

適用於 Kubernetes 和 Cloud Foundry 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 Kubernetes 和 Cloud Foundry 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 13

3 在 Kubernetes 環境中安裝 NCP 15

安裝 NSX-T Data Center CNI 外掛程式 15

安裝和設定 OVS 16

設定 Kubernetes 節點的 NSX-T Data Center 網路 17

安裝 NSX 節點代理程式 18

nsx-node-agent-ds.yml 中適用於 ncp.ini 的 Configmap 19

安裝 NSX-T Container Plug-in 22

ncp-rc.yml 中適用於 ncp.ini 的 Configmap 24

在 NCP 網繭中掛接 PEM 編碼憑證和私密金鑰 29

在 NCP 網繭中掛接憑證檔案 29

設定 Syslog 30

安全考量事項 40

設定網路資源的相關提示 43

4 在 Pivotal Cloud Foundry 環境中安裝 NCP 45

在 Pivotal Cloud Foundry 環境中安裝 NCP 45

5 負載平衡 48

設定負載平衡 48

6 管理 NSX-T Container Plug-in 55

從 NSX Manager GUI 管理 IP 區塊 55

從 NSX Manager GUI 檢視 IP 區塊子網路 56

CIF 連結邏輯連接埠 56

CLI 命令 57

錯誤碼 75

適用於 Kubernetes 和 Cloud Foundry 的 NSX-T Container Plug-in - 安裝和管理指南

本指南說明了如何安裝和管理 NSX-T Container Plug-in (NCP) 以整合 NSX-T Data Center 與 Kubernetes，以及整合 NSX-T Data Center 與 Pivotal Cloud Foundry (PCF)。

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、Kubernetes 和 Pivotal Cloud Foundry 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) 產品，例如 OpenShift 和 Pivotal Cloud Foundry。本指南說明了使用 Kubernetes 和 Pivotal Cloud Foundry 設定 NCP。

NCP 的主要元件會在容器中執行，且會與 NSX Manager 和 Kubernetes 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 Kubernetes 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 Kubernetes 叢集自動建立 NSX-T Data Center 邏輯拓撲，且為每個 Kubernetes 命名空間建立單獨的邏輯網路。
- 將 Kubernetes 網卡連線至邏輯網路，並配置 IP 和 MAC 位址。
- 支援網路位址轉譯 (NAT) 且為每個 Kubernetes 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 Data Center 分散式防火牆實作 Kubernetes 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。
 - 支援在其他命名空間中選取網卡。
- 實作 ClusterIP 類型的 Kubernetes 服務和 LoadBalancer 類型的服務。
- 透過 NSX-T 第 7 層負載平衡器實作 Kubernetes 入口。
 - 透過 TLS Edge 終止支援 HTTP 入口和 HTTPS 入口。
 - 支援入口預設後端組態。
 - 支援入口 URI 重新寫入。

- 為命名空間、網繭名稱和網繭標籤在 **NSX-T Data Center** 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** 安全群組和原則。

在此版本中，NCP 支援單一 **Kubernetes** 叢集。您可以具有使用相同 **NSX-T Data Center** 部署的多個 **Kubernetes** 叢集，每個叢集均有不同的 NCP 執行個體。

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

對於 **Kubernetes** 環境和 **Pivotal Cloud Foundry (PCF)** 環境，**NSX-T Container Plug-in (NCP)** 具有下列相容性需求。

表格 1-1. Kubernetes 環境的相容性需求

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 ■ Ubuntu KVM 16.04
容器主機作業系統	<ul style="list-style-type: none"> ■ Ubuntu 16.04、RHEL 7.4、7.5
容器協調工具	<ul style="list-style-type: none"> ■ Kubernetes 1.10 ■ Kubernetes 1.11
Container Host Open vSwitch	2.9.1 (隨附於 NSX-T 2.3 和 2.2)

表格 1-2. Cloud Foundry 環境的相容性需求

軟體產品	版本
容器主機虛擬機器的 Hypervisor	<ul style="list-style-type: none"> ■ 支援的 vSphere 版本
容器協調工具	<ul style="list-style-type: none"> ■ (僅限 NCP 2.3) Pivotal Application Service 2.1.x (2.1.0 除外) 和 Pivotal Operations Manager 2.1.x ■ Pivotal Application Service 2.2.0 和 Pivotal Operations Manager 2.2.0 ■ (僅限 NCP 2.3.1) Pivotal Application Service 2.3.x 和 Pivotal Operations Manager 2.3.x

