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

VMware NSX Container Plug-in 2.3、2.3.1、2.3.2
VMware NSX-T Data Center 2.3
VMware NSX-T Data Center 2.3.1

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

软件产品	版本
NSX-T Data Center	2.2、2.3
容器主机虚拟机的管理程序	■ 支持的 vSphere 版本 ■ RHEL KVM 7.4、7.5
容器主机操作系统	RHEL 7.4、7.5
平台即服务	OpenShift 3.9、3.10
容器主机 Open vSwitch	2.9.1（随 NSX-T 2.3 和 2.2 附带）

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 创建一个第 0 层逻辑路由器。

5 为 pod 创建 IP 块。

6 为 SNAT（源网络地址转换）创建 IP 池。

7 部署 OpenShift 虚拟机。

8 准备 Ansible hosts 文件。

9 （选项 1）使用单个 playbook 安装 NCP 和 OpenShift。

（选项 2）安装 CNI 插件、OVS (Open vSwitch) 和 NCP Docker 映像。然后安装 OpenShift 容器平台。

10 运行 NCP 和 NSX 节点代理。

如果使用提供的 playbook 安装 NCP，则不需要执行步骤 2 至 6。请参见使用单个 Playbook 安装 NCP 和 OpenShift 和安装 CNI 插件、OVS 和 NCP Docker 映像。

升级 NCP

要将 NCP 升级到 2.3.0，请执行以下步骤。

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

2 （可选）将 NSX-T Data Center 升级到 2.3。

NCP 2.3.0 支持 NSX-T 2.2，但也可以升级到 NSX-T Data Center 2.3。

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。您可以使用 NSX Manager GUI 手动配置这些资源，或者使用 Ansible playbook 自动完成该过程。

本节介绍了如何手动创建 NSX-T 资源。要自动完成该过程，请参阅安装 CNI 插件、OVS 和 NCP Docker 映像。

本章讨论了以下主题：

- 配置 NSX-T 资源
- 创建和配置第 0 层逻辑路由器

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。

可以在配置文件 `ncp.ini` 中使用 UUID 和名称配置 NSX-T Data Center 资源。

覆盖网络传输区域

登录到 NSX Manager 并导航到**结构层 > 传输区域**。查找用于容器网络的覆盖网络传输区域，或者创建新的传输区域。

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，然后导航到**网络 > 路由 > 路由器**。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。您可以通过登录到 NSX Manager 并导航到**网络 > 交换 > 交换机**来创建逻辑交换机。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，然后导航到**网络 > IPAM** 以创建一个或多个 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，然后导航到**清单 > 组 > IP 池**以创建池或查找现有池。

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。从 NCP 2.3.1 开始，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，导航到**安全 > 分布式防火墙 > 常规**并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。

创建和配置第 0 层逻辑路由器

第 0 层逻辑路由器将 Kubernetes 节点连接到外部网络。

步骤

1. 从浏览器中，登录到 https://*NSX Manager IP 地址* 中的 NSX Manager。
2. 导航到**网络 > 路由 > 路由器**，然后单击**添加 > 第 0 层路由器**。
3. 输入名称和可选的说明。
4. 从下拉菜单中选择一个现有的 Edge 群集以支持该第 0 层逻辑路由器。

5 选择一种高可用性模式。

选择活动-备用。

6 单击**保存**。

新的逻辑路由器将显示为一个链接。

7 单击逻辑路由器链接。

8 单击**路由 > 路由重新分发**。

9 单击**添加**以添加新的重新分发条件。

对于源，在路由的（非 NAT）拓扑中选择 **NSX 静态**。在 NAT 拓扑中，选择**第 0 层 NAT**。

10 单击**保存**。

11 单击新创建的路由器。

12 单击**配置 > 路由器端口**。

13 单击**添加**以添加一个上行链路端口。

14 选择一个传输节点。

15 选择以前创建的逻辑交换机。

16 在外部网络中指定一个 IP 地址。

17 单击**保存**。

新的逻辑路由器将显示为一个链接。

在 OpenShift 环境中安装 NCP

3

本章介绍了如何安装和配置 NSX-T Container Plug-in (NCP) 和 OpenShift。

本章讨论了以下主题：

- 部署 OpenShift 虚拟机
- 准备 Ansible hosts 文件
- 使用单个 Playbook 安装 NCP 和 OpenShift
- 安装 CNI 插件、OVS 和 NCP Docker 映像
- 安装 OpenShift 容器平台
- 运行 NCP 和 NSX 节点代理
- 设置说明

部署 OpenShift 虚拟机

在安装 NSX-T Container Plug-in 之前，必须安装 OpenShift。您必须部署至少一个主节点。

有关详细信息，请参见 https://docs.openshift.com。

后续步骤

准备 Ansible hosts 文件。请参见准备 Ansible hosts 文件。

准备 Ansible hosts 文件

Ansible `hosts` 文件定义 OpenShift 群集中的节点。

步骤

1 克隆 https://github.com/vmware/nsx-integration-for-openshift 中的 NCP GitHub 存储库。`hosts` 文件位于 `openshift-ansible-nsx` 目录中。您必须将 `hosts` 文件保留在 `openshift-ansible-nsx` 目录中。某些 playbook 会认为这是 `hosts` 文件的路径。

2 在 [masters] 和 [nodes] 部分中，指定 OpenShift 虚拟机的主机名和 IP 地址。例如，

```
    [masters]
    admin.rhel.osmaster ansible_ssh_host=101.101.101.4

    [single_master]
    admin.rhel.osmaster ansible_ssh_host=101.101.101.4

    [nodes]
    admin.rhel.osmaster ansible_ssh_host=101.101.101.4 openshift_ip=101.101.101.4
openshift_schedulable=true openshift_hostname=admin.rhel.osmaster
    admin.rhel.osnode ansible_ssh_host=101.101.101.5 openshift_ip=101.101.101.5
openshift_hostname=admin.rhel.osnode

    [etcd]

    [OSEv3:children]
    masters
    nodes
    etcd
```

请注意，`openshift_ip` 指定群集内部 IP，如果要使用的接口不是默认接口，则需要设置该值。主节点中的 ncp 相关角色使用 `single_master` 变量以仅执行一次某些任务，例如，NSX-T Data Center 管理层面资源配置。

