適用於 OpenShift 的 NSX Container Plug-in - 安裝和管理指南

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

vmware®

您可以在 VMware 網站上找到最新的技術文件，網址如下：

https://docs.vmware.com/tw/

VMware 網站也提供最新的產品更新。

如果您對於本文件有任何意見，歡迎寄至：

docfeedback@vmware.com

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

目錄

適用於 OpenShift 的 NSX-T Container Plug-in - 安裝和管理指南

本指南說明如何安裝和管理 NSX Container Plug-in (NCP)，以整合 NSX-T Data Center 與 OpenShift。

主要對象

本指南適用於系統和網路管理員。我們假設讀者熟悉 NSX-T Data Center 和 OpenShift 的安裝與管理。

VMware 技術出版品詞彙表

VMware 技術出版品將為您提供可能不熟悉的術語詞彙。如需 VMware 技術說明文件中所用專有詞彙的定義，請前往 http://www.vmware.com/support/pubs。

NSX-T Container Plug-in 概觀

1

NSX Container Plug-in(NCP) 可用來整合 NSX-T Data Center 與 Kubernetes 之類的容器協調工具，也可以整合 NSX-T Data Center 與容器型 PaaS (平台即服務) 軟體產品，例如 OpenShift。本指南說明如何使用 OpenShift 來設定 NCP。

NCP 的主要元件會在容器中執行，且會與 NSX Manager 和 OpenShift 控制平面通訊。NCP 會監控容器和其他資源的變更，且藉由呼叫 NSX API 來管理容器的網網路資源，例如邏輯連接埠、交換器、路由器和安全群組。

NSX CNI 外掛程式會在每個 OpenShift 節點上執行。它會監控容器的生命週期事件、將容器介面連線至客體 vSwitch，以及安排要標記的客體 vSwitch，並轉送容器介面與 VNIC 之間的容器流量。

NCP 提供下列功能：

- 為 OpenShift 叢集自動建立 NSX-T 邏輯拓撲，且為每個 OpenShift 命名空間建立單獨的邏輯網路。
- 將 OpenShift 網繭連線至邏輯網路，並配置 IP 和 MAC 位址。
- 支援網路位址轉譯 (NAT) 且為每個 OpenShift 命名空間配置單獨的 SNAT IP。

  備註 設定 NAT 時，轉譯的 IP 總數不能超過 1000。

- 透過 NSX-T 分散式防火牆實作 OpenShift 網路原則。
  - 支援入口和出口網路原則。
  - 支援網路原則中的 `IPBlock` 選取器。
  - 為網路原則指定標籤選取器時，支援 `matchLabels` 和 `matchExpression`。
- 透過 NSX-T 第 7 層負載平衡器實作 OpenShift 路由。
  - 透過 TLS Edge 終止支援 HTTP 路由和 HTTPS 路由。
  - 透過替代後端和萬用子網域支援路由。
- 為命名空間、網繭名稱和網繭標籤在 NSX-T 邏輯交換器連接埠上建立標記，並允許管理員根據標記定義 NSX-T Data Center 安全群組和原則。

在此版本中，NCP 支援單一 OpenShift 叢集。

本章包含以下主題：

相容性需求

NSX Container Plug-in (NCP) 具有下列相容性需求。

軟體產品	版本
NSX-T Data Center	2.3、2.4
容器主機虛擬機器的 Hypervisor	■ 支援的 vSphere 版本 ■ RHEL KVM 7.4、7.5、7.6
容器主機作業系統	RHEL 7.4、7.5、7.6
平台即服務	OpenShift 3.10、3.11
Container Host Open vSwitch	2.10.2 (隨附於 NSX-T Data Center 2.4)

安裝概觀

安裝和設定 NCP 涉及下列步驟。若要成功執行這些步驟，您必須熟悉 NSX-T Data Center 和 OpenShift 安裝與管理。

1 安裝 NSX-T Data Center。

2 建立覆疊傳輸區域。

3 建立覆疊邏輯交換器，並將節點連線至交換器。

4 建立第 0 層邏輯路由器。

5 建立網繭的 IP 區塊。

6 建立 SNAT (來源網路位址轉譯) 的 IP 集區。

7 準備 Ansible 主機檔案。

8 使用單一 Playbook 安裝 NCP 和 OpenShift。

升級 NCP

本節說明如何將 NCP 升級至 2.4.0。

程序

1 升級 CNI RPM 套件、NSX 節點代理程式 DaemonSet 和 NCP ReplicationController。

2 準備 Ansible 主機檔案。

每個節點必須具有指定的參數 `openshift_node_group_name`。例如，

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

3 (可選) 設定負載平衡。

新增步驟，為負載平衡器服務的外部 IP 位址指定不同的 IP 集區。例如，

`external_ip_pools_lb = <nsx ip pool name>`

設定 NSX-T 資源

2

您必須建立 NSX-T Data Center 資源以提供 OpenShift 節點所需的網路。

設定 NSX-T 資源

您需要設定的 NSX-T Data Center 資源包含覆疊傳輸區域、第 0 層邏輯路由器、要連線節點虛擬機器的邏輯交換器、Kubernetes 節點的 IP 區塊，以及 SNAT 的 IP 集區。