安裝概觀

在已安裝 **Kubernetes** 的環境中，安裝和設定 **NCP** 時通常須執行下列步驟。若要成功執行這些步驟，您必須熟悉 **NSX-T Data Center** 和 **Kubernetes** 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 Kubernetes 節點連線至交換器。
- 4 建立第 0 層邏輯路由器。
- 5 建立 Kubernetes 網繭的 IP 區塊。
- 6 建立 SNAT (來源網路位址轉譯) 的 IP 集區。
- 7 在每個節點上安裝 NSX CNI (容器網路介面) 外掛程式。
- 8 在每個節點上安裝 OVS (Open vSwitch)。
- 9 設定 Kubernetes 節點的 NSX-T 網路。
- 10 將 NSX 節點代理程式安裝為 DaemonSet。
- 11 將 NCP 安裝為 ReplicationController。
- 12 在 NCP 網繭中掛接安全性憑證。

升級 NCP

若要將 NCP 升級至 2.3.0，請執行下列步驟。

- 1 如果您正在執行 Pivotal Application Service (PAS) 2.1.x，請先將 PAS 升級至 2.2.x。
- 2 對於 Pivotal 應用程式服務環境，將 NCP 或 NSX-T 圖標升級至 2.3.0。
- 3 對於 Kubernetes 或 OpenShift 環境，升級 CNI RPM/DEB 套件、NSX 節點代理程式 DaemonSet 和 NCP ReplicationController。
- 4 (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

安裝 NSX-T Container Plug-in 之前，您必須先設定特定的 NSX-T Data Center 資源。

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。

也可以建立專用於非 SNAT 命名空間 (針對 Kubernetes) 或叢集 (針對 PCF) 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。如需詳細資訊, 請參閱 ncp-rc.yml 中適用於 ncp.ini 的 Configmap。

SNAT 的 IP 集區

NSX Manager 中的 IP 集區用來配置 IP 位址, 從而用於使用 SNAT 規則轉譯網繭 IP, 以及用於使用 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。

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

從 NCP 2.3.2 開始, 您可以透過變更組態檔並重新啟動 NCP, 變更為不同的 IP 集區。

限制 SNAT IP 集區為特定的 Kubernetes 命名空間或 PCF 組織

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 Kubernetes 命名空間或 PCF 組織。

- 對於 Kubernetes 命名空間: 範圍: ncp/owner, 標記: ns:<namespace_UUID>
- 對於 PCF 組織: 範圍: ncp/owner, 標記: org:<org_UUID>

您可以使用下列其中一個命令取得命名空間或組織 UUID:

```
kubect1 get ns -o yaml
cf org <org_name> --guid
```

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或組織配置 IP 後變更標記，將不會回收這些 IP，直到 Kubernetes 服務或 PCF 應用程式的 SNAT 組態變更或 NCP 重新啟動為止。
- 命名空間和 PCF 組織擁有者標記是可選的。如果沒有這些標記，則可從 SNAT IP 集區配置 IP 給任何命名空間或 PCF 組織。

為服務設定 SNAT IP 集區

您也可以透過新增註解至服務來為服務設定 SNAT IP 集區。例如，

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

從 NCP 2.3.1 開始，ncp/snat_pool 所指定的 IP 集區必須有標記 **{"ncp/owner": cluster:<cluster>}**。

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。如果 NCP 設定 SNAT 規則時發生錯誤，將會使用 **ncp/error.snat: <error>** 標註服務。可能的錯誤如下：

- IP_POOL_NOT_FOUND - 在 NSX Manager 中找不到 SNAT IP 集區。
- IP_POOL_EXHAUSTED - SNAT IP 集區已用盡。
- IP_POOL_NOT_UNIQUE - ncp/snat_pool 所指定的集區指的是 NSX Manager 中的多個集區。
- SNAT_POOL_ACCESS_DENY - 集區的擁有者標記不符合正在傳送配置要求之服務的命名空間。
- SNAT_RULE_OVERLAPPED - 建立新的 SNAT 規則，但 SNAT 服務的網繭也屬於另一個 SNAT 服務，也就是，同一個網繭具有多個 SNAT 規則。
- (適用於 NCP 2.3.1 及更新版本) POOL_ACCESS_DENIED - ncp/snat_pool 所指定的 IP 集區沒有標記 **{"ncp/owner": cluster:<cluster>}**，或集區的擁有者標記不符合正在傳送配置要求之服務的命名空間。

