适用于 OpenShift 的 NSX Container Plug-in - 安装和管理指南

VMware NSX Container Plug-in 2.4
VMware NSX-T Data Center 2.4

vmware®

您可以从 VMware 网站下载最新的技术文档：
https://docs.vmware.com/cn/。
VMware 网站还提供了最近的产品更新。
如果您对本文档有任何意见或建议，请将反馈信息发送至：
docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

北京办公室
北京市
朝阳区新源南路 8 号
启皓北京东塔 8 层 801
www.vmware.com/cn

上海办公室
上海市
淮海中路 333 号
瑞安大厦 804-809 室
www.vmware.com/cn

广州办公室
广州市
天河路 385 号
太古汇一座 3502 室
www.vmware.com/cn

目录

适用于 OpenShift 的 NSX-T Container Plug-in - 安装和管理指南

本指南介绍了如何安装和管理 NSX Container Plug-in (NCP) 以提供 NSX-T Data Center 和 OpenShift 之间的集成。

目标读者

本指南适用于系统和网络管理员。假定用户熟悉 NSX-T Data Center 和 OpenShift 的安装和管理。

VMware 技术出版物术语表

VMware 技术出版物提供了一个术语表，其中包含一些您可能不熟悉的术语。有关 VMware 技术文档中使用的术语的定义，请访问 http://www.vmware.com/support/pubs。

NSX-T Container Plug-in 概述

1

NSX Container Plug-in(NCP) 提供 NSX-T Data Center 和容器协调器（如 Kubernetes）之间的集成以及 NSX-T Data Center 和基于容器的 PaaS（平台即服务）的软件产品（如 OpenShift）之间的集成。本指南介绍了如何使用 OpenShift 设置 NCP。

NCP 的主要组件在容器中运行，并与 NSX Manager 和 OpenShift 控制层面进行通信。NCP 调用 NSX API 以监控对容器和其他资源的更改以及管理网络资源，如容器的逻辑端口、交换机、路由器和安全组。

NSX CNI 插件在每个 OpenShift 节点上运行。它监控容器生命周期事件，将容器接口连接到客户机 vSwitch，并对客户机 vSwitch 进行编程以标记和转发容器接口和 vNIC 之间的容器流量。

NCP 提供了以下功能：

- 自动为 OpenShift 群集创建 NSX-T 逻辑拓扑，并为每个 OpenShift 命名空间创建一个单独的逻辑网络。
- 将 OpenShift pod 连接到逻辑网络，并分配 IP 和 MAC 地址。
- 支持网络地址转换 (NAT) 并为每个 OpenShift 命名空间分配一个单独的 SNAT IP。

  注 配置 NAT 时，转换后 IP 的总数不能超过 1000。

- 使用 NSX-T 分布式防火墙实现 OpenShift 网络策略。
  - 支持输入和输出网络策略。
  - 支持网络策略中的 `IPBlock` 选择器。
  - 为网络策略指定标签选择器时，支持 `matchLabels` 和 `matchExpression`。
- 使用 NSX-T 第 7 层负载平衡器实现 OpenShift 路由。
  - 通过 TLS Edge 终止支持 HTTP 路由和 HTTPS 路由。
  - 支持具有备用后端和通配符子域的路由。
- 在 NSX-T 逻辑交换机端口上为命名空间、pod 名称和 pod 标签创建标记，并允许管理员根据标记定义 NSX-T Data Center 安全组和策略。

在该版本中，NCP 支持单个 OpenShift 群集。

本章讨论了以下主题：

兼容性要求

NSX Container Plug-in (NCP) 具有以下兼容性要求。

软件产品	版本
NSX-T Data Center	2.3、2.4
容器主机虚拟机的管理程序	■ 支持的 vSphere 版本 ■ RHEL KVM 7.4、7.5、7.6
容器主机操作系统	RHEL 7.4、7.5、7.6
平台即服务	OpenShift 3.10、3.11
容器主机 Open vSwitch	2.10.2（随 NSX-T Data Center 2.4 打包）

安装概述

安装和配置 NCP 包括以下步骤。要成功执行这些步骤，您必须熟悉 NSX-T Data Center 和 OpenShift 安装和管理。

1 安装 NSX-T Data Center。

2 创建一个覆盖网络传输区域。

3 创建一个覆盖网络逻辑交换机并将节点连接到该交换机。

4 创建一个 Tier-0 逻辑路由器。

5 为 pod 创建 IP 块。

6 为 SNAT（源网络地址转换）创建 IP 池。

7 准备 Ansible hosts 文件。

8 使用单个 playbook 安装 NCP 和 OpenShift。

升级 NCP

本节介绍如何将 NCP 升级到 2.4.0。

步骤

1 升级 CNI RPM 软件包、NSX 节点代理 DaemonSet 和 NCP ReplicationController。

2 准备 Ansible hosts 文件。

必须为每个节点指定参数 `openshift_node_group_name`。例如，

```
[nodes]
config-master.example.com openshift_hostname=config-master.example.com
openshift_node_group_name=config-master
```

3 （可选）配置负载平衡。

添加一个步骤，为 LoadBalancer 服务的外部 IP 地址指定不同的 IP 池。例如，

```
external_ip_pools_lb = <nsx ip pool name>
```

设置 NSX-T 资源

2

必须创建 NSX-T Data Center 资源以提供到 OpenShift 节点的网络。

配置 NSX-T 资源

您需要配置的 NSX-T Data Center 资源包括覆盖网络传输区域、Tier-0 逻辑路由器、用于连接节点虚拟机的逻辑交换机、Kubernetes 节点的 IP 块以及 SNAT 的 IP 池。

重要 如果与 NSX-T Data Center 2.4 或更高版本一起运行，则必须使用**高级网络和安全**选项卡配置 NSX-T 资源。

在 NCP 配置文件 `ncp.ini` 中，使用 UUID 或名称指定 NSX-T Data Center 资源。

覆盖网络传输区域

登录到 NSX Manager，然后查找用于容器网络的覆盖网络传输区域，或者创建新的覆盖网络传输区域。

通过在 `ncp.ini` 的 `[nsx_v3]` 部分中设置 `overlay_tz` 选项来指定群集的覆盖网络传输区域。此步骤是可选的。如果未设置 `overlay_tz`，NCP 将自动从 Tier-0 路由器检索覆盖网络传输区域 ID。

Tier-0 逻辑路由

登录到 NSX Manager，然后查找用于容器网络的路由器，或者创建新的路由器。

通过在 `ncp.ini` 的 `[nsx_v3]` 部分中设置 `tier0_router` 选项来指定群集的 Tier-0 逻辑路由器。

注 必须在活动-备用模式下创建路由器。

逻辑交换机

节点用于数据流量的 vNIC 必须连接到覆盖网络逻辑交换机。不要求将节点的管理接口连接到 NSX-T Data Center，但这样做将简化设置过程。可以通过登录到 NSX Manager 创建逻辑交换机。在交换机上创建逻辑端口，并将节点 vNIC 连接到这些端口。逻辑端口必须具有以下标记：

- 标记：`<cluster_name>`，范围：`ncp/cluster`
- 标记：`<node_name>`，范围：`ncp/node_name`

`<cluster_name>` 值必须与 `ncp.ini` 的 `[coe]` 部分中的 `cluster` 选项值匹配。

Kubernetes pod 的 IP 块

登录到 NSX Manager，然后创建一个或多个 IP 块。使用 CIDR 格式指定 IP 块。

通过在 `ncp.ini` 的 `[nsx_v3]` 部分中设置 `container_ip_blocks` 选项来指定 Kubernetes pod 的 IP 块。