3 设置 SSH 访问，以便从运行 Ansible 角色的节点（通常为主节点）中访问所有节点，而不使用密码：

```
    ssh-keygen
    ssh-copy-id -i ~/.ssh/id_rsa.pub root@admin.rhel.osnode
```

4 更新 [OSEv3:vars] 部分。可以在 OpenShift 容器平台文档网站的“高级安装”中找到所有参数的详细信息（在 https://docs.openshift.com 中搜索“高级安装”）。例如，

```
    # Set the default route fqdn
    openshift_master_default_subdomain=apps.corp.local

    os_sdn_network_plugin_name=cni
    openshift_use_openshift_sdn=false
    openshift_node_sdn_mtu=1500

    # If ansible_ssh_user is not root, ansible_become must be set to true
    ansible_become=true

    openshift_master_default_subdomain
      This is the default subdomain used in the OpenShift routes for External LB

    os_sdn_network_plugin_name
      Set to 'cni' for the NSX Integration

    openshift_use_openshift_sdn
      Set to false to disable the built-in OpenShift SDN solution

    openshift_hosted_manage_router
```

```
      Set to false to disable creation of router during installation. The router has to be manually
started after NCP and nsx-node-agent are running.

    openshift_hosted_manage_registry
      Set to false to disable creation of registry during installation. The registry has to be
manually started after NCP and nsx-node-agent are running.

    deployment_type
      Set to openshift-enterprise

    openshift_hosted_manage_registry
      Set to false to disable auto creation of registry

    openshift_hosted_manage_router
      Set to false to disable auto creation of router

    openshift_enable_service_catalog
      Set to false to disable service_catalog

    (For OpenShift 3.9 only) skip_sanity_checks
      Set to true

    (For OpenShift 3.9 only) openshift_web_console_install
      Set to false
```

5 检查您是否具有到所有主机的连接：

```
ansible OSEv3 -i /PATH/TO/HOSTS/hosts -m ping
```

结果应类似于以下内容。如果没有，请解决连接问题。

```
openshift-node1 | SUCCESS => {
   "changed": false,
   "ping": "pong"
}
openshift-master | SUCCESS => {
   "changed": false,
   "ping": "pong"
}
```

后续步骤

安装 CNI 插件和 OVS。请参见安装 CNI 插件、OVS 和 NCP Docker 映像。

使用单个 Playbook 安装 NCP 和 OpenShift

可以使用单个 playbook 安装 NCP 和 OpenShift，也可以单独执行安装。

单个 Ansible playbook `install.yaml` 执行以下任务：

- NCP 准备
- OpenShift 安装

- NCP 安装

或者，也可以按照以下两部分中的说明安装 NCP 和 OpenShift：安装 CNI 插件、OVS 和 NCP Docker 映像和安装 OpenShift 容器平台。

运行 `install.yaml` playbook 之前，请为 `ncp_prep` 和 `ncp` plabook 角色设置必选和可选参数。安装 CNI 插件、OVS 和 NCP Docker 映像中介绍了这些参数。

以下命令运行 playbook：

```
ansible-playbook -i /PATH/TO/HOSTS/hosts install.yaml
```

安装 CNI 插件、OVS 和 NCP Docker 映像

容器网络接口 (CNI) 插件、Open vSwitch (OVS) 和 NCP Docker 映像必须安装在 OpenShift 节点上。安装是通过运行 Ansible playbook 执行的。

注 如果使用单个 playbook 安装 NCP 和 OpenShift，则不需要执行此步骤。请参见使用单个 Playbook 安装 NCP 和 OpenShift。

playbook 包含为节点配置 NSX-T 资源的说明。您也可以手动配置 NSX-T Data Center 资源，如第 2 章，设置 NSX-T 资源中所述。`perform_nsx_config` 参数指示在运行 playbook 时是否配置资源。

步骤

1 更新 `roles/ncp_prep/default/main.yaml` 和 `roles/nsx_config/default/main.yaml` 中的参数值，包括可从中下载 CNI 插件 RPM、OVS 及其相应的内核模块 RPM 的 URL。此外，`uplink_port` 是节点虚拟机上的上行链路端口 VNIC 的名称。其余变量与 NSX-T Data Center 管理层面配置有关。

需要指定的参数：

- `perform_nsx_config`：是否执行资源配置。如果手动完成配置，请将其设置为 false，将不会运行 `nsx_config` 脚本。
- `nsx_manager_ip`：NSX Manager 的 IP
- `nsx_edge_cluster_name`：第 0 层路由器将使用的 Edge 群集的名称
- `nsx_transport_zone_name`：覆盖网络传输区域的名称
- `os_node_name_list`：以逗号分隔的节点名称列表

  例如 node1,node2,node3
- `subnet_cidr`：管理员将分配给节点上的 br-int 的 IP CIDR 地址
- `vc_host`：vCenter Server 的 IP 地址
- `vc_user`：vCenter Server 管理员的用户名
- `vc_password`：vCenter Server 管理员的密码
- `vms`：以逗号分隔的虚拟机名称列表。顺序必须与 `os_node_name_list` 匹配。

以下参数具有默认值。您可以根据需要进行修改。

- `nsx_t0_router_name`：群集的第 0 层逻辑路由器的名称。默认值：**`t0`**
- `pod_ipblock_name`：pod 的 IP 块的名称。默认值：**`podIPBlock`**
- `pod_ipblock_cidr`：该 IP 块的 CIDR 地址。默认值：**`172.20.0.0/16`**
- `snat_ippool_name`：SNAT 的 IP 块的名称。默认值为 `externalIP`。
- `snat_ippool_cidr`：该 IP 块的 CIDR 地址。默认值：**`172.30.0.0/16`**
- `start_range`：为此 IP 池指定的 CIDR 起始 IP 地址。默认值：**`172.30.0.1`**
- `end_range`：为此 IP 池指定的 CIDR 结束 IP 地址。默认值：**`172.30.255.254`**
- `os_cluster_name`：OpenShift 群集的名称。默认值：**`occl-one`**
- `nsx_node_ls_name`：连接到节点的逻辑交换机的名称。默认值：**`node_ls`**
- `nsx_node_lr_name`：交换机 **`node_ls`** 的逻辑路由器的名称。默认值：**`node_lr`**

