGP/EVPN ■ VPC Egress Gateway

■■■■ VPC Egress Gateway ■■■■ NetworkAttachmentDefinition ■■■■■■■■

■■■ VpcEgressGateway ■ `.spec` ■■■ `bgpConf` ■ `evpnConf` ■■■■■■■■■■■■■■■■■■■■

```yaml
apiVersion: kubeovn.io/v1
kind: VpcEgressGateway
metadata:
  name: gateway1
  namespace: default
spec:
  vpc: ovn-cluster
  replicas: 1
  externalIPs:
    - 10.0.1.13
  internalIPs:
    - 10.16.0.13
  externalSubnet: macvlan1
  bgpConf: bgp-conf-6502
  evpnConf: evpn-conf-1016
  policies:
    - snat: false
      subnets:
        - ovn-default
    - snat: false
      ipBlocks:
        - 10.17.0.0/16
```

■■■■■■■■■■■■ BGP ■ EVPN ■ VPC Egress Gateway■Gateway Pod ■■■■ FRR ■■■■■ BGP EVPN ■■■■■■■■■■■■■■■■■■ VXLAN ■■■■■■■■■■■■■ BGP/EVPN ■■■■■■■■ `snat` ■■■ `false` ■

■■■ BGP■■■■ EVPN■

■■■■■■ BGP ■■■■■■■■■■■ EVPN/VXLAN■■■■■■ `bgpConf` ■■■■■■■ `evpnConf` ■■■■■■■■ FRR ■■■■ BGP IPv4 Unicast ■■■■■■■■■■

```yaml
apiVersion: kubeovn.io/v1
kind: VpcEgressGateway
metadata:
  name: gateway-bgp-only
  namespace: default
spec:
  vpc: ovn-cluster
  replicas: 1
  externalSubnet: macvlan1
  bgpConf: bgp-conf-6502
  policies:
    - snat: true
      subnets:
        - ovn-default
```

5.4.4 ▯▯▯▯

BgpConf

▯▯	▯▯	▯▯	▯▯▯	▯▯	▯▯
`localASN`	`uint32`	▯	-	▯▯ AS ▯▯	`65002`
`peerASN`	`uint32`	▯	-	▯▯ AS ▯▯	`65001`
`routerId`	`string`	▯	Pod IP	BGP Router ID▯	`10.0.1.13`
`neighbours`	`string array`	▯	-	BGP ▯▯ IP ▯▯▯▯▯	`10.0.1.1`
`password`	`string`	▯	-	BGP ▯▯▯▯▯	`secret`
`holdTime`	`string (duration)`	▯	-	BGP Hold ▯▯▯	`90s`
`keepaliveTime`	`string (duration)`	▯	-	BGP Keepalive ▯ ▯▯	`30s`
`connectTime`	`string (duration)`	▯	-	BGP ▯▯▯▯▯▯	`10s`
`ebgpMultiHop`	`boolean`	▯	`false`	▯▯▯▯ EBGP Multi-Hop▯	`true`

EvpnConf

▯▯	▯▯	▯▯	▯▯▯	▯▯	▯▯
`vni`	`uint32`	▯	-	VXLAN Network Identifier▯	`1016`
`routeTargets`	`string array`	▯	-	▯▯▯▯▯▯▯▯ Route Target ▯▯▯	`65000:1016`

VpcEgressGateway ▯▯▯▯

▯▯▯▯▯ VpcEgressGateway `.spec` ▯▯▯▯ BGP/EVPN ▯▯▯▯▯▯▯▯ VpcEgressGateway ▯▯▯▯▯▯▯ Egress Gateway ▯▯▯▯▯▯▯▯▯▯▯▯

▯▯	▯▯	▯▯	▯▯▯	▯▯	▯▯
`bgpConf`	`string`	▯	-	▯▯▯ BgpConf ▯▯▯ ▯▯▯▯▯▯▯ Egress Gateway ▯▯▯ BGP▯	`bgp-conf-6502`
`evpnConf`	`string`	▯	-	▯▯▯ EvpnConf ▯▯▯ ▯▯▯▯▯▯▯▯ `bgpConf` ▯▯▯▯▯▯ Egress Gateway ▯ ▯▯ EVPN/VXLAN▯	`evpn-conf-1016`

PDF Slack Support

2025 Kube-OVN Team

2026□4□8□

2026□4□8□

GitHub 

5.4.5 □□

5.5 VPC NAT □□□□ Service

VPC NAT 网关可以直接使用 nftables 为 Kubernetes Service 提供 DNAT。控制器监听 Service 和 EndpointSlice，在网关上编程后端、VIP、回程路由和 hairpin SNAT。`LoadBalancer` Service 可通过 EIP 访问，VPC 内流量可通过 Service 的 ClusterIP 访问。

该功能默认关闭。

5.5.1 开启功能

控制器提供两个相互独立的开关：

参数	作用	默认值
`--enable-gw-nftable-lb-svc`	通过 EIP 为 `LoadBalancer` Service 提供入口地址	`false`
`--enable-gw-nftable-svc-cluster-ip`	在网关上持有并服务 Service ClusterIP	`false`

安装脚本变量为 `ENABLE_GW_NFTABLE_LB_SVC` 和 `ENABLE_GW_NFTABLE_SVC_CLUSTER_IP`；Helm v1/v2 分别使用 `func.ENABLE_GW_NFTABLE_LB_SVC` / `features.enableGwNftableLoadbalancerService` 与 `func.ENABLE_GW_NFTABLE_SVC_CLUSTER_IP` / `features.enableGwNftableServiceClusterIp`。

- `--enable-gw-nftable-svc-cluster-ip` 与 `--enable-lb` 互斥，两者不能同时持有 ClusterIP 数据面。
- `--enable-gw-nftable-lb-svc` 与 `--enable-lb-svc` 互斥，两者不能同时持有 `LoadBalancer` Service 入口地址。
- 两个网关开关可以同时开启；也可以让网关处理 EIP、让 OVN LoadBalancer 处理 ClusterIP。
- `--enable-lb-svc` 仍需要 `--enable-lb` 或 ClusterIP 网关开关提供后端数据面。

旧参数 `--enable-nftable-lb-svc` 的默认值为 `true`，现在已拆分为两个默认关闭的参数，且没有兼容别名；升级前必须同步更新 controller 参数、安装脚本变量或 Helm values。

5.5.2 配置 Service

`LoadBalancer` Service 必须指定网关和 EIP；ClusterIP Service 只需指定网关：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web
  annotations:
    ovn.kubernetes.io/vpc_nat_gw: gw1
    ovn.kubernetes.io/eip: eip1 # LoadBalancer Service
spec:
  type: LoadBalancer
  selector:
    app: web
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
```

控制器完成编程后会写入 `status.loadBalancer.ingress`。Admission 不要求 EIP 已存在或处于 Ready；控制器会在 EIP 可用后重新排队并完成编程。后端仅由 EndpointSlice 提供；所有 Ready 后端均参与负载均衡。运行中 Pod 的 Multus 网卡地址原地变化不会触发 EndpointSlice 更新。

`IptablesDnatRule` 的 `type: share` 对象只是控制器维护的记账记录，不能手工创建、修改或删除，也不会直接驱动数据面。可使用以下命令查看状态：

```
kubectl get iptables-dnat-rules -l ovn.kubernetes.io/nftable-lb-svc-name=web
kubectl -n kube-system exec <vpc-nat-gw-pod> -- nft list table ip kube-ovn
kubectl -n kube-system exec <vpc-nat-gw-pod> -- ip -4 addr show dev lo label lo:ko-vip
kubectl ko nbctl lr-policy-list <vpc>
```

5.5.3 约□和升级

- 同一 EIP、端口和协议只能绑定一个 Service；冲突会被 webhook 拒绝。
- Service 使用的 EIP 必须属于标注的网关；删除 Service 前不能删除仍在使用的 EIP。
- 网关和 EIP 绑定不可修改；需要更换时删除并重新创建 Service。清除网关和 EIP 两个 annotation 是退出该模式的方式，控制器会释放记录、VIP 路由、loopback 地址和 finalizer。

2025 Kube-OVN Team



- `ovn.kubernetes.io/vpc_nat_gw` 与 `ovn.kubernetes.io/service_external_ip_from_subnet` 不能同时指定，避免 Service 发布两个入口地址。
- 删除网关会释放其关联 Service 的记录、入口状态和 finalizer；正常情况下 webhook 会先阻止仍绑定 EIP 或 ClusterIP 的网关删除。
- 仅支持 IPv4 ˊTCP/UDP 和 `Cluster` 流量策略；不支持节点本地流量策略、拓扑感知路由或客户端源地址保留。
- `ClientIP` 会话亲和性支持整个 VIP 身份；多网卡后端必须在该 VPC 中存在可用网卡。

PDF Slack Support

2026年11月7日

2026年11月7日

GitHub

5.5.4 ⬚⬚

5.6 VPC QoS

Kube-OVN [illegible] QoSPolicy CRD [illegible] VPC [illegible]

5.6.1 [illegible]

QoSPolicy [illegible] `priority` [illegible] QoS [illegible]

[illegible]

[illegible]	[illegible]	[illegible]
EIP QoS	1	[illegible] EIP [illegible]
NATGW [illegible] QoS	2	[illegible] NATGW [illegible] IP [illegible]
NATGW net1 [illegible] QoS	3	[illegible]

[illegible]

- `shared=false` [illegible] QoS [illegible] `shared=true` [illegible]
- EIP [illegible] QoS [illegible] NATGW [illegible] QoS

5.6.2 EIP QoS

[illegible] EIP [illegible] 1Mbps[illegible] 1[illegible] `shared=false` [illegible] QoSPolicy [illegible] EIP [illegible] QoSPolicy [illegible] QoS [illegible]

QoSPolicy [illegible]

```yaml
apiVersion: kubeovn.io/v1
kind: QoSPolicy
metadata:
  name: qos-eip-example
spec:
  shared: false
  bindingType: EIP
  bandwidthLimitRules:
  - name: eip-ingress
    rateMax: "1" # Mbps
    burstMax: "1" # Mbps
    priority: 1
    direction: ingress
  - name: eip-egress
    rateMax: "1" # Mbps
    burstMax: "1" # Mbps
    priority: 1
    direction: egress
```

IptablesEIP [illegible]

```yaml
kind: IptablesEIP
apiVersion: kubeovn.io/v1
metadata:
  name: eip-1
spec:
  natGwDp: gw1
  qosPolicy: qos-eip-example
```

`.spec.qosPolicy` [illegible]

5.6.3 [illegible] QoS [illegible] EIP

[illegible] `label` [illegible] qos [illegible] eip[illegible]

```
# kubectl get eip  -l ovn.kubernetes.io/qos=qos-eip-example
NAME    IP            MAC               NAT   NATGWDP   READY
eip-1   172.18.11.24   00:00:00:34:41:0B   fip   gw1       true
```


2025 Kube-OVN Team

5.6.4 VPC NATGW net1 □□ QoS

□ VPC NATGW □ net1 □□□□□□□□□□□□□□ 10Mbps□□□□□ 3□□□ `shared=true` □□□□□ QoSPolicy □□□□□□□□□□□□□□□□□□□□□□□□ QoSPolicy □□□□

QoSPolicy □□□□□

```yaml
apiVersion: kubeovn.io/v1
kind: QoSPolicy
metadata:
  name: qos-natgw-example
spec:
  shared: true
  bindingType: NATGW
  bandwidthLimitRules:
  - name: net1-ingress
    interface: net1
    rateMax: "10" # Mbps
    burstMax: "10" # Mbps
    priority: 3
    direction: ingress
  - name: net1-egress
    interface: net1
    rateMax: "10" # Mbps
    burstMax: "10" # Mbps
    priority: 3
    direction: egress
```

VpcNatGateway □□□□□

```yaml
kind: VpcNatGateway
apiVersion: kubeovn.io/v1
metadata:
  name: gw1
spec:
  vpc: test-vpc-1
  subnet: net1
  lanIp: 10.0.1.254
  qosPolicy: qos-natgw-example
  selector:
    - "kubernetes.io/hostname: kube-ovn-worker"
    - "kubernetes.io/os: linux"
```

`.spec.qosPolicy` □□□□□□□□□□□□□□□□□□□

5.6.5 net1 □□□□□□ QoS

□ net1 □□□□□□□□□□□□□□□□□ 5Mbps□□□□□ 2□□□ `shared=true` □□□□□ QoSPolicy □□□□□□□□□□□□□□□□□□□□□□ QoSPolicy □□□□

QoSPolicy □□□□□

```yaml
apiVersion: kubeovn.io/v1
kind: QoSPolicy
metadata:
  name: qos-natgw-example
spec:
  shared: true
  bindingType: NATGW
  bandwidthLimitRules:
  - name: net1-extip-ingress
    interface: net1
    rateMax: "5" # Mbps
    burstMax: "5" # Mbps
    priority: 2
    direction: ingress
    matchType: ip
    matchValue: src 172.18.11.22/32
  - name: net1-extip-egress
    interface: net1
    rateMax: "5" # Mbps
    burstMax: "5" # Mbps
    priority: 2
    direction: egress
    matchType: ip
    matchValue: dst 172.18.11.23/32
```

VpcNatGateway □□□□□

```yaml
kind: VpcNatGateway
apiVersion: kubeovn.io/v1
metadata:
  name: gw1
spec:
```

```
  vpc: test-vpc-1
  subnet: net1
  lanIp: 10.0.1.254
  qosPolicy: qos-natgw-example
  selector:
    - "kubernetes.io/hostname: kube-ovn-worker"
    - "kubernetes.io/os: linux"
```

5.6.6 □□□□□ QoS □ NATGW

□□ `label` □□□□□□□□ qos □ eip□

```
# kubectl get vpc-nat-gw  -l ovn.kubernetes.io/qos=qos-natgw-example
NAME   VPC          SUBNET   LANIP
gw1    test-vpc-1   net1     10.0.1.254
```

5.6.7 □□ qos □□

```
# kubectl get qos -A
NAME               SHARED   BINDINGTYPE
qos-eip-example    false    EIP
qos-natgw-example  true     NATGW
```

5.6.8 □□

- □□□□□□□□□□□ QoS □□□□□□□□□ QoS □□□□□□□□□□□□□ QoS □ EIP □ NATGW□□□□□□□ `spec.qosPolicy` □□□

PDF Slack Support

2026□1□29□

2023□5□9□

GitHub

5.6.9 □□

5.7 自定义 VPC 内部负载均衡

Kubernetes 提供的 Service 可以用作集群内的负载均衡，但是在自定义 VPC 模式下，使用 Service 作为内部负载均衡存在如下几个问题：

1. Service IP 范围为集群资源，所有自定义 VPC 共享，无法重叠。
2. 用户无法按照自己意愿设置内部负载均衡的 IP 地址。

为了解决上述问题，Kube-OVN 在 1.11 引入 `SwitchLBRule` CRD，用户可以设置自定义 VPC 内的内部负载均衡规则。

`SwitchLBRule` 支持以下两种方式设置自定义 VPC 内的内部负载均衡规则。

5.7.1 `Selector` 动生成负载均衡规则

通过 `selector` 可以通过 `label` 自动关联 `pod` 配置生成负载均衡规则。

`SwitchLBRule` 样例如下：

```
apiVersion: kubeovn.io/v1
kind: SwitchLBRule
metadata:
  name:  cjh-slr-nginx
spec:
  vip: 1.1.1.1
  sessionAffinity: ClientIP
  namespace: default
  selector:
    - app: nginx
  ports:
  - name: dns
    port: 8888
    targetPort: 80
    protocol: TCP
```

- `selector`，`sessionAffinity` 和 `port` 使用方式同 Kubernetes Service。
- `vip`：自定义负载均衡的 IP 地址。
- `namespace`：`selector` 所选择 Pod 所在命名空间。

Kube-OVN 会根据 `SwitchLBRule` 定义选择的 Pod 得出 Pod 所在 VPC 并设置对应的 L2 LB。

5.7.2 `Endpoints` 自定义负载均衡规则

通过 `endpoints` 可以自定义负载均衡规则，用以支持无法通过 `selector` 自动生成负载均衡规则的场景，比如负载均衡后端是 `kubevirt` 创建的 `vm` 。

`SwitchLBRule` 样例如下：

```
apiVersion: kubeovn.io/v1
kind: SwitchLBRule
metadata:
  name:  cjh-slr-nginx
spec:
  vip: 1.1.1.1
  sessionAffinity: ClientIP
  namespace: default
  endpoints:
    - 192.168.0.101
    - 192.168.0.102
    - 192.168.0.103
  ports:
  - name: dns
    port: 8888
    targetPort: 80
    protocol: TCP
```

- `sessionAffinity` 和 `port` 使用方式同 Kubernetes Service。
- `vip`：自定义负载均衡的 IP 地址。
- `namespace`：负载均衡后端所在命名空间。
- `endpoints`：负载均衡后端 IP 列表。

2025 Kube-OVN Team

□□□□□□□ selector □ endpoints,□□□□□ selector □□□

5.7.3 □□□□

OVN □□ IPv4 □□□□□□□□□□□□□□□□□□ □□

[Health Checks](https://www.ovn.org/support/dist-docs/ovn-nb.5.html)

□□ ovn □□□□□□□□□□□□□□□ SwitchLBRule □□□□□□□□□□□ SwitchLBRule □□□□□□□□ VPC □ subnet □□□□□□□□□□ vip □□□□□□□□□□□□□□□ ip_port_mappings □ load_balancer_health_check □□□□□□□□□□□□□

- □□□□ vip □□□□□□□□ subnet □□□□□□□□□□□□□ subnet □□□□□□□□□□□□□□□□□□□□□□□□□□□ SwitchLBRule □□□□□□□□□□□□
- □□□□□□□□ Selector □□□□□□□□□□□□□

□□□□□□□□□

```
root@server:~# kubectl get po -o wide -n vulpecula
NAME                      READY   STATUS    RESTARTS   AGE     IP          NODE     NOMINATED NODE   READINESS GATES
nginx-78d9578975-f4qn4   1/1     Running   3          4d16h   10.16.0.4   worker   <none>           <none>
nginx-78d9578975-t8tm5   1/1     Running   3          4d16h   10.16.0.6   worker   <none>           <none>

#    slr
root@server:~# cat << END > slr.yaml
apiVersion: kubeovn.io/v1
kind: SwitchLBRule
metadata:
  name:  nginx
  namespace:  vulpecula
spec:
  vip: 1.1.1.1
  sessionAffinity: ClientIP
  namespace: default
  selector:
    - app: nginx
  ports:
  - name: dns
    port: 8888
    targetPort: 80
    protocol: TCP
END
root@server:~# kubectl apply -f slr.yaml
root@server:~# kubectl get slr
NAME              VIP       PORT(S)    SERVICE                       AGE
vulpecula-nginx   1.1.1.1   8888/TCP   default/slr-vulpecula-nginx   3d21h
```

□□□□□ subnet □□□ vip □□□□□□

```
#       vip

root@server:~# kubectl get vip
NAME          NS    V4IP        MAC                V6IP    PMAC   SUBNET        READY   TYPE
vulpecula-subnet    10.16.0.2   00:00:00:39:95:C1   <nil>          vulpecula-subnet   true
```

□□□□□□□□□□□□ Load_Balancer_Health_Check □ Service_Monitor □

```
root@server:~# kubectl ko nbctl list Load_Balancer
_uuid               : 3cbb6d43-44aa-4028-962f-30d2dba9f0b8
external_ids        : {}
health_check        : [5bee3f12-6b54-411c-9cc8-c9def8f67356]
ip_port_mappings    : {"10.16.0.4"="nginx-78d9578975-f4qn4.default:10.16.0.2", "10.16.0.6"="nginx-78d9578975-t8tm5.default:10.16.0.2"}
name                : cluster-tcp-session-loadbalancer
options             : {affinity_timeout="10800"}
protocol            : tcp
selection_fields    : [ip_src]
vips                : {"1.1.1.1:8888"="10.16.0.4:80,10.16.0.6:80"}

root@server:~# kubectl ko nbctl list Load_Balancer_Health_Check
_uuid               : 5bee3f12-6b54-411c-9cc8-c9def8f67356
external_ids        : {switch_lb_subnet=vulpecula-subnet}
options             : {failure_count="3", interval="5", success_count="3", timeout="20"}
vip                 : "1.1.1.1:8888"

root@server:~# kubectl ko sbctl list Service_Monitor
_uuid               : 1bddc541-cc49-44ea-9935-a4208f627a91
external_ids        : {}
ip                  : "10.16.0.4"
logical_port        : nginx-78d9578975-f4qn4.default
options             : {failure_count="3", interval="5", success_count="3", timeout="20"}
port                : 80
protocol            : tcp
```

```
src_ip              : "10.16.0.2"
src_mac             : "c6:d4:b8:08:54:e7"
status              : online

_uuid               : 84dd24c5-e1b4-4e97-9daa-13687ed59785
external_ids        : {}
ip                  : "10.16.0.6"
logical_port        : nginx-78d9578975-t8tm5.default
options             : {failure_count="3", interval="5", success_count="3", timeout="20"}
port                : 80
protocol            : tcp
src_ip              : "10.16.0.2"
src_mac             : "c6:d4:b8:08:54:e7"
status              : online
```

[illegible] `vip` [illegible]

```
root@server:~# kubectl exec -it -n vulpecula nginx-78d9578975-t8tm5 -- curl 1.1.1.1:8888
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

[illegible]

[illegible] `pod` [illegible]

```
kubectl delete po nginx-78d9578975-f4qn4
kubectl get po -o wide -n vulpecula
NAME                    READY   STATUS    RESTARTS   AGE     IP          NODE     NOMINATED NODE   READINESS GATES
nginx-78d9578975-lxmvh   1/1     Running   0          31s     10.16.0.8   worker   <none>           <none>
nginx-78d9578975-t8tm5   1/1     Running   3          4d16h   10.16.0.6   worker   <none>           <none>
```

[illegible] `Load_Balancer_Health_Check` [illegible] `Service_Monitor` [illegible]

```
root@server:~# kubectl ko nbctl list Load_Balancer
_uuid               : 3cbb6d43-44aa-4028-962f-30d2dba9f0b8
external_ids        : {}
health_check        : [5bee3f12-6b54-411c-9cc8-c9def8f67356]
ip_port_mappings    : {"10.16.0.4"="nginx-78d9578975-f4qn4.default:10.16.0.2", "10.16.0.6"="nginx-78d9578975-t8tm5.default:10.16.0.2",
"10.16.0.8"="nginx-78d9578975-lxmvh.default:10.16.0.2"}
name                : cluster-tcp-session-loadbalancer
options             : {affinity_timeout="10800"}
protocol            : tcp
selection_fields    : [ip_src]
vips                : {"1.1.1.1:8888"="10.16.0.6:80,10.16.0.8:80"}

root@server:~# kubectl ko nbctl list Load_Balancer_Health_Check
_uuid               : 5bee3f12-6b54-411c-9cc8-c9def8f67356
external_ids        : {switch_lb_subnet=vulpecula-subnet}
options             : {failure_count="3", interval="5", success_count="3", timeout="20"}
vip                 : "1.1.1.1:8888"

root@server:~# kubectl ko sbctl list Service_Monitor
_uuid               : 84dd24c5-e1b4-4e97-9daa-13687ed59785
external_ids        : {}
ip                  : "10.16.0.6"
logical_port        : nginx-78d9578975-t8tm5.default
options             : {failure_count="3", interval="5", success_count="3", timeout="20"}
port                : 80
protocol            : tcp
src_ip              : "10.16.0.2"
src_mac             : "c6:d4:b8:08:54:e7"
status              : online

_uuid               : 5917b7b7-a999-49f2-a42d-da81f1eeb28f
external_ids        : {}
ip                  : "10.16.0.8"
logical_port        : nginx-78d9578975-lxmvh.default
options             : {failure_count="3", interval="5", success_count="3", timeout="20"}
port                : 80
protocol            : tcp
src_ip              : "10.16.0.2"
src_mac             : "c6:d4:b8:08:54:e7"
status              : online
```

[illegible] `SwitchLBRule` [illegible] `Load_Balancer_Health_Check` [illegible] `Service_Monitor` [illegible] `vip` [illegible]

```
root@server:~# kubectl delete -f slr.yaml
switchlbrule.kubeovn.io "vulpecula-nginx" deleted
root@server:~# kubectl get vip
No resources found
```

```
root@server:~# kubectl ko sbctl list Service_Monitor
root@server:~#
root@server:~# kubectl ko nbctl list Load_Balancer_Health_Check
root@server:~#
```

5.7.4 ... VPC ...

> `RouterLBRule` ... OVN EIP ... OVN EIP FIP SNAT DNAT ... OVN ...

... `SwitchLBRule` ... `RouterLBRule` ... VPC ... OVN ... Load Balancer ... OVN EIP ...

`RouterLBRule` ...

```yaml
apiVersion: kubeovn.io/v1
kind: RouterLBRule
metadata:
  name: rlr-nginx
spec:
  ovnEip: eip-1
  vpc: vpc-1
  namespace: default
  selector:
    - app: nginx
  sessionAffinity: ClientIP
  ports:
  - name: http
    port: 80
    targetPort: 8080
    protocol: TCP
```

...

...	...
ovnEip	... OVN EIP ...
vpc	... VPC ...
namespace	... Pod ...
selector	... Pod ...
endpoints	... IP ... selector ... selector...
sessionAffinity	... Kubernetes Service
ports	... name / port / targetPort / protocol

...

```yaml
apiVersion: kubeovn.io/v1
kind: RouterLBRule
metadata:
  name: rlr-example
spec:
  ovnEip: eip-1
  vpc: vpc-1
  namespace: default
  endpoints:
    - 10.0.1.10
    - 10.0.1.11
    - 10.0.1.12
  ports:
  - name: web
    port: 443
    targetPort: 8443
    protocol: TCP
```

... `kubectl get rlr` ...

```
# kubectl get rlr
NAME           OVNEIP   VPC    PORT(S)     SERVICE                   AGE
rlr-example    eip-1    vpc-1  443/TCP     default/rlr-rlr-example   10s
```

PDF Slack Support

2025 Kube-OVN Team

2026⬜11⬜5⬜

2022⬜12⬜15⬜



GitHub 



5.7.5 ⬜⬜

5.8 ▯▯▯ VPC ▯▯ DNS

▯▯▯▯▯▯▯ VPC ▯▯▯ VPC ▯▯▯▯▯▯▯▯▯▯ VPC ▯▯▯▯▯▯▯▯▯▯▯▯ VPC ▯▯ coredns▯ ▯▯▯▯▯▯▯▯▯▯ VPC ▯▯▯ Kubernetes ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `vpc-dns` CRD ▯▯▯▯

▯ CRD ▯▯▯▯▯▯▯ coredns▯▯ Pod ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ VPC▯▯▯▯▯▯▯▯▯ VPC ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ VPC ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ VPC ▯▯▯▯▯▯▯▯▯▯▯▯

> **Note**
>
> ▯ DNS ▯▯▯▯▯▯▯▯▯ Pod ▯ VM ▯▯▯▯▯▯▯▯▯▯▯ Webhook ▯▯▯▯▯▯▯▯▯▯▯ `/etc/resolv.conf` ▯▯▯▯

5.8.1 ▯▯ vpc-dns ▯▯▯▯▯▯▯

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:vpc-dns
rules:
  - apiGroups:
    - ""
    resources:
    - endpoints
    - services
    - pods
    - namespaces
    verbs:
    - list
    - watch
  - apiGroups:
    - discovery.k8s.io
    resources:
    - endpointslices
    verbs:
    - list
    - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: vpc-dns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:vpc-dns
subjects:
- kind: ServiceAccount
  name: vpc-dns
  namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vpc-dns
  namespace: kube-system
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: vpc-dns-corefile
  namespace: kube-system
data:
  Corefile: |
    .:53 {
        errors
        health {
          lameduck 5s
        }
        ready
        kubernetes cluster.local in-addr.arpa ip6.arpa {
          pods insecure
          fallthrough in-addr.arpa ip6.arpa
        }
        prometheus :9153
        forward . /etc/resolv.conf {
          prefer_udp
        }
        cache 30
```

```
        loop
        reload
        loadbalance
    }
```

▯▯▯▯▯▯▯▯▯▯▯▯▯▯ nat-gw-pod ▯▯▯▯▯▯▯▯▯▯

5.8.2 ▯▯▯▯▯▯▯

```yaml
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ovn-nad
  namespace: default
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "kube-ovn",
      "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
      "provider": "ovn-nad.default.ovn"
    }'
```

5.8.3 ▯▯ vpc-dns ▯ Configmap

▯ kube-system ▯▯▯▯▯▯▯ configmap▯▯▯ vpc-dns ▯▯▯▯▯▯▯▯▯▯▯▯ vpc-dns ▯▯▯

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: vpc-dns-config
  namespace: kube-system
data:
  coredns-vip: 10.96.0.3
  enable-vpc-dns: "true"
  nad-name: ovn-nad
  nad-provider: ovn-nad.default.ovn
```

- `enable-vpc-dns` ▯▯▯▯▯▯▯▯▯▯▯ `true` ▯
- `coredns-image` ▯dns ▯▯▯▯▯▯▯▯▯▯▯ coredns ▯▯▯▯▯
- `coredns-vip` ▯▯ coredns ▯▯ lb ▯▯▯ vip▯
- `nad-name` ▯▯▯▯ `network-attachment-definitions` ▯▯▯▯▯
- `nad-provider` ▯▯▯▯ provider ▯▯▯
- `k8s-service-host` ▯▯▯ coredns ▯▯ k8s apiserver ▯▯▯ ip▯▯▯▯▯▯▯ apiserver ▯▯▯
- `k8s-service-port` ▯▯▯ coredns ▯▯ k8s apiserver ▯▯▯ port▯▯▯▯▯▯▯ apiserver ▯▯▯

5.8.4 ▯▯ vpc-dns

▯▯ vpc-dns yaml▯

```yaml
kind: VpcDns
apiVersion: kubeovn.io/v1
metadata:
  name: test-cjh1
spec:
  vpc: cjh-vpc-1
  subnet: cjh-subnet-1
  replicas: 2
```

- `vpc` ▯ ▯▯▯▯ dns ▯▯▯ vpc ▯▯▯
- `subnet` ▯▯▯▯▯ dns ▯▯▯▯▯▯▯▯
- `replicas` : vpc dns deployment replicas

▯▯▯▯▯▯▯▯▯▯▯

```
# kubectl get vpc-dns
NAME        ACTIVE   VPC         SUBNET
test-cjh1   false    cjh-vpc-1   cjh-subnet-1
test-cjh2   true     cjh-vpc-1   cjh-subnet-2
```

`ACTIVE`：`true` ⬚⬚⬚⬚⬚⬚ dns ⬚⬚⬚ `false` ⬚⬚⬚⬚

⬚⬚⬚⬚⬚ VPC ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ DNS ⬚⬚⬚

- ⬚⬚⬚ VPC ⬚⬚⬚⬚⬚ vpc-dns ⬚⬚⬚⬚⬚⬚⬚ VPC ⬚⬚⬚ subnet⬚⬚⬚⬚⬚⬚ vpc-dns ⬚⬚⬚⬚ `true` ⬚⬚⬚⬚ `false` ⬚
- ⬚ `true` ⬚ vpc-dns ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `false` ⬚ vpc-dns ⬚⬚⬚⬚⬚

5.8.5 ⬚⬚⬚⬚⬚⬚⬚

⬚⬚ vpc-dns Pod ⬚⬚⬚⬚⬚ label `app=vpc-dns` ⬚⬚⬚⬚⬚⬚⬚⬚ vpc-dns Pod ⬚⬚⬚

```
# kubectl -n kube-system get pods -l app=vpc-dns
NAME                                READY   STATUS    RESTARTS   AGE
vpc-dns-test-cjh1-7b878d96b4-g5979   1/1     Running   0          28s
vpc-dns-test-cjh1-7b878d96b4-ltmf9   1/1     Running   0          28s
```

⬚⬚ slr ⬚⬚⬚⬚⬚

```
# kubectl -n kube-system get slr
NAME               VIP          PORT(S)                  SERVICE                              AGE
vpc-dns-test-cjh1   10.96.0.3   53/UDP,53/TCP,9153/TCP   kube-system/slr-vpc-dns-test-cjh1   113s
```

⬚⬚⬚ VPC ⬚⬚ Pod⬚⬚⬚ dns ⬚⬚:

```
nslookup kubernetes.default.svc.cluster.local 10.96.0.3
```

⬚ VPC ⬚⬚ switch lb rule ⬚⬚⬚⬚⬚⬚⬚⬚ VPC ⬚⬚⬚⬚⬚⬚⬚ Pod ⬚⬚⬚⬚⬚⬚

PDF Slack Support

2026⬚4⬚28⬚

2022⬚12⬚26⬚

GitHub



5.8.6 ⬚⬚

5.9 SecurityGroup [illegible]

Kube-OVN [illegible] Pod [illegible]

> ⚠ **Warning**
>
> Kube-OVN [illegible] NetworkPolicy[illegible]Network Policy API[illegible]Subnet ACL [illegible] Security Group [illegible] OVN ACL [illegible] NetworkPolicy [illegible] NetworkPolicy API [illegible]
> [illegible]

5.9.1 [illegible]

```yaml
apiVersion: kubeovn.io/v1
kind: SecurityGroup
metadata:
  name: sg-example
spec:
  allowSameGroupTraffic: true
  egressRules:
  - ipVersion: ipv4
    policy: allow
    priority: 1
    protocol: all
    remoteAddress: 10.16.0.13 # 10.16.0.0/16
    remoteType: address
  ingressRules:
  - ipVersion: ipv4
    policy: deny
    priority: 1
    protocol: icmp
    remoteAddress: 10.16.0.14
    remoteType: address
```

[illegible] Kube-OVN [illegible]

Pod [illegible] `ovn.kubernetes.io/security_groups` annotation [illegible]

```yaml
    ovn.kubernetes.io/security_groups: sg-example
```

[illegible]Port Security[illegible]

5.9.2 [illegible]

[illegible] `tier` [illegible] ACL [illegible] ACL [illegible]

- **`tier`** [illegible] `0` [illegible] `1` [illegible] `0` [illegible]tier `0` [illegible] tier `0` [illegible] `policy: pass` [illegible] ACL [illegible] tier `1` [illegible]
- **`policy: pass`** [illegible] `allow` [illegible] `deny` [illegible] tier [illegible] 1 [illegible] `pass` [illegible]

[illegible]

[illegible] tier-0 [illegible] tier-1 [illegible]

[illegible]

[illegible]

```yaml
apiVersion: kubeovn.io/v1
kind: SecurityGroup
metadata:
  name: sg-tier0
spec:
  tier: 0
  allowSameGroupTraffic: true
  ingressRules:
  - ipVersion: ipv4
    policy: pass
    priority: 1
    protocol: tcp
    remoteAddress: 10.16.0.0/16
```

```
    remoteType: address
  - ipVersion: ipv4
    policy: deny
    priority: 2
    protocol: all
    remoteAddress: 0.0.0.0/0
    remoteType: address
---
apiVersion: kubeovn.io/v1
kind: SecurityGroup
metadata:
  name: sg-tier1
spec:
  tier: 1
  allowSameGroupTraffic: true
  ingressRules:
  - ipVersion: ipv4
    policy: allow
    priority: 1
    protocol: tcp
    remoteAddress: 10.16.0.0/16
    remoteType: address
    portRangeMin: 80
    portRangeMax: 443
```

... `sg-tier0` ... `10.16.0.0/16` ... TCP ... tier 1... `sg-tier1` ... 80-443 ... TCP ...

... Pod... annotation ...

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: web
  annotations:
    ovn.kubernetes.io/security_groups: 'sg-tier0,sg-tier1'
  name: multi-tier-pod
  namespace: default
spec:
  nodeName: kube-ovn-worker
  containers:
  - image: docker.io/library/nginx:alpine
    imagePullPolicy: IfNotPresent
    name: nginx
```

5.9.3 ...

... `localAddress` ...

- **`localAddress`** ... IP ... CIDR...
- **`sourcePortRangeMin`** / **`sourcePortRangeMax`** ...1-65535... TCP ... UDP ...

...

```
apiVersion: kubeovn.io/v1
kind: SecurityGroup
metadata:
  name: sg-local-filter
spec:
  allowSameGroupTraffic: true
  ingressRules:
  - ipVersion: ipv4
    policy: allow
    priority: 1
    protocol: tcp
    remoteAddress: 10.16.0.0/16
    remoteType: address
    portRangeMin: 8080
    portRangeMax: 8080
    localAddress: 10.16.0.100
    sourcePortRangeMin: 1024
    sourcePortRangeMax: 65535
```

... `10.16.0.0/16` ... `10.16.0.100` ... 8080 ... TCP ... 1024-65535...

5.9.4 ...

- ... ACL ...OVN ... ACL ... ACL ...

2025 Kube-OVN Team

- ███████ `priority` ███████ 1-16384████████████████████ ACL ████████████████████ ACL ████████████████ `ACL = 18484 -` ████████████████ ACL ████

- `tier` ██████ `0` ▯ `1` ▯ `policy: pass` ████ tier `0` ██████ tier `1` ████████████

- ████████████████████████████████Kube-OVN ▯▯ CNI▯▯▯ Pod ████ Pod ████████████████████████████ Pod ████ ContainerCreating ██████████ Running ▯▯▯

5.9.5 ████

████ YAML ▯▯ Pod▯▯ annotation ████████████████

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: static
  annotations:
    ovn.kubernetes.io/security_groups: 'sg-example'
  name: sg-test-pod
  namespace: default
spec:
  nodeName: kube-ovn-worker
  containers:
  - image: docker.io/library/nginx:alpine
    imagePullPolicy: IfNotPresent
    name: qatest
```

████████████

```
# kubectl get pod -o wide
NAME                   READY   STATUS              RESTARTS   AGE     IP           NODE                  NOMINATED NODE   READINESS GATES
sg-test-pod            0/1     ContainerCreating   0          5h32m   <none>       kube-ovn-worker       <none>           <none>
test-99fff7f86-52h9r   1/1     Running             0          5h41m   10.16.0.14   kube-ovn-control-plane   <none>           <none>
test-99fff7f86-qcgjw   1/1     Running             0          5h43m   10.16.0.13   kube-ovn-worker       <none>           <none>
```

▯▯ kubectl describe pod ▯▯ Pod ████████████████

```
# kubectl describe pod sg-test-pod
Name:         sg-test-pod
Namespace:    default
Priority:     0
Node:         kube-ovn-worker/172.18.0.2
Start Time:   Tue, 28 Feb 2023 10:29:36 +0800
Labels:       app=static
Annotations:  ovn.kubernetes.io/allocated: true
              ovn.kubernetes.io/cidr: 10.16.0.0/16
              ovn.kubernetes.io/gateway: 10.16.0.1
              ovn.kubernetes.io/ip_address: 10.16.0.15
              ovn.kubernetes.io/logical_router: ovn-cluster
              ovn.kubernetes.io/logical_switch: ovn-default
              ovn.kubernetes.io/mac_address: 00:00:00:FA:17:97
              ovn.kubernetes.io/pod_nic_type: veth-pair
              ovn.kubernetes.io/port_security: true
              ovn.kubernetes.io/routed: true
              ovn.kubernetes.io/security_groups: sg-allow-reject
Status:       Pending
IP:
IPs:          <none>
.
.
.
Events:
  Type     Reason                  Age                   From     Message
  ----     ------                  ----                  ----     -------
  Warning  FailedCreatePodSandBox  5m3s (x70 over 4h59m)  kubelet  (combined from similar events): Failed to create pod sandbox: rpc error: code = Unknown
desc = failed to setup network for sandbox "40636e0c7f1ade5500fa958486163d74f2e2300051a71522a9afd7ba0538afb6": plugin type="kube-ovn" failed (add): RPC
failed; request ip return 500 configure nic failed 10.16.0.15 network not ready after 200 ping 10.16.0.1
```

████████████████████████████████

```yaml
apiVersion: kubeovn.io/v1
kind: SecurityGroup
metadata:
  name: sg-gw-both
spec:
  allowSameGroupTraffic: true
  egressRules:
  - ipVersion: ipv4
    policy: allow
    priority: 2
    protocol: all
    remoteAddress: 10.16.0.13
    remoteType: address
```


2025 Kube-OVN Team

```
  - ipVersion: ipv4
    policy: allow
    priority: 1
    protocol: all
    remoteAddress: 10.16.0.1
    remoteType: address
  ingressRules:
  - ipVersion: ipv4
    policy: deny
    priority: 2
    protocol: icmp
    remoteAddress: 10.16.0.14
    remoteType: address
  - ipVersion: ipv4
    policy: allow
    priority: 1
    protocol: icmp
    remoteAddress: 10.16.0.1
    remoteType: address
```

分别在入方向和出方向规则中，添加允许到网关的访问规则，并且设置该规则的优先级最高。

利用以下 yaml 绑定安全组，部署 Pod 后，确认 Pod 可以正常运行：

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: static
  annotations:
    ovn.kubernetes.io/security_groups: 'sg-gw-both'
  name: sg-gw-both
  namespace: default
spec:
  nodeName: kube-ovn-worker
  containers:
  - image: docker.io/library/nginx:alpine
    imagePullPolicy: IfNotPresent
    name: qatest
```

部署后查看 Pod 信息：

```
# kubectl get pod -o wide
NAME                READY   STATUS              RESTARTS   AGE     IP           NODE                NOMINATED NODE   READINESS GATES
sg-test-pod         0/1     ContainerCreating   0          5h41m   <none>       kube-ovn-worker     <none>           <none>
sg-gw-both          1/1     Running             0          5h37m   10.16.0.19   kube-ovn-worker     <none>           <none>
```

因此对于安全组的使用，要特别明确添加的限制规则的作用。如果单纯是限制流量访问，可以考虑使用网络策略实现。

PDF Slack Support

2026年3月17日

2023年2月28日

GitHub

5.10 OVN EIP FIP SNAT DNAT □□

> **Note**
>
> □□□ VPC □□□□□□□□□ VPC □□□□□□□□□□□□□□□□
>
> □□□□□ VPC □□□□□□□□□□□□□VPC NAT □□□OVN □□□ Egress Gateway□□□ VPC NAT □□□ Kube-OVN □□□□□□□□□□□□□ VPC NAT □□□□□□□□□ Pod□□□□□□□□□□ VPC □□□□□□□□□□ Macvlan □□□□□□□□□□□ Pod □□ iptables □□□
>
> OVN □□□□□ OVN □□□□□□□□□□ NAT □□□□□□□□□□□□□□□□□□□□□□□□ OVN □□□ BFD □□□□□□□□□□□□□□ OVN □□□□□□□□□□□□ OVN □□□□□□□□□
>
> Egress Gateway □□□ VPC NAT □□□

□□□□ VPC □ OVN NAT □□□□□□□□□□ `provider-network vlan (external) subnet` □□□□□□□□□□□ VPC EIP/SNAT□□□

5.10.1 □□□□□□□□□□□□

- □□□□□□□□□□□□□□□□□□□□□ `kube-ovn-controller` □ `kube-ovn-cni` □□□□□□□□□ □□□□ `ovn-external-gw-config` □□ `VPC spec enableExternal` □□□□□□□□□□□□□
- `CRD` □□□ `provider-network` `vlan` `subnet` □□□□□□□ `VPC spec extraExternalSubnets` □□□□□□□□□□□□□□ `ovn-eip` □ `ovn-dnat` □ `ovn-fip` □ `ovn-snat` □ CRD □□□□

```
graph LR

pod-->subnet-->vpc-->lrp--bind-->gw-chassis-->snat-->lsp-->external-subnet
lrp-.-peer-.-lsp
```

Pod □□ SNAT □□□□□□□□□□□□□□□□□□□□□□□□□□□ Pod □□ Fip □□□□□□□□□□□□□□□

```
graph LR

pod-->subnet-->vpc-->lrp--bind-->local-chassis-->snat-->lsp-->external-subnet

lrp-.-peer-.-lsp
```

Pod □□□□□□□□ FIP (dnat_and_snat) □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□ CRD □□□□□□ iptables nat gw □□□□□□□□□□□□

- ovn eip: □□□□ ip □□□□ underlay provider network vlan subnet □□□
- ovn fip□ □□□ dnat snat□□ VPC □□ ip □□ vip □□□□□□□□□□□
- ovn snat□□□□□□□□□□ VPC □ ip □□□□ snat □□□□
- ovn dnat□□□ router lb □□, □□□□ ip + □□ □□□□ VPC □□ □□ endpoints

5.10.2 1. □□

□□□□□□ □□□□□

□□ OpenStack Neutron ovn□□□□□□□□□□□□ provider network □□□□□□□□□□□□□□□□□□□ VPC EIP/SNAT □□□□

□□□□□□□□□ vlan□□□ vlan 0□□□□□□□□□ vlan id□

```
#
# 1. kube-ovn-controller
        - --external-gateway-vlanid=204
        - --external-gateway-switch=external204

# 2. kube-ovn-cni        :
        - --external-gateway-switch=external204
```

```
###               vlan id                  underlay
```

□□□□□□□□□□□□□□□□□□□□□□□□

- □□□□□□□□□□□□ provider network□vlan□subnet □□□□
- □□□□□□□□□□□□ VPC enable_eip_snat □□□□□□□□□ vlan□subnet □□□□□□□□□□ ip □ ipam□
- □□□□□□□□ VPC □ enable_eip_snat □□, □□□□□□□□□ pod annotation □ fip snat□□□□□□□□□□□□□□
- □□□□□□□□□□□□□ VPC □ enable_eip_snat □□□□□□□□□□ vlan□subnet □□□□□□□□□□ VPC □□ eip snat □□□□

1.1 □□ **underlay** □□□□

```
#    provider-network  vlan  subnet
# cat 01-provider-network.yaml

apiVersion: kubeovn.io/v1
kind: ProviderNetwork
metadata:
  name: external204
spec:
  defaultInterface: vlan

# cat 02-vlan.yaml

apiVersion: kubeovn.io/v1
kind: Vlan
metadata:
  name: vlan204
spec:
  id: 204
  provider: external204

# cat 03-vlan-subnet.yaml

apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: external204
spec:
  protocol: IPv4
  cidrBlock: 10.5.204.0/24
  gateway: 10.5.204.254
  vlan: vlan204
  excludeIps:
  - 10.5.204.1..10.5.204.100
```

1.2 □□ **VPC** □□ **eip_snat**

```
#      VPC     underlay    provider subnet
# cat 00-centralized-external-gw-no-ip.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  name: ovn-external-gw-config
  namespace: kube-system
data:
  enable-external-gw: "true"
  external-gw-nodes: "pc-node-1,pc-node-2,pc-node-3"
  type: "centralized"
  external-gw-nic: "vlan" #      ovs
  external-gw-addr: "10.5.204.254/24" # underlay       ip
```

□□□□□□□□□□□□□□□ logical router port (lrp) ip □ mac□□□□□□ underlay □□□□□□□□□□□ lrp □□□□ ovn eip □□□□

□□□□□□□□□□□□□□□ ip □□□□□□□□□ lrp □□□□ ovn-eip□ □□□□□□□□□□□□□ lrp □□□□ ovn eip□

1.3 □□□□ **VPC** □□ **eip snat fip** □□

```
# cat 00-ns.yml

apiVersion: v1
kind: Namespace
metadata:
  name: vpc1

# cat 01-vpc-external.yml
```

```
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: vpc1
spec:
  namespaces:
  - vpc1
  enableExternal: true
  staticRoutes:
    - cidr: 0.0.0.0/0
      nextHopIP: 10.5.204.254
      policy: policyDst

# VPC    enableExternal       lrp

# cat 02-subnet.yml

apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: vpc1-subnet1
spec:
  cidrBlock: 192.168.0.0/24
  default: false
  disableGatewayCheck: false
  disableInterConnection: true
  gatewayNode: ""
  gatewayType: distributed
  #gatewayType: centralized
  natOutgoing: false
  private: false
  protocol: IPv4
  provider: ovn
  vpc: vpc1
  namespaces:
  - vpc1

#                    subnet
```

```
# kubectl ko nbctl show vpc1

router 87ad06fd-71d5-4ff8-a1f0-54fa3bba1a7f (vpc1)
    port vpc1-vpc1-subnet1
        mac: "00:00:00:ED:8E:C7"
        networks: ["192.168.0.1/24"]
    port vpc1-external204
        mac: "00:00:00:EF:05:C7"
        networks: ["10.5.204.105/24"]
        gateway chassis: [7cedd14f-265b-42e5-ac17-e03e7a1f2342 276baccb-fe9c-4476-b41d-05872a94976d fd9f140c-c45d-43db-a6c0-0d4f8ea298dd]
    nat 21d853b0-f7b4-40bd-9a53-31d2e2745739
        external ip: "10.5.204.115"
        logical ip: "192.168.0.0/24"
        type: "snat"
```

```
# kubectl ko nbctl lr-route-list vpc1

IPv4 Routes
Route Table <main>:
                0.0.0.0/0              10.5.204.254 dst-ip
#    VPC CRD
```

> enableExternal VPC CRD

1.4

1.4.1 UNDERLAY

eip snat fip 1 eip snat fip

```
#    provider-network  vlan  subnet
# cat 01-extra-provider-network.yaml
apiVersion: kubeovn.io/v1
kind: ProviderNetwork
metadata:
  name: extra
spec:
  defaultInterface: vlan
# cat 02-extra-vlan.yaml
apiVersion: kubeovn.io/v1
kind: Vlan
metadata:
  name: vlan0
spec:
  id: 0
  provider: extra
```

```
# cat 03-extra-vlan-subnet.yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: extra
spec:
  protocol: IPv4
  cidrBlock: 10.10.204.0/24
  gateway: 10.10.204.254
  vlan: vlan0
  excludeIps:
  - 10.10.204.1..10.10.204.100
```

1.4.2 [illegible] VPC [illegible]

```
apiVersion: kubeovn.io/v1
kind: Vpc
metadata:
  name: vpc1
spec:
  namespaces:
  - vpc1
  enableExternal: true  #    enableExternal   VPC               external   ls
  extraExternalSubnets: #    extraExternalSubnets
  - extra
```

以上模板应用后，应该可以看到如下资源存在

```
# kubectl ko nbctl show vpc1
router 87ad06fd-71d5-4ff8-a1f0-54fa3bba1a7f (vpc1)
    port vpc1-vpc1-subnet1
        mac: "00:00:00:ED:8E:C7"
        networks: ["192.168.0.1/24"]
    port vpc1-external204
        mac: "00:00:00:EF:05:C7"
        networks: ["10.5.204.105/24"]
        gateway chassis: [7cedd14f-265b-42e5-ac17-e03e7a1f2342 276baccb-fe9c-4476-b41d-05872a94976d fd9f140c-c45d-43db-a6c0-0d4f8ea298dd]
    port vpc1-extra
        mac: "00:00:00:EF:6A:C7"
        networks: ["10.10.204.105/24"]
        gateway chassis: [7cedd14f-265b-42e5-ac17-e03e7a1f2342 276baccb-fe9c-4476-b41d-05872a94976d fd9f140c-c45d-43db-a6c0-0d4f8ea298dd]
```

5.10.3 2. ovn-eip

该功能和 iptables-eip 设计和使用方式基本一致，ovn-eip 目前有三种 type

- nat: 是指 ovn dnat，fip, snat 这三种 nat 资源类型
- lrp: 软路由基于该端口和 underlay 公网互联，该 lrp 端口的 ip 可以被其他 dnat snat 复用
- lsp: 用于逻辑交换机端口（LSP）的 EIP

```
---
kind: OvnEip
apiVersion: kubeovn.io/v1
metadata:
  name: eip-static
spec:
  externalSubnet: external204
  type: nat

#       eip           fip
```

externalSubnet 字段可不进行配置，若未配置则会使用默认公网网络，在上述配置中默认公网网络为 external204。

若要使用额外公网网络，则需要通过 externalSubnet 显式指定需要扩展使用的公网网络，在上述配置中扩展公网网络为 extra。

2.1 ovn-fip 为 pod 绑定一个 fip

```
# kubectl get po -o wide -n vpc1 vpc-1-busybox01
NAME              READY   STATUS    RESTARTS   AGE     IP            NODE
vpc-1-busybox01   1/1     Running   0          3d15h   192.168.0.2   pc-node-2

# kubectl get ip vpc-1-busybox01.vpc1
NAME                  V4IP          V6IP   MAC                NODE        SUBNET
vpc-1-busybox01.vpc1  192.168.0.2          00:00:00:0A:DD:27  pc-node-2   vpc1-subnet1

---

kind: OvnEip
apiVersion: kubeovn.io/v1
```

```
metadata:
  name: eip-static
spec:
  externalSubnet: external204
  type: nat

---
kind: OvnFip
apiVersion: kubeovn.io/v1
metadata:
  name: eip-static
spec:
  ovnEip: eip-static
  ipName: vpc-1-busybox01.vpc1  #       ip crd
  type: "centralized"           # centralized    distributed

--
#         VPC       ip

kind: OvnFip
apiVersion: kubeovn.io/v1
metadata:
  name: eip-static
spec:
  ovnEip: eip-static
  vpc: vpc1
  v4Ip: 192.168.0.2
  type: "centralized"           # centralized    distributed
```

```
# kubectl get ofip
NAME          VPC    V4EIP          V4IP          READY   IPTYPE   IPNAME
eip-for-vip   vpc1   10.5.204.106   192.168.0.3   true    vip      test-fip-vip
eip-static    vpc1   10.5.204.101   192.168.0.2   true             vpc-1-busybox01.vpc1
# kubectl get ofip eip-static
NAME          VPC    V4EIP          V4IP          READY   IPTYPE   IPNAME
eip-static    vpc1   10.5.204.101   192.168.0.2   true             vpc-1-busybox01.vpc1

[root@pc-node-1 03-cust-vpc]# ping 10.5.204.101
PING 10.5.204.101 (10.5.204.101) 56(84) bytes of data.
64 bytes from 10.5.204.101: icmp_seq=2 ttl=62 time=1.21 ms
64 bytes from 10.5.204.101: icmp_seq=3 ttl=62 time=0.624 ms
64 bytes from 10.5.204.101: icmp_seq=4 ttl=62 time=0.368 ms
^C
--- 10.5.204.101 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3049ms
rtt min/avg/max/mdev = 0.368/0.734/1.210/0.352 ms
[root@pc-node-1 03-cust-vpc]#

#       node ping    VPC    pod     ip
```

```
#     ip
# kubectl ko nbctl show vpc1
router 87ad06fd-71d5-4ff8-a1f0-54fa3bba1a7f (vpc1)
    port vpc1-vpc1-subnet1
        mac: "00:00:00:ED:8E:C7"
        networks: ["192.168.0.1/24"]
    port vpc1-external204
        mac: "00:00:00:EF:05:C7"
        networks: ["10.5.204.105/24"]
        gateway chassis: [7cedd14f-265b-42e5-ac17-e03e7a1f2342 276baccb-fe9c-4476-b41d-05872a94976d fd9f140c-c45d-43db-a6c0-0d4f8ea298dd]
    nat 813523e7-c68c-408f-bd8c-cba30cb2e4f4
        external ip: "10.5.204.101"
        logical ip: "192.168.0.2"
        type: "dnat_and_snat"
```

2.2 ovn-fip vip fip

 vip kubevirt vip keepalived kube-vip

 fip VPC vip vip

```
#     vip eip    eip     vip
# cat vip.yaml

apiVersion: kubeovn.io/v1
kind: Vip
metadata:
  name: test-fip-vip
spec:
  subnet: vpc1-subnet1

# cat 04-fip.yaml

---
kind: OvnEip
apiVersion: kubeovn.io/v1
metadata:
  name: eip-for-vip
```

```
spec:
  externalSubnet: external204
  type: nat

---
kind: OvnFip
apiVersion: kubeovn.io/v1
metadata:
  name: eip-for-vip
spec:
  ovnEip: eip-for-vip
  ipType: vip         #       fip     pod ip              vip
  ipName: test-fip-vip

---
#         VPC       ip

kind: OvnFip
apiVersion: kubeovn.io/v1
metadata:
  name: eip-for-vip
spec:
  ovnEip: eip-for-vip
  ipType: vip         #       fip     pod ip              vip
  vpc: vpc1
  v4Ip: 192.168.0.3
```

```
# kubectl get ofip
NAME          VPC    V4EIP          V4IP          READY   IPTYPE   IPNAME
eip-for-vip   vpc1   10.5.204.106   192.168.0.3   true    vip      test-fip-vip


[root@pc-node-1 fip-vip]# ping  10.5.204.106
PING 10.5.204.106 (10.5.204.106) 56(84) bytes of data.
64 bytes from 10.5.204.106: icmp_seq=1 ttl=62 time=0.694 ms
64 bytes from 10.5.204.106: icmp_seq=2 ttl=62 time=0.436 ms

#   node    ping


# pod     ip

[root@pc-node-1 fip-vip]# kubectl -n vpc1 exec -it vpc-1-busybox03 -- bash
[root@vpc-1-busybox03 /]#
[root@vpc-1-busybox03 /]#
[root@vpc-1-busybox03 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
1568: eth0@if1569: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 00:00:00:56:40:e5 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.0.5/24 brd 192.168.0.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 192.168.0.3/24 scope global secondary eth0  #      vip
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe56:40e5/64 scope link
       valid_lft forever preferred_lft forever

[root@vpc-1-busybox03 /]# tcpdump -i eth0 host  192.168.0.3 -netvv
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
00:00:00:ed:8e:c7 > 00:00:00:56:40:e5, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 62, id 44830, offset 0, flags [DF], proto ICMP (1), length 84)
    10.5.32.51 > 192.168.0.3: ICMP echo request, id 177, seq 1, length 64
00:00:00:56:40:e5 > 00:00:00:ed:8e:c7, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 43962, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.0.3 > 10.5.32.51: ICMP echo reply, id 177, seq 1, length 64

# pod        fip     icmp
```

5.10.4 3. ovn-snat

3.1 ovn-snat □□□□ subnet □ cidr

□□□□ iptables-snat □□□□□□□□□□□□

```
# cat 03-subnet-snat.yaml

---
kind: OvnEip
apiVersion: kubeovn.io/v1
metadata:
  name: snat-for-subnet-in-vpc
spec:
  externalSubnet: external204
  type: nat

---
kind: OvnSnatRule
```

```
apiVersion: kubeovn.io/v1
metadata:
  name: snat-for-subnet-in-vpc
spec:
  ovnEip: snat-for-subnet-in-vpc
  vpcSubnet: vpc1-subnet1 # eip

---
#         VPC       subnet cidr

kind: OvnSnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: snat-for-subnet-in-vpc
spec:
  ovnEip: snat-for-subnet-in-vpc
  vpc: vpc1
  v4IpCidr: 192.168.0.0/24 #        cidr      ip
```

□□□□□□□□□□□□□□□□□ externalSubnet □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ extra□

3.2 ovn-snat □□□□□ pod ip

□□□□ iptables-snat □□□□□□□□□□□□

```
# cat 03-pod-snat.yaml

---
kind: OvnEip
apiVersion: kubeovn.io/v1
metadata:
  name: snat-for-pod-vpc-ip
spec:
  externalSubnet: external204
  type: nat

---
kind: OvnSnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: snat01
spec:
  ovnEip: snat-for-pod-vpc-ip
  ipName: vpc-1-busybox02.vpc1 # eip      pod ip

---
#         VPC       ip

kind: OvnSnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: snat-for-subnet-in-vpc
spec:
  ovnEip: snat-for-subnet-in-vpc
  vpc: vpc1
  v4IpCidr: 192.168.0.4
```

□□□□□□□□□□□□□□□□□ externalSubnet □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ extra□

□□□□□□□□□□□□□ snat □□□□□□□□□□□□□

```
# kubectl ko nbctl show vpc1
router 87ad06fd-71d5-4ff8-a1f0-54fa3bba1a7f (vpc1)
    port vpc1-vpc1-subnet1
        mac: "00:00:00:ED:8E:C7"
        networks: ["192.168.0.1/24"]
    port vpc1-external204
        mac: "00:00:00:EF:05:C7"
        networks: ["10.5.204.105/24"]
        gateway chassis: [7cedd14f-265b-42e5-ac17-e03e7a1f2342 276baccb-fe9c-4476-b41d-05872a94976d fd9f140c-c45d-43db-a6c0-0d4f8ea298dd]
    nat 21d853b0-f7b4-40bd-9a53-31d2e2745739
        external ip: "10.5.204.115"
        logical ip: "192.168.0.0/24"
        type: "snat"
    nat da77a11f-c523-439c-b1d1-72c664196a0f
        external ip: "10.5.204.116"
        logical ip: "192.168.0.4"
        type: "snat"
```

```
[root@pc-node-1 03-cust-vpc]# kubectl get po -A -o wide  | grep busy
vpc1            vpc-1-busybox01                              1/1     Running   0               3d15h   192.168.0.2   pc-node-2   <none>           <none>
vpc1            vpc-1-busybox02                              1/1     Running   0               17h     192.168.0.4   pc-node-1   <none>           <none>
vpc1            vpc-1-busybox03                              1/1     Running   0               17h     192.168.0.5   pc-node-1   <none>           <none>
vpc1            vpc-1-busybox04                              1/1     Running   0               17h     192.168.0.6   pc-node-3   <none>           <none>
vpc1            vpc-1-busybox05                              1/1     Running   0               17h     192.168.0.7   pc-node-1   <none>           <none>
```

```
# kubectl exec -it -n vpc1            vpc-1-busybox04   bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
[root@vpc-1-busybox04 /]#
[root@vpc-1-busybox04 /]#
[root@vpc-1-busybox04 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
17095: eth0@if17096: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 00:00:00:76:94:55 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.0.6/24 brd 192.168.0.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe76:9455/64 scope link
       valid_lft forever preferred_lft forever
[root@vpc-1-busybox04 /]# ping 223.5.5.5
PING 223.5.5.5 (223.5.5.5) 56(84) bytes of data.
64 bytes from 223.5.5.5: icmp_seq=1 ttl=114 time=22.2 ms
64 bytes from 223.5.5.5: icmp_seq=2 ttl=114 time=21.8 ms

[root@pc-node-1 03-cust-vpc]# kubectl exec -it -n vpc1            vpc-1-busybox02   bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
[root@vpc-1-busybox02 /]#
[root@vpc-1-busybox02 /]#
[root@vpc-1-busybox02 /]#
[root@vpc-1-busybox02 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
1566: eth0@if1567: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 00:00:00:0b:e9:d0 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.0.4/24 brd 192.168.0.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe0b:e9d0/64 scope link
       valid_lft forever preferred_lft forever
[root@vpc-1-busybox02 /]# ping 223.5.5.5
PING 223.5.5.5 (223.5.5.5) 56(84) bytes of data.
64 bytes from 223.5.5.5: icmp_seq=2 ttl=114 time=22.7 ms
64 bytes from 223.5.5.5: icmp_seq=3 ttl=114 time=22.6 ms
64 bytes from 223.5.5.5: icmp_seq=4 ttl=114 time=22.1 ms
^C
--- 223.5.5.5 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3064ms
rtt min/avg/max/mdev = 22.126/22.518/22.741/0.278 ms

#       pod          snat
```

5.10.5 4. ovn-dnat

4.1 ovn-dnat pod dnat

```yaml
kind: OvnEip
apiVersion: kubeovn.io/v1
metadata:
  name: eip-dnat
spec:
  externalSubnet: underlay

---
kind: OvnDnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: eip-dnat
spec:
  ovnEip: eip-dnat
  ipName: vpc-1-busybox01.vpc1 #      pod ip crd
  protocol: tcp
  internalPort: "22"
  externalPort: "22"


---
#         VPC      ip

kind: OvnDnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: eip-dnat
spec:
  ovnEip: eip-dnat
  protocol: tcp
  internalPort: "22"
  externalPort: "22"
  vpc: vpc1
  v4Ip: 192.168.0.3
```

⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ externalSubnet ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ extra⊠

OvnDnatRule ⊠⊠⊠⊠ IptablesDnatRule ⊠⊠

```
# kubectl get oeip eip-dnat
NAME       V4IP         V6IP   MAC                TYPE   READY
eip-dnat   10.5.49.4           00:00:00:4D:CE:49  dnat   true

# kubectl get odnat
NAME                EIP                 PROTOCOL   V4EIP       V4IP          INTERNALPORT   EXTERNALPORT   IPNAME                          READY
eip-dnat            eip-dnat            tcp        10.5.49.4   192.168.0.3   22             22             vpc-1-busybox01.vpc1            true
```

4.2 ovn-dnat ⊠ vip ⊠⊠⊠⊠ dnat

```
kind: OvnDnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: eip-dnat
spec:
  ipType: vip  #       dnat     pod ip              vip
  ovnEip: eip-dnat
  ipName: test-dnat-vip
  protocol: tcp
  internalPort: "22"
  externalPort: "22"

---
#          VPC       ip

kind: OvnDnatRule
apiVersion: kubeovn.io/v1
metadata:
  name: eip-dnat
spec:
  ipType: vip  #       dnat     pod ip              vip
  ovnEip: eip-dnat
  ipName: test-dnat-vip
  protocol: tcp
  internalPort: "22"
  externalPort: "22"
  vpc: vpc1
  v4Ip: 192.168.0.4
```

OvnDnatRule ⊠⊠⊠⊠ IptablesDnatRule ⊠⊠

```
# kubectl get vip test-dnat-vip
NAME            V4IP          PV4IP   MAC                PMAC   V6IP   PV6IP   SUBNET         READY
test-dnat-vip   192.168.0.4           00:00:00:D0:C0:B5                        vpc1-subnet1   true

# kubectl get oeip eip-dnat
NAME       V4IP         V6IP   MAC                TYPE   READY
eip-dnat   10.5.49.4           00:00:00:4D:CE:49  dnat   true