您还可以专门为非 SNAT 命名空间创建 IP 块。

通过在 `ncp.ini` 的 `[nsx_v3]` 部分中设置 `no_snat_ip_blocks` 选项来指定非 SNAT IP 块。

如果在 NCP 运行时创建非 SNAT IP 块，您必须重新启动 NCP。否则，NCP 将继续使用共享 IP 块，直到这些块用尽。

注 在创建 IP 块时，前缀不能大于 NCP 配置文件 `ncp.ini` 中的 `subnet_prefix` 参数值。

SNAT 的 IP 池

将使用 IP 池分配 IP 地址，这些地址将用于通过 SNAT 规则转换 pod IP 以及通过 SNAT/DNAT 规则公开 Ingress 控制器，就像 Openstack 浮动 IP 一样。这些 IP 地址也称为外部 IP。

多个 Kubernetes 群集使用相同的外部 IP 池。每个 NCP 实例将该池的一部分用于它管理的 Kubernetes 群集。默认情况下，将使用 pod 子网的相同子网前缀。要使用不同的子网大小，请更新 `ncp.ini` 的 `[nsx_v3]` 部分中的 `external_subnet_prefix` 选项。

登录到 NSX Manager，然后创建一个池，或者查找现有的池。

通过在 `ncp.ini` 的 `[nsx_v3]` 部分中设置 `external_ip_pools` 选项来指定 SNAT 的 IP 池。

您还可以通过向特定服务添加注释来为此服务配置 SNAT。例如，

```
apiVersion: v1
kind: Service
metadata:
  name: svc-example
  annotations:
    ncp/snat_pool: <external IP pool ID or name>
  selector:
    app: example
...
```

NCP 将为此服务配置 SNAT 规则。规则的源 IP 为后端 pod 集。目标 IP 是从指定外部 IP 池分配的 SNAT IP。请注意以下事项：

- 配置服务之前，`ncp/snat_pool` 指定的 IP 池应该已存在于 NSX-T Data Center 中。IP 池必须具有标记 **`{"ncp/owner": cluster:<cluster>}`**。
- 在 NSX-T Data Center 中，服务 SNAT 规则的优先级高于项目 SNAT 规则。
- 如果 pod 配置了多个 SNAT 规则，则只有一个起作用。

可以通过在 SNAT IP 池中添加以下标记指定可为哪些命名空间分配 IP 池中的 IP。

- 范围：`ncp/owner`，标记：`ns:<namespace_UUID>`

可以使用以下命令之一获取命名空间 UUID：

```
oc get ns –o yaml
```

请注意以下事项：

- 每个标记应指定一个 UUID。可以为同一个池创建多个标记。
- 如果根据旧标记为某些命名空间分配 IP 后更改了标记，则服务的 SNAT 配置更改或 NCP 重新启动之前，将不会回收这些 IP。
- 命名空间所有者标记是可选的。如果没有此标记，任何命名空间的 IP 都可以从 SNAT IP 池分配。

（可选）防火墙标记区域

要允许管理员创建防火墙规则且不干扰 NCP 基于网络策略创建的防火墙区域，请登录到 NSX Manager，然后创建两个防火墙区域。

通过在 `ncp.ini` 的 `[nsx_v3]` 部分中设置 `bottom_firewall_section_marker` 和 `top_firewall_section_marker` 选项来指定标记防火墙区域。

底部防火墙区域必须在顶部防火墙区域的下方。创建这些防火墙区域后，NCP 创建的用于隔离的所有防火墙区域将创建在底部防火墙区域的上方，NCP 创建的用于策略的所有防火墙区域将创建在顶部防火墙区域的下方。如果未创建这些标记区域，则将在底部创建所有隔离规则，在顶部创建所有策略区域。不支持每个群集中多个标记防火墙区域采用相同的值，这将会导致错误。

安装 NCP

3

NCP 已与 OpenShift 完全集成。在 Ansible hosts 文件中添加所需参数并安装 OpenShift 时，将自动安装 NCP。

本章讨论了以下主题：

- 系统要求
- 准备 Ansible hosts 文件

系统要求

在安装 OpenShift 之前，请确保您的环境满足特定要求。

常规要求

- Ansible 2.4 或更高版本。

虚拟机要求

Openshift 节点虚拟机必须具有两个 vNIC：

- 管理 vNIC，连接到具有到管理 Tier-1 路由器的上行链路的逻辑交换机。
- 辅助 vNIC，所有虚拟机上的辅助 vNIC 在 NSX-T 中必须具有以下标记，以便 NCP 知道哪个端口用作在特定 OpenShift 节点上运行的所有 POD 的父 VIF。

```
{'ncp/node_name':  '<node_name>'}
{'ncp/cluster': '<cluster_name>'}
```

裸机要求

- OpenShift 节点必须是 NSX-T 传输节点，且上述标记必须应用于传输节点，而不是 VIF。
- Ansible hosts 文件必须具有以下设置：`nsx_node_type='BAREMETAL'`。

NSX-T 要求

- 一个 Tier-0 路由器。

- 一个覆盖网络传输区域。
- POD 网络的 IP 块。
- （可选）已路由（无 NAT）POD 网络的 IP 块。
- SNAT 的 IP 池。默认情况下，POD 网络的 IP 块仅可在 NSX-T 内路由。NCP 使用此 IP 池提供与外部的连接。
- （可选）顶部和底部防火墙区域。NCP 将在这两个区域之间放置 Kubernetes 网络策略规则。
- Open vSwitch 和 CNI 插件 RPM 必须在可从 OpenShift 节点虚拟机访问的 HTTP 服务器上托管。

NCP Docker 映像

当前，NCP docker 映像未公开提供。在本地专用注册表中必须具有映像 `nsx-ncp`，或者执行以下操作：

`ansible-playbook [-i /path/to/inventory] playbooks/prerequisites.yml`

在所有节点上：

```
docker load -i nsx-ncp-rhel-xxx.yyyyyyyy.tar
docker image tag registry.local/xxx.yyyyyyyy/nsx-ncp-rhel nsx-ncp
ansible-playbook [-i /path/to/inventory] playbooks/deploy_cluster.yml
```

准备 Ansible hosts 文件

必须在 Ansible hosts 文件中指定 NCP 参数，NCP 才能与 OpenShift 集成。

在 Ansible hosts 文件中指定以下参数后，安装 OpenShift 时将自动安装 NCP。

- `openshift_use_nsx=True`
- `openshift_use_openshift_sdn=False`
- `os_sdn_network_plugin_name='cni'`
- `nsx_openshift_cluster_name='ocp-cluster1'`

  （必需）这是必需的参数，因为多个 Openshift/Kubernetes 群集可以连接到同一 NSX Manager。
- `nsx_api_managers='10.10.10.10'`

  （必需）NSX Manager 的 IP 地址。对于 NSX Manager 群集，指定以逗号分隔的 IP 地址。
- `nsx_tier0_router='MyT0Router'`

  （必需）项目的 Tier-1 路由器将连接到的 Tier-0 路由器的名称或 UUID。
- `nsx_overlay_transport_zone='my_overlay_tz'`

  （必需）将用于创建逻辑交换机的覆盖网络传输区域的名称或 UUID。
- `nsx_container_ip_block='ip_block_for_my_ocp_cluster'`

