

適用於 OpenShift 的 NSX Container Plug-in - 安裝和 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

Copyright © 2017–2019 VMware, Inc. 保留所有權利。 [版權與商標資訊](#)。

目錄

適用於 OpenShift 的 NSX-T Container Plug-in - 安裝和管理指南 4

1 NSX-T Container Plug-in 概觀 5

相容性需求 6

安裝概觀 6

升級 NCP 7

2 設定 NSX-T 資源 8

設定 NSX-T 資源 8

建立和設定第 0 層邏輯路由器 10

3 在 OpenShift 環境中安裝 NCP 12

部署 OpenShift 虛擬機器 12

準備 Ansible 主機檔案 12

使用單一 Playbook 安裝 NCP 和 OpenShift 14

安裝 CNI 外掛程式、OVS 和 NCP Docker 映像 15

安裝 OpenShift Container Platform 17

執行 NCP 和 NSX 節點代理程式 17

設定注意事項 19

4 在裸機環境中安裝 NCP 22

安裝 NSX-T Data Center CNI 外掛程式 22

設定 OpenShift 節點的 NSX-T Data Center 網路 22

安裝 NSX 節點代理程式 23

nsx-node-agent-ds.yml 中適用於 ncp.ini 的 Configmap 24

安裝 NSX-T Container Plug-in 27

ncp-rc.yml 中適用於 ncp.ini 的 Configmap 29

5 負載平衡 35

設定負載平衡 35

6 管理 NSX-T Container Plug-in 41

從 NSX Manager GUI 管理 IP 區塊 41

從 NSX Manager GUI 檢視 IP 區塊子網路 42

CIF 連結邏輯連接埠 42

CLI 命令 43

錯誤碼 54

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-T 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

NSX-T 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](#)
- [安裝概觀](#)
- [升級 NCP](#)

相容性需求

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

軟體產品	版本
NSX-T Data Center	2.2、2.3
容器主機虛擬機器的 Hypervisor	<ul style="list-style-type: none"> ■ 支援的 vSphere 版本 ■ RHEL KVM 7.4、7.5
容器主機作業系統	RHEL 7.4、7.5
平台即服務	OpenShift 3.9、3.10
Container Host Open vSwitch	2.9.1 (隨附於 NSX-T 2.3 和 2.2)

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 建立網蔴的 IP 區塊。
- 6 建立 SNAT (來源網路位址轉譯) 的 IP 集區。
- 7 部署 OpenShift 虛擬機器。
- 8 準備 Ansible 主機檔案。
- 9 (選項 1) 使用單一 Playbook 安裝 NCP 和 OpenShift。
(選項 2) 安裝 CNI 外掛程式、OVS (Open vSwitch) 和 NCP Docker 映像。然後，安裝 OpenShift Container Platform。
- 10 執行 NCP 和 NSX 節點代理程式。

如果您使用提供的 Playbook 安裝 NCP，則不需要步驟 2 至 6。請參閱[使用單一 Playbook 安裝 NCP 和 OpenShift](#)與[安裝 CNI 外掛程式、OVS 和 NCP Docker 映像](#)。

升級 NCP

若要將 NCP 升級至 2.3.0，請執行下列步驟。

- 1 升級 CNI RPM 套件、NSX 節點代理程式 DaemonSet 和 NCP ReplicationController。
- 2 (選擇性) 將 NSX-T Data Center 升級至 2.3。

NCP 2.3.0 支援 NSX-T 2.2，但您也可以升級至 NSX-T Data Center 2.3。

設定 NSX-T 資源

您必須建立 NSX-T Data Center 資源以提供 OpenShift 節點所需的網路。您可以手動使用 NSX Manager GUI 來設定這些資源，或使用 Ansible Playbook 來自動化程序。

本節說明如何手動建立 NSX-T 資源。如需自動化程序，請參閱[安裝 CNI 外掛程式](#)、[OVS](#) 和 [NCP Docker 映像](#)。

本章包含以下主題：

- [設定 NSX-T 資源](#)
- [建立和設定第 0 層邏輯路由器](#)