# kubectl get odnat eip-dnat
NAME       EIP        PROTOCOL   V4EIP       V4IP          INTERNALPORT   EXTERNALPORT   IPNAME          READY
eip-dnat   eip-dnat   tcp        10.5.49.4   192.168.0.4   22             22             test-dnat-vip   true
```

PDF Slack Support

2026⊠10⊠20⊠

2023⊠3⊠3⊠

GitHub









5.11 VPC 互联

VPC 互联提供了一种将两个 VPC 网络通过逻辑路由打通的机制，从而使两个 VPC 内的工作负载可以像在同一个私有网络一样，通过私有地址相互访问，无需通过外部网关进行 NAT 转发。

5.11.1 使用限制

1. 该功能只适用于用户自定义 VPC。
2. 为了避免路由重叠两个 VPC 内的子网 CIDR 不能重叠。
3. 目前只支持两个 VPC 的互联，更多组 VPC 之间的互联暂不支持。
4. 每个 `vpcPeerings` 条目都必须设置 `localConnectIP`，填写一个或多个带掩码的端点 IP 地址和 CIDR，多个值用逗号分隔，例如 `169.254.0.1/30` 或 `169.254.0.1/30,fd00::1/64`。每个端点的 CIDR 应与远端对应端点匹配，且不应与已有子网冲突。

5.11.2 使用示例

首先创建两个不互联的 VPC，每个 VPC 下各有一个 Subnet，Subnet 的 CIDR 互不重叠。

```
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: vpc-1
spec: {}
---
kind: Subnet
apiVersion: kubeovn.io/v1
metadata:
  name: net1
spec:
  vpc: vpc-1
  cidrBlock: 10.0.0.0/16
---
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: vpc-2
spec: {}
---
kind: Subnet
apiVersion: kubeovn.io/v1
metadata:
  name: net2
spec:
  vpc: vpc-2
  cidrBlock: 172.31.0.0/16
```

在每个 VPC 内分别增加 `vpcPeerings` 和对应的静态路由：

```
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: vpc-1
spec:
  vpcPeerings:
    - remoteVpc: vpc-2
      localConnectIP: 169.254.0.1/30
  staticRoutes:
    - cidr: 172.31.0.0/16
      nextHopIP: 169.254.0.2
      policy: policyDst
---
kind: Vpc
apiVersion: kubeovn.io/v1
metadata:
  name: vpc-2
spec:
  vpcPeerings:
    - remoteVpc: vpc-1
      localConnectIP: 169.254.0.2/30
  staticRoutes:
    - cidr: 10.0.0.0/16
      nextHopIP: 169.254.0.1
      policy: policyDst
```

- `remoteVpc`：互联的另一个 VPC 的名字。
- `localConnectIP`：必填，作为互联端点的一个或多个 IP 地址和 CIDR，多个值用逗号分隔，例如 `169.254.0.1/30` 或 `169.254.0.1/30,fd00::1/64`。每个端点的 CIDR 应与远端对应端点匹配，且不应与已有子网冲突。

- `cidr`：另一端 Subnet 的 CIDR。
- `nextHopIP`：互联 VPC 另一端的 `localConnectIP`。

参数校验

创建或更新 VPC 时，admission webhook 会校验每个 peering 条目的 `localConnectIP`。留空时请求会被拒绝，并返回类似以下错误：

```
localConnectIP is required for vpc peering with vpc-2, e.g. 169.254.0.1/30
```

非空值还必须是有效的 CIDR，逗号分隔的每个值都会分别校验；校验失败时不会创建互联逻辑路由端口。如果 admission webhook 被禁用，VPC controller 在 reconcile 时会执行相同的必填和 CIDR 校验并停止本次处理，因此必须在每个条目中显式填写 `localConnectIP`。

分别在两个 Subnet 下创建 Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    ovn.kubernetes.io/logical_switch: net1
  name: vpc-1-pod
spec:
  containers:
    - name: vpc-1-pod
      image: docker.io/library/nginx:alpine
---
apiVersion: v1
kind: Pod
metadata:
  annotations:
    ovn.kubernetes.io/logical_switch: net2
  name: vpc-2-pod
spec:
  containers:
    - name: vpc-2-pod
      image: docker.io/library/nginx:alpine
```

测试网络连通性

```
# kubectl exec -it vpc-1-pod -- ping $(kubectl get pod vpc-2-pod -o jsonpath='{.status.podIP}')
PING 172.31.0.2 (172.31.0.2): 56 data bytes
64 bytes from 172.31.0.2: seq=0 ttl=62 time=0.655 ms
64 bytes from 172.31.0.2: seq=1 ttl=62 time=0.086 ms
64 bytes from 172.31.0.2: seq=2 ttl=62 time=0.098 ms
^C
--- 172.31.0.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.086/0.279/0.655 ms
# kubectl exec -it vpc-2-pod -- ping $(kubectl get pod vpc-1-pod -o jsonpath='{.status.podIP}')
PING 10.0.0.2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: seq=0 ttl=62 time=0.594 ms
64 bytes from 10.0.0.2: seq=1 ttl=62 time=0.093 ms
64 bytes from 10.0.0.2: seq=2 ttl=62 time=0.088 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.088/0.258/0.594 ms
```

PDF Slack Support

2026⊠10⊠28⊠

2022年5月24日

GitHub

6. 故障排查

6.1 kubectl-ko 插件

`kubectl-ko` 是用于操作和诊断 Kube-OVN 的 Go 编写的 kubectl 插件。它通过 Kubernetes API 获取资源，并将需要节点权限的操作转发给独立的 `kubectl-ko-node-agent` DaemonSet。节点 agent 不是 `kube-ovn-cni`、`ovs`、`ovs-ovn` 或 `ovn-central` 容器，也不依赖这些容器中的 shell、OVN/OVS 工具或 `tcpdump`。本地无需安装 Bash、OVN/OVS、tar 或额外调用 kubectl 子进程；仅以 `kubectl ko` 形式调用插件时需要本地 `kubectl`。

6.1.1 安装

Kube-OVN 镜像和 CNI 安装脚本会将插件安装到已有的 `/usr/local/bin/kubectl-ko` 路径。集群外的工作站可从匹配 Kube-OVN 版本的 GitHub Release 下载 `kubectl-ko-<os>-<arch>.tar.gz`（Linux/macOS）或 `kubectl-ko-windows-<arch>.zip`，同时下载 `kubectl-ko-checksums.txt` 校验 SHA256，再将插件放入 `PATH`。工作站必须使用对应操作系统和架构的文件；节点侧的 OVN/OVS 操作仍然只能在 Linux 节点执行。Linux、macOS、Windows 均提供 amd64 和 arm64 版本，不能把 Linux Pod 中的二进制复制到其他系统的工作站。

Windows 使用与架构匹配的 ZIP 文件，以 `Get-FileHash` 校验 SHA256，再用 `Expand-Archive` 解压。将 `kubectl-ko.exe` 放在与 `kubectl.exe` 一样可通过 `PATH` 发现的目录，或直接在 PowerShell 中执行：

```
Get-FileHash .\kubectl-ko-windows-arm64.zip -Algorithm SHA256
Expand-Archive .\kubectl-ko-windows-arm64.zip .\kubectl-ko-windows-arm64
.\kubectl-ko-windows-arm64\kubectl-ko.exe version
.\kubectl-ko-windows-arm64\kubectl-ko.exe --context staging exec nbctl -- show
```

x64 Windows 应改用 amd64 文件。二进制 stdout 重定向（例如 `kubectl ko capture pod/web -- -w -`）请使用 PowerShell 7.4 或更高版本，或 `cmd.exe`；旧版 PowerShell 可能破坏 pcap 字节流。Windows 工作站无需 WSL。

需要访问节点的命令依赖 `kubectl-ko-node-agent` DaemonSet。使用 Helm Chart 安装 Kube-OVN 后等待该 DaemonSet 就绪，再运行节点诊断。插件通过 Kubernetes `pods/exec` 启动 agent helper，agent 使用自身挂载的节点 socket 和网络命名空间完成操作，不会进入 Kube-OVN 组件容器。集群安装脚本会在 agent DaemonSet 就绪后从其 `agent` 容器复制插件，不从 CNI 或 OVS/OVN 组件 Pod 复制。两个 Helm Chart 都部署独立 agent，并通过 `global.registry.imagePullSecrets` 拉取私有镜像；旧版 Chart 的 `full`、`controlPlaneOnly` 和 `dataPlaneOnly` 模式都包含 agent。v2 Chart 使用全局 镜像及拉取策略，并按 `ovsOvn.ovnDirectory`、`logging.directory`、`cni.configDirectory` 挂载宿主机数据。agent 只调度到 Linux 节点。

从源码构建可运行 `make build-kubectl-ko`，生成 `dist/images/kubectl-ko` 和仅适用于 Linux 的 `dist/images/kubectl-ko-node-agent`。此模块使用 Go `replace`，发布版本宜使用 Release 二进制，不宜直接运行 `go install ...@version`。Windows 本机构建可用 `go build -o kubectl-ko.exe ./cmd/kubectl-ko`；交叉构建 可用 `GOOS=windows GOARCH=amd64 make build-kubectl-ko`，ARM 工作站将架构 改为 `arm64`。

插件版本可在不访问集群的情况下查看：

```
kubectl ko version
kubectl plugin list
```

`kubectl plugin list` 也可发现遮蔽新版插件的旧二进制副本。

6.1.2 命令结构与能力

Go CLI 的旧命令名和位置参数不再是别名，需按以下命令树迁移。

```
kubectl ko
  exec
    (POD | TYPE/NAME) [-c CONTAINER] [flags] -- COMMAND [args...]
    -l SELECTOR -- COMMAND [args...]
    nbctl | sbctl | ic-nbctl | ic-sbctl -- [TOOL_ARGS...]
    vsctl | ofctl | dpctl | appctl --node NODE -- [TOOL_ARGS...]
  db
    health
    nb
      status | backup | kick SERVER_ID | restore
    sb
      status | backup | kick SERVER_ID
  trace TARGET --target IP [OPTIONS]
  capture TARGET [capture flags] -- [TCPDUMP_ARGS...]
  network inspect TARGET [--output table|json]
  diagnose
    cluster | node NODE | subnet SUBNET
    connectivity --target PROTOCOL://IP:PORT [--target ...]
    environment
  logs [OPTIONS]
  restart
  perf run [OPTIONS] | recovery --yes
```

```
acl decode COOKIE | listen --node NODE
version | completion bash|zsh|fish|powershell
```

命令	作用与副作用
`exec POD -- COMMAND`、`exec deploy/NAME -- COMMAND`、`exec svc/NAME -- COMMAND`	按 Pod 名称或 workload selector 确定一个 Running Pod，通过 Kubernetes `pods/exec` 执行命令。支持 `po`、`deploy`、`svc`、`ds`、`sts`、`rs`、`rc` 和 `job` 等类型。Workload 选择具有确定顺序，不会一次执行到多个 Pod。
`exec -l SELECTOR -- COMMAND`	按 Kubernetes label selector 选择一个 Running Pod，不能与资源引用同时使用。
`exec nbctl`、`exec sbctl`、`exec ic-nbctl`、`exec ic-sbctl`	分别寻找 OVN 或 interconnection 数据库 leader，通过 agent 挂载的节点 socket 执行客户端命令；写入和多命令事务仍可用。
`exec vsctl/ofctl/dpctl/appctl --node NODE`	通过指定节点的 agent 执行 OVS 工具；远端命令可能改变运行状态。
`db nb/sb status`、`db health`	前者显示所选 leader 的 RAFT 集群和存储状态；后者检查所有 Running central 容器的 NB/SB 存储，不依赖健康 leader，存储异常会导致失败。
`db nb/sb backup`	转换并下载独立数据库，校验数据库名和 SHA256，写入 JSON 来源信息；不覆盖已有文件，清理远端临时备份。
`db nb/sb kick SERVER_ID`	移除过期 RAFT 成员；`--dry-run` 仅打印 leader Pod 和具体命令，无需 agent；实际执行经 leader 节点的 agent。
`db nb restore`	用明确指定的 bootstrap 节点上已有的 NB 数据库重建 central NB/SB 集群，不导入任意本地备份，也不支持仅恢复 SB。操作会停止 central、保留原始文件和 RAFT header、重建并验证存储，然后重启 OVS；详细条件见下文。
`trace`	解析位置参数目标后执行 OVN；`--engine=all` 时继续执行 OVS。使用 `--target` 指定目的地址；`--source-ip` / `--source-mac` 指定合成报文源字段，localnet 使用 `--node` 选择 OVS 节点。
`capture`	通过独立 node agent 抓取位置参数目标。LRP 使用 `--observe-at lsp/NAME` 指定邻接点，Pod 或 secondary interface 使用 `-i IFACE`。
`network inspect`	检查位置参数 Pod、Node、Service、工作负载、VM 或 VMI；`--output=json` 输出机器可读结果；旧 `--pod` 已移除。
`diagnose cluster`、`diagnose node NODE`	检查集群配置、组件 rollout 和 leader。默认创建唯一的临时 NodePort Service，通过独立临时 Pod 在原 pinger 网络命名空间主动探测；node 模式限制到指定节点。
`diagnose subnet SUBNET`	创建隔离的临时 DaemonSet 和 NodePort Service，从子网 Pod 检查相互间 TCP/UDP、节点 ICMP 和 NodePort 连通性，无需可选的 CNI 节点 TCP/UDP listener。
`diagnose connectivity`	在现有 pinger 网络命名空间的独立临时 Pod 中检查指定 TCP/UDP IP endpoint，不创建 NodePort Service 或子网 DaemonSet。
`diagnose environment`	通过 Kubernetes API 发现所有 Linux 节点，使用各节点独立 agent 检查环境，包括没有 CNI Pod 的节点。
`logs`	并行收集组件文件、容器日志与 Linux 节点状态，限制单项大小并在 manifest 中记录部分；`netstat.log` 保存原始 `/proc/net/{tcp,tcp6,udp,udp6,unix}` 表，不解析进程 PID/名称。
`restart`	按依赖顺序重启并等待 central、OVS、controller、CNI、pinger 和 monitor。
`perf run`	创建隔离的探测 Pod/Service，测量 Pod、主机、Service 和组播性能；临时修改 OVN LB 与组播成员并清理所属变更，不删除 central leader。
`perf recovery --yes`	依次删除 NB、SB 和 northd leader Pod 并测量恢复，不运行流量基准测试。
`acl decode COOKIE`、`acl listen --node NODE`	解码 sample cookie，或从指定节点流式接收并解码采样 cookie，直到取消为止。
`version`、`help`、`completion`	无需访问集群即可查看本地版本、帮助和 shell 补全。

6.1.3 当前 target 引用与诊断参数

当前 `trace`、`capture` 和 `network inspect` 都使用一个位置参数 `TARGET`。裸名称表示 `pod/NAME`；命名空间资源使用 `KIND/NAME`，并通过 `-n NAMESPACE` 指定命名空间。`KIND/NAMESPACE/NAME` 三段格式会被拒绝。Node、localnet、LSP 和 LRP 引用是集群级对象。

支持 Pod（`pod`、`po`、`pods`）、Node（`node`、`no`、`nodes`）、Service 和工作负载（`service` / `svc`、`deployment` / `deploy`、`daemonset` / `ds`、`statefulset` / `sts`、`replicaset` / `rs`、`job` 及其别名）、KubeVirt VM/VMI（`virtualmachine` / `vm`、`virtualmachineinstance` / `vmi`）、`localnet/SUBNET`、`lsp/NAME` 和 `lrp/NAME`。Service 和工作负载目标会解析一个运行中的后端 Pod。

`trace` 使用 `trace TARGET --target IP [ ]`。`--target` 接受裸 IPv4/IPv6 或 `addr/IP`，替代旧版 `--dst-ip`。`--engine all` 依次执行 OVN 和 OVS，`--engine ovn` 只执行 OVN。`--source-ip` 和 `--source-mac` 可覆盖合成报文的源字段；localnet 源必须同时提供这两个值。`--node` 用于选择 localnet 或 localnet LSP 的 OVS 节点。未绑定 chassis 的普通 LSP 支持 OVN-only trace；LRP 没有 Linux 接口，始终只执行 OVN trace，默认使用 router port 的 IP 和 MAC。

`capture` 使用 `capture TARGET [ ] -- [ ]`。LRP 没有 Linux 接口，必须使用 `--observe-at lsp/NAME` 指定相邻且已绑定的 LSP，实际观测点会输出到 stderr。对于 Pod、Service/工作负载和 VM/VMI，`-i IFACE` 在选中的 Pod 网络命名空间内解析，可选择 `net1` 等 secondary interface。支持 `-D`、`-i`、`-c`、`-s`、`-nn` 和 `-w -`。

`network inspect` 使用 `network inspect TARGET [--output table|json]`，支持 Pod、Node、Service、工作负载、VM 和 VMI。Service、工作负载及 VM/VMI 结果同时标识请求资源和选中的 Pod；Node 目标读取宿主机网络命名空间。旧版 `network inspect --pod` 形式已移除。

```
kubectl ko trace -n app svc/api --target addr/10.0.0.8 --protocol tcp --dst-port 443
kubectl ko trace localnet/underlay-a --target 10.0.0.8 --source-ip 10.0.0.2 --source-mac 02:00:00:00:00:02 --node worker-a
kubectl ko trace lsp/web.app --target 10.0.0.8 --engine ovn
kubectl ko trace lrp/router-underlay-a --target 10.0.0.8
kubectl ko capture -n app pod/web -- -i net1 -c 10
kubectl ko capture lrp/router-underlay-a --observe-at lsp/web.app -- -c 10
kubectl ko network inspect node/worker-a
kubectl ko network inspect -n app svc/api --output json
```

6.1.4 □□□□□□□□

标准 Kubernetes 连接参数可放在子命令前后，但须位于远端参数分隔符 `--` 之前。kubeconfig 加载器支持 `--kubeconfig`、`KUBECONFIG`、context、TLS、认证插件和 impersonation。Workload `exec` 的命名空间由 `-n/--namespace`、kubeconfig 命名空间、`default` 依次确定；`TYPE/NAME` 中的斜杠分隔资源类型和名称，不是命名空间。`capture`、`trace` 等接收 Pod、VM 或 VMI 名称的命令可使用 `namespace/name` 覆盖命名空间。Kube-OVN 部署命名空间另由 `--kube-ovn-namespace`、`KUBE_OVN_NS`、`kube-system` 依次确定。

范围	参数与默认值
全局	`--timeout=0` 限制整个调用，零值允许长时间流 `--discovery-timeout=10s` 限制目标选择。Kubernetes `--request-timeout` 不会截断已建立的 exec 流。
Kubernetes exec	接受 `POD` 或 `TYPE/NAME`、`-c/--container`、`-l/--selector`、`-i/--stdin`、`-t/--tty`、`-q/--quiet` 和 `--pod-running-timeout=1m`。命令必须位于 `--` 后。
节点工具与抓包	OVS 工具需要 `--node`；抓包使用一个位置参数 `TARGET`，远端抓包参数位于 `--` 后。
网络检查	`network inspect` 使用一个位置参数 `TARGET`；默认 `--output=table`，自动化可用 `--output=json`。
Trace	`trace` 使用一个位置参数 `TARGET`，必须指定 `--target IP`；默认 `--protocol=icmp`、`--engine=all`，可选 `--dst-mac`。TCP/UDP 需要 `--dst-port=1..65535`；IPv4 ARP 使用 `--arp-op=request` 或 `--arp-op=reply`。
数据库备份与恢复	备份可指定 `--output FILE`，否则生成唯一的数据库专属文件名。恢复必须指 `--source-node NODE`，并指定 `--yes` 或 `--dry-run`。
诊断	cluster/node/subnet 可用 `--read-only` 跳过主动探测和临时资源。配置/探测模式可[`--skip-kube-proxy`，默认取自 `WITHOUT_KUBE_PROXY=true`。subnet 的 `--tcp-port`、`--udp-port` 依次取自 `TCP_CONN_CHECK_PORT`、`UDP_CONN_CHECK_PORT`，再默认到 8100、8101；显式参数优先。connectivity 至少需要一个 `--target tcp://IP:PORT` 或 `udp://IP:PORT`，IPv6 地址须加方括号。
日志	`--component=all`，可选 `all`、`kube-ovn`、`ovn`、`ovs`、`linux`；`--output-dir=kubectl-ko-log`、`--concurrency=4`、`--item-timeout=30s`、单项 `--max-bytes=268435456`，可选 `--strict`。
性能	`perf run` 默认 `--image=docker.io/kubeovn/test:v1.13.0`、`--duration=5s`、`--bandwidth=1G`。时长按单项测量，须为 1 秒至 5 分钟的整数秒；私有网络可指定内部镜像。中断 leader 是独立[`perf recovery --yes` 操作。

临时 NodePort 探测会请求 `PreferDualStack`，只检查 Service 实际分配的地址族。单栈集群只用其现有地址族，双栈集群检查两种地址族；若节点缺少已分配地址族的地址，命令会报告错误。

```
kubectl ko exec -n app pod/web -- date
kubectl ko exec nbctl --context staging -- --format=json show
kubectl ko exec nbctl -- -- ls-add example -- lsp-add example example-port
kubectl ko trace -n app pod/web --target 10.0.0.8 --protocol tcp --dst-port 443
kubectl ko trace localnet/underlay-a --node worker-a --target 2001:db8::8
kubectl ko network inspect -n app pod/web --output json
kubectl ko diagnose subnet ovn-default --tcp-port 8100 --udp-port 8101
kubectl ko diagnose connectivity --target tcp://10.0.0.8:8100 --target 'udp://[2001:db8::8]:8101'
kubectl ko db nb backup --output northbound.backup
kubectl ko logs --component ovn --output-dir ./support --strict
kubectl ko perf run --image registry.example/test:v1.13.0 --duration 2s
```


2025 Kube-OVN Team

6.1.5 Workload exec

`exec` [illegible] Pod [illegible] Kubernetes workload [illegible]workload [illegible] selector [illegible] Running Pod[illegible] Pod[illegible] `-l/--selector` [illegible] Pod [illegible] `-n/--namespace` [illegible] kubeconfig [illegible] `default` [illegible] leader [illegible]

[illegible] Kubernetes [illegible] `-c/--container` [illegible] `-i/--stdin` [illegible] `-t/--tty` [illegible] `-q/--quiet` [illegible] `--pod-running-timeout` [illegible] `-n/--namespace` [illegible] `--` [illegible] `--help` [illegible] `--timeout` [illegible] `-n` [illegible] `-c` [illegible]

```
kubectl ko exec deploy/web -- date
kubectl ko exec -n app pod/web -- date
kubectl ko exec svc/web -c app -- sh -c 'echo ready'
kubectl ko exec -n app -l app=web -- curl http://localhost:8080/healthz
kubectl ko exec nbctl -- --format=json show
kubectl ko exec vsctl --node worker-a -- --timeout=5 show
```

`exec nbctl --help` [illegible] `exec nbctl -- --help` [illegible] `ovn-nbctl` [illegible] `--` [illegible] TTY [illegible]

6.1.6 Pod[illegible]VM [illegible] VMI [illegible]

`capture` [illegible] `TARGET` [illegible] `pod/NAME` [illegible] `vm/NAME` [illegible] `vmi/NAME` [illegible] `-D` [illegible] `-i` [illegible] `-c` [illegible] `-s` [illegible] `-nn` [illegible] `-w -` [illegible] `-w -` [illegible] pcap [illegible] stdout[illegible] stderr[illegible]

```
kubectl ko capture -n app pod/web -- -w - > capture.pcap
kubectl ko capture -n app vm/database -- -w - > vm.pcap
kubectl ko capture -n app vmi/database -- -c 10
```

[illegible] agent [illegible] Linux `AF_PACKET` socket [illegible] `setns(2)` [illegible] `tcpdump` [illegible] `nsenter` [illegible] shell[illegible] Ethernet [illegible] pcap [illegible]

KubeVirt [illegible]

- `vm/NAME` [illegible] VMI [illegible] VM [illegible] `vmi/NAME` [illegible] VM [illegible] VMI[illegible]
- [illegible] VMI controller UID[illegible] launcher Pod[illegible]
- [illegible] launcher[illegible] launcher[illegible] launcher[illegible]
- launcher [illegible] Running [illegible] launcher[illegible] VMI/Pod [illegible]
- [illegible]

[illegible] VMI[illegible] launcher Pod[illegible] `vm/NAME` [illegible] VM[illegible] `network inspect` [illegible] OVSDB [illegible] Pod [illegible] OVS [illegible] Pod UID [illegible] cgroup [illegible] Pod [illegible] RX/TX [illegible] Pod [illegible] JSON [illegible] 64 [illegible] `stats64` [illegible] `stats` [illegible]

6.1.7 [illegible]

connectivity [illegible] UDP [illegible] `health check` [illegible] echo [illegible] DNS [illegible] UDP [illegible] subnet [illegible] listener[illegible]

cluster/node/subnet [illegible] Internet[illegible] ICMP [illegible] `--external-address IP` [illegible]

```
kubectl ko diagnose cluster --external-address 1.1.1.1 --external-address 2606:4700:4700::1111
```

[illegible] Pod [illegible] IP [illegible] UID[illegible]cluster/node/connectivity [illegible] pinger [illegible]service account[illegible] Pod [illegible] pinger UID [illegible] Ready [illegible] Pod [illegible] host PID [illegible] `SYS_ADMIN` [illegible] agent [illegible] service-account token[illegible] subnet [illegible] DaemonSet [illegible] Pod [illegible] Pod [illegible]

`logs` [illegible] `kubectl-ko-log/<node>/{kube-ovn,ovn,openvswitch,linux}` [illegible] central [illegible] `ovn/central-<pod>` [illegible] `manifest.json` [illegible] `--strict` [illegible]Unix [illegible]Windows [illegible] ACL [illegible]Unix [illegible] Windows ACL[illegible] NTFS[illegible]

Linux [illegible] Kubernetes API [illegible] Linux [illegible] CNI Pod[illegible]agent [illegible]XFRM [illegible] `nokeys` [illegible]IPsec [illegible] CNI Pod UID [illegible] charon [illegible] agent [illegible] strongSwan [illegible] socket[illegible] `ipsec.conf` [illegible] CNI [illegible] Pod [illegible]/[illegible]IPsec [illegible] daemon [illegible] Linux [illegible]

`db nb restore` 是从指定节点上已有的 NB 数据库重建集群，不能导入任意本地备份，也不支持只恢复 SB。必须显式指定 `--source-node`，该节点须是 `NODE_IPS` 的第一个成员，即镜像启动脚本的 bootstrap 节点。仅支持使用 字面量 `NODE_IPS` 成员列表、central hostPath 可写的标准 central Deployment。若 OVS 未挂载数据库目录，命令会用临时 hostNetwork helper 挂载同一 hostPath；若已有 OVS 数据库挂载，其 hostPath 必须与 central 一致。helper 使用 central 镜像和 security context，按 UID 清理。

先运行 `--dry-run`，确认条件后再用 `--yes` 执行。helper 和数据库文件检查会在 停止 central 前完成；所有 central Pod 停止后才修改文件。原文件和 RAFT header 保留在每个成员节点的独立目录，本地私有 JSON 记录最近恢复阶段。错误会停止 流程，并报告记录路径、远端目录和原副本数；命令不会猜测部分完成的恢复是否可 自动回滚。保留这些文件供人工决定后续恢复或回退。远端备份位于 `/etc/ovn/.kubectl-ko-recovery-<id>`，避免 central 启动时的文件权限 glob 破坏目录遍历权限。

`perf run` 在 Service 测量前最多等待 30 秒，直到临时 OVN LB 同步至所有 chassis，确保 qperf 动态数据端口可用；同步失败时停止测量并清理所属 LB。`restart`、原始 OVN/OVS 命令、数据库恢复以及显式的 `perf recovery --yes` 都可能改变运行中的集群状态。

6.1.8 ⊠⊠ agent ⊠⊠⊠

每个节点操作都通过 Kubernetes `pods/exec` 在独立 agent helper 中执行。请求 使用版本化 gRPC 流，多个并发请求使用独立流，不会重启 agent。agent 只挂载 执行所需的节点 socket、网络命名空间和日志路径，不挂载 service-account token，也不会在 `kube-ovn-cni`、`ovs-ovn` 或 `ovn-central` 容器中执行命令。短期探测 Pod 可通过普通 Kubernetes 流执行自身进程。workload exec 优先使用 WebSocket，仅在受支持的握手失败情形回退 SPDY。节点 helper 请求完成后会关闭 stdin 并等待外层 exec 进程退出；远端失败退出码会传播，不会自动重放写入命令。取消 agent 请求时会终止其 Linux 进程组及子进程，并返回退出码 130；主动 脱离到其他 session 的进程不在该进程组内。无效参数在创建客户端前报错。

使用插件的账号需要读取目标资源；节点操作还需要访问 agent 的 `pods/exec`。数据库恢复、组件重启和性能恢复操作需要额外的 workload 修改权限。请按组织的 最小权限策略授予这些权限。资源发现需要 get/list，agent 调用需要 `pods/exec`，日志需要 pods/log，探测需要 create/delete，rollout 和恢复需要 workload patch/scale。`pods/exec` 权限也能执行写入命令，不能视为数据库只读权限。用户端和 Kube-OVN 版本应匹配；远端镜像中可用的工具决定实际能力。

6.1.9 ▯▯▯▯▯

Go CLI ▯▯▯▯▯▯▯▯▯ Bash ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯CI ▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯ `logs --component COMPONENT --output-dir DIR` ▯

▯▯▯	▯▯▯
`nbctl ARGS` ▯ `sbctl ARGS`	`exec nbctl -- ARGS` ▯ `exec sbctl -- ARGS`
`icnbctl ARGS` ▯ `icsbctl ARGS`	`exec ic-nbctl -- ARGS` ▯ `exec ic-sbctl -- ARGS`
`vsctl/ofctl/dpctl/appctl NODE ARGS`	`exec TOOL --node NODE -- ARGS`
`nb/sb status` ▯ `nb/sb backup` ▯ `nb/sb kick ID`	`db nb/sb status` ▯ `db nb/sb backup` ▯ `db nb/sb kick ID`
`nb dbstatus` ▯ `sb dbstatus`	`db health`
`nb restore`	▯▯▯ `db nb restore --source-node NODE --dry-run` ▯▯▯▯▯▯ `--yes` ▯
`trace POD IP [MAC] PROTOCOL [PORT/OP]`	`trace POD --target IP [--dst-mac MAC] --protocol PROTOCOL [--dst-port PORT/--arp-op OP]`
`trace node//NODE ...`	▯ localnet ▯▯▯ `trace localnet/SUBNET --node NODE --target IP ...` ▯Node ▯▯▯▯ trace ▯▯
`ovn-trace ...`	`trace --engine ovn ...`
`tcpdump POD ARGS`	`capture POD -- ARGS`
`diagnose [all]`	`diagnose cluster`
`diagnose node NODE` ▯ `diagnose subnet SUBNET`	▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
`diagnose IPPorts tcp-IP-PORT,udp-IP-PORT`	`diagnose connectivity --target tcp://IP:PORT --target udp://IP:PORT`
`env-check`	`diagnose environment`
`log COMPONENT --output DIR`	`logs --component COMPONENT --output-dir DIR`
`reload`	`restart`
`perf IMAGE --duration 5`	`perf run --image IMAGE --duration 5s`
▯▯▯▯ leader ▯ `perf --include-disruption`	▯▯▯▯ `perf recovery --yes` ▯
`acl-sample decode/listen ...`	`acl decode/listen ...`

▯▯▯▯▯▯▯▯ `tcpdump` ▯ `ncat` / `netcat` ▯ `net-tools` ▯▯▯ `kubectl ko capture` ▯▯▯▯▯ `kubectl ko diagnose connectivity` ▯▯ TCP/UDP endpoint▯▯▯ `network inspect` ▯ `trace` ▯ `logs` ▯ `perf` ▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ DEBUG ▯▯▯DEBUG ▯▯▯▯▯▯▯ `gdb` ▯ `valgrind` ▯▯▯▯▯▯▯

▯▯ ▯ `kubectl ko` ▯▯▯▯▯ Helm values ▯▯▯ ▯ Bash ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ E2E fixture ▯▯▯▯

PDF Slack Support

2026⽉11⽉7⽇

2022⽉5⽉24⽇

GitHub 

6.1.10 ⽹⽹

6.2 生产环境诊断工具

生产版 Kube-OVN 镜像包含网络 daemon 和轻量的 `kubectl-ko-node-agent`。节点诊断由独立 agent 执行，agent 只为当前 `kubectl ko` 操作获取所需的权限。

生产基础镜像和 DPDK 镜像不包含下表中的通用诊断工具；debug 镜像可以继续提供 `gdb`、`valgrind` 和抓包工具用于 CI 或交互式排障。

旧工具	生产环境替代方式	说明
`tcpdump` 和其他 sniffer	`kubectl ko capture`	agent 在目标网络命名空间中打开 `AF_PACKET` socket，可使用 `-w -` 输出 pcap，不启动 `nsenter` 或 tcpdump 进程；`-D` 列出接口；pcap 目前需要 Ethernet 接口。
Wireshark	在本地工作站读取 pcap	节点只负责输出 pcap 流，图形界面不需要部署到节点。
`netcat` / `ncat`	`kubectl ko diagnose connectivity`	使用 Go TCP/UDP 探测报告端点可达性。
`nmap` Geneve 探测	agent 原生 UDP 探测	`diagnose environment` 报告 `open`、`closed` 或 `unknown`；没有响应的 UDP 报文不能证明服务已就绪。
`strace`、`readelf`、`gdb`	`kubectl ko` trace、日志、网络检查和性能命令	生产镜像不依赖这些工具；debug 镜像可以继续提供它们。

`capture` 支持 `-D`、`-i`、`-c`、`-s`、`-nn` 和 `-w -`。对于 Pod、Service/工作负载以及 VM/VMI 目标，`-i IFACE` 在目标 Pod 的网络命名空间中解析接口，因此可以选择 Multus 等 secondary interface，例如 `net1`：

```
kubectl ko capture -n app pod/web -- -i net1 -c 10
```

该命令进入选中 Pod 的网络命名空间，从 `net1` 抓取 10 个报文。抓包输出可以使用 `-w -` 保存到工作站，再由本地工具解析：

```
kubectl ko capture -n app pod/web -- -i net1 -c 10 -w - > capture.pcap
```

agent 自己通过 `setns(2)` 进入已挂载的网络命名空间，使抓包与 `kube-ovn-cni`、`ovs-ovn` 和 `ovn-central` 容器相互独立。由于依赖 Linux 的 `AF_PACKET` 和 network namespace，节点侧 agent 仅支持 Linux；Windows 支持的是 kubectl 客户端。

PDF Slack Support

2026年11月7日

2026年11月7日

GitHub

6.2.1 [illegible]

6.3 ▒▒▒▒▒▒▒

▒▒▒▒▒▒▒ Kubernetes ▒▒▒▒▒▒▒▒▒▒▒▒ `ovs-ovn` ▒▒▒▒▒ `ovn-controller` ▒▒▒▒▒▒▒▒▒▒▒▒ `ovn-central` ▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒ Kubernetes ▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ `ovn-central` ▒▒▒▒▒▒▒▒▒▒▒▒ ovn-central ▒▒▒

6.3.1 ▒▒▒▒▒▒▒▒▒▒▒

```
 # kubectl drain kube-ovn-worker --ignore-daemonsets --force
 node/kube-ovn-worker cordoned
 WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-ovn-cni-zt74b, kube-system/kube-ovn-pinger-5rxfs, kube-system/kube-proxy-jpmnm, kube-system/ovs-
ovn-v2kll
 evicting pod kube-system/coredns-64897985d-qsgpt
 evicting pod local-path-storage/local-path-provisioner-5ddd94ff66-llss6
 evicting pod kube-system/kube-ovn-controller-8459db5ff4-94lxb
 pod/kube-ovn-controller-8459db5ff4-94lxb evicted
 pod/coredns-64897985d-qsgpt evicted
 pod/local-path-provisioner-5ddd94ff66-llss6 evicted
 node/kube-ovn-worker drained
```

6.3.2 ▒▒ kubelet ▒ docker

▒▒▒▒▒▒▒ `ovs-ovn` ▒▒▒▒▒▒▒▒ `ovn-central` ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

```
systemctl stop kubelet
systemctl stop docker
```

Containerd

> ▒▒ containerd ▒▒ CRI ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ Pod/▒▒▒▒▒▒▒▒▒▒▒ `ovs-ovn` ▒

▒▒ containerd ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ `ovs-ovn` ▒▒▒▒

1. ▒▒ kubelet

   ```
   systemctl stop kubelet
   ```

2. ▒▒ ovs-ovn Pod

   ```
   crictl rm -f $(crictl ps | grep openvswitch | awk '{print $1}')
   ```

3. ▒▒ containerd

   ```
   systemctl stop containerd
   ```

6.3.3 ▒▒ Node ▒▒▒▒▒▒▒

```
rm -rf /var/run/openvswitch
rm -rf /var/run/ovn
rm -rf /etc/origin/openvswitch/
rm -rf /etc/origin/ovn/
rm -rf /etc/cni/net.d/00-kube-ovn.conflist
rm -rf /etc/cni/net.d/01-kube-ovn.conflist
rm -rf /var/log/openvswitch
rm -rf /var/log/ovn
```

6.3.4 ▒▒ kubectl ▒▒▒▒

```
kubectl delete no kube-ovn-01
```


2025 Kube-OVN Team

6.3.5 □□□□□□□□□□ ovn-sb □□□

□□□□□□ kube-ovn-worker □□□□□□□□

```
# kubectl ko sbctl show
Chassis "b0564934-5a0d-4804-a4c0-476c93596a17"
  hostname: kube-ovn-worker
  Encap geneve
      ip: "172.18.0.2"
      options: {csum="true"}
  Port_Binding kube-ovn-pinger-5rxfs.kube-system
Chassis "6a29de7e-d731-4eaf-bacd-2f239ee52b28"
  hostname: kube-ovn-control-plane
  Encap geneve
      ip: "172.18.0.3"
      options: {csum="true"}
  Port_Binding coredns-64897985d-nbfln.kube-system
  Port_Binding node-kube-ovn-control-plane
  Port_Binding local-path-provisioner-5ddd94ff66-h4tn9.local-path-storage
  Port_Binding kube-ovn-pinger-hf2p6.kube-system
  Port_Binding coredns-64897985d-fhwlw.kube-system
```

6.3.6 □□□□□□ chassis □□□□□□□□□□□□

uuid □□□□□□□□□□ Chassis □□ id□

```
# kubectl ko sbctl chassis-del b0564934-5a0d-4804-a4c0-476c93596a17
# kubectl ko sbctl show
Chassis "6a29de7e-d731-4eaf-bacd-2f239ee52b28"
  hostname: kube-ovn-control-plane
  Encap geneve
      ip: "172.18.0.3"
      options: {csum="true"}
  Port_Binding coredns-64897985d-nbfln.kube-system
  Port_Binding node-kube-ovn-control-plane
  Port_Binding local-path-provisioner-5ddd94ff66-h4tn9.local-path-storage
  Port_Binding kube-ovn-pinger-hf2p6.kube-system
  Port_Binding coredns-64897985d-fhwlw.kube-system
```

PDF Slack Support

2026□7□3□

2022□6□3□

GitHub  

6.3.7 □□

6.4 ⬚⬚ ovn-central ⬚⬚

由于 `ovn-central` 内的 `ovn-nb` 和 `ovn-sb` 分别建立了类似 etcd 的 Raft 集群，因此更换 `ovn-central` 节点需要额外的操作，保证集群状态的正确和数据的一致。建议每次只对一个节点进行上下线处理，以避免集群进入不可用状态，影响集群整体网络。

6.4.1 ovn-central ⬚⬚⬚⬚

本文档针对如下的集群情况，以下线 `kube-ovn-control-plane2` 节点为例，介绍如何将其从 `ovn-central` 集群中移除。

```
# kubectl -n kube-system get pod -o wide | grep central
ovn-central-6bf58cbc97-2cdhg                     1/1     Running   0             21m   172.18.0.3   kube-ovn-control-plane    <none>           <none>
ovn-central-6bf58cbc97-crmfp                     1/1     Running   0             21m   172.18.0.5   kube-ovn-control-plane2   <none>           <none>
ovn-central-6bf58cbc97-lxmpl                     1/1     Running   0             21m   172.18.0.4   kube-ovn-control-plane3   <none>           <none>
```

下线 **ovn-nb** 集群内对应节点

首先查看节点在集群内的 ID，以便后续操作。

```
# kubectl ko nb status
1b9a
Name: OVN_Northbound
Cluster ID: 32ca (32ca07fb-739b-4257-b510-12fa18e7cce8)
Server ID: 1b9a (1b9a5d76-e69b-410c-8085-39943d0cd38c)
Address: tcp:[172.18.0.3]:6643
Status: cluster member
Role: leader
Term: 1
Leader: self
Vote: self

Last Election started 2135194 ms ago, reason: timeout
Last Election won: 2135188 ms ago
Election timer: 5000
Log: [135, 135]
Entries not yet committed: 0
Entries not yet applied: 0
Connections: <-d64b ->d64b <-4984 ->4984
Disconnections: 0
Servers:
    4984 (4984 at tcp:[172.18.0.4]:6643) next_index=135 match_index=134 last msg 1084 ms ago
    1b9a (1b9a at tcp:[172.18.0.3]:6643) (self) next_index=2 match_index=134
    d64b (d64b at tcp:[172.18.0.5]:6643) next_index=135 match_index=134 last msg 1084 ms ago
status: ok
```

`kube-ovn-control-plane2` 对应节点 IP 为 `172.18.0.5`，集群内对应的 ID 为 `d64b`。接下来从 ovn-nb 集群中踢出该节点：

```
# kubectl ko nb kick d64b
started removal
```

确认节点踢出成功：

```
# kubectl ko nb status
1b9a
Name: OVN_Northbound
Cluster ID: 32ca (32ca07fb-739b-4257-b510-12fa18e7cce8)
Server ID: 1b9a (1b9a5d76-e69b-410c-8085-39943d0cd38c)
Address: tcp:[172.18.0.3]:6643
Status: cluster member
Role: leader
Term: 1
Leader: self
Vote: self

Last Election started 2297649 ms ago, reason: timeout
Last Election won: 2297643 ms ago
Election timer: 5000
Log: [136, 136]
Entries not yet committed: 0
Entries not yet applied: 0
Connections: <-4984 ->4984
Disconnections: 2
Servers:
    4984 (4984 at tcp:[172.18.0.4]:6643) next_index=136 match_index=135 last msg 1270 ms ago
    1b9a (1b9a at tcp:[172.18.0.3]:6643) (self) next_index=2 match_index=135
status: ok
```

██ ovn-sb ████████

████████ ovn-sb ████████████████ ID██████████

```
kubectl ko sb status
3722
Name: OVN_Southbound
Cluster ID: d4bd (d4bd37a4-0400-499f-b4df-b4fd389780f0)
Server ID: 3722 (3722d5ae-2ced-4820-a6b2-8b744d11fb3e)
Address: tcp:[172.18.0.3]:6644
Status: cluster member
Role: leader
Term: 1
Leader: self
Vote: self

Last Election started 2395317 ms ago, reason: timeout
Last Election won: 2395316 ms ago
Election timer: 5000
Log: [130, 130]
Entries not yet committed: 0
Entries not yet applied: 0
Connections: <-e9f7 ->e9f7 <-6e84 ->6e84
Disconnections: 0
Servers:
    e9f7 (e9f7 at tcp:[172.18.0.5]:6644) next_index=130 match_index=129 last msg 1006 ms ago
    6e84 (6e84 at tcp:[172.18.0.4]:6644) next_index=130 match_index=129 last msg 1004 ms ago
    3722 (3722 at tcp:[172.18.0.3]:6644) (self) next_index=2 match_index=129
status: ok
```

`kube-ovn-control-plane2` ████ IP ██ `172.18.0.5` ████████ ID ██ `e9f7` ████ ovn-sb ██████████

```
# kubectl ko sb kick e9f7
started removal
```

██████████

```
# kubectl ko sb status
3722
Name: OVN_Southbound
Cluster ID: d4bd (d4bd37a4-0400-499f-b4df-b4fd389780f0)
Server ID: 3722 (3722d5ae-2ced-4820-a6b2-8b744d11fb3e)
Address: tcp:[172.18.0.3]:6644
Status: cluster member
Role: leader
Term: 1
Leader: self
Vote: self

Last Election started 2481636 ms ago, reason: timeout
Last Election won: 2481635 ms ago
Election timer: 5000
Log: [131, 131]
Entries not yet committed: 0
Entries not yet applied: 0
Connections: <-6e84 ->6e84
Disconnections: 2
Servers:
    6e84 (6e84 at tcp:[172.18.0.4]:6644) next_index=131 match_index=130 last msg 642 ms ago
    3722 (3722 at tcp:[172.18.0.3]:6644) (self) next_index=2 match_index=130
status: ok
```

██████████ ovn-central

████ ovn-central ████ `NODE_IPS` ████████████

```
kubectl label node kube-ovn-control-plane2 kube-ovn/role-
kubectl scale deployment -n kube-system ovn-central --replicas=2
kubectl set env deployment/ovn-central -n kube-system NODE_IPS="172.18.0.3,172.18.0.4"
kubectl rollout status deployment/ovn-central -n kube-system
```

████████ ovn-central ██

██ `ovs-ovn` ██████████████

```
# kubectl set env daemonset/ovs-ovn -n kube-system OVN_DB_IPS="172.18.0.3,172.18.0.4"
daemonset.apps/ovs-ovn env updated
# kubectl delete pod -n kube-system -lapp=ovs
pod "ovs-ovn-4f6jc" deleted
```

```
pod "ovs-ovn-csn2w" deleted
pod "ovs-ovn-mpbmb" deleted
```

██ `kube-ovn-controller` ██████████████████

```
# kubectl set env deployment/kube-ovn-controller -n kube-system OVN_DB_IPS="172.18.0.3,172.18.0.4"
deployment.apps/kube-ovn-controller env updated

# kubectl rollout status deployment/kube-ovn-controller -n kube-system
Waiting for deployment "kube-ovn-controller" rollout to finish: 1 of 3 updated replicas are available...
Waiting for deployment "kube-ovn-controller" rollout to finish: 2 of 3 updated replicas are available...
deployment "kube-ovn-controller" successfully rolled out
```

████

██ `kube-ovn-control-plane2` ████████████████████████████

```
rm -rf /etc/origin/ovn
```

████████ Kubernetes ████████████████████

6.4.2 ovn-central ████

██████████ Kubernetes ████ `ovn-central` ███

████

███████ `/etc/origin/ovn` ███████ `ovnnb_db.db` ▯ `ovnsb_db.db` ██████████████

```
rm -rf /etc/origin/ovn
```

████ **ovn-central** ██████

███ `ovn-central` ██

```
# kubectl ko nb status
1b9a
Name: OVN_Northbound
Cluster ID: 32ca (32ca07fb-739b-4257-b510-12fa18e7cce8)
Server ID: 1b9a (1b9a5d76-e69b-410c-8085-39943d0cd38c)
Address: tcp:[172.18.0.3]:6643
Status: cluster member
Role: leader
Term: 44
Leader: self
Vote: self

Last Election started 1855739 ms ago, reason: timeout
Last Election won: 1855729 ms ago
Election timer: 5000
Log: [147, 147]
Entries not yet committed: 0
Entries not yet applied: 0
Connections: ->4984 <-4984
Disconnections: 0
Servers:
    4984 (4984 at tcp:[172.18.0.4]:6643) next_index=147 match_index=146 last msg 367 ms ago
    1b9a (1b9a at tcp:[172.18.0.3]:6643) (self) next_index=140 match_index=146
status: ok

# kubectl ko sb status
3722
Name: OVN_Southbound
Cluster ID: d4bd (d4bd37a4-0400-499f-b4df-b4fd389780f0)
Server ID: 3722 (3722d5ae-2ced-4820-a6b2-8b744d11fb3e)
Address: tcp:[172.18.0.3]:6644
Status: cluster member
Role: leader
Term: 33
Leader: self
Vote: self

Last Election started 1868589 ms ago, reason: timeout
Last Election won: 1868579 ms ago
Election timer: 5000
Log: [142, 142]
Entries not yet committed: 0
```

```
Entries not yet applied: 0
Connections: ->6e84 <-6e84
Disconnections: 0
Servers:
    6e84 (6e84 at tcp:[172.18.0.4]:6644) next_index=142 match_index=141 last msg 728 ms ago
    3722 (3722 at tcp:[172.18.0.3]:6644) (self) next_index=134 match_index=141
status: ok
```

⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

⬚⬚⬚⬚ ovn-central ⬚⬚⬚⬚ `NODE_IPS` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
kubectl label node kube-ovn-control-plane2 kube-ovn/role=master
kubectl scale deployment -n kube-system ovn-central --replicas=3
kubectl set env deployment/ovn-central -n kube-system NODE_IPS="172.18.0.3,172.18.0.4,172.18.0.5"
kubectl rollout status deployment/ovn-central -n kube-system
```

⬚⬚⬚⬚⬚⬚⬚⬚ **ovn-central** ⬚⬚

⬚⬚ `ovs-ovn` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
# kubectl set env daemonset/ovs-ovn -n kube-system OVN_DB_IPS="172.18.0.3,172.18.0.4,172.18.0.5"
daemonset.apps/ovs-ovn env updated
# kubectl delete pod -n kube-system -lapp=ovs
pod "ovs-ovn-4f6jc" deleted
pod "ovs-ovn-csn2w" deleted
pod "ovs-ovn-mpbmb" deleted
```

⬚⬚ `kube-ovn-controller` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
# kubectl set env deployment/kube-ovn-controller -n kube-system OVN_DB_IPS="172.18.0.3,172.18.0.4,172.18.0.5"
deployment.apps/kube-ovn-controller env updated

# kubectl rollout status deployment/kube-ovn-controller -n kube-system
Waiting for deployment "kube-ovn-controller" rollout to finish: 1 of 3 updated replicas are available...
Waiting for deployment "kube-ovn-controller" rollout to finish: 2 of 3 updated replicas are available...
deployment "kube-ovn-controller" successfully rolled out
```

PDF Slack Support

2025⬚9⬚10⬚

2022⬚5⬚24⬚

GitHub



6.5 OVN 数据库备份和恢复

本文档介绍如何进行数据库备份，以及在不同情况下如何通过已有的数据库文件进行集群恢复。

6.5.1 数据库备份

利用 kubectl 插件的 backup 命令可以对数据库文件进行备份，以用于故障时恢复：

```
# kubectl ko nb backup
tar: Removing leading `/' from member names
backup ovn-nb db to /root/ovnnb_db.060223191654183154.backup

# kubectl ko sb backup
tar: Removing leading `/' from member names
backup ovn-sb db to /root/ovnsb_db.060223191654183154.backup
```

6.5.2 集群部分故障恢复

若集群中存在部分节点因为断电，文件系统故障或磁盘空间不足导致工作异常，但是集群仍可正常工作可以通过如下步骤进行恢复。

查看日志确认状态异常

查看对应节点 `/var/log/ovn/ovn-northd.log`，若提示类似错误则可判断数据库存在异常

```
 * ovn-northd is not running
ovsdb-server: ovsdb error: error reading record 2739 from OVN_Northbound log: record 2739 advances commit index to 6308 but last log index is 6307
 * Starting ovsdb-nb
```

从集群中踢出对应节点

根据日志提示是 `OVN_Northbound` 还是 `OVN_Southbound` 选择对应的数据库进行操作。上述日志提示为 `OVN_Northbound` 则对 ovn-nb 进行操作：

```
# kubectl ko nb status
9182
Name: OVN_Northbound
Cluster ID: e75f (e75fa340-49ed-45ab-990e-26cb865ebc85)
Server ID: 9182 (9182e8dd-b5b0-4dd8-8518-598cc1e374f3)
Address: tcp:[10.0.128.61]:6643
Status: cluster member
Role: leader
Term: 1454
Leader: self
Vote: self

Last Election started 1732603 ms ago, reason: timeout
Last Election won: 1732587 ms ago
Election timer: 1000
Log: [7332, 12512]
Entries not yet committed: 1
Entries not yet applied: 1
Connections: ->f080 <-f080 <-e631 ->e631
Disconnections: 1
Servers:
    f080 (f080 at tcp:[10.0.129.139]:6643) next_index=12512 match_index=12510 last msg 63 ms ago
    9182 (9182 at tcp:[10.0.128.61]:6643) (self) next_index=10394 match_index=12510
    e631 (e631 at tcp:[10.0.131.173]:6643) next_index=12512 match_index=0
```

从集群中踢出状态异常节点：

```
kubectl ko nb kick e631
```

登录异常节点，删除对应的数据库文件：

```
mv /etc/origin/ovn/ovnnb_db.db /tmp
```

删除对应节点的 `ovn-central` Pod，等待集群自动恢复：

```
kubectl delete pod -n kube-system ovn-central-xxxx
```


2025 Kube-OVN Team

6.5.3 [illegible]

[illegible] leader[illegible]

[illegible] ovn-central

[illegible] `ovn-central` [illegible] `ovn-central` [illegible]

```
kubectl scale deployment -n kube-system ovn-central --replicas=0
```

[illegible]

[illegible] [illegible] [illegible]

> [illegible] [illegible] `ovsdb-tool cluster-to-standalone` [illegible]

[illegible] `ovn-central` [illegible] `NODE_IPS` [illegible] [illegible] `/etc/origin/ovn` [illegible] [illegible]

[illegible] `docker` [illegible]

```
docker run -it -v /etc/origin/ovn:/etc/ovn kubeovn/kube-ovn:v1.17.0 bash
cd /etc/ovn/
ovsdb-tool cluster-to-standalone ovnnb_db_standalone.db ovnnb_db.db
ovsdb-tool cluster-to-standalone ovnsb_db_standalone.db ovnsb_db.db
```

[illegible] containerd[illegible] docker[illegible] `ctr` [illegible]

```
ctr -n k8s.io image pull docker.io/kubeovn/kube-ovn:v1.17.0
ctr -n k8s.io run --rm -t \
  --mount type=bind,src=/etc/origin/ovn,dst=/etc/ovn,options=rbind:rw \
  docker.io/kubeovn/kube-ovn:v1.17.0 ovn-recover bash
cd /etc/ovn/
ovsdb-tool cluster-to-standalone ovnnb_db_standalone.db ovnnb_db.db
ovsdb-tool cluster-to-standalone ovnsb_db_standalone.db ovnsb_db.db
exit
```

[illegible] `ovn-central` Pod [illegible] `ovsdb-tool cluster-to-standalone` [illegible] `kubectl cp` [illegible]

[illegible] ovn-central [illegible]

[illegible]

```
mv /etc/origin/ovn/ovnnb_db.db /tmp
mv /etc/origin/ovn/ovnsb_db.db /tmp
```

[illegible]

[illegible] `ovnnb_db.db` [illegible] `ovnsb_db.db` [illegible] `ovn-central` [illegible] `NODE_IPS` [illegible] `/etc/origin/ovn/` [illegible]

```
mv /etc/origin/ovn/ovnnb_db_standalone.db /etc/origin/ovn/ovnnb_db.db
mv /etc/origin/ovn/ovnsb_db_standalone.db /etc/origin/ovn/ovnsb_db.db
```

[illegible] `ovn-central` [illegible]

```
kubectl scale deployment -n kube-system ovn-central --replicas=3
kubectl rollout status deployment/ovn-central -n kube-system
```

PDF Slack Support

2026⬚4⬚28⬚

2022⬚5⬚24⬚

GitHub 

6.5.4 ⬚⬚

2025 Kube-OVN Team

6.6 ▯▯▯▯ CIDR

▯▯▯▯▯▯▯ CIDR ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

> ▯▯▯▯ CIDR ▯▯▯▯▯▯▯ Pod ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ CIDR ▯▯▯▯▯▯▯▯▯▯ ▯▯ Join ▯▯ CIDR ▯▯▯▯▯▯ Join ▯▯ CIDR▯

6.6.1 ▯▯▯▯

▯▯ `kubectl edit` ▯▯▯▯ `cidrBlock` ▯ `gateway` ▯ `excludeIps` ▯

```
kubectl edit subnet test-subnet
```

6.6.2 ▯▯▯▯▯▯▯▯▯ Namespace ▯▯▯ Pod

▯▯▯▯▯ `test` Namespace ▯▯▯

```
for pod in $(kubectl get pod --no-headers -n test --field-selector spec.restartPolicy=Always -o custom-columns=NAME:.metadata.name,HOST:spec.hostNetwork |
awk '{if ($2!="true") print $1}'); do
  kubectl delete pod "$pod" -n test --ignore-not-found
done
```

▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ host ▯▯▯▯▯ Pod▯

```
for ns in $(kubectl get ns --no-headers -o custom-columns=NAME:.metadata.name); do
  for pod in $(kubectl get pod --no-headers -n "$ns" --field-selector spec.restartPolicy=Always -o custom-columns=NAME:.metadata.name,HOST:spec.hostNetwork |
awk '{if ($2!="true") print $1}'); do
    kubectl delete pod "$pod" -n "$ns" --ignore-not-found
  done
done
```

6.6.3 ▯▯▯▯▯▯▯▯▯

▯▯▯▯▯▯▯▯▯▯ CIDR ▯▯▯▯▯ `kube-ovn-controller` Deployment ▯▯▯▯▯▯

```
args:
- --default-cidr=10.17.0.0/16
- --default-gateway=10.17.0.1
- --default-exclude-ips=10.17.0.1
```

PDF Slack Support

2026▯4▯28▯

2022▯5▯24▯

GitHub



6.6.4 ▯▯

6.7 ▯▯ Join ▯▯ CIDR

▯▯▯▯▯▯ Join ▯▯ CIDR ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

> ▯▯ Join ▯▯ CIDR ▯▯▯▯▯▯▯▯ Pod ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯▯▯

6.7.1 ▯▯ Join ▯▯

```
kubectl patch subnet join --type='json' -p '[{"op": "replace", "path": "/metadata/finalizers", "value": []}]'
kubectl delete subnet join
```

6.7.2 ▯▯▯▯▯▯▯▯▯

```
kubectl annotate node ovn.kubernetes.io/allocated=false --all --overwrite
```

6.7.3 ▯▯ Join ▯▯▯▯▯▯▯

▯▯ `kube-ovn-controller` ▯ Join ▯▯▯▯▯▯▯▯

```
kubectl edit deployment -n kube-system kube-ovn-controller
```

▯▯▯▯▯▯▯▯

```
args:
- --node-switch-cidr=100.51.0.0/16
```

▯▯ `kube-ovn-controller` ▯▯ `join` ▯▯▯

```
kubectl delete pod -n kube-system -lapp=kube-ovn-controller
```

▯▯▯▯ Join ▯▯▯▯▯

```
# kubectl get subnet
NAME          PROVIDER   VPC           PROTOCOL   CIDR            PRIVATE   NAT     DEFAULT   GATEWAYTYPE   V4USED   V4AVAILABLE   V6USED   V6AVAILABLE
EXCLUDEIPS
join          ovn        ovn-cluster   IPv4       100.51.0.0/16   false     false   false     distributed   2        65531         0        0
["100.51.0.1"]
ovn-default   ovn        ovn-cluster   IPv4       10.17.0.0/16    false     true    true      distributed   5        65528         0        0
["10.17.0.1"]
```

6.7.4 ▯▯▯▯ ovn0 ▯▯▯▯

▯▯▯▯▯ `ovn0` ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `kube-ovn-cni` ▯▯▯▯

```
kubectl delete pod -n kube-system -l app=kube-ovn-cni
```

PDF Slack Support

2025▯9▯10▯

2022▯5▯24▯

GitHub



6.7.5 ▯▯

6.8 [illegible]

[illegible] `kube-ovn.yaml` [illegible]

```
vi kube-ovn.yaml
# ...
        - name: kube-ovn-controller
          image: "docker.io/kubeovn/kube-ovn:v1.17.0"
          imagePullPolicy: IfNotPresent
          args:
          - /kube-ovn/start-controller.sh
          - --v=3
# ...
#
```

PDF Slack Support

2025[illegible]9[illegible]10[illegible]

2023[illegible]4[illegible]4[illegible]

GitHub 

6.8.1 [illegible]

6.9 ■■■■■■■

■■■■■■■■■■■■■■■■ ■■■■■ ■■ Kube-OVN ■■■■■■■■■■■■■■■

6.9.1 ■■■■■■■

1. ■■■■■■■■■■
    - ■■ Pod ■ Pod ■■■■■■■
    - ■■ Pod ■■■■■■■■■■■■
    - ■■ Pod ■■ Service ■■■■■
    - ■■ Pod ■■■■■■■■■■
    - ■■ Pod ■■■■■■■■■■■■
    - ■■■■■■■■ `kube-ovn-pinger` ■■■■■■■■■■■■■■■■■■■■■■■■
2. ■■■■■■■■■■■■■■■■■■■■■■■■■■
    - ■■ Pod ■■■■■■■■■■■■■■■■■■■■■■■
    - ■■■■■ kube-ovn-cni ■■■■■■ CNI ■■■■■■■■■■
    - ■■■■■ ovs-ovn ■■■■■■■■■■■ OVS ■■■■■
    - ■■ kube-ovn-controller ■■■■■■■■■■■■■■■■■■■■
    - ■■ ovn-central ■■■■■■■■■■■ OVN ■■■■■
    - ■■■■■ dmesg ■■■■■■■■■■■■■■■■■■■
    - ■■■■■ `netstat -s` ■■■■■■■■■■■■■■■■
3. ■■■■■■■■■■■■■■■■■■■■ CPU, ■■■■ IO ■■■■■■■■

■■■■■■■■■■■■■ `kubectl ko log` ■■■■■■■■

6.9.2 Pod ■■■■■■■■■■■ IP ■■

■■

Pod ■■■■■ Running ■■■kubectl describe Pod ■■■■■ `duplicate IPv4 address <ip> found on logical switch port <port>`

■■■■■

1. ■■■■■ Pod ■■■■■ IP Pod■■■■■■ IP ■■■■ IP ■■■■■■■■■■■ IP ■■■■■■■ IP Pod
2. ■■ kube-ovn-controller ■■■■■■■■ IP ■■■■■■■■■■
3. ■■■■■■■ kube-ovn-controller ■■■■■■■■■■■■■ IP ■■■■■
4. `kubectl ko nbctl show` ■■ OVN ■■■■■■■■■■■■■■ IP ■■
5. ■ OVN ■■■■■■■■■■■ Kubernetes ■■■■■■■■■■■■■ IP ■■■■■■■■■■■ `kubectl ko nbctl lsp-del <port>` ■■■■■

6.9.3 Pod ■■■■■■■■■■■ ping gateway failed

■■

Pod ■■■■■ Running ■■■kubectl describe Pod ■■■■■ `network <ip> with gateway <gw ip> is not ready for interface eth0 after 30 checks`

■■■■■

1. `kubectl ko sbctl show` ■■ Pod ■■■■■■■■■■■■■
2. ■■■■■■■■■■■ ovn-central ■ ovs-ovn ■■■■■■■■■■■■■■■■■■■■■■■■■■■ ovn-central ■ ovs-ovn ■■■■■

2025 Kube-OVN Team

3. 若信息存在，观察是否存在网络策略阻止 Pod 访问网关
4. 若为 Underlay 网络，询问网络管理员底层网络是否有安全限制，参考 Underlay 网络安装 文档

6.9.4 Pod 无法访问外部网络

现象

默认 VPC 下 Pod 在集群内访问正常，无法访问集群外部网络。

解决方法

1. 使用 `kubectl ko trace` 判断 OVN 逻辑流表链路上是否存在 ACL 限制
2. 若存在 ACL 限制，调整 ACL 策略
3. 若不存在 ACL 限制，查看 Subnet stats 信息，确认网关节点信息是否正确，且网关节点网络正常
4. 若网关节点信息不正确，或对应节点出现网络异常，修改 Subnet Spec 更新为新的网关节点
5. 收集 `kube-ovn-controller` 日志，确认是否存在网关切换的异常日志

6.9.5 Pod IP 不在指定的 CIDR 内

现象

Pod 创建成功，但是 IP 不在指定的 CIDR 内。

解决方法

1. 查找主机路径的 `/etc/cni/net.d/` 目录下是否存在其他非 Kube-OVN 的 CNI 配置文件
2. 若存在，重命名该文件，或者删除该文件
3. 重启 kubelet 以及对应节点上的所有 Pod

6.9.6 Debug Pod 无法正常创建

现象

kubectl debug 创建的 Pod 一直处于 ContainerCreating 状态，Pod Event 报错 network not ready 或 no address allocated

解决方法

创建 debug Pod 后，导出其 yaml 文件，删除 yaml 中的 Annotation：

```
ovn.kubernetes.io/ip_address
ovn.kubernetes.io/mac_address
ovn.kubernetes.io/allocated
ovn.kubernetes.io/routed
```

删除 debug Pod，使用修改后的 yaml 创建新的 debug Pod。

6.9.7 ARM 系统跨主机容器访问间歇失败

现象

ARM 系统和部分网卡 Offload 配合存在问题，会导致容器网络间歇故障。

使用 `netstat` 确认问题：

```
# netstat -us
IcmpMsg:
    InType0: 22
    InType3: 24
    InType8: 117852
    OutType0: 117852
```

```
    OutType3: 29
    OutType8: 22
Udp:
    3040636 packets received
    0 packets to unknown port received.
    4 packet receive errors
    602 packets sent
    0 receive buffer errors
    0 send buffer errors
    InCsumErrors: 4
UdpLite:
IpExt:
    InBcastPkts: 10244
    InOctets: 4446320361
    OutOctets: 1496815600
    InBcastOctets: 3095950
    InNoECTPkts: 7683903
```

若存在 `InCsumErrors`，且随着访问失败增加，可确认是该问题。

解决方法

根本解决需要和操作系统以及对应网卡厂商沟通，更新系统和驱动。临时解决可先关闭物理网卡的 `tx offload` 但是会导致 TCP 性能有较明显下降。