  （必需）在 NSX-T 上配置的 IP 块的名称或 UUID。在此 IP 块之外，每个项目都将有一个子网。这些网络将位于 SNAT 后面且不可路由。

- `nsx_ovs_uplink_port='ens224'`

  （必需）如果处于 `HOSTVM` 模式。NSX-T 需要对 OCP 节点上的 POD 网络使用辅助 vNIC，且该 vNIC 与管理 vNIC 不同。强烈建议将这两个 vNIC 连接到 NSX-T 逻辑交换机。必须在此处提供辅助（非管理）vNIC。对于裸机，不需要此参数。

- `nsx_cni_url='http://myserver/nsx-cni.rpm'`

  （必需）NCP 可以引导节点之前的临时要求。我们需要将 `nsx-cni` 放置在 http 服务器上。

- `nsx_ovs_url='http://myserver/openvswitch.rpm'`

- `nsx_kmod_ovs_url='http://myserver/kmod-openvswitch.rpm'`

  （必需）NCP 可以引导节点之前的临时参数。在裸机设置中，可以忽略此参数。

- `nsx_node_type='HOSTVM'`

  （可选）默认为 `HOSTVM`。如果 OpenShift 未在虚拟机中运行，则设置为 `BAREMETAL`。

- `nsx_k8s_api_ip=192.168.10.10`

  （可选）如果设置，则 NCP 将与此 IP 地址通信，否则与 Kubernetes 服务 IP 通信。

- `nsx_k8s_api_port=192.168.10.10`

  （可选）对于 Kubernetes 服务，默认为 443。如果将其与 `nsx_k8s_api_ip` 组合使用以指定主节点 IP，则设置为 8443。

- `nsx_insecure_ssl=true`

  （可选）默认为 `true`，因为 NSX Manager 附带不受信任的证书。如果已将证书更改为受信任的证书，则可以设置为 `false`。

- `nsx_api_user='admin'`

- `nsx_api_password='super_secret_password'`

- `nsx_subnet_prefix=24`

  （可选）默认为 24。这是每个 Openshift 项目将专用的子网大小。如果 POD 的数量超过子网大小，则将具有相同子网大小的新逻辑交换机添加到项目。

- `nsx_use_loadbalancer=true`

  （可选）默认为 `true`。如果不希望将 NSX-T 负载平衡器用于 LoadBalancer 类型的 OpenShift 路由和服务，则设置为 `false`。

- `nsx_lb_service_size='SMALL'`

  （可选）默认为 `SMALL`。也可以设置为 `MEDIUM` 或 `LARGE`，具体取决于 NSX Edge 大小。

- `nsx_no_snat_ip_block='router_ip_block_for_my_ocp_cluster'`

  （可选）如果在项目或命名空间上应用 `ncp/no_snat=true` 注释，则子网将从此 IP 块获取且不存在 SNAT。应可进行路由。

- `nsx_external_ip_pool='external_pool_for_snat'`

  （必需）SNAT 和负载平衡器的 IP 池（如果未定义 `nsx_external_ip_pool_lb`）。

- `nsx_external_ip_pool_lb='my_ip_pool_for_lb'`

  （可选）如果需要对 `Router` 和 `SvcTypeLB` 使用不同的 IP 池，请设置此参数。

- `nsx_top_fw_section='top_section'`

  （可选）Kubernetes 网络策略规则将转换为 NSX-T 防火墙规则，并放置在此区域之下。

- `nsx_bottom_fw_section='bottom_section'`

  （可选）Kubernetes 网络策略规则将转换为 NSX-T 防火墙规则，并放置在此区域之上。

- `nsx_api_cert='/path/to/cert/nsx.crt'`

- `nsx_api_private_key='/path/to/key/nsx.key`

  （可选）如果设置，则将忽略 `nsx_api_user` 和 `nsx_api_password`。必须将证书上载到 NSX-T，且必须手动创建使用此证书进行身份验证的主体身份用户。

- `nsx_lb_default_cert='/path/to/cert/nsx.crt'`

- `nsx_lb_default_key='/path/to/key/nsx.key`

  （可选）NSX-T 负载平衡器需要默认证书，才能为基于 TLS 的路由创建 SNI。仅当未配置路由时，才提供此证书。如果未提供，将生成自签名证书。

Ansible Hosts 文件示例

```
[OSEv3:children]
masters
nodes
etcd

[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=origin

openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true',
'kind': 'HTPasswdPasswordIdentityProvider'}]
openshift_master_htpasswd_users={'yasen' : 'password'}

openshift_master_default_subdomain=demo.corp.local
openshift_use_nsx=true
os_sdn_network_plugin_name=cni
openshift_use_openshift_sdn=false
openshift_node_sdn_mtu=1500

# NSX specific configuration
nsx_openshift_cluster_name='ocp-cluster1'
nsx_api_managers='192.168.110.201'
nsx_api_user='admin'
nsx_api_password='VMware1!'
```

```
nsx_tier0_router='DefaultT0Router'
nsx_overlay_transport_zone='overlay-tz'
nsx_container_ip_block='ocp-pod-networking'
nsx_no_snat_ip_block='ocp-nonat-pod-networking'
nsx_external_ip_pool='ocp-external'
nsx_top_fw_section='openshift-top'
nsx_bottom_fw_section='openshift-bottom'
nsx_ovs_uplink_port='ens224'
nsx_cni_url='http://1.1.1.1/nsx-cni-2.3.2.x86_64.rpm'
nsx_ovs_url='http://1.1.1.1/openvswitch-2.9.1.rhel75-1.x86_64.rpm'
nsx_kmod_ovs_url='http://1.1.1.1/kmod-openvswitch-2.9.1.rhel75-1.el7.x86_64.rpm'

[masters]
ocp-master.corp.local

[etcd]
ocp-master.corp.local

[nodes]
ocp-master.corp.local ansible_ssh_host=10.1.0.10 openshift_node_group_name='node-config-master'
ocp-node1.corp.local ansible_ssh_host=10.1.0.11 openshift_node_group_name='node-config-infra'
ocp-node2.corp.local ansible_ssh_host=10.1.0.12 openshift_node_group_name='node-config-infra'
ocp-node3.corp.local ansible_ssh_host=10.1.0.13 openshift_node_group_name='node-config-compute'
ocp-node4.corp.local ansible_ssh_host=10.1.0.14 openshift_node_group_name='node-config-compute'
```

负载平衡

4

NSX-T Data Center 负载平衡器与 OpenShift 集成，并充当 OpenShift 路由器。

NCP 监视 OpenShift 路由和端点事件，并根据路由规范在负载平衡器上配置负载平衡规则。因此，NSX-T Data Center 负载平衡器会根据规则将入站第 7 层流量转发到适当的后端容器。

配置负载平衡

配置负载平衡需要配置 Kubernetes LoadBalancer 服务或 OpenShift 路由。此外，还需要配置 NCP 复制控制器。LoadBalancer 服务用于第 4 层流量，而 OpenShift 路由用于第 7 层流量。

配置 Kubernetes LoadBalancer 服务时，会从您配置的外部 IP 块为该服务分配一个 IP 地址。会在此 IP 地址和服务端口上公开负载平衡器。可以使用 `loadBalancerIP` 规范在 LoadBalancer 定义中指定 IP 池的名称或 ID。将从该 IP 池分配 Loadbalancer 服务的 IP。如果 `loadBalancerIP` 规范为空，将从您配置的外部 IP 块分配 IP。