`nsx-config` playbook 仅支持创建一个 IP 池和一个 IP 块。如果需要更多，则必须手动创建。

2 转到 `openshift-ansible-nsx` 目录并运行 `ncp_prep` 角色。

```
ansible-playbook -i /PATH/TO/HOSTS/hosts ncp_prep.yaml
```

playbook 包含执行以下操作的说明：

- 下载 CNI 插件安装文件。

  文件名是 `nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm`，其中 *xxxxxxx* 是内部版本号。

- 安装 CNI 插件安装文件。

  该插件安装在 `/opt/cni/bin` 中。CNI 配置文件 10.net.conf 将复制到 `/etc/cni/net.d` 中。该 rpm 还会为环回插件安装 `/etc/cni/net.d/99-loopback.conf` 配置文件。

- 下载并安装 OVS 安装文件。

  这些文件为 `openvswitch-2.9.1.xxxxxxx-1.x86_64.rpm` 和 `openvswitch-kmod-2.9.1.xxxxxxx-1.el7.x86_64.rpm`，其中 *xxxxxxx* 是内部版本号。

- 如果尚未创建 *br-int* 实例，请创建该实例。

```
# ovs-vsctl add-br br-int
```

- 将连接到节点逻辑交换机的网络接口 (*node-if*) 添加到 *br-int* 中。
- 确保 *br-int* 和 *node-if link* 状态为 up。

```
# ip link set br-int up
# ip link set <node-if> up
```

- 更新网络配置文件，以确保网络接口在重新引导后处于已启动状态。

- 下载 NCP tar 文件并从 tar 文件加载 Docker 映像。
- 下载 `ncp-rbac` yaml 文件并将 `apiVersion` 更改为 **v1**。
- 在 NSX-T Data Center 中创建逻辑拓扑和相关资源并对其创建标记，以便它们可由 NCP 识别。
- 使用 NSX-T Data Center 资源信息更新 `ncp.ini`。

后续步骤

安装 OpenShift 容器平台。请参见安装 OpenShift 容器平台。

安装 OpenShift 容器平台

OpenShift 容器平台 (OCP) 是一个平台即服务 (PaaS) 产品，该产品汇集了 Docker 和 Kubernetes。

注 如果使用单个 playbook 安装 NCP 和 OpenShift，则不需要执行此步骤。请参见使用单个 Playbook 安装 NCP 和 OpenShift。

有关安装 OCP 的信息，请参见 https://docs.openshift.com。

后续步骤

运行 NCP 和 NSX 节点代理。请参见运行 NCP 和 NSX 节点代理。

运行 NCP 和 NSX 节点代理

设置和运行 NCP 和 NSX 节点代理。

步骤

1 编辑 `roles/ncp/defaults/main.yaml`，并指定 OpenShift API 服务器 IP、NSX Manager IP 以及 NCP ReplicationController yaml 和 nsx-node-agent DaemonSet yaml 的下载 URL。

2 从 openshift-ansible-nsx 目录中，运行 ncp 角色：

```
ansible-playbook -i /PATH/TO/HOSTS/hosts ncp.yaml
```

ncp 角色执行以下步骤：

- 检查 nsx-system 项目是否存在；如果不存在，则创建一个项目。

```
oc new-project nsx-system
```

- 下载 `ncp-rbac` yaml 文件并将 `apiVersion` 更改为 **v1**。
- 创建 NCP pod 的服务帐户，然后创建一个群集角色指定 NCP 可以访问的资源，并将群集角色绑定到 NCP 服务帐户。

- 创建 nsx-node-agent pod 的服务帐户，然后创建一个群集角色指定节点代理可以访问的资源，并将群集角色绑定到节点代理服务帐户。

```
    oc apply -f /tmp/ncp-rbac.yml
```

- 获取与上述服务帐户关联的令牌，并将其存储在 /etc/nsx-ujo/<service_account>_token 中。

```
    secret=`kubectl get serviceaccount ncp-svc-account -o yaml | grep -A1 secrets | tail -n1 | awk {'print $3'}`
    kubectl get secret $secret -o yaml | grep 'token:' | awk {'print $2'} | base64 -d > /etc/nsx-ujo/ncp_token
    secret=`kubectl get serviceaccount nsx-node-agent-svc-account -o yaml | grep -A1 secrets | tail -n1 | awk {'print $3'}`
    kubectl get secret $secret -o yaml | grep 'token:' | awk {'print $2'} | base64 -d > /etc/nsx-ujo/node_agent_token
```

- 为 NCP 下载 SecurityContextConstraint (SCC) yaml 文件 ncp-os-scc.yml，并根据该 yaml 创建 SCC。

```
    oc apply -f ncp-os-scc.yml
```

  该 SCC yaml 文件将 SELinux 类型指定为 spc_t，以确保 NCP/nsx-node-agent 在 SELinux 下具有访问权限。即，

```
    seLinuxContext:
      seLinuxOptions:
      type: spc_t
```

  在 SCC yaml 文件中的 seLinuxContext 的 seLinuxOptions 下，基于标签的 SELinux 访问控制级别设置为 s0:c0:c1023，以允许 ncp/node-agent 访问不同文件类别的目标。

- 将 SCC 添加到创建 NCP 和 NSX 节点代理 pod 的用户。例如，将 SCC 添加到当前项目中的默认用户：

```
    oc adm policy add-scc-to-user ncp-scc -z default
```

- 将 SCC 添加到 NCP 和 NSX 节点代理服务帐户：

```
    oc adm policy add-scc-to-user ncp-scc -z ncp-svc-account
    oc adm policy add-scc-to-user ncp-scc -z nsx-node-agent-svc-account
```

- 为 NCP ReplicationController (RC) 和 nsx-node-agent DaemonSet (DS) 下载 yaml 文件并更新 ConfigMap。
- 下载并加载 NCP 映像（nsx-node-agent 使用相同的映像）。
- 配置该服务帐户并为 NCP 和 nsx_node_agent 设置所需的 SecurityContextConstraint。

- 创建 NCP ReplicationController 和 nsx-node-agent DaemonSet。

注 NCP 打开与 Kubernetes API 服务器的持续 HTTP 连接，以监视 Kubernetes 资源的生命周期事件。如果 API 服务器故障或网络故障导致 NCP 的 TCP 连接失效，则必须重新启动 NCP，以便其可以重新建立与 API 服务器的连接。否则，NCP 将错过新事件。