```
ethtool -K eth0 tx off
```

经社区用户反馈，CentOS 7 和麒麟操作系统都可能出现类似问题，其中麒麟操作系统在 `4.19.90-25.16.v2101` 内核后可以解决该问题。

6.9.8 Pod 访问 Service 超时

现象

Pod 内无法访问 Service 对应的服务，`dmesg` 显示异常：

```
netlink: Unknown conntrack attr (type=6, max=5)
openvswitch: netlink: Flow actions may not be safe on all matching packets.
```

该日志说明内核内 OVS 版本过低不支持对应 NAT 操作。

解决方法

1. 升级内核模块或手动编译 OVS 内核模块。
2. 若只使用 Overlay 网络可以更改 `kube-ovn-controller` 启动参数设置 `--enable-lb=false` 关闭 OVN LB 使用 kube-proxy 进行 Service 转发。

6.9.9 ovn-central 占用过高

现象

从 v1.11.x 版本开始，1 万个 Pod 以上的集群，如果 OVN NB 或者 SB 出现频繁选主的情况，可能原因是 Kube-OVN 周期进行了 ovsdb-server/compact 动作，影响到选主逻辑。

解决方法

为缓解该问题，自 v1.13 起 `ENABLE_COMPACT` 已默认置为 `false`（参见 `dist/images/install.sh` 与 `start-db.sh`），无需再手动配置。如果集群是从更早版本升级而来，请确认 ovn-central 容器的环境变量未显式设置为 `true`。

```
- name: ENABLE_COMPACT
  value: "false"
```

PDF Slack Support

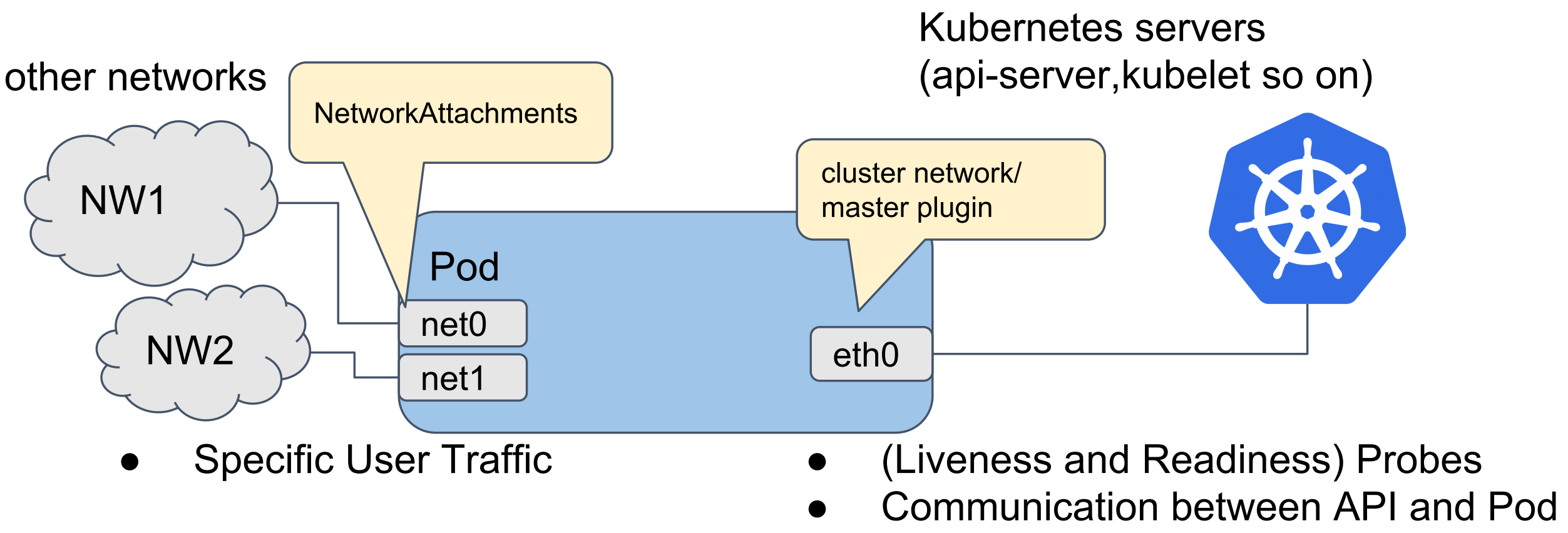
2025 Kube-OVN Team

2026▯11▯7▯

2022▯5▯24▯

GitHub 

6.9.10 ▯▯



7. [illegible]

7.1 [illegible]

Kube-OVN 可以为其他 CNI 网络插件（如 macvlan、vlan、host-device 等）提供集群级别的 IPAM 能力，使这些网络插件能够使用 Kube-OVN 的子网和固定 IP 功能。

Kube-OVN 还支持在多块网卡均为 Kube-OVN 类型时的地址管理。

7.1.1 [illegible]

多网卡管理：

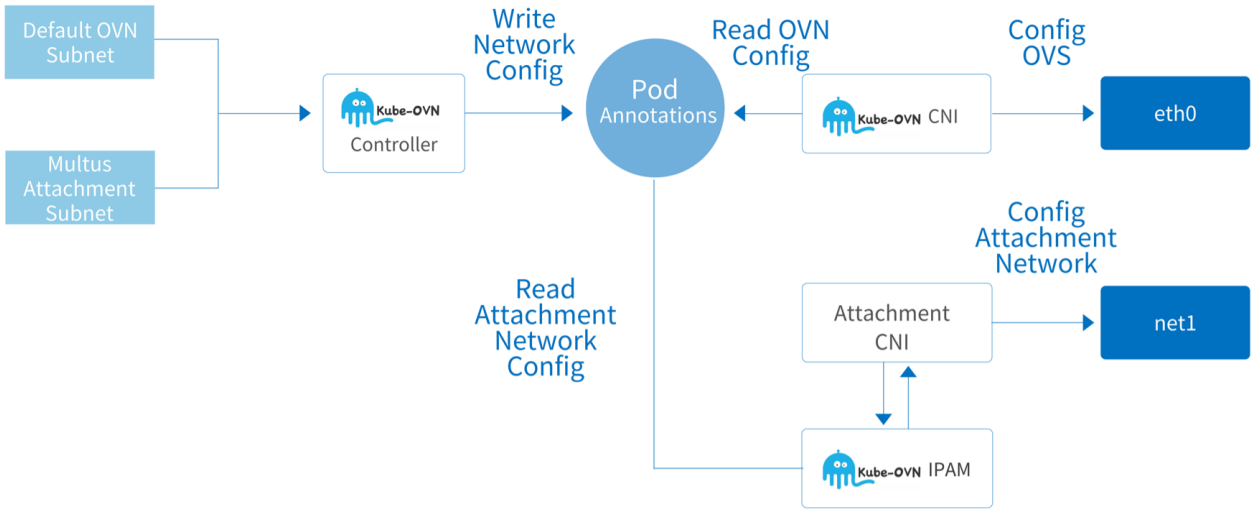
下面是由 Multus CNI 提供的连接到 pod 的网络接口示意图。图中显示 pod 有三个接口：eth0、net0 和 net1。eth0 连接 kubernetes 集群网络，用于连接 kubernetes 服务器（如 kubernetes api-server、kubelet 等）。net0 和 net1 是附属网络接口，通过使用其他 CNI 插件（如 vlan/vxlan/ptp）连接其他网络。

IPAM：

通过使用 Multus CNI，我们可以给一个 Pod 添加多块不同网络的网卡。然而我们仍然缺乏对集群范围内不同网络的 IP 地址进行管理的能力。在 Kube-OVN 中，我们已经能够通过 Subnet 和 IP 的 CRD 来进行 IP 的高级管理，例如子网管理、IP 预留、随机分配、固定分配等。现在我们对子网进行扩展，以接入其他不同的网络插件，使得其他网络插件也可以使用 Kube-OVN 的 IPAM 功能。

2025 Kube-OVN Team

⬚⬚⬚⬚



⬚⬚⬚⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ IP ⬚⬚⬚⬚⬚⬚⬚⬚ eth0 ⬚⬚⬚⬚ OVN ⬚⬚⬚net1 ⬚⬚⬚⬚⬚⬚ CNI ⬚⬚⬚ net1 ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ multus-cni ⬚⬚ NetworkAttachmentDefinition ⬚⬚⬚⬚⬚

⬚ Pod ⬚⬚⬚⬚ `kube-ovn-controller` ⬚⬚⬚⬚ Pod ⬚⬚⬚⬚⬚⬚⬚⬚ Pod ⬚⬚ annotation ⬚⬚⬚⬚⬚⬚⬚ Subnet ⬚⬚⬚⬚⬚ IP ⬚⬚⬚⬚⬚⬚⬚ ⬚⬚ Pod ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Pod annotation ⬚⬚

⬚⬚⬚⬚⬚⬚⬚⬚⬚CNI ⬚⬚⬚⬚⬚⬚⬚⬚⬚ `kube-ovn-cni` ⬚⬚⬚ ipam ⬚⬚⬚ `kube-ovn-cni` ⬚⬚⬚⬚ Pod annotation ⬚⬚⬚⬚⬚⬚⬚⬚ CNI ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ CNI ⬚⬚⬚

7.1.2 ⬚⬚⬚⬚⬚

`NetworkAttachmentDefinition` ⬚⬚ `spec` ⬚⬚⬚multus ⬚⬚⬚⬚ `defaultConfDir` ⬚⬚⬚⬚⬚ CNI ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-conf-2
```

⬚⬚ `kube-ovn-controller` ⬚⬚⬚⬚⬚⬚⬚⬚ `NetworkAttachmentDefinition` ⬚⬚ `provider` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `spec` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `kube-ovn-controller` ⬚⬚⬚⬚⬚⬚⬚⬚ `provider` ⬚⬚⬚⬚⬚⬚ `spec` ⬚⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚ IPAM ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

7.1.3 ⬚⬚⬚⬚

⬚⬚ Kube-OVN ⬚ Multus

⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚ ⬚ Multus how to use ⬚⬚⬚ Kube-OVN ⬚ Multus-CNI⬚

⬚⬚⬚ CNI ⬚⬚ IPAM

⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ CNI⬚

⬚⬚ NETWORKATTACHMENTDEFINITION

⬚⬚⬚⬚⬚⬚ macvlan ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ ipam ⬚⬚⬚ `kube-ovn` ⬚

```
#    macvlan
sudo modprobe macvlan
```


2025 Kube-OVN Team

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan
  namespace: default
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "macvlan",
      "master": "eth0",
      "mode": "bridge",
      "ipam": {
        "type": "kube-ovn",
        "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
        "provider": "macvlan.default"
      }
    }'
```

- `spec.config.ipam.type`：需要设置为 `kube-ovn` 来调用 kube-ovn 的插件获取地址信息。
- `server_socket`：Kube-OVN 通信使用的 socket 文件，默认位置为 `/run/openvswitch/kube-ovn-daemon.sock`。
- `provider`：当前 NetworkAttachmentDefinition 的 `<name>.<namespace>`，Kube-OVN 将会使用这些信息找到对应的 Subnet 资源。
- `master`：宿主机的物理网卡

> **Info**
>
> 这里的 `provider` 和 Underlay 中的 `ProviderNetwork` 是两个不同的概念，并没有直接关联。由于早期命名的原因可能会造成误解，请注意区分。

[illegible] KUBE-OVN SUBNET

创建一个 Kube-OVN Subnet，设置对应的 `cidrBlock` 和 `exclude_ips`，`provider` 应该设置为对应的 NetworkAttachmentDefinition 的 `<name>.<namespace>`。例如用 macvlan 提供附属网卡，创建 Subnet 如下：

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: macvlan
spec:
  protocol: IPv4
  provider: macvlan.default
  cidrBlock: 172.17.0.0/16
  gateway: 172.17.0.1
  excludeIps:
  - 172.17.0.0..172.17.0.10
```

> 注意：`gateway`、`private`、`nat` 字段只对 `provider` 类型为 ovn 的网络生效，不适用于 attachment network（附属网络）。

创建一个多网络的 Pod

对于地址随机分配的 Pod，只需要添加如下 annotation `k8s.v1.cni.cncf.io/networks`，取值为对应的 NetworkAttachmentDefinition 的 `<namespace>/<name>`：

```
apiVersion: v1
kind: Pod
metadata:
  name: samplepod
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: default/macvlan
spec:
  containers:
  - name: samplepod
    command: ["/bin/ash", "-c", "trap : TERM INT; sleep infinity & wait"]
    image: docker.io/library/alpine:edge
```

创建固定 IP 的 Pod

对于固定 IP 的 Pod，添加 `<networkAttachmentName>.<networkAttachmentNamespace>.kubernetes.io/ip_address` annotation：

```
apiVersion: v1
kind: Pod
metadata:
  name: static-ip
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: default/macvlan
    ovn.kubernetes.io/ip_address: 10.16.0.15
    ovn.kubernetes.io/mac_address: 00:00:00:53:6B:B6
```


2025 Kube-OVN Team

```
    macvlan.default.kubernetes.io/ip_address: 172.17.0.100
    macvlan.default.kubernetes.io/mac_address: 00:00:00:53:6B:BB
spec:
  containers:
  - name: static-ip
    image: docker.io/library/nginx:alpine
```

... IP ...

... ippool ... `<networkAttachmentName>.<networkAttachmentNamespace>.kubernetes.io/ip_pool` annotation...

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: default
  name: static-workload
  labels:
    app: static-workload
spec:
  replicas: 2
  selector:
    matchLabels:
      app: static-workload
  template:
    metadata:
      labels:
        app: static-workload
      annotations:
        k8s.v1.cni.cncf.io/networks: default/macvlan
        ovn.kubernetes.io/ip_pool: 10.16.0.15,10.16.0.16,10.16.0.17
        macvlan.default.kubernetes.io/ip_pool: 172.17.0.200,172.17.0.201,172.17.0.202
    spec:
      containers:
      - name: static-workload
        image: docker.io/library/nginx:alpine
```

... macvlan ... Pod

... macvlan ... Pod...Pod ... annotation... `default-route` ...

```
apiVersion: v1
kind: Pod
metadata:
  name: samplepod-route
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: '[{
      "name": "macvlan",
      "namespace": "default",
      "default-route": ["172.17.0.1"]
    }]'
spec:
  containers:
  - name: samplepod-route
    command: ["/bin/ash", "-c", "trap : TERM INT; sleep infinity & wait"]
    image: docker.io/library/alpine:edge
```

... macvlan ... Pod

... macvlan ... Pod... annotation `v1.multus-cni.io/default-network` ... NetworkAttachmentDefinition ... `<namespace>/<name>` ...

```
apiVersion: v1
kind: Pod
metadata:
  name: samplepod-macvlan
  namespace: default
  annotations:
    v1.multus-cni.io/default-network: default/macvlan
spec:
  containers:
  - name: samplepod-macvlan
    command: ["/bin/ash", "-c", "trap : TERM INT; sleep infinity & wait"]
    image: docker.io/library/alpine:edge
```

... KUBE-OVN SUBNET...PROVIDER OVN...

... `provider` ... ovn ... subnet ... IP ... `provider` ... ovn ... Kube-OVN Subnet... `cidrBlock` ... `exclude_ips` ... Subnet ...

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: macvlan
spec:
  protocol: IPv4
  provider: ovn
```


2025 Kube-OVN Team

```
  cidrBlock: 172.17.0.0/16
  gateway: 172.17.0.1
  excludeIps:
  - 172.17.0.0..172.17.0.10
```

▒▒▒▒▒▒▒▒ **Pod**

▒▒▒▒▒ `provider` ▒▒▒ ovn ▒ subnet ▒▒▒ IP ▒ Pod▒▒▒ annotation `k8s.v1.cni.cncf.io/networks` ▒ `<networkAttachmentName>.<networkAttachmentNamespace>.kubernetes.io/logical_switch` ▒▒▒▒▒

```
apiVersion: v1
kind: Pod
metadata:
  name: samplepod
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: default/macvlan
    macvlan.default.kubernetes.io/logical_switch: macvlan
spec:
  containers:
  - name: samplepod
    command: ["/bin/ash", "-c", "trap : TERM INT; sleep infinity & wait"]
    image: docker.io/library/alpine:edge
```

- `k8s.v1.cni.cncf.io/networks` : ▒▒▒▒▒▒ NetworkAttachmentDefinition ▒ `<namespace>/<name>`
- `macvlan.default.kubernetes.io/logical_switch` : ▒▒▒▒▒▒

> ▒▒▒
>
> - ▒▒ `<networkAttachmentName>.<networkAttachmentNamespace>.kubernetes.io/logical_switch` ▒▒▒▒▒▒▒▒▒▒▒▒ provider ▒▒▒▒▒
> - ▒▒ ovn ▒▒▒ subnet ▒▒ ipam ▒▒▒▒▒▒▒▒ IP ▒ Pod▒▒▒▒▒▒▒ IP ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ macvlan ▒ Pod▒
> - ▒▒▒▒▒▒▒▒▒▒▒ macvlan ▒ Pod▒

▒▒▒▒▒ **Kube-OVN** ▒▒▒▒

▒▒▒▒▒▒▒▒ Kube-OVN ▒▒▒▒▒

▒▒ NETWORKATTACHMENTDEFINITION

▒ `provider` ▒▒▒▒▒▒ `ovn` ▒

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: attachnet
  namespace: default
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "kube-ovn",
      "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
      "provider": "attachnet.default.ovn"
    }'
```

- `spec.config.type` : ▒▒▒ `kube-ovn` ▒▒▒ CNI ▒▒▒▒ Kube-OVN ▒▒▒
- `server_socket` : Kube-OVN ▒▒▒▒▒ socket ▒▒▒▒▒▒▒▒ `/run/openvswitch/kube-ovn-daemon.sock` ▒
- `provider` : ▒▒ NetworkAttachmentDefinition ▒ `<name>.<namespace>.ovn` ▒Kube-OVN ▒▒▒▒▒▒▒▒▒▒▒▒▒ Subnet ▒▒▒▒▒▒▒▒▒▒▒▒ ovn▒

▒▒▒▒ KUBE-OVN SUBNET

▒▒▒ Kube-OVN ▒▒▒▒▒▒▒▒ `provider` ▒▒▒▒▒▒▒▒ NetworkAttachmentDefinition ▒ `<name>.<namespace>.ovn` ▒▒▒▒ `ovn` ▒▒▒▒▒▒▒ ▒ Kube-OVN ▒▒▒▒▒▒▒▒▒▒ Subnet ▒▒▒▒▒

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: attachnet
spec:
  protocol: IPv4
  provider: attachnet.default.ovn
  cidrBlock: 172.17.0.0/16
  gateway: 172.17.0.1
  excludeIps:
  - 172.17.0.0..172.17.0.10
```


2025 Kube-OVN Team

... Pod

... Pod... annotation `k8s.v1.cni.cncf.io/networks` ... NetworkAttachmentDefinition ... `<namespace>/<name>` ...

```
apiVersion: v1
kind: Pod
metadata:
  name: samplepod
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: default/attachnet
spec:
  containers:
  - name: samplepod
    command: ["/bin/ash", "-c", "trap : TERM INT; sleep infinity & wait"]
    image: docker.io/library/alpine:edge
```

... IP/MAC

... NetworkAttachmentDefinition ... IP/MAC... `k8s.v1.cni.cncf.io/networks` ... `interface` ... annotation... `<nadName>.<nadNamespace>.kubernetes.io/ip_address.<interfaceName>` ... `<nadName>.<nadNamespace>.kubernetes.io/mac_address.<interfaceName>` ... - ... IP/MAC...

...

... Kube-OVN ... Pod... `<networkAttachmentName>.<networkAttachmentNamespace>.ovn.kubernetes.io/routes` annotation ...

```
apiVersion: v1
kind: Pod
metadata:
  name: custom-routes-attach
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: default/attachnet
    attachnet.default.ovn.kubernetes.io/routes: |
      [{
        "dst": "192.168.0.101/24",
        "gw": "172.17.0.254"
      }, {
        "gw": "172.17.0.254"
      }]
spec:
  containers:
  - name: custom-routes-attach
    command: ["/bin/ash", "-c", "trap : TERM INT; sleep infinity & wait"]
    image: docker.io/library/alpine:edge
```

> `dst` ...

... Deployment...DaemonSet ... StatefulSet... Annotation ... `.spec.template.metadata.annotations` ...

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: custom-routes-attach
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.v1.cni.cncf.io/networks: default/attachnet
        attachnet.default.ovn.kubernetes.io/routes: |
          [{
            "dst": "192.168.0.101/24",
            "gw": "172.17.0.254"
          }, {
            "gw": "172.17.0.254"
          }]
    spec:
      containers:
      - name: nginx
        image: docker.io/library/nginx:alpine
```

... KUBE-OVN SUBNET...PROVIDER OVN...

... `provider` ... ovn ... subnet ... IP ... `provider` ... ovn ... Kube-OVN Subnet... `cidrBlock` ... `exclude_ips` ... Subnet ...

2025 Kube-OVN Team

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: attachnet
spec:
  protocol: IPv4
  provider: ovn
  cidrBlock: 172.17.0.0/16
  gateway: 172.17.0.1
  excludeIps:
  - 172.17.0.0..172.17.0.10
```

▯▯▯▯▯▯▯▯ **Pod**

▯▯▯▯▯ `provider` ▯▯▯ ovn ▯ subnet ▯▯▯ IP ▯ Pod▯▯▯ annotation `k8s.v1.cni.cncf.io/networks` ▯ `<networkAttachmentName>.<networkAttachmentNamespace>.ovn.kubernetes.io/logical_switch` ▯▯▯▯▯

```
apiVersion: v1
kind: Pod
metadata:
  name: samplepod
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: default/attachnet
    attachnet.default.ovn.kubernetes.io/logical_switch: attachnet
spec:
  containers:
  - name: samplepod
    command: ["/bin/ash", "-c", "trap : TERM INT; sleep infinity & wait"]
    image: docker.io/library/alpine:edge
```

- `k8s.v1.cni.cncf.io/networks` : ▯▯▯▯▯▯ NetworkAttachmentDefinition ▯ `<namespace>/<name>`
- `attachnet.default.ovn.kubernetes.io/logical_switch` : ▯▯▯▯▯▯

> ▯▯▯
>
> - ▯▯ `<networkAttachmentName>.<networkAttachmentNamespace>.ovn.kubernetes.io/logical_switch` ▯▯▯▯▯▯▯▯▯▯▯▯ provider ▯▯▯▯▯
> - ▯▯▯▯▯▯▯ Kube-OVN ▯▯▯ Pod▯▯▯▯▯▯▯ IP ▯ Pod▯▯▯▯▯▯▯ IP ▯▯▯▯▯▯▯▯▯▯▯▯▯ macvlan ▯ Pod▯▯▯▯▯▯▯▯▯▯▯▯▯ Kube-OVN ▯▯▯ Pod▯▯▯▯▯▯▯▯▯▯▯▯▯
> - ▯▯▯▯▯▯▯ Kube-OVN ▯▯▯ Pod▯▯▯▯▯▯▯▯ `<networkAttachmentName>.<networkAttachmentNamespace>.ovn.kubernetes.io/routes` annotation ▯▯▯▯▯ ▯▯▯

PDF Slack Support

2026▯2▯11▯

2022▯5▯20▯

GitHub



7.2 ...

...Kube-OVN ...

...

...Kube-OVN ...

7.2.1 ...

...

1. CNI ... CNI ...
2. ... 10ns ... 20ns...
3. ... CPU... CPU ...
4. ... Macvlan ... SR-IOV ...

7.2.2 ...

> ... ...

Overlay ...

...

- Kubernetes: 1.22.0
- OS: CentOS 7
- Kube-OVN: 1.8.0 *Overlay* ...
- CPU: Intel(R) Xeon(R) E-2278G
- Network: 2*10Gbps, xmit_hash_policy=layer3+4

... `qperf -t 60 <server ip> -ub -oo msg_size:1 -vu tcp_lat tcp_bw udp_lat udp_bw` ... 1 ... tcp/udp ...

Type	tcp_lat (us)	udp_lat (us)	tcp_bw (Mb/s)	udp_bw(Mb/s)
Kube-OVN Default	25.7	22.9	27.1	1.59
Kube-OVN Optimized	13.9	12.9	27.6	5.57
HOST Network	13.1	12.4	28.2	6.02

Overlay Underlay ...

... Kube-OVN ... Overlay ... Underlay ...

Environment:

- Kubernetes: 1.22.0
- OS: CentOS 7
- Kube-OVN: 1.8.0
- CPU: AMD EPYC 7402P 24-Core Processor
- Network: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28

2025 Kube-OVN Team

`qperf -t 60 <server ip> -ub -oo msg_size:1 -vu tcp_lat tcp_bw udp_lat udp_bw`

Type	tcp_lat (us)	udp_lat (us)	tcp_bw (Mb/s)	udp_bw(Mb/s)
Kube-OVN Overlay	15.2	14.6	23.6	2.65
Kube-OVN Underlay	14.3	13.8	24.2	3.46
HOST Network	16.6	15.4	24.8	2.64

`qperf -t 60 <server ip> -ub -oo msg_size:1K -vu tcp_lat tcp_bw udp_lat udp_bw`

Type	tcp_lat (us)	udp_lat (us)	tcp_bw (Gb/s)	udp_bw(Gb/s)
Kube-OVN Overlay	16.5	15.8	10.2	2.77
Kube-OVN Underlay	15.9	14.5	9.6	3.22
HOST Network	18.1	16.6	9.32	2.66

`qperf -t 60 <server ip> -ub -oo msg_size:4K -vu tcp_lat tcp_bw udp_lat udp_bw`

Type	tcp_lat (us)	udp_lat (us)	tcp_bw (Gb/s)	udp_bw(Gb/s)
Kube-OVN Overlay	34.7	41.6	16.0	9.23
Kube-OVN Underlay	32.6	44	15.1	6.71
HOST Network	35.9	45.9	14.6	5.59

> [illegible] netfilter [illegible] `kube-proxy` [illegible] netfilter[illegible] [illegible]

7.2.3 [illegible]

[illegible]

CPU [illegible]

[illegible] CPU [illegible] CPU [illegible]

```
cpupower frequency-set -g performance
```

[illegible]

[illegible]

[illegible]

[illegible]

```
# ethtool -g eno1
 Ring parameters for eno1:
 Pre-set maximums:
 RX:             4096
 RX Mini:        0
 RX Jumbo:       0
 TX:             4096
 Current hardware settings:
 RX:             255
 RX Mini:        0
 RX Jumbo:       0
 TX:             255
```

[illegible]

```
ethtool -G eno1 rx 4096
ethtool -G eno1 tx 4096
```

[illegible] **tuned** [illegible]

tuned [illegible] profile [illegible]

[illegible]

```
tuned-adm profile network-latency
```

[illegible]

```
tuned-adm profile network-throughput
```

[illegible]

[illegible] `irqbalance` [illegible] CPU [illegible] CPU [illegible]

[illegible] **OVN LB**

OVN [illegible] L2 LB [illegible] `conntrack` [illegible] recirculate [illegible] CPU [illegible] 20% [illegible] CPU [illegible] [illegible] Overlay [illegible] `kube-proxy` [illegible] Service [illegible] [illegible] Pod-to-Pod [illegible] `kube-ovn-controller` [illegible]

```
command:
- /kube-ovn/start-controller.sh
args:
...
- --enable-lb=false
...
```

> Underlay [illegible] `kube-proxy` [illegible] iptables [illegible] ipvs [illegible] LB [illegible] Service [illegible]

[illegible] **conntrack** [illegible]

[illegible] OVN LB [illegible] Service [illegible] Service [illegible] NetworkPolicy [illegible] Subnet A [illegible] Pod [illegible] Subnet B [illegible] Service[illegible] [illegible] kube-ovn-controller [illegible] `--skip-conntrack-dst-cidrs` [illegible] conntrack [illegible]

```
    --skip-conntrack-dst-cidrs="10.17.0.0/16,169.254.169.245/32"
```

[illegible] **FastPath** [illegible]

[illegible] network ns[illegible] netfilter [illegible] 20% [illegible] CPU [illegible] [illegible] netfilter [illegible] `FastPath` [illegible] netfilter [illegible] CPU [illegible] [illegible]

> [illegible] netfilter [illegible] iptables[illegible]ipvs[illegible]nftables [illegible]

[illegible]

[illegible] FastPath [illegible]

[illegible] `insmod kube_ovn_fastpath.ko` [illegible] `FastPath` [illegible] `dmesg` [illegible]

```
# dmesg
...
[619631.323788] init_module,kube_ovn_fastpath_local_out
[619631.323798] init_module,kube_ovn_fastpath_post_routing
[619631.323800] init_module,kube_ovn_fastpath_pre_routing
[619631.323801] init_module,kube_ovn_fastpath_local_in
...
```

OVS [illegible]

OVS [illegible] flow [illegible] 10% [illegible] CPU [illegible] x86 CPU [illegible] `popcnt` [illegible] `sse4.2` [illegible] [illegible]flow [illegible] CPU [illegible] [illegible] 5% [illegible]

[illegible] `FastPath` [illegible]

2025 Kube-OVN Team

[illegible] CPU [illegible]

```
cat /proc/cpuinfo  | grep popcnt
cat /proc/cpuinfo  | grep sse4_2
```

CENTOS [illegible]

[illegible]

```
yum install -y gcc kernel-devel-$(uname -r) python3 autoconf automake libtool rpm-build openssl-devel
```

[illegible] OVS [illegible] RPM [illegible]:

```
git clone -b branch-3.5 --depth=1 https://github.com/openvswitch/ovs.git
cd ovs
curl -s  https://github.com/kubeovn/ovs/commit/2d2c83c26d4217446918f39d5cd5838e9ac27b32.patch |  git apply
./boot.sh
./configure --with-linux=/lib/modules/$(uname -r)/build CFLAGS="-g -O2 -mpopcnt -msse4.2"
make rpm-fedora-kmod
cd rpm/rpmbuild/RPMS/x86_64/
```

[illegible] RPM [illegible]

```
rpm -i openvswitch-kmod-3.5.1-1.el7.x86_64.rpm
```

[illegible] Kube-OVN[illegible] OVS [illegible]

UBUNTU [illegible]

[illegible]

```
apt install -y autoconf automake libtool gcc build-essential libssl-dev
```

[illegible] OVS [illegible]

```
apt install -y autoconf automake libtool gcc build-essential libssl-dev

git clone -b branch-3.5 --depth=1 https://github.com/openvswitch/ovs.git
cd ovs
curl -s  https://github.com/kubeovn/ovs/commit/2d2c83c26d4217446918f39d5cd5838e9ac27b32.patch |  git apply
./boot.sh
./configure --prefix=/usr/ --localstatedir=/var --enable-ssl --with-linux=/lib/modules/$(uname -r)/build
make -j `nproc`
make install
make modules_install

cat > /etc/depmod.d/openvswitch.conf << EOF
override openvswitch * extra
override vport-* * extra
EOF

depmod -a
cp debian/openvswitch-switch.init /etc/init.d/openvswitch-switch
/etc/init.d/openvswitch-switch force-reload-kmod
```

[illegible] Kube-OVN[illegible] OVS [illegible]

[illegible] STT [illegible]

> **Warning**
>
> OpenVswitch [illegible] 3.6 [illegible] STT [illegible] Tunnel [illegible]

[illegible] Geneve [illegible] Vxlan [illegible] UDP [illegible] UDP [illegible] TCP [illegible] TCP [illegible] offload [illegible] TCP [illegible] CPU [illegible] TCP [illegible]

STT [illegible] TCP [illegible] TCP [illegible] TCP [illegible] TCP [illegible] TCP [illegible]

STT [illegible] OVS [illegible]OVS [illegible]

2025 Kube-OVN Team

STT ⬚⬚⬚⬚⬚

```
kubectl set env daemonset/ovs-ovn -n kube-system TUNNEL_TYPE=stt

kubectl delete pod -n kube-system -lapp=ovs
```

PDF Slack Support

2025⬚10⬚28⬚

2022⬚5⬚24⬚

GitHub 

7.2.4 ⬚⬚

7.3 □□□□ FastPath □□

经过数据平面的性能 Profile，`Netfilter` 在容器内和宿主机上的相关处理消耗了 20% 左右的 CPU 资源，FastPath 模块可以绕过 `Netfilter` 从而降低 CPU 的消耗和延迟，并提升吞吐量。本文档将介绍如何手动编译 FastPath 模块。

7.3.1 下载相关内核模块□码

```
git clone --depth=1 https://github.com/kubeovn/kube-ovn.git
```

7.3.2 □□□赖

针对 RPM 系统（如 CentOS、RHEL）：

```
yum install -y kernel-devel-$(uname -r) gcc elfutils-libelf-devel
```

针对 DEB 系统（如 Ubuntu、Debian）：

```
apt install -y linux-headers-$(uname -r) build-essential
```

7.3.3 编译相关模块

针对 3.x 的内核：

```
cd kube-ovn/fastpath/3.x
make all
```

针对 4.x ~ 6.x 的内核：

```
cd kube-ovn/fastpath/4.x-6.x
make all
```

7.3.4 安□内核模块

将 `kube_ovn_fastpath.ko` 复制到每个需要性能优化的节点，并执行以下命令安装：

```
insmod kube_ovn_fastpath.ko
```

使用 `dmesg` 确认安装成功：

```
# dmesg
[619631.323788] init_module,kube_ovn_fastpath_local_out
[619631.323798] init_module,kube_ovn_fastpath_post_routing
[619631.323800] init_module,kube_ovn_fastpath_pre_routing
[619631.323801] init_module,kube_ovn_fastpath_local_in
```

如需卸载模块，可使用以下命令：

```
rmmod kube_ovn_fastpath.ko
```

> 该模块在机器重启后不会自动加载，如需自动加载请根据系统配置编写相应自启动脚本。

PDF Slack Support

2026⬚2⬚11⬚

2022⬚5⬚24⬚



GitHub 



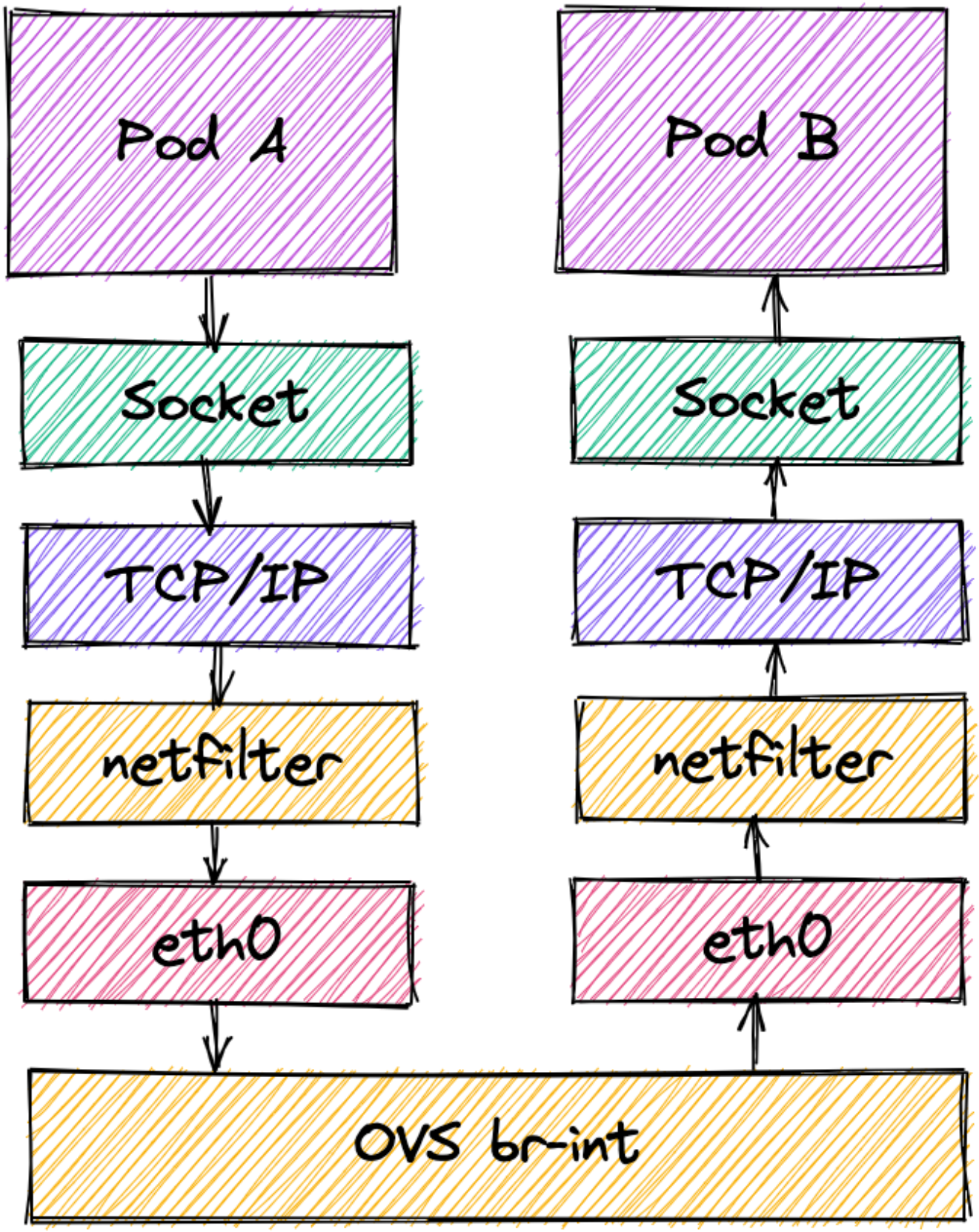
7.3.5 ⬚⬚

7.4 使用 eBPF 加速节点内 TCP 通信

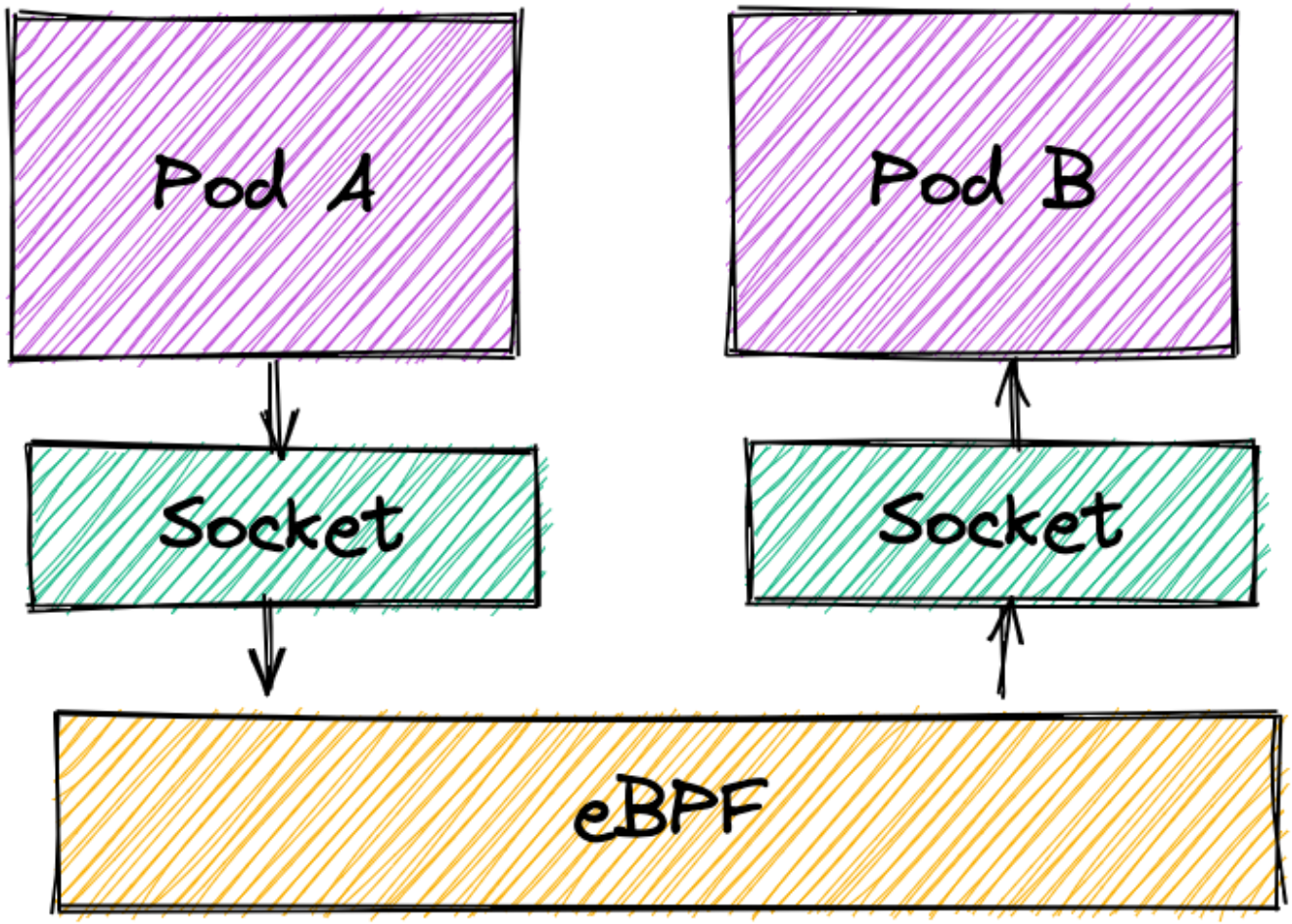
在一些边缘和 5G 的场景下，同节点内的 Pod 之间会进行大量的 TCP 通信，通过使用 Intel 开源的 istio-tcpip-bypass 项目，Pod 可以借助 eBPF 的能力绕过主机的 TCP/IP 协议栈，直接进行 socket 通信，从而大幅降低延迟并提升吞吐量。

7.4.1 [illegible]

在当前的实现下，同主机的两个 Pod 进行 TCP 通信需要经过大量的网络栈，包括 TCP/IP, netfilter, OVS 等如下图所示:



istio-tcpip-bypass 插件可以自动分析并识别同主机内的 TCP 通信，绕过复杂的内核栈，直接进行 socket 间的数据传输，从而降低网络栈处理开销，如下图所示：



▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ TCP ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ Service Mesh ▒▒▒▒▒▒▒▒▒▒▒▒▒ Service Mesh ▒▒▒▒▒▒▒

▒▒▒▒▒▒▒▒▒▒▒▒▒ Tanzu Service Mesh Acceleration using eBPF▒

7.4.2 ▒▒▒▒

eBPF ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ Ubuntu 20.04 ▒ Linux 5.4.0-74-generic ▒▒▒▒▒▒▒▒▒▒

7.4.3 ▒▒▒▒

▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ Pod▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ `nodeSelector` ▒

```
# kubectl create deployment perf --image=kubeovn/perf:dev --replicas=2
deployment.apps/perf created
# kubectl get pod -o wide
NAME                    READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
perf-7697bc6ddf-b2cpv   1/1     Running   0          28s   100.64.0.3   sealos   <none>           <none>
perf-7697bc6ddf-p2xpt   1/1     Running   0          28s   100.64.0.2   sealos   <none>           <none>
```

▒▒▒▒▒▒ Pod ▒▒ qperf server▒▒▒▒▒▒ Pod ▒▒▒ qperf client ▒▒▒▒▒▒▒▒

```
# kubectl exec -it perf-7697bc6ddf-b2cpv sh
/ # qperf

# kubectl exec -it perf-7697bc6ddf-p2xpt sh
/ # qperf -t 60 100.64.0.3 -ub -oo msg_size:1:16K:*4 -vu tcp_lat tcp_bw
```

▒▒ istio-tcpip-bypass ▒▒▒

```
kubectl apply -f https://raw.githubusercontent.com/intel/istio-tcpip-bypass/main/bypass-tcpip-daemonset.yaml
```



再次进入 perf client 容器进行性能测试:

```
# kubectl exec -it perf-7697bc6ddf-p2xpt sh
/ # qperf -t 60 100.64.0.3 -ub -oo msg_size:1:16K:*4 -vu tcp_lat tcp_bw
```

7.4.4 测试结果

根据测试结果 TCP 延迟在不同数据包大小的情况下会有 40% ~ 60% 的延迟下降，在数据包大于 1024 字节时吞吐量会有 40% ~ 80% 提升。

Packet Size (byte)	eBPF tcp_lat (us)	Default tcp_lat (us)	eBPF tcp_bw (Mb/s)	Default tcp_bw(Mb/s)
1	20.2	44.5	1.36	4.27
4	20.2	48.7	5.48	16.7
16	19.6	41.6	21.7	63.5
64	18.8	41.3	96.8	201
256	19.2	36	395	539
1024	18.3	42.4	1360	846
4096	16.5	62.6	4460	2430
16384	20.2	58.8	9600	6900

> 在测试的硬件环境下，数据包小于 512 字节时，使用 eBPF 优化吞吐量指标会低于默认配置下的吞吐量。该情况可能和默认配置下网卡开启 TCP 聚合优化相关。如果应用场景对小包吞吐量敏感，需要在相应环境下进行测试判断是否开启 eBPF 优化。我们也会后续对 eBPF TCP 小包场景的吞吐量进行优化。

7.4.5 参考资[illegible]

1. istio-tcpip-bypass
2. Deep Dive TCP/IP Bypass with eBPF in Service Mesh
3. Tanzu Service Mesh Acceleration using eBPF

PDF Slack Support

2025年9月10日

2022年9月27日

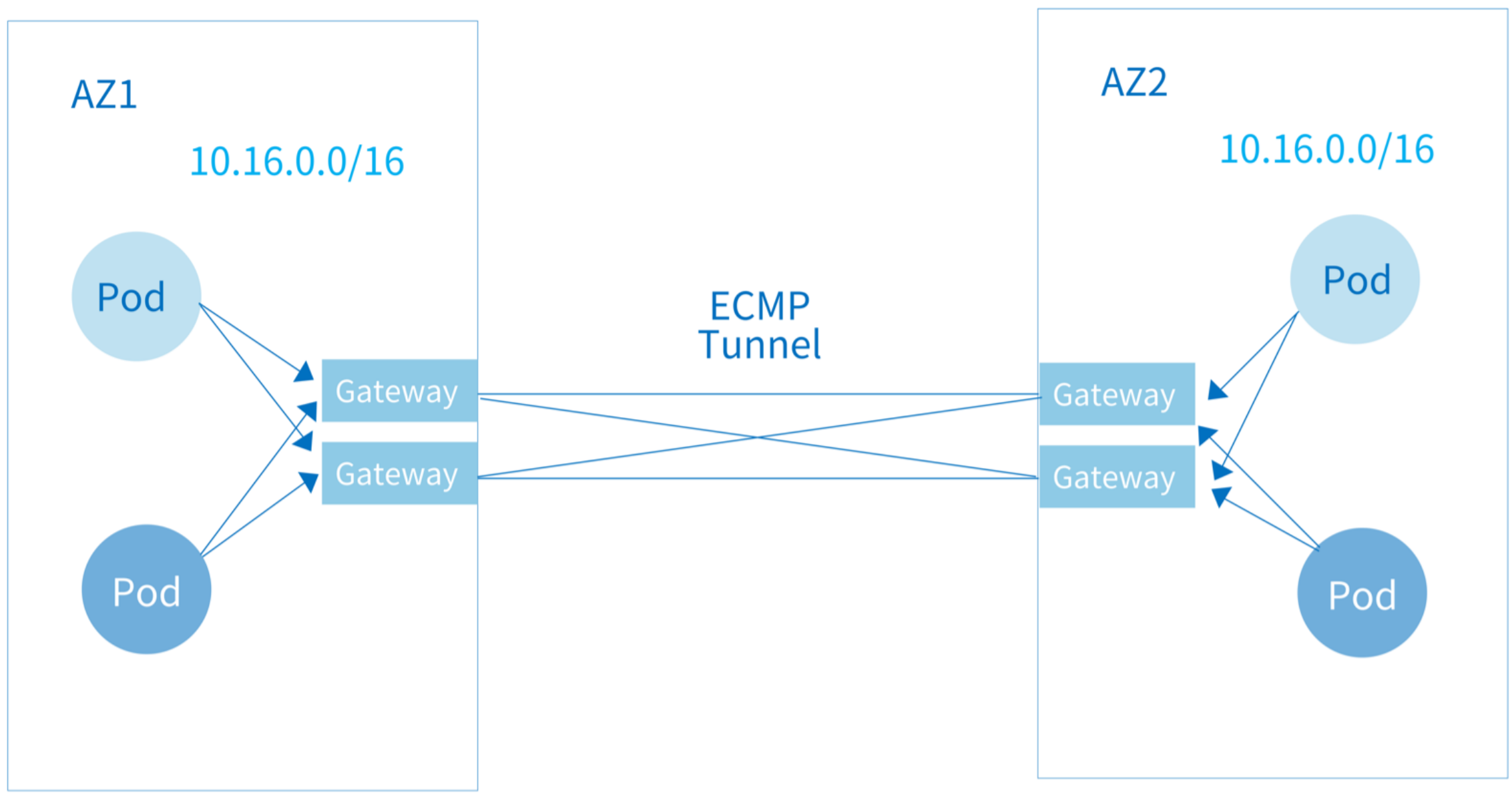
GitHub

7.4.6 评论

7.5 ▯▯ OVN-IC ▯▯▯▯▯▯▯

Kube-OVN ▯▯▯▯ OVN-IC ▯▯▯ Kubernetes ▯▯ Pod ▯▯▯▯▯▯▯▯▯▯▯▯▯ Pod ▯▯▯▯ Pod IP ▯▯▯▯▯▯▯ Kube-OVN ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ IP ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

> ▯▯▯▯▯▯▯▯▯▯ Overlay ▯▯▯▯▯Underlay ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯



Limitation

OVN-IC ▯▯▯▯▯▯▯ Pod IP ▯▯▯▯▯▯▯▯▯▯ Service▯DNS ▯ NetworkPolicy ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ Istio ▯▯▯▯▯▯▯▯▯▯▯▯

7.5.1 ▯▯▯▯

1. 1.11.16 ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `install.sh` ▯▯▯▯▯▯▯▯

   ```
   ENABLE_IC=true
   ```

   ▯▯▯▯▯▯▯▯▯▯▯▯▯▯ deployment ovn-ic-controller▯

2. ▯▯▯▯▯▯▯▯▯▯▯▯▯ CIDR ▯▯
3. ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `kube-ovn-controller` ▯▯ IP ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
4. ▯▯▯▯▯▯▯▯▯▯▯▯ IP ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
5. ▯▯▯▯▯▯▯ VPC ▯▯▯▯▯▯▯▯ VPC ▯▯▯▯▯▯▯▯▯

7.5.2 ▯▯▯▯▯ OVN-IC ▯▯▯

▯▯▯▯▯▯▯ 1

▯▯▯▯▯▯ 1▯Kube-OVN v1.11.16 ▯▯▯▯▯

▯▯▯▯▯▯ "▯▯▯" ▯▯ "▯▯▯▯▯▯" ▯▯▯▯▯▯▯▯ Deployment ▯▯▯▯▯▯ master ▯▯▯▯▯▯ master ▯▯▯ 1▯▯▯▯▯▯▯▯master ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

2025 Kube-OVN Team

[illegible] `install-ovn-ic.sh` [illegible]

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/dist/images/install-ic-server.sh
```

[illegible] `TS_NUM` [illegible] ECMP Path [illegible]

```
sed 's/VERSION=.*/VERSION=v1.17.0/' dist/images/install-ic-server.sh | TS_NUM=3 bash
```

[illegible]

```
deployment.apps/ovn-ic-server created
Waiting for deployment spec update to be observed...
Waiting for deployment "ovn-ic-server" rollout to finish: 0 out of 3 new replicas have been updated...
Waiting for deployment "ovn-ic-server" rollout to finish: 0 of 3 updated replicas are available...
Waiting for deployment "ovn-ic-server" rollout to finish: 1 of 3 updated replicas are available...
Waiting for deployment "ovn-ic-server" rollout to finish: 2 of 3 updated replicas are available...
deployment "ovn-ic-server" successfully rolled out
OVN IC Server installed Successfully
```

[illegible] `kubectl ko icsbctl show` [illegible]

```
kubectl ko icsbctl show
availability-zone az0
    gateway 059b5c54-c540-4d77-b009-02d65f181a02
        hostname: kube-ovn-worker
        type: geneve
            ip: 172.18.0.3
        port ts-az0
            transit switch: ts
            address: ["00:00:00:B4:8E:BE 169.254.100.97/24"]
    gateway 74ee4b9a-ba48-4a07-861e-1a8e4b9f905f
        hostname: kube-ovn-worker2
        type: geneve
            ip: 172.18.0.2
        port ts1-az0
            transit switch: ts1
            address: ["00:00:00:19:2E:F7 169.254.101.90/24"]
    gateway 7e2428b6-344c-4dd5-a0d5-972c1ccec581
        hostname: kube-ovn-control-plane
        type: geneve
            ip: 172.18.0.4
        port ts2-az0
            transit switch: ts2
            address: ["00:00:00:EA:32:BA 169.254.102.103/24"]
availability-zone az1
    gateway 034da7cb-3826-4318-81ce-6a877a9bf285
        hostname: kube-ovn1-worker
        type: geneve
            ip: 172.18.0.6
        port ts-az1
            transit switch: ts
            address: ["00:00:00:25:3A:B9 169.254.100.51/24"]
    gateway 2531a683-283e-4fb8-a619-bdbcb33539b8
        hostname: kube-ovn1-worker2
        type: geneve
            ip: 172.18.0.5
        port ts1-az1
            transit switch: ts1
            address: ["00:00:00:52:87:F4 169.254.101.118/24"]
    gateway b0efb0be-e5a7-4323-ad4b-317637a757c4
        hostname: kube-ovn1-control-plane
        type: geneve
            ip: 172.18.0.8
        port ts2-az1
            transit switch: ts2
            address: ["00:00:00:F6:93:1A 169.254.102.17/24"]
```

[illegible] **2**

[illegible] `kube-ovn-controller` [illegible] IP [illegible] `OVN-IC` [illegible]

[illegible] `docker` [illegible] `OVN-IC` [illegible]

```
docker run --name=ovn-ic-db -d --env "ENABLE_OVN_LEADER_CHECK="false"" --network=host --privileged  -v /etc/ovn/:/etc/ovn -v /var/run/ovn:/var/run/ovn -v /
var/log/ovn:/var/log/ovn kubeovn/kube-ovn:v1.17.0 bash start-ic-db.sh
```

[illegible] `containerd` [illegible] `docker` [illegible]

2025 Kube-OVN Team

```
ctr -n k8s.io run -d --env "ENABLE_OVN_LEADER_CHECK="false"" --net-host --privileged --mount="type=bind,src=/etc/ovn/,dst=/etc/ovn,options=rbind:rw" --
mount="type=bind,src=/var/run/ovn,dst=/var/run/ovn,options=rbind:rw" --mount="type=bind,src=/var/log/ovn,dst=/var/log/ovn,options=rbind:rw" docker.io/kubeovn/
kube-ovn:v1.17.0 ovn-ic-db bash start-ic-db.sh
```

7.5.3 ■■■■■■■

■■■■■■■■■■■■■■■■■■■■■■ VPC ■ Subnet ■ CIDR ■■■■■■ `OVN-IC` ■■■■■■■■■■■■■ Subnet CIDR ■■■■■■■

■ `kube-system` Namespace ■■■ `ovn-ic-config` ConfigMap■

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ovn-ic-config
  namespace: kube-system
data:
  enable-ic: "true"
  az-name: "az1"
  ic-db-host: "192.168.65.3"
  ic-nb-port: "6645"
  ic-sb-port: "6646"
  gw-nodes: "az1-gw"
  auto-route: "true"
```

- `enable-ic`：■■■■■■■■■■
- `az-name`：■■■■■■■■■■■■■■■■■■■■■■■■■■■■
- `ic-db-host`：■■ `OVN-IC` ■■■■■■■■■■
- `ic-nb-port`：`OVN-IC` ■■■■■■■■■■■■■ 6645■
- `ic-sb-port`：`OVN-IC` ■■■■■■■■■■■■■ 6646■
- `gw-nodes`：■■■■■■■■■■■■■■■■■■■■■■■■■■■■
- `auto-route`：■■■■■■■■■■■■■■■■■

■■■ ■■■■■■■■■■■■■ `ovn-ic-config` ■■ ConfigMap ■■■■■■■■■■■■■■■■■■■■■■ ConfigMap■■■■■■■■■ ConfigMap■

■ `ovn-ic` ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ `ts` ■

```
# ovn-ic-sbctl show
availability-zone az1
    gateway deee03e0-af16-4f45-91e9-b50c3960f809
        hostname: az1-gw
        type: geneve
            ip: 192.168.42.145
        port ts-az1
            transit switch: ts
            address: ["00:00:00:50:AC:8C 169.254.100.45/24"]
availability-zone az2
    gateway e94cc831-8143-40e3-a478-90352773327b
        hostname: az2-gw
        type: geneve
            ip: 192.168.42.149
        port ts-az2
            transit switch: ts
            address: ["00:00:00:07:4A:59 169.254.100.63/24"]
```

■■■■■■■■■■■■■■■■■■■■■■■■■■■■

```
# kubectl ko nbctl lr-route-list ovn-cluster
IPv4 Routes
                10.42.1.1            169.254.100.45 dst-ip (learned)
                10.42.1.3                100.64.0.2 dst-ip
                10.16.0.2                100.64.0.2 src-ip
                10.16.0.3                100.64.0.2 src-ip
                10.16.0.4                100.64.0.2 src-ip
                10.16.0.6                100.64.0.2 src-ip
             10.17.0.0/16            169.254.100.45 dst-ip (learned)
            100.65.0.0/16            169.254.100.45 dst-ip (learned)
```

■■■■■■■■■■■ 1 ■■■■■ Pod ■■■ `ping` ■■ 2 ■■■■■ Pod IP ■■■■■■■■■■

■■■■■■■■■■■■■■■■■■■■■■■■■■■■ Subnet ■■ `disableInterConnection` ■■■■■■■■■

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
```


2025 Kube-OVN Team

```yaml
  name: no-advertise
spec:
  cidrBlock: 10.199.0.0/16
  disableInterConnection: true
```

7.5.4 □□□□□□

□□□□□□□□ CIDR □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□ `kube-system` Namespace □□□ `ovn-ic-config` ConfigMap□□□ `auto-route` □□□ `false` □

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: ovn-ic-config
  namespace: kube-system
data:
  enable-ic: "true"
  az-name: "az1"
  ic-db-host: "192.168.65.3"
  ic-nb-port: "6645"
  ic-sb-port: "6646"
  gw-nodes: "az1-gw"
  auto-route: "false"
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
[root@az1 ~]# kubectl ko nbctl show
switch a391d3a1-14a0-4841-9836-4bd930c447fb (ts)
    port ts-az1
        type: router
        router-port: az1-ts
    port ts-az2
        type: remote
        addresses: ["00:00:00:4B:E2:9F 169.254.100.31/24"]

[root@az2 ~]# kubectl ko nbctl show
switch da6138b8-de81-4908-abf9-b2224ec4edf3 (ts)
    port ts-az2
        type: router
        router-port: az2-ts
    port ts-az1
        type: remote
        addresses: ["00:00:00:FB:2A:F7 169.254.100.79/24"]
```

□□□□□□□□ `az1` □ □□ `az2` □□□□□□ `169.254.100.31` □ `az2` □ `az1` □□□□□□ `169.254.100.79` □

□□□□□□□□□□□□□□□□ `az1` □□□□ CIDR □ `10.16.0.0/24` □□□ `az2` □□□□ CIDR □ `10.17.0.0/24` □

□□□ `az1` □□□□□ `az2` □□□□

```
kubectl ko nbctl lr-route-add ovn-cluster 10.17.0.0/24 169.254.100.31
```

□□□ `az2` □□□□□ `az1` □□□□

```
kubectl ko nbctl lr-route-add ovn-cluster 10.16.0.0/24 169.254.100.79
```

7.5.5 □□□ OVN-IC □□□□□

□□□□□□□ 1

□□□□□□ 1□Kube-OVN v1.11.16 □□□□□

□□□[□□□□□□□ 1]

□□□□□□□ 2

`OVN-IC` □□□□□□□□□ Raft □□□□□□□□□□□□□□□□□□□□□ 3 □□□□

□□□□□□□□□□□ `OVN-IC` □□□□ leader□

□□ `docker` □□□□□□□□□□□□□□□□

2025 Kube-OVN Team

```
docker run --name=ovn-ic-db -d --env "ENABLE_OVN_LEADER_CHECK="false"" --network=host --privileged -v /etc/ovn/:/etc/ovn -v /var/run/ovn:/var/run/ovn -v /var/
log/ovn:/var/log/ovn -e LOCAL_IP="192.168.65.3"  -e NODE_IPS="192.168.65.3,192.168.65.2,192.168.65.1"   kubeovn/kube-ovn:v1.17.0 bash start-ic-db.sh
```

 `containerd`

```
ctr -n k8s.io run -d --env "ENABLE_OVN_LEADER_CHECK="false"" --net-host --privileged --mount="type=bind,src=/etc/ovn/,dst=/etc/ovn,options=rbind:rw" --
mount="type=bind,src=/var/run/ovn,dst=/var/run/ovn,options=rbind:rw" --mount="type=bind,src=/var/log/ovn,dst=/var/log/ovn,options=rbind:rw"  --
env="NODE_IPS="192.168.65.3,192.168.65.2,192.168.65.1"" --env="LOCAL_IP="192.168.65.3"" docker.io/kubeovn/kube-ovn:v1.17.0 ovn-ic-db bash start-ic-db.sh
```

- `LOCAL_IP` IP
- `NODE_IPS` `OVN-IC` IP

 `OVN-IC` follower

 `docker`

```
docker run --name=ovn-ic-db -d --network=host --privileged -v /etc/ovn/:/etc/ovn -v /var/run/ovn:/var/run/ovn -v /var/log/ovn:/var/log/ovn -e
LOCAL_IP="192.168.65.2"  -e NODE_IPS="192.168.65.3,192.168.65.2,192.168.65.1" -e LEADER_IP="192.168.65.3"  kubeovn/kube-ovn:v1.17.0 bash start-ic-db.sh
```

 `containerd`

```
ctr -n k8s.io run -d --net-host --privileged --mount="type=bind,src=/etc/ovn/,dst=/etc/ovn,options=rbind:rw" --mount="type=bind,src=/var/run/ovn,dst=/var/run/
ovn,options=rbind:rw" --mount="type=bind,src=/var/log/ovn,dst=/var/log/ovn,options=rbind:rw"  --env="NODE_IPS="192.168.65.3,192.168.65.2,192.168.65.1"" --
env="LOCAL_IP="192.168.65.2"" --env="LEADER_IP="192.168.65.3"" docker.io/kubeovn/kube-ovn:v1.17.0 ovn-ic-db bash start-ic-db.sh
```

- `LOCAL_IP` IP
- `NODE_IPS` `OVN-IC` IP
- `LEADER_IP` : `OVN-IC` leader IP

 `ovn-ic-config` `OVN-IC`

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: ovn-ic-config
  namespace: kube-system
data:
  enable-ic: "true"
  az-name: "az1"
  ic-db-host: "192.168.65.3,192.168.65.2,192.168.65.1"
  ic-nb-port: "6645"
  ic-sb-port: "6646"
  gw-nodes: "az1-gw"
  auto-route: "true"
```

7.5.6 ECMP

1

ECMPECMP path 3 ECMP path

```
kubectl edit deployment ovn-ic-server -n kube-system
```

'TS_NUM' `TS_NUM` ECMP Path

7.5.7

Configmap

```
kubectl -n kube-system delete cm ovn-ic-config
```

 `ts`

```
kubectl ko nbctl ls-del ts
```

[illegible]

7.5.8 az-name

[illegible] `kubectl edit` [illegible] `ovn-ic-config` configmap `az-name` [illegible] ovn-cni pod [illegible] 10 [illegible]

```
ovn-appctl -t ovn-controller inc-engine/recompute
```

7.5.9 [illegible]

[illegible] `ovn-ic-config` Configmap

```
kubectl -n kube-system delete cm ovn-ic-config
```

[illegible] `ts` [illegible]

```
kubectl ko nbctl ls-del ts
```

[illegible] OVN-IC [illegible]

[illegible] `docker` [illegible]

```
docker stop ovn-ic-db
docker rm ovn-ic-db
```

[illegible] `containerd` [illegible]

```
ctr -n k8s.io task kill ovn-ic-db
ctr -n k8s.io containers rm ovn-ic-db
```

[illegible] deployment `ovn-ic-server` [illegible]

```
kubectl delete deployment ovn-ic-server -n kube-system
```

[illegible] master [illegible] DB[illegible]

```
rm -f /etc/origin/ovn/ovn_ic_nb_db.db
rm -f /etc/origin/ovn/ovn_ic_sb_db.db
```

PDF Slack Support

2025[illegible]6[illegible]19[illegible]

2022[illegible]5[illegible]24[illegible]

GitHub 

7.5.10 [illegible]

7.6 使用 Submariner 进行多集群互联

Submariner 作为可以打通多个 Kubernetes 集群 Pod 和 Service 网络的开源网络组件，能够帮助 Kube-OVN 实现多集群互联。

相比通过 OVN-IC 打通多集群网络的方式，Submariner 可以打通 Kube-OVN 和非 Kube-OVN 的集群网络，并能提供 Service 的跨集群能力。但是 Submariner 目前只能实现默认子网的打通，无法实现多子网选择性打通。

7.6.1 [illegible]

- 两个集群的 Service CIDR 和默认子网的 CIDR 不能重叠。

7.6.2 [illegible] Submariner

> **Note**
>
> Submariner 自 v0.22.0 后使用 nftables 作为默认的转发后端，会和 Kube-OVN 规则产生冲突，请先在所有需要集群互联的集群执行下列命令，再安装 Submariner：
>
> ```
> kubectl create namespace submariner-operator
> kubectl create configmap submariner-global --namespace=submariner-operator --from-literal=use-nftables=false
> ```

下载 `subctl` 二进制文件，并部署到相应路径：