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。

使用組態檔 `ncp.ini` 中的 UUID 或名稱設定 NSX-T Data Center 資源。

覆疊傳輸區域

登入 NSX Manager 並導覽至 **網狀架構 > 傳輸區域**。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 並導覽至 **網路 > 路由 > 路由器**。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。透過登入 NSX Manager 並導覽至 **網路 > 交換 > 交換器**，可以建立邏輯交換器。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 並導覽至 **網路 > IPAM**，以建立一或多個 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 並導覽至 **詳細目錄 > 群組 > IP 集區**，以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。從 NCP 2.3.1 開始，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，導覽至**安全性 > Distributed Firewall > 一般**，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。

建立和設定第 0 層邏輯路由器

第 0 層邏輯路由器會將 Kubernetes 節點連線至外部網路。

程序

- 1 從瀏覽器登入 NSX Manager，網址為 `https://nsx-manager-ip-address`。

- 2 導覽至**網路 > 路由 > 路由器**，然後按一下**新增 > 第 0 層路由器**。
- 3 輸入名稱和 (選用) 說明。
- 4 從下拉式功能表中選取現有的 **Edge** 叢集，用以支援此 第 0 層邏輯路由器。
- 5 選取高可用性模式。
選取主動備用。
- 6 按一下**儲存**。
新的邏輯路由器會顯示為連結。
- 7 按一下邏輯路由器連結。
- 8 按一下**路由 > 路由重新分配**。
- 9 按一下**新增**來新增重新分配準則。
針對資源，在路由的 (非 NAT) 拓撲中，選取 **NSX 靜態**。在 NAT 拓撲中，選取**第 0 層 NAT**。
- 10 按一下**儲存**。
- 11 按一下新建立的路由器。
- 12 按一下**組態 > 路由器連接埠**
- 13 按一下**新增**以新增上行連接埠。
- 14 選取傳輸節點。
- 15 選取先前建立的邏輯交換器。
- 16 指定您外部網路中的 IP 位址。
- 17 按一下**儲存**。
新的邏輯路由器會顯示為連結。

在 OpenShift 環境中安裝 NCP

本章說明如何安裝和設定 NSX-T Container Plug-in (NCP) 和 OpenShift。

本章包含以下主題：

- 部署 OpenShift 虛擬機器
- 準備 Ansible 主機檔案
- 使用單一 Playbook 安裝 NCP 和 OpenShift
- 安裝 CNI 外掛程式、OVS 和 NCP Docker 映像
- 安裝 OpenShift Container Platform
- 執行 NCP 和 NSX 節點代理程式
- 設定注意事項

部署 OpenShift 虛擬機器

在安裝 NSX-T Container Plug-in 之前，必須先安裝 OpenShift。您必須部署至少一個主機。

如需詳細資訊，請參閱 <https://docs.openshift.com>。

後續步驟

準備 Ansible 主機檔案。請參閱[準備 Ansible 主機檔案](#)。

準備 Ansible 主機檔案

Ansible 主機檔案會定義 OpenShift 叢集中的節點。

程序

- 1 透過 <https://github.com/vmware/nsx-integration-for-openshift> 複製 NCP GitHub 存放庫。hosts 檔案位於 openshift-ansible-nsx 目錄中。必須將 hosts 檔案保留在 openshift-ansible-nsx 目錄中。某些 Playbook 假設這是 hosts 檔案的路徑。