`loadBalancerIP` 指定的 IP 池必须具有标记 **`{"ncp/owner": cluster:<cluster>}`**。

要使用 NSX-T Data Center 负载平衡器，必须在 NCP 中配置负载平衡。在 `ncp_rc.yml` 文件中，执行以下操作：

1 将 `use_native_loadbalancer` 设置为 `True`。

2 将 `pool_algorithm` 设置为 `WEIGHTED_ROUND_ROBIN`。

3 将 lb_default_cert_path 和 lb_priv_key_path 分别设置为 CA 签名证书文件和私钥文件的完整路径名称。有关用于生成 CA 签名证书的示例脚本，请参见下文。此外，将默认证书和密钥挂载到 NCP pod 中。有关说明，请参见下文。

4 （可选）使用参数 `l4_persistence` 和 `l7_persistence` 指定持久性设置。可用于设置第 4 层持久性的选项为源 IP。可用于设置第 7 层持久性的选项为 cookie 和源 IP。默认值为 <None>。例如，

```
# Choice of persistence type for ingress traffic through L7 Loadbalancer.
# Accepted values:
# 'cookie'
# 'source_ip'
l7_persistence = cookie
```

```
# Choice of persistence type for ingress traffic through L4 Loadbalancer.
# Accepted values:
# 'source_ip'
l4_persistence = source_ip
```

5 （可选）将 `service_size` 设置为 `SMALL`、`MEDIUM` 或 `LARGE`。默认值为 `SMALL`。

6 如果运行的是 OpenShift 3.11，则必须执行以下配置，OpenShift 才不会向 LoadBalancer 服务分配 IP。

- 在 `/etc/origin/master/master-config.yaml` 文件中的 `networkConfig` 下，将 `ingressIPNetworkCIDR` 设置为 0.0.0.0/32。
- 使用以下命令重新启动 API 服务器和控制器：

```
master-restart api
master-restart controllers
```

对于 Kubernetes LoadBalancer 服务，如果禁用了全局第 4 层持久性（即 `l4_persistence` 设置为 `<None>`），您还可以在服务规范上指定 `sessionAffinity` 以配置服务持久性行为。如果 `l4_persistence` 设置为 `source_ip`，则可以使用服务规范上的 `sessionAffinity` 自定义服务持久性超时。默认第 4 层持久性超时为 10800 秒，与 Kubernetes 文档 (https://kubernetes.io/docs/concepts/services-networking/service) 中指定的服务超时相同。具有默认持久性超时的所有服务将使用相同的 NSX-T 负载平衡器持久性配置文件。将为每个具有非默认持久性超时的服务创建专用的配置文件。

注 如果 Ingress 的后端服务的服务类型为 LoadBalancer，则该服务的第 4 层虚拟服务器和 Ingress 的第 7 层虚拟服务器不能具有不同的持久性设置，例如，对于第 4 层，采用 `source_ip`，而对于第 7 层，采用 `cookie`。在这种情况下，这两个虚拟服务器的持久性设置必须相同（`source_ip`、`cookie` 或 `None`），或者其中一个为 `None`（另一个的设置可以为 `source_ip` 或 `cookie`）。下面列出了这种情况的一个示例：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
-----
apiVersion: v1
kind: Service
metadata:
  name: tea-svc <==== same as the Ingress backend above
  labels:
    app: tea
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
    name: tcp
  selector:
    app: tea
  type: LoadBalancer
```

路由器分片

NCP 始终处理 TLS Edge 终止和 HTTP 路由，并跳过 TLS 直通路由和 TLS 重新加密路由，而不管其命名空间或命名空间标签为何。要将 OpenShift 路由器限制为仅处理 TLS 重新加密路由和直通路由，必须执行以下步骤：

- 将命名空间标签选择器添加到 Openshift 路由器。

- 将命名空间标签添加到目标命名空间。
- 在目标命名空间中创建 TLS 重新加密路由/直通路由。

例如，要为路由器配置命名空间标签选择器，请运行以下命令（假设路由器的服务帐户名称为 `router`）：

```
oc set env dc/router NAMESPACE_LABELS="router=r1"
```

该路由器现在将处理来自选定命名空间的路由。要使此选择器与命名空间匹配，请运行以下命令（假设命名空间名为 `ns1`）：

```
oc label namespace ns1 "router=r1"
```

第 7 层负载平衡器示例

以下 YAML 文件会配置两个复制控制器（tea-rc 和 coffee-rc）、两个服务（tea-svc 和 coffee-svc）以及两个路由（cafe-route-multi 和 cafe-route），以提供第 7 层负载平衡。

```
# RC
apiVersion: v1
kind: ReplicationController
metadata:
  name: tea-rc
spec:
  replicas: 2
  template:
    metadata:
       labels:
         app: tea
    spec:
      containers:
      - name: tea
        image: nginxdemos/hello
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: coffee-rc
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
      - name: coffee
        image: nginxdemos/hello
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
---
```

```
# Services
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
    app: tea
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
    name: http
  selector:
    app: tea
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
  labels:
    app: coffee
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
    name: http
  selector:
    app: coffee
---
# Routes
apiVersion: v1
kind: Route
metadata:
  name: cafe-route-multi
spec:
  host: www.cafe.com
  path: /drinks
  to:
    kind: Service
    name: tea-svc
    weight: 1
  alternateBackends:
  - kind: Service
    name: coffee-svc
    weight: 2
---
apiVersion: v1
kind: Route
metadata:
  name: cafe-route
spec:
  host: www.cafe.com
  path: /tea-svc
```

```
  to:
    kind: Service
    name: tea-svc
    weight: 1
```

其他说明

- 仅对 HTTPS 流量支持 Edge 终止。
- 支持通配符子域。例如，如果 `wildcardPolicy` 设置为 **`Subdomain`**，且主机名设置为 **`wildcard.example.com`**，则会处理针对 **`*.example.com`** 的任何请求。
- 如果 NCP 在处理路由事件期间由于配置错误而引发错误，则需要更正路由 YAML 文件，删除并重新创建路由资源。
- NCP 不按命名空间实施主机名所有权。
- 每个 Kubernetes 群集支持一个 Loadbalancer 服务。
- NSX-T Data Center 将为每个 LoadBalancer 服务端口创建一个第 4 层负载平衡器虚拟服务器和池。TCP 和 UDP 均受支持。
- NSX-T Data Center 负载平衡器有多种不同大小。有关配置 NSX-T Data Center 负载平衡器的信息，请参见《NSX-T Data Center 管理指南》。

  创建负载平衡器后，无法通过更新配置文件来更改负载平衡器大小，但是可以通过 UI 或 API 进行更改。
- 支持自动缩放第 4 层负载平衡器。如果创建或修改 Kubernetes LoadBalancer 服务以便需要更多的虚拟服务器，而现有的第 4 层负载平衡器没有足够的容量，将创建新的第 4 层负载平衡器。NCP 也将删除不再连接虚拟服务器的第 4 层负载平衡器。默认情况下，将启用该功能。可以通过在 NCP ConfigMap 中将 `l4_lb_auto_scaling` 设置为 **`false`** 禁用此功能。

用于生成 CA 签名证书的示例脚本

以下脚本可分别生成存储在文件 <filename>.crt 和 <finename>.key 中的 CA 签名证书和私钥。`genrsa` 命令可生成 CA 密钥。应对 CA 密钥进行加密。您可以使用 `aes256` 等命令指定加密方法。