```
curl -Ls https://get.submariner.io | bash
export PATH=$PATH:~/.local/bin
echo export PATH=\$PATH:~/.local/bin >> ~/.profile
```

切换 `kubeconfig` 至希望部署 `submariner-broker` 的集群进行部署：

```
subctl deploy-broker
```

在本文档中 `cluster0` 的默认子网 CIDR 为 `10.16.0.0/16`，join 子网 CIDR 为 `100.64.0.0/16`，`cluster1` 的默认子网 CIDR 为 `11.16.0.0/16`，join 子网 CIDR 为 `100.68.0.0/16`。

切换 `kubeconfig` 至 `cluster0` 注册集群至 broker，并注册网关节点：

```
subctl  join broker-info.subm --clusterid  cluster0 --clustercidr 100.64.0.0/16,10.16.0.0/16  --natt=false --cable-driver vxlan --health-check=false
kubectl label nodes cluster0 submariner.io/gateway=true
```

切换 `kubeconfig` 至 `cluster1` 注册集群至 broker，并注册网关节点：

```
subctl  join broker-info.subm --clusterid  cluster1 --clustercidr 100.68.0.0/16,11.16.0.0/16  --natt=false --cable-driver vxlan --health-check=false
kubectl label nodes cluster1 submariner.io/gateway=true
```

如果执行 `join` 命令之后没有新的 `gateway, routeagent` pod 出现的话，请为 `submariner-operator` 这个 `clusterrole` 增加以下权限：

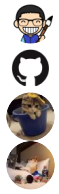
```
- apiGroups:
  - "apps"
  resources:
  - daemonsets
  verbs:
  - create
  - get
  - list
  - watch
  - update
```

对于多节点的集群，需要将默认的 `subnet ovn-default` 的网关配置改为 `centralized`，并显式指定 `gatewayNode`（集中式网关必填，否则子网无法生效）：

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: ovn-default
spec:
  gatewayType: centralized
  gatewayNode: "node1,node2"   #   submariner     gateway
```

为 submariner 配置的 `gateway` 节点需要和 `gatewayNode` 字段中列出的节点完全相同。

2025 Kube-OVN Team



接下来可以在两个集群内分别启动 Pod 并尝试使用 IP 进行相互访问。

如果出现网络互通问题可通过 `subctl` 命令进行诊断：

```
subctl show all
subctl diagnose all
```

更多 Submariner 相关操作请查看 Submariner 用户手册。

PDF Slack Support

2026年4月28日

2022年5月24日

GitHub ## 7.6.3 ⬚⬚

7.7 Overlay

Underlay Pod IP

> VPC Overlay Pod IP IP

7.7.1

- ip_forward
- iptables forward Drop
- ct INVALID

7.7.2

natOutgoing false nat Pod IP

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: routed
spec:
  protocol: IPv4
  cidrBlock: 10.166.0.0/16
  default: false
  excludeIps:
  - 10.166.0.1
  gateway: 10.166.0.1
  gatewayType: distributed
  natOutgoing: false
```

Pod

Kubernetes

```
ip route add 10.166.0.0/16 via 192.168.2.10 dev eth0
```

10.166.0.0/16 192.168.2.10 Kubernetes

Keepalived VIP VIP

Subnet gatewayType centralized gatewayNode IP

```yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: routed
spec:
  protocol: IPv4
  cidrBlock: 10.166.0.0/16
  default: false
  excludeIps:
  - 10.166.0.1
  gateway: 10.166.0.1
  gatewayType: centralized
  gatewayNode: "node1"
  natOutgoing: false
```

nat VPC NAT

PDF Slack Support

2025 Kube-OVN Team

2023⊠12⊠25⊠

2022⊠7⊠6⊠

GitHub 

7.7.3 ⊠⊠

2025 Kube-OVN Team

7.8 Overlay [illegible]

[illegible] Overlay [illegible] tunnel [illegible]

- [illegible]
- [illegible]
- [illegible]

7.8.1 [illegible]

[illegible]Kube-OVN [illegible] Kubernetes Node IP [illegible]

[illegible]

```
IFACE=eth1
```

[illegible] `ens[a-z0-9]*,eth[a-z0-9]*` [illegible]

[illegible] `kube-ovn-cni` DaemonSet [illegible]

```
args:
- --iface=eth1
```

[illegible] annotation `ovn.kubernetes.io/tunnel_interface` [illegible] annotation [illegible] `iface` [illegible] annotation[illegible]

```
kubectl annotate node no1 ovn.kubernetes.io/tunnel_interface=ethx
```

7.8.2 [illegible]

[illegible]Kube-OVN [illegible] IP [illegible] tunnel [illegible]

[illegible]

- [illegible] IP [illegible]
- [illegible]
- [illegible] Overlay [illegible]

[illegible]

1. [illegible] Node [illegible] IP[illegible]
2. [illegible] Subnet [illegible] `nodeNetwork` [illegible]
3. kube-ovn-daemon [illegible] IP [illegible] OVS[illegible]
4. [illegible] Pod [illegible] `nodeNetwork` [illegible] Pod [illegible] OVS [illegible] IP[illegible]

[illegible]

[illegible] `ovn.kubernetes.io/node_networks` [illegible] JSON [illegible]key [illegible]value [illegible] IP[illegible]

```
kubectl annotate node <node-name> ovn.kubernetes.io/node_networks='{"storage": "192.168.100.10", "app": "172.16.0.10"}'
```

[illegible]

- `storage` [illegible] IP `192.168.100.10` [illegible] IP [illegible]
- `app` [illegible] IP `172.16.0.10` [illegible] IP [illegible]

[illegible] IP[illegible]

[illegible]

[illegible] `spec.nodeNetwork` [illegible]

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: storage-subnet
spec:
  protocol: IPv4
  cidrBlock: 10.100.0.0/16
  gateway: 10.100.0.1
  nodeNetwork: storage
```

[illegible] `storage-subnet` [illegible] Pod [illegible] `storage` [illegible] tunnel [illegible]

[illegible] `nodeNetwork` [illegible] IP[illegible] `IFACE` [illegible] IP[illegible]

[illegible]

[illegible] OVS [illegible] IP [illegible]

[illegible] OVS [illegible] IP[illegible]

```
ovs-vsctl get open . external-ids:ovn-encap-ip
```

[illegible] IP[illegible]

```
"192.168.1.10,192.168.100.10,172.16.0.10"
```

[illegible] IP[illegible]

```
ovs-vsctl get open . external-ids:ovn-encap-ip-default
```

[illegible] POD [illegible] IP

[illegible] Pod [illegible] Pod [illegible] OVS [illegible] IP [illegible]

```
ovs-vsctl --columns=external_ids find interface external-ids:iface-id="<pod-name>.<namespace>"
```

[illegible] `nodeNetwork` [illegible] `encap-ip` [illegible]

```
external_ids        : {encap-ip="192.168.100.10", iface-id="test-pod.default", ...}
```

[illegible]

[illegible]

[illegible]

1. [illegible]

[illegible]

- `eth0` [illegible]IP [illegible] `192.168.1.10` [illegible] `192.168.1.11`
- `eth1` [illegible]IP [illegible] `10.10.10.10` [illegible] `10.10.10.11`

```
kubectl annotate node node1 ovn.kubernetes.io/node_networks='{"storage": "10.10.10.10"}'
kubectl annotate node node2 ovn.kubernetes.io/node_networks='{"storage": "10.10.10.11"}'
```

2. [illegible]

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: storage-net
spec:
  protocol: IPv4
  cidrBlock: 10.200.0.0/16
  gateway: 10.200.0.1
  nodeNetwork: storage
```

```
  namespaces:
  - storage-namespace
```

3. ⬚⬚⬚⬚⬚⬚⬚⬚ POD

```
apiVersion: v1
kind: Pod
metadata:
  name: storage-pod
  namespace: storage-namespace
spec:
  containers:
  - name: app
    image: docker.io/library/nginx:alpine
```

该 Pod 的网络流量将通过存储网卡（`eth1`）进行转发。

注意事项

1. 确保同一网络平面内的所有节点之间网络互通。
2. 节点注解中配置的 IP 必须是该节点上实际存在的有效 IP 地址。
3. 该功能仅适用于 Overlay 类型子网，Underlay 子网不支持此配置。如需配置 Underlay 网络，请参考 Underlay 网络安装。
4. 如果子网未配置 `nodeNetwork`，或配置的网络名称在节点上不存在，将使用默认封装 IP。
5. 如果 Pod 调度到的节点上不存在子网指定的 `nodeNetwork` 对应的注解，Pod 将会运行失败。请确保所有可能调度到的节点都配置了相应的网络注解。
6. 在进行节点增加或节点网络调整时，需要及时更新对应节点的 `ovn.kubernetes.io/node_networks` 注解，避免 Pod 运行失败。

PDF Slack Support

2025年11月28日

2025年11月28日

GitHub

7.8.3 ⬚⬚

7.9 BGP □□

Kube-OVN □□□ Pods□Subnets□Services □ EIPs □ IP □□□□ BGP □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ IP □□□ □□□□□□□□□□□□□□□□□□□□ `kube-ovn-speaker` □□□□□□□□ Pod □ Subnet □□□□□ annotation□

□□□□ EIP □□□ BGP□□□□□□□□□□□ VPC NAT Gateway□□□□□□□□□□□□□□ EIPs□

7.9.1 □□ kube-ovn-speaker

`kube-ovn-speaker` □□□ GoBGP □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□ `kube-ovn-speaker` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
kubectl label nodes speaker-node-1 ovn.kubernetes.io/bgp=true
kubectl label nodes speaker-node-2 ovn.kubernetes.io/bgp=true
```

> □□□□□ kube-ovn-speaker □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ ECMP□

□□□□ yaml:

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/yamls/speaker.yaml
```

□□ yaml □□□□□□

□□□□□□□□□□□□

```
- --neighbor-address=10.32.32.254
- --neighbor-ipv6-address=2409:AB00:AB00:2000::AFB:8AFE
- --neighbor-as=65030
- --cluster-as=65000
#        IP
# - --allowed-source-addresses=10.32.32.2,10.32.32.3,10.32.32.4,10.32.32.5
#
# - --next-hops=192.0.2.100
# - --next-hops=2001:db8::100
```

□□□□□□□□□□□□

```
- --neighbor-address=10.32.32.252,10.32.32.253
- --neighbor-ipv6-address=2409:AB00:AB00:2000::AFB:8AFC,2409:AB00:AB00:2000::AFB:8AFD
- --neighbor-as=65030
- --cluster-as=65000
#        IP
# - --allowed-source-addresses=10.32.32.2,10.32.32.3,10.32.32.4,10.32.32.5
#
# - --next-hops=192.0.2.100
# - --next-hops=2001:db8::100
```

- `neighbor-address` : BGP Peer □□□□□□□□□□□□□□□□□
- `neighbor-as` : BGP Peer □ AS □□
- `cluster-as` : □□□□□ AS □□
- `allowed-source-addresses` : Speaker □□□□□□□□ IP □□□□□□□□□□□□□□□□□□□Speaker □□□ `ip route get` □□□□ BGP Peer □□□□□□□□□□□□ IP □□□□□□□□□□□□□□□□□□ Speaker □□□□□□□□□□□□□□ ECMP □□□□□□□□□ IP □□□□□□□□□ IP □□□□□□□□□□□□□□
- `next-hops` : □□□□□□□□□□□□□□□□□□□□□□ IPv4 □ IPv6 □□□□□□□□□ `allowed-source-addresses` □□□□□□ `node-ips` □□□□□□□□□□□□□□ BGP □□□□□□□□□□□□□□□ OVN LRP EIP□□□

□□ yaml:

```
kubectl apply -f speaker.yaml
```

□□□□□ YAML □□

□□□□□□□□□□□□ kube-ovn-speaker □□□□ YAML □□□□□□□□□ `--config=/etc/kube-ovn/$(NODE_NAME).yaml` □□□□□□□□□□□□□□□□ `KUBE_OVN_BGP_SPEAKER_CONFIG_FILE` □□□□□□□□□□ `--config` □□□□□

2025 Kube-OVN Team

可以将每个节点的配置文件作为 ConfigMap 的独立 key 挂载。key 名称就是挂载后的文件名，格式必须为 `<spec.nodeName>.yaml`，去掉 `.yaml` 后缀后才与对应 Pod 的 `spec.nodeName` 匹配；下面的 `speaker-node-1` 和 `speaker-node-2` 是节点名示例：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-ovn-bgp-speaker-config
  namespace: kube-system
data:
  speaker-node-1.yaml: |
    cluster-as: 65000
    neighbor-address:
      - 10.32.32.254
    neighbor-as: 65030
    router-id: 10.32.32.2
  speaker-node-2.yaml: |
    cluster-as: 65001
    neighbor-address:
      - 10.32.32.254
    neighbor-as: 65030
    router-id: 10.32.32.3
```

使用 v2 Helm Chart 时，通过 `bgpSpeaker.configMap` 挂载 ConfigMap，并通过 `bgpSpeaker.extraEnv` 让每个 Pod 根据 Downward API 的 `NODE_NAME` 选择对应文件：

```
bgpSpeaker:
  enabled: true
  configMap: kube-ovn-bgp-speaker-config
  configMapMountPath: /etc/kube-ovn/
  extraEnv:
    - name: KUBE_OVN_BGP_SPEAKER_CONFIG_FILE
      value: /etc/kube-ovn/$(NODE_NAME).yaml
```

使用其他部署清单时，需要以相同方式将 ConfigMap 挂载到 speaker 容器，并按顺序定义 `NODE_NAME` 和配置文件环境变量：

```
env:
  - name: NODE_NAME
    valueFrom:
      fieldRef:
        fieldPath: spec.nodeName
  - name: KUBE_OVN_BGP_SPEAKER_CONFIG_FILE
    value: /etc/kube-ovn/$(NODE_NAME).yaml
```

也可以将 `--config` 直接设置为挂载后的实际绝对路径。配置文件只能包含一个 YAML 文档，未知字段会导致 speaker 启动失败。

文件中的字段名使用参数名去掉前缀后的 kebab-case 形式。常用字段格式如下：

- `neighbor-address` 和 `neighbor-ipv6-address`：IPv4 或 IPv6 地址列表。
- `allowed-source-addresses` 和 `allowed-source-ipv6-addresses`：源地址白名单列表。
- `cluster-as` 和 `neighbor-as`：AS 号，使用整数。
- `grpc-host`：IPv4 或 IPv6 地址字符串。
- `router-id`：IPv4 地址字符串；BGP router ID 不支持 IPv6。
- `node-ips`：地址映射，键为 `IPv4` 或 `IPv6`，值必须匹配对应地址族。
- `next-hops`：地址映射，键为 `IPv4` 或 `IPv6`，值必须匹配对应地址族。例如 `next-hops: {IPv4: 192.0.2.100, IPv6: 2001:db8::100}`。
- `holdtime`、`graceful-restart-time` 和 `graceful-restart-deferral-time`：持续时间字符串（如 `90s`、`5m`、`1h`），也可以使用整数秒。
- `announce-cluster-ip`、`graceful-restart`、`passivemode`、`extended-nexthop`、`nat-gw-mode`、`enable-metrics` 和 `enable-bfd`：布尔值。
- `bfd-min-tx`、`bfd-min-rx` 和 `bfd-detection-multiplier`：整数。

配置文件中明确设置的字段会覆盖同名 CLI 参数和默认值；未出现的字段继续使用 CLI 参数或默认值。未设置 `next-hops` 时保留原有的源地址/路由查询/ `node-ips` 选择逻辑。需要清空 CLI 列表时，可在文件中设置为空列表；需要清除 `next-hops` 时，可将该字段设为空 map，或将单个地址族设为 `null`，例如 `next-hops: {IPv4: null}`。

7.9.2 ⬚⬚ Pod/Subnet ⬚⬚

如需使用 BGP 对外发布路由，首先需要将对应 Subnet 的 `natOutgoing` 设置为 `false`，使得 Pod IP 可以直接进入底层网络。

增加 annotation 对外发布：

```
kubectl annotate pod sample ovn.kubernetes.io/bgp=true
kubectl annotate subnet ovn-default ovn.kubernetes.io/bgp=true
```

annotation

```
kubectl annotate pod sample ovn.kubernetes.io/bgp-
kubectl annotate subnet ovn-default ovn.kubernetes.io/bgp-
```

BGP

7.9.3 `ClusterIP` Service

Service ClusterIP `kube-ovn-speaker` `--announce-cluster-ip` `true` BGP

annotation

```
kubectl annotate service sample ovn.kubernetes.io/bgp=true
```

annotation

```
kubectl annotate service sample ovn.kubernetes.io/bgp-
```

7.9.4 EIPs

EIPs VPC NAT Gateway `VpcNatGateway` BGP BGP Sidecar

VPC NAT Gateway BGP BGP Speaker Sidecar `NetworkAttachmentDefinition` NAD VPC Subnet Sidecar Kubernetes API EIPs VPC CoreDNS NAD

`NetworkAttachmentDefinition` `Subnet` `provider` `{nadName}.{nadNamespace}.ovn`

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: api-ovn-nad
  namespace: default
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "kube-ovn",
      "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
      "provider": "api-ovn-nad.default.ovn"
    }'
---
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: vpc-apiserver-subnet
spec:
  protocol: IPv4
  cidrBlock: 100.100.100.0/24
  provider: api-ovn-nad.default.ovn
```

`ovn-vpc-nat-config` ConfigMap `apiNadProvider` BGP Speaker:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ovn-vpc-nat-config
  namespace: kube-system
data:
  apiNadProvider: api-ovn-nad.default.ovn             # What NetworkAttachmentDefinition provider to use so that the sidecar
                                                      # can access the K8S API, as it can't by default due to VPC segmentation
  bgpSpeakerImage: docker.io/kubeovn/kube-ovn:v1.13.0  # Sets the BGP speaker image used
  image: docker.io/kubeovn/vpc-nat-gateway:v1.13.0
```

`ovn-default` `provider`

```
provider: api-ovn-nad.default.ovn
```

VPC NAT Gateway BGP

```
kind: VpcNatGateway
apiVersion: kubeovn.io/v1
metadata:
  name: vpc-natgw
spec:
```

```
  vpc: vpc1
  subnet: net1
  lanIp: 10.0.1.10
  bgpSpeaker:
    enabled: true
    asn: 65500
    remoteAsn: 65000
    neighbors:
      - 100.127.4.161
      - fd:01::1
    enableGracefulRestart: true # Optional
    routerId: 1.1.1.1           # Optional
    holdTime: 1m                # Optional
    password: "password123"     # Optional
    extraArgs:                  # Optional, passed directly to the BGP speaker
      - -v5                     # Enables verbose debugging of the BGP speaker sidecar
  selector:
    - "kubernetes.io/os: linux"
  externalSubnets:
  - ovn-vpc-external-network # Network on which we'll speak BGP and receive/send traffic to the outside world
                             # BGP neighbors need to be on that network
```

... BGP ... EIP。

```
kubectl annotate eip sample ovn.kubernetes.io/bgp=true
```

7.9.5 ...

`kube-ovn-speaker` ...:

- **Cluster**: ... speaker ... Pod IPs/Subnet CIDRs... IP ... CIDR ... Pod... speaker ... Pod... Pod ... Subnet ...
- **Local**: ...Pod IPs ... Pod ... Cluster ... Pod ...

> ...: ... `Local` ... `kube-ovn-speaker` ... Pod ... speaker ...

... `Cluster` ... Pod/Subnet ... annotation `ovn.kubernetes.io/bgp` ...

- `ovn.kubernetes.io/bgp=cluster` ... `ovn.kubernetes.io/bgp=true` ... `Cluster` ...
- `ovn.kubernetes.io/bgp=local` ... `Local` ...

> ... Service ... `kube-proxy` ... `ClusterIP` ... Service ... `Cluster` ...

7.9.6 BGP ...

`kube-ovn-speaker` ... BGP ...

- `announce-cluster-ip` : ... Service ... `false` ...
- `auth-password` : BGP peer ...
- `holdtime` : BGP ... 90 ...
- `graceful-restart` : ... BGP Graceful Restart...
- `graceful-restart-time` : BGP Graceful restart time ... RFC4724 3...
- `graceful-restart-deferral-time` : BGP Graceful restart deferral time ... RFC4724 4.1...
- `passivemode` : Speaker ... passive ... peer...
- `ebgp-multihop` : ebgp ttl ... 1...
- `allowed-source-addresses` : ... IP ...Speaker ... `ip route get` ... BGP Peer ... IP ... ECMP ... IP ...
- `next-hops` : ... IPv4 ... IPv6 ... IP ... `node-ips` ... OVN LRP EIP ...
- `enable-bfd` : ... `kube-ovn-speaker` ... BFD... `false` ...“... `kube-ovn-speaker` ... BFD”...
- `bfd-min-tx` : `kube-ovn-speaker` ... BFD ... `1000` ...1 ...
- `bfd-min-rx` : `kube-ovn-speaker` ... BFD ... `1000` ...1 ...

- `bfd-detection-multiplier`：`kube-ovn-speaker` BFD 3 = `bfd-min-rx * bfd-detection-multiplier`

7.9.7 BFD

BGP BFD（Bidirectional Forwarding Detection） BGP ECMP

Kube-OVN BFD

- **host network** Kubernetes Node host network BFD BGP ECMP
- **VPC** `OVN logical router BFD` `VPC` CRD VPC kube-ovn controller VPC BFD

```
flowchart TD
    A[BFD     ] --> B[    :     host network]
    A --> C[    : VPC    ]
    B --> D[    : kube-ovn-speaker    BFD]
    B --> E[    : openbfdd     ]
    C --> F[OVN logical router BFD]
```

host network

- **`kube-ovn-speaker` BFD** `kube-ovn-speaker` GoBGP BFD node host network BFD
- **`openbfdd`** BFD DaemonSet node host network BFD

> BFD `OVN logical router BFD` VPC host network BGP ECMP

`kube-ovn-speaker` BFD

`kube-ovn-speaker` GoBGP BFD [RFC 5880](#) / [RFC 5881](#) BGP Speaker BFD `kube-ovn-speaker` GoBGP node host network BFD

`kube-ovn-speaker` BFD

```yaml
args:
  - --enable-bfd=true
  #
  - --bfd-min-tx=1000
  - --bfd-min-rx=1000
  - --bfd-detection-multiplier=3
```

`1000ms × 3 = 3`

300ms

```yaml
args:
  - --enable-bfd=true
  - --bfd-min-tx=100
  - --bfd-min-rx=100
  - --bfd-detection-multiplier=3
```

- **BFD** BFD BFD
- UDP `3784` BFD
- BFD CPU BGP

`gobgp` BFD JSON

```
gobgp -j neighbor <peer-address> | jq '{bfd_config: .bfd, bfd_state: .state.bfd_state}'
```

输出示例:

```
{
  "bfd_config": {
    "enabled": true,
    "port": 3784,
    "desired_minimum_tx_interval": 1000000,
    "required_minimum_receive": 1000000,
    "detection_multiplier": 3
  },
  "bfd_state": {
    "session_state": 1,
    "bfd_async": {
      "transmitted_packets": 100,
      "received_packets": 99
    }
  }
}
```

其中 `session_state` 的值: `1` = UP, `2` = DOWN, `3` = ADMIN_DOWN, `4` = INIT。

当 BFD 会话检测到对端不可达时, GoBGP 会立即对对应的 BGP 邻居执行硬重置 (hard reset), 通知消息为 `BFD is down`。BGP 会话断开后, 路由将被撤回, 上游路由器可以快速切换到备用路径。

方案二: 独立部署 `openbfdd`

如果你希望在主机侧独立维护 BFD 会话, 而不是直接依赖 `kube-ovn-speaker` 内置实现, 可以使用基于 openbfdd 的 BFD DaemonSet。该方案以独立进程方式运行在 node host network 层面, 用于在 BGP 节点上与物理三层交换机网关建立 BFD 会话。默认配置下故障检测时间为 `BFD_MULTI * max(BFD_MIN_TX, BFD_MIN_RX) = 3 * 1000ms = 3 `, 可通过调整参数实现更快或更保守的检测。

DaemonSet 使用 `hostNetwork: true` 模式部署, 每个 Pod 包含三个容器:

- **init-peer**: 初始化容器, 通过 `ip route get` 发现到网关的本地源 IP, 并验证是否在白名单内, 将结果写入共享 Volume。
- **bfdd**: openbfdd 守护进程, 使用 init 容器发现的本地 IP 与交换机网关建立 BFD 会话。启动时通过 `start-bfdd.sh` 脚本初始化, `bfdd-prestart.sh` 作为 startupProbe 验证会话参数。
- **reconcile**: 对账循环容器, 每 5 秒检查一次 BFD 会话状态, 如果发现网关 peer 缺失则自动添加。

下载对应的 yaml:

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/yamls/bfdd-daemonset.yaml
```

修改 `GATEWAY_ADDRESS` 和 `ALLOWED_SOURCE_ADDRESSES` 以匹配实际网络环境, 其中白名单应与 Speaker 的 `--allowed-source-addresses` 参数保持一致:

```
env:
  - name: GATEWAY_ADDRESS
    value: "10.32.32.1"                          #
  - name: ALLOWED_SOURCE_ADDRESSES
    value: "10.32.32.2,10.32.32.3,10.32.32.4,10.32.32.5"  #   IP
  - name: BFD_MIN_TX
    value: "1000"                                #
  - name: BFD_MIN_RX
    value: "1000"                                #
  - name: BFD_MULTI
    value: "3"                                   #
```

部署 DaemonSet:

```
kubectl apply -f bfdd-daemonset.yaml
```

BFD

```
#    BFD daemon
bfdd-control status

#      BFD    remote=  , local=    IP
bfdd-control status remote <gateway-ip> local <local-ip>

#    BFD
bfdd-control allow <gateway-ip>

#
bfdd-control session new set mintx <ms> ms   #
bfdd-control session new set minrx <ms> ms   #
bfdd-control session new set multi <n>       #

#
bfdd-control log type command no
```

```
#    init        IP
cat /bfdd-peer/local-ip

#    reconcile sidecar
kubectl logs -n kube-system ds/openbfdd -c reconcile

#    init
kubectl logs -n kube-system ds/openbfdd -c init-peer
```

**OVN BFD **

OVN BFD kube-ovn controller

```
#    OVN BFD
kubectl ko nbctl list bfd

#           BFD
kubectl ko nbctl find bfd logical_port=<lrp-name>

#    OVN BFD
kubectl ko nbctl destroy bfd <uuid>
```

7.9.8 BGP routes debug

```
# show peer neighbor
gobgp neighbor

# show announced routes to one peer
gobgp neighbor 10.32.32.254 adj-out
```

PDF Slack Support

2026-11-7

2022-6-14

GitHub 

7.9.9

7.10 □□ MetalLB □ Kube-OVN Underlay □□

MetalLB □□□□ Kubernetes □□□□□□□□□□□□□□□□□□□□□□ Kubernetes □□□□ Service □□□□□□□□□□□□□□□

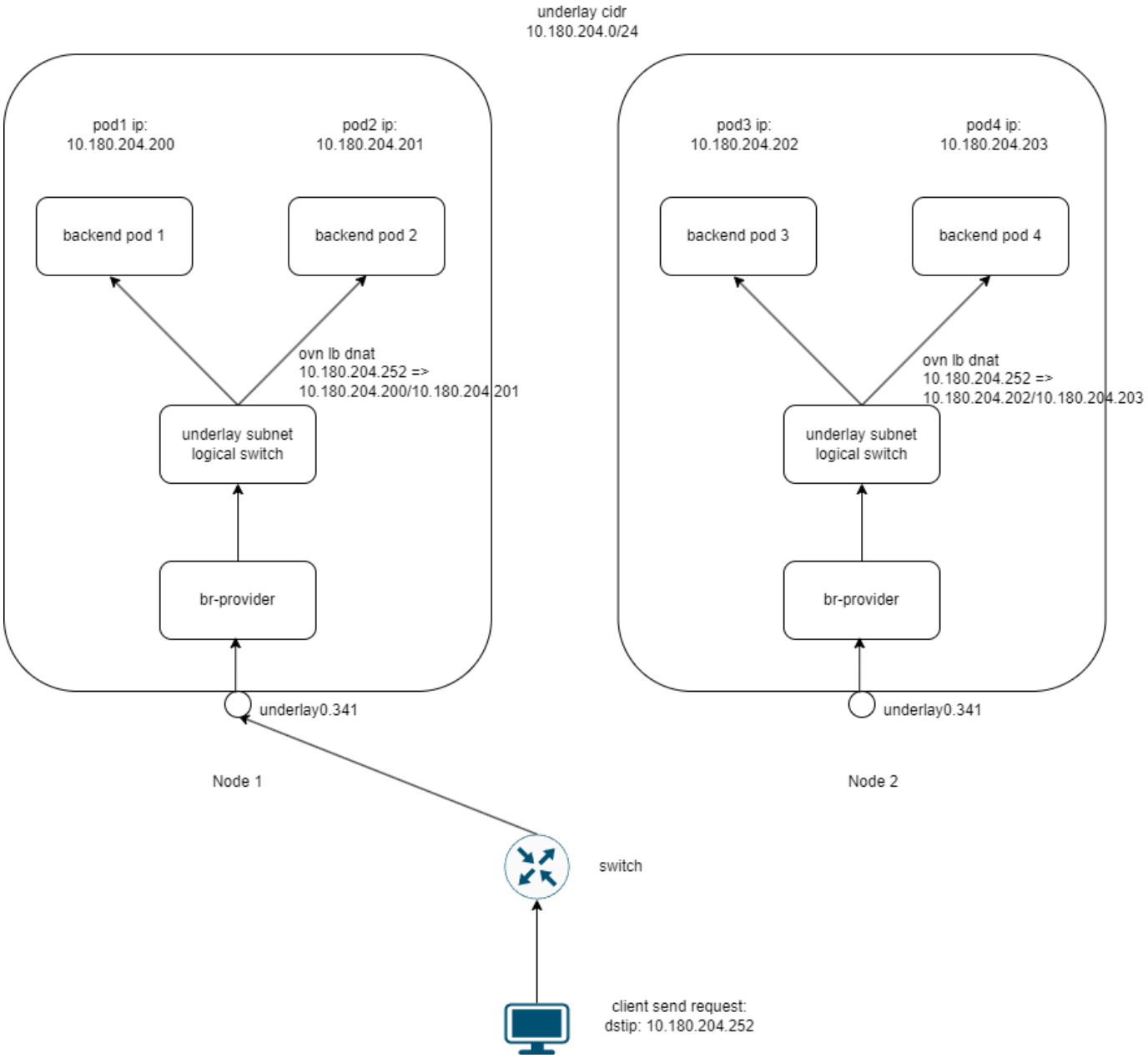
□□□□□□□□ MetalLB □ Kube-OVN □ Underlay □□□□□□□□

7.10.1 □□□□

Kube-OVN □□□ 1.14.0 □□□□ MetalLB □ Underlay □□□□□□□□□□□□□□□□□□

- □□□□□□□□□□□ MetalLB □□□ IP □□□□□□□□□□□□
- □□□□□ Pod □ MetalLB VIP □□□□ Underlay □□□
- □□□□□□ IP □□□□□□□□□□ SNAT

7.10.2 □□□□



2025 Kube-OVN Team

图 1: MetalLB VIP 与 Kube-OVN Underlay 集成的网络流量路径

MetalLB 与 Kube-OVN Underlay 集成的流量走向如下:

1. 外部客户端发送请求至目标 VIP（例如 10.180.204.252），该 IP 由 MetalLB 通过 L2 模式宣告，该流量图中是由 Node1 声明 metallb 的 VIP
2. 请求通过物理网络到达宣告该 VIP 的节点，进入节点上的 `underlay0.341` 网络接口
3. 流量到达节点上的 `br-provider` 网桥，作为 Underlay 网络的入口点
4. `br-provider` 通过 OpenFlow 流表规则将流量转发到 OVN 的逻辑网络
5. 流量进入 `underlay subnet` 逻辑交换机，由 OVN 负载均衡器（`ovn lb dnat`）处理
6. OVN 负载均衡器转发给本地节点的任意 Pod:

整个子网网段为 10.180.204.0/24，包括 VIP 和后端 Pod 的 IP 地址都位于该网段内。

7.10.3 前提条件

- Kube-OVN 控制器启用 `--enable-ovn-lb-prefer-local=true` 选项
- Underlay 子网配置 `enableExternalLBAddress=true`
- 在 Underlay 子网的 `excludeIps` 中添加 MetalLB 地址池使用的 IP 范围

7.10.4 部署步骤

1. 部署 Kube-OVN

按照常规方式部署 Kube-OVN，确保 Kube-OVN 控制器启用 `--enable-ovn-lb-prefer-local=true` 和 `--ls-ct-skip-dst-lport-ips=false` 选项:

```
#   kube-ovn-controller   Deployment
kubectl edit deployment -n kube-system kube-ovn-controller
```

添加以下参数到命令行:

```
--enable-ovn-lb-prefer-local=true
--ls-ct-skip-dst-lport-ips=false
```

2. 配置 Underlay 子网

创建或修改 Underlay 子网，启用外部 LoadBalancer 地址支持，并在 excludeIps 中排除 MetalLB 将使用的 IP 范围:

```
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: underlay-subnet
spec:
  protocol: IPv4
  provider: ovn
  cidrBlock: 10.180.204.0/24  #
  gateway: 10.180.204.1
  excludeIps:
  - 10.180.204.250
  - 10.180.204.251
  - 10.180.204.252  # MetalLB            VIP 10.180.204.252
  natOutgoing: false
  enableExternalLBAddress: true   #     subnet   cidr    ip      metallb vip
```

3. 部署 MetalLB

按照 MetalLB 官方文档 部署 MetalLB:

```
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.13.7/config/manifests/metallb-native.yaml
```

配置 MetalLB 的地址池和 L2 通告模式:

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
```


2025 Kube-OVN Team

```
  name: underlay-pool
  namespace: metallb-system
spec:
  addresses:
  - 10.180.204.250-10.180.204.254  #      VIP 10.180.204.252
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: l2-advert
  namespace: metallb-system
spec:
  ipAddressPools:
  - underlay-pool
```

4. □□ LoadBalancer □□□ Service

□□□□ LoadBalancer □□□ Service□□□□□□ Underlay □□□□ Pod□

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: deploy-169402624
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        ovn.kubernetes.io/logical_switch: underlay-subnet
      labels:
        app: nginx
    spec:
      containers:
      - args:
        - netexec
        - --http-port
        - "80"
        image: kubeovn/agnhost:2.47
        imagePullPolicy: IfNotPresent
        name: nginx
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-lb
spec:
  externalTrafficPolicy: Local
  ipFamilies:
  - IPv4
  ipFamilyPolicy: PreferDualStack
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

7.10.5 □□□□□

1. □□ Service □□□□ MetalLB □□□ IP □□□

```
kubectl get svc nginx-lb
```

□□□□ EXTERNAL-IP □□□□□□ IP □□□□ 10.180.204.252□□

1. □□□□□ Service □ IP □□□

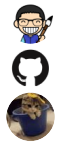
```
curl http://10.180.204.252
```

1. □□□□□□□□□□□□□□□ Pod□

□□□□□□ Service □ endpoints □ Pod □□□□□□□□□□□□□□□□□□□□

```
#    Service   endpoints
kubectl get endpoints nginx-lb
```


2025 Kube-OVN Team



```
#    Pod
kubectl get pods -l app=nginx -o wide
```

1. 验证客户端 IP 保留情况：

在 nginx Pod 中可以查看访问日志，确认记录的客户端 IP 是原始客户端的真实 IP，而不是经过 SNAT 后的 IP：

```
kubectl exec -it $(kubectl get pods -l app=nginx -o name | head -n1) -- cat /var/log/nginx/access.log
```

7.10.6 ⬚⬚⬚⬚

IP 地址池配置

MetalLB 的地址池范围必须是 Underlay 子网 CIDR 的子集，并且需要在 Underlay 子网 `excludeIps` 字段中明确排除，以避免 IP 分配冲突。

网络接口要求

MetalLB 必须使用与 Kube-OVN Underlay 子网相同的网络接口（如示例中 `underlay0.341`）。该接口应配置为 VLAN 子接口，以确保能够正确广播带有 VLAN 标签的 ARP 报文，从而正确通告 MetalLB VIP。

本地流量策略

要启用本地优先功能，需要满足两个条件：- Kube-OVN 控制器启 `--enable-ovn-lb-prefer-local=true` 参数 - Service 配置中设置 `externalTrafficPolicy: Local`

PDF Slack Support

2025年9月10日

2025年4月2日

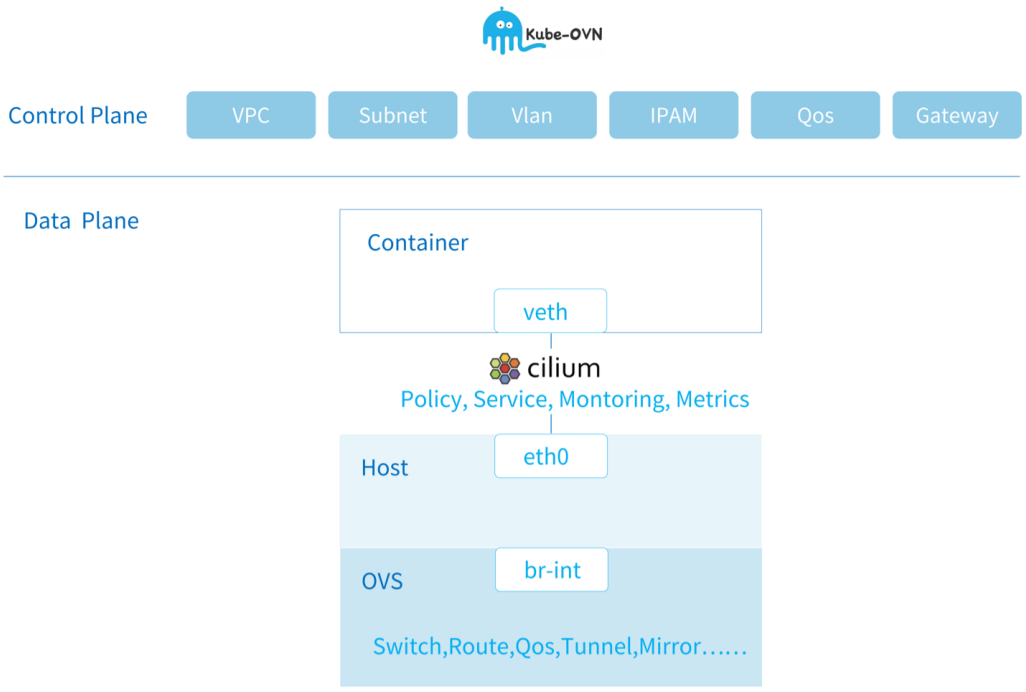
GitHub ## 7.10.7 ⬚⬚

7.11 Cilium ⬚⬚

Cilium ⬚⬚⬚⬚ eBPF ⬚⬚⬚⬚⬚⬚⬚⬚Kube-OVN ⬚⬚⬚⬚ CNI Chaining ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ ⬚⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ eBPF ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

⬚⬚⬚⬚ Cilium⬚Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

- ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚
- ⬚⬚ Hubble ⬚⬚⬚⬚⬚⬚



7.11.1 ⬚⬚⬚⬚

1. Linux ⬚⬚⬚⬚⬚⬚ 4.19 ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ eBPF ⬚⬚⬚⬚⬚
2. ⬚⬚⬚⬚ Helm ⬚⬚⬚ Cilium ⬚⬚⬚⬚⬚⬚ Helm ⬚⬚⬚ Installing Helm⬚

7.11.2 ⬚⬚ Kube-OVN

⬚⬚⬚⬚⬚⬚ Cilium ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚ `networkpolicy` ⬚⬚⬚⬚⬚⬚ CNI ⬚⬚⬚⬚⬚⬚

⬚ `install.sh` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
ENABLE_NP=false
CNI_CONFIG_PRIORITY=10
```

⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `kube-ovn-controller` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ `networkpolicy` ⬚

```
args:
- --enable-np=false
```

⬚⬚ `kube-ovn-cni` ⬚⬚⬚⬚⬚⬚ CNI ⬚⬚⬚⬚⬚⬚

```
args:
- --cni-conf-name=10-kube-ovn.conflist
```


2025 Kube-OVN Team

⬚⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Cilium ⬚⬚⬚⬚⬚

```
mv /etc/cni/net.d/01-kube-ovn.conflist /etc/cni/net.d/10-kube-ovn.conflist
```

7.11.3 ⬚⬚ Cilium

⬚⬚ `chaining.yaml` ⬚⬚⬚⬚⬚⬚⬚ Cilium ⬚ generic-veth ⬚⬚⬚

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cni-configuration
  namespace: kube-system
data:
  cni-config: |-
    {
      "name": "generic-veth",
      "cniVersion": "0.3.1",
      "plugins": [
        {
          "type": "kube-ovn",
          "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
          "ipam": {
              "type": "kube-ovn",
              "server_socket": "/run/openvswitch/kube-ovn-daemon.sock"
          }
        },
        {
          "type": "portmap",
          "snat": true,
          "capabilities": {"portMappings": true}
        },
        {
          "type": "cilium-cni"
        }
      ]
    }
```

⬚⬚⬚⬚⬚⬚⬚

```
kubectl apply -f chaining.yaml
```

⬚⬚ Helm ⬚⬚ Cilium⬚

```
helm repo add cilium https://helm.cilium.io/
helm install cilium cilium/cilium --version 1.11.6 \
    --namespace kube-system \
    --set cni.chainingMode=generic-veth \
    --set cni.customConf=true \
    --set cni.configMap=cni-configuration \
    --set tunnel=disabled \
    --set enableIPv4Masquerade=false \
    --set devices="eth+ ovn0 genev_sys_6081 vxlan_sys_4789" \
    --set enableIdentityMark=false
```

⬚⬚ Cilium ⬚⬚⬚⬚⬚

```
# cilium  status
    /¯¯\
 /¯¯\__/¯¯\    Cilium:         OK
 \__/¯¯\__/    Operator:       OK
 /¯¯\__/¯¯\    Hubble:         disabled
 \__/¯¯\__/    ClusterMesh:    disabled
    \__/

DaemonSet         cilium             Desired: 2, Ready: 2/2, Available: 2/2
Deployment        cilium-operator    Desired: 2, Ready: 2/2, Available: 2/2
Containers:       cilium             Running: 2
                  cilium-operator    Running: 2
Cluster Pods:     8/11 managed by Cilium
Image versions    cilium             quay.io/cilium/cilium:v1.11.6: 2
                  cilium-operator    quay.io/cilium/operator-generic:v1.11.6: 2
```

7.11.4 ⬚ Cilium ⬚⬚⬚⬚ Kube-OVN NAT ⬚⬚

Cilium ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ NAT ⬚⬚ Pod ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ ⬚⬚ NAT ⬚⬚⬚⬚⬚ VPC ⬚ Pod ⬚⬚⬚⬚⬚ SNAT/DNAT ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ IP ⬚⬚ Pod ⬚⬚⬚ IP⬚

Cilium ⬚⬚ **SIP**⬚Source IP⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚——⬚⬚⬚⬚⬚⬚ Pod IP ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Hubble ⬚⬚⬚ **DROPPED**⬚⬚







该机制有助于防止集群内 IP 欺骗，但会导致 NAT 网关功能失效。

按命名空间/Pod 粒度控制（推荐，**Cilium ≥ 1.20**）

Cilium 1.20 起（cilium/cilium#43505），支持通过双层 **annotation** 在 Pod 粒度关闭 SIP 校验：

1. 集群管理员在命名空间上添加授权 annotation：

```
kubectl annotate namespace <nat-gateway-namespace> \
  config.cilium.io/delegate-source-ip-verification=true
```

2. NAT 网关 Pod 声明关闭 SIP 校验的 annotation：

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    config.cilium.io/disable-source-ip-verification: "true"
```

这种方式仅对特定 NAT 网关 Pod 生效，可避免集群层面的安全降级，推荐优先采用。

全局关闭（**Cilium < 1.20**）

对于旧版本 Cilium，可在 Helm Chart 中通过设置 `enableSourceIPVerification` 为 `false` 全局关闭 SIP 校验：

```
enableSourceIPVerification: false
```

PDF Slack Support

2026年4月28日

2022524

GitHub 7.11.5

2025 Kube-OVN Team

7.12 Cilium NetworkPolicy ⬚⬚

Kube-OVN ⬚⬚⬚⬚⬚⬚ Cilium ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Cilium ⬚⬚⬚

⬚⬚⬚ Cilium ⬚⬚⬚⬚⬚⬚⬚ Cilium ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Cilium L3 ⬚ L4 ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

7.12.1 ⬚⬚⬚⬚

⬚⬚⬚⬚ **Pod**

⬚⬚ namespace `test` ⬚⬚⬚⬚⬚ yaml⬚⬚ test namespace ⬚⬚⬚⬚⬚ label `app=test` ⬚ Pod⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Pod⬚

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: test
  name: test
  namespace: test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
      - image: docker.io/library/nginx:alpine
        imagePullPolicy: IfNotPresent
        name: nginx
```

⬚⬚⬚⬚⬚⬚ yaml⬚⬚ default namespace ⬚⬚⬚⬚⬚ label `app=dynamic` ⬚ Pod ⬚⬚⬚⬚⬚⬚⬚⬚ Pod⬚

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: dynamic
  name: dynamic
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: dynamic
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: dynamic
    spec:
      containers:
      - image: docker.io/library/nginx:alpine
        imagePullPolicy: IfNotPresent
        name: nginx
```

⬚⬚⬚⬚ Pod ⬚⬚ Label ⬚⬚:

```
# kubectl get pod -o wide --show-labels
NAME                       READY   STATUS    RESTARTS   AGE   IP           NODE                   NOMINATED NODE   READINESS GATES   LABELS
dynamic-7d8d7874f5-9v5c4   1/1     Running   0          28h   10.16.0.35   kube-ovn-worker        <none>           <none>            app=dynamic,pod-
template-hash=7d8d7874f5
dynamic-7d8d7874f5-s8z2n   1/1     Running   0          28h   10.16.0.36   kube-ovn-control-plane   <none>           <none>            app=dynamic,pod-
template-hash=7d8d7874f5
# kubectl get pod -o wide -n test --show-labels
NAME                       READY   STATUS    RESTARTS   AGE     IP           NODE                   NOMINATED NODE   READINESS GATES   LABELS
dynamic-7d8d7874f5-6dsg6   1/1     Running   0          7h20m   10.16.0.2    kube-ovn-control-plane   <none>           <none>            app=dynamic,pod-
template-hash=7d8d7874f5
```


2025 Kube-OVN Team

```
dynamic-7d8d7874f5-tjgtp       1/1     Running   0          7h46m   10.16.0.42   kube-ovn-worker          <none>           <none>           app=dynamic,pod-
template-hash=7d8d7874f5
label-test1-77b6764857-swq4k   1/1     Running   0          3h43m   10.16.0.12   kube-ovn-worker          <none>           <none>           app=test1,pod-
template-hash=77b6764857

#          Pod
test-54c98bc466-mft5s          1/1     Running   0          8h      10.16.0.41   kube-ovn-worker          <none>           <none>           app=test,pod-
template-hash=54c98bc466
```

L3 ...

... yaml... `CiliumNetworkPolicy` ...:

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "l3-rule"
  namespace: test
spec:
  endpointSelector:
    matchLabels:
      app: test
  ingress:
  - fromEndpoints:
    - matchLabels:
        app: dynamic
```

... default namespace ... Pod ... Pod ... ... test namespace ... Pod ...

default namespace ...:

```
# kubectl exec -it dynamic-7d8d7874f5-9v5c4 -- bash
bash-5.0# ping -c 3 10.16.0.41
PING 10.16.0.41 (10.16.0.41): 56 data bytes

--- 10.16.0.41 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
```

test namespace ... Pod ...:

```
# kubectl exec -it -n test dynamic-7d8d7874f5-6dsg6 -- bash
bash-5.0# ping -c 3 10.16.0.41
PING 10.16.0.41 (10.16.0.41): 56 data bytes
64 bytes from 10.16.0.41: seq=0 ttl=64 time=2.558 ms
64 bytes from 10.16.0.41: seq=1 ttl=64 time=0.223 ms
64 bytes from 10.16.0.41: seq=2 ttl=64 time=0.304 ms

--- 10.16.0.41 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.223/1.028/2.558 ms
```

... Cilium ... `CiliumNetworkPolicy` ... `Namespace` ... Cilium ...

... **Namespace** ... Pod ... **Namespace** ... Pod ...

... Namespace ... Namespace ...

... `CiliumNetworkPolicy` ... namespace ...:

```yaml
  ingress:
  - fromEndpoints:
    - matchLabels:
        app: dynamic
        k8s:io.kubernetes.pod.namespace: default    #     Namespace    Pod
```

... `CiliumNetworkPolicy` ...:

```
# kubectl get cnp -n test  -o yaml l3-rule
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: l3-rule
  namespace: test
spec:
  endpointSelector:
    matchLabels:
      app: test
  ingress:
  - fromEndpoints:
    - matchLabels:
```

```
        app: dynamic
    - matchLabels:
        app: dynamic
        k8s:io.kubernetes.pod.namespace: default
```

... default namespace ... Pod ... Pod ...:

```
# kubectl exec -it dynamic-7d8d7874f5-9v5c4 -n test -- bash
bash-5.0# ping -c 3 10.16.0.41
PING 10.16.0.41 (10.16.0.41): 56 data bytes
64 bytes from 10.16.0.41: seq=0 ttl=64 time=2.383 ms
64 bytes from 10.16.0.41: seq=1 ttl=64 time=0.115 ms
64 bytes from 10.16.0.41: seq=2 ttl=64 time=0.142 ms

--- 10.16.0.41 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.115/0.880/2.383 ms
```

... Kubernetes ... networkpolicy... Cilium ... Namespace ... Namespace ...

... Kube-OVN ...Kube-OVN ... k8s ... Namespace ... **Pod**... Pod... Namespace ... Namespace ... Pod... Pod ...

L4 ...

... yaml... L4 ...:

```
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "l4-rule"
  namespace: test
spec:
  endpointSelector:
    matchLabels:
      app: test
  ingress:
  - fromEndpoints:
    - matchLabels:
        app: dynamic
    toPorts:
    - ports:
      - port: "80"
        protocol: TCP
```

... Namespace ... Pod ...

```
# kubectl exec -it -n test dynamic-7d8d7874f5-6dsg6 -- bash
bash-5.0# ping -c 3 10.16.0.41
PING 10.16.0.41 (10.16.0.41): 56 data bytes

--- 10.16.0.41 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
bash-5.0#
bash-5.0# curl 10.16.0.41:80
<html>
<head>
        <title>Hello World!</title>
        <link href='//fonts.googleapis.com/css?family=Open+Sans:400,700' rel='stylesheet' type='text/css'>
        <style>
        body {
                background-color: white;
                text-align: center;
                padding: 50px;
                font-family: "Open Sans","Helvetica Neue",Helvetica,Arial,sans-serif;
        }
        #logo {
                margin-bottom: 40px;
        }
        </style>
</head>
<body>
                <h1>Hello World!</h1>
                                <h3>Links found</h3>
        <h3>I am on  test-54c98bc466-mft5s</h3>
        <h3>Cookie                  =</h3>
                                    <b>KUBERNETES</b> listening in 443 available at tcp://10.96.0.1:443<br />
                                            <h3>my name is hanhouchao!</h3>
                        <h3> RequestURI='/'</h3>
</body>
</html>
```



相同 Namespace 下，不符合网络策略规则的 Pod 访问测试

```
# kubectl exec -it -n test label-test1-77b6764857-swq4k -- bash
bash-5.0# ping -c 3 10.16.0.41
PING 10.16.0.41 (10.16.0.41): 56 data bytes

--- 10.16.0.41 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
bash-5.0#
bash-5.0# curl -v 10.16.0.41:80 --connect-timeout 10
*   Trying 10.16.0.41:80...
* After 10000ms connect time, move on!
* connect to 10.16.0.41 port 80 failed: Operation timed out
* Connection timeout after 10001 ms
* Closing connection 0
curl: (28) Connection timeout after 10001 ms
```

网络策略生效后，跨 Namespace 的访问，依然是被禁止的，跟 L3 网络策略测试结果一致。

在 L4 网络规则生效后，ping 无法使用，但是符合策略规则的 TCP 访问，是可以正常执行的。

关于 ICMP 的限制，可以参考官方说明 L4 限制说明。

L7 网络策略测试

chaining 模式下，L7 网络策略目前是存在问题的。在 Cilium 官方文档中，对这种情况给出了说明，参考 Generic Veth Chaining 。

这个问题使用 issue 12454 跟踪，目前还没有解决。

PDF Slack Support

2026年7月4日

2022年8月2日

GitHub

7.12.2 ⛓

7.13 Cilium [illegible]

Kube-OVN [illegible] Cilium [illegible] Cilium [illegible]

Cilium [illegible] Hubble [illegible]Hubble [illegible]

7.13.1 [illegible] Hubble

[illegible] Cilium [illegible] Hubble [illegible] Hubble[illegible]

[illegible] helm [illegible] Hubble[illegible]

```
helm upgrade cilium cilium/cilium --version 1.11.6 \
   --namespace kube-system \
   --reuse-values \
   --set hubble.relay.enabled=true \
   --set hubble.ui.enabled=true
```

[illegible] Hubble [illegible] `cilium status` [illegible] Hubble [illegible]

```
# cilium status
    /¯¯\
 /¯¯\__/¯¯\    Cilium:         OK
 \__/¯¯\__/    Operator:       OK
 /¯¯\__/¯¯\    Hubble:         OK
 \__/¯¯\__/    ClusterMesh:    disabled
    \__/

Deployment        hubble-relay       Desired: 1, Ready: 1/1, Available: 1/1
Deployment        cilium-operator    Desired: 2, Ready: 2/2, Available: 2/2
DaemonSet         cilium             Desired: 2, Ready: 2/2, Available: 2/2
Deployment        hubble-ui          Desired: 1, Ready: 1/1, Available: 1/1
Containers:       cilium             Running: 2
                  hubble-ui          Running: 1
                  hubble-relay       Running: 1
                  cilium-operator    Running: 2
Cluster Pods:     16/17 managed by Cilium
Image versions    hubble-relay       quay.io/cilium/hubble-relay:v1.11.6@sha256:fd9034a2d04d5b973f1e8ed44f230ea195b89c37955ff32e34e5aa68f3ed675a: 1
                  cilium-operator    quay.io/cilium/operator-generic:v1.11.6@sha256:9f6063c7bcaede801a39315ec7c166309f6a6783e98665f6693939cf1701bc17: 2
                  cilium             quay.io/cilium/cilium:v1.11.6@sha256:f7f93c26739b6641a3fa3d76b1e1605b15989f25d06625260099e01c8243f54c: 2
                  hubble-ui          quay.io/cilium/hubble-ui:v0.9.0@sha256:0ef04e9a29212925da6bdfd0ba5b581765e41a01f1cc30563cef9b30b457fea0: 1
                  hubble-ui          quay.io/cilium/hubble-ui-backend:v0.9.0@sha256:000df6b76719f607a9edefb9af94dfd1811a6f1b6a8a9c537cba90bf12df474b: 1
apple@bogon cilium %
```

[illegible] Hubble [illegible] Hubble CLI :

```
curl -L --fail --remote-name-all https://github.com/cilium/hubble/releases/download/v0.10.0/hubble-linux-amd64.tar.gz
sudo tar xzvfC hubble-linux-amd64.tar.gz /usr/local/bin
```

7.13.2 [illegible]

Cilium [illegible]

[illegible] `cilium connectivity test` [illegible]Cilium [illegible] `cilium-test` [illegible] Namespace[illegible] cilium-test [illegible]

[illegible] cilium-test namespace [illegible]

```
# kubectl get all -n cilium-test
NAME                                   READY   STATUS    RESTARTS   AGE
pod/client-7df6cfbf7b-z5t2j            1/1     Running   0          21s
pod/client2-547996d7d8-nvgxg           1/1     Running   0          21s
pod/echo-other-node-d79544ccf-hl4gg    2/2     Running   0          21s
pod/echo-same-node-5d466d5444-ml7tc    2/2     Running   0          21s

NAME                      TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/echo-other-node   NodePort   10.109.58.126   <none>        8080:32269/TCP   21s
service/echo-same-node    NodePort   10.108.70.32    <none>        8080:32490/TCP   21s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/client            1/1     1            1           21s
deployment.apps/client2           1/1     1            1           21s
deployment.apps/echo-other-node   1/1     1            1           21s
deployment.apps/echo-same-node    1/1     1            1           21s

NAME                                         DESIRED   CURRENT   READY   AGE
```


2025 Kube-OVN Team

```
replicaset.apps/client-7df6cfbf7b           1         1         1       21s
replicaset.apps/client2-547996d7d8          1         1         1       21s
replicaset.apps/echo-other-node-d79544ccf   1         1         1       21s
replicaset.apps/echo-same-node-5d466d5444   1         1         1       21s
```

7.13.3 [illegible]

[illegible] Cilium [illegible] [illegible] `kube-system namespace` [illegible] Cilium [illegible] pod [illegible] `hubble observe` [illegible]

```
# kubectl get pod -n kube-system -o wide
NAME                                      READY   STATUS    RESTARTS   AGE     IP           NODE                    NOMINATED NODE   READINESS GATES
cilium-d6h56                              1/1     Running   0          2d20h   172.18.0.2   kube-ovn-worker         <none>           <none>
cilium-operator-5887f78bbb-c7sb2          1/1     Running   0          2d20h   172.18.0.2   kube-ovn-worker         <none>           <none>
cilium-operator-5887f78bbb-wj8gt          1/1     Running   0          2d20h   172.18.0.3   kube-ovn-control-plane  <none>           <none>
cilium-tq5xb                              1/1     Running   0          2d20h   172.18.0.3   kube-ovn-control-plane  <none>           <none>
kube-ovn-pinger-7lgk8                     1/1     Running   0          21h     10.16.0.19   kube-ovn-control-plane  <none>           <none>
kube-ovn-pinger-msvcn                     1/1     Running   0          21h     10.16.0.18   kube-ovn-worker         <none>           <none>

# kubectl exec -it -n kube-system cilium-d6h56 -- bash
root@kube-ovn-worker:/home/cilium# hubble observe --from-namespace kube-system
Jul 29 03:24:25.551: kube-system/kube-ovn-pinger-msvcn:35576 -> 172.18.0.3:6642 to-stack FORWARDED (TCP Flags: ACK, PSH)
Jul 29 03:24:25.561: kube-system/kube-ovn-pinger-msvcn:35576 -> 172.18.0.3:6642 to-stack FORWARDED (TCP Flags: RST)
Jul 29 03:24:25.561: kube-system/kube-ovn-pinger-msvcn:35576 -> 172.18.0.3:6642 to-stack FORWARDED (TCP Flags: ACK, RST)
Jul 29 03:24:25.572: kube-system/kube-ovn-pinger-msvcn:35578 -> 172.18.0.3:6642 to-stack FORWARDED (TCP Flags: SYN)
Jul 29 03:24:25.572: kube-system/kube-ovn-pinger-msvcn:35578 -> 172.18.0.3:6642 to-stack FORWARDED (TCP Flags: ACK)
Jul 29 03:24:25.651: kube-system/kube-ovn-pinger-msvcn:35578 -> 172.18.0.3:6642 to-stack FORWARDED (TCP Flags: ACK, PSH)
Jul 29 03:24:25.661: kube-system/kube-ovn-pinger-msvcn:35578 -> 172.18.0.3:6642 to-stack FORWARDED (TCP Flags: RST)
Jul 29 03:24:25.661: kube-system/kube-ovn-pinger-msvcn:35578 -> 172.18.0.3:6642 to-stack FORWARDED (TCP Flags: ACK, RST)
Jul 29 03:24:25.761: kube-system/kube-ovn-pinger-msvcn:52004 -> 172.18.0.3:6443 to-stack FORWARDED (TCP Flags: ACK, PSH)
Jul 29 03:24:25.779: kube-system/kube-ovn-pinger-msvcn -> kube-system/kube-ovn-pinger-7lgk8 to-stack FORWARDED (ICMPv4 EchoRequest)
Jul 29 03:24:25.779: kube-system/kube-ovn-pinger-msvcn <- kube-system/kube-ovn-pinger-7lgk8 to-endpoint FORWARDED (ICMPv4 EchoReply)
Jul 29 03:24:25.866: kube-system/hubble-ui-7596f7ff6f-7j6f2:55836 <- kube-system/hubble-relay-959988db5-zc5vv:4245 to-stack FORWARDED (TCP Flags: ACK)
Jul 29 03:24:25.866: kube-system/hubble-ui-7596f7ff6f-7j6f2:55836 <- kube-system/hubble-relay-959988db5-zc5vv:80 to-endpoint FORWARDED (TCP Flags: ACK)
Jul 29 03:24:25.866: kube-system/hubble-ui-7596f7ff6f-7j6f2:55836 -> kube-system/hubble-relay-959988db5-zc5vv:4245 to-stack FORWARDED (TCP Flags: ACK)
Jul 29 03:24:25.866: kube-system/hubble-ui-7596f7ff6f-7j6f2:55836 -> kube-system/hubble-relay-959988db5-zc5vv:4245 to-endpoint FORWARDED (TCP Flags: ACK)
Jul 29 03:24:25.975: kube-system/kube-ovn-pinger-7lgk8 -> kube-system/kube-ovn-pinger-msvcn to-endpoint FORWARDED (ICMPv4 EchoRequest)
Jul 29 03:24:25.975: kube-system/kube-ovn-pinger-7lgk8 <- kube-system/kube-ovn-pinger-msvcn to-stack FORWARDED (ICMPv4 EchoReply)
Jul 29 03:24:25.979: kube-system/kube-ovn-pinger-msvcn -> 172.18.0.3 to-stack FORWARDED (ICMPv4 EchoRequest)
Jul 29 03:24:26.037: kube-system/coredns-6d4b75cb6d-lbgjg:36430 -> 172.18.0.3:6443 to-stack FORWARDED (TCP Flags: ACK)
Jul 29 03:24:26.282: kube-system/kube-ovn-pinger-msvcn -> 172.18.0.2 to-stack FORWARDED (ICMPv4 EchoRequest)
```

[illegible] Hubble Relay [illegible]Hubble [illegible]

[illegible]

[illegible] Hubble API[illegible] Hubble Service[illegible] `kubectl port-forward deployment/hubble-relay -n kube-system 4245:4245` [illegible] [illegible]

[illegible] [illegible]

`kubectl port-forward` [illegible]

[illegible] `hubble status` [illegible]

```
# hubble status
Healthcheck (via localhost:4245): Ok
Current/Max Flows: 8,190/8,190 (100.00%)
Flows/s: 22.86
Connected Nodes: 2/2
```

命令行观测

在终端上执行 `hubble observe` 命令，查看集群的流量信息。观测到的 cilium-test 相关的测试流量参考如下：

```
apple@bogon ovn-test % hubble observe
Jul 28 08:00:07.033: cilium-test/client-7df6cfbf7b-qn7q6:56906 (ID:15432) -> kube-system/coredns-6d4b75cb6d-b444j:53 (ID:11307) to-endpoint FORWARDED (UDP)
Jul 28 08:00:07.033: cilium-test/client-7df6cfbf7b-qn7q6:56906 (ID:15432) <- kube-system/coredns-6d4b75cb6d-b444j:53 (ID:11307) to-stack FORWARDED (UDP)
Jul 28 08:00:07.095: 100.64.0.3:43037 (world) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: SYN)
Jul 28 08:00:07.095: 100.64.0.3:43037 (world) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: SYN, ACK)
Jul 28 08:00:07.096: 100.64.0.3:43037 (world) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.096: 100.64.0.3:43037 (world) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.096: 100.64.0.3:43037 (world) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.097: 100.64.0.3:43037 (world) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.249: 100.64.0.2:33419 (host) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: SYN)
Jul 28 08:00:07.249: 100.64.0.2:33419 (host) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: SYN, ACK)
Jul 28 08:00:07.250: 100.64.0.2:33419 (host) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.251: 100.64.0.2:33419 (host) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.251: 100.64.0.2:33419 (host) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.251: 100.64.0.2:33419 (host) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.323: cilium-test/client2-547996d7d8-jgblc:34927 (ID:14804) <- 172.18.0.3:31514 (kube-apiserver) to-endpoint FORWARDED (TCP Flags: SYN, ACK)
Jul 28 08:00:07.323: cilium-test/client2-547996d7d8-jgblc:34927 (ID:14804) -> 172.18.0.3:31514 (kube-apiserver) to-stack FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.324: 100.64.0.2:34927 (world) -> cilium-test/echo-same-node-5d466d5444-4vllm:8080 (ID:15976) to-endpoint FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.324: cilium-test/client2-547996d7d8-jgblc:34927 (ID:14804) -> 172.18.0.3:31514 (kube-apiserver) to-stack FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.324: 100.64.0.2:34927 (world) -> cilium-test/echo-same-node-5d466d5444-4vllm:8080 (ID:15976) to-endpoint FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.324: 100.64.0.2:34927 (world) <- cilium-test/echo-same-node-5d466d5444-4vllm:8080 (ID:15976) to-stack FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.324: cilium-test/client2-547996d7d8-jgblc:34927 (ID:14804) <- 172.18.0.3:31514 (kube-apiserver) to-endpoint FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.324: cilium-test/client2-547996d7d8-jgblc:34927 (ID:14804) -> 172.18.0.3:31514 (kube-apiserver) to-stack FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.324: 100.64.0.2:34927 (world) -> cilium-test/echo-same-node-5d466d5444-4vllm:8080 (ID:15976) to-endpoint FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.340: kube-system/hubble-relay-959988db5-zc5vv:46938 (ID:53347) -> 172.18.0.2:4244 (host) to-stack FORWARDED (TCP Flags: ACK, PSH)
Jul 28 08:00:07.340: kube-system/hubble-relay-959988db5-zc5vv:44656 (ID:53347) -> 172.18.0.3:4244 (kube-apiserver) to-stack FORWARDED (TCP Flags: ACK, PSH)
Jul 28 08:00:07.340: kube-system/hubble-relay-959988db5-zc5vv:46938 (ID:53347) <- 172.18.0.2:4244 (host) to-endpoint FORWARDED (TCP Flags: ACK, PSH)
Jul 28 08:00:07.341: kube-system/hubble-relay-959988db5-zc5vv:44656 (ID:53347) <- 172.18.0.3:4244 (kube-apiserver) to-endpoint FORWARDED (TCP Flags: ACK, PSH)
Jul 28 08:00:07.409: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: SYN)
Jul 28 08:00:07.409: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: SYN)
Jul 28 08:00:07.409: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: SYN, ACK)
Jul 28 08:00:07.410: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: SYN, ACK)
Jul 28 08:00:07.410: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: ACK, PSH)
Jul 28 08:00:07.410: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: ACK)
Jul 28 08:00:07.410: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK, PSH)
Jul 28 08:00:07.411: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: ACK, PSH)
Jul 28 08:00:07.411: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK, PSH)
Jul 28 08:00:07.412: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.412: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) -> cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.412: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-stack FORWARDED (TCP Flags: ACK, FIN)
Jul 28 08:00:07.412: cilium-test/client-7df6cfbf7b-qn7q6:57326 (ID:15432) <- cilium-test/echo-other-node-d79544ccf-r688b:8080 (ID:50956) to-endpoint FORWARDED (TCP Flags: ACK, FIN)
apple@bogon ovn-test %
```