- 2 在 [主機] 和 [節點] 區段中，指定 OpenShift 虛擬機器的主機名稱和 IP 位址。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
```

請注意，`openshift_ip` 會識別叢集內部 IP，且若要使用的介面不是預設值，則必須進行設定。主節點中的 `ncp` 相關角色會使用 `single_master` 變數執行一次特定工作，例如 **NSX-T Data Center** 管理平面資源設定。

- 3 設定 SSH 存取，使執行 Ansible 角色所在的節點 (通常為主節點) 不需密碼即可直接存取所有節點。

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

- 4 更新 [OSEv3:vars] 區段。您可以在 OpenShift Container Platform 說明文件中找到所有參數的相關詳細資料以進行進階安裝 (在 <https://docs.openshift.com> 中搜尋「進階安裝」)。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 確認您可以連線至所有主機：

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

結果看起來應如下所示。若非如此，請解決連線問題。

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

後續步驟

安裝 CNI 外掛程式和 OVS。請參閱[安裝 CNI 外掛程式](#)、[OVS](#) 和 [NCP Docker 映像](#)。

使用單一 Playbook 安裝 NCP 和 OpenShift

您可以使用單一 Playbook 安裝 NCP 和 OpenShift，或在單獨的步驟中執行安裝。

單一 Ansible Playbook `install.yaml` 可執行下列工作：

- NCP 準備
- OpenShift 安裝

■ NCP 安裝

或者，您可以按照以下兩節中的指示安裝 NCP 和 OpenShift：[安裝 CNI 外掛程式、OVS 和 NCP Docker 映像](#)和[安裝 OpenShift Container Platform](#)。

執行 `install.yaml` Playbook 之前，設定 `ncp_prep` 和 `ncp` Plabook 角色的必要和可選參數。[安裝 CNI 外掛程式、OVS 和 NCP Docker 映像](#)中說明了參數。

下列命令會執行 Playbook：

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

安裝 CNI 外掛程式、OVS 和 NCP Docker 映像

容器網路介面 (CNI) 外掛程式、Open vSwitch (OVS) 和 NCP Docker 映像必須安裝在 OpenShift 節點上。此安裝可藉由執行 Ansible Playbook 來執行。

備註 如果您使用單一 Playbook 安裝 NCP 和 OpenShift，則不需要執行此步驟。請參閱[使用單一 Playbook 安裝 NCP 和 OpenShift](#)。

Playbook 中包含為節點設定 NSX-T 資源的指示。您也可以依照第 2 章，[設定 NSX-T 資源](#)中的說明，以手動方式設定 NSX-T Data Center 資源。當 Playbook 執行時，參數 `perform_nsx_config` 表示是否要設定資源。

程序

- 1 更新 `roles/ncp_prep/default/main.yaml` 和 `roles/nsx_config/default/main.yaml` 中的參數值，包括可從中下載 CNI 外掛程式 RPM、OVS 及其對應核心模組 RPM 的 URL。此外，`uplink_port` 是節點虛擬機器上之上行連接埠 VNIC 的名稱。剩餘變數則與 NSX-T Data Center 管理平面組態有關。

需要指定的參數：

- `perform_nsx_config`：是否執行資源設定。如果要手動執行設定，且不會執行 `nsx_config` 指令碼，請將其設為 `false`。
- `nsx_manager_ip`：NSX Manager 的 IP
- `nsx_edge_cluster_name`：第 0 層路由器所要使用之 Edge 叢集的名稱
- `nsx_transport_zone_name`：覆疊傳輸區域的名稱
- `os_node_name_list`：節點名稱的逗點分隔清單
例如，`node1,node2,node3`
- `subnet_cidr`：IP 管理員的 CIDR 位址將指派給節點上的 `br-int`
- `vc_host`：vCenter Server 的 IP 位址
- `vc_user`：vCenter Server 管理員的使用者名稱
- `vc_password`：vCenter Server 管理員的密碼

- **vms**: 虛擬機器名稱的逗點分隔清單。順序必須符合 **os_node_name_list**。

下列參數具有預設值。您可以視需要進行修改。

- **nsx_t0_router_name**: 叢集的第 0 層邏輯路由器的名稱。預設值: **t0**
- **pod_ipblock_name**: 網繭之 IP 區塊的名稱。預設值: **podIPBlock**
- **pod_ipblock_cidr**: 此 IP 區塊的 CIDR 位址。預設值: **172.20.0.0/16**
- **snat_ippool_name**: SNAT 之 IP 區塊的名稱。預設值為 **externalIP**。
- **snat_ippool_cidr**: 此 IP 區塊的 CIDR 位址。預設值: **172.30.0.0/16**
- **start_range**: 針對此 IP 集區指定的 CIDR 的起始 IP 位址。預設值: **172.30.0.1**
- **end_range**: 針對此 IP 集區指定的 CIDR 的結束 IP 位址。預設值: **172.30.255.254**
- **os_cluster_name**: OpenShift 叢集的名稱。預設值: **occl-one**
- **nsx_node_ls_name**: 連線至節點之邏輯交換器的名稱。預設值: **node_ls**
- **nsx_node_lr_name**: 交換器 **node_ls** 之邏輯路由器的名稱。預設值: **node_lr**

nsx-config Playbook 僅支援建立一個 IP 集區和一個 IP 區塊。如果您想要更多，則必須手動建立。

2 切換至 openshift-ansible-nsx 目錄，並執行 ncp_prep 角色。