請注意下列事項：

- ncp/snat_pool 所指定的集區應在服務設定之前已存在於 NSX-T Data Center 中。
- 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。
- 從 NCP 2.3.2 開始，您可以透過變更註解並重新啟動 NCP，變更為不同的 IP 集區。

為命名空間設定 SNAT IP 集區

從 NCP 2.3.2 開始，您也可以透過新增註解至命名空間來為命名空間設定 SNAT IP 集區。例如，

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns-sample
  annotations:
    ncp/snat_pool: <external IP pool ID or name>
  ...
```

NCP 將針對此命名空間設定 SNAT 規則。規則的來源 IP 為後端網蔴集。目的地 IP 是從指定的外部 IP 集區進行配置的 SNAT IP。如果 NCP 設定 SNAT 規則時發生錯誤，將會使用 `ncp/error.snat: <error>` 標註命名空間。可能的錯誤如下：

- IP_POOL_NOT_FOUND - 在 NSX Manager 中找不到 SNAT IP 集區。
- IP_POOL_EXHAUSTED - SNAT IP 集區已用盡。
- IP_POOL_NOT_UNIQUE - `ncp/snat_pool` 所指定的集區指的是 NSX Manager 中的多個集區。
- POOL_ACCESS_DENIED - `ncp/snat_pool` 所指定的 IP 集區沒有標記 `{"ncp/owner": cluster:<cluster>}`，或集區的擁有者標記不符合正在傳送配置要求的命名空間。

請注意下列事項：

- 您只能在註解中指定一個 SNAT IP 集區。
- 不需要在 `ncp.ini` 中設定 SNAT IP 集區。
- `ncp/snat_pool` 所指定的 IP 集區必須有標記 `{"ncp/owner": cluster:<cluster>}`。
- `ncp/snat_pool` 所指定的 IP 集區也可具有命名空間標記 `{"ncp/owner": ns:<namespace_UUID>}`。
- 如果缺少 `ncp/snat_pool` 註解，命名空間會將 SNAT IP 集區用於叢集。
- 您可以透過變更註解並重新啟動 NCP，變更為不同的 IP 集區。

為 PAS 應用程式設定 SNAT 集區

依預設，NCP 會為 PAS (Pivotal Application Service) 組織設定 SNAT IP。您可以透過使用包含 SNAT IP 集區資訊的 `manifest.xml` 建立應用程式，來為應用程式設定 SNAT IP。例如，

```
applications:
- name: frontend
  memory: 32M
  disk_quota: 32M
  buildpack: go_buildpack
```

```
env:
  GOPACKAGENAME: example-apps/cats-and-dogs/frontend
  NCP_SNAT_POOL: <external IP pool ID or name>
  ...
```

NCP 將針對此應用程式設定 SNAT 規則。規則的來源 IP 是執行個體的 IP 集合，規則的目的地 IP 是從外部 IP 集區配置的 SNAT IP。請注意下列事項：

- NCP_SNAT_POOL 所指定的集區應在推送應用程式之前已存在於 NSX-T Data Center 中。
- 應用程式的 SNAT 規則的優先順序高於組織的 SNAT 規則。
- 一個應用程式只能設定一個 SNAT IP。
- 從 NCP 2.3.2 開始，您可以透過變更組態並重新啟動 NCP，變更為不同的 IP 集區。

為 PCF 第 3 版設定 SNAT

透過 PCF 第 3 版，您可以採用兩種方式設定 SNAT：

- 建立應用程式時在 `manifest.yml` 中設定 NCP_SNAT_POOL。

例如，應用程式名為 `bread`，並且 `manifest.yml` 具有下列資訊：

```
applications:
- name: bread
  stack: cflinuxfs2
  random-route: true
  env:
    NCP_SNAT_POOL: AppSnatExternalIppool
  processes:
  - type: web
    disk_quota: 1024M
    instances: 2
    memory: 512M
    health-check-type: port
  - type: worker
    disk_quota: 1024M
    health-check-type: process
    instances: 2
    memory: 256M
    timeout: 15
```

執行下列命令：

```
cf v3-push bread
cf v3-apply-manifest -f manifest.yml
cf v3-apps
cf v3-restart bread
```

- 使用 `cf v3-set-env` 命令設定 NCP_SNAT_POOL。

執行下列命令 (假設應用程式名為 `app3`):