```
#!/bin/bash
host="www.example.com"
filename=server

openssl genrsa -out ca.key 4096
openssl req -key ca.key -new -x509 -days 365 -sha256 -extensions v3_ca -out ca.crt -subj
"/C=US/ST=CA/L=Palo Alto/O=OS3/OU=Eng/CN=${host}"
openssl req -out ${filename}.csr -new -newkey rsa:2048 -nodes -keyout ${filename}.key -subj
"/C=US/ST=CA/L=Palo Alto/O=OS3/OU=Eng/CN=${host}"
openssl x509 -req -days 360 -in ${filename}.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out $
{filename}.crt -sha256
```

将默认证书和密钥挂载到 NCP Pod 中

生成证书和私钥后，请将其置于主机虚拟机上的目录 `/etc/nsx-ujo` 中。假设证书文件和密钥文件分别命名为 `lb-default.crt` 和 `lb-default.key`，请编辑 `ncp-rc.yaml` 以便主机上的这些文件挂载到 pod 中。例如，

```
spec:
  ...
  containers:
  - name: nsx-ncp
    ...
    volumeMounts:
    ...
    - name: lb-default-cert
      # Mount path must match nsx_v3 option "lb_default_cert_path"
      mountPath: /etc/nsx-ujo/lb-default.crt
    - name: lb-priv-key
      # Mount path must match nsx_v3 option "lb_priv_key_path"
      mountPath: /etc/nsx-ujo/lb-default.key
  volumes:
  ...
  - name: lb-default-cert
    hostPath:
      path: /etc/nsx-ujo/lb-default.crt
  - name: lb-priv-key
    hostPath:
      path: /etc/nsx-ujo/lb-default.key
```

管理 NSX Container Plug-in 5

您可以从 NSX Manager GUI 或命令行界面 (Command Line Interface, CLI) 中管理 NSX Container Plug-in。

注 如果容器主机虚拟机在 ESXi 6.5 上运行，并且该虚拟机通过 vMotion 迁移到另一个 ESXi 6.5 主机，则运行在容器主机上的容器将断开与运行在其他容器主机上的容器的连接。您可以通过先断开然后重新连接容器主机的 vNIC 来解决该问题。ESXi 6.5 Update 1 或更高版本不会出现此问题。

Hyperbus 会在管理程序中保留 VLAN ID 4094 用于 PVLAN 配置，且不能更改此 ID。为避免任何 VLAN 冲突，请勿将 VLAN 逻辑交换机或 VTEP vmknic 配置为具有相同的 VLAN ID。

本章讨论了以下主题：

从 NSX Manager GUI 中管理 IP 块

您可以从 NSX Manager GUI 中添加、删除和编辑 IP 块，查看 IP 块详细信息以及管理 IP 块标记。

步骤

1 从浏览器中，登录到 `https://<NSX Manager IP 地址或域名>` 中的 NSX Manager。

2 导航到**网络 > IPAM**。

将显示现有 IP 块列表。

3 执行任何以下操作。

选项	操作
添加 IP 块	单击**添加**。
删除一个或多个 IP 块	选择一个或多个 IP 块，然后单击**删除**。
编辑 IP 块	选择一个 IP 块，然后单击**编辑**。

选项	操作
查看有关 IP 块的详细信息	单击 IP 块名称。单击**概览**选项卡以查看常规信息。单击**子网**选项卡以查看该 IP 块的子网。
管理 IP 块的标记	选择一个 IP 块，然后单击**操作 > 管理标记**。

您无法删除已分配子网的 IP 块。

从 NSX Manager GUI 查看 IP 块子网

可以从 NSX Manager GUI 查看 IP 块的子网。不建议在安装并运行 NCP 后添加或删除 IP 块子网。

步骤

1 从浏览器中，登录到 `https://<NSX Manager IP 地址或域名>` 中的 NSX Manager。

2 导航到**网络 > IPAM**。

将显示现有 IP 块列表。

3 单击一个 IP 块名称。

4 单击**子网**选项卡。

CIF 连接的逻辑端口

CIF（容器接口）是容器上连接到交换机上的逻辑端口的网络接口。这些端口称为 CIF 连接的逻辑端口。

您可以从 NSX Manager GUI 中管理 CIF 连接的逻辑端口。

管理 CIF 连接的逻辑端口

导航到**网络 > 交换 > 端口**以查看所有逻辑端口，包括 CIF 连接的逻辑端口。单击 CIF 连接的逻辑端口的连接链接以查看连接信息。单击逻辑端口链接以打开包含四个选项卡的窗格：“概览”、“监控”、“管理”和“相关”。单击**相关 > 逻辑端口**将显示上行链路交换机上的相关逻辑端口。有关交换机端口的详细信息，请参阅《NSX-T 管理指南》。

网络监控工具

以下工具支持 CIF 连接的逻辑端口。有关这些工具的详细信息，请参阅《NSX-T 管理指南》。

- 跟踪流
- 端口连接
- IPFIX
- 支持使用连接到容器的逻辑交换机端口的 GRE 封装的远程端口镜像。有关详细信息，请参阅《NSX-T 管理指南》中的“了解端口镜像交换配置文件”。但是，不支持通过管理器 UI 执行 CIF 到 VIF 端口的端口镜像。

CLI 命令

要运行 CLI 命令，请登录到 NSX Container Plug-in 容器，然后打开一个终端并运行 `nsxcli` 命令。

也可以在节点上运行以下命令以显示 CLI 提示符：

```
kubectl exec –it <pod name> nsxcli
```

表 5-1. 用于 NCP 容器的 CLI 命令

类型	命令
状态	get ncp-master status
状态	get ncp-nsx status
状态	get ncp-watcher <watcher-name>
状态	get ncp-watchers
状态	get ncp-k8s-api-server status
状态	check projects
状态	check project <project-name>
缓存	get project-cache <project-name>
缓存	get project-caches
缓存	get namespace-cache <namespace-name>
缓存	get namespace-caches
缓存	get pod-cache <pod-name>
缓存	get pod-caches
缓存	get ingress-caches
缓存	get ingress-cache <ingress-name>
缓存	get ingress-controllers
缓存	get ingress-controller <ingress-controller-name>
缓存	get network-policy-caches
缓存	get network-policy-cache <pod-name>
支持	get ncp-log file <filename>
支持	get ncp-log-level
支持	set ncp-log-level <log-level>
支持	get support-bundle file <filename>
支持	get node-agent-log file <filename>
支持	get node-agent-log file <filename> <node-name>

表 5-2. 用于 NSX 节点代理容器的 CLI 命令

类型	命令
状态	get node-agent-hyperbus status
缓存	get container-cache <container-name>
缓存	get container-caches

表 5-3. 用于 NSX Kube 代理容器的 CLI 命令

类型	命令
状态	get ncp-k8s-api-server status
状态	get kube-proxy-watcher <watcher-name>
状态	get kube-proxy-watchers
状态	dump ovs-flows

用于 NCP 容器的状态命令

- 显示 NCP 主节点的状态

  ```
  get ncp-master status
  ```

  示例：

  ```
  kubenode> get ncp-master status
  This instance is not the NCP master
  Current NCP Master id is a4h83eh1-b8dd-4e74-c71c-cbb7cc9c4c1c
  Last master update at Wed Oct 25 22:46:40 2017
  ```

- 显示 NCP 和 NSX Manager 之间的连接状态

  ```
  get ncp-nsx status
  ```

  示例：

  ```
  kubenode> get ncp-nsx status
  NSX Manager status: Healthy
  ```

- 显示 Ingress、命名空间、pod 和服务的监视程序状态

  ```
  get ncp-watcher <watcher-name>
  get ncp-watchers
  ```

示例 1：

```
kubenode> get ncp-watcher pod
    Average event processing time: 1174 msec (in past 3600-sec window)
    Current watcher started time: Mar 02 2017 10:47:35 PST
    Number of events processed: 1 (in past 3600-sec window)
    Total events processed by current watcher: 1
    Total events processed since watcher thread created: 1
    Total watcher recycle count: 0
    Watcher thread created time: Mar 02 2017 10:47:35 PST
    Watcher thread status: Up
```

示例 2：