设置说明

在设置 OpenShift 和 NCP 之前，请注意以下事项。

- pod 不能包含超过 11 个标签，而命名空间不能包含超过 12 个标签。
- NCP 将忽略为 OpenShift 内部使用而添加的标签（例如，在其键中具有前缀 openshift.io 的标签），因此，用户看不到在相关的 NSX 资源上创建的相应标记。此处是 OpenShift 使用的标签前缀列表，您应该避免使用以任何以下内容开头的标签键：

```
openshift.io
pod-template
```

- 节点需要访问 pod，例如，进行 Kubelet 运行状况检查。请确保主机管理接口能够访问 pod 网络。
- 攻击者可能会利用 Linux 功能 NET_ADMIN 和 NET_RAW 以破坏 pod 网络。您应该禁用不受信任的容器的这两个功能。默认情况下，对于受限制和 anyuid SCC，不会授予 NET_ADMIN。请注意明确启用 NET_ADMIN 或允许 pod 在特权模式下运行的任何 SCC。此外，对于不受信任的容器，请根据某些内容（如 anyuid SCC）创建一个单独的 SCC 并移除 NET_RAW 功能。可以将 NET_RAW 添加到 SCC 定义中的“requiredDropCapabilities”列表以完成该操作。
- 允许在 pod/容器中进行 root 访问（仅用于测试）。下面的命令需要在当前登录到的 oc 项目的所有 pod 中具有 root 访问权限。

```
oc new-project test-project
oc project test-project
oc adm policy add-scc-to-user anyuid -z default
```

- 配置（添加）OpenShift 注册表。

```
oc login -u system:admin -n default
oc adm registry --service-account=registry --config=/etc/origin/master/admin.kubeconfig
```

- 删除 OpenShift 注册表

```
oc login -u system:admin -n default
oc delete svc/docker-registry dc/docker-registry
```

- 缺少 IPtables 防火墙规则以允许将 DNS 请求从 Docker 默认网桥容器发送到节点上的 dnsmasq 进程。需要手动将其打开。编辑 /etc/sysconfig/iptables，并在该文件末尾的 COMMIT 前面添加以下规则：

```
-A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A OS_FIREWALL_ALLOW -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
COMMIT
```

- 重新启动 iptables、docker 和 origin-node（重新启动 kube 代理和 kubelet）。

```
systemctl restart iptables
systemctl restart docker
systemctl restart origin-node
```

- 需要允许 OpenShift 的内部 docker 注册表使用非 TLS，以使 OpenShift 正常工作。通常，这是 OpenShift Ansible 安装程序自动添加的，但似乎它当前无法正常工作。编辑 /etc/sysconfig/docker 并添加：

```
INSECURE_REGISTRY='--insecure-registry 172.30.0.0/16'
```

- 重新启动 Docker。

```
systemctl restart docker
```

- NCP 的网络策略支持与 Kubernetes 所提供的支持一样，并且取决于 OpenShift 使用的 Kubernetes 版本。
  - OpenShift 3.9 - 网络策略中的规则子句最多只能包含 `namespaceSelector`、`podSelector` 和 `ipBlock` 中的一个选择器。
- Kubernetes API 服务器不对网络策略规范进行验证。可以创建无效的网络策略。NCP 将拒绝此类网络策略。如果更新网络策略以使其有效，NCP 仍不会处理该网络策略。必须删除该网络策略，然后使用有效的规范重新创建一个。
- 某些版本的 Kubernetes 具有与 `subPath` 相关的问题（请参见 https://github.com/kubernetes/kubernetes/issues/61076）。如果 OpenShift 版本不包含此问题的修复，NCP pod 创建将失败并显示错误“ `CreateContainerConfigError: 无法为 volumeMount 准备 subPath (CreateContainerConfigError: failed to prepare subPath for volumeMount)`”。通过从 NCP yaml 中移除 `subPath` 可以解决此问题。具体来说，移除包含 `subPath: ncp.ini` 的行并将 `volumes` 的配置替换为以下内容：

```
volumes:
  - name: config-volume
    # ConfigMap nsx-ncp-config is expected to supply ncp.ini
    configMap:
      name: nsx-ncp-config
      items:
        - key: ncp.ini
          path: ncp.ini
```

  此更改的副作用是整个 /etc/nsx-ujo 目录将变成只读目录。其结果是，使用证书和私钥将无法与 NSX-T 建立连接，因为 NCP 无法在 /etc/nsx-ujo 下创建临时文件，因此无法将证书和私钥移到单个文件中。

- 如果正在运行或升级到 OpenShift 3.10 群集，请注意以下事项：
  - 必须指定特定于 OpenShift 3.10 群集的节点组的配置。必须在清单主机文件中提供节点 configmap 配置。
  - 在清单主机文件的 `[nodes]` 组中定义的所有主机都必须分配给一个节点组名称。
  - 从 Ansible playbook 升级 OpenShift 群集可能会导致网络丢失。请确保在 openshift-ansible 存储库上添加修补程序 (https://github.com/openshift/openshift-ansible/pull/8016/files#diff-2386e21861da3f95091dbb27d72ca366) 以移除 Open vSwitch 软件包的停止/卸载操作。
- 从 OpenShift 3.10 开始，kube-proxy 已从 openshift-node 服务移至 DaemonSet。它不再默认启动。请执行以下步骤手动启动 kube-proxy（假定已克隆 openshift-ansible 存储库）：
  - 转到 `openshift ansible` 目录，在 `[defaults]` 下，设置以下内容：

    ```
    library = roles/lib_utils/library/
    ```

  - 在 playbooks 目录中创建 create_proxy.yaml 文件并在其中包含以下条目：

    ```
    - import_playbook: byo/openshift_facts.yml
    - hosts: masters
      run_once: True
      roles:
        - kube_proxy_and_dns
    ```