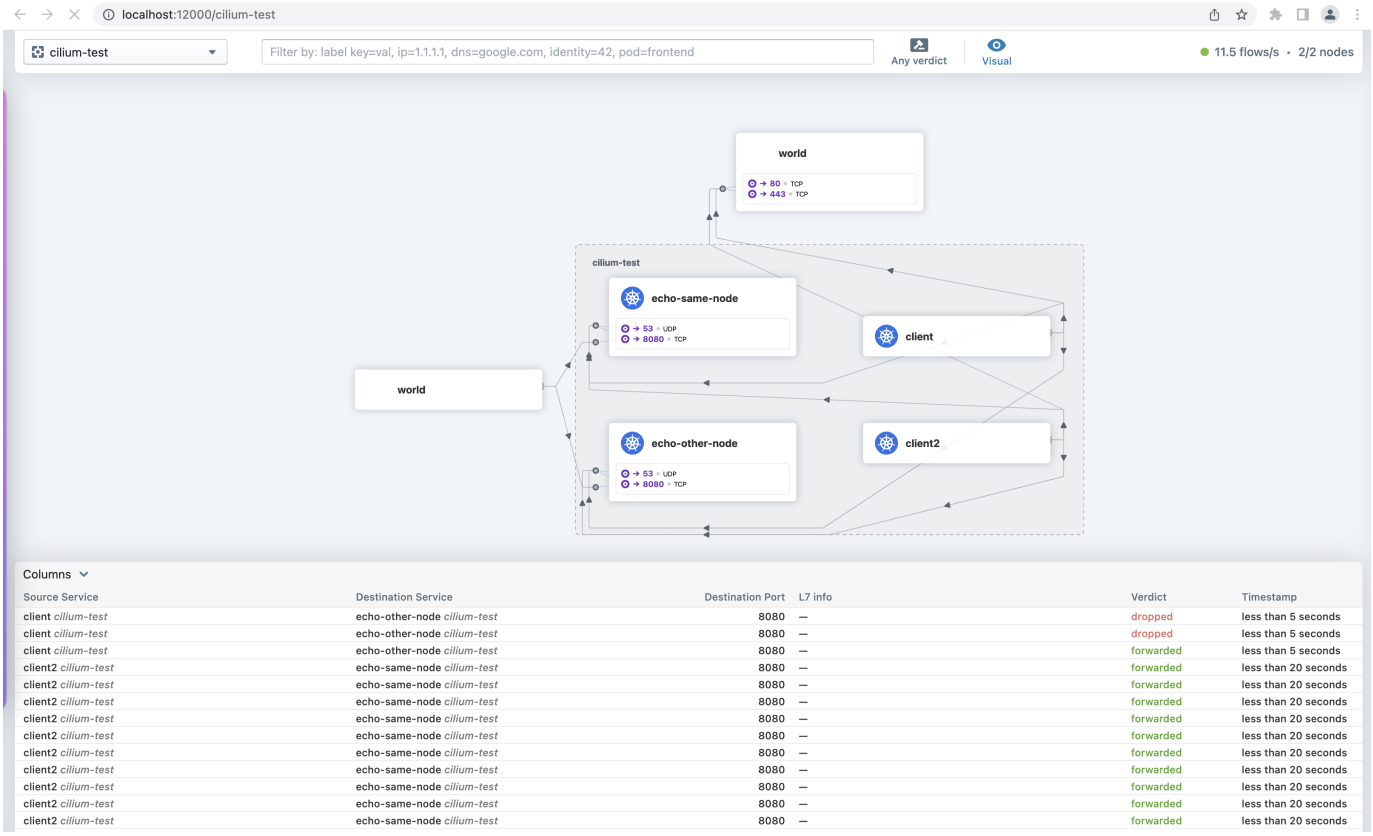
需要注意的是，`hubble observe` 命令的显示结果，是当前命令行执行时查询到的流量信息。多次执行命令行，可以查看到不同的流量信息。更多详细的观测信息，可以执行 `hubble help observe` 命令查看 Hubble CLI 的详细使用方式。

7.13.4 使用 UI 进行流量观测

执行 `cilium status` 命令，确认 Hubble UI 已经安装成功。在第二步的 Hubble 安装中，已经补充了 UI 的安装。

执行命令 `cilium hubble ui` 可以自动创建端口转发，将 `hubble-ui service` 映射到本地端口。正常情况下，执行完命令后，会自动打开本地的浏览器，跳转到 Hubble UI 界面。如果没有自动跳转，在浏览器中输入 `http://localhost:12000` 打开 UI 观察界面。

□□□□□□□□□□ `cilium-test` namespace□□□ Cilium □□□□□□□□□□□



7.13.5 Hubble □□□□

Hubble □□□□□□□□ Pod □□□□□□□□□□□□□□□ Hubble □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□ `hubble.metrics.enabled` □□□:

```
helm upgrade cilium cilium/cilium --version 1.11.6 \
   --namespace kube-system \
   --reuse-values \
   --set hubble.relay.enabled=true \
   --set hubble.ui.enabled=true \
   --set hubble.metrics.enabled="{dns,drop,tcp,flow,icmp,http}"
```

□□□□□□□ kube-system namespace □□□□□□ `hubble-metrics` □□□□□□□□ Endpoints □□ Hubble □□□□□□□□□□□□□□:

```
# curl 172.18.0.2:9091/metrics
# HELP hubble_drop_total Number of drops
# TYPE hubble_drop_total counter
hubble_drop_total{protocol="ICMPv6",reason="Unsupported L3 protocol"} 2
# HELP hubble_flows_processed_total Total number of flows processed
# TYPE hubble_flows_processed_total counter
hubble_flows_processed_total{protocol="ICMPv4",subtype="to-endpoint",type="Trace",verdict="FORWARDED"} 335
hubble_flows_processed_total{protocol="ICMPv4",subtype="to-stack",type="Trace",verdict="FORWARDED"} 335
hubble_flows_processed_total{protocol="ICMPv6",subtype="",type="Drop",verdict="DROPPED"} 2
hubble_flows_processed_total{protocol="TCP",subtype="to-endpoint",type="Trace",verdict="FORWARDED"} 8282
hubble_flows_processed_total{protocol="TCP",subtype="to-stack",type="Trace",verdict="FORWARDED"} 6767
hubble_flows_processed_total{protocol="UDP",subtype="to-endpoint",type="Trace",verdict="FORWARDED"} 1642
hubble_flows_processed_total{protocol="UDP",subtype="to-stack",type="Trace",verdict="FORWARDED"} 1642
# HELP hubble_icmp_total Number of ICMP messages
# TYPE hubble_icmp_total counter
hubble_icmp_total{family="IPv4",type="EchoReply"} 335
hubble_icmp_total{family="IPv4",type="EchoRequest"} 335
hubble_icmp_total{family="IPv4",type="RouterSolicitation"} 2
# HELP hubble_tcp_flags_total TCP flag occurrences
# TYPE hubble_tcp_flags_total counter
hubble_tcp_flags_total{family="IPv4",flag="FIN"} 2043
hubble_tcp_flags_total{family="IPv4",flag="RST"} 301
hubble_tcp_flags_total{family="IPv4",flag="SYN"} 1169
hubble_tcp_flags_total{family="IPv4",flag="SYN-ACK"} 1169
```

PDF Slack Support

2023⬚3⬚26⬚

2022⬚8⬚2⬚

GitHub 

7.13.6 ⬚⬚

2025 Kube-OVN Team

7.14 [illegible]

[illegible] Kube-OVN [illegible]

7.14.1 [illegible]

```yaml
kind: Subnet
apiVersion: kubeovn.io/v1
metadata:
  name: external
spec:
  cidrBlock: 172.31.0.0/16
  gatewayType: centralized
  natOutgoing: false
  externalEgressGateway: 192.168.0.1
  policyRoutingTableID: 1000
  policyRoutingPriority: 1500
```

- `natOutgoing`: [illegible] `false` [illegible]
- `externalEgressGateway` [illegible]
- `policyRoutingTableID` [illegible] TableID [illegible]
- `policyRoutingPriority` [illegible]

PDF Slack Support

2022[illegible]6[illegible]8[illegible]

2022[illegible]5[illegible]24[illegible]

GitHub 

7.14.2 [illegible]

2025 Kube-OVN Team

7.15 VIP 和 IP

VIP 即虚拟 IP，用于预留 IP 资源。VIP 设计之初参考了 OpenStack Neutron Allowed-Address-Pairs（AAP）的功能，后来又扩展支持在子网内为 OVN Switch LB Rule 提供前端 VIP。Pod 地址分配由 `IP` 资源和 IPAM 管理，VIP 不再用于为 Pod 固定地址。目前 VIP 只有两种使用场景：

- Allowed-Address-Pairs VIP
- Switch LB rule VIP

7.15.1 1. Allowed-Address-Pairs VIP

在该场景下我们希望动态的预留一部分 IP 但是并不分配给 Pod 而是分配给其他的基础设施启用，例如：

- Kubernetes 嵌套 Kubernetes 的场景中上层 Kubernetes 使用 Underlay 网络会占用底层 Subnet 可用地址。
- LB 或其他网络基础设施需要使用一个 Subnet 内的 IP，但不会单独起 Pod。

此外，VIP 还可以为 Allowed-Address-Pairs 预留 IP 用来支持单个网卡配置多个 IP 的场景，例如：

- Keepalived 通过配置额外的 IP 地址对，可以帮助实现快速故障切换和灵活的负载均衡架构

1.1 自动分配地址 VIP

如果只是为了预留若干 IP 而对 IP 地址本身没有要求可以使用下面的 yaml 进行创建：

```
apiVersion: kubeovn.io/v1
kind: Vip
metadata:
  name: vip-dynamic-01
spec:
  subnet: ovn-default
  type: ""
```

- `subnet`：将从该 Subnet 中预留 IP。
- `type`：目前支持两种类型，为空表示仅用于 ipam ip 占位，`switch_lb_vip` 表示该 vip 仅用于 switch lb 前端 vip 和后端 ip 需处于同一子网。

创建成功后查询该 VIP：

```
# kubectl get vip
NAME             V4IP         PV4IP   MAC                PMAC   V6IP   PV6IP   SUBNET        READY
vip-dynamic-01   10.16.0.12           00:00:00:F0:DB:25                       ovn-default   true
```

可见该 VIP 被分配了 `10.16.0.12` 的 IP 地址，该地址可以之后供其他网络基础设施使用。

1.2 使用固定地址 VIP

如对预留的 VIP 的 IP 地址有需求可使用下面的 yaml 进行固定分配：

```
apiVersion: kubeovn.io/v1
kind: Vip
metadata:
  name: static-vip01
spec:
  subnet: ovn-default
  v4ip: "10.16.0.121"
```

- `subnet`：将从该 Subnet 中预留 IP。
- `v4ip`：固定分配的 IP 地址，该地址需在 `subnet` 的 CIDR 范围内。

创建成功后查询该 VIP：

```
# kubectl get vip
NAME             V4IP          PV4IP   MAC                PMAC   V6IP   PV6IP   SUBNET        READY
static-vip01   10.16.0.121           00:00:00:F0:DB:26                       ovn-default   true
```

可见该 VIP 被分配了所预期的 IP 地址。

2025 Kube-OVN Team

1.3 Pod 绑定 VIP 和 AAP

Pod 通过 annotation 绑定 VIP 和 AAP 的。Pod 的 labels 匹配 VIP 的 `selector` 来绑定的。

Pod annotation 中指定 VIP，指定的是ovn.kubernetes.io/aaps: vip-aap,vip-aap2,vip-aap3

AAP 可以绑定多个 Pod 也可以一个AAP 给 Pod 绑定 VIP 和 subnet 中的 Port 的。

1.3.1 创建 VIP 和 AAP

```
apiVersion: kubeovn.io/v1
kind: Vip
metadata:
  name: vip-aap
spec:
  subnet: ovn-default
  namespace: default
  selector:
    - "app: aap1"
```

VIP 的创建有两个参数可以选择

- `namespace` : AAP 可以为VIP 指定命名空间，VIP 只能被同一个命名空间的 AAP 绑定
- `selector` : AAP 可以指定 VIP 可以为 Pod 指定命名空间 `namespace` 和 Pod指定的标签 `key: value` 匹配，可以指定多个，多个为 AND 关系

创建完成后 VIP 中的 Port：

```
# kubectl ko nbctl show ovn-default
switch e32e1d3b-c539-45f4-ab19-be4e33a061f6 (ovn-default)
    port aap-vip
        type: virtual
```

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  annotations:
    ovn.kubernetes.io/aaps: vip-aap
  labels:
    app: aap1
spec:
  containers:
    - name: busybox
      image: busybox
      command: ["sleep", "3600"]
      securityContext:
        capabilities:
          add:
            - NET_ADMIN
```

创建完成后 AAP 的信息：

```
# kubectl ko nbctl list logical_switch_port aap-vip
_uuid               : cd930750-0533-4f06-a6c0-217ddac73272
addresses           : []
dhcpv4_options      : []
dhcpv6_options      : []
dynamic_addresses   : []
enabled             : []
external_ids        : {ls=ovn-default, vendor=kube-ovn}
ha_chassis_group    : []
mirror_rules        : []
name                : aap-vip
options             : {virtual-ip="10.16.0.100", virtual-parents="busybox.default"}
parent_name         : []
port_security       : []
tag                 : []
tag_request         : []
type                : virtual
up                  : false
```

virtual-ip 中指定 VIP 的 IP，virtual-parents 指定了 AAP 绑定 Pod 的 Port：

创建完成后 Pod 的信息：

```
# kubectl exec -it busybox -- ip addr add 10.16.0.100/16 dev eth0
# kubectl exec -it busybox01 -- ip addr show eth0
35: eth0@if36: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1400 qdisc noqueue
    link/ether 00:00:00:e2:ab:0c brd ff:ff:ff:ff:ff:ff
    inet 10.16.0.7/16 brd 10.16.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

```
    inet 10.16.0.100/16 scope global secondary eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fee2:ab0c/64 scope link
       valid_lft forever preferred_lft forever
```

除 Pod 创建时自动分配的 IP，VIP 的 IP 也被成功绑定，并且当前 subnet 内的其它 Pod 能和这两个 IP 进行通信。

7.15.2 2. Switch LB rule vip

```yaml
apiVersion: kubeovn.io/v1
kind: Vip
metadata:
  name: slr-01
spec:
  subnet: ovn-default
  type: switch_lb_vip
```

- `subnet`：将从该 Subnet 中预留 IP。
- `type`：目前支持两种类型，为空表示仅用于 ipam ip 占位，`switch_lb_vip` 表示该 vip 仅用于 switch lb 前端 vip 和后端 ip 需处于同一子网。

7.15.3 3. 从 v1.17 之前的版本升级

v1.17 移除了通过 `ovn.kubernetes.io/vip` 为 Pod 固定 IP 的功能，以及 `kube_host_vm_vip` 类型。升级前请删除使用该 annotation 的 Pod 或对应的 VIP，避免旧的 VIP 预留与 Pod 重新分配的 IP 发生冲突。

- 升级后，使用 `ovn.kubernetes.io/vip` 的 Pod 会获得普通 IPAM 地址；该 annotation 会被忽略并记录告警。
- 升级后删除这类 Pod 会释放其 IPAM 地址，但不会释放 VIP，也不会将地址交还给 VIP；其他 Pod 可能先占用该地址。只有 controller 重启且地址仍空闲时，VIP 才可能重新占用该地址。
- `kube_host_vm_vip` 不再创建或维护普通逻辑交换机端口。遗留端口会在两轮垃圾回收后被清理；如果外部工作负载仍在使用，请在升级前改用外部方式管理该端口。
- v1.17 之前 VIP 上遗留的 `ip_reserved` 标签不会自动删除，也不再参与地址预留。旧 Pod 仍占用地址时，Subnet 的 `UsingIPs` 可能暂时比实际使用量高；Pod 及其 IP 记录删除后，这个额外计数会消失，VIP 自身的预留仍会正常计数。

需要为 Pod 预留或固定地址时，请使用 IPPool 或 Pod annotation。AAP 和 Switch LB 仍继续使用 VIP。

7.15.4 4. StatefulSet 与 Kubevirt VM 固定 IP

StatefulSet 和 VM 的 IP 保留机制不依赖 VIP 资源。

对于 `StatefulSet` 和 `VM`，其 Pod 销毁后重新创建时会复用之前分配的 IP。

VM 保留 IP 需要确保 `kube-ovn-controller` 的 `keep-vm-ip` 参数为 `true`。请参考 Kubevirt VM 固定地址开启设置。

PDF Slack Support

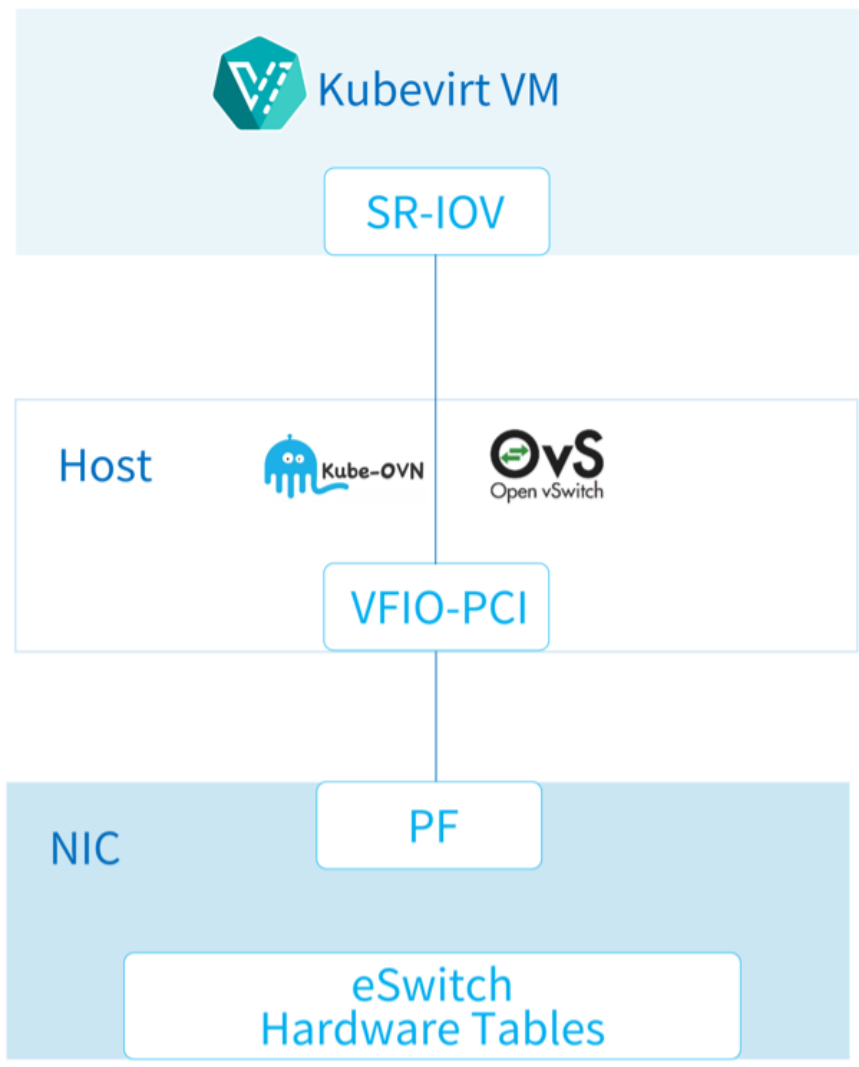
2026年10月28日

2022年5月24日

GitHub

+3

7.15.5 评论



7.16 Mellanox ⬚⬚ Offload ⬚⬚

Kube-OVN 在最终的数据平面使用 OVS 来完成流量转发，相关的流表匹配，隧道封装等功能为 CPU 密集型，在大流量下会消耗大量 CPU 资源并导致 延迟上升和吞吐量下降。Mellanox 的 Accelerated Switching And Packet Processing (ASAP²) 技术可以将 OVS 相关的操作卸载到硬件网卡内的 eSwitch 上执行。该技术可以在无需对 OVS 控制平面进行修改的情况下，缩短数据路径，避免对主机 CPU 资源的使用，大幅降低延迟并显著提升吞吐量。

Note

本文所述方案在 2022 年经过验证，但目前硬件网卡可能已经有了新的特性，当时的一些限制也可能已经被解决。请咨询您的硬件供应商，了解最新的技术限制和能力。

7.16.1 ⬚⬚⬚⬚

- Mellanox CX5/CX6/CX7/BlueField 等支持 ASAP² 的硬件网卡。
- CentOS 8 Stream 或上游 Linux 5.7 以上内核支持。
- 由于当前网卡不支持 `dp_hash` 和 `hash` 操作卸载，需关闭 OVN LB 功能。
- 为了配置卸载模式，网卡不能绑定 bond。

7.16.2 ⬚⬚ SR-IOV ⬚ Device Plugin

Mellanox 网卡支持两种配置 offload 的方式，一种手动配置网卡 SR-IOV 和 Device Plugin，另一种通过 sriov-network-operator 进行自动配置。

手动配置 SR-IOV 和 Device Plugin

⬚⬚ SR-IOV

查询网卡的设备 ID，下面的例子中为 `84:00.0` 和 `84.00.1`：

```
# lspci -nn | grep ConnectX-5
84:00.0 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5] [15b3:1017]
84:00.1 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5] [15b3:1017]
```


2025 Kube-OVN Team

根据设备 ID 找到对应网卡:

```
# ls -l /sys/class/net/ | grep 84:00.0
lrwxrwxrwx 1 root root 0 Feb 4 16:16 enp132s0f0np0 -> ../../devices/pci0000:80/0000:80:08.0/0000:84:00.0/net/enp132s0f0np0
# ls -l /sys/class/net/ | grep 84:00.1
lrwxrwxrwx 1 root root 0 Feb 4 16:16 enp132s0f1np1 -> ../../devices/pci0000:80/0000:80:08.0/0000:84:00.1/net/enp132s0f1np1
```

检查网卡是否绑定 bond:

> 本示例中网卡 enp132s0f0np0 和 enp132s0f1np1 绑定 bond1

```
# ip link show enp132s0f0np0 | grep bond
160: enp132s0f0np0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master bond1 state UP mode DEFAULT group default qlen 1000
# ip link show enp132s0f1np1 | grep bond
169: enp132s0f1np1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master bond1 state UP mode DEFAULT group default qlen 1000
```

移除 bond 和现有的 VF:

```
ifenslave -d bond1 enp132s0f0np0
ifenslave -d bond1 enp132s0f1np1
echo 0 > /sys/class/net/enp132s0f0np0/device/sriov_numvfs
echo 0 > /sys/class/net/enp132s0f1np1/device/sriov_numvfs
ip link set enp132s0f0np0 down
ip link set enp132s0f1np1 down
```

配置规则下发模式:

> OVS 内核支持两种规则插入硬件的模式: SMFS 和 DMFS

- SMFS (software-managed flow steering): 默认模式，规则由软件（驱动程序）直接插入硬件。这种模式对规则插入进行了优化。
- DMFS (device-managed flow steering): 规则插入是通过固件命令完成的。该模式针对系统中少量规则的吞吐量进行了优化。

可在支持该模式的内核中通过 sysfs 或 devlink API 进行配置:

```
#    sysfs
echo <smfs|dmfs> > /sys/class/net/enp132s0f0np0/compat/devlink/steering_mode
echo <smfs|dmfs> > /sys/class/net/enp132s0f1np1/compat/devlink/steering_mode
#    devlink
devlink dev param set pci/84:00.0 name flow_steering_mode value smfs cmode runtime
devlink dev param set pci/84:00.1 name flow_steering_mode value smfs cmode runtime
```

> 注意: 若不了解应该选择哪个模式，则可使用默认模式，无需进行配置。

检查可用 VF 数量:

```
# cat /sys/class/net/enp132s0f0np0/device/sriov_totalvfs
127
# cat /sys/class/net/enp132s1f0np1/device/sriov_totalvfs
127
```

创建 VF，总数不要超过上面查询出的数量:

```
# echo '4' > /sys/class/net/enp132s0f0np0/device/sriov_numvfs
# echo '4' > /sys/class/net/enp132s1f0np1/device/sriov_numvfs
# ip link show enp132s0f0np0
160: enp132s0f0np0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT group default qlen 1000
    link/ether 08:c0:eb:74:c3:4a brd ff:ff:ff:ff:ff:ff
    vf 0 link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state disable, trust off, query_rss off
    vf 1 link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state disable, trust off, query_rss off
    vf 2 link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state disable, trust off, query_rss off
    vf 3 link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state disable, trust off, query_rss off
# ip link show enp132s0f1np1
169: enp132s0f1np1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT group default qlen 1000
    link/ether 08:c0:eb:74:c3:4b brd ff:ff:ff:ff:ff:ff
    vf 0 link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state disable, trust off, query_rss off
    vf 1 link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state disable, trust off, query_rss off
    vf 2 link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state disable, trust off, query_rss off
    vf 3 link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state disable, trust off, query_rss off
# ip link set enp132s0f0np0 up
# ip link set enp132s0f1np1 up
```

找到上述 VF 对应的设备 ID:

```
# lspci -nn | grep ConnectX-5 | grep Virtual
84:00.2 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
84:00.3 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
84:00.4 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
```

```
84:00.5 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
84:00.6 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
84:00.7 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
84:01.0 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
84:01.1 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
```

▯ VF ▯▯▯▯▯▯▯

```
echo 0000:84:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:84:00.3 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:84:00.4 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:84:00.5 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:84:00.6 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:84:00.7 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:84:01.0 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:84:01.1 > /sys/bus/pci/drivers/mlx5_core/unbind
```

▯▯ eSwitch ▯▯▯▯▯▯▯▯▯▯▯▯

```
devlink dev eswitch set pci/0000:84:00.0 mode switchdev
devlink dev eswitch set pci/0000:84:00.1 mode switchdev
ethtool -K enp132s0f0np0 hw-tc-offload on
ethtool -K enp132s0f1np1 hw-tc-offload on
```

SR-IOV VF ▯▯▯▯▯▯▯▯

SR-IOV VF LAG ▯▯▯▯▯ PF ▯▯ OVS ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ e-switch ▯▯▯▯▯ bond ▯▯▯▯▯

- Active-backup
- XOR
- LACP

> SR-IOV VF LAG ▯▯ LAG ▯▯▯▯▯▯▯▯▯▯bond ▯▯▯▯▯▯▯▯ bond PF ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ VF▯▯▯ VF ▯▯▯▯▯▯ bond ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ PF ▯▯▯▯▯ ▯▯▯▯▯▯ VF ▯▯▯▯▯▯▯▯ PF▯▯ XOR ▯ LACP ▯▯▯▯▯▯▯▯ PF ▯▯▯▯▯▯▯▯▯▯▯▯ VF ▯▯▯▯▯▯▯▯▯▯ PF ▯▯▯▯▯

▯▯▯▯▯▯▯▯ LACP ▯▯▯▯▯▯▯▯▯▯▯▯

```
modprobe bonding mode=802.3ad
ip link set enp132s0f0np0 master bond1
ip link set enp132s0f1np1 master bond1
ip link set enp132s0f0np0 up
ip link set enp132s0f1np1 up
ip link set bond1 up
```

> ▯▯▯▯▯▯▯▯▯▯ bond▯▯▯▯▯▯▯▯▯▯

▯▯▯▯▯▯▯▯▯▯ VF ▯▯▯

```
echo 0000:84:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:84:00.3 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:84:00.4 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:84:00.5 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:84:00.6 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:84:00.7 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:84:01.0 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:84:01.1 > /sys/bus/pci/drivers/mlx5_core/bind
```

`NetworkManager` ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `NetworkManager` ▯▯▯▯▯▯▯

```
systemctl stop NetworkManager
systemctl disable NetworkManager
```

▯▯ DEVICE PLUGIN

▯▯▯▯▯▯▯▯ VF ▯▯▯▯▯▯▯▯▯▯▯▯▯ Pod ▯▯▯ VF ▯▯▯▯▯▯▯▯▯▯ SR-IOV Device Plugin ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯

▯▯ SR-IOV ▯▯ Configmap▯

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: sriovdp-config
  namespace: kube-system
data:
  config.json: |
```

```
    {
      "resourceList": [{
          "resourcePrefix": "mellanox.com",
          "resourceName": "cx5_sriov_switchdev",
          "selectors": {
                  "vendors": ["15b3"],
                  "devices": ["1018"],
                  "drivers": ["mlx5_core"]
              }
      }
      ]
    }
```

SR-IOV Device Plugin ▯▯▯▯▯▯ ConfigMap ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ConfigMap ▯ name ▯ sriovdp-config▯

- `selectors` : VF ▯▯▯
- `vendors` : ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
- `devices` : ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
- `drivers` : ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

`selectors` ▯▯▯▯▯ `pciAddresses` ▯ `acpiIndexes` ▯▯▯▯▯ VF ▯▯▯▯▯▯▯▯▯▯▯▯▯▯SR-IOV ConfigMap ▯▯

▯▯ SR-IOV ▯▯▯▯▯▯

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/sriov-network-device-plugin/v3.6.2/deployments/sriovdp-daemonset.yaml
```

▯▯ SR-IOV ▯▯▯▯▯▯▯▯▯ Kubernetes Node ▯▯

```
kubectl describe node kube-ovn-01  | grep mellanox

mellanox.com/cx5_sriov_switchdev:  8
mellanox.com/cx5_sriov_switchdev:  8
mellanox.com/cx5_sriov_switchdev  0           0
```

▯▯ sriov-network-operator ▯▯ SR-IOV ▯ Device Plugin

▯▯ node-feature-discovery▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
kubectl apply -k https://github.com/kubernetes-sigs/node-feature-discovery/deployment/overlays/default?ref=v0.11.3
```

▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ offload ▯▯▯▯▯▯▯▯ annotation:

```
kubectl label nodes [offloadNicNode] feature.node.kubernetes.io/network-sriov.capable=true
```

▯▯▯▯▯▯▯▯▯▯ Operator▯

```
git clone --depth=1 https://github.com/kubeovn/sriov-network-operator.git
kubectl apply -k sriov-network-operator/deploy
```

▯▯ Operator ▯▯▯▯▯▯▯▯▯▯

```
# kubectl get -n kube-system all | grep sriov
NAME                                          READY   STATUS    RESTARTS   AGE
pod/sriov-network-config-daemon-bf9nt         1/1     Running   0          8s
pod/sriov-network-operator-54d7545f65-296gb   1/1     Running   0          10s

NAME                                        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR                                              AGE
daemonset.apps/sriov-network-config-daemon   1         1         1       1            1           beta.kubernetes.io/os=linux,feature.node.kubernetes.io/
network-sriov.capable=true   8s

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/sriov-network-operator   1/1     1            1           10s

NAME                                               DESIRED   CURRENT   READY   AGE
replicaset.apps/sriov-network-operator-54d7545f65   1         1         1       10s
```

▯▯ `SriovNetworkNodeState` ▯▯▯▯ `node1` ▯▯▯▯▯▯▯▯▯▯▯▯ Mellanox ▯▯▯

```
# kubectl get sriovnetworknodestates.sriovnetwork.openshift.io -n kube-system node1 -o yaml
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodeState
spec: ...
```

```
status:
  interfaces:
  - deviceID: "1017"
    driver: mlx5_core
    mtu: 1500
    pciAddress: "0000:5f:00.0"
    totalvfs: 8
    vendor: "15b3"
    linkSeed: 25000Mb/s
    linkType: ETH
    mac: 08:c0:eb:f4:85:bb
    name: ens41f0np0
  - deviceID: "1017"
    driver: mlx5_core
    mtu: 1500
    pciAddress: "0000:5f:00.1"
    totalvfs: 8
    vendor: "15b3"
    linkSeed: 25000Mb/s
    linkType: ETH
    mac: 08:c0:eb:f4:85:bb
    name: ens41f1np1
```

 `SriovNetworkNodePolicy` `nicSelector`

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: policy
  namespace: kube-system
spec:
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"
  eSwitchMode: switchdev
  numVfs: 3
  nicSelector:
    pfNames:
    - ens41f0np0
    - ens41f1np1
  resourceName: cx_sriov_switchdev
```

 `SriovNetworkNodeState` `status`

```
# kubectl get sriovnetworknodestates.sriovnetwork.openshift.io -n kube-system node1 -o yaml

...
spec:
  interfaces:
  - eSwitchMode: switchdev
    name: ens41f0np0
    numVfs: 3
    pciAddress: 0000:5f:00.0
    vfGroups:
    - policyName: policy
      vfRange: 0-2
      resourceName: cx_sriov_switchdev
  - eSwitchMode: switchdev
    name: ens41f1np1
    numVfs: 3
    pciAddress: 0000:5f:00.1
    vfGroups:
    - policyName: policy
      vfRange: 0-2
      resourceName: cx_sriov_switchdev
status:
  interfaces
  - Vfs:
    - deviceID: 1018
      driver: mlx5_core
      pciAddress: 0000:5f:00.2
      vendor: "15b3"
    - deviceID: 1018
      driver: mlx5_core
      pciAddress: 0000:5f:00.3
      vendor: "15b3"
    - deviceID: 1018
      driver: mlx5_core
      pciAddress: 0000:5f:00.4
      vendor: "15b3"
    deviceID: "1017"
    driver: mlx5_core
    linkSeed: 25000Mb/s
    linkType: ETH
    mac: 08:c0:eb:f4:85:ab
    mtu: 1500
    name: ens41f0np0
    numVfs: 3
    pciAddress: 0000:5f:00.0
    totalvfs: 3
    vendor: "15b3"
  - Vfs:
```

```
    - deviceID: 1018
      driver: mlx5_core
      pciAddress: 0000:5f:00.5
      vendor: "15b3"
    - deviceID: 1018
      driver: mlx5_core
      pciAddress: 0000:5f:00.6
      vendor: "15b3"
    - deviceID: 1018
      driver: mlx5_core
      pciAddress: 0000:5f:00.7
      vendor: "15b3"
    deviceID: "1017"
    driver: mlx5_core
    linkSeed: 25000Mb/s
    linkType: ETH
    mac: 08:c0:eb:f4:85:bb
    mtu: 1500
    name: ens41f1np1
    numVfs: 3
    pciAddress: 0000:5f:00.1
    totalvfs: 3
    vendor: "15b3"
```

▯▯ VF ▯▯▯▯

```
# lspci -nn | grep ConnectX
5f:00.0 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5] [15b3:1017]
5f:00.1 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5] [15b3:1017]
5f:00.2 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
5f:00.3 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
5f:00.4 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
5f:00.5 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
5f:00.6 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
5f:00.7 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function] [15b3:1018]
```

▯▯ PF ▯▯▯▯▯

```
# cat /sys/class/net/ens41f0np0/compat/devlink/mode
switchdev
```

7.16.3 ▯▯ Multus-CNI

SR-IOV Device Plugin ▯▯▯▯▯▯▯▯ ID ▯▯▯▯ Multus-CNI ▯▯▯ Kube-OVN▯▯▯▯▯▯▯ Multus-CNI ▯▯▯▯▯▯▯▯▯

▯▯ Multus-CNI ▯▯▯▯▯▯▯

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/multus-cni/v4.0.2/deployments/multus-daemonset-thick.yml
```

> ▯▯▯multus ▯▯▯ Thin ▯ Thick ▯▯▯▯▯▯▯▯▯▯ SR-IOV ▯▯▯▯▯ Thick ▯▯▯

▯▯ `NetworkAttachmentDefinition` ▯

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: sriov
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/resourceName: mellanox.com/cx5_sriov_switchdev
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "kube-ovn",
    "plugins":[
        {
            "type":"kube-ovn",
            "server_socket":"/run/openvswitch/kube-ovn-daemon.sock",
            "provider": "sriov.default.ovn"
        },
        {
            "type":"portmap",
            "capabilities":{
                "portMappings":true
            }
        }
    ]
}'
```

- `provider`：▯▯▯▯▯ `NetworkAttachmentDefinition` ▯ {name}.{namespace}.ovn▯

7.16.4 Overlay ⬚⬚

Kube-OVN ⬚⬚⬚⬚⬚⬚⬚

⬚⬚⬚⬚⬚⬚⬚

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/dist/images/install.sh
```

⬚⬚⬚⬚⬚⬚⬚ `IFACE` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ IP⬚

```
ENABLE_MIRROR=${ENABLE_MIRROR:-false}
HW_OFFLOAD=${HW_OFFLOAD:-true}
ENABLE_LB=${ENABLE_LB:-false}
IFACE="bond1"
#       SR-IOV   Device Plugin           bond    IFACE     bond1      bond     IFACE     enp132s0f0np0   enp132s0f1np1
```

⬚⬚ Kube-OVN⬚

```
bash install.sh
```

⬚⬚⬚⬚ **VF ⬚⬚⬚ Pod**

⬚⬚⬚⬚⬚⬚ yaml ⬚⬚⬚⬚⬚⬚ VF ⬚⬚⬚⬚⬚⬚⬚⬚⬚ Pod:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-overlay
  annotations:
    v1.multus-cni.io/default-network: default/sriov
    sriov.default.ovn.kubernetes.io/logical_switch: ovn-default
spec:
  containers:
  - name: nginx-overlay
    image: docker.io/library/nginx:alpine
    resources:
      requests:
        mellanox.com/cx5_sriov_switchdev: '1'
      limits:
        mellanox.com/cx5_sriov_switchdev: '1'
```

- `v1.multus-cni.io/default-network`：⬚⬚⬚⬚⬚⬚ `NetworkAttachmentDefinition` ⬚ {namespace}/{name}⬚
- `sriov.default.ovn.kubernetes.io/logical_switch`：⬚⬚ Pod ⬚⬚⬚⬚⬚⬚⬚⬚ Pod ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

7.16.5 Underlay ⬚⬚

Kube-OVN ⬚⬚⬚⬚⬚⬚⬚

⬚⬚⬚⬚⬚⬚⬚

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/dist/images/install.sh
```

⬚⬚⬚⬚⬚⬚⬚ `IFACE` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ IP⬚

```
ENABLE_MIRROR=${ENABLE_MIRROR:-false}
HW_OFFLOAD=${HW_OFFLOAD:-true}
ENABLE_LB=${ENABLE_LB:-false}
IFACE=""
#     Underlay    IFACE         PF      IFACE          K8s                 PF
```

⬚⬚ Kube-OVN⬚

```
bash install.sh
```

⬚⬚⬚⬚ **VF ⬚⬚⬚ Pod**

⬚⬚⬚⬚⬚⬚ yaml ⬚⬚⬚⬚⬚⬚ VF ⬚⬚⬚⬚⬚⬚⬚⬚⬚ Pod:

2025 Kube-OVN Team

```
apiVersion: kubeovn.io/v1
kind: ProviderNetwork
metadata:
  name: underlay-offload
spec:
  defaultInterface: bond1

---
apiVersion: kubeovn.io/v1
kind: Vlan
metadata:
  name: vlan0
spec:
  id: 0
  provider: underlay-offload

---
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: vlan0
spec:
  protocol: IPv4
  provider: ovn
  cidrBlock: 10.10.204.0/24
  gateway: 10.10.204.254
  vlan: vlan0
  excludeIps:
  - 10.10.204.1..10.10.10.204.100

---
apiVersion: v1
kind: Pod
metadata:
  name: nginx-underlay
  annotations:
    k8s.v1.cni.cncf.io/networks: '[{
      "name": "sriov",
      "namespace": "default",
      "default-route": ["10.10.204.254"]
    }]'
    sriov.default.ovn.kubernetes.io/logical_switch: vlan0
spec:
  containers:
  - name: nginx-underlay
    image: docker.io/library/nginx:alpine
    resources:
      requests:
        mellanox.com/cx5_sriov_switchdev: '1'
      limits:
        mellanox.com/cx5_sriov_switchdev: '1'
```

- `v1.multus-cni.io/default-network` : ... `NetworkAttachmentDefinition` ... {namespace}/{name}...

> ... multus ... VF ... Pod... VF ... Pod ... VF ... Pod ... multus ...

... yaml ... VF ... Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-underlay-noVF
  annotations:
    ovn.kubernetes.io/logical_switch: vlan0
spec:
  containers:
  - name: nginx-underlay-noVF
    image: docker.io/library/nginx:alpine
```

... VF ... Pod... ovs-kernel ... e-switch ...

7.16.6 ...

... Pod ... `ovs-ovn` ...

```
# ovs-appctl dpctl/dump-flows -m type=offloaded
ufid:91cc45de-e7e9-4935-8f82-1890430b0f66, skb_priority(0/0),skb_mark(0/0),ct_state(0/0x23),ct_zone(0/0),ct_mark(0/0),ct_label(0/0x1),recirc_id(0),dp_hash(0/
0),in_port(5b45c61b307e_h),packet_type(ns=0/0,id=0/
0),eth(src=00:00:00:c5:6d:4e,dst=00:00:00:e7:16:ce),eth_type(0x0800),ipv4(src=0.0.0.0/0.0.0.0,dst=0.0.0.0/0.0.0.0,proto=0/0,tos=0/0,ttl=0/0,frag=no),
packets:941539, bytes:62142230, used:0.260s, offloaded:yes, dp:tc, actions:54235e5753b8_h
ufid:e00768d7-e652-4d79-8182-3291d852b791, skb_priority(0/0),skb_mark(0/0),ct_state(0/0x23),ct_zone(0/0),ct_mark(0/0),ct_label(0/0x1),recirc_id(0),dp_hash(0/
0),in_port(54235e5753b8_h),packet_type(ns=0/0,id=0/
0),eth(src=00:00:00:e7:16:ce,dst=00:00:00:c5:6d:4e),eth_type(0x0800),ipv4(src=0.0.0.0/0.0.0.0,dst=0.0.0.0/0.0.0.0,proto=0/0,tos=0/0,ttl=0/0,frag=no),
packets:82386659, bytes:115944854173, used:0.260s, offloaded:yes, dp:tc, actions:5b45c61b307e_h
```

▯▯▯ `offloaded:yes, dp:tc` ▯▯▯▯▯▯▯▯▯▯

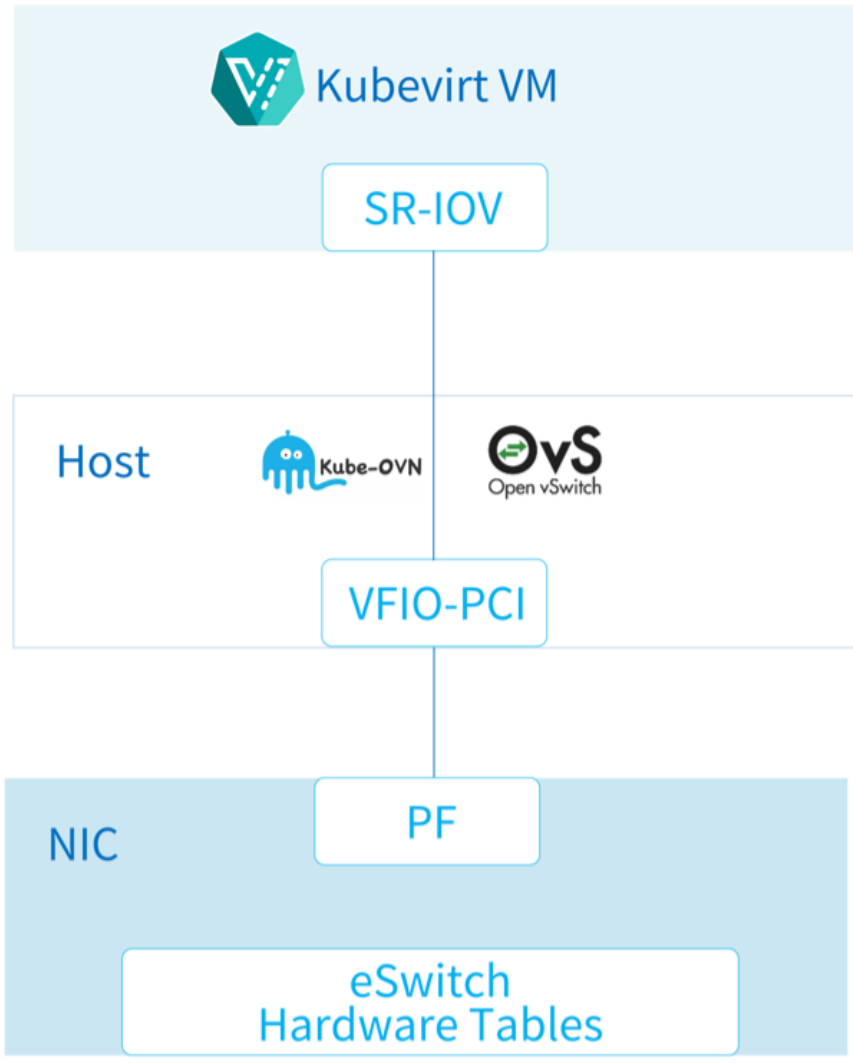
PDF Slack Support

2025▯9▯10▯

2022▯5▯24▯

GitHub 

7.16.7 ▯▯



7.17 □□□□□ Offload □□

Kube-OVN 在最终的数据平面使用 OVS 来完成流量转发，相关的流表匹配，隧道封装等功能为 CPU 密集型，在大流量下会消耗大量 CPU 资源并导致 延迟上升和吞吐量下降。芯启源的 Agilio CX 系列智能网卡可以将 OVS 相关的操作卸载到硬件网卡中执行。该技术可以在无需对 OVS 控制平面进行修改的情况下，缩短数据路径，避免对主机 CPU 资源的使用，大幅降低延迟并显著提升吞吐量。

Note

本文所述方案在 2022 年经过验证，但目前硬件网卡可能已经有了新的特性，当时的一些限制也可能已经被解决。请咨询您的硬件供应商，了解最新的技术限制和能力。

7.17.1 前置条件

- 芯启源 Agilio CX 系列的硬件网卡。
- CentOS 8 Stream 或上游 Linux 5.7 以上内核支持。
- 由于当前网卡不支持 `dp_hash` 和 `hash` 操作卸载，需关闭 OVN LB 功能。

7.17.2 设□网卡 SR-IOV □式

用户可参考 Agilio Open vSwitch TC User Guide 获得该网卡使用的更多详细信息。

保存下列脚本用于后续执行固件相关操作:

```
#!/bin/bash
DEVICE=${1}
DEFAULT_ASSY=scan
ASSY=${2:-${DEFAULT_ASSY}}
APP=${3:-flower}

if [ "x${DEVICE}" = "x" -o ! -e /sys/class/net/${DEVICE} ]; then
    echo Syntax: ${0} device [ASSY] [APP]
    echo
    echo This script associates the TC Offload firmware
    echo with a Netronome SmartNIC.
    echo
    echo device: is the network device associated with the SmartNIC
```


2025 Kube-OVN Team

```
    echo ASSY: defaults to ${DEFAULT_ASSY}
    echo APP: defaults to flower. flower-next is supported if updated
    echo      firmware has been installed.
    exit 1
fi

# It is recommended that the assembly be determined by inspection
# The following code determines the value via the debug interface
if [ "${ASSY}x" = "scanx" ]; then
    ethtool -W ${DEVICE} 0
    DEBUG=$(ethtool -w ${DEVICE} data /dev/stdout | strings)
    SERIAL=$(echo "${DEBUG}" | grep "^SN:")
    ASSY=$(echo ${SERIAL} | grep -oE AMDA[0-9]{4})
fi

PCIADDR=$(basename $(readlink -e /sys/class/net/${DEVICE}/device))
FWDIR="/lib/firmware/netronome"

# AMDA0081 and AMDA0097 uses the same firmware
if [ "${ASSY}" = "AMDA0081" ]; then
    if [ ! -e ${FWDIR}/${APP}/nic_AMDA0081.nffw ]; then
       ln -sf nic_AMDA0097.nffw ${FWDIR}/${APP}/nic_AMDA0081.nffw
   fi
fi

FW="${FWDIR}/pci-${PCIADDR}.nffw"
ln -sf "${APP}/nic_${ASSY}.nffw" "${FW}"

# insert distro-specific initramfs section here...
```

[redacted]

```
./agilio-tc-fw-select.sh ens47np0 scan
rmmod nfp
modprobe nfp
```

[redacted] VF [redacted] VF[redacted]

```
# cat /sys/class/net/ens3/device/sriov_totalvfs
65

# echo 4 > /sys/class/net/ens47/device/sriov_numvfs
```

7.17.3 [redacted] SR-IOV Device Plugin

[redacted] VF [redacted] Pod [redacted] VF [redacted] SR-IOV Device Plugin [redacted] [redacted]

[redacted] SR-IOV [redacted] Configmap[redacted]

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: sriovdp-config
  namespace: kube-system
data:
  config.json: |
    {
      "resourceList": [{
          "resourcePrefix": "corigine.com",
          "resourceName": "agilio_sriov",
          "selectors": {
                  "vendors": ["19ee"],
                  "devices": ["6003"],
                  "drivers": ["nfp_netvf"]
              }
      }
      ]
    }
```

[redacted] SR-IOV Network Device Plugin [redacted][redacted] intel/sriov-network-device-plugin [redacted] k8snetworkplumbingwg[redacted]

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/sriov-network-device-plugin/master/deployments/sriovdp-daemonset.yaml
```

[redacted] SR-IOV [redacted] Kubernetes Node [redacted]

```
kubectl describe no containerserver  | grep corigine

corigine.com/agilio_sriov:  4
corigine.com/agilio_sriov:  4
corigine.com/agilio_sriov  0           0
```

7.17.4 □□ Multus-CNI

SR-IOV Device Plugin □□□□□□□□ ID □□□□ Multus-CNI □□□ Kube-OVN□□□□□□□ Multus-CNI □□□□□□□□□

□□ Multus-CNI □□□□□□□

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/multus-cni/master/deployments/multus-daemonset.yml
```

□□ `NetworkAttachmentDefinition` □

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: default
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/resourceName: corigine.com/agilio_sriov
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "kube-ovn",
    "plugins":[
        {
            "type":"kube-ovn",
            "server_socket":"/run/openvswitch/kube-ovn-daemon.sock",
            "provider": "default.default.ovn"
        },
        {
            "type":"portmap",
            "capabilities":{
                "portMappings":true
            }
        }
    ]
}'
```

- `provider` : □□□□□ `NetworkAttachmentDefinition` □ {name}.{namespace}.ovn□

7.17.5 Kube-OVN □□□□□□□

□□□□□□□

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/dist/images/install.sh
```

□□□□□□□ `IFACE` □□□□□□□□□□□□□□□□□□□ IP□

```
ENABLE_MIRROR=${ENABLE_MIRROR:-false}
HW_OFFLOAD=${HW_OFFLOAD:-true}
ENABLE_LB=${ENABLE_LB:-false}
IFACE="ensp01"
```

□□ Kube-OVN□

```
bash install.sh
```

7.17.6 □□□□ VF □□□ Pod

□□□□□□ yaml □□□□□□ VF □□□□□□□□□ Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: default
  annotations:
    v1.multus-cni.io/default-network: default/default
spec:
  containers:
    - name: nginx
      image: docker.io/library/nginx:alpine
      resources:
        requests:
          corigine.com/agilio_sriov: '1'
```


2025 Kube-OVN Team

```
        limits:
          corigine.com/agilio_sriov: '1'
```

- `v1.multus-cni.io/default-network`：⬚⬚⬚⬚⬚⬚ `NetworkAttachmentDefinition` ⬚ {namespace}/{name}⬚

⬚⬚⬚⬚ Pod ⬚⬚⬚⬚⬚ `ovs-ovn` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
# ovs-appctl dpctl/dump-flows -m type=offloaded
ufid:91cc45de-e7e9-4935-8f82-1890430b0f66, skb_priority(0/0),skb_mark(0/0),ct_state(0/0x23),ct_zone(0/0),ct_mark(0/0),ct_label(0/0x1),recirc_id(0),dp_hash(0/
0),in_port(5b45c61b307e_h),packet_type(ns=0/0,id=0/
0),eth(src=00:00:00:c5:6d:4e,dst=00:00:00:e7:16:ce),eth_type(0x0800),ipv4(src=0.0.0.0/0.0.0.0,dst=0.0.0.0/0.0.0.0,proto=0/0,tos=0/0,ttl=0/0,frag=no),
packets:941539, bytes:62142230, used:0.260s, offloaded:yes, dp:tc, actions:54235e5753b8_h
ufid:e00768d7-e652-4d79-8182-3291d852b791, skb_priority(0/0),skb_mark(0/0),ct_state(0/0x23),ct_zone(0/0),ct_mark(0/0),ct_label(0/0x1),recirc_id(0),dp_hash(0/
0),in_port(54235e5753b8_h),packet_type(ns=0/0,id=0/
0),eth(src=00:00:00:e7:16:ce,dst=00:00:00:c5:6d:4e),eth_type(0x0800),ipv4(src=0.0.0.0/0.0.0.0,dst=0.0.0.0/0.0.0.0,proto=0/0,tos=0/0,ttl=0/0,frag=no),
packets:82386659, bytes:115944854173, used:0.260s, offloaded:yes, dp:tc, actions:5b45c61b307e_h
```

⬚⬚⬚ `offloaded:yes, dp:tc` ⬚⬚⬚⬚⬚⬚⬚⬚⬚

PDF Slack Support

2026⬚4⬚28⬚

2022⬚5⬚24⬚

GitHub



7.17.7 ⬚⬚

7.18 云脉芯联网卡 Offload 支持

Kube-OVN 在最终的数据平面使用 OVS 来完成流量转发，相关的流表匹配，隧道封装等功能为 CPU 密集型，在大流量下会消耗大量 CPU 资源并导致 延迟上升和吞吐量下降。云脉芯联的 metaScale 系列智能网卡可以将 OVS 相关的操作卸载到硬件网卡上执行。该技术可以在无需对 OVS 控制平面进行修改的情况下，缩短数据路径，避免对主机 CPU 资源的使用，大幅降低延迟并显著提升吞吐量。

> **Note**
>
> 1. 本文所述方案在 2024 年经过验证，但目前硬件网卡可能已经有了新的特性，当时的一些限制也可能已经被解决。请咨询您的硬件供应商，了解最新的技术限制和能力。
> 2. 目前云脉芯联只适配了 1.11 系列版本的 Kube-OVN，部分最新功能无法使用。

7.18.1 [illegible]

- 云脉芯联 metaScale 系列智能网卡。
- MCR 驱动安装包。
- BIOS 开启 SR-IOV 和 VT-d。

7.18.2 [illegible]

启用 **hw-offload** 模式安装 **Kube-OVN**

1. 下载 安装脚本

```
wget https://raw.githubusercontent.com/yunsilicon/kube-ovn/release-1.11/dist/images/install.sh
```

1. 配置节点

修改每个节点的 `/opt/ovs-config/ovs-dpdk-config` 文件:

```
# specify log level for ovs dpdk, the value is info or dbg, default is info
VLOG=info
# specify nic offload, the value is true or false, default is true
HW_OFFLOAD=true
# specify cpu mask for ovs dpdk, not specified by default
CPU_MASK=0x02
# specify socket memory, not specified by default
SOCKET_MEM="2048,2048"
# specify encap IP
ENCAP_IP=6.6.6.208/24
# specify pci device
DPDK_DEV=0000:b3:00.0
# specify mtu, default is 1500
PF_MTU=1500
# specify bond name if bond enabled, not specified by default
BR_PHY_BOND_NAME=bond0
```

1. 安装 Kube-OVN

```
bash install.sh
```

配置 **SR-IOV**

1. 找到 metaScale 设备的设备 ID，下面是 `b3:00.0`：

```
[root@k8s-master ~]# lspci -d 1f67:
b3:00.0 Ethernet controller: Device 1f67:1111 (rev 02)
b3:00.1 Ethernet controller: Device 1f67:1111 (rev 02)
```

1. 找到与设备 ID 相关的接口，下面是 `p3p1`：

2025 Kube-OVN Team

```
ls -l /sys/class/net/ | grep b3:00.0
lrwxrwxrwx 1 root root 0 May  7 16:30 p3p1 -> ../../devices/pci0000:b2/0000:b2:00.0/0000:b3:00.0/net/p3p1
```

1. ▯▯▯▯▯ VF ▯▯▯

```
cat /sys/class/net/p3p1/device/sriov_totalvfs
512
```

1. ▯▯ VF▯

```
echo '10' > /sys/class/net/p3p1/device/sriov_numvfs
```

1. ▯▯ VF ▯▯▯▯▯

```
lspci -d 1f67:
b3:00.0 Ethernet controller: Device 1f67:1111 (rev 02)
b3:00.1 Ethernet controller: Device 1f67:1111 (rev 02)
b3:00.2 Ethernet controller: Device 1f67:1112
b3:00.3 Ethernet controller: Device 1f67:1112
b3:00.4 Ethernet controller: Device 1f67:1112
b3:00.5 Ethernet controller: Device 1f67:1112
b3:00.6 Ethernet controller: Device 1f67:1112
b3:00.7 Ethernet controller: Device 1f67:1112
b3:01.0 Ethernet controller: Device 1f67:1112
b3:01.1 Ethernet controller: Device 1f67:1112
b3:01.2 Ethernet controller: Device 1f67:1112
b3:01.3 Ethernet controller: Device 1f67:1112
```

1. ▯▯ switchdev ▯▯▯

```
devlink dev eswitch set pci/0000:b3:00.0 mode switchdev
```

▯▯ SR-IOV Device Plugin

1. ▯▯ SR-IOV ▯▯ ConfigMap▯

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: sriovdp-config
  namespace: kube-system
data:
  config.json: |
    {
        "resourceList": [{
                "resourceName": "xsc_sriov",
                "resourcePrefix": "yunsilicon.com",
                "selectors": {
                    "vendors": ["1f67"],
                    "devices": ["1012", "1112"]
                }}
        ]
    }
```

1. ▯▯ SR-IOV Device Plugin ▯▯ DevicePlugin▯
2. ▯▯▯▯▯▯▯▯▯▯ SR-IOV ▯▯▯

```
# kubectl describe node <node name> | grep yunsilicon.com/xsc_sriov
  yunsilicon.com/xsc_sriov:  10
  yunsilicon.com/xsc_sriov:  10
  yunsilicon.com/xsc_sriov  0            0
```

▯▯ Multus-CNI

1. ▯▯ Multus-CNI ▯▯▯ Multus-CNI

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/multus-cni/master/deployments/multus-daemonset.yml
```

1. ▯▯ NetworkAttachmentDefinition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
```


2025 Kube-OVN Team

```
metadata:
  name: sriov-net1
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/resourceName: yunsilicon.com/xsc_sriov
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "kube-ovn",
    "plugins":[
        {
            "type":"kube-ovn",
            "server_socket":"/run/openvswitch/kube-ovn-daemon.sock",
            "provider": "sriov-net1.default.ovn"
        },
        {
            "type":"portmap",
            "capabilities":{
                "portMappings":true
            }
        }
    ]
}'
```

... SR-IOV ... Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  annotations:
    v1.multus-cni.io/default-network: default/sriov-net1
spec:
  containers:
    - name: nginx
      image: docker.io/library/nginx:alpine
      resources:
        requests:
          yunsilicon.com/xsc_sriov: '1'
        limits:
          yunsilicon.com/xsc_sriov: '1'
```

Offload ...

... Pod ... `ovs-ovn` ...