```
kubenode> get ncp-watchers
    pod:
        Average event processing time: 1145 msec (in past 3600-sec window)
        Current watcher started time: Mar 02 2017 10:51:37 PST
        Number of events processed: 1 (in past 3600-sec window)
        Total events processed by current watcher: 1
        Total events processed since watcher thread created: 1
        Total watcher recycle count: 0
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up

    namespace:
        Average event processing time: 68 msec (in past 3600-sec window)
        Current watcher started time: Mar 02 2017 10:51:37 PST
        Number of events processed: 2 (in past 3600-sec window)
        Total events processed by current watcher: 2
        Total events processed since watcher thread created: 2
        Total watcher recycle count: 0
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up

    ingress:
        Average event processing time: 0 msec (in past 3600-sec window)
        Current watcher started time: Mar 02 2017 10:51:37 PST
        Number of events processed: 0 (in past 3600-sec window)
        Total events processed by current watcher: 0
        Total events processed since watcher thread created: 0
        Total watcher recycle count: 0
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up

    service:
        Average event processing time: 3 msec (in past 3600-sec window)
        Current watcher started time: Mar 02 2017 10:51:37 PST
        Number of events processed: 1 (in past 3600-sec window)
        Total events processed by current watcher: 1
```

```
        Total events processed since watcher thread created: 1
        Total watcher recycle count: 0
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up
```

- 显示 NCP 和 Kubernetes API 服务器之间的连接状态

```
get ncp-k8s-api-server status
```

示例：

```
kubenode> get ncp-k8s-api-server status
Kubernetes ApiServer status: Healthy
```

- 检查所有项目或特定项目

```
check projects
check project <project-name>
```

示例：

```
kubenode> check projects
    default:
        Tier-1 link port for router 1b90a61f-0f2c-4768-9eb6-ea8954b4f327 is missing
        Switch 40a6829d-c3aa-4e17-ae8a-7f7910fdf2c6 is missing

    ns1:
        Router 8accc9cd-9883-45f6-81b3-0d1fb2583180 is missing

kubenode> check project default
    Tier-1 link port for router 1b90a61f-0f2c-4768-9eb6-ea8954b4f327 is missing
    Switch 40a6829d-c3aa-4e17-ae8a-7f7910fdf2c6 is missing
```

用于 NCP 容器的缓存命令

- 获取项目或命名空间的内部缓存

```
get project-cache <project-name>
get project-caches
get namespace-cache <namespace-name>
get namespace-caches
```

示例：

```
kubenode> get project-caches
    default:
        logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
        logical-switch:
            id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
            ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
            subnet: 10.0.0.0/24
```

```
            subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435

    kube-system:
        logical-router: 5032b299-acad-448e-a521-19d272a08c46
        logical-switch:
            id: 85233651-602d-445d-ab10-1c84096cc22a
            ip_pool_id: ab1c5b09-7004-4206-ac56-85d9d94bffa2
            subnet: 10.0.1.0/24
            subnet_id: 73e450af-b4b8-4a61-a6e3-c7ddd15ce751

    testns:
        ext_pool_id: 346a0f36-7b5a-4ecc-ad32-338dcb92316f
        labels:
            ns: myns
            project: myproject
        logical-router: 4dc8f8a9-69b4-4ff7-8fb7-d2625dc77efa
        logical-switch:
            id: 6111a99a-6e06-4faa-a131-649f10f7c815
            ip_pool_id: 51ca058d-c3dc-41fd-8f2d-e69006ab1b3d
            subnet: 50.0.2.0/24
            subnet_id: 34f79811-bd29-4048-a67d-67ceac97eb98
        project_nsgroup: 9606afee-6348-4780-9dbe-91abfd23e475
        snat_ip: 4.4.0.3

kubenode> get project-cache default
    logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
    logical-switch:
        id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
        ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
        subnet: 10.0.0.0/24
        subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435

kubenode> get namespace-caches
    default:
        logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
        logical-switch:
            id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
            ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
            subnet: 10.0.0.0/24
            subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435

    kube-system:
        logical-router: 5032b299-acad-448e-a521-19d272a08c46
        logical-switch:
            id: 85233651-602d-445d-ab10-1c84096cc22a
            ip_pool_id: ab1c5b09-7004-4206-ac56-85d9d94bffa2
            subnet: 10.0.1.0/24
            subnet_id: 73e450af-b4b8-4a61-a6e3-c7ddd15ce751

    testns:
        ext_pool_id: 346a0f36-7b5a-4ecc-ad32-338dcb92316f
        labels:
            ns: myns
            project: myproject
        logical-router: 4dc8f8a9-69b4-4ff7-8fb7-d2625dc77efa
```

```
        logical-switch:
            id: 6111a99a-6e06-4faa-a131-649f10f7c815
            ip_pool_id: 51ca058d-c3dc-41fd-8f2d-e69006ab1b3d
            subnet: 50.0.2.0/24
            subnet_id: 34f79811-bd29-4048-a67d-67ceac97eb98
        project_nsgroup: 9606afee-6348-4780-9dbe-91abfd23e475
        snat_ip: 4.4.0.3

kubenode> get namespace-cache default
    logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
    logical-switch:
        id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
        ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
        subnet: 10.0.0.0/24
        subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435
```

- 获取 pod 的内部缓存

```
get pod-cache <pod-name>
get pod-caches
```

示例：

```
kubenode> get pod-caches
    nsx.default.nginx-rc-uq2lv:
        cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
        gateway_ip: 10.0.0.1
        host_vif: d6210773-5c07-4817-98db-451bd1f01937
        id: 1c8b5c52-3795-11e8-ab42-005056b198fb
        ingress_controller: False
        ip: 10.0.0.2/24
        labels:
            app: nginx
        mac: 02:50:56:00:08:00
        port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
        vlan: 1

    nsx.testns.web-pod-1:
        cif_id: ce134f21-6be5-43fe-afbf-aaca8c06b5cf
        gateway_ip: 50.0.2.1
        host_vif: d6210773-5c07-4817-98db-451bd1f01937
        id: 3180b521-270e-11e8-ab42-005056b198fb
        ingress_controller: False
        ip: 50.0.2.3/24
        labels:
            app: nginx-new
            role: db
            tier: cache
        mac: 02:50:56:00:20:02
        port_id: 81bc2b8e-d902-4cad-9fc1-aabdc32ecaf8
        vlan: 3

kubenode> get pod-cache nsx.default.nginx-rc-uq2lv
    cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
```

```
    gateway_ip: 10.0.0.1
    host_vif: d6210773-5c07-4817-98db-451bd1f01937
    id: 1c8b5c52-3795-11e8-ab42-005056b198fb
    ingress_controller: False
    ip: 10.0.0.2/24
    labels:
        app: nginx
    mac: 02:50:56:00:08:00
    port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
    vlan: 1
```

- 获取网络策略缓存或特定缓存

```
get network-policy caches
get network-policy-cache <network-policy-name>
```

示例：