重要 如果您正在使用 NSX-T Data Center 2.4 或更新版本執行，則必須使用**進階網路與安全性**索引標籤設定 NSX-T 資源。

在 NCP 組態檔 `ncp.ini` 中，NSX-T Data Center 資源是使用其 UUID 或名稱指定的。

覆疊傳輸區域

登入 NSX Manager，並尋找用於容器網路的覆疊傳輸區域，或是建立新的覆疊傳輸區域。

透過在 `ncp.ini` 的 `[nsx_v3]` 區段中設定 `overlay_tz` 選項，指定叢集的覆疊傳輸區域。此步驟是可選的。如果沒有設定 `overlay_tz`，NCP 將自動從第 0 層路由器擷取覆疊傳輸區域識別碼。

第 0 層邏輯路由

登入 NSX Manager，並尋找用於容器網路的路由器，或是建立新的路由器。

透過在 `ncp.ini` 的 `[nsx_v3]` 區段中設定 `tier0_router` 選項，指定叢集的第 0 層邏輯路由器。

備註 路由器必須在主動備用模式中建立。

邏輯交換器

節點用於資料流量的 vNIC 必須連線至覆疊邏輯交換器。節點的管理介面並非強制連線至 NSX-T Data Center，儘管這麼做可以更輕鬆地進行設定。您可以透過登入 NSX Manager 來建立邏輯交換器。在交換器上，建立邏輯連接埠並向其附加節點 vNIC。邏輯連接埠必須具有下列標記：

- 標記：`<cluster_name>`，範圍：`ncp/cluster`
- 標記：`<node_name>`，範圍：`ncp/node_name`

`<cluster_name>` 值必須符合 `ncp.ini` 之 `[coe]` 區段中 `cluster` 選項的值。

Kubernetes 網繭的 IP 區塊

登入 NSX Manager，並建立一或多個 IP 區塊。以 CIDR 格式指定 IP 區塊。

透過在 `ncp.ini` 的 `[nsx_v3]` 區段中設定 `container_ip_blocks` 選項，指定 Kubernetes 網繭的 IP 區塊。

您也可以建立非 SNAT 命名空間專用的 IP 區塊。

透過在 `ncp.ini` 的 `[nsx_v3]` 區段中設定 `no_snat_ip_blocks` 選項，指定非 SNAT IP 區塊。

如果您在 NCP 執行時建立無 SNAT IP 區塊，則必須重新啟動 NCP。否則，NCP 將會繼續使用共用的 IP 區塊，直到耗盡為止。

備註 當您建立 IP 區塊時，首碼長度不得大於 NCP 之組態檔 `ncp.ini` 中的 `subnet_prefix` 參數值。

SNAT 的 IP 集區

IP 集區用來配置 IP 位址，從而用於透過 SNAT 規則轉譯網繭 IP，以及用於透過 SNAT/DNAT 規則公開入口控制站，如同 Openstack 浮動 IP。這些 IP 位址也稱為「外部 IP」。

多個 Kubernetes 叢集會使用相同的外部 IP 集區。每個 NCP 執行個體皆會針對其管理的 Kubernetes 叢集使用此集區的子網路。依預設，系統會使用網繭子網路的相同子網路首碼。如需使用不同的子網路大小，請更新 `ncp.ini` 之 `[nsx_v3]` 區段中的 `external_subnet_prefix` 選項。

登入 NSX Manager，並建立集區或尋找現有集區。

透過在 `ncp.ini` 的 `[nsx_v3]` 區段中設定 `external_ip_pools` 選項，指定 SNAT 的 IP 集區。

您也可以透過新增註解至服務來設定特定服務的 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 將針對此服務設定 SNAT 規則。規則的來源 IP 為後端網繭集。目的地 IP 是從指定的外部 IP 集區進行配置的 SNAT IP。請注意下列事項：

- `ncp/snat_pool` 所指定的 IP 集區應在服務設定之前已存在於 NSX-T Data Center 中。IP 集區必須有標記 **`{"ncp/owner": cluster:<cluster>}`**。
- 在 NSX-T Data Center 中，服務的 SNAT 規則優先順序高於專案的優先順序。
- 如果網繭已設定多個 SNAT 規則，則只有一個規則適用。

您可以透過將下列標記新增至 IP 集區，指定可從 SNAT IP 集區配置 IP 的命名空間。

- 範圍：ncp/owner，標記：ns:<namespace_UUID>

您可以使用下列其中一個命令取得命名空間 UUID：

```
oc get ns –o yaml
```

請注意下列事項：

- 每個標記應指定一個 UUID。您可以為同一個集區建立多個標記。
- 如果您根據舊標記為一些命名空間配置 IP 後變更標記，將不會回收這些 IP，直到服務的 SNAT 組態變更或 NCP 重新啟動為止。
- 命名空間擁有者標記是可選的。如果沒有此標記，則可從 SNAT IP 集區配置 IP 給任何命名空間。

(選擇性) 防火牆標記區段

若要允許管理員建立防火牆規則同時不影響 NCP 根據網路原則建立的防火牆區段，請登入 NSX Manager，然後建立兩個防火牆區段。

透過在 ncp.ini 的 [nsx_v3] 區段中設定 bottom_firewall_section_marker 和 top_firewall_section_marker 選項，指定標記防火牆區段。