  - 运行 playbook：

    ```
    ansible-playbook -i hosts playbooks/create_proxy.yaml
    ```

    您将看到指示某些操作失败的错误消息。可以忽略这些消息。可以通过运行命令 `oc get po --all-namespaces` 验证结果。

在裸机环境中安装 NCP

4

在裸机环境中安装 NSX-T Container Plug-in (NCP) 的步骤与在非裸机环境中安装 NCP 的步骤类似。本节介绍不同的步骤。

本章讨论了以下主题：

- 安装 NSX-T Data Center CNI 插件
- 为 OpenShift 节点配置 NSX-T Data Center 网络
- 安装 NSX 节点代理
- nsx-node-agent-ds.yml 中的 ncp.ini 的 ConfigMap
- 安装 NSX-T Container Plug-in
- ncp-rc.yml 中的 ncp.ini 的 ConfigMap

安装 NSX-T Data Center CNI 插件

NSX-T Data Center CNI 插件必须安装在 OpenShift 节点上。

步骤

1 下载适用于您的 Linux 发布版本的安装文件。

文件名是 `nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm`，其中 *xxxxxxx* 是内部版本号。

2 安装在步骤 1 中下载的 rpm 文件。

该插件安装在 `/opt/cni/bin` 中。CNI 配置文件 10.net.conf 将复制到 `/etc/cni/net.d` 中。该 rpm 还会为环回插件安装 `/etc/cni/net.d/99-loopback.conf` 配置文件。

为 OpenShift 节点配置 NSX-T Data Center 网络

本节介绍了如何为 OpenShift 主节点和计算节点配置 NSX-T Data Center 网络。

每个节点必须向 NSX Manager 注册为操作系统类型 `RHEL Container`。节点的管理接口可用于加入 OpenShift 群集，可以位于 NSX-T Data Center 结构层，也可以不位于 NSX-T 结构层。其他接口为容器提供网络连接，并且必须位于 NSX-T Data Center 结构层。

相应的传输节点必须具有以下标记：

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

您可以通过从 NSX Manager GUI 导航到**结构层 > 节点**来标识 OpenShift 节点的传输节点。

如果 OpenShift 节点名称发生更改，您必须更新 `ncp/node_name` 标记并重新启动 NCP。您可以使用以下命令获取节点名称：

```
oc get nodes
```

如果在运行 NCP 时将节点添加到群集中，您必须在运行 `oc cluster add` 命令之前将标记添加到传输节点。否则，新节点将没有网络连接。如果标记不正确或丢失，您可以执行以下步骤以解决该问题：

- 将正确标记应用于传输节点。
- 重新启动 NCP。

安装 NSX 节点代理

NSX 节点代理是一个 DaemonSet，每个 pod 将在其中运行两个容器。一个容器运行 NSX 节点代理，其主要职责是管理容器网络接口。它与 CNI 插件和 Kubernetes API 服务器进行交互。另一个容器运行 NSX Kube 代理，其唯一的职责是将群集 IP 转换为 pod IP 以实施 Kubernetes 服务抽象。它实施与上游 Kube 代理相同的功能。

步骤

1 下载 NCP Docker 映像。

文件名是 `nsx-ncp-xxxxxxx.tar`，其中 *xxxxxxx* 是内部版本号。

2 下载 NSX 节点代理 DaemonSet yaml 模板。

文件名是 `nsx-node-agent-ds.yml`。您可以编辑该文件，或者将其作为您自己的模板文件的示例。

3 将 NCP Docker 映像加载到您的映像注册表中。

```
docker load -i <tar file>
```

4 编辑 `nsx-node-agent-ds.yml`。

将映像名称更改为加载的映像。

进行以下更改：

```
[coe]
node_type = 'BAREMETAL'
...
[nsx_node_agent]
ovs_bridge = 'nsx-managed'
```

取消以下几行的注释：

```
securityContext:
  capabilities:
    add:
      - NET_ADMIN
      - SYS_ADMIN
      - SYS_PTRACE
      - DAC_READ_SEARCH
      # For BMC usecase
      - DAC_OVERRIDE
volumeMounts:
…
# mount nestdb-sock for baremetal node
- name: nestdb-sock
mountPath: /var/run/vmware/nestdb/nestdb-server.sock
volumes:
…
# volume for baremetal node
- name: nestdb-sock
   hostPath:
   path: /var/run/vmware/nestdb/nestdb-server.sock
   type: Socket
```

注 在该 yaml 文件中，您必须指定为 `ncp.ini` 生成的 ConfigMap 必须挂载为只读卷。下载的 yaml 文件已具有该规范，不应对其进行更改。

5 使用以下命令创建 NSX 节点代理 DaemonSet。

```
oc apply -f nsx-node-agent-ds.yml
```

nsx-node-agent-ds.yml 中的 ncp.ini 的 ConfigMap

示例 yaml 文件 `nsx-node-agent-ds.yml` 包含 NSX 节点代理的配置文件 `ncp.ini` 的 ConfigMap。该 ConfigMap 部分包含一些参数，您可以指定这些参数以自定义节点代理安装。