```
cf v3-set-env app3 NCP_SNAT_POOL AppSnatExternalIppool
(optional) cf v3-stage app3 -package-guid <package-guid> (You can get package-guid with "cf v3-
packages app3".)
cf v3-restart app3
```

(選用) (僅限 Kubernetes) 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 Kubernetes 環境中安裝 NCP

安裝 NSX-T Container Plug-in (NCP) 時需要在主機和 Kubernetes 節點上安裝元件。

本章包含以下主題：

- 安裝 NSX-T Data Center CNI 外掛程式
- 安裝和設定 OVS
- 設定 Kubernetes 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
- 在 NCP 網繭中掛接 PEM 編碼憑證和私密金鑰
- 在 NCP 網繭中掛接憑證檔案
- 設定 Syslog
- 安全考量事項
- 設定網路資源的相關提示

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 Kubernetes 節點上。

針對 Ubuntu，安裝 NSX-T CNI 外掛程式會將 AppArmor 設定檔檔案 `ncp-apparmor` 複製到 `/etc/apparmor.d` 並將其載入。安裝之前，AppArmor 服務必須處於執行中狀態，且目錄 `/etc/apparmor.d` 必須存在。否則，安裝將會失敗。您可以檢查是否已透過下列命令啟用 AppArmor 模組：

```
sudo cat /sys/module/apparmor/parameters/enabled
```

您可以檢查是否已透過下列命令啟動 AppArmor 服務：

```
sudo /etc/init.d/apparmor status
```

如果 AppArmor 服務在您安裝 NSX-T CNI 外掛程式時並未執行，則安裝完成後將會顯示下列訊息：

```
subprocess installed post-installation script returned error exit status 1
```

該訊息指出除了載入 AppArmor 設定檔外，所有安裝步驟皆已完成。

ncp-apparmor 設定檔檔案會針對名為 node-agent-apparmor 的 NSX 節點代理程式提供 AppArmor 設定檔，其與 docker-default 設定檔之間存在下列差異：

- deny mount 規則便會移除。
- mount 規則便會新增。
- 會新增一些 network、capability、file 和 umount 選項。

您可以使用其他設定檔取代 node-agent-apparmor 設定檔。但是，設定檔名稱 node-agent-apparmor 引用於 nsx-node-agent-ds.yml 檔案中，且用於安裝 NSX 節點代理程式。如果使用其他設定檔，您必須在 nsx-node-agent-ds.yml 中的 spec:template:metadata:annotations 區段下，透過下列項目指定設定檔名稱：

```
container.apparmor.security.beta.kubernetes.io/<container-name>: localhost/<profile-name>
```

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 或 nsx-cni-1.0.0.0.0.xxxxxxx.deb，其中，xxxxxxx 是組建編號。

- 2 安裝在步驟 1 中下載的 rpm 或 deb 檔案。

此外掛程式會安裝在 /opt/cni/bin 中。CNI 組態檔 10-nsx.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。

安裝和設定 OVS

在部屬節點上安裝和設定 OVS (Open vSwitch)。

程序

- 1 下載您的 Linux 發行版適用的安裝檔案。

檔案名稱為 openvswitch-common_2.7.0.6383646-1_amd64.deb、openvswitch-datapath-dkms_2.7.0.xxxxxxx-1_all.deb 和 openvswitch-switch_2.7.0.xxxxxxx-1_amd64.deb，其中，xxxxxxx 是組建編號。

- 2 安裝在步驟 1 中下載的 deb 檔案。
- 3 對於 Ubuntu，執行下列命令以重新載入 OVS 核心模組。

```
# systemctl openvswitch-switch force-reload-kmod
```


4 確定 OVS 正在執行。

```
# system openvswitch-switch status
```

5 建立 *br-int* 執行個體 (若尚未建立)。

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

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

```
# ovs-vsctl add-port br-int <node-if> -- set Interface <node-if> ofport_request=1
```

請執行下列命令以確認 *ofport* 為何，因為若 *ofport* 1 無法使用，OVS 將會指派可用的連接埠。

```
# ovs-vsctl --columns=ofport list interface <node-if>
```

如果 *ofport* 不是 1，請據以在 NSX 節點代理程式 DaemonSet yaml 檔案的 *nsx_kube_proxy* 區段中設定 *ovs_uplink_port* 選項。

7 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
```