底部防火牆區段必須位於頂部防火牆區段下方。建立這些防火牆區段之後，由 NCP 建立用於隔離的所有防火牆區段將建立於底部防火牆區段之上，而 NCP 建立用於原則的所有防火牆區段將建立於頂部防火牆區段之下。如果沒有建立這些標記區段，則所有隔離規則將會在底部建立，而所有原則區段將會在頂部建立。每個叢集的各個標記防火牆區段的值必須是唯一的，否則將導致錯誤。

安裝 NCP

3

NCP 與 OpenShift 完全整合。您在 Ansible 主機檔案中新增所需參數並安裝 OpenShift 時，NCP 會自動安裝。

本章包含以下主題：

系統需求

安裝 OpenShift 之前，請確保您的環境符合特定需求。

一般需求

- Ansible 2.4 或更新版本。

虛擬機器需求

OpenShift 節點虛擬機器必須具有兩個 vNIC：

- 連線至邏輯交換器 (具有至管理第 1 層路由器的上行) 的管理 vNIC。
- 所有虛擬機器上的第二個 vNIC 必須具有 NSX-T 中的下列標記，以便 NCP 知道哪一個連接埠用作特定 OpenShift 節點上執行的所有網繭的父系 VIF。

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

裸機機器需求

- OpenShift 節點必須是 NSX-T 傳輸節點，且必須在傳輸節點 (而不是 VIF) 上套用上述標記。
- Ansible 主機檔案必須具有此設定：`nsx_node_type='BAREMETAL'`。

NSX-T 需求

- 第 0 層路由器。

- 覆疊傳輸區域。
- 網繭網路的 IP 區塊。
- (選用) 路由 (無 NAT) 網繭網路的 IP 區塊。
- SNAT 的 IP 集區。依預設，網繭網路的 IP 區塊只能在 NSX-T 內路由。NCP 使用此 IP 集區提供外部連線。
- (選用) 頂部和底部防火牆區段。NCP 將在這兩個區段之間放置 Kubernetes 網路原則規則。
- Open vSwitch 和 CNI 外掛程式 RPM 必須主控於可從 OpenShift 節點虛擬機器連線的 HTTP 伺服器上。

NCP Docker 映像

目前不公開提供 NCP Docker 映像。您必須在本機私人登錄中具有映像 `nsx-ncp`，或執行下列操作：

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

在所有節點上：

```
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
```

準備 Ansible 主機檔案

您必須在 Ansible 主機檔案中為將與 OpenShift 整合的 NCP 指定 NCP 參數。

在 Ansible 主機檔案中指定下列參數後，安裝 OpenShift 時將自動安裝 NCP。

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

  (必要) 這是必要的，因為多個 OpenShift/Kubernetes 叢集可以連線至同一 NSX Manager。
- `nsx_api_managers='10.10.10.10'`

  (必要) NSX Manager 的 IP 位址。針對 NSX Manager 叢集，指定以逗號分隔的 IP 位址。
- `nsx_tier0_router='MyT0Router'`

  (必要) 專案的第 1 層路由器將連線到的第 0 層路由器的名稱或 UUID。
- `nsx_overlay_transport_zone='my_overlay_tz'`

  (必要) 將用於建立邏輯交換器的覆疊傳輸區域的名稱或 UUID。

- `nsx_container_ip_block='ip_block_for_my_ocp_cluster'`

  (必要) NSX-T 上設定的 IP 區塊的名稱或 UUID。此 IP 區塊之外的每個專案均有一個子網路。這些網路位於 SNAT 之後，且無法路由。

- `nsx_ovs_uplink_port='ens224'`

  (必要) 如果處於 `HOSTVM` 模式。對於 OCP 節點上的網繭網路，NSX-T 需要不同於管理 vNIC 的第二個 vNIC。強烈建議將兩個 vNIC 均連線至 NSX-T 邏輯交換器。必須在此處提供第二個 (非管理) vNIC。對於裸機，不需要此參數。

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

  (必要) 暫時需求，直到 NCP 可以對節點執行啟動程序。我們必須在 HTTP 伺服器上放置 `nsx-cni`。

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

  (必要) 暫時參數，直到 NCP 可以對節點執行啟動程序。可在裸機設定中忽略。

- `nsx_node_type='HOSTVM'`

  (選用) 預設為 `HOSTVM`。如果 OpenShift 未在虛擬機器中執行，則設定為 `BAREMETAL`。

- `nsx_k8s_api_ip=192.168.10.10`

  (選用) 如果設定，NCP 將告知此 IP 位址，否則告知 Kubernetes 服務 IP。

- `nsx_k8s_api_port=192.168.10.10`

  (選用) 對於 Kubernetes 服務，預設為 443。如果您將其與 `nsx_k8s_api_ip` 組合使用來指定主節點 IP，則設定為 8443。

- `nsx_insecure_ssl=true`

  (選用) NSX Manager 隨附不受信任的憑證時，預設值為 `true`。如果您使用受信任的憑證變更了此憑證，則可以將其設定為 `false`。

- `nsx_api_user='admin'`
- `nsx_api_password='super_secret_password'`
- `nsx_subnet_prefix=24`