```
ansible-playbook -i /PATH/T0/HOSTS/hosts ncp_prep.yaml
```

Playbook 中包含執行下列動作的指示：

- 下載 CNI 外掛程式安裝檔案。

檔案名稱為 **nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm**，其中，**xxxxxxx** 是組建編號。

- 安裝 CNI 外掛程式安裝檔案。

此外掛程式會安裝在 **/opt/cni/bin** 中。CNI 組態檔 **10.net.conf** 會複製到 **/etc/cni/net.d**。rpm 也將安裝回送外掛程式的組態檔 **/etc/cni/net.d/99-loopback.conf**。

- 下載並安裝 OVS 安裝檔案。

檔案名稱為 **openvswitch-2.9.1.xxxxxxx-1.x86_64.rpm** 和 **openvswitch-kmod-2.9.1.xxxxxxx-1.el7.x86_64.rpm**，其中，**xxxxxxx** 是組建編號。

- 建立 **br-int** 執行個體 (若尚未建立)。

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

- 將連結至節點邏輯交換器的網路介面 (**node-if**) 新增至 **br-int**。

- 確定 **br-int** 和 **node-if link** 的狀態皆為開啟。

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


- 更新網路組態檔，以確保網路介面會在重新開機後開啟。
- 下載 NCP tar 檔案，然後從 tar 檔案載入 Docker 映像。
- 下載 ncp-rbac yaml 檔案，並將 apiVersion 變更為 v1。
- 在 NSX-T Data Center 中建立邏輯拓撲和相關資源，並在其上建立標記以便 NCP 可以辨識。
- 使用 NSX-T Data Center 資源資訊更新 ncp.ini。

後續步驟

安裝 OpenShift Container Platform。請參閱[安裝 OpenShift Container Platform](#)。

安裝 OpenShift Container Platform

OpenShift Container Platform (OCP) 是可整合 Docker 和 Kubernetes 的平台即服務 (PaaS) 產品。

備註 如果您使用單一 Playbook 安裝 NCP 和 OpenShift，則不需要執行此步驟。請參閱[使用單一 Playbook 安裝 NCP 和 OpenShift](#)。

如需安裝 OCP 的相關資訊，請參閱 <https://docs.openshift.com>。

後續步驟

執行 NCP 和 NSX 節點代理程式。請參閱[執行 NCP 和 NSX 節點代理程式](#)。

執行 NCP 和 NSX 節點代理程式

設定並執行 NCP 和 NSX 節點代理程式。

程序

- 1 編輯 roles/ncp/defaults/main.yaml，並指定 OpenShift API 伺服器 IP、NSX Manager IP，以及用來下載 NCP ReplicationController yaml 和 nsx-node-agent DaemonSet yaml 的 URL。
- 2 從 openshift-ansible-nsx 目錄執行 ncp 角色：

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

ncp 角色會執行下列步驟：

- 檢查 nsx-system 專案是否存在，若不存在則加以建立。

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

- 下載 ncp-rbac yaml 檔案，並將 apiVersion 變更為 v1。
- 為 NCP 網繭建立服務帳戶，然後建立指定 NCP 可存取資源的叢集角色，並將此叢集角色與 NCP 服務帳戶繫結。

- 為 **nsx-node-agent** 網繭建立服務帳戶，然後建立指定節點代理程式可存取資源的叢集角色，並將此叢集角色與節點代理程式服務帳戶繫結。

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

- 取得與上述服務帳戶相關聯的 **Token**，並將其儲存在 **/etc/nsx-ujo/<service_account>_token** 下：

```
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
```

- 下載 NCP 的 **SecurityContextConstraint (SCC)** yaml 檔案 **ncp-os-scc.yml**，並根據 **yaml** 建立 **SCC**。

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

SCC **yaml** 檔案將 **SELinux** 類型指定為 **spc_t**，以確保 **NCP/nsx-node-agent** 具有 **SELinux** 下的存取權限。亦即，

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

在 **SCC** **yaml** 檔案中 **seLinuxContext** 的 **seLinuxOptions** 下，**SELinux** 以標籤為基礎的存取控制層級設定為 **s0:c0:c1023**，可允許 **ncp/node-agent** 從不同的檔案類別存取目標。

- 將 **SCC** 新增至建立 **NCP** 和 **NSX** 節點代理程式網繭的使用者。例如，將 **SCC** 新增至目前專案中的預設使用者：

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

- 將 **SCC** 新增至 **NCP** 和 **NSX** 節點代理程式服務帳戶：