8 更新網路組態檔，以確保網路介面會在重新開機後開啟。

對於 Ubuntu，請更新 */etc/network/interfaces*，並新增下列幾行：

```
auto <node-if>
iface <node-if> inet manual
up ip link set <node-if> up
```

對於 RHEL，請更新 */etc/sysconfig/network-scripts/ifcfg-<node-if>*，並新增下列幾行：

```
ONBOOT=yes
```

設定 Kubernetes 節點的 NSX-T Data Center 網路

本節說明如何設定 Kubernetes 主要及部屬節點的 NSX-T Data Center 網路。

每個節點必須至少具有兩個網路介面。第一個是管理介面，不一定會位於 NSX-T Data Center 網狀架構上。另一個介面會為網繭提供網路，位於 NSX-T Data Center 網狀架構上，且會連線至邏輯交換器，也稱為節點邏輯交換器。管理與網繭 IP 位址必須可供路由傳送，如此 Kubernetes 健全狀況檢查才能正常運作。針對管理介面與網繭之間的通訊，NCP 會自動建立 DFW 規則，以允許健全狀況檢查和其他管理流量。您可以在 NSX Manager GUI 中看到此規則的詳細資料。此規則不應變更或刪除。

針對每個節點虛擬機器，請確定針對容器網路指定的 vNIC 連結至節點邏輯交換器。

NSX-T Container Plug-in (NCP) 必須已知用於每個節點中容器流量之 vNIC 的 VIF 識別碼。對應的邏輯交換器連接埠必須以下列方式加以標記：

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

您可以識別節點虛擬機器的邏輯交換器連接埠，方法為從 NSX Manager GUI 導覽至[詳細目錄 > 虛擬機器](#)。

如果 Kubernetes 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：

```
kubectl get nodes
```

如果您在 NCP 執行時將節點新增至叢集，則在執行 `kubeadm join` 命令之前必須將標記新增至邏輯交換器連接埠。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：

- 將正確的標記套用至邏輯交換器連接埠。
- 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.yaml`。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.yaml`。

將映像名稱變更為已載入的映像名稱。

針對 Ubuntu，yaml 檔案會假設 AppArmor 已啟用。若要確認 AppArmor 是否啟用，請檢查 `/sys/module/apparmor/parameters/enabled` 檔案。如果 AppArmor 未啟用，請進行下列變更：

- 將下列一行刪除或取消註解：

```
container.apparmor.security.beta.kubernetes.io/nsx-node-agent: localhost/node-agent-apparmor
```

- 針對 `nsx-node-agent` 容器和 `nsx-kube-proxy` 容器，在 `securityContext` 下方新增 `privileged:true` 一行。例如：

```
securityContext:
  privileged:true
```

備註 目前的已知問題是，如果 kubelet 在使用 hyperkube 映像的容器內執行，則 kubelet 一律會將 AppArmor 報告為已停用狀態，無論實際狀態為何。您必須對 yaml 檔案進行如上所述的相同變更。

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。

```
kubectl 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 = HOSTVM

[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 會用於所有從網繭邏輯交換器之 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。

在輪流更新 NCP ReplicationController 期間，在下列情況下，您可能會在輪流更新後看到正在執行的兩個 NCP 網繭：

- 在輪流更新期間將容器主機重新開機。
- 輪流更新最初會失敗，因為新的映像不存在於 Kubernetes 節點上。下載映像並重新執行輪流更新，它會成功。

如果您看到兩個正在執行的 NCP 網繭，請執行下列操作：

- 刪除其中一個 NCP 網繭。刪除任一網繭皆可。例如，

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

- 刪除命名空間 `nsx-system`。例如，

```
kubectl 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 = HOSTVM

[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>
```

在 NCP 網繭中掛接 PEM 編碼憑證和私密金鑰

如果您擁有 PEM 編碼憑證和私密金鑰，則可以更新 yaml 檔案中的 NCP 網繭定義以在 NCP 網繭中掛接 TLS 密碼。

- 1 建立憑證和私密金鑰的 TLS 密碼。

```
kubectl create secret tls SECRET_NAME --cert=/path/to/tls.crt --key=/path/to/tls.key
```

- 2 更新 NCP 網繭規格 yaml，以將密碼掛接為 NCP 網繭規格中的檔案。