  (選用) 預設為 24。這是每個 Openshift 專案將專用的子網路大小。如果網繭數目超過子網路大小，具有相同子網路大小的新邏輯交換器將新增至專案。

- `nsx_use_loadbalancer=true`

  (選用) 預設為 `true`。如果您不想針對 OpenShift 路由和類型為負載平衡器的服務使用 NSX-T 負載平衡器，則設定為 `false`。

- `nsx_lb_service_size='SMALL'`

  (選用) 預設為 `SMALL`。取決於 NSX Edge 大小，`MEDIUM` 或 `LARGE` 也有可能。

- `nsx_no_snat_ip_block='router_ip_block_for_my_ocp_cluster'`

  (選用) 如果在專案或命名空間上套用 `ncp/no_snat=true` 註解，將從此 IP 區塊擷取子網路，且它沒有 SNAT。預期可路由。

- `nsx_external_ip_pool='external_pool_for_snat'`

  (必要) 如果未定義 `nsx_external_ip_pool_lb`，則為 SNAT 和負載平衡器的 IP 集區。

- `nsx_external_ip_pool_lb='my_ip_pool_for_lb'`

  (選用) 如果您對 `Router` 和 `SvcTypeLB` 需要不同的 IP 集區，請設定此項。

- `nsx_top_fw_section='top_section'`

  (選用) 將 Kubernetes 網路原則規則轉譯為 NSX-T 防火牆規則，並將其放置在此區段下方。

- `nsx_bottom_fw_section='bottom_section'`

  (選用) 將 Kubernetes 網路原則規則轉譯為 NSX-T 防火牆規則，並將其放置在此區段上方。

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

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

  (選用) 如果設定，`nsx_api_user` 和 `nsx_api_password` 將被忽略。必須將憑證上傳到 NSX-T，且必須手動建立使用此憑證驗證的主體身分識別使用者。

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

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

  (選用) NSX-T 負載平衡器需要預設憑證，才能為 TLS 式路由建立 SNI。僅當未設定路由時，才會顯示此憑證。如果未提供，將會產生自我簽署憑證。

Ansible 主機檔案範例

```
[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'
```

負載平衡

4

NSX-T Data Center 負載平衡器與 OpenShift 整合，並用作 OpenShift 路由器。

NCP 監看 OpenShift 路由和端點事件，且根據路由規格設定負載平衡器上的負載平衡規則。因此，NSX-T Data Center 負載平衡器會根據規則將傳入第 7 層流量轉送到適當的後端網繭。

設定負載平衡

設定負載平衡包括設定 Kubernetes 負載平衡器服務或 OpenShift 路由。您還需要設定 NCP 複寫控制站。負載平衡器服務適用於第 4 層流量，而 OpenShift 路由適用於第 7 層流量。

當您設定 Kubernetes 負載平衡器服務時，將從您設定的外部 IP 區塊為該服務配置 IP 位址。負載平衡器將在此 IP 位址和服務連接埠上公開。您可以使用負載平衡器定義中的 `loadBalancerIP` 規格指定 IP 集區的名稱或識別碼。負載平衡器服務的 IP 將從此 IP 集區配置。如果 `loadBalancerIP` 規格為空白，將從您設定的外部 IP 區塊配置 IP。

`loadBalancerIP` 所指定的 IP 集區必須有標記 **`{"ncp/owner": cluster:<cluster>}`**。

若要使用 NSX-T Data Center 負載平衡器，您必須在 NCP 中設定負載平衡。在 `ncp_rc.yml` 檔案中，執行下列操作：

1 將 `use_native_loadbalancer` 設為 `True`。

2 將 `pool_algorithm` 設為 `WEIGHTED_ROUND_ROBIN`。

3 將 lb_default_cert_path 和 lb_priv_key_path 分別設定為 CA 簽署憑證檔案和私密金鑰檔案的完整路徑名稱。請參閱以下產生 CA 簽署憑證的範例指令碼。此外，將預設憑證和金鑰掛接到 NCP 網繭。如需相關指示，請參閱以下內容。

4 (選擇性) 使用 `l4_persistence` 和 `l7_persistence` 參數指定持續性設定。第 4 層持續性的可用選項為來源 IP。第 7 層持續性的可用選項為 Cookie 和來源 IP。預設值為 <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 (選擇性) 將 `service_size` 設為 `SMALL`、`MEDIUM` 或 `LARGE`。預設值為 `SMALL`。

6 如果您正在執行 OpenShift 3.11，您必須執行下列組態，以便 OpenShift 不會將 IP 指派給負載平衡器服務。

- 在 `/etc/origin/master/master-config.yaml` 檔案中，在 `networkConfig` 下將 `ingressIPNetworkCIDR` 設定為 0.0.0.0/32。
- 使用下列命令重新啟動 API 伺服器和控制器：