```
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
```

- 下載 **NCP ReplicationController (RC)** 和 **nsx-node-agent DaemonSet (DS)** 的 **yaml** 檔案，並更新 **ConfigMap**。
- 下載並載入 **NCP** 映像 (**nsx-node-agent** 會使用相同映像)。
- 設定服務帳戶，並設定 **NCP** 和 **nsx_node_agent** 所需的 **SecurityContextConstraint**。

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

備註 NCP 開啟與 Kubernetes API 伺服器的持續性 HTTP 連線，以監看 Kubernetes 資源的生命週期事件。如果 API 伺服器失敗或網路失敗導致 NCP 的 TCP 連線失效，您必須重新啟動 NCP，使其能夠重新建立與 API 伺服器的連線。否則，NCP 將會錯過新事件。

設定注意事項

設定 OpenShift 和 NCP 之前，請記下以下資訊。

- 網繭的標籤不得超過 11 個，而命名空間的標籤則不得超過 12 個。
- 新增供 OpenShift 內部使用的標籤 (例如，在索引鍵中具有首碼 openshift.io 的標籤) 將被 NCP 忽略，因此使用者將不會看見建立在相關 NSX 資源上的對應標籤。以下是 OpenShift 所使用的標籤首碼清單，您應避免使用以下列任一項開頭的標籤索引鍵：

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

- 節點將需要存取網繭，例如，用來進行 Kubelet 健全狀況檢查。請確定主機管理介面能夠存取網繭網路。
- 攻擊者可以利用 Linux 功能 NET_ADMIN 和 NET_RAW 來入侵網繭網路。您應停用不受信任容器的這兩項功能。依預設並不會對受限和 anyuid SCC 授與 NET_ADMIN。請留意是否有任何 SCC 明確啟用了 NET_ADMIN，或是允許網繭在特殊權限模式下執行。此外，針對不受信任的容器，請根據 anyuid SCC (舉例而言) 建立移除 NET_RAW 功能的個別 SCC。此作業可藉由將 NET_RAW 新增至 SCC 定義中的「requiredDropCapabilities」清單來完成。
- 允許網繭/容器中的根存取 (限用於測試)。下列命令將需要在您目前登入之 oc 專案的所有網繭中進行根存取。

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

- 設定 (新增) OpenShift 登錄。

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

- 刪除 OpenShift 登錄

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

- 其中缺少的 **IPtables** 防火牆規則可允許來自 **Docker** 預設橋接器容器的 **DNS** 要求傳送至節點上的 **dnsmasq** 程序。您必須手動加以開啟。請編輯 `/etc/sysconfig/iptables`，並在檔案底部的 **COMMIT** 前面新增下列規則：

```
-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
```

- 重新啟動 **Iptables**、**Docker** 和 **Origin** 節點 (重新啟動 **kube-proxy** 和 **kubelet**)。

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

- OpenShift** 的內部 **Docker** 登錄必須能夠使用非 **TLS** 才能讓 **OpenShift** 正常運作。一般而言，**OpenShift Ansible** 安裝程式應該會自動新增，但這目前似乎沒有作用。請編輯 `/etc/sysconfig/docker` 並新增：

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

- 重新啟動 **Docker**。

```
systemctl restart docker
```

- NCP** 的網路原則支援與 **Kubernetes** 所提供的支援相同，視 **OpenShift** 使用的 **Kubernetes** 版本而定。
 - OpenShift 3.9** - 網路原則中的規則子句可包含 **namespaceSelector**、**podSelector** 和 **ipBlock** 中的最多一個選取器。
- Kubernetes API** 伺服器不會執行網路原則規格驗證。可能會建立無效的網路原則。**NCP** 會拒絕此類網路原則。如果您更新該網路原則使其有效，**NCP** 仍不會處理該網路原則。您必須刪除該網路原則，並重新建立一個具有有效規格的網路原則。
- 特定版本的 **Kubernetes** 具有與 **subPath** 相關的問題 (請參閱 <https://github.com/kubernetes/kubernetes/issues/61076>)。如果 **OpenShift** 版本不包含此問題的修正，**NCP** 網繭建立會失敗並顯示錯誤 **CreateContainerConfigError: 無法準備 volumeMount 的子路徑**。您可以從 **NCP yaml** 移除對 **subPath** 的使用，以解決此問題。具體來說，移除包含 **subPath**：**ncp.ini** 的行，並使用下列內容取代 **volumes** 的組態：