```
ovs-appctl dpctl/dump-flows type=offloaded
flow-dump from pmd on cpu core: 9
ct_state(-new+est-rel+rpl+trk),ct_mark(0/
0x3),recirc_id(0x2d277),in_port(15),packet_type(ns=0,id=0),eth(src=00:00:00:9d:fb:1a,dst=00:00:00:ce:cf:b9),eth_type(0x0800),ipv4(dst=10.16.0.14,frag=no),
packets:6, bytes:588, used:7.276s, actions:ct(zone=4,nat),recirc(0x2d278)
ct_state(-new+est-rel-rpl+trk),ct_mark(0/
0x3),recirc_id(0x2d275),in_port(8),packet_type(ns=0,id=0),eth(src=00:00:00:ce:cf:b9,dst=00:00:00:9d:fb:1a),eth_type(0x0800),ipv4(dst=10.16.0.18,frag=no),
packets:5, bytes:490, used:7.434s, actions:ct(zone=6,nat),recirc(0x2d276)
ct_state(-new+est-rel-rpl+trk),ct_mark(0/0x1),recirc_id(0x2d276),in_port(8),packet_type(ns=0,id=0),eth(src=00:00:00:ce:cf:b9,dst=00:00:00:9d:fb:1a/
01:00:00:00:00:00),eth_type(0x0800),ipv4(frag=no), packets:5, bytes:490, used:7.434s, actions:15
recirc_id(0),in_port(15),packet_type(ns=0,id=0),eth(src=00:00:00:9d:fb:1a/
01:00:00:00:00:00,dst=00:00:00:ce:cf:b9),eth_type(0x0800),ipv4(dst=10.16.0.14/255.192.0.0,frag=no), packets:6, bytes:588, used:7.277s,
actions:ct(zone=6,nat),recirc(0x2d277)
recirc_id(0),in_port(8),packet_type(ns=0,id=0),eth(src=00:00:00:ce:cf:b9/01:00:00:00:00:00,dst=00:00:00:9d:fb:1a),eth_type(0x0800),ipv4(dst=10.16.0.18/255.192.0
packets:6, bytes:588, used:7.434s, actions:ct(zone=4,nat),recirc(0x2d275)
ct_state(-new+est-rel+rpl+trk),ct_mark(0/
0x1),recirc_id(0x2d278),in_port(15),packet_type(ns=0,id=0),eth(dst=00:00:00:ce:cf:b9/01:00:00:00:00:00),eth_type(0x0800),ipv4(frag=no), packets:6, bytes:588,
used:7.277s, actions:8
```

...

PDF Slack Support

2026...4...28...

2024...5...29...

GitHub 

2025 Kube-OVN Team

7.18.3 ▯▯

7.19 中科驭数网卡 Offload 支持

> **Note**
>
> 本文所述方案在 2024 年经过验证，但目前硬件网卡可能已经有了新的特性，当时的一些限制也可能已经被解决。请咨询您的硬件供应商，了解最新的技术限制和能力。

7.19.1 前置条件

- 中科驭数 2200E 系列智能网卡
- 中科驭数 HADOS 敏捷异构软件开发平台
- BIOS 开启 SR-IOV

7.19.2 安装流程

配置 SR-IOV

1. 根据中科驭数 2200E 网卡的 vendor ID（1f47），找到本网卡在主机上的设备 ID，以下为（00:0a.0）和（00:0b.0），分别对应 2200E 上的两个光口，可以根据和光纤连接情况选用。

```
lspci | grep 1f47
00:0a.0 Ethernet controller: Device 1f47:1001 (rev 10)
00:0b.0 Ethernet controller: Device 1f47:1001 (rev 10)
```

1. 查看每一个总线对应的设备最大可分配的 VF 数量：

```
cat /sys/bus/pci/devices/0000\:00\:0a.0/sriov_totalvfs
256
```

1. 按需创建 VF，总数不要超过上述查询的 VF 数量：

```
echo 7 > /sys/bus/pci/devices/0000\:00\:0a.0/sriov_numvfs
```

1. 确认 VF 创建情况

```
lspci | grep 1f47
00:0a.0 Ethernet controller: Device 1f47:1001 (rev 10)
00:0a.1 Ethernet controller: Device 1f47:110f (rev 10)
00:0a.2 Ethernet controller: Device 1f47:110f (rev 10)
00:0a.3 Ethernet controller: Device 1f47:110f (rev 10)
00:0a.4 Ethernet controller: Device 1f47:110f (rev 10)
00:0a.5 Ethernet controller: Device 1f47:110f (rev 10)
00:0a.6 Ethernet controller: Device 1f47:110f (rev 10)
00:0a.7 Ethernet controller: Device 1f47:110f (rev 10)
00:0b.0 Ethernet controller: Device 1f47:1001 (rev 10)
```

安装并运行 SR-IOV Device Plugin

1. 创建 SR-IOV 相关 Configmap，便于后续安装的 SR-IOV Device Plugin 根据该配置找到节点上的 VF 资源并提供给 Pod 使用：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: sriovdp-config
  namespace: kube-system
data:
  config.json: |
    {
        "resourceList": [{
                "resourceName": "sriov_dpu",
                "resourcePrefix": "yusur.tech",
                "selectors": {
                    "vendors": ["1f47"],
                    "devices": ["110f"]
                }}
```


2025 Kube-OVN Team

```
        ]
    }
```

1. □□□□□ SR-IOV Device Plugin

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/sriov-network-device-plugin/v3.6.2/deployments/sriovdp-daemonset.yaml
```

1. □□ SR-IOV □□□□□□□□□□ kubernetes Node □□□□□□

```
kubectl describe node node1 | grep yusur
  yusur.tech/sriov_dpu:  7
  yusur.tech/sriov_dpu:  7
  yusur.tech/sriov_dpu  0           0
```

7.19.3 □□ Multus-CNI

□□ Multus-CNI□□□□ Kube-OVN □□□□ SRIOV □□□ Device ID□

```
kubectl apply -f https://raw.githubusercontent.com/k8snetworkplumbingwg/multus-cni/v4.0.2/deployments/multus-daemonset-thick.yml
```

□□ `NetworkAttachmentDefinition` □

```
apiVersion:
  "k8s.cni.cncf.io/v1"
kind:
  NetworkAttachmentDefinition
metadata:
  name: test
  namespace: kube-system
  annotations:
    k8s.v1.cni.cncf.io/resourceName: yusur.tech/sriov_dpu
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "kube-ovn",
    "plugins":[
        {
            "type":"kube-ovn",
            "server_socket":"/run/openvswitch/kube-ovn-daemon.sock",
            "provider": "test.kube-system.ovn"
        },
        {
            "type":"portmap",
            "capabilities":{
                "portMappings":true
            }
        }
    ]
}'
```

- `provider` : □□□□□ NetworkAttachmentDefinition □ `{name}.{namespace}.ovn` □

7.19.4 Kube-OVN □□□□□□□

1. □□ □□□□

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.12/dist/images/install.sh
```

1. □□□□□□□IFACE □□□□□□□□□□□□□□□□□□□ IP□

```
ENABLE_MIRROR=${ENABLE_MIRROR:-false}
HW_OFFLOAD=${HW_OFFLOAD:-true}
ENABLE_LB=${ENABLE_LB:-false}
IFACE="p0"
```

1. □□ kube-ovn

```
bash install.sh
```

□□□□ **VF □□□ pod**

□□□□□□ yaml □□□□□□ VF □□□□□□□□□ Pod□

2025 Kube-OVN Team

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: default
  annotations:
    v1.multus-cni.io/default-network: kube-system/test
spec:
  containers:
    - name: nginx
      image: docker.io/library/nginx:alpine
      resources:
        requests:
          yusur.tech/sriov_dpu: '1'
        limits:
          yusur.tech/sriov_dpu: '1'
```

- `v1.multus-cni.io/default-network`：⬚⬚⬚⬚⬚⬚ `NetworkAttachmentDefinition` ⬚ `{namespace}/{name}` ⬚

Offload ⬚⬚

⬚⬚⬚⬚ Pod ⬚⬚⬚⬚⬚ `ovs-ovn` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
# ovs-appctl dpctl/dump-flows -m type=offloaded
ufid:67c2e10f-92d4-4574-be70-d072815ff166, skb_priority(0/0),skb_mark(0/0),ct_state(0/0x23),ct_zone(0/0),ct_mark(0/0),ct_label(0/0),recirc_id(0),dp_hash(0/
0),in_port(d85b161b6840_h),packet_type(ns=0/0,id=0/
0),eth(src=0a:c9:1c:70:01:09,dst=8a:18:a4:22:b7:7d),eth_type(0x0800),ipv4(src=10.0.1.10,dst=10.0.1.6,proto=6,tos=0/0x3,ttl=0/
0,frag=no),tcp(src=60774,dst=9001), packets:75021, bytes:109521630, offload_packets:75019, offload_bytes:109521498, used:3.990s,offloaded:yes,dp:tc,
actions:set(tunnel(tun_id=0x5,dst=192.168.201.12,ttl=64,tp_dst=6081,geneve({class=0x102,type=0x80,len=4,0xa0006}),flags(csum|key))),genev_sys_6081
ufid:7940666e-a0bd-42a5-8116-1e84e81bb338, skb_priority(0/0),tunnel(tun_id=0x5,src=192.168.201.12,dst=192.168.201.11,ttl=0/
0,tp_dst=6081,geneve({class=0x102,type=0x80,len=4,0x6000a}),flags(+key)),skb_mark(0/0),ct_state(0/0),ct_zone(0/0),ct_mark(0/0),ct_label(0/
0),recirc_id(0),dp_hash(0/0),in_port(genev_sys_6081),packet_type(ns=0/0,id=0/
0),eth(src=8a:18:a4:22:b7:7d,dst=0a:c9:1c:70:01:09),eth_type(0x0800),ipv4(src=10.0.1.6,dst=10.0.1.10,proto=6,tos=0/0,ttl=0/
0,frag=no),tcp(src=9001,dst=60774), packets:6946, bytes:459664, offload_packets:6944, offload_bytes:459532, used:4.170s,
dp:tc,offloaded:yes,actions:d85b161b6840_h
```

PDF Slack Support

2026⬚4⬚28⬚

2024⬚8⬚13⬚

GitHub



7.19.5 ⬚⬚

7.20 DPDK ⬚⬚

⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚ OVS-DPDK ⬚⬚⬚⬚ KubeVirt ⬚⬚⬚⬚ DPDK ⬚⬚⬚⬚⬚⬚⬚

> ⬚⬚⬚ KubeVirt ⬚⬚⬚⬚⬚⬚ OVS-DPDK⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ patch Vhostuser implementation ⬚⬚ KubeVirt ⬚ KVM Device Plugin ⬚⬚⬚ OVS-DPDK⬚

7.20.1 ⬚⬚⬚⬚

- ⬚⬚⬚⬚⬚⬚⬚⬚ DPDK ⬚⬚⬚⬚⬚⬚⬚⬚
- ⬚⬚⬚⬚⬚ Hugepages⬚

7.20.2 ⬚⬚⬚⬚ DPDK ⬚⬚

⬚⬚⬚⬚⬚⬚ `driverctl` ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ DPDK ⬚⬚⬚⬚⬚⬚

```
driverctl set-override 0000:00:0b.0 uio_pci_generic
```

7.20.3 ⬚⬚⬚⬚

⬚⬚⬚ OVS-DPDK ⬚⬚⬚⬚⬚⬚⬚⬚⬚ Kube-OVN ⬚⬚⬚⬚⬚⬚⬚⬚

```
kubectl label nodes <node> ovn.kubernetes.io/ovs_dp_type="userspace"
```

⬚⬚⬚ OVS-DPDK ⬚⬚⬚ `/opt/ovs-config` ⬚⬚⬚⬚⬚⬚⬚⬚⬚ `ovs-dpdk-config` ⬚

```
ENCAP_IP=192.168.122.193/24
DPDK_DEV=0000:00:0b.0
```

- `ENCAP_IP` : ⬚⬚⬚⬚⬚⬚⬚⬚
- `DPDK_DEV` : ⬚⬚⬚ PCI ID⬚

7.20.4 ⬚⬚ Kube-OVN

⬚⬚⬚⬚⬚⬚⬚⬚

```
wget https://raw.githubusercontent.com/kubeovn/kube-ovn/release-1.17/dist/images/install.sh
```

⬚⬚ DPDK ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚

```
bash install.sh --with-hybrid-dpdk
```

7.20.5 ⬚⬚⬚⬚

⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ vhostuser ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ OVS-DPDK ⬚⬚⬚

⬚⬚ KVM Device Plugin ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ KVM Device Plugin⬚

```
kubectl apply -f https://raw.githubusercontent.com/kubevirt/kubernetes-device-plugins/master/manifests/kvm-ds.yml
```

⬚⬚ NetworkAttachmentDefinition⬚

```yaml
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: ovn-dpdk
  namespace: default
spec:
  config: >-
    {
        "cniVersion": "0.3.0",
        "type": "kube-ovn",
```

```
        "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
        "provider": "ovn-dpdk.default.ovn",
        "vhost_user_socket_volume_name": "vhostuser-sockets",
        "vhost_user_socket_name": "sock"
    }
```

□□□□□ Dockerfile □□ VM □□□

```
FROM quay.io/kubevirt/virt-launcher:v0.46.1

# wget http://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud.qcow2
COPY CentOS-7-x86_64-GenericCloud.qcow2 /var/lib/libvirt/images/CentOS-7-x86_64-GenericCloud.qcow2
```

□□□□□□

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: vm-config
data:
  start.sh: |
    chmod u+w /etc/libvirt/qemu.conf
    echo "hugetlbfs_mount = \"/dev/hugepages\"" >> /etc/libvirt/qemu.conf
    virtlogd &
    libvirtd &

    mkdir /var/lock

    sleep 5

    virsh define /root/vm/vm.xml
    virsh start vm

    tail -f /dev/null
  vm.xml: |
    <domain type='kvm'>
      <name>vm</name>
      <uuid>4a9b3f53-fa2a-47f3-a757-dd87720d9d1d</uuid>
      <memory unit='KiB'>2097152</memory>
      <currentMemory unit='KiB'>2097152</currentMemory>
      <memoryBacking>
        <hugepages>
          <page size='2' unit='M' nodeset='0'/>
        </hugepages>
      </memoryBacking>
      <vcpu placement='static'>2</vcpu>
      <cputune>
        <shares>4096</shares>
        <vcpupin vcpu='0' cpuset='4'/>
        <vcpupin vcpu='1' cpuset='5'/>
        <emulatorpin cpuset='1,3'/>
      </cputune>
      <os>
        <type arch='x86_64' machine='pc'>hvm</type>
        <boot dev='hd'/>
      </os>
      <features>
        <acpi/>
        <apic/>
      </features>
      <cpu mode='host-model'>
        <model fallback='allow'/>
        <topology sockets='1' cores='2' threads='1'/>
        <numa>
          <cell id='0' cpus='0-1' memory='2097152' unit='KiB' memAccess='shared'/>
        </numa>
      </cpu>
      <on_reboot>restart</on_reboot>
      <devices>
        <emulator>/usr/libexec/qemu-kvm</emulator>
        <disk type='file' device='disk'>
          <driver name='qemu' type='qcow2' cache='none'/>
          <source file='/var/lib/libvirt/images/CentOS-7-x86_64-GenericCloud.qcow2'/>
          <target dev='vda' bus='virtio'/>
        </disk>

        <interface type='vhostuser'>
          <mac address='00:00:00:0A:30:89'/>
          <source type='unix' path='/var/run/vm/sock' mode='server'/>
           <model type='virtio'/>
          <driver queues='2'>
            <host mrg_rxbuf='off'/>
          </driver>
        </interface>
        <serial type='pty'>
          <target type='isa-serial' port='0'>
            <model name='isa-serial'/>
          </target>
        </serial>
        <console type='pty'>
          <target type='serial' port='0'/>
```

```
        </console>
        <channel type='unix'>
          <source mode='bind' path='/var/lib/libvirt/qemu/channel/target/domain-1-vm/org.qemu.guest_agent.0'/>
          <target type='virtio' name='org.qemu.guest_agent.0' state='connected'/>
          <alias name='channel0'/>
          <address type='virtio-serial' controller='0' bus='0' port='1'/>
        </channel>

      </devices>
    </domain>
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vm-deployment
  labels:
    app: vm
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vm
  template:
    metadata:
      labels:
        app: vm
      annotations:
        k8s.v1.cni.cncf.io/networks: default/ovn-dpdk
        ovn-dpdk.default.ovn.kubernetes.io/ip_address: 10.16.0.96
        ovn-dpdk.default.ovn.kubernetes.io/mac_address: 00:00:00:0A:30:89
    spec:
      nodeSelector:
        ovn.kubernetes.io/ovs_dp_type: userspace
      securityContext:
        runAsUser: 0
      volumes:
        - name: vhostuser-sockets
          emptyDir: {}
        - name: xml
          configMap:
            name: vm-config
        - name: hugepage
          emptyDir:
            medium: HugePages-2Mi
        - name: libvirt-runtime
          emptyDir: {}
      containers:
        - name: vm
          image: vm-vhostuser:latest
          command: ["bash", "/root/vm/start.sh"]
          securityContext:
            capabilities:
              add:
                - NET_BIND_SERVICE
                - SYS_NICE
                - NET_RAW
                - NET_ADMIN
            privileged: false
            runAsUser: 0
          resources:
            limits:
              cpu: '2'
              devices.kubevirt.io/kvm: '1'
              memory: '8784969729'
              hugepages-2Mi: 2Gi
            requests:
              cpu: 666m
              devices.kubevirt.io/kvm: '1'
              ephemeral-storage: 50M
              memory: '4490002433'
          volumeMounts:
            - name: vhostuser-sockets
              mountPath: /var/run/vm
            - name: xml
              mountPath: /root/vm/
            - mountPath: /dev/hugepages
              name: hugepage
            - name: libvirt-runtime
              mountPath: /var/run/libvirt
```

▯▯▯▯▯▯▯▯▯▯▯▯▯ Pod ▯▯▯▯▯▯▯▯

```
# virsh set-user-password vm root 12345
Password set successfully for root in vm

# virsh console vm
Connected to domain 'vm'
Escape character is ^] (Ctrl + ])

CentOS Linux 7 (Core)
Kernel 3.10.0-1127.el7.x86_64 on an x86_64
```

```
localhost login: root
Password:
Last login: Fri Feb 25 09:52:54 on ttyS0
```

▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
ip link set eth0 mtu 1400
ip addr add 10.16.0.96/16 dev eth0
ip ro add default via 10.16.0.1
ping 114.114.114.114
```

PDF Slack Support

2022▯7▯16▯

2022▯5▯24▯

GitHub 

7.20.6 ▯▯

7.21 OpenStack 集成

本节介绍如何将 OpenStack 虚拟机与 Kubernetes 集群通过 OpenStack Neutron 的 OVN 网络插件与 Kube-OVN 的 OVN 集成，实现 OpenStack 与 Kubernetes 网络互通。

7.21.1 集成原理

本方案使用 OVN-IC 多集群互联功能将 Kubernetes 集群与 OpenStack 的 Kubernetes 集群互联。

集成步骤

1. 规划好 OpenStack 与 Kubernetes 网络的 CIDR 避免冲突
2. 部署一套独立的 OVN-IC 数据库
3. 配置 Kubernetes 集群使用 OpenStack 网络的 VPC。

部署 OVN-IC 数据库

在任意节点运行 `OVN-IC` 数据库

```
docker run --name=ovn-ic-db -d --network=host -v /etc/ovn/:/etc/ovn -v /var/run/ovn:/var/run/ovn -v /var/log/ovn:/var/log/ovn kubeovn/kube-ovn:v1.17.0 bash
start-ic-db.sh
```

Kubernetes 集群

在 `kube-system` Namespace 下创建 `ovn-ic-config` ConfigMap。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ovn-ic-config
  namespace: kube-system
data:
  enable-ic: "true"
  az-name: "az1"
  ic-db-host: "192.168.65.3"
  ic-nb-port: "6645"
  ic-sb-port: "6646"
  gw-nodes: "az1-gw"
  auto-route: "true"
```

- `enable-ic`：启用互联
- `az-name`：本集群的可用区名称
- `ic-db-host`：为 `OVN-IC` 数据库的地址
- `ic-nb-port`：`OVN-IC` 数据库的北向端口 6645。
- `ic-sb-port`：`OVN-IC` 数据库的南向端口 6646。
- `gw-nodes`：本集群的网关节点名称
- `auto-route`：自动发布路由

OpenStack 集群

在 Kubernetes 集群中创建

```
# openstack router create router0
# openstack router list
+--------------------------------------+---------+--------+-------+----------------------------------+
| ID                                   | Name    | Status | State | Project                          |
+--------------------------------------+---------+--------+-------+----------------------------------+
| d5b38655-249a-4192-8046-71aa4d2b4af1 | router0 | ACTIVE | UP    | 98a29ab7388347e7b5ff8bdd181ba4f9 |
+--------------------------------------+---------+--------+-------+----------------------------------+
```


2025 Kube-OVN Team

▒ OpenStack ▒▒ OVN ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

```
ovn-nbctl set NB_Global . name=op-az
```

▒▒▒▒ `OVN-IC` ▒▒▒▒▒▒▒▒▒ `OVN-IC` ▒▒▒▒

```
/usr/share/ovn/scripts/ovn-ctl --ovn-ic-nb-db=tcp:192.168.65.3:6645 \
  --ovn-ic-sb-db=tcp:192.168.65.3:6646 \
  --ovn-northd-nb-db=unix:/run/ovn/ovnnb_db.sock \
  --ovn-northd-sb-db=unix:/run/ovn/ovnsb_db.sock \
  start_ic
```

- `ovn-ic-nb-db` ▒ `ovn-ic-sb-db` : OVN-IC ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
- `ovn-northd-nb-db` ▒ `ovn-northd-sb-db` : ▒▒▒▒ OVN ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

▒▒▒▒▒▒▒▒▒▒

```
ovs-vsctl set open_vswitch . external_ids:ovn-is-interconn=true
```

▒▒▒▒▒▒ OpenStack ▒ OVN ▒▒▒▒▒▒▒▒▒▒▒▒▒

▒▒ `ts` ▒▒▒▒▒▒▒ `router0` ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

```
ovn-nbctl lrp-add router0 lrp-router0-ts 00:02:ef:11:39:4f 169.254.100.73/24
ovn-nbctl lsp-add ts lsp-ts-router0 -- lsp-set-addresses lsp-ts-router0 router \
  -- lsp-set-type lsp-ts-router0 router \
  -- lsp-set-options lsp-ts-router0  router-port=lrp-router0-ts
ovn-nbctl lrp-set-gateway-chassis lrp-router0-ts {gateway chassis} 1000
ovn-nbctl set NB_Global . options:ic-route-adv=true options:ic-route-learn=true
```

▒▒▒▒▒▒ Kubernetes ▒▒▒▒▒▒

```
# ovn-nbctl lr-route-list router0
IPv4 Routes
               10.0.0.22            169.254.100.34 dst-ip (learned)
            10.16.0.0/16            169.254.100.34 dst-ip (learned)
```

▒▒▒▒▒▒ `router0` ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ Kubernetes ▒ Pod ▒▒▒

7.21.2 ▒▒▒▒ OVN

▒▒▒▒▒▒OpenStack ▒ Kubernetes ▒▒▒▒▒▒▒▒ OVN▒▒▒▒▒▒▒▒▒▒ VPC ▒ Subnet ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

▒▒▒▒▒▒▒▒▒▒▒▒ Kube-OVN ▒▒ OVN▒OpenStack ▒▒ Neutron ▒▒▒▒▒▒▒▒▒▒ OVN ▒▒▒▒OpenStack ▒▒▒ networking-ovn ▒▒ Neutron ▒▒▒▒▒▒

Neutron ▒▒▒▒

▒▒ Neutron ▒▒▒▒ `/etc/neutron/plugins/ml2/ml2_conf.ini` ▒

```
[ovn]
...
ovn_nb_connection = tcp:[192.168.137.176]:6641,tcp:[192.168.137.177]:6641,tcp:[192.168.137.178]:6641
ovn_sb_connection = tcp:[192.168.137.176]:6642,tcp:[192.168.137.177]:6642,tcp:[192.168.137.178]:6642
ovn_l3_scheduler = OVN_L3_SCHEDULER
```

- `ovn_nb_connection` ▒ `ovn_sb_connection` : ▒▒▒▒▒▒▒ Kube-OVN ▒▒ `ovn-central` ▒▒▒▒▒▒▒

▒▒▒▒▒▒▒▒ OVS ▒▒▒

```
ovs-vsctl set open . external-ids:ovn-remote=tcp:[192.168.137.176]:6642,tcp:[192.168.137.177]:6642,tcp:[192.168.137.178]:6642
ovs-vsctl set open . external-ids:ovn-encap-type=geneve
ovs-vsctl set open . external-ids:ovn-encap-ip=192.168.137.200
```

- `external-ids:ovn-remote` : ▒▒▒▒▒▒▒ Kube-OVN ▒▒ `ovn-central` ▒▒▒▒▒▒▒
- `ovn-encap-ip` : ▒▒▒▒▒▒▒▒▒ IP ▒▒▒

2025 Kube-OVN Team

▒ **Kubernetes ▒▒▒ OpenStack ▒▒▒**

▒▒▒▒▒▒▒▒ Kubernetes ▒▒▒ OpenStack ▒▒▒▒▒▒▒ OpenStack ▒▒▒▒▒▒ Pod▒

> **Note**
>
> ▒▒▒▒▒▒▒▒ `kube-ovn-controller` ▒ args ▒▒▒▒▒▒▒ `--enable-external-vpc=true` ▒

▒▒ OpenStack ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

```
# openstack router list
+--------------------------------------+---------+--------+-------+----------------------------------+
| ID                                   | Name    | Status | State | Project                          |
+--------------------------------------+---------+--------+-------+----------------------------------+
| 22040ed5-0598-4f77-bffd-e7fd4db47e93 | router0 | ACTIVE | UP    | 62381a21d569404aa236a5dd8712449c |
+--------------------------------------+---------+--------+-------+----------------------------------+
# openstack network list
+--------------------------------------+----------+--------------------------------------+
| ID                                   | Name     | Subnets                              |
+--------------------------------------+----------+--------------------------------------+
| cd59e36a-37db-4c27-b709-d35379a7920f | provider | 01d73d9f-fdaa-426c-9b60-aa34abbfacae |
+--------------------------------------+----------+--------------------------------------+
# openstack subnet list
+--------------------------------------+-------------+--------------------------------------+----------------+
| ID                                   | Name        | Network                              | Subnet         |
+--------------------------------------+-------------+--------------------------------------+----------------+
| 01d73d9f-fdaa-426c-9b60-aa34abbfacae | provider-v4 | cd59e36a-37db-4c27-b709-d35379a7920f | 192.168.1.0/24 |
+--------------------------------------+-------------+--------------------------------------+----------------+
# openstack server list
+--------------------------------------+-------------------+--------+----------------------+--------+--------+
| ID                                   | Name              | Status | Networks             | Image  | Flavor |
+--------------------------------------+-------------------+--------+----------------------+--------+--------+
| 8433d622-a8d6-41a7-8b31-49abfd64f639 | provider-instance | ACTIVE | provider=192.168.1.61 | ubuntu | m1     |
+--------------------------------------+-------------------+--------+----------------------+--------+--------+
```

▒ Kubernetes ▒▒▒▒ VPC ▒▒▒

```
# kubectl get vpc
NAME                                    STANDBY   SUBNETS
neutron-22040ed5-0598-4f77-bffd-e7fd4db47e93   true      ["neutron-cd59e36a-37db-4c27-b709-d35379a7920f"]
ovn-cluster                             true      ["join","ovn-default"]
```

`neutron-22040ed5-0598-4f77-bffd-e7fd4db47e93` ▒▒ OpenStack ▒▒▒▒▒ VPC ▒▒▒

▒▒▒▒▒▒▒ Kube-OVN ▒▒▒ VPC ▒ Subnet ▒▒▒▒ Pod ▒▒▒▒

VPC, Subnet ▒▒ Namespace `net2` ▒▒▒▒ Pod:

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: net2
---
apiVersion: kubeovn.io/v1
kind: Vpc
metadata:
  creationTimestamp: "2021-06-20T13:34:11Z"
  generation: 2
  labels:
    ovn.kubernetes.io/vpc_external: "true"
  name: neutron-22040ed5-0598-4f77-bffd-e7fd4db47e93
  resourceVersion: "583728"
  uid: 18d4c654-f511-4def-a3a0-a6434d237c1e
spec:
  namespaces:
  - net2
---
kind: Subnet
apiVersion: kubeovn.io/v1
metadata:
  name: net2
spec:
  vpc: neutron-22040ed5-0598-4f77-bffd-e7fd4db47e93
  namespaces:
    - net2
  cidrBlock: 12.0.1.0/24
  natOutgoing: false
---
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu
```


2025 Kube-OVN Team

```
  namespace: net2
spec:
  containers:
    - image: docker.io/kubeovn/kube-ovn:v1.8.0
      command:
        - "sleep"
        - "604800"
      imagePullPolicy: IfNotPresent
      name: ubuntu
  restartPolicy: Always
```

PDF Slack Support

2025▯8▯4▯

2022▯5▯24▯

GitHub 

7.21.3 ▯▯

7.22 使用 IPsec 加密节点间通信

Kube-OVN 可以使用 OVN/OVS 内置的 IPsec 能力加密节点间的 Geneve、VXLAN 和 STT 隧道。当前 IPsec 运行时由 `kube-ovn-cni` Pod 中的独立 `ipsec` 容器负责，CNI 容器不再签发证书、启动 IKE/strongSwan、监听 CA Secret 或清理主机 XFRM 状态。

7.22.1 前置条件

- 节点之间允许 UDP 500（IKE）和 UDP 4500（NAT-T）。
- 节点内核启用 `xfrm`、`af_key` 等模块，并提供 OVN/OVS IPsec 所需的 strongSwan 或 libreswan 运行时。
- Kube-OVN 使用的镜像包含匹配当前 OVN/OVS 的 IPsec agent。agent 会在启动时检查镜像中的输出保护能力，未应用对应 OVS monitor 补丁的镜像会拒绝启动。

7.22.2 证书与首次启用顺序

首次启用时必须保持 IPsec 关闭，先让普通 Pod 网络可用，再准备证书签发链路：

1. 使用 `ENABLE_OVN_IPSEC=false`（或对应 Helm 值）安装或升级 Kube-OVN。
2. 部署 cert-manager 及其 CA injector（如果使用外部签发后端），创建专用的 ClusterIssuer，并等待 cert-manager 组件和 ClusterIssuer 为 `Ready`。
3. 确认 controller 和 CNI DaemonSet 已就绪，且普通 Pod 网络可以访问 Kubernetes API。无需为 cert-manager 设置 `hostNetwork`。
4. 再将 controller 和 CNI 的 IPsec 开关设为 `true`，并等待 rollout 完成。

Kube-OVN 自带 signer 时不需要 cert-manager，但仍应先在关闭状态完成普通网络和 controller 的准备。节点始终通过 Kubernetes CSR 请求身份；controller 会校验绑定 Pod、Node UID、chassis 和请求的 CN/SAN 后签发或转交给 cert-manager。CNI ServiceAccount 没有创建 CertificateRequest 的权限，只有 controller 可以把已授权的 CSR 转发给指定的 ClusterIssuer。

7.22.3 启用 IPsec

安装脚本可以使用：

```
ENABLE_OVN_IPSEC=true
```

使用 Helm 时，v1 chart 使用 `func.ENABLE_OVN_IPSEC`，v2 chart 使用 `features.enableOvnIpsec`。如果通过 cert-manager 签发，还需启用 `--cert-manager-ipsec-cert=true`（v1）或 chart 中的 `ipsec.certManager.enabled` 并设置专用 issuer 名称。证书有效期、请求超时、优先级和资源限制位于 chart 的 `ipsec` 配置段。

启用后，CNI Pod 会渲染一个同镜像的 `ipsec` 容器，运行 `/kube-ovn/kube-ovn-ipsec`。该容器拥有 IPsec 密钥目录，并获得运行 IPsec 所需的 `NET_ADMIN`、`NET_BIND_SERVICE` 和 `SYS_NICE`；CNI 容器显式丢弃 `SYS_NICE`，不因启用 IPsec 而获得额外调度或密钥权限。Pod 的 host network、host PID 和 ServiceAccount 仍是共享边界，`ipsec` 容器不是独立的 Kubernetes 身份。

首次启用由 controller 协调：它冻结 DaemonSet UID、Pod 模板、信任 bundle 和目标 Node UID，先等待所有节点完成身份准备并确认本地 agent 就绪，再依次安装保护、启动运行时，最后切换 OVN 的全局 IPsec 开关。节点不能仅凭 Pod Ready 或未签名的 Node annotation 通过这道屏障。CNI DaemonSet 使用 `maxSurge: 0`、`maxUnavailable: 1`，以便一次只替换一个节点并保持已有保护。

7.22.4 运行时与续期

IPsec agent 持久化待处理私钥、CSR 绑定和当前身份 generation。完整的密钥、证书和信任 generation 会在一次 OVSDB 更新中激活，其他 `other_config` 条目保持不变。证书续期时间在叶子证书生命周期的 40% 到 60% 之间抖动，避免所有节点同时续期；已有有效身份时，失败的续期不会覆盖它。agent 监督 IKE 和 monitor 前台进程，运行时故障采用 1 到 30 秒退避。

状态和就绪检查会公开 phase、reason、configurationApplied、runtimeHealthy、protectionArmed、expires，以及 Node UID、chassis、generation 和证书/信任摘要；不包含 PEM、私钥或 SA key。`readyz` 需要当前证书未过期、进程健康、内核保护已读回，以及当前信任摘要已由本地 monitor 确认；这表示配置刷新完成，不表示每一条旧 SA 都已重新建立。IPsec 就绪失败会让 Pod 整体 NotReady，但不会重启 CNI 容器。

验证节点状态：

```
kubectl get pods -n kube-system -l app.kubernetes.io/name=kube-ovn-cni
kubectl get csr
kubectl get clusterissuer
kubectl exec -n kube-system <kube-ovn-cni-pod> -c ipsec -- \
  ovs-appctl ipsec/show
```


2025 Kube-OVN Team

7.22.5 ████████

████████ Pod ██ `ipsec` ███████ controller ▯ CNI ▯ IPsec ████ `false` ▯controller ████ `Disabling` challenge██████ NB/SB ██████████████████ chart ██████ `ipsec` ████████ `ipsec-cleanup` init container▯

- ████ IPsec ████████████████ IKE/monitor███ CA█████ ███████
- ████████ root ████ `NET_ADMIN` ▯ `NET_BIND_SERVICE` ████ `SYS_NICE` ███ `restartPolicy` ▯probe ████████
- ███████ tunnel██████ SA ███ drain██████ Node UID▯ Pod UID▯CSR████ controller challenge██████████ owner ████ drain ██████████████████
- ██████████lease ████████ live ████████████████ output lease ███ guard██████ released receipt ██████████ owner ▯ SA▯policy ████

controller ████████████ receipt ████ `Disabled` ███ `maxUnavailable: 1` ███ rollout███████ init ████████ Pod▯ ███ Release epoch ████████████ init ████ Release barrier▯ controller ████ DaemonSet███████ Kubernetes ████████ ████████████████████████

7.22.6 ██████████

- `kubectl get csr` ██ IPsec CSR ████ `kube-ovn-cni` Pod ▯ Node UID▯ ████████████████
- ██ cert-manager ▯▯ClusterIssuer ████ Kube-OVN IPsec▯controller ▯ ▯▯ issuer▯namespace████ ServiceAccount▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ███ cert-manager ▯▯ approver ████████
- controller ▯ Secret ▯ CertificateRequest ████████ namespace▯CSR ████████████ signer ████████CNI ██████ CertificateRequest ████
- ▯▯ `ovs-appctl ipsec/show` ▯ agent ▯ `readyz` ████████████ monitor readback███ guard ▯▯ UDP 500/4500 ███████ OVN/OVS ███
- ██████ XFRM SA▯IPsec ████ `ovn-ipsec-coordination` ConfigMap▯ ████ controller ▯ agent ████████████

PDF Slack Support

2026▯10▯29▯

2023▯4▯18▯

GitHub 

7.22.7 ▯▯

7.23 OVN □□□□

□□□□□□□□ Pod□□□□□□□□□□□□ GRE/ERSPAN □□□□□□□□□□□□

> □□□□□ Kube-OVN □□□□□ v1.12□

7.23.1 □□ Multus-CNI

□□ Multus-CNI □□ □□ Multus□

7.23.2 □□□□□□□

□□□□□□□□□□□□□□□

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: attachnet
  namespace: default
spec:
  config: |
    {
      "cniVersion": "0.3.1",
      "type": "kube-ovn",
      "server_socket": "/run/openvswitch/kube-ovn-daemon.sock",
      "provider": "attachnet.default.ovn"
    }
```

□□ `provider` □□□□□ `<NAME>.<NAMESPACE>.ovn` □

7.23.3 □□ Underlay □□

□□□□□□□□□□□□□□□□□□□□□□□□□□□MTU □□□□□□□□ LSP/Pod□□□□□□□ Underlay □□□□□□□

□□□□□□□□ Underlay □□□

```
apiVersion: kubeovn.io/v1
kind: ProviderNetwork
metadata:
  name: net1
spec:
  defaultInterface: eth1
---
apiVersion: kubeovn.io/v1
kind: Vlan
metadata:
  name: vlan1
spec:
  id: 0
  provider: net1
---
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: subnet1
spec:
  protocol: IPv4
  cidrBlock: 172.19.0.0/16
  excludeIps:
  - 172.19.0.2..172.19.0.20
  gateway: 172.19.0.1
  vlan: vlan1
  provider: attachnet.default.ovn
```

□□□□□□ `provider` □□□□□□□□ `provider` □□□

7.23.4 □□□□□□ Pod

□□□□□□□□□□□□□□□□□□ Pod□

```
apiVersion: v1
kind: Pod
```


2025 Kube-OVN Team

```
metadata:
  name: pod1
  annotations:
    k8s.v1.cni.cncf.io/networks: default/attachnet
spec:
  containers:
  - name: bash
    image: docker.io/kubeovn/kube-ovn:v1.17.0
    args:
    - bash
    - -c
    - sleep infinity
    securityContext:
      privileged: true
```

▒▒▒▒▒▒▒▒▒ Pod ▒ IP ▒▒▒

```
$ kubectl get ips | grep pod1
pod1.default                        10.16.0.12   00:00:00:FF:34:24  kube-ovn-worker  ovn-default
pod1.default.attachnet.default.ovn  172.19.0.21  00:00:00:A0:30:68  kube-ovn-worker  subnet1
```

▒▒▒▒▒▒▒ IP ▒▒ `172.19.0.21` ▒

7.23.5 ▒▒ OVN ▒▒▒▒

▒▒▒▒▒▒▒▒▒ OVN ▒▒▒▒▒▒

```
kubectl ko nbctl mirror-add mirror1 gre 99 from-lport 172.19.0.21
kubectl ko nbctl lsp-attach-mirror coredns-787d4945fb-gpnkb.kube-system mirror1
```

▒▒ `coredns-787d4945fb-gpnkb.kube-system` ▒ OVN LSP ▒▒▒▒▒▒▒▒▒▒ `<POD_NAME>.<POD_NAMESPACE>` ▒

▒▒▒ OVN ▒▒▒▒▒▒▒▒▒▒

```
ovn-nbctl mirror-add <NAME> <TYPE> <INDEX> <FILTER> <IP>

NAME   - add a mirror with given name
TYPE   - specify TYPE 'gre' or 'erspan'
INDEX  - specify the tunnel INDEX value
         (indicates key if GRE, erpsan_idx if ERSPAN)
FILTER - specify FILTER for mirroring selection
         ('to-lport' / 'from-lport')
IP     - specify Sink / Destination i.e. Remote IP

ovn-nbctl mirror-del [NAME]         remove mirrors
ovn-nbctl mirror-list               print mirrors

ovn-nbctl lsp-attach-mirror PORT MIRROR   attach source PORT to MIRROR
ovn-nbctl lsp-detach-mirror PORT MIRROR   detach source PORT from MIRROR
```

7.23.6 ▒▒▒▒▒▒ Pod

▒▒▒▒▒▒ Pod ▒▒▒▒▒▒▒▒▒

```
root@pod1:/kube-ovn# ip link add mirror1 type gretap local 172.19.0.21 key 99 dev net1
root@pod1:/kube-ovn# ip link set mirror1 up
```

▒▒▒▒▒▒▒▒▒▒▒▒▒ Pod ▒▒▒▒▒▒▒▒▒

```
root@pod1:/kube-ovn# tcpdump -i mirror1 -nnve
tcpdump: listening on mirror1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
05:13:30.328808 00:00:00:a3:f5:e2 > 00:00:00:97:0f:6e, ethertype ARP (0x0806), length 42: Ethernet (len 6), IPv4 (len 4), Request who-has 10.16.0.7 tell
10.16.0.4, length 28
05:13:30.559167 00:00:00:a3:f5:e2 > 00:00:00:89:d5:cc, ethertype IPv4 (0x0800), length 212: (tos 0x0, ttl 64, id 57364, offset 0, flags [DF], proto UDP (17),
length 198)
    10.16.0.4.53 > 10.16.0.6.50472: 34511 NXDomain*- 0/1/1 (170)
05:13:30.559343 00:00:00:a3:f5:e2 > 00:00:00:89:d5:cc, ethertype IPv4 (0x0800), length 212: (tos 0x0, ttl 64, id 57365, offset 0, flags [DF], proto UDP (17),
length 198)
    10.16.0.4.53 > 10.16.0.6.45177: 1659 NXDomain*- 0/1/1 (170)
05:13:30.560625 00:00:00:a3:f5:e2 > 00:00:00:89:d5:cc, ethertype IPv4 (0x0800), length 200: (tos 0x0, ttl 64, id 57367, offset 0, flags [DF], proto UDP (17),
length 186)
    10.16.0.4.53 > 10.16.0.6.43848: 2636*- 0/1/1 (158)
05:13:30.562774 00:00:00:a3:f5:e2 > 00:00:00:89:d5:cc, ethertype IPv4 (0x0800), length 191: (tos 0x0, ttl 64, id 57368, offset 0, flags [DF], proto UDP (17),
length 177)
    10.16.0.4.53 > 10.16.0.6.37755: 48737 NXDomain*- 0/1/1 (149)
05:13:30.563523 00:00:00:a3:f5:e2 > 00:00:00:89:d5:cc, ethertype IPv4 (0x0800), length 187: (tos 0x0, ttl 64, id 57369, offset 0, flags [DF], proto UDP (17),
length 173)
    10.16.0.4.53 > 10.16.0.6.53887: 45519 NXDomain*- 0/1/1 (145)
05:13:30.564940 00:00:00:a3:f5:e2 > 00:00:00:89:d5:cc, ethertype IPv4 (0x0800), length 201: (tos 0x0, ttl 64, id 57370, offset 0, flags [DF], proto UDP (17),
```

```
length 187)
    10.16.0.4.53 > 10.16.0.6.40846: 25745 NXDomain*- 0/1/1 (159)
05:13:30.565140 00:00:00:a3:f5:e2 > 00:00:00:89:d5:cc, ethertype IPv4 (0x0800), length 201: (tos 0x0, ttl 64, id 57371, offset 0, flags [DF], proto UDP (17),
length 187)
    10.16.0.4.53 > 10.16.0.6.45214: 61875 NXDomain*- 0/1/1 (159)
05:13:30.566023 00:00:00:a3:f5:e2 > 00:00:00:55:e4:4e, ethertype IPv4 (0x0800), length 80: (tos 0x0, ttl 64, id 45937, offset 0, flags [DF], proto UDP (17),
length 66)
    10.16.0.4.44116 > 172.18.0.1.53: 16025+ [1au] AAAA? kube-ovn.io. (38)
```

7.23.7 ▯▯▯▯

1. ▯▯▯▯ ERSPAN ▯▯▯▯▯▯▯OVN ▯▯▯▯▯▯▯▯ Linux ▯▯▯▯▯▯▯▯ 4.14▯▯▯▯ ERSPAN ▯▯▯▯▯▯▯▯▯ IPv6 ▯▯▯▯▯▯▯Linux ▯▯▯▯▯▯▯▯ 4.16▯
2. ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ OVN ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

PDF Slack Support

2025▯7▯2▯

2023▯4▯20▯

GitHub 

7.23.8 ▯▯

2025 Kube-OVN Team

7.24 ▯▯ OVN Central

托管控制平面（HCP）模式会为工作负载集群运行独立的、由 PVC 持久化的 `ovn-central` StatefulSet。这样 OVN northbound 和 southbound 数据库可以位于工作负载集群之外，并通过 HCP 配置暴露访问地址。

两种 Chart 布局都支持 HCP 模式：

- classic chart：`ovn-central.hcp.*`
- v2 chart：`central.hcp.*`

7.24.1 ▯▯

安装 Chart 前先创建 HCP namespace。Helm release 可以继续安装在 Kube-OVN 的常规 namespace 中，HCP 资源会渲染到 `hcp.namespace` 指定的 namespace。

```
kubectl create namespace hcp
```

下面是 classic chart 示例。启用 HCP 时必须设置 `nbAddress` 和 `sbAddress`。每个地址都要包含连接协议和端口，并且必须能被工作负载集群访问。

```
namespace: kube-system

ovn-central:
  hcp:
    enabled: true
    namespace: hcp
    replicas: 3
    nbAddress: tcp:ovn-nb.example.com:6641
    sbAddress: tcp:ovn-sb.example.com:6642
    service:
      type: NodePort
      nbNodePort: 30641
      sbNodePort: 30642
    storage:
      size: 5Gi
      storageClassName: ""
```

v2 chart 在 `central.hcp` 下使用相同的配置：

```
central:
  hcp:
    enabled: true
    namespace: hcp
    replicas: 3
    nbAddress: tcp:ovn-nb.example.com:6641
    sbAddress: tcp:ovn-sb.example.com:6642
    service:
      type: NodePort
      nbNodePort: 30641
      sbNodePort: 30642
    storage:
      size: 5Gi
      storageClassName: ""
```

namespace 创建后，使用 values 文件安装 Chart：

```
helm upgrade --install kube-ovn ./charts/kube-ovn -n kube-system -f hcp-values.yaml
```

安装 v2 chart 时，将 `./charts/kube-ovn` 替换为 v2 chart 路径。

7.24.2 ▯▯▯▯▯

启用 HCP 后，Chart 会在 `hcp.namespace` 中创建以下资源：

- 一个 `ovn-central` StatefulSet，副本数由 `hcp.replicas` 设置，默认值为 3。
- 每个 StatefulSet Pod 一个由 PVC 提供的 `ovn-data` 卷。默认请求容量为 5Gi，访问模式为 `ReadWriteOnce`；集群没有合适的默认 StorageClass 时设置 `storageClassName`。
- 一个无头 `ovn-central` Service，为内部 Raft 流量提供 6643 和 6644 端口。
- `ovn-nb` 和 `ovn-sb` Service，为客户端流量提供 6641 和 6642 端口。

2025 Kube-OVN Team

□□ HCP Pod □□□□□□ `kubernetes.io/hostname` □□□ Pod □□□□□□ □□□□□□□□ `ovn-central` Pod □□□□□□□□Pod □□□□ `NoSchedule` □ `NoExecute` □ `CriticalAddonsOnly` □□□□□ Chart □□□ Linux □□□□classic chart □□□ `ovn-central.nodeSelector` □v2 chart □□□□□□□ `central.nodeAffinity` □□□HCP □□□□ □□□□□ master □□□□□□□□□ □□□□□□□ StatefulSet □□□□□□□

7.24.3 Service □□

HCP □ `service.type` □□□ `NodePort` □NB □□ NodePort □ 30641□SB □□ NodePort □ 30642□□□□□□□□□□□□□□□□□ `service.type` □□□ `LoadBalancer` □□□□□ `NodePort` □ `LoadBalancer` □□ `nbNodePort` □ `sbNodePort` □□□ NB/SB Service□

□□ NB □ SB Service □□□□□□□□□□□□□ `ovn-central` Service □□ StatefulSet □□ Raft □□□ `ovn-northd` □□□□□□□□□ ClusterIP Service□ □□□□□□□□□□□□□ `nbAddress` □ `sbAddress` □□□□□□□□□□□□□ □ NB/SB Service□

7.24.4 □□□□

HCP □□□□□ `ovn-central` □□□□□□□□□□□□□□□□□□ agent □ CRD □□□□□□□□□□□□□□□□□ data-plane □□□□ Kube-OVN □□□ □□ OVN NB/SB □□□□ HCP □ `nbAddress` □ `sbAddress` □

HCP namespace□□□□□□□□□□□□□□□□□ NB/SB □□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□ PVC□□□□□□□□□□□□□□□ Service □□□

PDF | Slack | Support

2026□11□7□

2026□11□7□

GitHub



7.24.5 □□

2025 Kube-OVN Team

7.25 □□□□ DNS □□□ Kube-OVN □□

NodeLocal DNSCache □□□□□□□□□□ DaemonSet □□ DNS □□□□□□□□ DNS □□□□□□□□□□□ Kube-OVN □□□

7.25.1 □□□□ DNS □□□□

□□ Kubernetes □□□□□ DNS □□

□□□□□ Kubernetes □□□□ Nodelocaldnscache□

□□□□□□□□□□

```
#!bin/bash

localdns=169.254.20.10
domain=cluster.local
kubedns=10.96.0.10

wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/cluster/addons/dns/nodelocaldns/nodelocaldns.yaml
sed -i "s/__PILLAR__LOCAL__DNS__/$localdns/g; s/__PILLAR__DNS__DOMAIN__/$domain/g; s/,__PILLAR__DNS__SERVER__//g; s/__PILLAR__CLUSTER__DNS__/$kubedns/g"
nodelocaldns.yaml

kubectl apply -f nodelocaldns.yaml
```

□□□□□□□□ kubelet □□□□□□□ `/var/lib/kubelet/config.yaml` □□ clusterDNS □□□□□□□□ DNS IP 169.254.20.10□□□□□ kubelet □□□

Kube-OVN □□ DNS □□

□□□ Kubernetes □ Nodelocal DNScache □□□□Kube-OVN □□□□□□□□□

UNDERLAY SUBNET □□ U2O □□

□□□ Underlay Subnet □□□□□□□ DNS □□□□□□□□ U2O □□□□□ `kubectl edit subnet {your subnet}` □□□□ `spec.u2oInterconnection = true` , □□□ Overlay Subnet □□□□□□□□□□

□ KUBE-OVN-CONTROLLER □□□□□□□□ DNS IP

```
kubectl edit deployment kube-ovn-controller -n kube-system
```

□ `spec.template.spec.containers.args` □□□□□ `--node-local-dns-ip=169.254.20.10`

□□□□□□□□ POD

□□□□□□ Pod □□□□□ `/etc/resolv.conf` □ nameserver □□□□□ DNS IP□□□□□□□□ Pod □ nameserver □□□□□□□□□ DNS ClusterIP□□□□ u2o □□□□□□□□□□□□□ Pod □□□□□□ Pod □ □□

7.25.2 □□□□□□□ DNS □□□□

□□□□□□□□□□□□ Pod □□□□□□□□□□ Pod □ DNS □□□□□□□□□ 169.254.20.10□□□□□□□□□□

```
# kubectl exec -it pod1 -- nslookup github.com
Server:         169.254.20.10
Address:        169.254.20.10:53


Name:   github.com
Address: 20.205.243.166
```

□□□□□□□□□□□□□□□□□□□ DNS □□□□□□□ ovn0 □□□□□□□□ DNS □□□DNS □□□□□□□□□□

```
# tcpdump -i any port 53

06:20:00.441889 659246098c56_h P   ifindex 17 00:00:00:73:f1:06 ethertype IPv4 (0x0800), length 75: 10.16.0.2.40230 > 169.254.20.10.53: 1291+ A? baidu.com.
(27)
06:20:00.441889 ovn0  In  ifindex 7 00:00:00:50:32:cd ethertype IPv4 (0x0800), length 75: 10.16.0.2.40230 > 169.254.20.10.53: 1291+ A? baidu.com. (27)
06:20:00.441950 659246098c56_h P   ifindex 17 00:00:00:73:f1:06 ethertype IPv4 (0x0800), length 75: 10.16.0.2.40230 > 169.254.20.10.53: 1611+ AAAA?
baidu.com. (27)
06:20:00.441950 ovn0  In  ifindex 7 00:00:00:50:32:cd ethertype IPv4 (0x0800), length 75: 10.16.0.2.40230 > 169.254.20.10.53: 1611+ AAAA? baidu.com. (27)
06:20:00.442203 ovn0  Out ifindex 7 00:00:00:52:99:d8 ethertype IPv4 (0x0800), length 145: 169.254.20.10.53 > 10.16.0.2.40230: 1611* 0/1/0 (97)
```

```
06:20:00.442219 659246098c56_h Out ifindex 17 00:00:00:ea:b3:5e ethertype IPv4 (0x0800), length 145: 169.254.20.10.53 > 10.16.0.2.40230: 1611* 0/1/0 (97)
06:20:00.442273 ovn0  Out ifindex 7 00:00:00:52:99:d8 ethertype IPv4 (0x0800), length 125: 169.254.20.10.53 > 10.16.0.2.40230: 1291* 2/0/0 A 39.156.66.10, A
110.242.68.66 (77)
06:20:00.442278 659246098c56_h Out ifindex 17 00:00:00:ea:b3:5e ethertype IPv4 (0x0800), length 125: 169.254.20.10.53 > 10.16.0.2.40230: 1291* 2/0/0 A
39.156.66.10, A 110.242.68.66 (77)
```

7.25.3 ████

⚠️ ███

████████ NetworkPolicy██████ NetworkPolicy ███████ DNS IP██ 169.254.20.10████ CIDR████ NetworkPolicy ██ DNS ██████████ Pod ██████████

NetworkPolicy ████

███████ Pod ████ DNS ████████ NetworkPolicy ████

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-local-dns-and-node-cidr
  namespace: default  #
spec:
  podSelector: {}  #       Pod
  policyTypes:
  - Ingress
  - Egress
  egress:
  #       DNS
  - to:
    - ipBlock:
        cidr: 169.254.20.10/32
  #       CIDR          CIDR
  - to:
    - ipBlock:
        cidr: 10.0.0.0/8  #      CIDR
  ingress:
  #       DNS
  - from:
    - ipBlock:
        cidr: 169.254.20.10/32
  #       CIDR              CIDR
  - from:
    - ipBlock:
        cidr: 10.0.0.0/8  #      CIDR
```

████

- `169.254.20.10/32` ███ DNS ███ IP ██
- `10.0.0.0/8` ████ CIDR████████████████

PDF Slack Support

2025年7月3日

2023年5月5日



GitHub 



7.25.4 ██

2025 Kube-OVN Team



7.26 [illegible] VPC NAT [illegible]

7.26.1 [illegible]

默认 VPC 下的 Overlay 子网，打开 `natOutgoing` 开关时，Subnet 下的所有 Pod 访问外网都需要做 SNAT 成节点的 IP，但是有些场景我们并不希望子网内所有 Pod 访问外网都做 SNAT。

因此 NAT 策略就是为了提供一个接口让用户决定子网内的哪些 CIDR 或者 IP 访问外网做 SNAT。

7.26.2 [illegible]

在 `subnet.Spec` 中开启 `natOutgoing` 开关，并且添加字段 `natOutgoingPolicyRules` 如下：

```
spec:
  natOutgoing: true
  natOutgoingPolicyRules:
    - action: forward
      match:
        srcIPs: 10.0.11.0/30,10.0.11.254
    - action: nat
      match:
        srcIPs: 10.0.11.128/26
        dstIPs: 114.114.114.114,8.8.8.8
```

以上案例表示有两条 NAT 策略规则：

1. 源 IP 是 10.0.11.0/30 或者 10.0.11.254 的报文访问外网时不会做 SNAT。
2. 源 IP 是 10.0.11.128/26 并且目的 IP 是 114.114.114.114 或者 8.8.8.8 的报文访问外网时会做 SNAT。

字段描述：

`action`：满足 `match` 对应条件的报文，会执行的 action，action 分为两种 `forward` 和 `nat`，`forward` 表示报文出外网不做 SNAT，`nat` 表示报文出外网做 SNAT。没有配置 natOutgoingPolicyRules 时，默认情况报文仍然是做 SNAT。

`match`：表示报文的匹配段，匹配段有 `srcIPs` 和 `dstIPs`，这里表示从子网内到外网方向上的报文的源 IP 和目的 IP。`match.srcIPs` 和 `match.dstIPs` 支持多个 CIDR 和 IP，之间用逗号间隔。

如果出现多个 match 规则重叠，则按照 `natOutgoingPolicyRules` 数组顺序进行匹配，最先被匹配的 action 会被执行。

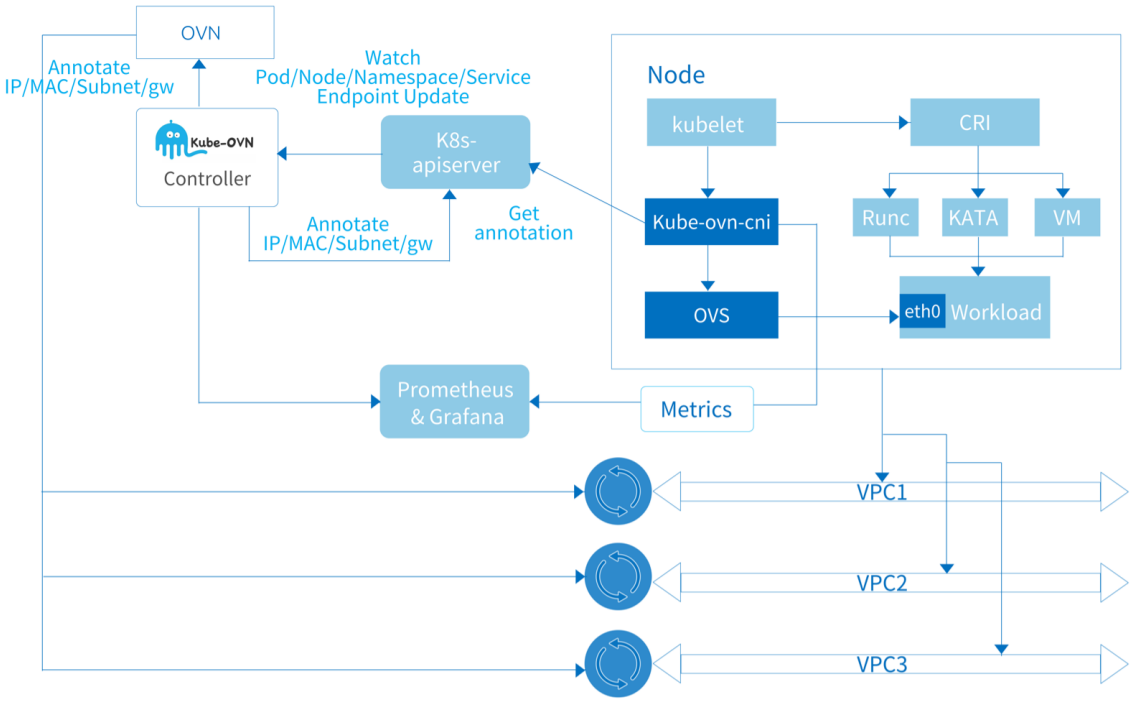
PDF Slack Support

2023年10月25日

2023年6月5日

GitHub

7.26.3 [illegible]



8. 架构设计

8.1 架构设计

本文档将介绍 Kube-OVN 的总体架构，和各个组件的功能以及其之间的交互。

总体来看，Kube-OVN 作为 Kubernetes 和 OVN 之间的一个桥梁，将成熟的 SDN 和云原生相结合。这意味着 Kube-OVN 不仅通过 OVN 实现了 Kubernetes 下的网络规范，例如 CNI，Service 和 Networkpolicy，还将大量的 SDN 领域能力带入云原生，例如逻辑交换机，逻辑路由器，VPC，网关，QoS，ACL 和流量镜像。

同时 Kube-OVN 还保持了良好的开放性可以和诸多技术方案集成，例如 Cilium，Submariner，Prometheus，KubeVirt 等等。

8.1.1 架构设计

Kube-OVN 的组件可以大致分为三类：

- 上游 OVN/OVS 组件。
- 核心控制器和 Agent。
- 监控，运维工具和扩展组件。

上游 **OVN/OVS** 组件

该类型组件来自 OVN/OVS 社区，并针对 Kube-OVN 的使用场景做了特定修改。OVN/OVS 本身是一套成熟的管理虚机和容器的 SDN 系统，我们强烈建议对 Kube-OVN 实现感兴趣的用户先去读一下 ovn-architecture(7) 来了解什么是 OVN 以及如何和它进行集成。Kube-OVN 使用 OVN 的北向接口创建和调整虚拟网络，并将其中的网络概念映射到 Kubernetes 之内。

所有 OVN/OVS 相关组件都已打包成对应镜像，并可在 Kubernetes 中运行。

OVN-CENTRAL

`ovn-central` Deployment 运行 OVN 的管理平面组件，包括 `ovn-nb`，`ovn-sb`，和 `ovn-northd`。

- `ovn-nb`：保存虚拟网络配置，并提供 API 进行虚拟网络管理。`kube-ovn-controller` 将会主要和 `ovn-nb` 交互配置虚拟网络。

2025 Kube-OVN Team

- `ovn-sb` ████ `ovn-nb` ████████████████████████████████
- `ovn-northd` ██ `ovn-nb` ████████ `ovn-sb` ███████

██ `ovn-central` ████ Raft ████████████

OVS-OVN

`ovs-ovn` ▯ DaemonSet ██████████ Pod ████ `openvswitch`, `ovsdb`, ▯ `ovn-controller` ███████ `ovn-central` ▯ Agent ██████████████

██████ Agent

████ Kube-OVN ████████ OVN ▯ Kubernetes ████████████████████████████ ████████████████

KUBE-OVN-CONTROLLER

██████ Deployment ████ Kubernetes ████ OVN ██████████████ Kube-OVN ████████ `kube-ovn-controller` ████████████████████████████ OVN ████ ████████████ Pod▯Service▯Endpoint▯Node▯NetworkPolicy▯VPC▯Subnet▯Vlan▯ProviderNetwork▯

▯ Pod ████ `kube-ovn-controller` ███ Pod ████████████ IPAM ██████████ `ovn-central` ██ ████████████ ACL ██████ `kube-ovn-controller` ██████ ████████ CIDR████████████ Pod ▯ annotation ███ annotation ████ `kube-ovn-cni` ████████████

KUBE-OVN-CNI

██████ DaemonSet ████████████ CNI ██████████ OVS ███████

▯ DaemonSet ███ `kube-ovn` ████████████ `kubelet` ▯ `kube-ovn-cni` ████████████ CNI ██ ███ `kube-ovn-cni` ████████████████ `/opt/cni/bin` ████

`kube-ovn-cni` ████████████████████████████

1. ██ `ovn-controller` ▯ `vswitchd` ▯
2. ██ CNI add/del ███
   a. ████ veth ██ OVS ████
   b. ██ OVS ████
   c. ██████ iptables/ipset/route ████
3. ██████ QoS.
4. ████ `ovn0` ████████████████
5. ██████████ Vlan/Underlay/EIP ████
6. ████████████

██████████████

████████████████████████████ Kube-OVN ████████████████████████

KUBE-OVN-SPEAKER

██████ DaemonSet ██ Pod IP ████

████████████ BGP ███

KUBE-OVN-PINGER

██████ DaemonSet ██████████ OVS ████████████████████████████████ Kube-OVN ████▯

KUBE-OVN-MONITOR

██████ Deployment ██ OVN ██████████████████ Kube-OVN ████▯

KUBECTL-KO

████ kubectl ████████████████████████ kubectl ████▯

2025 Kube-OVN Team

PDF Slack Support

2025⬚9⬚10⬚

2022⬚5⬚24⬚

GitHub 

8.1.2 ⬚⬚

8.2 What's Next

This document lists the features merged into the master branch for the next minor release.

8.2.1 Post-v1.16.0

- Add routed subnet mode with `/32` IPv4 or `/128` IPv6 Pod addresses and gateway hairpinning through the OVN logical router. #7294
- Add experimental NetworkPolicy ACL sampling with node-local decoding and `kubectl ko acl` troubleshooting commands. #7172
- QoS: support configurable ingress/egress burst limits via per-NIC annotations (`ovn.kubernetes.io/ingress_burst` and `ovn.kubernetes.io/egress_burst`). #6628
- VPC Egress Gateway: accept integer Mbps values and `M`, `Mi`, `G`, or `Gi` bit-rate quantities for per-replica bandwidth limits. #7115
- VPC NAT Gateway: allow placing the NAT gateway StatefulSet and its iptables-EIP custom resources in a user-specified namespace. #6595
- iptables-EIP: surface the bound external subnet in the `kubectl get iptables-eip` printer. #6672
- Helm Chart (kube-ovn-v2): ship PrometheusRule and Grafana dashboards out of the box. #6630

8.2.2 Post-v1.15.0

- NetworkPolicy now supports provider-scoped policies for multi-network pods using the `ovn.kubernetes.io/policy-for` annotation. #6223
- Support static IP/MAC for multiple interfaces on the same logical switch. #6060
- MetalLB underlay integration now supports IPv6 and dual-stack. #6159
- KubeVirt live-migration multi-chassis options now apply to all VM NICs, not just the primary one. #6241
- Add human-readable descriptions to all Kube-OVN CRD fields for better `kubectl explain` output. #6133 #6147
- Support BGP and EVPN (L3VPN) for VPC Egress Gateway, running FRR in the egress gateway Pod. #6224
- Support per-pod NIC granular DHCP control via annotations, overriding subnet-level DHCP settings. #6475
- Extend SecurityGroup API with tiers, larger priority range (1-16384), localAddress and port matches. #6330
- Use BigInt for SubnetStatus IP count fields to support large address ranges. #6403
- Improve hairpin logic to support FIP, SNAT from any CIDR within the VPC and LoadBalancers. #6445
- VPC NAT Gateway
- Support user-defined annotations on NAT gateway Pod template. #6256
- Allow SNAT EIP to FIP EIP traffic. #6329
- Allow any EIP to share external subnet gateway in one native VLAN. #6426
- VPC Egress Gateway
- Support custom resources and bandwidth limits. #6407
- Interconnection
- Add vendor ID to transit switches to avoid conflicts with other OVN controllers. #6186
- Reliability
- OpenFlow synchronization: detect and recover from stale or missing OVS flows automatically. #6117
- OVN DB: back up Raft header before rejoining cluster to improve recovery. #6106
- Performance
- Strip `managedFields` from informer cache to reduce memory usage. #6119
- Add field selectors to informer factory to reduce API server load. #6091
- Security
- Replace wildcard RBAC verbs with explicit verb lists. #6233
- Specify ephemeral storage limits for containers. #6259

2025 Kube-OVN Team

- Helm Chart
- Make DaemonSet update strategy configurable via `values.yaml`. #6136
- Introduce `extraEnv` variable for all components. #6142
- Add `affinity` and `nodeSelector` support for ovs-ovn and ovs-ovn-dpdk DaemonSets. #6308
- Add `external-gateway-config-ns` option for controller. #6211
- Introduce ServiceMonitor and labels to services. #6340

8.2.3 Post-v1.14.0

- ACL log supports ratelimiting. #5938
- Subnet with centralized gateway now supports nodeSelectors. #5956
- Overlay encapsulation NIC selection. #5946
- Performace: skip conntrack for specific dst CIDRs. #5821
- NetworkPolicy supports `lax` mode which only deny traffic type of TCP, UDP and SCTP. That means ARP, ICMP and DHCP traffic are alaways allowed. #5745
- Remove internal-port type interface code. #5794
- IPPool
- Multiple IPPools now can bind to the same Namespace. #5731
- Pods in a bound namespace will only get IPs from the bound pool(s), not other ranges in the subnet. #5731
- IPPool will create an AddressSet that can be work with VPC Policy Route and ACL. #5920
- `AdminNetworkPolicy` now supports specify egress peers using FQDNs. #5703
- Using ARP for IPv4 network ready check: now you don't need ACL allow rules for gateway to make Pod running. #5716
- Non-primary CNI mode: you can run Kube-OVN as the secondary only network, without annoying unused annotations and logical switch port allocations. #5618
- VPC NAT Gateway:
- No default EIP mode: the secondary interface can initialize without a default EIP to avoid the waste. #5605
- Custom routes: you can control the route rules within the vpc-nat-gateway Pods to control traffic paths. #5608
- Gratuitous ARP: VPC NAT Gateway automatically sends gratuitous ARP packets during initialization to accelerate network convergence. #5607
- Healthchecks for static endpoints in `SwitchLBRules`: SLR with both selector or endpoints key can support healthchecks. #5435
- Underlay
- Node Selectors for `ProviderNetwork`: instead of adding/removing nodes to the `ProviderNetwork` one by one, you can use node selectors to simplify the workflow. #5518
- Different `NetworkProvider`s can now share the same VLAN. #5471
- Auto create VLAN sub-interfaces. #5966
- Auto move VLAN sub-interfaces to OVS bridges. #5949
- Adding `pod_name` and `pod_namespace` labels to interface metrics. #5463
- IPSec
- Support `cert-manager` to issue certificates. #5365
- Request new certificate if current certificate is not trusted. #5710
- kubectl-ko
- Collect IPSec and xFRM information. #5472
- Replace `Endpoint` with `EndpointSlice`. #5425
- NetworkAttachment caching: reduce APIServer load in large-scale deployments with Multus. #5386
- Upgrade `OVS` to 3.5 and `OVN` to 25.03. #5537

2025 Kube-OVN Team

PDF Slack Support

2026⬚10⬚29⬚

2025⬚9⬚15⬚



GitHub 



8.2.4 ⬚⬚

8.3 ████████

██████████████████Kube-OVN ██████ ██ ████████████████

8.3.1 ███████

Kube-OVN ██████████████████

1. `default` ██████ Pod ██ IP ████████████ CIDR ▯ `10.16.0.0/16` ████ `10.16.0.1` ▯
2. `join` ██████ Node ▯ Pod ████████████████, ██ CIDR ▯ `100.64.0.0/16` ████ `100.64.0.1` ▯

████████████████████████

```
POD_CIDR="10.16.0.0/16"
POD_GATEWAY="10.16.0.1"
JOIN_CIDR="100.64.0.0/16"
EXCLUDE_IPS=""
```

`EXCLUDE_IP` ███ `POD_CIDR` ██████████████████ 192.168.10.20..192.168.10.30 ▯

████ Overlay ████████████████████ Service CIDR ███

████████████████████████████████████ Join ███

8.3.2 Service ████

████ `kube-proxy` ███ iptables ███████ Kube-OVN ████████████ Kube-OVN ████ Service ▯ CIDR ████████████

████████████████

```
SVC_CIDR="10.96.0.0/12"
```

███████

████████████ `kube-ovn-controller` Deployment ████

```
args:
- --service-cluster-ip-range=10.96.0.0/12
```

███████

8.3.3 Overlay ████

██████████████████Kube-OVN ██████ Kubernetes Node IP ████████████████████████████████

████████████████████████████████████

```
IFACE=eth1
```

████████████████████ `ens[a-z0-9]*,eth[a-z0-9]*` ▯

██████████ `kube-ovn-cni` DaemonSet ██████████

```
args:
- --iface=eth1
```

████████████████████████████████ annotation `ovn.kubernetes.io/tunnel_interface` ██████████████████ annotation ██████ `iface` ████████ annotation▯

```
kubectl annotate node no1 ovn.kubernetes.io/tunnel_interface=ethx
```

████████████████████ Overlay ██████████▯

2025 Kube-OVN Team

[illegible] /32 [illegible]

[illegible] `/32` [illegible] `/128` [illegible]Kube-OVN [illegible] IP [illegible] `HOST_TUNNEL_SRC=true` [illegible]

```
HOST_TUNNEL_SRC=true
```

[illegible] `HOST_TUNNEL_SRC=true` [illegible] Kube-OVN daemon[illegible] Helm [illegible] chart [illegible]

```
# charts/kube-ovn
func:
  HOST_TUNNEL_SRC: true

# charts/kube-ovn-v2
features:
  enableHostTunnelSrc: true
```

[illegible] Helm [illegible] `false` [illegible]daemon [illegible] Overlay [illegible]

8.3.4 MTU [illegible]

[illegible] Overlay [illegible]Kube-OVN [illegible] MTU [illegible] MTU [illegible] Overlay [illegible] Pod [illegible] MTU [illegible] MTU - 100[illegible]Underlay [illegible]Pod [illegible] MTU[illegible]

[illegible] Overlay [illegible] MTU [illegible] `kube-ovn-cni` DaemonSet [illegible]

```
args:
- --mtu=1333
```

8.3.5 [illegible]

[illegible]Kube-OVN [illegible] `mirror0` [illegible] tcpdump [illegible]

```
ENABLE_MIRROR=true
```

[illegible] `kube-ovn-cni` DaemonSet [illegible]:

```
args:
- --enable-mirror=true
```

[illegible]

8.3.6 LB [illegible]

[illegible] Underlay [illegible]kube-proxy [illegible] Service [illegible] OVN [illegible] L2 LB [illegible] ClusterIP [illegible] Service [illegible] LB [illegible] ClusterIP [illegible] kube-proxy [illegible] kube-proxy[illegible]

[illegible]

```
ENABLE_LB=false
```

[illegible] `kube-ovn-controller` Deployment [illegible]

```
args:
- --enable-lb=false
```

LB [illegible]

[illegible] Kube-OVN v1.12.0 [illegible] subnet crd [illegible] spec [illegible] `enableLb` [illegible] Kube-OVN [illegible] LB [illegible] LB [illegible] `kube-ovn-controller` Deployment [illegible] `enable-lb` [illegible] load-balancer [illegible] `enableLb` [illegible] load-balancer [illegible] v1.12.0 [illegible] `enableLb` [illegible]

2025 Kube-OVN Team

8.3.7 NetworkPolicy ...

Kube-OVN ... OVN ... ACL ... NetworkPolicy... NetworkPolicy ... ... Cilium Chain ... eBPF ... NetworkPolicy... ... Kube-OVN ... NetworkPolicy ...