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

如果已關閉全域第 4 層持續性，您也可以為 Kubernetes 負載平衡器服務指定服務規格上的 `sessionAffinity`，以設定服務的持續性行為，也就是將 `l4_persistence` 設定為 <None>。如果將 `l4_persistence` 設為 `source_ip`，服務規格的 `sessionAffinity` 則可用於自訂服務的持續性逾時。預設的第 4 層持續性逾時為 10800 秒 (如同服務 Kubernetes 說明文件中指定的值 (https://kubernetes.io/docs/concepts/services-networking/service)。具有預設持續性逾時的所有服務，將共用相同的 NSX-T 負載平衡器持續性設定檔。會為每個使用非預設持續性逾時的服務建立專用的設定檔。

備註 如果入口的後端服務是一種類型為負載平衡器的服務，則此服務的第 4 層虛擬伺服器和此入口的第 7 層虛擬伺服器不能有不同的持續性設定，例如，第 4 層的 `source_ip` 和第 7 層的 `cookie`。在此案例中，這兩個虛擬伺服器的持續性設定必須相同 (`source_ip`、`cookie` 或 `None`)，或者其中一個虛擬伺服器是 `None` (則另一個設定可以是 `source_ip` 或 `cookie`)。此案例的範例：

```
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
```

路由器分區

NCP 一律處理 TLS Edge 終止和 HTTP 路由，且略過 TLS 傳遞路由和 TLS 重新加密路由，無論其命名空間或命名空間標籤為何。若要限制 OpenShift 路由器僅處理 TLS 重新加密和傳遞路由，您必須執行下列步驟：

- 將命名空間標籤選取器新增至 Openshift 路由器。

- 將命名空間標籤新增至目標命名空間。
- 在目標命名空間中建立 TLS 重新加密/傳遞路由。

例如，若要設定具有命名空間標籤選取器的路由器，請執行下列命令 (假設路由器的服務帳戶名稱是 router)：

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

路由器將立即處理所選命名空間中的路由。若要使此選取器符合命名空間，請執行下列命令 (假設命名空間稱為 ns1)：

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

第 7 層負載平衡器範例

下列 YAML 檔案設定兩個複寫控制站 (tea-rc 和 coffee-rc)、兩項服務 (tea-svc 和 coffee-svc) 和兩個路由 (cafe-route-multi 和 cafe-route)，以提供第 7 層負載平衡。

```
# 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
```

其他附註

- HTTPS 流量僅支援 Edge 終止。
- 支援萬用字元子網域。例如，如果 `wildcardPolicy` 設為 **`Subdomain`**，且主機名稱設為 **`wildcard.example.com`**，將為 **`*.example.com`** 的任何要求提供服務。
- 如果 NCP 因錯誤組態在路由事件處理期間擲回錯誤，則需要更正路由 YAML 檔案、刪除並重新建立路由資源。
- NCP 不會依命名空間強制執行主機名稱擁有權。
- 每個 Kubernetes 叢集支援一個負載平衡器服務。
- NSX-T Data Center 將為每個負載平衡器服務連接埠建立第 4 層負載平衡器虛擬伺服器和集區。TCP 和 UDP 均受支援。
- NSX-T Data Center 負載平衡器的大小不同。如需設定 NSX-T Data Center 負載平衡器的相關資訊，請參閱《NSX-T Data Center 管理指南》。

  建立負載平衡器後，無法透過更新組態檔來變更負載平衡器大小。可透過 UI 或 API 進行變更。
- 支援自動調整第 4 層負載平衡器。如果 Kubernetes 負載平衡器服務已建立或修改，使其需要額外的虛擬伺服器，而現有的第 4 層負載平衡器沒有容量，將會建立新的第 4 層負載平衡器。NCP 也將刪除不再連結有虛擬伺服器的第 4 層負載平衡器。此功能預設為啟用狀態。可以透過在 NCP ConfigMap 中將 `l4_lb_auto_scaling` 設定為 **`false`**，將其停用。