```
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
```

這項變更的負面影響是整個 `/etc/nsx-ujo` 目錄變成唯讀狀態。如此一來，使用憑證和私密金鑰連線 NSX-T 將無法運作，因為 NCP 無法在 `/etc/nsx-ujo` 下建立暫存檔以將憑證和私密金鑰移到單一檔案。

- 如果您執行或升級至 OpenShift 3.10 叢集，請注意下列項目：
 - 您必須指定特定於 OpenShift 3.10 叢集的節點群組的組態。必須在詳細目錄 `hosts` 檔案中提供節點組態對應組態。
 - 必須將詳細目錄 `hosts` 檔案中 `[nodes]` 群組中定義的所有主機指派給節點群組名稱。
 - 從 Ansible 指導手冊升級 OpenShift 叢集可能會導致網路中斷。請確保在 `openshift-ansible` 存放庫上新增修補程式 (<https://github.com/openshift/openshift-ansible/pull/8016/files#diff-2386e21861da3f95091dbb27d72ca366>)，以移除停止/解除安裝 Open vSwitch 套件。
- 從 OpenShift 3.10 開始，`kube-proxy` 已從 `openshift-node` 服務移到 `DaemonSet`。依預設，不會再啟動它。執行下列步驟以手動啟動 `kube-proxy` (假設已複製 `openshift-ansible` 存放庫)：
 - 前往 `openshift-ansible` 目錄，在 `[defaults]` 下設定下列項目：

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

- 透過下列項目在 `playbooks` 目錄中建立 `create_proxy.yaml` 檔案：

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

- 執行 Playbook：

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

您將看到錯誤訊息，指示部分作業失敗。可以忽略這些訊息。您可以透過執行命令 `oc get po --all-namespaces` 驗證結果。

在裸機環境中安裝 NCP

在裸機環境中安裝 NSX-T Container Plug-in (NCP) 的步驟與在非裸機環境中安裝 NCP 的步驟類似。本節說明了存在差異的步驟。

本章包含以下主題：

- 安裝 NSX-T Data Center CNI 外掛程式
- 設定 OpenShift 節點的 NSX-T Data Center 網路
- 安裝 NSX 節點代理程式
- nsx-node-agent-ds.yml 中適用於 ncp.ini 的 Configmap
- 安裝 NSX-T Container Plug-in
- ncp-rc.yml 中適用於 ncp.ini 的 Configmap

安裝 NSX-T Data Center CNI 外掛程式

NSX-T Data Center CNI 外掛程式必須安裝在 OpenShift 節點上。

程序

- 1 下載您 Linux 發行版適用的安裝檔案。
檔案名稱為 nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm，其中，xxxxxxx 是組建編號。
- 2 安裝在步驟 1 中下載的 rpm 檔案。
此外掛程式會安裝在 /opt/cni/bin 中。CNI 組態檔 10.net.conf 會複製到 /etc/cni/net.d。rpm 也將安裝回送外掛程式的組態檔 /etc/cni/net.d/99-loopback.conf。

設定 OpenShift 節點的 NSX-T Data Center 網路

本節說明如何設定 OpenShift 主要及運算節點的 NSX-T Data Center 網路。

每個節點必須以作業系統類型 RHEL Container 向 NSX Manager 登錄。節點的管理介面可用來加入 OpenShift 叢集，並且可以在 NSX-T Data Center 網狀架構上，也可以不在其上。其他介面提供網際網路，且必須在 NSX-T Data Center 網狀架構上。

對應的傳輸節點必須具有下列標記：

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

您可以透過導覽至**網狀架構 > 節點**從 NSX Manager GUI 識別 OpenShift 節點的傳輸節點。

如果 OpenShift 節點名稱有所變更，則您必須更新標記 `ncp/node_name` 並重新啟動 NCP。您可以使用下列命令來取得節點名稱：