您下载的示例 `nsx-node-agent-ds.yml` 包含以下 `ncp.ini` 信息：

```
# ConfigMap for ncp.ini
apiVersion: v1
kind: ConfigMap
metadata:
  name: nsx-node-agent-config
  labels:
    version: v1
data:
  ncp.ini: |
    [DEFAULT]

    # Set to True to enable logging to stderr
    #use_stderr = True
```

```
# Set to True to send logs to the syslog daemon
#use_syslog = False
# Enabler debug-level logging for the root logger. If set to True, the
# root logger debug level will be DEBUG, otherwise it will be INFO.
#debug = True

# The log file path must be set to something like '/var/log/nsx-ujo/'. By
# default, logging to file is disabled.
#log_dir = None

# Name of log file to send logging output to. If log_dir is set but log_file is
# not, the binary name will be used, i.e., ncp.log, nsx_node_agent.log and
# nsx_kube_proxy.log.
#log_file = None

# max MB for each compressed file. Defaults to 100 MB
#log_rotation_file_max_mb = 100

# Total number of compressed backup files to store. Defaults to 5.
#log_rotation_backup_count = 5
[coe]
#
# Common options for Container Orchestrators
#

# Container orchestrator adaptor to plug in
# Options: kubernetes (default), openshift, pcf.
#adaptor = kubernetes

# Specify cluster for adaptor. It is a prefix of NSX resources name to
# distinguish multiple clusters who are using the same NSX.
# This is also used as the tag of IP blocks for cluster to allocate
# IP addresses. Different clusters should have different IP blocks.
#cluster = k8scluster

# Log level for the NCP operations. If set, overrides the level specified
# for the root logger. Possible values are NOTSET, DEBUG, INFO, WARNING,
# ERROR, CRITICAL
#loglevel=None

# Log level for the NSX API client operations. If set, overrides the level
# specified for the root logger. Possible values are NOTSET, DEBUG, INFO,
# WARNING, ERROR, CRITICAL
nsxlib_loglevel=INFO

# Once enabled, all projects in this cluster will be mapped to a NAT
# topology in NSX backend
#enable_snat = True

# The type of container node. Possible values are HOSTVM, BAREMETAL.
node_type = BAREMETAL

[ha]
#
# NCP High Availability configuration options
```

```
#

# Time duration in seconds of mastership timeout. NCP instance will
# remain master for this duration after elected. Note that the heartbeat
# period plus the update timeout must be less than this period. This
# is done to ensure that the master instance will either confirm
# liveness or fail before the timeout.
#master_timeout = 9

# Time in seconds between heartbeats for elected leader. Once an NCP
# instance is elected master, it will periodically confirm liveness based
# on this value.
#heartbeat_period = 3

# Timeout duration in seconds for update to election resource. If the
# update request does not complete before the timeout it will be
# aborted. Used for master heartbeats to ensure that the update finishes
# or is aborted before the master timeout occurs.
#update_timeout = 3

[k8s]
#
# From kubernetes
#

# IP address of the Kubernetes API Server. If not set, will try to
# read and use the Kubernetes Service IP from environment variable
# KUBERNETES_SERVICE_HOST.
#apiserver_host_ip = <ip_address>

# Port of the Kubernetes API Server.
# Set to 6443 for https. If not set, will try to
# read and use the Kubernetes Service port from environment
# variable KUBERNETES_SERVICE_PORT.
#apiserver_host_port = <port>

# Specify a CA bundle file to use in verifying the Kubernetes API server
# certificate. (string value)
#ca_file = <None>
ca_file = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

# Full path of the Token file to use for authenticating with the k8s API server.
#client_token_file = <None>
client_token_file = /var/run/secrets/kubernetes.io/serviceaccount/token

# Full path of the client certificate file to use for authenticating
# with the k8s API server. It must be specified together with
# "client_private_key_file"
#client_cert_file = <None>

# Full path of the client certificate file to use for authenticating
# with the k8s API server. It must be specified together with
# "client_cert_file"
#client_private_key_file = <None>
```

```
# Log level for the kubernetes adaptor. If set, overrides the level specified
# for the root logger. Possible values are NOTSET, DEBUG, INFO, WARNING,
# ERROR, CRITICAL
#loglevel=None

[nsx_node_agent]
#
# Configuration for nsx_node_agent
#

# Needs to mount node /proc to container if nsx_node_agent runs in a container.
# By default node /proc will be mounted to /host/proc, the prefix is /host.
# It should be the same setting with mounted path in the daemonset yaml file.
# Set the path to '' if nsx_node_agent is running as a process in minion node.
#proc_mount_path_prefix = /host

# The OVS bridge to configure container interface.
#ovs_bridge = br-int

[nsx_kube_proxy]
#
# Configuration for nsx_kube_proxy
#

# The OVS uplink OpenFlow port where to apply the NAT rules to.
# If not specified, the port that gets assigned ofport=1 is used.
#ovs_uplink_port = <None>
```

安装 NSX-T Container Plug-in

NSX-T Container Plug-in (NCP) 是作为 Docker 映像提供的。应在节点上运行 NCP 以提供基础架构服务。不建议在主节点上运行 NCP。

步骤

1 下载 NCP Docker 映像。

文件名是 `nsx-ncp-xxxxxxx.tar`，其中 *xxxxxxx* 是内部版本号。

2 下载 NCP ReplicationController yaml 模板。

文件名是 `ncp-rc.yml`。您可以编辑该文件，或者将其作为您自己的模板文件的示例。

3 将 NCP Docker 映像加载到您的映像注册表中。

```
docker load -i <tar file>
```

4 编辑 `ncp-rc.yml`。

将节点类型设置为裸机。

```
[coe]
node_type = 'BAREMETAL'
```

将映像名称更改为加载的映像。

指定 nsx_api_managers 参数。该版本支持单个 Kubernetes 节点群集和单个 NSX Manager 实例。例如：

```
nsx_api_managers = 192.168.1.180
```

（可选）在 [nsx_v3] 部分中指定 ca_file 参数。该值应该是在验证 NSX Manager 服务器证书时使用的 CA 包文件。如果未设置，将使用系统根 CA。

指定 nsx_api_cert_file 和 nsx_api_private_key_file 参数以使用 NSX-T Data Center 进行身份验证。

nsx_api_cert_file 是 PEM 格式的客户端证书文件的完整路径。该文件的内容应如下所示：

```
-----BEGIN CERTIFICATE-----
<certificate_data_base64_encoded>
-----END CERTIFICATE-----
```

nsx_api_private_key_file 是 PEM 格式的客户端私钥文件的完整路径。该文件的内容应如下所示：

```
-----BEGIN PRIVATE KEY-----
<private_key_data_base64_encoded>
-----END PRIVATE KEY-----
```

如果 Ingress 控制器配置为在 NAT 模式下运行，请指定 ingress_mode = nat 参数。

默认情况下，子网前缀 24 用于从 pod 逻辑交换机的 IP 块中分配的所有子网。要使用不同的子网大小，请在 [nsx_v3] 部分中更新 subnet_prefix 选项。

注 在该 yaml 文件中，您必须指定为 ncp.ini 生成的 ConfigMap 应挂载为只读卷。下载的 yaml 文件已具有该规范，不应对其进行更改。

5 创建 NCP ReplicationController。

```
kubectl create -f ncp-rc.yml
```

注 NCP 打开与 Kubernetes API 服务器的持续 HTTP 连接，以监视 Kubernetes 资源的生命周期事件。如果 API 服务器故障或网络故障导致 NCP 的 TCP 连接失效，则必须重新启动 NCP，以便其可以重新建立与 API 服务器的连接。否则，NCP 将错过新事件。