產生 CA 簽署憑證的範例指令碼

下列指令碼會產生 CA 簽署憑證和私密金鑰，其分別儲存在 <filename>.crt 和 <finename>.key 檔案中。`genrsa` 命令會產生 CA 金鑰。應加密 CA 金鑰。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
```

將預設憑證和金鑰掛接到 NCP 網繭

產生憑證和私密金鑰後，將其放置在主機虛擬機器上的目錄 /etc/nsx-ujo 中。假設憑證和金鑰檔案的名稱分別為 lb-default.crt 和 lb-default.key，編輯 ncp-rc.yaml，以便主機上的這些檔案掛接到網繭中。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

您可以從 NSX Manager GUI 或從命令列介面 (CLI) 來管理 NSX Container Plug-in。

備註 如果容器主機虛擬機器正在 ESXi 6.5 上執行且該虛擬機器已透過 vMotion 移轉至其他 ESXi 6.5 主機，則容器主機上執行的容器將與其他容器主機上執行的容器中斷連線。您可以透過中斷連線並連線容器主機的 vNIC 來解決此問題。ESXi 6.5 Update 1 或更新版本不會發生此問題。

Hyperbus 會保留 Hypervisor 上的 VLAN 識別碼 4094，以用於 PVLAN 組態，並且此識別碼無法變更。若要避免任何 VLAN 衝突，請勿使用相同的 VLAN 識別碼設定 VLAN 邏輯交換器或 VTEP vmknic。

本章包含以下主題：

從 NSX Manager GUI 管理 IP 區塊

您可以從 NSX Manager GUI 新增、刪除、編輯和檢視 IP 區塊的詳細資料，以及管理 IP 區塊的標記。

程序

1 從瀏覽器登入 NSX Manager，網址為 `https://<nsx-manager-IP-address-or-domain-name>`。

2 導覽至**網路 > IPAM**。

現有的 IP 區塊清單隨即顯示。

3 執行下列任何動作。

選項	動作
新增 IP 區塊	按一下**新增**。
刪除一或多個 IP 區塊	選取一或多個 IP 區塊，然後按一下**刪除**。
編輯 IP 區塊	選取 IP 區塊，然後按一下**編輯**。

選項	動作
檢視關於 IP 區塊的詳細資料	按一下 IP 區塊名稱。按一下**概觀**索引標籤以檢視一般資訊。按一下**子網路**索引標籤以檢視此 IP 區塊的子網路。
管理 IP 區塊的標記	選取 IP 區塊，然後按一下**動作 > 管理標記**。

您無法刪除已配置子網路的 IP 區塊。

從 NSX Manager GUI 檢視 IP 區塊子網路

您可以從 NSX Manager GUI 檢視 IP 區塊的子網路。建議不要在 NCP 安裝並執行後新增或刪除 IP 區塊子網路。

程序

1 從瀏覽器登入 NSX Manager，網址為 `https://<nsx-manager-IP-address-or-domain-name>`。

2 導覽至**網路 > IPAM**。

現有的 IP 區塊清單隨即顯示。

3 按一下 IP 區塊名稱

4 按一下**子網路**索引標籤。

CIF 連結邏輯連接埠

CIF (容器介面) 是容器上的網路介面，可連線至交換器上的邏輯連接埠。這些連接埠稱為 CIF 連結邏輯連接埠。

您可以從 NSX Manager GUI 管理 CIF 連結邏輯連接埠。

管理 CIF 連結邏輯連接埠

導覽至**網路 > 交換 > 連接埠**可查看所有邏輯連接埠，包括 CIF 連結邏輯連接埠。按一下 CIF 連結邏輯連接埠的附加連結，可查看附加資訊。按一下邏輯連接埠連結，可開啟包含下列四個索引標籤的視窗窗格：「概觀」、「監控」、「管理」和「相關」。按一下**相關 > 邏輯連接埠**會在上行交換器上顯示相關的邏輯連接埠。如需關於交換器連接埠的詳細資訊，請參閱《NSX-T 管理指南》。

網路監控工具

下列工具支援 CIF 連結邏輯連接埠。如需關於這些工具的詳細資訊，請參閱《NSX-T 管理指南》。

- Traceflow
- 連接埠連線
- IPFIX
- 支援使用連線至容器之邏輯交換器連接埠的 GRE 封裝進行遠端連接埠鏡像。如需詳細資訊，請參閱《NSX-T 管理指南》中的〈瞭解連接埠鏡像交換設定檔〉。不過，透過管理程式 UI 不支援 CIF 到 VIF 連接埠的連接埠鏡像。

CLI 命令

若要執行 CLI 命令，請登入 NSX Container Plug-in 容器，接著開啟終端機，然後執行 `nsxcli` 命令。

您也可以透過在節點上執行下列命令來取得 CLI 提示：

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

表格 5-1. NCP 容器的 CLI 命令

類型	命令
狀態	get ncp-master status
狀態	取得 ncp-nsx 狀態
狀態	取得 ncp-watcher <watcher-name>
狀態	取得 ncp-watchers
狀態	取得 ncp-k8s-api-server 狀態
狀態	check projects
狀態	check project <project-name>
快取	取得 project-cache <project-name>
快取	取得 project-caches
快取	取得 namespace-cache <namespace-name>
快取	取得 namespace-caches
快取	取得 pod-cache <pod-name>
快取	取得 pod-caches
快取	取得 ingress-caches
快取	get ingress-cache <ingress-name>
快取	get ingress-controllers
快取	get ingress-controller <ingress-controller-name>
快取	get network-policy-caches
快取	get network-policy-cache <pod-name>
支援	取得 ncp-log 檔案 <filename>
支援	get ncp-log-level
支援	set ncp-log-level <log-level>
支援	取得 support-bundle 檔案 <filename>
支援	取得 node-agent-log 檔案 <filename>
支援	取得 node-agent-log 檔案 <filename> <node-name>

表格 5-2. NSX 節點代理程式容器的 CLI 命令

類型	命令
狀態	取得 node-agent-hyperbus 狀態
快取	get container-cache <container-name>
快取	get container-caches

表格 5-3. NSX Kube Proxy 容器的 CLI 命令

類型	命令
狀態	取得 ncp-k8s-api-server 狀態
狀態	取得 kube-proxy-watcher <watcher-name>
狀態	取得 kube-proxy-watchers
狀態	傾印 ovs-flows

NCP 容器的狀態命令

- 顯示 NCP 主機的狀態

  ```
  get ncp-master status
  ```

  範例：

  ```
  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
  ```

- 顯示 NCP 與 NSX Manager 之間的連線狀態

  ```
  get ncp-nsx status
  ```

  範例：

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

- 顯示入口、命名空間、網繭和服務的監看員狀態

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

範例 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
```