```
kubenode> get network-policy-caches
    nsx.testns.allow-tcp-80:
        dest_labels: None
        dest_pods:
            50.0.2.3
        match_expressions:
            key: tier
            operator: In
            values:
                cache
        name: allow-tcp-80
        np_dest_ip_set_ids:
            22f82d76-004f-4d12-9504-ce1cb9c8aa00
            np_except_ip_set_ids:
        np_ip_set_ids:
            14f7f825-f1a0-408f-bbd9-bb2f75d44666
        np_isol_section_id: c8d93597-9066-42e3-991c-c550c46b2270
        np_section_id: 04693136-7925-44f2-8616-d809d02cd2a9
        ns_name: testns
        src_egress_rules: None
        src_egress_rules_hash: 97d170e1550eee4afc0af065b78cda302a97674c
        src_pods:
            50.0.2.0/24
        src_rules:
            from:
                namespaceSelector:
                    matchExpressions:
                        key: tier
                        operator: DoesNotExist
                    matchLabels:
                        ns: myns
            ports:
                port: 80
                protocol: TCP
        src_rules_hash: e4ea7b8d91c1e722670a59f971f8fcc1a5ac51f1
```

```
kubenode> get network-policy-cache nsx.testns.allow-tcp-80
    dest_labels: None
    dest_pods:
        50.0.2.3
    match_expressions:
        key: tier
        operator: In
        values:
            cache
    name: allow-tcp-80
    np_dest_ip_set_ids:
        22f82d76-004f-4d12-9504-ce1cb9c8aa00
        np_except_ip_set_ids:
    np_ip_set_ids:
        14f7f825-f1a0-408f-bbd9-bb2f75d44666
    np_isol_section_id: c8d93597-9066-42e3-991c-c550c46b2270
    np_section_id: 04693136-7925-44f2-8616-d809d02cd2a9
    ns_name: testns
    src_egress_rules: None
    src_egress_rules_hash: 97d170e1550eee4afc0af065b78cda302a97674c
    src_pods:
        50.0.2.0/24
    src_rules:
        from:
            namespaceSelector:
                matchExpressions:
                    key: tier
                    operator: DoesNotExist
                matchLabels:
                    ns: myns
        ports:
            port: 80
            protocol: TCP
    src_rules_hash: e4ea7b8d91c1e722670a59f971f8fcc1a5ac51f1
```

用于 NCP 容器的支持命令

- 在文件存储中保存 NCP 支持包

  支持包包含 pod 中的所有容器的日志文件并具有 **`tier:nsx-networking`** 标签。包文件采用 tgz 格式，并保存在 CLI 默认文件存储目录 `/var/vmware/nsx/file-store` 中。您可以使用 CLI file-store 命令将包文件复制到远程站点中。

  ```
  get support-bundle file <filename>
  ```

  示例：

  ```
  kubenode>get support-bundle file foo
  Bundle file foo created in tgz format
  kubenode>copy file foo url scp://nicira@10.0.0.1:/tmp
  ```

- 在文件存储中保存 NCP 日志

  日志文件以 tgz 格式保存在 CLI 默认文件存储目录 /var/vmware/nsx/file-store 中。您可以使用 CLI file-store 命令将包文件复制到远程站点中。

  ```
  get ncp-log file <filename>
  ```

  示例：

  ```
  kubenode>get ncp-log file foo
  Log file foo created in tgz format
  ```

- 在文件存储中保存节点代理日志

  保存一个节点或所有节点的节点代理日志。日志以 tgz 格式保存在 CLI 默认文件存储目录 /var/vmware/nsx/file-store 中。您可以使用 CLI file-store 命令将包文件复制到远程站点中。

  ```
  get node-agent-log file <filename>
  get node-agent-log file <filename> <node-name>
  ```

  示例：

  ```
  kubenode>get node-agent-log file foo
  Log file foo created in tgz format
  ```

- 获取并设置日志级别

  可用的日志级别为 NOTSET、DEBUG、INFO、WARNING、ERROR 和 CRITICAL。

  ```
  get ncp-log-level
  set ncp-log-level <log level>
  ```

  示例：

  ```
  kubenode>get ncp-log-level
  NCP log level is INFO

  kubenode>set ncp-log-level DEBUG
  NCP log level is changed to DEBUG
  ```

用于 NSX 节点代理容器的状态命令

- 显示节点代理和该节点上的 HyperBus 之间的连接状态。

  ```
  get node-agent-hyperbus status
  ```

示例：

```
kubenode> get node-agent-hyperbus status
HyperBus status: Healthy
```

用于 NSX 节点代理容器的缓存命令

- 获取 NSX 节点代理容器的内部缓存。

```
get container-cache <container-name>
get container-caches
```

示例 1：

```
kubenode> get container-cache cif104
    ip: 192.168.0.14/32
    mac: 50:01:01:01:01:14
    gateway_ip: 169.254.1.254/16
    vlan_id: 104
```

示例 2：

```
kubenode> get container-caches
    cif104:
        ip: 192.168.0.14/32
        mac: 50:01:01:01:01:14
        gateway_ip: 169.254.1.254/16
        vlan_id: 104
```

用于 NSX Kube 代理容器的状态命令

- 显示 Kube 代理和 Kubernetes API 服务器之间的连接状态

```
get ncp-k8s-api-server status
```

示例：

```
kubenode> get kube-proxy-k8s-api-server status
Kubernetes ApiServer status: Healthy
```

- 显示 Kube 代理监视程序状态

```
get kube-proxy-watcher <watcher-name>
get kube-proxy-watchers
```

示例 1：

```
kubenode> get kube-proxy-watcher endpoint
    Average event processing time: 15 msec (in past 3600-sec window)
    Current watcher started time: May 01 2017 15:06:24 PDT
    Number of events processed: 90 (in past 3600-sec window)
    Total events processed by current watcher: 90
    Total events processed since watcher thread created: 90
    Total watcher recycle count: 0
    Watcher thread created time: May 01 2017 15:06:24 PDT
    Watcher thread status: Up
```

示例 2：

```
kubenode> get kube-proxy-watchers
    endpoint:
        Average event processing time: 15 msec (in past 3600-sec window)
        Current watcher started time: May 01 2017 15:06:24 PDT
        Number of events processed: 90 (in past 3600-sec window)
        Total events processed by current watcher: 90
        Total events processed since watcher thread created: 90
        Total watcher recycle count: 0
        Watcher thread created time: May 01 2017 15:06:24 PDT
        Watcher thread status: Up

     service:
        Average event processing time: 8 msec (in past 3600-sec window)
        Current watcher started time: May 01 2017 15:06:24 PDT
        Number of events processed: 2 (in past 3600-sec window)
        Total events processed by current watcher: 2
        Total events processed since watcher thread created: 2
        Total watcher recycle count: 0
        Watcher thread created time: May 01 2017 15:06:24 PDT
        Watcher thread status: Up
```

- 在节点上转储 OVS 流量

```
dump ovs-flows
```

示例：

```
kubenode> dump ovs-flows
    NXST_FLOW reply (xid=0x4):
    cookie=0x0, duration=8.876s, table=0, n_packets=0, n_bytes=0, idle_age=8, priority=100,ip
actions=ct(table=1)
    cookie=0x0, duration=8.898s, table=0, n_packets=0, n_bytes=0, idle_age=8, priority=0
actions=NORMAL
    cookie=0x0, duration=8.759s, table=1, n_packets=0, n_bytes=0, idle_age=8,
priority=100,tcp,nw_dst=10.96.0.1,tp_dst=443 actions=mod_tp_dst:443
    cookie=0x0, duration=8.719s, table=1, n_packets=0, n_bytes=0, idle_age=8,
priority=100,ip,nw_dst=10.96.0.10 actions=drop
    cookie=0x0, duration=8.819s, table=1, n_packets=0, n_bytes=0, idle_age=8,
```