在滚动更新 NCP ReplicationController 期间，请勿重新引导容器主机。如果主机因任何原因重新引导，您可能会看到重新引导后运行两个 NCP pod。在这种情况下，请执行以下操作：

- 删除其中一个 NCP pod。具体删除哪一个无关紧要。例如，

```
oc delete pods <NCP pod name> -n nsx-system
```

- 删除命名空间 `nsx-system`。例如，

```
oc delete -f ncp-rc.yml -n nsx-system
```

ncp-rc.yml 中的 ncp.ini 的 ConfigMap

示例 yaml 文件 `ncp-rc.yml` 包含配置文件 `ncp.ini` 的 ConfigMap。该 ConfigMap 部分包含一些参数，您必须在安装 NCP 之前指定这些参数，如上一节中所述。

您下载的示例 `ncp-rc.yml` 包含以下 `ncp.ini` 信息：

```
# ConfigMap for ncp.ini
apiVersion: v1
kind: ConfigMap
metadata:
  name: nsx-ncp-config
  labels:
    version: v1
data:
  ncp.ini: |
    [DEFAULT]

    # Set to True to enable logging to stderr
    #use_stderr = True
    # Set to True to send logs to the syslog daemon
    #use_syslog = False
    # Enabler debug-level logging for the root logger. If set to True, the
    # root logger debug level will be DEBUG, otherwise it will be INFO.
    #debug = True

    # The log file path must be set to something like '/var/log/nsx-ujo/'. By
    # default, logging to file is disabled.
    #log_dir = None

    # Name of log file to send logging output to. If log_dir is set but log_file is
    # not, the binary name will be used, i.e., ncp.log, nsx_node_agent.log and
    # nsx_kube_proxy.log.
    #log_file = None
```

```
# max MB for each compressed file. Defaults to 100 MB
#log_rotation_file_max_mb = 100

# Total number of compressed backup files to store. Defaults to 5.
#log_rotation_backup_count = 5
[coe]
#
# Common options for Container Orchestrators
#

# Container orchestrator adaptor to plug in
# Options: kubernetes (default), openshift, pcf.
#adaptor = kubernetes

# Specify cluster for adaptor. It is a prefix of NSX resources name to
# distinguish multiple clusters who are using the same NSX.
# This is also used as the tag of IP blocks for cluster to allocate
# IP addresses. Different clusters should have different IP blocks.
#cluster = k8scluster

# Log level for the NCP operations. If set, overrides the level specified
# for the root logger. Possible values are NOTSET, DEBUG, INFO, WARNING,
# ERROR, CRITICAL
#loglevel=None

# Log level for the NSX API client operations. If set, overrides the level
# specified for the root logger. Possible values are NOTSET, DEBUG, INFO,
# WARNING, ERROR, CRITICAL
nsxlib_loglevel=INFO

# Once enabled, all projects in this cluster will be mapped to a NAT
# topology in NSX backend
#enable_snat = True

# The type of container node. Possible values are HOSTVM, BAREMETAL.
node_type = BAREMETAL

[ha]
#
# NCP High Availability configuration options
#

# Time duration in seconds of mastership timeout. NCP instance will
# remain master for this duration after elected. Note that the heartbeat
# period plus the update timeout must be less than this period. This
# is done to ensure that the master instance will either confirm
# liveness or fail before the timeout.
#master_timeout = 9

# Time in seconds between heartbeats for elected leader. Once an NCP
# instance is elected master, it will periodically confirm liveness based
# on this value.
#heartbeat_period = 3

# Timeout duration in seconds for update to election resource. If the
```

```
# update request does not complete before the timeout it will be
# aborted. Used for master heartbeats to ensure that the update finishes
# or is aborted before the master timeout occurs.
#update_timeout = 3

[k8s]
#
# From kubernetes
#

# IP address of the Kubernetes API Server. If not set, will try to
# read and use the Kubernetes Service IP from environment variable
# KUBERNETES_SERVICE_HOST.
#apiserver_host_ip = <ip_address>

# Port of the Kubernetes API Server.
# Set to 6443 for https. If not set, will try to
# read and use the Kubernetes Service port from environment
# variable KUBERNETES_SERVICE_PORT.
#apiserver_host_port = <port>

# Specify a CA bundle file to use in verifying the Kubernetes API server
# certificate. (string value)
#ca_file = <None>
ca_file = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

# Full path of the Token file to use for authenticating with the k8s API server.
#client_token_file = <None>
client_token_file = /var/run/secrets/kubernetes.io/serviceaccount/token

# Full path of the client certificate file to use for authenticating
# with the k8s API server. It must be specified together with
# "client_private_key_file"
#client_cert_file = <None>

# Full path of the client certificate file to use for authenticating
# with the k8s API server. It must be specified together with
# "client_cert_file"
#client_private_key_file = <None>

# Log level for the kubernetes adaptor. If set, overrides the level specified
# for the root logger. Possible values are NOTSET, DEBUG, INFO, WARNING,
# ERROR, CRITICAL
#loglevel=None

# Specify how ingress controllers are expected to be deployed. Possible values:
# hostnetwork or nat. NSX will create NAT rules only in the second case.
#ingress_mode = hostnetwork

[nsx_v3]
#
# From nsx
#

# IP address of one or more NSX managers separated by commas. The IP address
```

```
# should be of the form (list value):
# <ip_address1>[:<port1>],<ip_address2>[:<port2>],...
# HTTPS will be used for communication with NSX. If port is not provided,
# port 443 will be used.
#nsx_api_managers = <ip_address>

# If true, the NSX Manager server certificate is not verified. If false the CA
# bundle specified via "ca_file" will be used or if unsest the default system
# root CAs will be used. (boolean value)
#insecure = False

# Specify one or a list of CA bundle files to use in verifying the NSX Manager
# server certificate. This option is ignored if "insecure" is set to True. If
# "insecure" is set to False and ca_file is unset, the system root CAs will be
# used to verify the server certificate. (list value)
#ca_file = <None>

# Path to NSX client certificate file. If specified, the nsx_api_user and
# nsx_api_passsword options will be ignored. This option must be specified
# along with "nsx_api_private_key_file" option.
#nsx_api_cert_file = <None>

# Path to NSX client private key file. If specified, the nsx_api_user and
# nsx_api_passsword options will be ignored. This option must be specified
# along with "nsx_api_cert_file" option.
#nsx_api_private_key_file = <None>

# The time in seconds before aborting a HTTP connection to a NSX manager.
# (integer value)
#http_timeout = 10

# The time in seconds before aborting a HTTP read response from a NSX manager.
# (integer value)
#http_read_timeout = 180

# Maximum number of times to retry a HTTP connection. (integer value)
#http_retries = 3

# Maximum concurrent connections to each NSX manager. (integer value)
#concurrent_connections = 10

# The amount of time in seconds to wait before ensuring connectivity to the NSX
# manager if no manager connection has been used. (integer value)
#conn_idle_timeout = 10

# Number of times a HTTP redirect should be followed. (integer value)
#redirects = 2

# Maximum number of times to retry API requests upon stale revision errors.
# (integer value)
#retries = 10

# Subnet prefix of IP block. IP block will be retrieved from NSX API and
# recognised by tag 'cluster'.
# Prefix should be less than 31, as two addresses(the first and last addresses)
```

```
# need to be network address and broadcast address.
# The prefix is fixed after the first subnet is created. It can be changed only
# if there is no subnets in IP block.
#subnet_prefix = 24

# Indicates whether distributed firewall DENY rules are logged.
#log_dropped_traffic = False

# Option to use native loadbalancer support.
#use_native_loadbalancer = False

# Used when ingress class annotation is missing
# if set to true, the ingress will be handled by nsx lbs
# otherwise will be handled by 3rd party ingress controller (e.g. nginx)
#default_ingress_class_nsx = True

# Path to the default certificate file for HTTPS load balancing
#lb_default_cert_path = <None>

# Path to the private key file for default certificate for HTTPS load balancing
#lb_priv_key_path = <None>

# Option to set load balancing algorithm in load balancer pool object.
# Available choices are
# ROUND_ROBIN/LEAST_CONNECTION/IP_HASH/WEIGHTED_ROUND_ROBIN
#pool_algorithm = 'ROUND_ROBIN'

# Option to set load balancer service size. Available choices are
# SMALL/MEDIUM/LARGE.
# MEDIUM Edge VM (4 vCPU, 8GB) only supports SMALL LB.
# LARGE Edge VM (8 vCPU, 16GB) only supports MEDIUM and SMALL LB.
# Bare Metal Edge (IvyBridge, 2 socket, 128GB) supports LARGE, MEDIUM and
# SMALL LB
#service_size = 'SMALL'

# Choice of persistence type for ingress traffic through L7 Loadbalancer.
# Accepted values:
# 'cookie'
# 'source_ip'
#l7_persistence = <None>

# Choice of persistence type for ingress traffic through L4 Loadbalancer.
# Accepted values:
# 'source_ip'
#l4_persistence = <None>

# Name or UUID of the tier0 router that project tier1 routers connect to
#tier0_router = <None>

# Name or UUID of the NSX overlay transport zone that will be used for creating
# logical switches for container networking. It must refer to an existing
# transport zone on NSX and every hypervisor that hosts the Kubernetes
# node VMs must join this transport zone
#overlay_tz = <None>
```

```
# Name or UUID of the NSX lb service that can be attached by virtual servers
#lb_service = <None>

# Name or UUID of the container ip blocks that will be used for creating
# subnets. If name, it must be unique
#container_ip_blocks = <None>

# Name or UUID of the container ip blocks that will be used for creating
# subnets for no-SNAT projects. If specified, no-SNAT projects will use these
# ip blocks ONLY. Otherwise they will use container_ip_blocks
#no_snat_ip_blocks = <None>

# Name or UUID of the external ip pools that will be used for allocating IP
# addresses which will be used for translating container IPs via SNAT rules
#external_ip_pools = <None>

# Firewall sections for this cluster will be created below this mark section
#top_firewall_section_marker = <None>

# Firewall sections for this cluster will be created above this mark section
#bottom_firewall_section_marker = <None>
```