```
oc get nodes
```

如果您在 NCP 執行時將節點新增至叢集，則在執行 `oc cluster add` 命令之前必須將標記新增至傳輸節點。否則，新的節點將不具有網路連線。如果標記不正確或遺漏，則您可以採取下列步驟來解決問題：

- 將正確的標記套用到傳輸節點。
- 重新啟動 NCP。

安裝 NSX 節點代理程式

NSX 節點代理程式是每個網繭用來執行兩個容器的 **DaemonSet**。一個容器會執行 NSX 節點代理程式，其主要工作是管理容器網路介面。它會與 CNI 外掛程式和 Kubernetes API 伺服器互動。另一個容器會執行 NSX kube-proxy，而其唯一的工作是將叢集 IP 轉譯為網繭 IP，以實作 Kubernetes 服務擷取。它會實作與上游 kube-proxy 相同的功能。

程序

- 1 下載 NCP Docker 映像。

檔案名稱為 `nsx-ncp-xxxxxxx.tar`，其中，`xxxxxxx` 是組建編號。

- 2 下載 NSX 節點代理程式 DaemonSet yaml 範本。

檔案名稱為 `nsx-node-agent-ds.yml`。您可以編輯此檔案，或將其用作您自己的範本檔範例。

- 3 將 NCP Docker 映像載入至您的映像登錄。

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

- 4 編輯 `nsx-node-agent-ds.yml`。

將映像名稱變更為已載入的映像名稱。

進行下列變更：

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

取消註解下列幾行：

```
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
```

備註 在 yaml 檔案中，您必須指定為 `ncp.ini` 產生的 ConfigMap 必須掛接為「唯讀」磁碟區。下載的 yaml 檔案已具有此規格而不應變更。

5 使用下列命令建立 NSX 節點代理程式 DaemonSet。

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

nsx-node-agent-ds.yml 中適用於 ncp.ini 的 Configmap

範例 yaml 檔案 `nsx-node-agent-ds.yml` 包含的 ConfigMap 適用於 NSX 節點代理程式的組態檔 `ncp.ini`。您可以指定此 ConfigMap 區段包含的參數來自訂節點代理程式安裝。

您所下載的 `nsx-node-agent-ds.yml` 範例具有下列 `ncp.ini` 資訊：

```
# 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>
```

安裝 NSX-T Container Plug-in

NSX-T Container Plug-in (NCP) 會以 Docker 映像的形式提供。NCP 應執行於基礎結構服務的節點上。不建議在主節點上執行 NCP。

程序

1 下載 NCP Docker 映像。

檔案名稱為 `nsx-ncp-xxxxxxx.tar`，其中，`xxxxxxx` 是組建編號。

2 下載 NCP ReplicationController yaml 範本。

檔案名稱為 `ncp-rc.yml`。您可以編輯此檔案，或將其用作您自己的範本檔範例。

3 將 NCP Docker 映像載入至您的映像登錄。

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

4 編輯 `ncp-rc.yml`。

將節點類型設定為裸機。

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

將映像名稱變更為已載入的映像名稱。

指定 `nsx_api_managers` 參數。此版本支援單一 Kubernetes 節點叢集和單一 NSX Manager 執行個體。例如：

```
nsx_api_managers = 192.168.1.180
```

(選用) 在 `[nsx_v3]` 區段中指定參數 `ca_file`。該值應為用來驗證 NSX Manager 伺服器憑證的 CA 服務包檔案。若未設定，則系統將會使用系統根 CA。

指定參數 `nsx_api_cert_file` 和 `nsx_api_private_key_file` 以便用於 NSX-T Data Center 的驗證。

`nsx_api_cert_file` 是 PEM 格式之用戶端憑證檔案的完整路徑。此檔案的內容應如下所示：

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

`nsx_api_private_key_file` 是 PEM 格式之用戶端私密金鑰檔案的完整路徑。此檔案的內容應如下所示：

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

如果入口控制器設定為在 NAT 模式中執行，請指定參數 `ingress_mode = nat`。

依預設，子網路首碼 24 會用於所有從網蔴邏輯交換器之 IP 區塊配置的子網路。若要使用不同的子網路大小，請更新 `[nsx_v3]` 區段中的 `subnet_prefix` 選項。

備註 在 yaml 檔案中，您必須指定為 `ncp.ini` 產生的 ConfigMap 應掛接為「唯讀」磁碟區。下載的 yml 檔案已具有此規格而不應變更。

5 建立 NCP ReplicationController。

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

備註 NCP 開啟與 Kubernetes API 伺服器的持續性 HTTP 連線，以監看 Kubernetes 資源的生命週期事件。如果 API 伺服器失敗或網路失敗導致 NCP 的 TCP 連線失效，您必須重新啟動 NCP，使其能夠重新建立與 API 伺服器的連線。否則，NCP 將會錯過新事件。

在 NCP 複寫控制站定期更新期間，請勿將容器主機重新開機。如果主機因任何原因重新開機，則在重新開機後，您可能會看到兩個 NCP 網繭。在此情況下，執行下列操作：

- 刪除其中一個 NCP 網繭。刪除任一網繭皆可。例如，

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

- 刪除命名空間 `nsx-system`。例如，

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

ncp-rc.yml 中適用於 ncp.ini 的 Configmap

範例 YAML 檔案 `ncp-rc.yml` 包含的 ConfigMap 適用於組態檔 `ncp.ini`。您在安裝 NCP 之前，必須指定此 ConfigMap 區段包含的參數，如上一個區段中所述。

您所下載的 `ncp-rc.yml` 範例具有下列 `ncp.ini` 資訊：