...

```
ENABLE_NP=false
```

... `kube-ovn-controller` Deployment ...

```
args:
- --enable-np=false
```

NetworkPolicy ...

8.3.8 EIP ... SNAT ...

... EIP ... SNAT ... `kube-ovn-controller` ... ...

...

```
ENABLE_EIP_SNAT=false
```

... `kube-ovn-controller` Deployment ...

```
args:
- --enable-eip-snat=false
```

EIP ... SNAT ... EIP ... SNAT ...

8.3.9 Load Balancer ... Service ...

... VPC ... Load Balancer ... Service... LoadBalancer ... Service...

...

```
ENABLE_LB_SVC=true
```

... `kube-ovn-controller` Deployment ...

```
args:
- --enable-lb-svc=true
```

8.3.10 ... ECMP ...

... ECMP ... ECMP ... ... `kube-ovn-controller` Deployment ...:

```
args:
- --enable-ecmp=true
```

... Kube-OVN v1.12.0 ... subnet crd ... spec ... `enableEcmp` ... ECMP ... ECMP ... `kube-ovn-controller` Deployment ... `enable-ecmp` ... v1.12.0 ...

...

8.3.11 Kubevirt VM ...

... Kubevirt ... VM ... `kube-ovn-controller` ... StatefulSet Pod ... IP ... ... VM ...

... 1.10.6 ... `kube-ovn-controller` Deployment ...

2025 Kube-OVN Team

```yaml
args:
- --keep-vm-ip=false
```

8.3.12 CNI ▯▯▯▯▯▯

Kube-OVN ▯▯▯▯ `/opt/cni/bin` ▯▯▯▯▯ CNI ▯▯▯▯▯▯ `/etc/cni/net.d` ▯▯▯▯▯ CNI ▯▯▯▯ `01-kube-ovn.conflist` ▯ ▯▯▯▯▯▯▯▯▯▯▯ CNI ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
CNI_CONF_DIR="/etc/cni/net.d"
CNI_BIN_DIR="/opt/cni/bin"
CNI_CONFIG_PRIORITY="01"
```

▯▯▯▯▯▯▯▯ `kube-ovn-cni` DaemonSet ▯ Volume ▯▯▯▯▯▯▯▯▯

```yaml
volumes:
- name: cni-conf
  hostPath:
    path: "/etc/cni/net.d"
- name: cni-bin
  hostPath:
    path:"/opt/cni/bin"
...
args:
- --cni-conf-name=01-kube-ovn.conflist
```

▯▯▯▯▯▯▯▯▯▯▯▯▯ Kube-OVN ▯▯▯▯▯ `loopback` ▯ `portmap` ▯ `macvlan` ▯ `ipvlan` ▯ CNI ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
INSTALL_CNI_PLUGINS=false
```

▯▯▯ `false` ▯▯▯▯▯▯▯▯▯▯Kube-OVN ▯▯▯ CNI ▯▯▯▯▯▯▯▯▯▯▯▯ Helm v1 ▯▯▯ `cni_conf.INSTALL_CNI_PLUGINS: false` ▯▯▯ Helm v2 ▯▯▯ `cni.installCNIPlugins: false` ▯

8.3.13 CNI ▯▯ recompute

`kube-ovn-cni` DaemonSet ▯▯▯▯▯ OVN controller recompute ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
ENABLE_RECOMPUTE=true
```

Helm v1 ▯ Helm v2 ▯▯▯▯▯▯▯▯▯▯

```yaml
# Helm v1
func:
  ENABLE_RECOMPUTE: true

# Helm v2
features:
  enableRecompute: true
```

▯▯▯▯▯▯▯▯ `kube-ovn-cni` DaemonSet ▯▯▯ `--enable-recompute=true` ▯▯▯▯▯daemon ▯ 10 ▯▯▯▯▯▯▯▯▯ recompute▯▯▯▯▯▯ `false` ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

8.3.14 ▯▯▯▯▯▯▯

Kube-OVN ▯▯ Overlay ▯▯▯▯▯▯ Geneve▯▯▯▯▯▯▯ Vxlan ▯ STT▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
TUNNEL_TYPE="vxlan"
```

▯▯▯▯▯▯▯▯ `ovs-ovn` DaemonSet ▯▯▯▯▯▯▯

```yaml
env:
- name: TUNNEL_TYPE
  value: "vxlan"
```

▯▯▯▯▯▯ STT ▯▯▯▯▯▯▯▯ ovs ▯▯▯▯▯▯▯▯▯▯▯▯▯▯

▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

2025 Kube-OVN Team

8.3.15 SSL □□

OVN NB/SB □□□ API □ Kube-OVN secure-serving □□□□□ TLS□TLS □□□□□ □□□□□□□□□□□□□□□□ OVN central/DB □ TLS□□□□ secure-serving □□□□ TLS □□□ cipher suites□

```
ENABLE_SSL=true
TLS_MIN_VERSION="TLS12"
TLS_MAX_VERSION="TLS13"
TLS_CIPHER_SUITES="TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AES_128_GCM_SHA256"
```

`TLS_MIN_VERSION` □ `TLS_MAX_VERSION` □□ `TLS10` □ `TLS11` □ `TLS12` □ `TLS13` □ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□ OVN NB/SB □□□□□□□

`TLS_CIPHER_SUITES` □□□□□□□ Go TLS cipher suite □□□OVN DB □□□□□□ □□□□□□□ OpenSSL □□□□□□□□□□□□□□

```
TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
```

□□ cipher suites □□□□ `kube-ovn-tls` Secret □□□□□□□□□□□□□□□□ `kube-ovn-tls` □□ RSA □□□□□□□□ ECDSA cipher suites □□□□□□□ RSA □□□□□ TLS □□□□□□□□□ ECDSA-only suites□□□□□□□□ ECDSA □□□ □□□

□□ Helm □□□□ chart □ `componentTLSArgs` □□□□□□□□□ `--tls-min-version` □ `--tls-max-version` □ `--tls-cipher-suites` □□□□□□□□□ □□ secure-serving □ □□□□□□□□□□ `TLS_*` □□□□□□□□□□□□□ □□□□□ TLS □□□□□□ secure-serving□□□□□□□ chart □□□□□□ classic chart □□ `func.SECURE_SERVING` □v2 chart □□ `features.enableSecureServing` □□ TLS □□□ cipher suites □□□ `networking` □

classic Helm chart □□□□□□

```yaml
func:
  SECURE_SERVING: true
networking:
  ENABLE_SSL: true
  TLS_MIN_VERSION: TLS12
  TLS_MAX_VERSION: TLS13
  TLS_CIPHER_SUITES:
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_AES_128_GCM_SHA256
```

v2 Helm chart □□□□□□□

```yaml
features:
  enableSecureServing: true
networking:
  enableSsl: true
  tlsMinVersion: TLS12
  tlsMaxVersion: TLS13
  tlsCipherSuites:
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_AES_128_GCM_SHA256
```

□ `ENABLE_SSL=true` □□ `networking.enableSsl=true` □□□Kube-OVN □□ `kube-ovn-tls` Secret□□ Secret □□ `cacert` □ `cert` □ `key` □□□□□ `/var/run/tls` □ classic Helm chart □□□□□□ v2 Helm chart □ Secret □□□□□ □□□□□□□ classic Helm chart □ `installMode: dataPlaneOnly` □□□□□□□ `networking.ENABLE_SSL=true` □□□□□□□□□□□□□□□ `kube-ovn-tls` □□□□ □□□□ OVN central □□□□□ CA□□□□□□□classic chart □□□□□□ Secret □□□□□□□□□□□□□ Secret□v2 Helm chart □□ `installMode: dataPlaneOnly` □□□□□□□□□□□□□□□ v2 chart□

□□ SSL □□ `kube-ovn-controller` □□□□□□□ `kube-ovn-tls` □ Secret□□ Secret □□□□□ `cacert` □ `cert` □ `key` □□□□□□ Secret □□□□□□□ Kube-OVN □□□ hash annotation□controller □□□□□□□□□□□□□□□□□□

□□□□□□□ `KUBE_OVN_TLS_ROTATION_INTERVAL` □□□□□□□ `8760h` □□□□□□□□ `0` □□□

```
ENABLE_SSL=true
KUBE_OVN_TLS_ROTATION_INTERVAL=8760h
```

□□ Helm □□□□□□□□□ Chart □□□classic Chart□ `charts/kube-ovn` □□□□□□□

```
networking:
  ENABLE_SSL: true
  KUBE_OVN_TLS_ROTATION_INTERVAL: 8760h
```

classic Chart 使用 `installMode: dataPlaneOnly` 时会强制将该间隔设为 `0`，因为 `kube-ovn-tls` 由管理集群维护并负责轮转。

v2 Chart（`charts/kube-ovn-v2`）使用 camel case 键：

```
networking:
  enableSsl: true
  kubeOvnTlsRotationInterval: 8760h
```

Secret 更新后，`ovn-central`、`ovs-ovn` 中的 OVN 进程以及 Kube-OVN 的 Go client 会重新加载挂载的证书文件或在下一次 TLS 握手时读取新证书。该过程不需要重启 Kube-OVN Pod；可以检查 `kube-ovn-tls` 的 `kube-ovn.io/kube-ovn-tls-cert-hash` annotation、组件日志和 Pod 的容器重启次数，确认 adoption 或轮转已经生效。

8.3.16 跳过 conntrack 的目的 CIDR 列表

针对部分需要绕过 conntrack 处理的目的网段（例如已通过其他方式做 NAT/会话保持的旁路流量），可在安装脚本或 `kube-ovn-controller` 启动参数中配置跳过列表，多个 CIDR 以逗号分隔：

```
SKIP_CONNTRACK_DST_CIDRS="10.10.0.0/16,fd00:10::/64"
```

```
args:
- --skip-conntrack-dst-cidrs=10.10.0.0/16,fd00:10::/64
```

默认为空。

8.3.17 启用 OVN IPSec

启用后 Kube-OVN 会通过 OVN 内置的 IPSec 能力对节点间的 Geneve/Vxlan/STT 隧道进行加密，并自动签发证书。

```
ENABLE_OVN_IPSEC=true
```

控制器与 daemon 均需要打开该开关：

```
args:
- --enable-ovn-ipsec=true
```

详细配置参考 OVN IPSec。

8.3.18 AdminNetworkPolicy / DNS 名称解析支持

启用 AdminNetworkPolicy（ANP/CNP）以及基于 DNS 名称的访问控制（依赖 `DNSNameResolver` CRD）：

```
ENABLE_ANP=true
ENABLE_DNS_NAME_RESOLVER=true
```

```
args:
- --enable-anp=true
- --enable-dns-name-resolver=true
```

详细使用见基于域名的访问控制。

8.3.19 VPC NAT Gateway 总开关

控制是否启用 VPC NAT Gateway 相关 controller。默认开启：

```
ENABLE_NAT_GW=true
```

或在 `ovn-vpc-nat-gw-config` ConfigMap 中通过 `enable-vpc-nat-gw` 字段控制运行时开关。

8.3.20 ovn-northd 线程数

针对大规模集群，可调整 `ovn-northd` 处理线程数以加速增量计算：

2025 Kube-OVN Team

```
OVN_NORTHD_N_THREADS=4
```

▯▯ 1▯▯▯▯▯▯ CPU ▯▯▯▯▯

8.3.21 NetworkPolicy ▯▯▯▯

Kube-OVN ▯▯ `standard` / `lax` ▯▯ NetworkPolicy ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
NP_ENFORCEMENT=standard
```

```yaml
args:
- --np-enforcement=standard
```

▯▯▯▯▯ NetworkPolicy ▯▯▯ `ovn.kubernetes.io/network_policy_enforcement` annotation ▯▯▯▯▯ NetworkPolicy ▯▯▯

8.3.22 VXLAN ▯▯ TX checksum offload

▯▯▯▯ VXLAN ▯▯▯▯▯▯ checksum offload ▯▯▯▯▯▯▯▯▯

```
SET_VXLAN_TX_OFF=true
```

```yaml
args:
- --set-vxlan-tx-off=true
```

8.3.23 Geneve ▯▯ TX checksum offload

▯▯▯▯▯▯▯▯▯ Geneve ▯▯▯▯▯▯▯▯▯▯ checksum offload ▯▯▯▯▯▯▯▯▯▯Kube-OVN ▯▯ Geneve TX checksum offload ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
SET_GENEVE_TX_OFF=true
```

▯▯▯▯▯▯▯▯▯▯ `kube-ovn-daemon` ▯ `--set-geneve-tx-off` ▯▯▯▯▯▯▯▯▯▯ `kube-ovn-cni` DaemonSet ▯▯▯▯

```yaml
args:
- --set-geneve-tx-off=true
```

▯▯ Helm ▯▯▯▯classic chart ▯ v2 chart ▯▯▯▯▯▯▯▯▯

```yaml
# classic chart
func:
  SET_GENEVE_TX_OFF: true

# v2 chart
features:
  SET_GENEVE_TX_OFF: true
```

▯▯▯▯daemon ▯▯▯▯▯ `genev_sys_6081` ▯▯ TX checksum offload▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `true` ▯▯ `false` ▯▯▯▯▯▯▯▯▯▯ offload▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ offload ▯▯▯ `SET_GENEVE_TX_OFF` ▯ `SET_VXLAN_TX_OFF` ▯▯▯▯ Geneve ▯ VXLAN ▯▯▯▯▯▯▯▯

8.3.24 KubeVirt ▯▯▯▯▯▯

▯▯▯▯▯▯▯▯ KubeVirt ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

```
ENABLE_LIVE_MIGRATION_OPTIMIZE=true
```

8.3.25 OVN LB ▯▯▯▯ endpoint

▯▯▯ OVN ▯▯ LoadBalancer ▯▯▯▯▯▯▯ endpoint▯▯▯ `externalTrafficPolicy=Local` ▯▯▯

```
ENABLE_OVN_LB_PREFER_LOCAL=true
```

```yaml
args:
- --enable-ovn-lb-prefer-local=true
```


2025 Kube-OVN Team

8.3.26 ConfigMap

ConfigMap `external-gw-config` `kube-system`

```
EXTERNAL_GATEWAY_CONFIG_NS=kube-system
```

8.3.27 Non-Primary CNI

Kube-OVN attachment network Pod IP CNI controller daemon

```
args:
- --non-primary-cni-mode=true
```

Helm `cni_conf.NON_PRIMARY_CNI=true` chart v1/ `cni.nonPrimaryCNI=true` chart v2 Non-Primary CNI

8.3.28 Controller Leader Election

controller Kubernetes leader election duration leader controller

```
args:
- --leader-elect-lease-duration=30s
- --leader-elect-renew-deadline=20s
- --leader-elect-retry-period=6s
```

`30s` `20s` `6s` durationcontroller Kubernetes client

- `lease-duration` `renew-deadline`
- `renew-deadline` `retry-period` Kubernetes client-go leader election jitter factor `1.2`

```
export LEADER_ELECT_LEASE_DURATION=1m
export LEADER_ELECT_RENEW_DEADLINE=40s
export LEADER_ELECT_RETRY_PERIOD=10s
bash install.sh
```

Helm chart classic chart `kube-ovn-controller.leaderElection`

```
kube-ovn-controller:
  leaderElection:
    leaseDuration: "30s"
    renewDeadline: "20s"
    retryPeriod: "6s"
```

v2 chart `controller.leaderElection`

```
controller:
  leaderElection:
    leaseDuration: "30s"
    renewDeadline: "20s"
    retryPeriod: "6s"
```

8.3.29 kube-ovn-monitor

`kube-ovn-monitor` Helm Deployment

```
# classic chart
kube-ovn-monitor:
  replicas: 1

# v2 chart
monitor:
  replicas: 1
```

classic chart `kube-ovn-monitor.replicas` v2 chart `monitor.replicas` `1` OVN central central OVN Raft `kube-ovn-monitor`

Deployment master Pod master central Pod `Pending`

2025 Kube-OVN Team

Helm Deployment chart 1 Helm values kubectl scale Deployment

8.3.30 kube-ovn-v2 Deployment

kube-ovn-v2 Chart

- controller.tolerations kube-ovn-controller NoSchedule operator: Exists key: CriticalAddonsOnly operator: Exists
- central.tolerations ovn-central NoExecute operator: Exists
- monitor.tolerations kube-ovn-monitor controller.tolerations
- validatingWebhook.tolerations kube-ovn-webhook NoSchedule operator: Exists
- ic.tolerations ovn-ic-controller central.tolerations

Pod spec.template.spec.tolerations central.hcp.enabled false central Deployment true StatefulSet ic features.enableOvnInterconnections values key effect node.kubernetes.io/unschedulable master drain cordon Pod cordon key

```
controller:
  tolerations:
    - key: node-role.kubernetes.io/control-plane
      operator: Exists
      effect: NoSchedule
    - key: node-role.kubernetes.io/master
      operator: Exists
      effect: NoSchedule
    - key: CriticalAddonsOnly
      operator: Exists

central:
  tolerations:
    - key: node-role.kubernetes.io/control-plane
      operator: Exists
      effect: NoSchedule
    - key: node-role.kubernetes.io/master
      operator: Exists
      effect: NoSchedule
    - key: CriticalAddonsOnly
      operator: Exists
```

monitor validatingWebhook ic ovs-ovn kube-ovn-agent kube-ovn-pinger kube-ovn-speaker DaemonSet

8.3.31 ip

kube-ovn-controller/kube-ovn-cni/kube-ovn-monitor ip 0.0.0.0 ip

```
ENABLE_BIND_LOCAL_IP=true
```

kube-ovn-monitor pod ip

```
# netstat -tunlp |grep kube-ovn
tcp        0      0 172.18.0.5:10661        0.0.0.0:*               LISTEN      2612/./kube-ovn-mon
```

deployment daemonSet

```
env:
- name: ENABLE_BIND_LOCAL_IP
  value: "false"
```

PDF Slack Support

2025 Kube-OVN Team

2026⬜11⬜7⬜

2022⬜5⬜24⬜

GitHub    

8.3.32 ⬜⬜

8.4 [illegible]

Kube-OVN 目前主要发布 Minor 版本和 Patch 版本。Minor 版本中会有新功能的增加，OVN/OVS 大版本升级，内部架构调整，API 变化。Patch 版本主要进行 Bug 修复，安全漏洞修复，依赖升级，同时兼容之前的 API。

8.4.1 维护策略

Kube-OVN 目前会持续维护主干分支和最近两个发版分支，例如 `master`，`release-1.12`，`release-1.11`。其中最新发版分支（例如 `release-1.12`）会进行较为频繁的迭代和发版，所有 Bug 修复，安全漏洞和依赖升级都会尽可能 backport 到最新发版分支。

前一个发版分支（例如 `release-1.11`）会 backport 较为重大的 Bug 修复以及影响面较大的安全漏洞修复。

8.4.2 发版周期

Minor 版以主干分支是否有重大新功或重大的架构调整完成为契机按需发版，目前约为半年一次发布。Patch 版本根据分支 Bug 修复情况触发，一般会在 Bug 修复进入后一周内发布。

8.4.3 Patch 版本发布方式

目前 Patch 版本大部分工作都可以通过 hack/release.sh 脚本以自动化的方式来实现，主要发布步骤描述如下:

1. 检查当前分支 Build 情况（自动化）
2. 推送新 tag 镜像到 Docker Hub（自动化）
3. 推送新 tag 代码到 Github（自动化）
4. 修改代码中的版本信息（自动化）
5. 修改文档仓库版本信息（自动化）
6. 生成 Release Note PR（自动化）
7. 合并 Release Note (手动)
8. 手动 Merge github action 生成的 Release Note PR
9. 修改 Github Release 信息（手动）
10. 在 Github Release 页面编辑新生成的 Release，将标题修改为对应版本号（例如 `v1.12.12`），并复制上一步生成的 Release Note 到 Release 详情

8.4.4 Minor 版本发布方式

目前 Minor 分支主要工作还需要通过手动的方式来完成，主要发布步骤描述如下:

1. 在 Github 上推送一个新的发布分支，例如 `release-1.13` (手动)
2. 将主干分支的 `VERSION`，`dist/images/install.sh`，`charts/kube-ovn/values.yaml` 和 `charts/kube-ovn/Chart.yaml` 中的版本信息修改为下个 Minor 版本，例如 `v1.14.0` (手动)
3. 推送新 tag 镜像到 Docker Hub (手动)
4. 在发版分支推送新 tag 代码到 Github (手动)
5. 在文档仓库新建发版分支，例如 `v1.13`，修改 `mkdocs.yml` 文件中的 `version` 和 `branch` 信息 (手动)
6. 生成 Release Note PR（自动化）
7. 合并 Release Note (手动)
8. 手动 Merge github action 生成的 Release Note PR
9. 修改 Github Release 信息（手动）
10. 在 Github Release 页面编辑新生成的 Release，将标题修改为对应版本号（例如 `v1.13.0`），并复制上一步生成的 Release Note 到 Release 详情
11. 修改发版分支的 `VERSION` 文件版本信息为下一个 Patch 版本，例如 `v1.13.1`

PDF Slack Support

2025 Kube-OVN Team

2024⬚5⬚16⬚

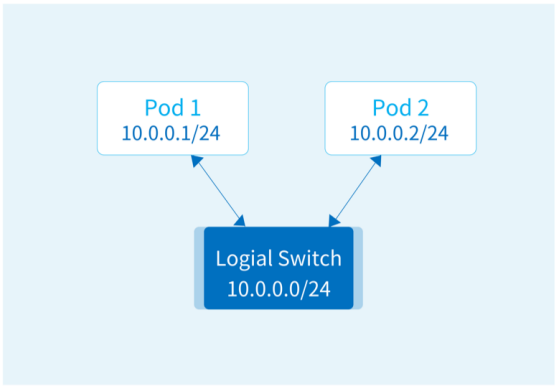
2024⬚5⬚8⬚



GitHub 



8.4.5 ⬚⬚



8.5 Underlay □□□□

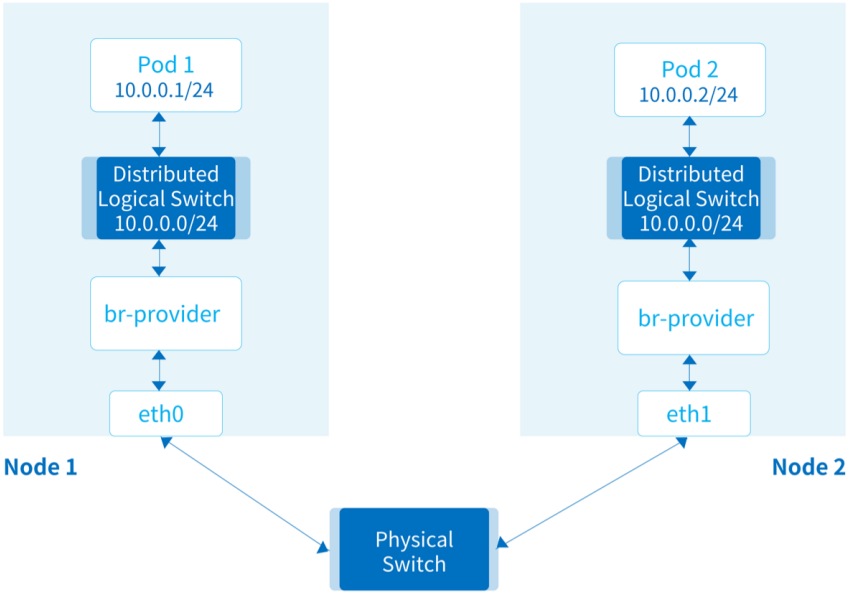
本文档介绍 Underlay 模式下流量在不同情况下的转发路径。

8.5.1 □□□□□□

内部逻辑交换机直接交换数据包，不进入外部网络。

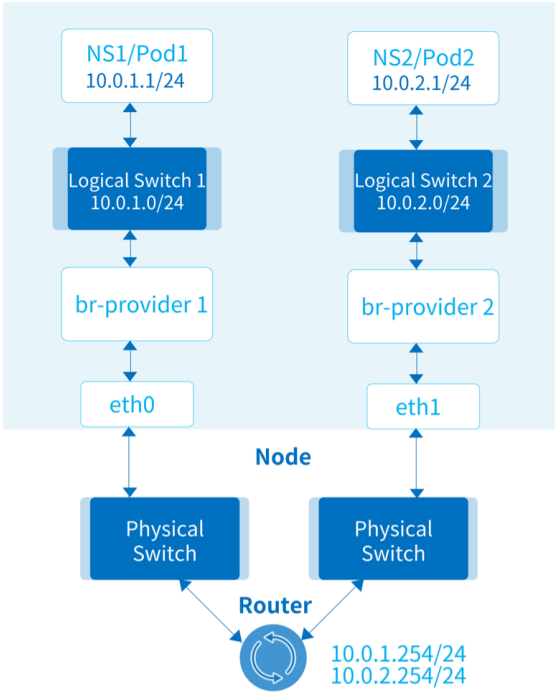
8.5.2 □□□□□□

数据包经由节点网卡进入外部交换机，由外部交换机进行交换。



8.5.3 ████████

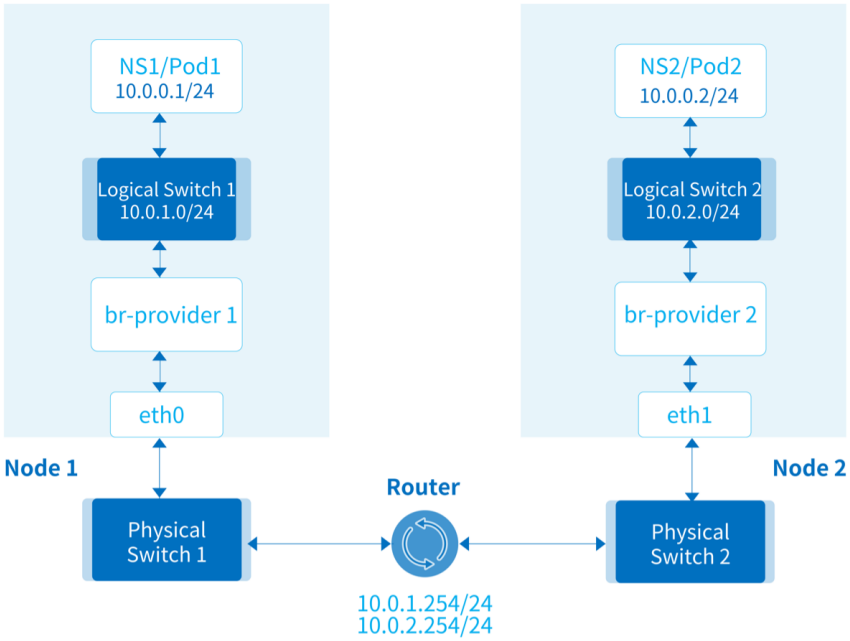
██



> ██ br-provider-1 ▯ br-provider-2 ██████ OVS ████████████████████ Provider Network▯

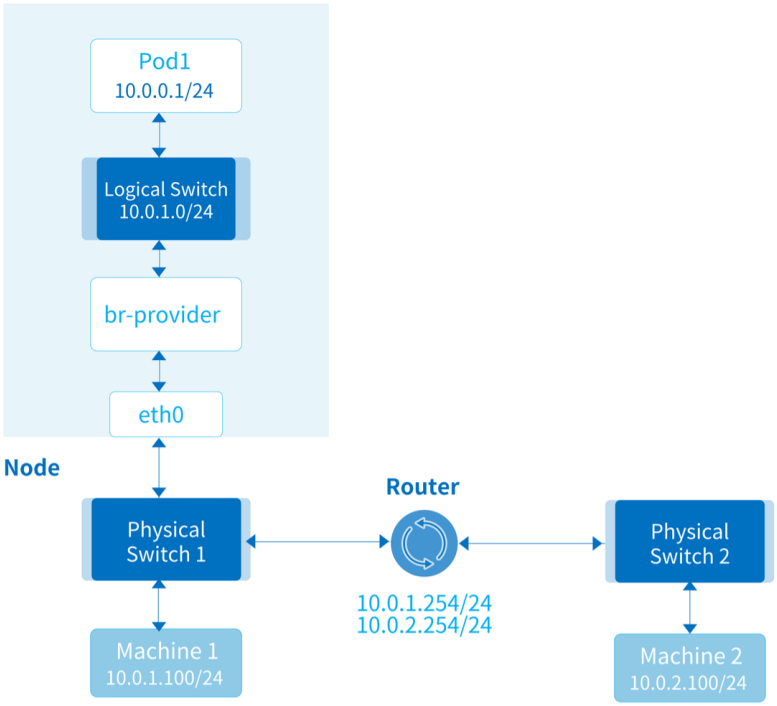
8.5.4 ████████

██


NS1/Pod1
10.0.0.1/24
Logical Switch 1
10.0.1.0/24
br-provider 1
eth0
Node 1
Physical
Switch 1
Router
10.0.1.254/24
10.0.2.254/24
NS2/Pod2
10.0.0.2/24
Logical Switch 2
10.0.2.0/24
br-provider 2
eth1
Node 2
Physical
Switch 2

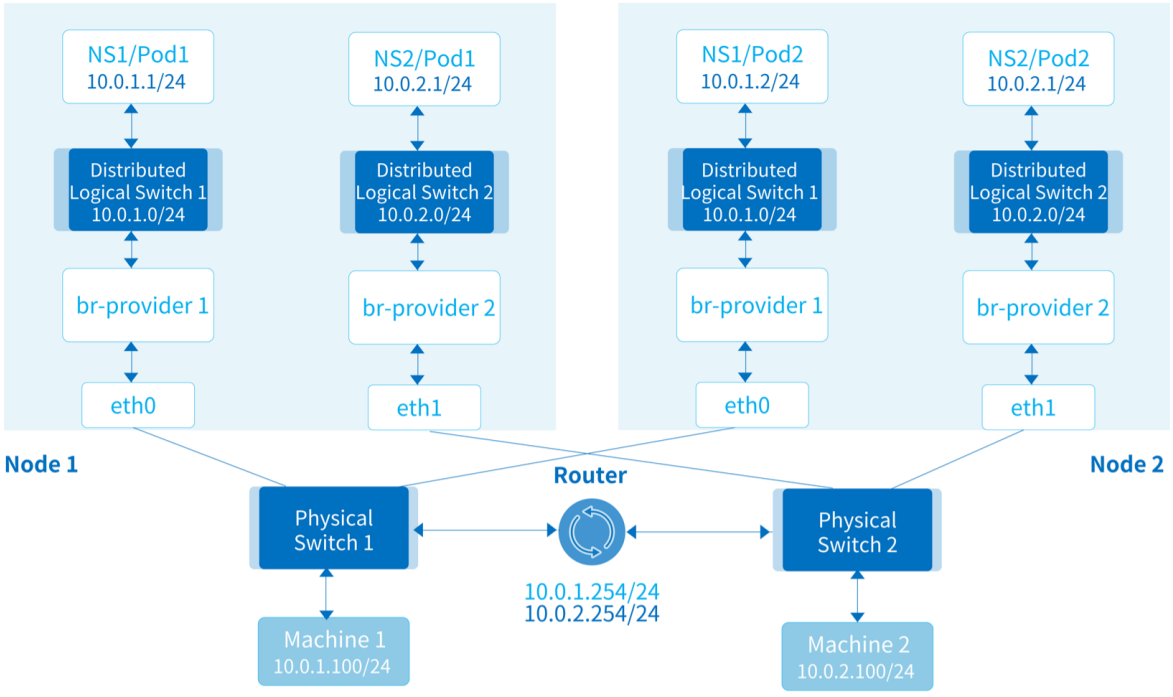
8.5.5 ████

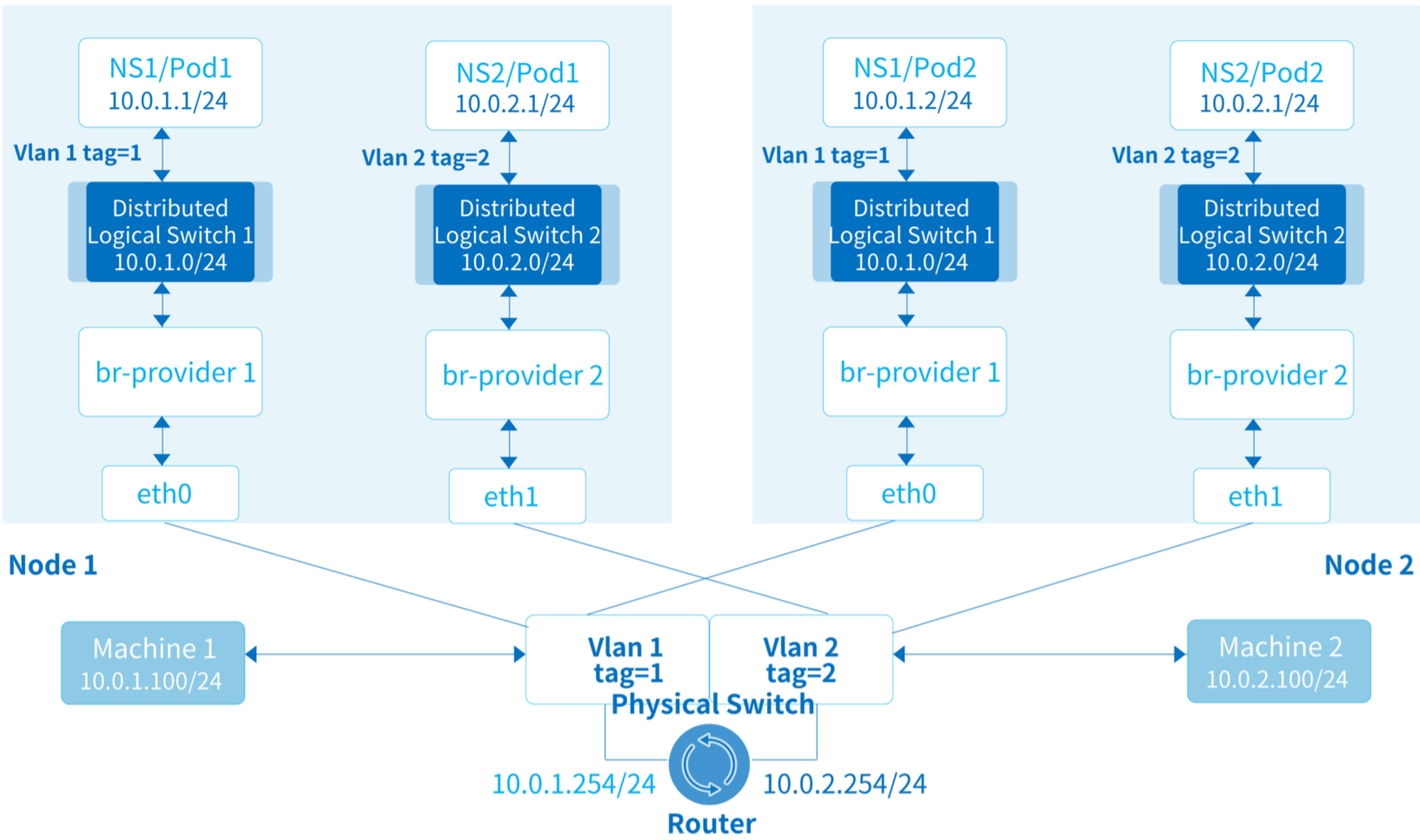
██



Pod

8.5.6 Vlan Tag





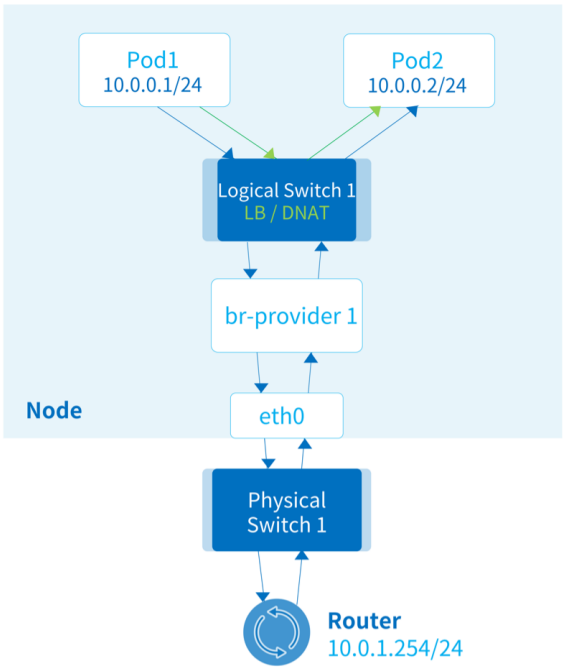
8.5.7 VLAN

8.5.8 Pod Service IP

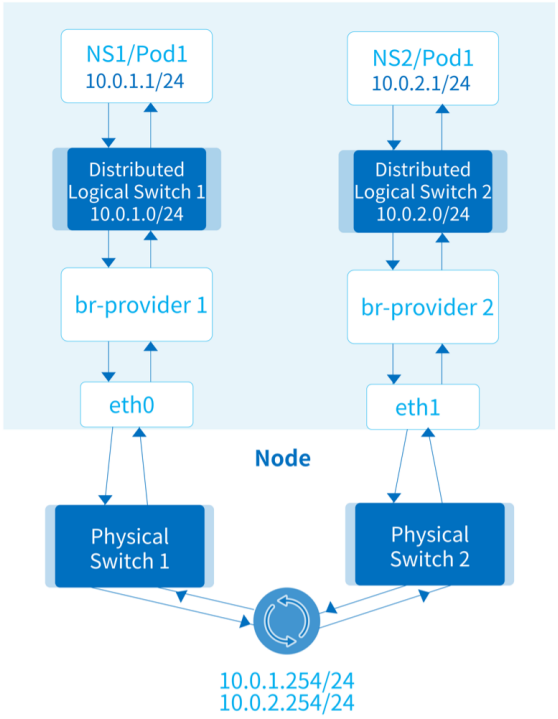
Kube-OVN 为每个 Kubernetes Service 在每个子网的逻辑交换机上配置了负载均衡。当 Pod 通过访问 Service IP 访问其它 Pod 时，会构造一个目的地址为 Service IP、目的 MAC 地址为网关 MAC 地址的网络包。网络包进入逻辑交换机后，负载均衡会对网络包进行拦截和 DNAT 处理，将目的 IP 和端口修改为 Service 对应的某个 Endpoint 的 IP 和端口。由于逻辑交换机并未修改网络包的二层目的 MAC 地址，网络包在进入外部交换机后仍然会送到外部网关，此时需要外部网关对网络包进行转发。

2025 Kube-OVN Team

Service ▯▯▯▯▯▯▯▯▯▯ Pod



Service ▯▯▯▯▯▯▯▯▯▯ Pod



PDF Slack Support

2025 Kube-OVN Team

2022▯9▯23▯

2022▯5▯20▯

GitHub 

8.5.9 ▯▯

2025 Kube-OVN Team

8.6 ▯▯ Netfilter ▯▯

Kube-OVN 使用网关 Netfilter 后端实现默认 VPC 下容器网络（Overlay）的网关 NAT 功能。后端可以使用 `iptables` 或 `nftables`，本文同时保留 iptables 后端的规则参考。

8.6.1 ▯▯▯▯▯▯

`kube-ovn-cni` daemon 支持三种网关 Netfilter 模式：

模式	行为
`auto`	自动检测本地 kube-proxy 模式：`iptables` 和 `ipvs` 使用 iptables 后端；`nftables` 使用 nftables 后端。
`iptables`	始终使用 Kube-OVN 的 iptables 和 ipset 规则。
`nftables`	始终使用 Kube-OVN 的 nftables 表和规则。

默认模式为 `auto`。冷启动时会轮询 kube-proxy 的 `proxyMode` 接口；如果在检测超时前接口不可用，则读取 `kube-system` 中的 `kube-proxy` ConfigMap。默认接口为 `http://localhost:10249/proxyMode`，默认超时为 30 秒。运行期间连续三次观察到模式变化后才会切换后端，避免一次临时响应触发切换。

运行 `bash install.sh` 前，可以通过环境变量设置模式和检测参数；也可以在 `kube-ovn-cni` DaemonSet 中设置：

```
export GATEWAY_NETFILTER_MODE=auto
export KUBE_PROXY_MODE_ENDPOINT=http://localhost:10249/proxyMode
export GATEWAY_NETFILTER_DETECT_TIMEOUT=30s
```

```
args:
- --gateway-netfilter-mode=auto
- --kube-proxy-mode-endpoint=http://localhost:10249/proxyMode
- --gateway-netfilter-detect-timeout=30s
```

使用 Helm 安装时，根据 Chart 版本设置对应的值：

```
# charts/kube-ovn
networking:
  GATEWAY_NETFILTER_MODE: auto
  KUBE_PROXY_MODE_ENDPOINT: http://localhost:10249/proxyMode
  GATEWAY_NETFILTER_DETECT_TIMEOUT: 30s
```

```
# charts/kube-ovn-v2
agent:
  gatewayNetfilterMode: auto
  kubeProxyModeEndpoint: http://localhost:10249/proxyMode
  gatewayNetfilterDetectTimeout: 30s
```

nftables 后端在适用的 IPv4 / IPv6 nftables family 中分别管理 Kube-OVN 自有的 `kube-ovn` 表，用于 Service 和 NodePort NAT、子网 NAT / NAT policy / TProxy / filter 和 mangle 规则以及子网流量统计。它不会读取或修改 kube-proxy 私有的 nftables 对象或标记，也不会清理 conntrack 表项。

后端切换时，Kube-OVN 会先准备并同步目标后端，再清理旧后端。切换和卸载时只清理 Kube-OVN 自有的 iptables / ipset 和 nftables 对象。如果清理失败，会保留当前后端并在后续 reconcile 中重试。

8.6.2 Nftable LoadBalancer Service

控制器参数 `--enable-nftable-lb-svc` 用于启用由 VPC NAT Gateway nftable share-DNAT 实现的 LoadBalancer Service。当前 master 中该参数默认值为 `true`，且需要全局参数 `--enable-lb=true` 才能生效。

对应的 Helm 配置键如下：

```
# charts/kube-ovn
func:
  ENABLE_NFTABLE_LB_SVC: true

# charts/kube-ovn-v2
features:
  enableNftableLoadbalancerService: false
```

v2 chart 仅在 `enableNftableLoadbalancerService` 为 true 时渲染控制器参数。当该值为 false 时会省略参数，而控制器默认值仍为 true；在 chart 提供有效的关闭方式前，如需关闭应显式设置控制器参数。

nftable LoadBalancer Service 进入终止状态时，会立即释放其 `EIP:port:protocol` 标识。后续 Service 可以复用该标识并被编程，无需等待已删除 Service 的 finalizer 完成。

2025 Kube-OVN Team

□□□ ipset □□□□□□

□□□**IPv4/IPv6**□	□□	□□□□
ovn40services/ovn60services	hash:net	Service □□
ovn40subnets/ovn60subnets	hash:net	Overlay □□□□□□ NodeLocal DNS IP □□
ovn40subnets-nat/ovn60subnets-nat	hash:net	□□ NatOutgoing □ Overlay □□□□
ovn40subnets-distributed-gw/ovn60subnets-distributed-gw	hash:net	□□□□□□□□ Overlay □□□□
ovn40other-node/ovn60other-node	hash:net	□□□□□□□ IP □□
ovn40local-pod-ip-nat/ovn60local-pod-ip-nat	hash:ip	□□□
ovn40subnets-nat-policy	hash:net	□□□ natOutgoingPolicyRules □□□□□□□
ovn40natpr-418e79269dc5-dst	hash:net	natOutgoingPolicyRules □ rule □□□ dstIPs
ovn40natpr-418e79269dc5-src	hash:net	natOutgoingPolicyRules □ rule □□□ srcIPs

□□□ iptables □□□IPv4□□□□□□□

表	链	规则	作用	备注
filter	INPUT	-m set --match-set ovn40services src -j ACCEPT	允许 k8s Service 到 Pod 的流量	--
filter	INPUT	-m set --match-set ovn40services dst -j ACCEPT	允许	--
filter	INPUT	-m set --match-set ovn40subnets src -j ACCEPT	允许	--
filter	INPUT	-m set --match-set ovn40subnets dst -j ACCEPT	允许	--
filter	FORWARD	-m set --match-set ovn40services src -j ACCEPT	允许	--
filter	FORWARD	-m set --match-set ovn40services dst -j ACCEPT	允许	--
filter	FORWARD	-m set --match-set ovn40subnets src -j ACCEPT	允许	--
filter	FORWARD	-m set --match-set ovn40subnets dst -j ACCEPT	允许	--
filter	FORWARD	-s 10.16.0.0/16 -m comment --comment "ovn-subnet-gateway,ovn-default"	允许从 subnet 发出的流量	10.16.0.0/16 为 subnet 的 cidr 由comment 标记为 ovn-subnet-gateway 用于在 iptables 中区分 subnet 的流量，默认为 ovn-default 可 subnet 配置
filter	FORWARD	-d 10.16.0.0/16 -m comment --comment "ovn-subnet-gateway,ovn-default"	允许发往 subnet 的流量	可
filter	OUTPUT	-p udp -m udp --dport 6081 -j MARK --set-xmark 0x0	清除 SNAT	UDP: bad checksum on VXLAN interface
nat	PREROUTING	-m comment --comment "kube-ovn prerouting rules" -j OVN-PREROUTING	跳转 OVN-PREROUTING 链	--
nat	POSTROUTING	-m comment --comment "kube-ovn postrouting rules" -j OVN-POSTROUTING	跳转 OVN-POSTROUTING 链	--
nat	OVN-PREROUTING	-i ovn0 -m set --match-set ovn40subnets src -m set --match-set ovn40services dst -j MARK --set-xmark 0x4000/0x4000	为 Pod 访问 Service 的流量 masquerade 标记	用于后续 LB 规则
nat	OVN-PREROUTING	-p tcp -m addrtype --dst-type LOCAL -m set --match-	为 kube-proxy 的 ipvs 模式流量	

		set KUBE-NODE-PORT-LOCAL-TCP dst -j MARK --set-xmark 0x80000/0x80000	ExternalTrafficPolicy Local Service TCP	
nat	OVN-PREROUTING	-p udp -m addrtype --dst-type LOCAL -m set --match-set KUBE-NODE-PORT-LOCAL-UDP dst -j MARK --set-xmark 0x80000/0x80000	ExternalTrafficPolicy Local Service UDP	
nat	OVN-POSTROUTING	-m set --match-set ovn40services src -m set --match-set ovn40subnets dst -m mark --mark 0x4000/0x4000 -j SNAT --to-source	Service IP Overlay Pod IP IP	kube-proxy ipvs
nat	OVN-POSTROUTING	-m mark --mark 0x4000/0x4000 -j MASQUERADE	SNAT	--
nat	OVN-POSTROUTING	-m set --match-set ovn40subnets src -m set --match-set ovn40subnets dst -j MASQUERADE	Pod Service SNAT	--
nat	OVN-POSTROUTING	-m mark --mark 0x80000/0x80000 -m set --match-set ovn40subnets-distributed-gw dst -j RETURN	ExternalTrafficPolicy Local Service Endpoint SNAT	--
nat	OVN-POSTROUTING	-m mark --mark 0x80000/0x80000 -j MASQUERADE	ExternalTrafficPolicy Local Service Endpoint SNAT	--
nat	OVN-POSTROUTING	-p tcp -m tcp --tcp-flags SYN NONE -m conntrack --ctstate NEW -j RETURN	Pod IP SNAT	--
nat	OVN-POSTROUTING	-s 10.16.0.0/16 -m set ! --match-set ovn40subnets dst -j SNAT --to-source 192.168.0.101	Pod NatOutgoing IP SNAT	10.16.0.0/16 192.168.0.101 IP
nat	OVN-POSTROUTING	-m set --match-set ovn40subnets-nat src -m set ! --match-set ovn40subnets dst -j MASQUERADE	Pod NatOutgoing SNAT	--
nat	OVN-POSTROUTING	-m set --match-set ovn40subnets-nat-policy src -m set ! --match-set ovn40subnets dst -j OVN-NAT-POLICY	Pod natOutgoingPolicyRules SNAT	natOutgoingPolicyRules OVN-NAT-POLICY
nat				

2025 Kube-OVN Team

⬚	⬚	⬚⬚	⬚⬚	⬚⬚
	OVN-POSTROUTING	-m mark --mark 0x90001/0x90001 -j MASQUERADE --random-fully		⬚ OVN-NAT-POLICY ⬚⬚⬚⬚⬚⬚⬚⬚ tag 0x90001/0x90001 ⬚⬚⬚ SNAT
nat	OVN-POSTROUTING	-m mark --mark 0x90002/0x90002 -j RETURN	⬚⬚	⬚ OVN-NAT-POLICY ⬚⬚⬚, ⬚⬚⬚⬚⬚ tag 0x90002/0x90002 ⬚⬚⬚ SNAT
nat	OVN-NAT-POLICY	-s 10.0.11.0/24 -m comment --comment natPolicySubnet-net1 -j OVN-NAT-PSUBNET-aa98851157c5	⬚⬚	10.0.11.0/24 ⬚⬚⬚⬚ net1 ⬚ CIDR⬚ OVN-NAT-PSUBNET-aa98851157c5 ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ natOutgoingPolicyRules ⬚⬚
nat	OVN-NAT-PSUBNET-xxxxxxxxxxxx	-m set --match-set ovn40natpr-418e79269dc5-src src -m set --match-set ovn40natpr-418e79269dc5-dst dst -j MARK --set-xmark 0x90002/0x90002	⬚⬚	418e79269dc5 ⬚⬚ natOutgoingPolicyRules ⬚⬚⬚⬚⬚⬚ ID⬚⬚⬚⬚⬚ status.natOutgoingPolicyRules[index].Rule ⬚⬚⬚⬚ ⬚⬚ srcIPs ⬚⬚ ovn40natpr-418e79269dc src⬚ dstIPS ⬚⬚ ovn40natpr-418e79269dc5-dst ⬚⬚⬚ tag 0x90002
mangle	OVN-OUTPUT	-d 10.241.39.2/32 -p tcp -m tcp --dport 80 -j MARK --set-xmark 0x90003/0x90003	⬚ kubelet ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ ⬚ tproxy	
mangle	OVN-PREROUTING	-d 10.241.39.2/32 -p tcp -m tcp --dport 80 -j TPROXY --on-port 8102 --on-ip 172.18.0.3 --tproxy-mark 0x90004/0x90004	⬚ kubelet ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ ⬚ tproxy	

PDF Slack Support

2026⬚10⬚29⬚

2022⬚9⬚6⬚

GitHub



8.6.3 ⬚⬚

2025 Kube-OVN Team

8.7 □□□□□□□□

8.7.1 贡献流程

Kube-OVN 并没有复杂的贡献流程，一切工作都可以在 Github 上发生。当你想提交新的功能/修复某个问题，只需要在 Github 上创建 Issue 和 PR，经过 Maintainer 的 Review 且所有 Github Action 通过，代码就可以被合并。

8.7.2 环境准备

Kube-OVN 使用 Go 开发并使用 Go Modules 管理依赖，请确认环境变量 `GO111MODULE="on"` 。

golangci-lint 被用来扫描代码规范性问题，需要在开发环境安装，请参考local-installation。

为了降低最终生成镜像大小，Kube-OVN 使用了部分 Docker buildx 试验特性，请更新 Docker 至最新版本并开启 buildx:

```
docker buildx create --use
```

8.7.3 构建镜□

使用下面的命令下载代码，并生成运行 Kube-OVN 所需镜像:

```
git clone https://github.com/kubeovn/kube-ovn.git
cd kube-ovn
make release
```

如需构建在 ARM 环境下运行的镜像，请执行下面的命令:

```
make release-arm
```

8.7.4 构建 base 镜像

如需要更改操作系统版本，依赖库，OVS/OVN 代码等，需要对 base 镜像进行重新构建。

base 镜像使用的 Dockerfile 为 `dist/images/Dockerfile.base` 。

构建方法:

```
# build x86 base image
make base-amd64

# build arm base image
make base-arm64
```

8.7.5 运行 E2E

Kube-OVN 使用:

- KIND 来构建本地 Kubernetes 集群: `go install sigs.k8s.io/kind@latest`
- jinjanator 来渲染模板: `pip install jinjanator`
- Ginkgo 来运行测试代码: `go install github.com/onsi/ginkgo/v2/ginkgo; go get github.com/onsi/gomega/...`

> 详情请参考相关文档进行依赖安装。

本地执行 E2E 测试:

```
make kind-init
make kind-install
make e2e
```

如需运行 Underlay E2E 测试，执行下列命令:

2025 Kube-OVN Team

```
make kind-init
make kind-install-underlay
make e2e-underlay-single-nic
```

▯▯▯▯ ovn vpc nat gw eip, fip, snat, dnat ▯▯▯▯▯▯▯▯▯▯

```
make kind-init
make kind-install
make ovn-vpc-nat-gw-conformance-e2e
```

▯▯▯▯ iptables vpc nat gw eip, fip, snat, dnat ▯▯▯▯▯▯▯▯▯▯

```
make kind-init
make kind-install-vpc-nat-gw
make iptables-vpc-nat-gw-conformance-e2e
```

▯▯▯▯ loadbalancer service ▯▯▯▯▯▯▯▯▯▯

```
make kind-init
make kind-install-lb-svc
make kube-ovn-lb-svc-conformance-e2e
```

▯▯▯▯▯▯▯▯▯▯▯▯

```
make kind-clean
```

PDF Slack Support

2025▯6▯29▯

2022▯5▯20▯

GitHub

+1

8.7.6 ▯▯

8.8 [illegible] OVS/OVN [illegible]

上游 OVN/OVS 最初设计目标为通用 SDN 控制器和数据平面。由于 Kubernetes 网络存在一些特殊的用法，并且 Kube-OVN 只重点使用了部分功能，为了达到更好的性能、稳定性和特定的功能，Kube-OVN 对上游 OVN/OVS 做了部分修改。用户如果使用自己的 OVN/OVS 配合 Kube-OVN 的控制器进行工作时需要注意下述的改动可能造成的影响。

未合入上游修改：

- 4228eab1d7 修复 vswitchd ofport_usage 内存泄露并定期修剪内存。
- 54056ea65d 调整选举 timer，避免大规模集群选举抖动。
- 6b4dcb311f 添加 fdb 更新日志。
- f627b7721e 修复 hairpin 环境下 fdb 学习错误的问题。
- 3f3e3a436f 为 ovsdb-tool 的 join-cluster 子命令添加 Server ID 参数。
- a6cb8215a8 修复 QoS 内存泄露问题。
- d4d76ddb2e ovsdb-tool: 添加 fix-cluster 命令。
- ffd2328d4a netdev: 减少获取设备地址的 CPU 利用率。
- d088c5d8c2 ovs-router: 跳过获取 kube-ipvs0 的源地址。
- 1b31f07dc6 增加大规模集群的默认探测间隔。
- 54b7678229 更新 ovs-sandbox 以支持 docker run。
- 9ee66bd91b 修改源路由优先级。
- e889d46924 修复 Underlay 模式下达到 resubmit 限制的问题。
- f9e97031b5 ovn-controller: 不为 Kube-OVN 端口在 localnet 上发送 GARP。
- 78cade0187 添加有条件跳过 conntrack 的支持。
- 85aa6263ad northd: 访问节点本地 DNS IP 时跳过 conntrack。
- 34dc3e3fcf lflow: 不将 lport 之间的直接流量发送到 conntrack。
- a297b840c2 在逻辑交换机入口管道中将 DNAT 数据包直接输出到 lsp。
- 03e35ed9c5 ovn-controller: 使激活策略在单机箱环境下工作。
- e7d3ba53cd 设置 ACL 时跳过节点本地 DNS IP 的 conntrack。
- 9286e1fd57 优先选择本地后端。
- e5916eb53a 修复具有多个分布式网关端口的 lr-lb DNAT 问题。
- e4e6ea9c5f 支持专用的 BFD LRP。
- e76880e792 northd: 添加 nb 选项 version_compatibility。
- 477695a010 northd: 跳过从 localnet 端口对 lrp 地址的 arp/nd 请求。
- 2209d9a335 修复 ovn-controller 在大规模 LSP 场景下的处理。
- 52b727b331 优化数据结构性能，降低 ovn-northd 增量计算耗时。
- 53b9837f7f 修复 LB 与 ACL 联动场景下的转发问题。
- 737d9f932e 修复多 GW 场景下的会话粘滞问题。
- sbrec-delete-null-check 修复 ovn-controller 在 sbrec 删除场景下空指针访问。

已合入上游修改：

- 20626ea909 组播流量绕过 LB 和 ACL 处理阶段，以提高特定数据链路性能。
- a2d9ff3ccd Deb 构建增加编译优化选项。

PDF Slack Support

2025 Kube-OVN Team

2026⬚4⬚28⬚

2022⬚5⬚24⬚

GitHub 

8.8.1 ⬚⬚

8.9

Kube-OVN OVN/OVS `Geneve` `Vxlan` `STT`   OVN OVN Architecture Design Decision

8.9.1 Geneve

`Geneve` Kube-OVN OVN   Offload `Geneve` 24bit datapath datapath 32768

 Mellanox OVS `Geneve` 5.4   backport

 UDP TCP over UDP TCP CPU

8.9.2 Vxlan

`Vxlan` OVN Offload   OVN datapath 4096 datapath datapath 4096 `inport` ACL

 Mellanox OVS `Vxlan`

 UDP TCP over UDP TCP CPU

8.9.3 STT

> **Warning**
>
> OpenVswitch 3.6 STT Tunnel

`STT` OVN TCP TCP TCP OVN datapath

 OVS

 OVS

8.9.4

- VXLAN vs GENEVE: Understand The Difference
- OVN FAQ
- What is Geneve

PDF | Slack | Support

20259月10日

2022年6月17日

GitHub

8.9.5 ▯▯

8.10 Kube-OVN □□□□

□□□□□ Kube-OVN □□□□□□□□□□

8.10.1 ovn-monitor

OVN □□□□□□□□□□

□□	□□□	□□
Gauge	kube_ovn_ovn_status	OVN □□□□□ (2) □ follower□ (1) □ leader, (0) □□□□□□
Gauge	kube_ovn_failed_req_count	OVN □□□□□□□
Gauge	kube_ovn_log_file_size_bytes	OVN □□□□□□□□□
Gauge	kube_ovn_db_file_size_bytes	OVN □□□□□□□□□□
Gauge	kube_ovn_chassis_info	OVN chassis □□ (1) □□□□(0) □□□
Gauge	kube_ovn_db_status	OVN □□□□□, (1) □□□□ (0) □□□□
Gauge	kube_ovn_logical_switch_info	OVN logical switch □□□□□ (1)□□□□□□ logical switch □□□
Gauge	kube_ovn_logical_switch_external_id	OVN logical switch external_id □□□□□ (1)□□□□□□ external-id □□□
Gauge	kube_ovn_logical_switch_port_binding	OVN logical switch □ logical switch port □□□□□□□ (1)□□□□□□□□□□
Gauge	kube_ovn_logical_switch_tunnel_key	□ OVN logical switch □□□ tunnel key □□□
Gauge	kube_ovn_logical_switch_ports_num	OVN logical switch □ logical port □□□□
Gauge	kube_ovn_logical_switch_port_info	OVN logical switch port □□□□□ (1)□□□□□□□□□□□
Gauge	kube_ovn_logical_switch_port_tunnel_key	□ OVN logical switch port □□□ tunnel key □□□
Gauge	kube_ovn_cluster_enabled	(1) OVN □□□□□□□□□ (0) OVN □□□□□□□□□□
Gauge	kube_ovn_cluster_role	□□□□□□□□□□□□□ (1)□□□□□□□□□□□□□□□
Gauge	kube_ovn_cluster_status	□□□□□□□□□□□□□ (1)□□□□□□□□□□□□□□□
Gauge	kube_ovn_cluster_term	RAFT term □□□
Gauge	kube_ovn_cluster_leader_self	□□□□□□□□□□ leader (1) □□ (0) □□□
Gauge	kube_ovn_cluster_vote_self	□□□□□□□□□□□□□□□ leader (1) □□ (0) □□□
Gauge	kube_ovn_cluster_election_timer	□□ election timer □□
Gauge	kube_ovn_cluster_log_not_committed	□ commit □ RAFT □□□□□
Gauge	kube_ovn_cluster_log_not_applied	□ apply □ RAFT □□□□□
Gauge	kube_ovn_cluster_log_index_start	□□ RAFT □□□□□□□□□
Gauge	kube_ovn_cluster_log_index_next	RAFT □□□□□□□□□□
Gauge	kube_ovn_cluster_inbound_connections_total	□□□□□□□□□□□□
Gauge	kube_ovn_cluster_outbound_connections_total	□□□□□□□□□□□□
Gauge	kube_ovn_cluster_inbound_connections_error_total	□□□□□□□□□□□□□□□
Gauge	kube_ovn_cluster_outbound_connections_error_total	□□□□□□□□□□□□□□□