负载平衡

5

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。

从 NCP 2.3.1 开始，`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
```

注 如果同时配置第 4 层和第 7 层负载平衡器，则可以将 `l4_persistence` 和/或 `l7_persistence` 设置为 `source_ip`，但不能将 `l4_persistence` 设置为 `source_ip`，将 `l7_persistence` 设置为 `cookie`。如果错误地将 `l4_persistence` 设置为 `source_ip`，将 `l7_persistence` 设置为 `cookie`，LoadBalancer 服务将不起作用。要解决此问题，必须删除 Ingress 资源和 LoadBalancer 服务，更改持久性设置，重新启动 NCP，并重新创建 Ingress 资源和 LoadBalancer 服务。

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 管理指南》。

  小型 NSX-T Data Center 负载平衡器支持下列各项：

  - 10 个 NSX-T 虚拟服务器。
  - 10 个 NSX-T 池。
  - 30 个 NSX-T 池成员。
  - 8 个端口用于 LoadBalancer 服务。
  - LoadBalancer 服务和路由资源定义的总共 10 个端口。
  - LoadBalancer 服务和路由资源引用的总共 30 个端点。

  中型 NSX-T Data Center 负载平衡器支持下列各项：

  - 100 个 NSX-T 虚拟服务器。

- 100 个 NSX-T 池。
- 300 个 NSX-T 池成员。
- 98 个端口用于 LoadBalancer 服务。
- LoadBalancer 服务和路由资源定义的总共 100 个端口。
- LoadBalancer 服务和路由资源引用的总共 300 个端点。

大型 NSX-T Data Center 负载平衡器支持下列各项：

- 1000 个 NSX-T 虚拟服务器。
- 1000 个 NSX-T 池。
- 3000 个 NSX-T 池成员。
- 998 个端口用于 LoadBalancer 服务。
- LoadBalancer 服务和路由资源定义的总共 1000 个端口。
- LoadBalancer 服务和路由资源引用的总共 3000 个端点。

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。

- 从 NCP 2.3.1 开始，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。此功能需要 NSX-T Data Center 2.3 或更高版本。

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-T Container Plug-in

6

您可以从 NSX Manager GUI 或命令行界面 (Command Line Interface, CLI) 中管理 NSX-T 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-T 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 6-1. 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 6-2. 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 6-3. 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