```
priority=90,ip,in_port=1 actions=ct(table=2,nat)
    cookie=0x0, duration=8.799s, table=1, n_packets=0, n_bytes=0, idle_age=8, priority=80,ip
actions=NORMAL
    cookie=0x0, duration=8.856s, table=2, n_packets=0, n_bytes=0, idle_age=8, actions=NORMAL
```

错误代码

本部分列出了各种组件生成的错误代码。

NCP 错误代码

错误代码	说明
NCP00001	配置无效
NCP00002	初始化失败
NCP00003	状态无效
NCP00004	适配器无效
NCP00005	找不到证书
NCP00006	找不到令牌
NCP00007	NSX 配置无效
NCP00008	NSX 标记无效
NCP00009	NSX 连接失败
NCP00010	找不到节点标记
NCP00011	节点逻辑交换机端口无效
NCP00012	父 VIF 更新失败
NCP00013	VLAN 用尽
NCP00014	VLAN 释放失败
NCP00015	IP 池用尽
NCP00016	IP 释放失败
NCP00017	IP 块用尽
NCP00018	IP 子网创建失败
NCP00019	IP 子网删除失败
NCP00020	IP 池创建失败
NCP00021	IP 池删除失败
NCP00022	逻辑路由器创建失败
NCP00023	逻辑路由器更新失败
NCP00024	逻辑路由器删除失败
NCP00025	逻辑交换机创建失败

错误代码	说明
NCP00026	逻辑交换机更新失败
NCP00027	逻辑交换机删除失败
NCP00028	逻辑路由器端口创建失败
NCP00029	逻辑路由器端口删除失败
NCP00030	逻辑交换机端口创建失败
NCP00031	逻辑交换机端口更新失败
NCP00032	逻辑交换机端口删除失败
NCP00033	找不到网络策略
NCP00034	防火墙创建失败
NCP00035	防火墙读取失败
NCP00036	防火墙更新失败
NCP00037	防火墙删除失败
NCP00038	找到多个防火墙
NCP00039	NS 组创建失败
NCP00040	NS 组删除失败
NCP00041	IP 集创建失败
NCP00042	IP 集更新失败
NCP00043	IP 集删除失败
NCP00044	SNAT 规则创建失败
NCP00045	SNAT 规则删除失败
NCP00046	适配器 API 连接失败
NCP00047	适配器监控程序异常
NCP00048	负载平衡器服务删除失败
NCP00049	负载平衡器虚拟服务器创建失败
NCP00050	负载平衡器虚拟服务器更新失败

错误代码	说明
NCP00051	负载平衡器虚拟服务器删除失败
NCP00052	负载平衡器池创建失败
NCP00053	负载平衡器池更新失败
NCP00054	负载平衡器池删除失败
NCP00055	负载平衡器规则创建失败
NCP00056	负载平衡器规则更新失败
NCP00057	负载平衡器规则删除失败
NCP00058	负载平衡器池 IP 释放失败

错误代码	说明
NCP00059	找不到负载平衡器虚拟服务器和服务关联
NCP00060	NS 组更新失败
NCP00061	防火墙规则获取失败
NCP00062	NS 组没有标准
NCP00063	找不到节点虚拟机
NCP00064	找不到节点 VIF
NCP00065	证书导入失败
NCP00066	证书取消导入失败
NCP00067	SSL 绑定更新失败
NCP00068	未找到 SSL 配置文件
NCP00069	找不到 IP 池
NCP00070	找不到 T0 Edge 群集
NCP00071	IP 池更新失败
NCP00072	调度程序失败
NCP00073	NAT 规则删除失败
NCP00074	逻辑路由器端口获取失败
NCP00075	NSX 配置验证失败

错误代码	说明
NCP00076	SNAT 规则更新失败
NCP00077	SNAT 规则重叠
NCP00078	负载平衡器端点添加失败
NCP00079	负载平衡器端点更新失败
NCP00080	负载平衡器规则池创建失败
NCP00081	找不到负载平衡器虚拟服务器
NCP00082	IP 集读取失败
NCP00083	SNAT 池获取失败
NCP00084	负载平衡器服务创建失败
NCP00085	负载平衡器服务更新失败
NCP00086	逻辑路由器端口更新失败
NCP00087	负载平衡器初始化失败
NCP00088	IP 池不唯一
NCP00089	第 7 层负载平衡器缓存同步错误
NCP00090	负载平衡器池不存在错误
NCP00091	负载平衡器规则缓存初始化错误

错误代码	说明
NCP00092	SNAT 进程失败
NCP00093	负载平衡器默认证书错误
NCP00094	负载平衡器端点删除失败
NCP00095	找不到项目
NCP00096	池访问被拒绝
NCP00097	无法获取负载平衡器服务
NCP00098	无法创建负载平衡器服务
NCP00099	负载平衡器池缓存同步错误

NSX 节点代理错误代码

错误代码	说明
NCP01001	找不到 OVS 上行链路
NCP01002	找不到主机 MAC
NCP01003	OVS 端口创建失败
NCP01004	无任何 pod 配置
NCP01005	Pod 配置失败
NCP01006	Pod 取消配置失败
NCP01007	找不到 CNI 套接字
NCP01008	CNI 连接失败
NCP01009	CNI 版本不匹配
NCP01010	CNI 消息接收失败
NCP01011	CNI 消息传输失败
NCP01012	Hyperbus 连接失败
NCP01013	Hyperbus 版本不匹配
NCP01014	Hyperbus 消息接收失败
NCP01015	Hyperbus 消息传输失败
NCP01016	GARP 发送失败
NCP01017	接口配置失败

nsx-kube-proxy 错误代码

错误代码	说明
NCP02001	代理网关端口无效
NCP02002	代理命令失败
NCP02003	代理验证失败

CLI 错误代码

错误代码	说明
NCP03001	CLI 启动失败
NCP03002	CLI 套接字创建失败
NCP03003	CLI 套接字异常
NCP03004	CLI 客户端请求无效
NCP03005	CLI 服务器传输失败
NCP03006	CLI 服务器接收失败
NCP03007	CLI 命令执行失败

Kubernetes 错误代码

错误代码	说明
NCP05001	Kubernetes 连接失败
NCP05002	Kubernetes 配置无效
NCP05003	Kubernetes 请求失败
NCP05004	找不到 Kubernetes 密钥
NCP05005	找不到 Kubernetes 类型
NCP05006	Kubernetes 监控程序异常
NCP05007	Kubernetes 资源长度无效
NCP05008	Kubernetes 资源类型无效
NCP05009	Kubernetes 资源句柄失败
NCP05010	Kubernetes 服务句柄失败
NCP05011	Kubernetes 端点句柄失败
NCP05012	Kubernetes Ingress 句柄失败
NCP05013	Kubernetes 网络策略句柄失败
NCP05014	Kubernetes 节点句柄失败
NCP05015	Kubernetes 命名空间句柄失败

错误代码	说明
NCP05016	Kubernetes pod 句柄失败
NCP05017	Kubernetes 密钥句柄失败
NCP05018	Kubernetes 默认后端失败
NCP05019	Kubernetes 匹配表达式不受支持
NCP05020	Kubernetes 状态更新失败
NCP05021	Kubernetes 注释更新失败
NCP05022	找不到 Kubernetes 命名空间缓存
NCP05023	找不到 Kubernetes 密钥
NCP05024	Kubernetes 默认后端正在使用中
NCP05025	Kubernetes LoadBalancer 服务句柄失败

OpenShift 错误代码

错误代码	说明
NCP07001	OC 路由句柄失败
NCP07002	OC 路由状态更新失败