```
# 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 unset 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_password 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_password 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>
```

負載平衡

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。

從 NCP 2.3.1 開始，`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
```

備註 如果您同時設定第 4 層和第 7 層負載平衡器，則可以將 `l4_persistence` 或 `l7_persistence` (或兩者同時) 設定為 `source_ip`，但您無法將 `l4_persistence` 設定為 `source_ip` 以及將 `l7_persistence` 設定為 `cookie`。如果誤將 `l4_persistence` 設定為 `source_ip` 以及將 `l7_persistence` 設定為 `cookie`，負載平衡器服務將無法運作。若要解決此問題，您必須刪除入口資源和負載平衡器服務、變更持續性設定、重新啟動 NCP，然後重新建立入口資源和負載平衡器服務。

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 管理指南》。

小型 NSX-T Data Center 負載平衡器支援下列內容：

- 10 部 NSX-T 虛擬伺服器。
- 10 個 NSX-T 集區。
- 30 個 NSX-T 集區成員。
- 8 個用於負載平衡器服務的連接埠。
- 由負載平衡器服務和路由資源定義的總共 10 個連接埠。
- 由負載平衡器服務和路由資源引用的總共 30 個端點。

中型 NSX-T Data Center 負載平衡器支援下列內容：

- 100 部 NSX-T 虛擬伺服器。
- 100 個 NSX-T 集區。
- 300 個 NSX-T 集區成員。
- 98 個用於負載平衡器服務的連接埠。
- 由負載平衡器服務和路由資源定義的總共 100 個連接埠。
- 由負載平衡器服務和路由資源引用的總共 300 個端點。

大型 NSX-T Data Center 負載平衡器支援下列內容：

- 1000 部 NSX-T 虛擬伺服器。
- 1000 個 NSX-T 集區。
- 3000 個 NSX-T 集區成員。
- 998 個用於負載平衡器服務的連接埠。
- 由負載平衡器服務和路由資源定義的總共 1000 個連接埠。
- 由負載平衡器服務和路由資源引用的總共 3000 個端點。

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。

- 從 NCP 2.3.1 開始，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。此功能需要 NSX-T Data Center 2.3 或更新版本。

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-uj0` 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-uj0/lb-default.crt
    - name: lb-priv-key
      # Mount path must match nsx_v3 option "lb_priv_key_path"
      mountPath: /etc/nsx-uj0/lb-default.key
  volumes:
  ...
  - name: lb-default-cert
    hostPath:
      path: /etc/nsx-uj0/lb-default.crt
  - name: lb-priv-key
    hostPath:
      path: /etc/nsx-uj0/lb-default.key
```


管理 NSX-T Container Plug-in

您可以從 NSX Manager GUI 或從命令列介面 (CLI) 來管理 NSX-T 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 vmknics。

本章包含以下主題：

- 從 NSX Manager GUI 管理 IP 區塊
- 從 NSX Manager GUI 檢視 IP 區塊子網路
- CIF 連結邏輯連接埠
- CLI 命令
- 錯誤碼

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-T 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 6-1. NCP 容器的 CLI 命令

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

表格 6-2. 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 6-3. 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：

```
kubecall> 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：

```
kubecall> 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:

```
kubecall> 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:

```
kubecall> 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

```

■ 取得網蘭的內部快取

```

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
```

```
kubenset> 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：

```
kubenset>get support-bundle file foo
Bundle file foo created in tgz format
kubenset>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：

```
kubecall> 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：

```
kubecall> 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：

```
kubecall> 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：

```
kubecall> 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	沒有網蔴組態
NCP01005	網蔴設定失敗
NCP01006	網蔴取消設定失敗
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