範例 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
```

- 顯示 NCP 與 Kubernetes API 伺服器之間的連線狀態

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

範例：

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

- 檢查所有專案或特定專案

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

範例：

```
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
```

NCP 容器的快取命令

- 取得專案或命名空間的內部快取

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

範例：

```
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
```

- 取得網繭的內部快取

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

範例：

```
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
```

- 取得網路原則快取或特定快取

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

範例：

```
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
```

NCP 容器的支援命令

- 將 NCP 支援服務包儲存在 Filestore 中

  支援服務包是由網繭中所有容器的記錄檔所組成，並帶有標記 **`tier:nsx-networking`**。服務包檔案為 tgz 格式，並儲存於 CLI 預設 Filestore 目錄 `/var/vmware/nsx/file-store`。您可以使用 CLI file-store 命令，將服務包檔案複製到遠端網站。

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

  範例：

  ```
  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
  ```

- 將 NCP 記錄儲存在 Filestore 中

  記錄檔是以 tgz 格式儲存於 CLI 預設 Filestore 目錄 `/var/vmware/nsx/file-store`。您可以使用 CLI file-store 命令，將服務包檔案複製到遠端網站。

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

  範例：

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

- 將節點代理程式記錄儲存在 Filestore 中

  儲存來自一個節點或所有節點的節點代理程式記錄。記錄是以 tgz 格式儲存於 CLI 預設 Filestore 目錄 `/var/vmware/nsx/file-store`。您可以使用 CLI file-store 命令，將服務包檔案複製到遠端網站。

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

  範例：

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

- 取得並設定記錄層級

  可用的記錄層級為 `NOTSET`、`DEBUG`、`INFO`、`WARNING`、`ERROR` 和 `CRITICAL`。

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

  範例：

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

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

NSX 節點代理程式容器的狀態命令

- 顯示節點代理程式與此節點上的 HyperBus 之間的連線狀態。

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

範例：

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

NSX 節點代理程式容器的快取命令

- 取得 NSX 節點代理程式容器的內部快取。

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

範例 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
```

範例 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
```

NSX Kube-Proxy 容器的狀態命令

- 顯示 Kube Proxy 與 Kubernetes API 伺服器之間的連線狀態

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

範例：

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

- 顯示 Kube Proxy 監看員狀態

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

範例 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
```