2025 Kube-OVN Team

8.10.2 ovs-monitor

`ovsdb` 和 `vswitchd` 的运行状态

类型	指标项	描述
Gauge	ovs_status	OVS 健康状态，(1) 为正常，(0) 为异常。
Gauge	ovs_info	OVS 基础信息，值为 (1)，标签中包含对应信息。
Gauge	failed_req_count	OVS 失败请求数量。
Gauge	log_file_size_bytes	OVS 组件日志文件大小（字节）。
Gauge	db_file_size_bytes	OVS 组件数据库文件大小（字节）。
Gauge	datapath	Datapath 基础信息，值为 (1)，标签中包含对应信息。
Gauge	dp_total	当前 OVS 中 datapath 数量。
Gauge	dp_if	Datapath 接口基础信息，值为 (1)，标签中包含对应信息。
Gauge	dp_if_total	当前 datapath 中 port 数量。
Gauge	dp_flows_total	Datapath 中 flow 数量。
Gauge	dp_flows_lookup_hit	Datapath 中命中当前 flow 数据包数量。
Gauge	dp_flows_lookup_missed	Datapath 中未命中当前 flow 数据包数量。
Gauge	dp_flows_lookup_lost	Datapath 中需要发送给 userspace 处理的数据包数量。
Gauge	dp_masks_hit	Datapath 中命中当前 mask 数据包数量。
Gauge	dp_masks_total	Datapath 中 mask 的数量。
Gauge	dp_masks_hit_ratio	Datapath 中数据包命中 mask 的比率。
Gauge	interface	OVS 接口基础信息，值为 (1)，标签中包含对应信息。
Gauge	interface_admin_state	接口管理状态信息 (0) 为 down, (1) 为 up, (2) 为其他状态。
Gauge	interface_link_state	接口链路状态信息 (0) 为 down, (1) 为 up, (2) 为其他状态。
Gauge	interface_mac_in_use	OVS Interface 使用的 MAC 地址
Gauge	interface_mtu	OVS Interface 使用的 MTU。
Gauge	interface_of_port	OVS Interface 关联的 OpenFlow Port ID。
Gauge	interface_if_index	OVS Interface 关联的 Index。
Gauge	interface_tx_packets	OVS Interface 发送包数量。
Gauge	interface_tx_bytes	OVS Interface 发送包大小。
Gauge	interface_rx_packets	OVS Interface 接收包数量。
Gauge	interface_rx_bytes	OVS Interface 接收包大小。
Gauge	interface_rx_crc_err	OVS Interface 接收包校验和错误数量。
Gauge	interface_rx_dropped	OVS Interface 接收包丢弃数量。
Gauge	interface_rx_errors	OVS Interface 接收包错误数量。
Gauge	interface_rx_frame_err	OVS Interface 接收帧错误数量。
Gauge	interface_rx_missed_err	OVS Interface 接收包 miss 数量。
Gauge	interface_rx_over_err	OVS Interface 接收包 overrun 数量。
Gauge	interface_tx_dropped	OVS Interface 发送包丢弃数量。

□□	□□□	□□
Gauge	interface_tx_errors	OVS Interface □□□□□□□□
Gauge	interface_collisions	OVS interface □□□□□

8.10.3 kube-ovn-pinger

□□□□□□□□□□□□

□□	□□□	□□
Gauge	pinger_ovs_up	□□ OVS □□□
Gauge	pinger_ovs_down	□□ OVS □□□
Gauge	pinger_ovn_controller_up	□□ ovn-controller □□□
Gauge	pinger_ovn_controller_down	□□ ovn-controller □□□
Gauge	pinger_inconsistent_port_binding	OVN-SB □ portbinding □□□□□ OVS interface □□□□□□□
Gauge	pinger_apiserver_healthy	kube-ovn-pinger □□□□ apiserver□
Gauge	pinger_apiserver_unhealthy	kube-ovn-pinger □□□□ apiserver□
Histogram	pinger_apiserver_latency_ms	kube-ovn-pinger □□ apiserver □□□
Gauge	pinger_internal_dns_healthy	kube-ovn-pinger □□□□□□□□□
Gauge	pinger_internal_dns_unhealthy	kube-ovn-pinger □□□□□□□□□
Histogram	pinger_internal_dns_latency_ms	kube-ovn-pinger □□□□□□□□□
Gauge	pinger_external_dns_healthy	kube-ovn-pinger □□□□□□□□□
Gauge	pinger_external_dns_unhealthy	kube-ovn-pinger □□□□□□□□□
Histogram	pinger_external_dns_latency_ms	kube-ovn-pinger □□□□□□□□□
Histogram	pinger_pod_ping_latency_ms	kube-ovn-pinger ping Pod □□□
Gauge	pinger_pod_ping_lost_total	kube-ovn-pinger ping Pod □□□□□
Gauge	pinger_pod_ping_count_total	kube-ovn-pinger ping Pod □□□
Histogram	pinger_node_ping_latency_ms	kube-ovn-pinger ping Node □□□
Gauge	pinger_node_ping_lost_total	kube-ovn-pinger ping Node □□□
Gauge	pinger_node_ping_count_total	kube-ovn-pinger ping Node □□□
Histogram	pinger_external_ping_latency_ms	kube-ovn-pinger ping □□□□ □□□
Gauge	pinger_external_ping_lost_total	kube-ovn-pinger ping □□□□□□□

2025 Kube-OVN Team

8.10.4 kube-ovn-controller

`kube-ovn-controller` [illegible]

[illegible]	[illegible]	[illegible]
Histogram	rest_client_request_latency_seconds	[illegible] apiserver [illegible]
Counter	rest_client_requests_total	[illegible] apiserver [illegible]
Counter	lists_total	API list [illegible]
Summary	list_duration_seconds	API list [illegible]
Summary	items_per_list	API list [illegible]
Counter	watches_total	API watch [illegible]
Counter	short_watches_total	[illegible] API watch [illegible]
Summary	watch_duration_seconds	API watch [illegible]
Summary	items_per_watch	API watch [illegible]
Gauge	last_resource_version	[illegible] resource version[illegible]
Histogram	ovs_client_request_latency_milliseconds	[illegible] OVN [illegible]
Gauge	subnet_available_ip_count	[illegible] IP [illegible]
Gauge	subnet_used_ip_count	[illegible] IP [illegible]

8.10.5 kube-ovn-cni

`kube-ovn-cni` [illegible]

[illegible]	[illegible]	[illegible]
Histogram	cni_op_latency_seconds	CNI [illegible]
Counter	cni_wait_address_seconds_total	CNI [illegible]
Counter	cni_wait_connectivity_seconds_total	CNI [illegible]
Counter	cni_wait_route_seconds_total	CNI [illegible]
Histogram	rest_client_request_latency_seconds	[illegible] apiserver [illegible]
Counter	rest_client_requests_total	[illegible] apiserver [illegible]
Counter	lists_total	API list [illegible]
Summary	list_duration_seconds	API list [illegible]
Summary	items_per_list	API list [illegible]
Counter	watches_total	API watch [illegible]
Counter	short_watches_total	[illegible] API watch [illegible]
Summary	watch_duration_seconds	API watch [illegible]
Summary	items_per_watch	API watch [illegible]
Gauge	last_resource_version	[illegible] resource version[illegible]
Histogram	ovs_client_request_latency_milliseconds	[illegible] OVN [illegible]

2025 Kube-OVN Team

PDF Slack Support

2026⬚4⬚28⬚

2022⬚6⬚21⬚

GitHub 

8.10.6 ⬚⬚

2025 Kube-OVN Team

8.11 Kube-OVN □□□□

基于 Kube-OVN 最新版本，整理了 Kube-OVN 支持的 CRD 资源列表，列出 CRD 定义各字段的取值类型和含义，以供参考。

8.11.1 □□□ Condition □□

属性名称	类型	描述
type	String	状态类型
status	String	状态值，取值为 `True`，`False` 或 `Unknown`
reason	String	状态变化的原因
message	String	状态变化的具体信息
observedGeneration	Int64	观察到的资源版本
lastUpdateTime	Time	上次状态更新时间
lastTransitionTime	Time	上次状态类型发生变化的时间

在各 CRD 的定义中，Status 中的 Condition 字段，都遵循上述格式，因此提前进行说明。

8.11.2 □□□□□□

Subnet 定义

SUBNET

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均[`Subnet`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	SubnetSpec	Subnet 具体配置信息字段
status	SubnetStatus	Subnet 状态信息字段

SubnetSpec

属性名称	类型	描述
default	Bool	该子网是否为默认子网
vpc	String	子网所属 VPC，默认为 ovn-cluster
protocol	String	IP 协议，取值可以为 `IPv4`、`IPv6`、`Dual` 或仅分配 MAC 的 Underlay 子网使用的 `Mac`
namespaces	[]String	该子网所绑定的 namespace 列表
cidrBlock	String	子网的网段范围，如 10.16.0.0/16
gateway	String	子网网关地址，默认为该子网 CIDRBlock 下的第一个可用地址
excludeIps	[]String	该子网下不会被自动分配的地址范围
provider	String	默认为 OVN。多网卡情况下可以配置取值为 NetworkAttachmentDefinition [illegible]，Kube-OVN 将会使用这些信息找到对应的 Subnet 资源
gatewayType	String	Overlay 模式下的网关类型，取值可以[illegible] `distributed` 或 `centralized`
gatewayNode	String	当网关模式为 centralized 时的网关节点，可以为逗号分隔的多个节点
gatewayNodeSelectors	[]LabelSelector	网关节点选择器
natOutgoing	Bool	出网流量是否进行 NAT。该参数[illegible] `externalEgressGateway` 参数不能同时设置。
externalEgressGateway	String	外部网关地址。需要和子网网关节点在同一个二层可达域，该参[illegible] `natOutgoing` 参数不能同时设置
policyRoutingPriority	Uint32	策略路由优先级。添加策略路由使用参数，控制流量经子网网关之后，转发到外部网关地址
policyRoutingTableID	Uint32	使用的本地策略路由表的 TableID，每个子网均需不同以避免冲突
mtu	Uint32	子网的 MTU 大小
private	Bool	标识该子网是否为私有子网，私有子网默认拒绝子网外的地址访问
routed	Bool	是否启用路由子网模式。启用后 Pod 使[illegible] `/32`（IPv4）或 `/128`（IPv6）地址，并通过子网网关访问；需要 OVN 逻辑路由器端口
allowSubnets	[]String	子网为私有子网的情况下，允许访问该子网地址的集合
vlan	String	子网绑定的 Vlan 名称
vips	[]String	子网下 virtual 类型 lsp 的 virtual-ip 参数信息
logicalGateway	Bool	是否启用逻辑网关
disableGatewayCheck	Bool	创建 Pod 时是否跳过网关联通性检查
disableInterConnection	Bool	控制是否开启子网跨集群互联
enableDHCP	Bool	控制是否配置子网下 lsp 的 dhcp 配置选项
dhcpV4Options	String	子网下 lsp dhcpv4_options 关联的 DHCP_Options 记录
dhcpV6Options	String	子网下 lsp dhcpv6_options 关联的 DHCP_Options 记录
enableIPv6RA	Bool	控制子网连接路由器的 lrp 端口，是否配置 ipv6_ra_configs 参数
ipv6RAConfigs	String	子网连接路由器的 lrp 端口，ipv6_ra_configs 参数配置信息
acls	[]Acl	子网对应 logical-switch 关联的 acls 记录
allowEWTraffic	Bool	是否允许东西向流量
natOutgoingPolicyRules	[]NatOutgoingPolicyRule	NAT 出网策略规则
u2oInterconnectionIP	String	Underlay/Overlay 互联使用的 IP 地址
u2oInterconnection	Bool	是否开启 Overlay/Underlay 的互联模式

2025 Kube-OVN Team

属性名称	类型	描述
u2oFeatures	U2OFeatures	Underlay/Overlay 互联特性配置
enableLb	*Bool	控制子网对应的 logical-switch 是否关联 load-balancer 记录
enableEcmp	Bool	集中式网关，是否开启 ECMP 路由
enableMulticastSnoop	Bool	是否启用组播侦听
enableExternalLBAddress	Bool	是否启用外部负载均衡器地址
routeTable	String	路由表名称
namespaceSelectors	[]LabelSelector	命名空间选择器
nodeNetwork	String	Underlay 节点网络名称

U2OFeatures

属性名称	类型	描述
overlayOnlyRouting	Bool	是否仅对 Overlay CIDR 使用 U2O 路由

Acl

属性名称	类型	描述
direction	String	Acl 限制方向，取值为 `from-lport` 或者 `to-lport`
priority	Int	Acl 优先级，取值范围 0 到 32767
match	String	Acl 规则匹配表达式
action	String	Acl 规则动作，取值为 `allow-related`、`allow-stateless`、`allow`、`drop`、`reject` 其中一个

NatOutgoingPolicyRule

属性名称	类型	描述
match	NatOutGoingPolicyMatch	匹配条件
action	String	动作

NatOutGoingPolicyMatch

属性名称	类型	描述
srcIPs	String	源 IP 地址范围
dstIPs	String	目标 IP 地址范围

2025 Kube-OVN Team

SubnetStatus

[illegible]	[illegible]	[illegible]
conditions	[]SubnetCondition	[illegible] Condition [illegible]
v4availableIPs	BigInt	[illegible] IPv4 IP [illegible] IPv6 [illegible] float64 [illegible]
v4availableIPrange	String	[illegible] IPv4 [illegible]
v4usingIPs	BigInt	[illegible] IPv4 IP [illegible]
v4usingIPrange	String	[illegible] IPv4 [illegible]
v6availableIPs	BigInt	[illegible] IPv6 IP [illegible]
v6availableIPrange	String	[illegible] IPv6 [illegible]
v6usingIPs	BigInt	[illegible] IPv6 IP [illegible]
v6usingIPrange	String	[illegible] IPv6 [illegible]
activateGateway	String	[illegible]
dhcpV4OptionsUUID	String	[illegible] lsp dhcpv4_options [illegible] DHCP_Options [illegible]
dhcpV6OptionsUUID	String	[illegible] lsp dhcpv6_options [illegible] DHCP_Options [illegible]
u2oInterconnectionIP	String	[illegible] Overlay/Underlay [illegible] IP [illegible]
u2oInterconnectionMAC	String	[illegible] Overlay/Underlay [illegible] MAC [illegible]
u2oInterconnectionVPC	String	[illegible] Overlay/Underlay [illegible] VPC
natOutgoingPolicyRules	[]NatOutgoingPolicyRuleStatus	NAT [illegible]
gatewayMAC	String	[illegible] MAC [illegible]
mcastQuerierIP	String	[illegible] IP [illegible]
mcastQuerierMAC	String	[illegible] MAC [illegible]

IP [illegible]

IP

[illegible]	[illegible]	[illegible]
apiVersion	String	[illegible] Kubernetes [illegible] kubeovn.io/v1
kind	String	[illegible] Kubernetes [illegible] `IP`
metadata	ObjectMeta	[illegible] Kubernetes [illegible]
spec	IPSpec	IP [illegible]

2025 Kube-OVN Team

IPSpec

[illegible]	[illegible]	[illegible]
podName	String	[illegible] Pod [illegible]
namespace	String	[illegible] Pod [illegible] Namespace [illegible]
subnet	String	IP [illegible] Subnet
attachSubnets	[]String	[illegible] IP [illegible]
nodeName	String	[illegible] Pod [illegible]
ipAddress	String	IP [illegible] `v4IP` `v6IP` [illegible]
v4IpAddress	String	IPv4 IP [illegible]
v6IpAddress	String	IPv6 IP [illegible]
attachIps	[]String	[illegible] IP [illegible] IP [illegible]
macAddress	String	[illegible] Pod [illegible] MAC [illegible]
attachMacs	[]String	[illegible] IP [illegible] MAC [illegible]
containerID	String	[illegible] Pod [illegible] Container ID
podType	String	[illegible] Pod[illegible] `StatefulSet` [illegible] `VirtualMachine` [illegible]

Vpc [illegible]

VPC

[illegible]	[illegible]	[illegible]
apiVersion	String	[illegible] Kubernetes [illegible] kubeovn.io/v1
kind	String	[illegible] Kubernetes [illegible]`Vpc`
metadata	ObjectMeta	[illegible] Kubernetes [illegible]
spec	VpcSpec	Vpc [illegible]
status	VpcStatus	Vpc [illegible]

VpcSpec

[illegible]	[illegible]	[illegible]
defaultSubnet	String	[illegible]
namespaces	[]String	Vpc [illegible]
staticRoutes	[]StaticRoute	[illegible]
policyRoutes	[]PolicyRoute	[illegible]
vpcPeerings	[]VpcPeering	VPC [illegible]
enableExternal	Bool	[illegible]
extraExternalSubnets	[]String	[illegible]
enableBfd	Bool	[illegible] BFD ([illegible])
bfdPort	BFDPort	BFD [illegible]

2025 Kube-OVN Team

StaticRoute

[illegible]	[illegible]	[illegible]
policy	String	[illegible]
cidr	String	[illegible]
nextHopIP	String	[illegible] IP [illegible]
ecmpMode	String	ECMP [illegible]
bfdId	String	BFD ID
routeTable	String	[illegible]

PolicyRoute

[illegible]	[illegible]	[illegible]
priority	Int	[illegible]
match	String	[illegible]
action	String	[illegible] `allow` [illegible] `drop` [illegible] `reroute`
nextHopIP	String	[illegible] IP [illegible] action [illegible] reroute [illegible]

VpcPeering

[illegible]	[illegible]	[illegible]
remoteVpc	String	[illegible] VPC [illegible]
localConnectIP	String	[illegible] IP [illegible] CIDR[illegible] `169.254.0.1/30` [illegible] `169.254.0.1/30,fd00::1/64` [illegible] CIDR [illegible]

BFDPort

[illegible]	[illegible]	[illegible]
enabled	Bool	[illegible] BFD
ip	String	BFD [illegible] IP [illegible]
nodeSelector	LabelSelector	[illegible] BFD LRP [illegible]

BFDPortStatus

[illegible]	[illegible]	[illegible]
name	String	BFD [illegible]
ip	String	BFD [illegible] IP [illegible]
nodes	[]String	BFD [illegible]

2025 Kube-OVN Team

VpcStatus

属性名称	类型	描述
conditions	[]VpcCondition	Vpc 状态变化信息，具体字段参考文档开头 Condition 定义
standby	Bool	是否为备用 VPC
default	Bool	是否为默认 VPC
defaultLogicalSwitch	String	默认逻辑交换机名称
router	String	关联的路由器名称
tcpLoadBalancer	String	TCP 负载均衡器名称
udpLoadBalancer	String	UDP 负载均衡器名称
sctpLoadBalancer	String	SCTP 负载均衡器名称
tcpSessionLoadBalancer	String	TCP 会话负载均衡器名称
udpSessionLoadBalancer	String	UDP 会话负载均衡器名称
sctpSessionLoadBalancer	String	SCTP 会话负载均衡器名称
subnets	[]String	VPC 下的子网列表
vpcPeerings	[]String	VPC 对等连接列表
enableExternal	Bool	是否启用外部连接
extraExternalSubnets	[]String	额外的外部子网
enableBfd	Bool	是否启用 BFD
bfdPort	BFDPortStatus	BFD 端口状态

8.11.3 Underlay 网络管理

Vlan

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均为 `Vlan`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	VlanSpec	Vlan 具体配置信息字段
status	VlanStatus	Vlan 状态信息字段

VLANSPEC

属性名称	类型	描述
vlanId	Int	已废弃，请使用 id
providerInterfaceName	String	已废弃，请使用 provider
id	Int	Vlan tag 号，取值范围为 0~4095，创建后不可修改
provider	String	Vlan 绑定的 ProviderNetwork 名称，创建后不可修改

2025 Kube-OVN Team

VLANSTATUS

■■■■	■■	■■
subnets	[]String	Vlan ■■■■■■■■
conflict	Bool	■■■■■■■
conditions	[]VlanCondition	Vlan ■■■■■■■■■■■■■■■■■■■ Condition ■■

ProviderNetwork

■■■■	■■	■■
apiVersion	String	■■ Kubernetes ■■■■■■■■■■■■■■■■■■■■■ kubeovn.io/v1
kind	String	■■ Kubernetes ■■■■■■■■■■■■■■■■■■■■ `ProviderNetwork`
metadata	ObjectMeta	■■ Kubernetes ■■■■■■■■
spec	ProviderNetworkSpec	ProviderNetwork ■■■■■■■■■
status	ProviderNetworkStatus	ProviderNetwork ■■■■■■■

PROVIDERNETWORKSPEC

■■■■	■■	■■
defaultInterface	String	■■■■■■■■■■■■■■■■■■
customInterfaces	[]CustomInterface	■■■■■■■■■■■■■■■■
nodeSelector	LabelSelector	■■■■■■■■■■■■■ OVS ■■■■■■■■■ matchLabels ■ matchExpressions■■■■■ nodeSelector■ excludeNodes ■■■■■■■
excludeNodes	[]String	■■■■■■■■■■■■■■■■
exchangeLinkName	Bool	■■■■■■■■■■■■ OVS ■■■■■
autoCreateVlanSubinterfaces	Bool	■■■■■■■ VLAN ■■■■
preserveVlanInterfaces	Bool	■■■■■■■■■■■■ VLAN ■■
vlanInterfaces	[]String	■■■■■ VLAN ■■■■■■■■■ eth0.10, bond0.20■

CustomInterface

■■■■	■■	■■
interface	String	Underlay ■■■■■■■■■
nodes	[]String	■■■■■■■■■■■■■■■■

PROVIDERNETWORKSTATUS

■■■■	■■	■■
ready	Bool	■■■■■■■■■■■■■■■■
readyNodes	[]String	■■■■■■■■■■■■■■■■■■
notReadyNodes	[]String	■■■■■■■■■■■■■■■■■■■■
vlans	[]String	■■■■■■■■ Vlan ■■
conditions	[]ProviderNetworkCondition	ProviderNetwork ■■■■■■■■■■■■■■■■■■■■ Condition ■■

2025 Kube-OVN Team

8.11.4 ...

SecurityGroup

...	...	...
apiVersion	String	... Kubernetes ... kubeovn.io/v1
kind	String	... Kubernetes ... `SecurityGroup`
metadata	ObjectMeta	... Kubernetes ...
spec	SecurityGroupSpec	SecurityGroup ...
status	SecurityGroupStatus	SecurityGroup ...

SECURITYGROUPSPEC

...	...	...
ingressRules	[]SecurityGroupRule	...
egressRules	[]SecurityGroupRule	...
allowSameGroupTraffic	Bool	...
tier	Int	ACL ...

SecurityGroupRule

...	...	...
ipVersion	String	IP ... `ipv4` ... `ipv6`
protocol	SgProtocol	... `all` ... `icmp` ... `tcp` ... `udp`
priority	Int	... 1-16384...
remoteType	SgRemoteType	... `address` ... `securityGroup`
remoteAddress	String	...
remoteSecurityGroup	String	...
portRangeMin	Int	... 1
portRangeMax	Int	... 65535
localAddress	String	... CIDR
sourcePortRangeMin	Int	... 1
sourcePortRangeMax	Int	... 65535
policy	SgPolicy	... `allow` ... `pass` ... `drop`

2025 Kube-OVN Team

SECURITYGROUPSTATUS

属性名称	类型	描述
portGroup	String	关联的端口组
allowSameGroupTraffic	Bool	是否允许同安全组内的流量
ingressMd5	String	入站规则的 MD5 值
egressMd5	String	出站规则的 MD5 值
ingressLastSyncSuccess	Bool	入站规则最后一次同步是否成功
egressLastSyncSuccess	Bool	出站规则最后一次同步是否成功

8.11.5 □□□□□□□□ IP

Vip

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均`Vip`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	VipSpec	Vip 具体配置信息字段
status	VipStatus	Vip 状态信息字段

VIPSPEC

属性名称	类型	描述
namespace	String	VIP 所属的命名空间
subnet	String	VIP 所属的子网
type	String	VIP 类型
v4ip	String	IPv4 地址
v6ip	String	IPv6 地址
macAddress	String	MAC 地址
selector	[]String	选择器
attachSubnets	[]String	附加的子网列表

VIPSTATUS

属性名称	类型	描述
conditions	[]VipCondition	VIP 状态变化信息，具体字段参考文档开头 Condition 定义
type	String	VIP 类型
v4ip	String	IPv4 地址
v6ip	String	IPv6 地址
mac	String	MAC 地址
selector	[]String	选中的 Pod 名称列表

2025 Kube-OVN Team

SwitchLBRule

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均为 `SwitchLBRule`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	SwitchLBRuleSpec	SwitchLBRule 具体配置信息字段
status	SwitchLBRuleStatus	SwitchLBRule 状态信息字段

SWITCHLBRULESPEC

属性名称	类型	描述
vip	String	虚拟 IP 地址
namespace	String	所属命名空间
selector	[]String	后端选择器
endpoints	[]String	后端端点列表
sessionAffinity	String	会话亲和性
ports	[]SwitchLBRulePort	端口配置

SwitchLBRulePort

属性名称	类型	描述
name	String	端口名称
port	Int32	端口号
targetPort	Int32	目标端口号
protocol	String	协议类型

SWITCHLBRULESTATUS

属性名称	类型	描述
conditions	[]SwitchLBRuleCondition	SwitchLBRule 状态变化信息，具体字段参考文档开头 Condition 定义
ports	String	SwitchLBRule 端口信息
service	String	SwitchLBRule 提供服务的 service 名称

RouterLBRule

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均为 `RouterLBRule`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	RouterLBRuleSpec	RouterLBRule 具体配置信息字段
status	RouterLBRuleStatus	RouterLBRule 状态信息字段

2025 Kube-OVN Team

ROUTERLBRULESPEC

字段名	类型	说明
ovnEip	String	关联 OVN EIP 名称
vpc	String	所属 VPC
namespace	String	命名空间
selector	[]String	选择器
endpoints	[]String	端点列表
sessionAffinity	String	会话亲和
ports	[]RouterLBRulePort	端口列表

RouterLBRulePort

字段名	类型	说明
name	String	端口名
port	Int32	端口号
targetPort	Int32	目标端口
protocol	String	协议类型

ROUTERLBRULESTATUS

字段名	类型	说明
conditions	[]RouterLBRuleCondition	RouterLBRule 当前状态的 Condition 列表
ports	String	RouterLBRule 端口列表
service	String	RouterLBRule 关联的 service 名称

8.11.6 QoS 和 IP 管理

QoSPolicy

字段名	类型	说明
apiVersion	String	遵循 Kubernetes 的 API 版本规范 kubeovn.io/v1
kind	String	遵循 Kubernetes 的资源类型规范，`QoSPolicy`
metadata	ObjectMeta	遵循 Kubernetes 的元数据规范
spec	QoSPolicySpec	QoSPolicy 的详细规格

QOSPOLICYSPEC

字段名	类型	说明
bandwidthLimitRules	QoSPolicyBandwidthLimitRules	带宽限速规则
shared	Bool	是否共享
bindingType	QoSPolicyBindingType	绑定类型 `EIP` 或 `NATGW`

2025 Kube-OVN Team

QoSPolicyBandwidthLimitRule

字段名	类型	说明
name	String	规则名称
interface	String	网卡名称
rateMax	String	最大带宽Mbps，支持 `100` 或 `0.5` 这样500Kbps的
burstMax	String	最大突发带宽MB，支持 `10` 或 `0.5` 这样500KB的
priority	Int	规则优先级
direction	QoSPolicyRuleDirection	流量方向，可选 `ingress` 或 `egress`
matchType	QoSPolicyRuleMatchType	匹配类型，可选 `ip`
matchValue	String	匹配内容

QOSPOLICYSTATUS

字段名	类型	说明
bandwidthLimitRules	QoSPolicyBandwidthLimitRules	带宽限制规则列表
shared	Bool	是否共享
bindingType	QoSPolicyBindingType	绑定类型
conditions	[]Condition	QoSPolicy 状态条件

IPPool

字段名	类型	说明
apiVersion	String	遵循 Kubernetes 约定，可选 kubeovn.io/v1
kind	String	遵循 Kubernetes 约定，可选为 `IPPool`
metadata	ObjectMeta	遵循 Kubernetes 约定
spec	IPPoolSpec	IPPool 期望状态
status	IPPoolStatus	IPPool 实际状态

IPPOOLSPEC

字段名	类型	说明
subnet	String	子网名称
namespaces	[]String	该 IPPool 可被哪些命名空间使用，默认为空表示不限制 IPPool 被 Pod
ips	[]String	IP 或者 CIDR 地址，例如 ipCount 等
ipCount	Int32	需要分配多少个连续的 IPv4 或 IPv6 地址，范围 1-32，例如 ips 等
enableAddressSet	Bool	是否为 AddressSet 创建对应 ACL

IPPOOLSTATUS

属性名称	类型	描述
v4AvailableIPs	BigInt	IPv4 可用 IP 地址数量
v4AvailableIPRange	String	IPv4 可用 IP 地址范围
v4UsingIPs	BigInt	IPv4 已使用 IP 地址数量
v4UsingIPRange	String	IPv4 已使用 IP 地址范围
v6AvailableIPs	BigInt	IPv6 可用 IP 地址数量
v6AvailableIPRange	String	IPv6 可用 IP 地址范围
v6UsingIPs	BigInt	IPv6 已使用 IP 地址数量
v6UsingIPRange	String	IPv6 已使用 IP 地址范围
ips	[]String	池中当前包含的 IP 地址，按分配顺序排列；由控制器维护
conditions	[]IPPoolCondition	IP 池状态变化信息，具体字段参考文档开头 Condition 定义

8.11.7 NAT □□□ IP □□

IptablesEIP

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均[`IptablesEIP`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	IptablesEIPSpec	IptablesEIP 具体配置信息字段
status	IptablesEIPStatus	IptablesEIP 状态信息字段

IPTABLESEIPSPEC

属性名称	类型	描述
v4ip	String	IPv4 地址
v6ip	String	IPv6 地址
macAddress	String	MAC 地址
natGwDp	String	NAT 网关数据路径
qosPolicy	String	QoS 策略
externalSubnet	String	外部子网
namespace	String	NAT 网关工作负载（Deployment 或 StatefulSet）及 Pod 所在的 Namespace；为空时从引用的 VpcNatGateway 或 controller 所在的 Namespace 推导；显式设置时必须与网关的有效 Namespace 一致

2025 Kube-OVN Team

IPTABLESEIPSTATUS

...	...	...
ready	Bool	IptablesEIP ...
ip	String	IptablesEIP ... IP ... IPv4 ...
redo	String	IptablesEIP CRD ...
nat	String	IptablesEIP ... `fip` ... `snat` ... `dnat`
qosPolicy	String	QoS ...
conditions	[]IptablesEIPCondition	IptablesEIP ... Condition ...

OvnEip

...	...	...
apiVersion	String	... Kubernetes ... kubeovn.io/v1
kind	String	... Kubernetes ... `OvnEip`
metadata	ObjectMeta	... Kubernetes ...
spec	OvnEipSpec	OvnEip ...
status	OvnEipStatus	OvnEip ...

OVNEIPSPEC

...	...	...
externalSubnet	String	...
v4Ip	String	IPv4 ...
v6Ip	String	IPv6 ...
macAddress	String	MAC ...
type	String	... lrp、lsp ... nat

OVNEIPSTATUS

...	...	...
conditions	[]OvnEipCondition	OvnEip ...
type	String	OVN EIP ...
nat	String	NAT ...
ready	Bool	EIP ...
v4Ip	String	IPv4 ...
v6Ip	String	IPv6 ...
macAddress	String	MAC ...

IptablesFIPRule

...	...	...
apiVersion	String	... Kubernetes ... kubeovn.io/v1
kind	String	... Kubernetes ... `IptablesFIPRule`
metadata	ObjectMeta	... Kubernetes ...
spec	IptablesFIPRuleSpec	IptablesFIPRule ...

IPTABLESFIPRULESPEC

...	...	...
eip	String	... IP ...
internalIP	String	... IP ...

IPTABLESFIPRULESTATUS

...	...	...
conditions	[]IptablesFIPRuleCondition	IptablesFIPRule ...
ready	Bool	FIP ...
v4ip	String	IPv4 EIP ...
v6ip	String	IPv6 EIP ...
natGwDp	String	NAT ...
redo	String	IptablesFIPRule CRD ...
internalIP	String	... IP ...

OvnFip

...	...	...
apiVersion	String	... Kubernetes ... kubeovn.io/v1
kind	String	... Kubernetes ... `OvnFip`
metadata	ObjectMeta	... Kubernetes ...
spec	OvnFipSpec	OvnFip ...
status	OvnFipStatus	OvnFip ...

2025 Kube-OVN Team

OVNFIPSPEC

□□□□	□□	□□
ovnEip	String	□□□ OVN EIP
ipType	String	IP □□□□□□ vip □ ip
ipName	String	IP □□
vpc	String	□□ VPC
v4Ip	String	IPv4 □□
v6Ip	String	IPv6 □□
type	String	□□□□□□ distributed □ centralized

OVNFIPSTATUS

□□□□	□□	□□
conditions	[]OvnFipCondition	OvnFip □□□□□□
ready	Bool	FIP □□□□□□
vpc	String	□□ VPC
v4Eip	String	IPv4 EIP □□
v6Eip	String	IPv6 EIP □□
v4Ip	String	IPv4 □□
v6Ip	String	IPv6 □□
macAddress	String	MAC □□

IptablesDnatRule

□□□□	□□	□□
apiVersion	String	□□ Kubernetes □□□□□□□□□□□□□□□□□□□□ kubeovn.io/v1
kind	String	□□ Kubernetes □□□□□□□□□□□□□□□□□□ `IptablesDnatRule`
metadata	ObjectMeta	□□ Kubernetes □□□□□□□□
spec	IptablesDnatRuleSpec	IptablesDnatRule □□□□□□□□

2025 Kube-OVN Team

IPTABLESDNATRULESPEC

属性名称	类型	描述
eip	String	弹性 IP 地址；当 clusterIP 设置时可选
clusterIP	String	Kubernetes Service ClusterIP，用于 VPC 内流量访问
externalPort	String	外部端口
protocol	String	协议类型
internalIP	String	内部 IP 地址
internalPort	String	内部端口
type	String	DNAT 规则类型，取值为 `exclusive`（独占，默认）或 `share`（共享负载均衡）
sessionAffinity	String	会话亲和性，仅 share 类型支持，取值`""`（无）或 `ClientIP`
sessionAffinityTimeoutSeconds	Int32	ClientIP 会话亲和性超时秒数，默认 10800（3 小时）

IPTABLESDNATRULESTATUS

属性名称	类型	描述
conditions	[]IptablesDnatRuleCondition	IptablesDnatRule 状态变化信息
ready	Bool	DNAT 规则是否配置完成
v4ip	String	IPv4 EIP 地址
v6ip	String	IPv6 EIP 地址
natGwDp	String	NAT 网关数据路径
redo	String	IptablesDnatRule CRD 重建状态
protocol	String	协议类型
internalIP	String	内部 IP 地址
internalPort	String	内部端口
externalPort	String	外部端口

OvnDnatRule

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均`OvnDnatRule`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	OvnDnatRuleSpec	OvnDnatRule 具体配置信息字段
status	OvnDnatRuleStatus	OvnDnatRule 状态信息字段

OVNDNATRULESPEC

□□□□	□□	□□
ovnEip	String	□□□ OVN EIP
ipType	String	IP □□□□□□ vip □ ip
ipName	String	IP □□
internalPort	String	□□□□
externalPort	String	□□□□
protocol	String	□□□□
vpc	String	□□ VPC
v4Ip	String	IPv4 □□
v6Ip	String	IPv6 □□

OVNDNATRULESTATUS

□□□□	□□	□□
vpc	String	□□ VPC
v4Eip	String	IPv4 EIP □□
v6Eip	String	IPv6 EIP □□
externalPort	String	□□□□
v4Ip	String	IPv4 □□
v6Ip	String	IPv6 □□
internalPort	String	□□□□
protocol	String	□□□□
ipName	String	IP □□
ready	Bool	DNAT □□□□□□□□
conditions	[]OvnDnatRuleCondition	OVN DNAT □□□□□□□□□□□□□□□□□□□□ Condition □□

IptablesSnatRule

□□□□	□□	□□
apiVersion	String	□□ Kubernetes □□□□□□□□□□□□□□□□□□□ kubeovn.io/v1
kind	String	□□ Kubernetes □□□□□□□□□□□□□□□□□□□ `IptablesSnatRule`
metadata	ObjectMeta	□□ Kubernetes □□□□□□□□
spec	IptablesSnatRuleSpec	IptablesSnatRule □□□□□□□□

IPTABLESSNATRULESPEC

□□□□	□□	□□
eip	String	□□ IP □□
internalCIDR	String	□□ CIDR □□

2025 Kube-OVN Team

IPTABLESSNATRULESTATUS

□□□□	□□	□□
conditions	[]IptablesSnatRuleCondition	IptablesSnatRule □□□□□□
ready	Bool	SNAT □□□□□□□□
v4ip	String	IPv4 EIP □□
v6ip	String	IPv6 EIP □□
natGwDp	String	NAT □□□□□□
redo	String	IptablesSnatRule CRD □□□□
internalCIDR	String	□□ CIDR □□

OvnSnatRule

□□□□	□□	□□
apiVersion	String	□□ Kubernetes □□□□□□□□□□□□□□□□□□□□ kubeovn.io/v1
kind	String	□□ Kubernetes □□□□□□□□□□□□□□□□□□□□ `OvnSnatRule`
metadata	ObjectMeta	□□ Kubernetes □□□□□□□□
spec	OvnSnatRuleSpec	OvnSnatRule □□□□□□□□
status	OvnSnatRuleStatus	OvnSnatRule □□□□□□□

OVNSNATRULESPEC

□□□□	□□	□□
ovnEip	String	□□□ OVN EIP
vpcSubnet	String	VPC □□
ipName	String	IP □□
vpc	String	□□ VPC
v4IpCidr	String	IPv4 CIDR □□
v6IpCidr	String	IPv6 CIDR □□

OVNSNATRULESTATUS

□□□□	□□	□□
conditions	[]OvnSnatRuleCondition	OvnSnatRule □□□□□□□
ready	Bool	SNAT □□□□□□□
vpc	String	□□ VPC
v4Eip	String	IPv4 EIP □□
v6Eip	String	IPv6 EIP □□
v4IpCidr	String	IPv4 CIDR □□
v6IpCidr	String	IPv6 CIDR □□

2025 Kube-OVN Team

8.11.8 VPC 网关

VpcNatGateway

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均为 `VpcNatGateway`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	VpcNatGatewaySpec	VpcNatGateway 具体配置信息字段
status	VpcNatGatewayStatus	VpcNatGateway 状态信息字段

VPCNATGATEWAYSPEC

属性名称	类型	描述
vpc	String	VPC 网关 Pod 所在的 VPC 名称
subnet	String	VPC 网关 Pod 所属的子网名称
namespace	String	VPC 网关工作负载（Deployment 或 StatefulSet）及 Pod 所在的 Namespace；为空时使用 controller 所在的 Namespace；创建后不可修改
externalSubnets	[]String	外部子网列表
lanIp	String	可选的 LAN IP。非 HA 模式下留空可动态分配，观察到的地址会持久化到此字段；HA 模式下忽略此字段。
replicas	Int32	网关副本数。未设置时默认使用 1 个副本；大于 1 时使用 Deployment 实现高可用。
selector	[]String	标准 Kubernetes Selector 匹配信息
tolerations	[]Toleration	标准 Kubernetes 容忍信息
affinity	Affinity	标准 Kubernetes 亲和性配置
qosPolicy	String	QoS 策略名称
bgpSpeaker	VpcBgpSpeaker	BGP speaker 配置
bfd	VpcNatGatewayBFDConfig	BFD 健康检测配置（HA 模式）
internalSubnets	[]String	按子网名称指定的内部流量子网，发往 0.0.0.0/0 或 ::/0 的流量将被路由到 NAT 网关实例
internalCIDRs	[]String	按 CIDR 指定的内部流量网段，发往 0.0.0.0/0 或 ::/0 的流量将被路由到 NAT 网关实例
routes	[]Route	路由配置列表
noDefaultEIP	Bool	是否禁用默认 EIP 分配
annotations	map[String]String	StatefulSet NAT 网关 Pod 模板的自定义 annotations，仅在创建时生效

2025 Kube-OVN Team

VpcBgpSpeaker

■■■■	■■	■■
enabled	Bool	■■■■ BGP speaker
asn	Uint32	■■■■■■■■
remoteAsn	Uint32	■■■■■■■■
neighbors	[]String	BGP ■■■■
holdTime	Duration	BGP ■■■■
routerId	String	BGP ■■■ ID
password	String	BGP ■■■■
enableGracefulRestart	Bool	■■■■■■■■■
extraArgs	[]String	■■■■■■■

VpcNatGatewayBFDConfig

■■■■	■■	■■
enabled	Bool	■■■■ BFD ■■■■■■■■■ VPC ■ spec.bfd.enabled ■ true
minRX	Int32	BFD ■■■■■■■■ms■■■■ 1000
minTX	Int32	BFD ■■■■■■■■ms■■■■ 1000
multiplier	Int32	BFD ■■■■■■■■ 3

Route

■■■■	■■	■■
cidr	String	■■■■ CIDR
nextHopIP	String	■■■ IP ■■

VPCNATGATEWAYSTATUS

■■■■	■■	■■
qosPolicy	String	QoS ■■■■
externalSubnets	[]String	■■■■■■■
selector	[]String	■■ Kubernetes Selector ■■■■
tolerations	[]Toleration	■■ Kubernetes ■■■■
affinity	Affinity	■■ Kubernetes ■■■■■
lanIp	String	■■■■ LAN ■■■HA ■■■■■■■■■■■■■■■■■■■
replicas	Int32	■■■ `spec.replicas` ■■■■■ `spec.replicas` ■ workload ■■■■ 1 ■■■■■■■■■■■■■■■ 0■
ready	Bool	■■ workload ■■■■■
workload	VpcNatWorkload	■■■■■■■■■■
internalSubnets	[]String	■■■■■■■■■■
internalCIDRs	[]String	■■■■■ CIDR ■■

2025 Kube-OVN Team

VpcNatWorkload

...	...	...
apiVersion	String	... API ...
kind	String	... `Deployment` ... `StatefulSet`
name	String	...
nodes	[]String	...

`VpcNatGateway` ... `spec.lanIp` ... controller ... `spec.lanIp` ... `spec.replicas` ... 1 ... `spec.lanIp` ... `status.lanIp` ...HA ... `status.replicas` ... `spec.replicas` ... `status.ready` ... workload ... `spec.replicas` ... workload ... 1 ... `status.replicas` ... 0...

VpcEgressGateway

...	...	...
apiVersion	String	... Kubernetes ... kubeovn.io/v1
kind	String	... Kubernetes ... `VpcEgressGateway`
metadata	ObjectMeta	... Kubernetes ...
spec	VpcEgressGatewaySpec	VpcEgressGateway ...
status	VpcEgressGatewayStatus	VpcEgressGateway ...

2025 Kube-OVN Team

VPCEGRESSGATEWAYSPEC

...	...	...
vpc	String	... VPC...VPC
bgpConf	String	... `BgpConf` ...
evpnConf	String	... `EvpnConf` ... EVPN ...
replicas	Int32	... 1
prefix	String	...`<prefix><veg-name>`
image	String	... kube-ovn-controller ...
internalSubnet	String	... VPC ... CNI ... provider ... `<nad-name>.<nad-namespace>.ovn` ... NetworkAttachmentDefinition ...
externalSubnet	String	...
externalNetworkMode	String	...`Attachment` ... externalSubnet... `PrimaryCNI` ... Pod ...
internalIPs	[]String	... IP ... `v4,v6` ...
externalIPs	[]String	... IP ... `v4,v6` ...
internalIPPool	String	... `IPPool` ... `internalSubnet` ... `internalIPs` ...
externalIPPool	String	... `IPPool` ... `externalSubnet` ... `externalIPs` ...
selectors	[]VpcEgressGatewaySelector	.../Pod ... CNI ...`policies`
trafficPolicy	String	...`Cluster` ... `Local` ... VPC ...
podAntiAffinity	String	Pod ...`Required` ...`Preferred` ...
bfd	VpcEgressGatewayBFDConfig	BFD ...
policies	[]VpcEgressGatewayPolicy	...
nodeSelector	[]VpcEgressGatewayNodeSelector	...
tolerations	[]Toleration	... Kubernetes ...
priorityClassName	String	... Pod ... PriorityClass ...
resources	ResourceRequirements	...
bandwidth	BandwidthLimit	.../... Mbps ... bit-rate quantity
observability	VpcEgressGatewayObservability	...conntrack ...JSON ... ServiceMonitor ...

VpcEgressGatewaySelector

...	...	...
namespaceSelector	LabelSelector	...
podSelector	LabelSelector	Pod ...

2025 Kube-OVN Team

VpcEgressGatewayBFDConfig

■■■■	■■	■■
enabled	Boolean	■■■■ BFD■■■■■ VPC ■ `.spec.bfd.enabled=true` ■■■■ false
minRX	Int32	BFD ■■■■■■■ms■■■■ 1000
minTX	Int32	BFD ■■■■■■■ms■■■■ 1000
multiplier	Int32	BFD ■■■■■■■ 3

VpcEgressGatewayPolicy

■■■■	■■	■■
snat	Boolean	■■■■■■■■■■ SNAT/MASQUERADE■■■ false
ipBlocks	[]String	■■■ CIDR ■■
subnets	[]String	■■■■■■■■■■

VpcEgressGatewayNodeSelector

■■■■	■■	■■
matchLabels	map[String]String	■■■■■■■
matchExpressions	[]NodeSelectorRequirement	■■■■■■■■
matchFields	[]NodeSelectorRequirement	■■■■■■■■

BandwidthLimit

■■■■	■■	■■
ingress	Int64 ■ String	■■■■■■■■ VPC ■■■■■■■■■■■■■■■■■■■■■■ Mbps■quantity ■ `M` ■ `Mi` ■ `G` ■ `Gi` bit/s
egress	Int64 ■ String	■ VPC ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ Mbps■quantity ■ `M` ■ `Mi` ■ `G` ■ `Gi` bit/s

VpcEgressGatewayObservability

■■ Kubernetes 1.29 ■■■■■■■■■■■■■■■■■■■■■■■■ schema ■■■■■■■■■■ VPC Egress Gateway ■■■■■■

■■■■	■■	■■
resources	ResourceRequirements	■■■observer sidecar ■■■■■■ request ■ `20m` CPU ■ `64Mi` ■■■limit ■ `200m` CPU ■ `256Mi` ■■
interfaceMetrics	VpcEgressGatewayObservabilityFeature	■■■■■■■■■■
conntrack	VpcEgressGatewayConntrackObservability	conntrack ■■■■■■■■■
serviceMonitor	VpcEgressGatewayServiceMonitor	■■■■■■■■ ServiceMonitor ■■■■■

VpcEgressGatewayObservabilityFeature

■■■■	■■	■■
enabled	Boolean	■■■■■■■■■■■ false

2025 Kube-OVN Team

VpcEgressGatewayConntrackObservability

■■■■	■■	■■
metrics	VpcEgressGatewayObservabilityFeature	■■■ NAT conntrack ■■■■
log	VpcEgressGatewayConntrackLog	JSON Lines NAT ■■■■■

VpcEgressGatewayConntrackLog

■■■■	■■	■■
enabled	Boolean	■■■ JSON ■■■■■ observer ■■■■■■■■■ false
events	[]String	■■■■■■■■■■■■■■■ `start` ■/■ `end` ■■■■■■■■■
rateLimit	VpcEgressGatewayConntrackLogRateLimit	■■ Pod ■■■■■■■
filters	VpcEgressGatewayConntrackLogFilters	■■■include ■ exclude ■■

VpcEgressGatewayConntrackLogRateLimit

■■■■	■■	■■
recordsPerSecond	Int32	■■■■■■■■■■■■ 1 ■ 100000■■■ 100
burst	Int32	■■■■■■■■■■■■ 1 ■ 1000000■■■ 1000

VpcEgressGatewayConntrackLogFilters

■■■■	■■	■■
include	[]VpcEgressGatewayConntrackLogFilter	■■■■■ 64 ■ include ■■■■■■■■■■ NAT ■
exclude	[]VpcEgressGatewayConntrackLogFilter	■■■■■ 64 ■ exclude ■■■exclude ■■■■

VpcEgressGatewayConntrackLogFilter

■■■■■■■■■■■■■ AND ■■■■■■■■■■■■■■■■■■■■■■■■■ OR ■■■

■■■■	■■	■■
addressFamilies	[]String	■■■ `ipv4` ■/■ `ipv6`
protocols	[]String	■■■ `tcp` ■ `udp` ■ `sctp` ■ `icmp` ■ `icmpv6` ■/■ `other`
natTypes	[]String	■■■ `snat` ■ `dnat` ■/■ `snat_dnat`
original	VpcEgressGatewayConntrackTupleFilter	■■ tuple ■■
translated	VpcEgressGatewayConntrackTupleFilter	■■■ tuple ■■

VpcEgressGatewayConntrackTupleFilter

■■■■	■■	■■
sourceCIDRs	[]String	■■■■ CIDR■■■ 64 ■
destinationCIDRs	[]String	■■■■■ CIDR■■■ 64 ■
sourcePorts	[]VpcEgressGatewayPortRange	■■■■■■■■■■■■■■■■■ 64 ■
destinationPorts	[]VpcEgressGatewayPortRange	■■■■■■■■■■■■■■■■■■■ 64 ■

2025 Kube-OVN Team

VpcEgressGatewayPortRange

...	...	...
start	Int32	... 0 ... 65535
end	Int32	... `start` ... 65535

VpcEgressGatewayServiceMonitor

...	...	...
labels	map[String]String	...ServiceMonitor ... selector ...
annotations	map[String]String	...ServiceMonitor annotations

VPCEGRESSGATEWAYSTATUS

...	...	...
replicas	Int32	... scale ...
labelSelector	String	... scale ...
ready	Boolean	...
phase	String	... `Pending` ... `Processing` ... `Completed`
internalIPs	[]String	... IP ...
externalIPs	[]String	... IP ...
conditions	[]Condition	...
workload	VpcEgressWorkload	...

VpcEgressWorkload

...	...	...
apiVersion	String	... API ...
kind	String	... `Deployment`
name	String	...
nodes	[]String	...

BgpConf

...cluster-scoped... `VpcEgressGateway.spec.bgpConf` ...

...	...	...
apiVersion	String	... Kubernetes ... kubeovn.io/v1
kind	String	... Kubernetes ... `BgpConf`
metadata	ObjectMeta	... Kubernetes ...
spec	BgpConfSpec	BgpConf ...

2025 Kube-OVN Team

BGPCONFSPEC

■■■■	■■	■■
localASN	UInt32	■■ BGP AS Number
peerASN	UInt32	■■ BGP AS Number
routerId	String	■■■BGP Router ID
neighbours	[]String	BGP ■■ IP ■■
password	String	■■■TCP MD5 ■■■■
holdTime	Duration	■■■BGP Hold time
keepaliveTime	Duration	■■■BGP Keepalive ■■
connectTime	Duration	■■■BGP Connect ■■
ebgpMultiHop	Boolean	■■■■■■■ eBGP MultiHop
gracefulRestart	Boolean	■■■■■■■ BGP Graceful Restart

EvpnConf

■■■■cluster-scoped■■■■■ `VpcEgressGateway.spec.evpnConf` ■■■■■■ EVPN ■■■■■ VNI ■ Route Target■

■■■■	■■	■■
apiVersion	String	■■ Kubernetes ■■■■■■■■■■■■■■■■■■■■ kubeovn.io/v1
kind	String	■■ Kubernetes ■■■■■■■■■■■■■■■■■■■■ `EvpnConf`
metadata	ObjectMeta	■■ Kubernetes ■■■■■■■■
spec	EvpnConfSpec	EvpnConf ■■■■■■■■■

EVPNCONFSPEC

■■■■	■■	■■
vni	UInt32	EVPN VNI
routeTargets	[]String	Route Target ■■

DNSNameResolver

■■■■cluster-scoped■■■■■ `enable-dns-name-resolver` ■■■ ANP/CNP ■■■■■■■■■■■■ DNS ■■■■■■

■■■■	■■	■■
apiVersion	String	■■ Kubernetes ■■■■■■■■■■■■■■■■■■■■ kubeovn.io/v1
kind	String	■■ Kubernetes ■■■■■■■■■■■■■■■■■■■■ `DNSNameResolver`
metadata	ObjectMeta	■■ Kubernetes ■■■■■■■■
spec	DNSNameResolverSpec	DNSNameResolver ■■■■■■■■■
status	DNSNameResolverStatus	DNSNameResolver ■■■■■■■

2025 Kube-OVN Team

DNSNAMERESOLVERSPEC

属性名称	类型	描述
name	String	需要解析的 DNS 名称（普通名称或以 `*.` 开头的通配名称），创建后不可变

DNSNAMERESOLVERSTATUS

属性名称	类型	描述
resolvedNames	[]DNSNameResolverResolvedName	实际解析得到的 DNS 名称及其 IP 详情

DNSNameResolverResolvedName

属性名称	类型	描述
dnsName	String	解析得到的实际 DNS 名称
resolvedAddresses	[]DNSNameResolverResolvedAddress	解析到的 IP 和 TTL 与最近解析时间
conditions	[]Condition	状态条件，已知类型 `Degraded` 表示最近一次解析失败
resolutionFailures	Int32	连续解析失败次数，成功后清零，超过 5 次且 TTL 全部过期后会被移除

DNSNameResolverResolvedAddress

属性名称	类型	描述
ip	String	解析到的 IP 地址
ttlSeconds	Int32	TTL 秒数；`lastLookupTime + ttlSeconds` 之后失效
lastLookupTime	Time	最近一次成功解析的时间戳

VpcDns

属性名称	类型	描述
apiVersion	String	标准 Kubernetes 版本信息字段，所有自定义资源该值均为 kubeovn.io/v1
kind	String	标准 Kubernetes 资源类型字段，本资源所有实例该值均为 `VpcDns`
metadata	ObjectMeta	标准 Kubernetes 资源元数据信息
spec	VpcDNSSpec	VpcDns 具体配置信息字段
status	VpcDNSStatus	VpcDns 状态信息字段

VPCDNSSPEC

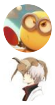
属性名称	类型	描述
replicas	Int32	副本数量
vpc	String	所属 VPC
subnet	String	所属子网

PDF Slack Support

2025 Kube-OVN Team

2026▯11▯5▯

2023▯2▯16▯



GitHub



+3

8.11.9 ▯▯

8.12 Annotation □□□□

Kube-OVN 使用了大量的 Pod 和 Node Annotation 进行功能设置和信息传递，用户可以通过本文档了解各个 Annotation 的使用，方便问题排查和信息获取。

> 注意：部分 Annotation 可能会随着代码调整而进行变化。

2025 Kube-OVN Team

8.12.1 Pod Annotation

Key	Value	Description
ovn.kubernetes.io/allocated	`true` or `false`	Pod
ovn.kubernetes.io/routed	`true` or `false`	Pod OVN
ovn.kubernetes.io/routes	String	Pod
ovn.kubernetes.io/mac_address	String	Pod Mac Pod Annotation Mac
ovn.kubernetes.io/ip_address	String	Pod IP Pod Annotation IP
ovn.kubernetes.io/ip_family	`ipv4` or `ipv6`	Pod IP
\<nadName>.\<nadNamespace>.kubernetes.io/ip_address.\<interfaceName>	String	IPinterfaceName Multus interface ovn.kubernetes.io/ip_address
\<nadName>.\<nadNamespace>.kubernetes.io/mac_address.\<interfaceName>	String	MACinterfaceName Multus interface
ovn.kubernetes.io/cidr	String	Pod CIDR
ovn.kubernetes.io/gateway	String	Pod Gateway
ovn.kubernetes.io/ip_pool	IP	Pod Workload IP
ovn.kubernetes.io/bgp	`true`, `cluster`, `local`	BGP Pod
ovn.kubernetes.io/snat	String	Pod SNAT
ovn.kubernetes.io/eip	String	Pod EIP
ovn.kubernetes.io/vip	String	v1.17 Annotation `ovn.kubernetes.io/ip_address` `ovn.kubernetes.io/ip_pool` IPPool Pod
ovn.kubernetes.io/aaps	String	Pod AAPs (Additional Allowed Addresses Pairs)
ovn.kubernetes.io/virtualmachine	String	Pod VirtualMachineInstance
ovn.kubernetes.io/activation_strategy	String	Pod
ovn.kubernetes.io/logical_router	String	Pod VPC
ovn.kubernetes.io/layer2_forward	`true` or `false`	Pod OVN LSP `unknown`
ovn.kubernetes.io/port_security	`true` or `false`	Pod Port Security
ovn.kubernetes.io/logical_switch	String	Pod
ovn.kubernetes.io/vlan_id	Int	Pod Vlan ID
ovn.kubernetes.io/ingress_rate	Int	Pod Mbits/s
ovn.kubernetes.io/egress_rate	Int	Pod Mbits/s
ovn.kubernetes.io/ingress_burst	Int	Pod burst Mbit 80% `0` burst
ovn.kubernetes.io/egress_burst	Int	Pod burst Mbit 80% `0` burst
ovn.kubernetes.io/security_groups	String	Pod Security Group
ovn.kubernetes.io/default_route	`true` or `false`	
ovn.kubernetes.io/provider_network	String	Pod ProviderNetwork
ovn.kubernetes.io/mirror	`true` or `false`	Pod

2025 Kube-OVN Team

Key	Value	Description
ovn.kubernetes.io/north_gateway	String	Pod
ovn.kubernetes.io/latency	Int	Pod ms
ovn.kubernetes.io/limit	Int	Pod qdisc
ovn.kubernetes.io/loss	Float	Pod
ovn.kubernetes.io/jitter	Int	Pod ms
ovn.kubernetes.io/generate-hash	`true` or `false`	Pod
ovn.kubernetes.io/attachmentprovider	String	Pod
ovn.kubernetes.io/vpc_nat_gw	String	VPC NAT Gateway Pod
ovn.kubernetes.io/vpc_nat_gw_init	`true` or `false`	VPC NAT Gateway Pod
ovn.kubernetes.io/vpc_nat_gw_container_restarted	`true` or `false`	VPC NAT Gateway
ovn.kubernetes.io/vpc_nat_gw_activated	`true` or `false`	VPC NAT Gateway
\<provider>.kubernetes.io/ip_pool	IP	provider IP
\<provider>.kubernetes.io/ip_family	`ipv4` or `ipv6`	provider IP
\<provider>.kubernetes.io/dhcp-v4-options	String	provider DHCPv4
\<provider>.kubernetes.io/port_security	`true` or `false`	provider
\<provider>.kubernetes.io/ingress_burst	Int	provider burst Mbit
\<provider>.kubernetes.io/egress_burst	Int	provider burst Mbit

8.12.2 Node Annotation

Key	Value	Description
ovn.kubernetes.io/allocated	`true` or `false`	`ovn0` `join`
ovn.kubernetes.io/mac_address	String	Node `ovn0` Mac
ovn.kubernetes.io/ip_address	String	Node `ovn0` IP
ovn.kubernetes.io/cidr	String	Node `ovn0` `join` CIDR
ovn.kubernetes.io/gateway	String	Node `ovn0` `join` Gateway
ovn.kubernetes.io/chassis	String	Node OVN-SouthBoundDB Chassis ID
ovn.kubernetes.io/port_name	String	Node `ovn0` OVN-NorthboundDB LSP
ovn.kubernetes.io/logical_switch	String	Node `ovn0`
ovn.kubernetes.io/tunnel_interface	String	
ovn.kubernetes.io/node_networks	String	Pod attachment networks NetworkAttachmentDefinition

8.12.3 Namespace Annotation

Key	Value	Description
ovn.kubernetes.io/cidr	CIDR	Namespace CIDR
ovn.kubernetes.io/exclude_ips	excludeIPs	Namespace excludeIPs

2025 Kube-OVN Team

8.12.4 Subnet Annotation

Key	Value	Description
ovn.kubernetes.io/bgp	`true`, `cluster`, `local`	□□□□□□ BGP □□□□□□
ovn.kubernetes.io/vpc	String	□□□□□ VPC□□□□□□□□□□ VPC □□□

8.12.5 Service Annotation

Key	Value	Description
ovn.kubernetes.io/bgp	`true`, `cluster`, `local`	□□□□□□ BGP □□ Service □□□□ ClusterIP Service □□□□□□ Pod/Subnet □□□
ovn.kubernetes.io/switch_lb_vip	String	Service □ Kube-OVN □□□□□□ VIP □□
ovn.kubernetes.io/vpc	String	Service □□□ VPC
ovn.kubernetes.io/service_external_ip_from_subnet	`true` or `false`	Service □□ IP □□□□□□□
ovn.kubernetes.io/service_health_check	`true` or `false`	Service □□□□□□□□
ovn.kubernetes.io/lb_svc_img	String	□□□□□□□□□□□

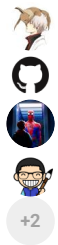
8.12.6 Networkpolicy Annotation

Key	Value	Description
ovn.kubernetes.io/enable_log	`true` or `false`	□□□□ NetworkPolicy □□
ovn.kubernetes.io/log_acl_actions	"allow,drop,pass" □□□□□□□□ □□	□□□□ Action ACL □□□
ovn.kubernetes.io/acl_log_meter_rate	Int	NetworkPolicy □□□□□□□□□□□□□□□/□
ovn.kubernetes.io/network_policy_enforcement	`standard` or `lax`	□□ NetworkPolicy □□□□□□□□□□□□□□
ovn.kubernetes.io/network_policy_for	String	□□□ NetworkPolicy □□□□□□□ provider□□□ `<nadName>.<nadNamespace>.ovn` □

PDF Slack Support

2026□10□29□

2024□6□12□

GitHub 

8.12.7 □□

2025 Kube-OVN Team

8.13 □□□□

为了保证文档风格一致，请在提交文档时遵循下列的风格规范。

8.13.1 □□

中文文档中文本内容所有标点应使用中文格式标点，英文文档中所有文本内容中应使用英文标点。

Bad	Good
这里提供了一键安装脚本，可以帮助你快速安装一个高可用，生产就绪的容器网络。	这里提供了一键安装脚本，可以帮助你快速安装一个高可用，生产就绪的容器网络。

英文数字和中文应该用空格进行分隔。

Bad	Good
Kube-OVN 提供了一键安装脚本来安装 1.10 版本 Kube-OVN。	Kube-OVN 提供了一键安装脚本来安装 1.10 版本 Kube-OVN。

示例内容应该以 `` 开启，其他句尾需要用 `` 结束。

Bad	Good
安装前请确认环境配置正确 使用下面的命令下载安装脚本: `wget 127.0.0.1`	安装前请确认环境配置正确。使用下面的命令下载安装脚本: `wget 127.0.0.1`

8.13.2 □□□

yaml 代码块需要标识为 yaml。

Bad	Good

``
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: attach-subnet
``
	``yaml
	apiVersion: kubeovn.io/v1
	kind: Subnet
	metadata:
	  name: attach-subnet
	``

命令行操作示例代码块需要标识为 bash。

Bad	Good

``
wget 127.0.0.1
``
	``bash
	wget 127.0.0.1
	``

2025 Kube-OVN Team

▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `#` ▯▯▯▯▯▯▯▯▯▯▯▯▯

Bad	Good

`oilbeater@macdeMac-3 ~ ping 114.114.114.114 -c 3`
`PING 114.114.114.114 (114.114.114.114): 56 data bytes`
`64 bytes from 114.114.114.114: icmp_seq=0 ttl=83 time=10.429 ms`
`64 bytes from 114.114.114.114: icmp_seq=1 ttl=79 time=11.360 ms`
`64 bytes from 114.114.114.114: icmp_seq=2 ttl=76 time=10.794 ms`
 	
`--- 114.114.114.114 ping statistics ---`
`3 packets transmitted, 3 packets received, 0.0% packet loss`
`round-trip min/avg/max/stddev = 10.429/10.861/11.360/0.383 ms`
	`# ping 114.114.114.114 -c 3`
	`PING 114.114.114.114 (114.114.114.114): 56 data bytes`
	`64 bytes from 114.114.114.114: icmp_seq=0 ttl=83 time=10.429 ms`
	`64 bytes from 114.114.114.114: icmp_seq=1 ttl=79 time=11.360 ms`
	`64 bytes from 114.114.114.114: icmp_seq=2 ttl=76 time=10.794 ms`
	
	`--- 114.114.114.114 ping statistics ---`
	`3 packets transmitted, 3 packets received, 0.0% packet loss`
	`round-trip min/avg/max/stddev = 10.429/10.861/11.360/0.383 ms`

▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ `#` ▯▯▯

Bad	Good
`# mv /etc/origin/ovn/ovnnb_db.db /tmp` `# mv /etc/origin/ovn/ovnsb_db.db /tmp`	`mv /etc/origin/ovn/ovnnb_db.db /tmp` `mv /etc/origin/ovn/ovnsb_db.db /tmp`

8.13.3 ▯▯

▯▯▯▯▯▯▯▯ `md` ▯▯▯▯▯

Bad	Good
`[ ](http://kubeovn.github.io/prepare)`	`[ ](../start/prepare.md)`

Bad	Good
`[Kubernetes ](http://kubernetes.io)`	`[Kubernetes ](http://kubernetes.io){: target="_blank" }`

8.13.4 ▯▯

▯▯

Bad	Good

```` ```bash
wget 127.0.0.1
``` ````
	```` ```bash
	wget 127.0.0.1
	``` ````

▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯

Bad	Good

```` ```bash
wget 127.0.0.1
``` ````
	```` ```bash
	wget 127.0.0.1
	``` ````

PDF Slack Support

2025 Kube-OVN Team

2026▯11▯7▯

2022▯7▯19▯

GitHub 

8.13.5 ▯▯



9. [illegible]

Kube-OVN 社区欢迎所有用户和开发者参与，可通过下面的方式与我们沟通：

9.1 在线交流

- Slack: 在 CNCF Slack 加入后访问 #kube-ovn 频道
- LinkedIn: Kube-OVN 官方主页
- Issue / 讨论: 在 GitHub Issues 反馈问题，GitHub Discussions 参与设计讨论

9.2 中文社区

- 关注公众号获取最新动态，请扫描下方二维码：

9.3 [illegible]

- 阅读 开发指南 与 架构说明 了解参与方式
- 加入 USERS.md 列出您的使用场景

PDF Slack Support

2026年4月28日

2022年6月20日

GitHub

2025 Kube-OVN Team