範例 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
```

- 傾印節點上的 OVS 流量

```
dump ovs-flows
```

範例：

```
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
```

錯誤碼

本節列出了各種元件所產生的錯誤碼。

NCP 錯誤碼

錯誤碼	說明
NCP00001	無效的組態
NCP00002	初始化失敗
NCP00003	無效的狀態
NCP00004	無效的介面卡
NCP00005	找不到憑證
NCP00006	找不到 Token
NCP00007	無效的 NSX 組態
NCP00008	無效的 NSX 標記
NCP00009	NSX 連線失敗
NCP00010	找不到節點標記
NCP00011	無效的節點邏輯交換器連接埠
NCP00012	父系 VIF 更新失敗
NCP00013	VLAN 已用盡
NCP00014	VLAN 釋放失敗
NCP00015	IP 集區已用盡
NCP00016	IP 釋放失敗
NCP00017	IP 區塊已用盡
NCP00018	IP 子網路建立失敗
NCP00019	IP 子網路刪除失敗
NCP00020	IP 集區建立失敗
NCP00021	IP 集區刪除失敗
NCP00022	邏輯路由器建立失敗
NCP00023	邏輯路由器更新失敗
NCP00024	邏輯路由器刪除失敗
NCP00025	邏輯交換器建立失敗

錯誤碼	說明
NCP00026	邏輯交換器更新失敗
NCP00027	邏輯交換器刪除失敗
NCP00028	邏輯路由器連接埠建立失敗
NCP00029	邏輯路由器連接埠刪除失敗
NCP00030	邏輯交換器連接埠建立失敗
NCP00031	邏輯交換器連接埠更新失敗
NCP00032	邏輯交換器連接埠刪除失敗
NCP00033	找不到網路原則
NCP00034	防火牆建立失敗
NCP00035	防火牆讀取失敗
NCP00036	防火牆更新失敗
NCP00037	防火牆刪除失敗
NCP00038	找到多個防火牆
NCP00039	NSGroup 建立失敗
NCP00040	NSGroup 刪除失敗
NCP00041	IP 集合建立失敗
NCP00042	IP 集合更新失敗
NCP00043	IP 集合刪除失敗
NCP00044	SNAT 規則建立失敗
NCP00045	SNAT 規則刪除失敗
NCP00046	介面卡 API 連線失敗
NCP00047	介面卡監看程式例外狀況
NCP00048	負載平衡器服務刪除失敗
NCP00049	負載平衡器虛擬伺服器建立失敗
NCP00050	負載平衡器虛擬伺服器更新失敗

錯誤碼	說明
NCP00051	負載平衡器虛擬伺服器刪除失敗
NCP00052	負載平衡器集區建立失敗
NCP00053	負載平衡器集區更新失敗
NCP00054	負載平衡器集區刪除失敗
NCP00055	負載平衡器規則建立失敗
NCP00056	負載平衡器規則更新失敗
NCP00057	負載平衡器規則刪除失敗
NCP00058	負載平衡器集區 IP 釋放失敗

錯誤碼	說明
NCP00059	找不到負載平衡器虛擬伺服器和服務關聯
NCP00060	NSGroup 更新失敗
NCP00061	取得防火牆規則失敗
NCP00062	NSGroup 無準則
NCP00063	找不到節點虛擬機器
NCP00064	找不到節點 VIF
NCP00065	憑證匯入失敗
NCP00066	憑證取消匯入失敗
NCP00067	SSL 繫結更新失敗
NCP00068	找不到 SSL 設定檔
NCP00069	找不到 IP 集區
NCP00070	找不到第 0 層 Edge 叢集
NCP00071	IP 集區更新失敗
NCP00072	發送器失敗
NCP00073	NAT 規則刪除失敗
NCP00074	取得邏輯路由器連接埠失敗
NCP00075	NSX 組態驗證失敗

錯誤碼	說明
NCP00076	SNAT 規則更新失敗
NCP00077	SNAT 規則重疊
NCP00078	負載平衡器端點新增失敗
NCP00079	負載平衡器端點更新失敗
NCP00080	負載平衡器規則集區建立失敗
NCP00081	找不到負載平衡器虛擬伺服器
NCP00082	IP 集合讀取失敗
NCP00083	取得 SNAT 集區失敗
NCP00084	負載平衡器服務建立失敗
NCP00085	負載平衡器服務更新失敗
NCP00086	邏輯路由器連接埠更新失敗
NCP00087	負載平衡器初始化失敗
NCP00088	IP 集區不是唯一的
NCP00089	第 7 層負載平衡器快取同步錯誤
NCP00090	負載平衡器集區不存在錯誤
NCP00091	負載平衡器規則快取初始化錯誤

錯誤碼	說明
NCP00092	SNAT 處理失敗
NCP00093	負載平衡器預設憑證錯誤
NCP00094	負載平衡器端點刪除失敗
NCP00095	找不到專案
NCP00096	集區存取遭拒
NCP00097	無法取得負載平衡器服務
NCP00098	無法建立負載平衡器服務
NCP00099	負載平衡器集區快取同步錯誤

NSX 節點代理程式錯誤碼

錯誤碼	說明
NCP01001	找不到 OVS 上行
NCP01002	找不到主機 MAC
NCP01003	OVS 連接埠建立失敗
NCP01004	沒有網繭組態
NCP01005	網繭設定失敗
NCP01006	網繭取消設定失敗
NCP01007	找不到 CNI 通訊端
NCP01008	CNI 連線失敗
NCP01009	CNI 版本不相符
NCP01010	CNI 訊息接收失敗
NCP01011	CNI 訊息傳輸失敗
NCP01012	Hyperbus 連線失敗
NCP01013	Hyperbus 版本不相符
NCP01014	Hyperbus 訊息接收失敗
NCP01015	Hyperbus 訊息傳輸失敗
NCP01016	GARP 傳送失敗
NCP01017	介面設定失敗

nsx-kube-proxy 錯誤碼

錯誤碼	說明
NCP02001	Proxy 無效的閘道連接埠
NCP02002	Proxy 命令失敗
NCP02003	Proxy 驗證失敗

CLI 錯誤碼

錯誤碼	說明
NCP03001	CLI 啟動失敗
NCP03002	CLI 通訊端建立失敗
NCP03003	CLI 通訊端例外狀況
NCP03004	CLI 用戶端要求無效
NCP03005	CLI 伺服器傳輸失敗
NCP03006	CLI 伺服器接收失敗
NCP03007	執行 CLI 命令失敗

Kubernetes 錯誤碼

錯誤碼	說明
NCP05001	Kubernetes 連線失敗
NCP05002	Kubernetes 組態無效
NCP05003	Kubernetes 要求失敗
NCP05004	找不到 Kubernetes 金鑰
NCP05005	找不到 Kubernetes 類型
NCP05006	Kubernetes 監看程式例外狀況
NCP05007	Kubernetes 資源長度無效
NCP05008	Kubernetes 資源類型無效
NCP05009	Kubernetes 資源處理失敗
NCP05010	Kubernetes 服務處理失敗
NCP05011	Kubernetes 端點處理失敗
NCP05012	Kubernetes 入口處理失敗
NCP05013	Kubernetes 網路原則處理失敗
NCP05014	Kubernetes 節點處理失敗
NCP05015	Kubernetes 命名空間處理失敗

錯誤碼	說明
NCP05016	Kubernetes 網繭處理失敗
NCP05017	Kubernetes 密碼處理失敗
NCP05018	Kubernetes 預設後端失敗
NCP05019	Kubernetes 不支援的比對運算式
NCP05020	Kubernetes 狀態更新失敗
NCP05021	Kubernetes 註解更新失敗
NCP05022	找不到 Kubernetes 命名空間快取
NCP05023	找不到 Kubernetes 密碼
NCP05024	Kubernetes 預設後端正在使用中
NCP05025	Kubernetes 負載平衡器服務處理失敗

OpenShift 錯誤碼

錯誤碼	說明
NCP07001	OC 路由處理失敗
NCP07002	OC 路由狀態更新失敗