```
spec:
  ...
  containers:
  - name: nsx-ncp
    ...
    volumeMounts:
    ...
    - name: nsx-cert
      mountPath: /etc/nsx-ujo/nsx-cert
      readOnly: true
  volumes:
  ...
  - name: nsx-cert
    secret:
      secretName: SECRET_NAME
```

- 3 更新 yaml 檔案中的 nsx_v3 選項 nsx_api_cert_file 和 nsx_api_private_key_file。

```
nsx_api_cert_file = /etc/nsx-ujo/nsx-cert/tls.crt
nsx_api_private_key_file = /etc/nsx-ujo/nsx-cert/tls.key
```

在 NCP 網繭中掛接憑證檔案

如果您在節點檔案系統中擁有憑證檔案，則可以更新 NCP 網繭規格以在 NCP 網繭中掛接該檔案。

例如，

```
spec:
  ...
  containers:
  - name: nsx-ncp
    ...
    volumeMounts:
    ...
    - name: nsx-cert
```

```

# Mount path must match nsx_v3 option "nsx_api_cert_file"
mountPath: /etc/nsx-ujo/nsx_cert
- name: nsx-priv-key
# Mount path must match nsx_v3 option "nsx_api_private_key_file"
mountPath: /etc/nsx-ujo/nsx_priv_key
volumes:
...
- name: nsx-cert
  hostPath:
    path: <host-filesystem-cert-path>
- name: nsx-priv-key
  hostPath:
    path: <host-filesystem-priv-key-path>

```

設定 Syslog

您可以在容器中執行 Syslog 代理程式 (例如 `rsyslog` 或 `syslog-ng`)，以將 NCP 中的記錄和相關元件傳送至 Syslog 伺服器。

建議使用下列方法。如需關於登入 Kubernetes 的詳細資訊，請參閱 <https://kubernetes.io/docs/concepts/cluster-administration/logging>。

- 建立在 NCP 或 `nsx-node-agent` 網繭中執行的側車容器。
- 在每個節點上執行 DaemonSet 複寫。

備註 透過側車容器方法，NSX CNI 外掛程式記錄無法傳送至 Syslog 伺服器，因為外掛程式並未在網繭中執行。

建立 Syslog 的側車容器

您可以為 Syslog 設定側車容器，以便在與 NCP 相同的網繭中執行。下列程序假設 Syslog 代理程式映像為 `example/rsyslog`。

程序

- 1 設定 NCP 和 NSX 節點代理程式以記錄至檔案。

在 NCP 和 NSX 節點代理程式的 `yaml` 檔案中設定 `log_dir` 參數，並指定要掛接的磁碟區。例如，

```

[DEFAULT]
log_dir = /var/log/nsx-ujo/
...

spec:
  ...
  containers:
    - name: nsx-ncp
      ...
      volumeMounts:
        - name: nsx-ujo-log-dir
          # Mount path must match [DEFAULT] option "log_dir"
          mountPath: /var/log/nsx-ujo

```

```
volumes:
  ...
  - name: nsx-ujo-log-dir
    hostPath:
      path: /var/log/nsx-ujo
```

您可以藉由設定 `log_file` 參數來變更記錄檔名稱。預設名稱為 `ncp.log`、`nsx_node_agent.log` 和 `nsx_kube_proxy.log`。如果 `log_dir` 選項設為 `/var/log/nsx-ujo` 以外的路徑，則必須建立 `hostPath` 磁碟區或 `emptyDir` 磁碟區，並將其掛接至對應的網繭規格。

2 確保主機路徑存在且可由使用者 `nsx-ncp` 寫入。

a 執行下列命令。

```
mkdir -p <host-filesystem-log-dir-path>
chmod +w <host-filesystem-log-dir-path>
```

b 新增使用者 `nsx-ncp` 或將主機路徑的模式變更為 `777`。

```
useradd -s /bin/bash nsx-ncp
chown nsx-ncp:nsx-ncp <host-filesystem-log-dir-path>
or
chmod 777 <host-filesystem-log-dir-path>
```

3 在 NCP 網繭的規格 `yaml` 檔案中，新增 Syslog 的 `ConfigMap`。例如，

```
kind: ConfigMap
metadata:
  name: rsyslog-config
  labels:
    version: v1
data:
  ncp.conf: |
    module(load="imfile")

    ruleset(name="remote") {
      action(type="omfwd"
        Protocol="tcp"
        Target="nsx.example.com"
        Port="514")

      stop
    }

    input(type="imfile"
      File="/var/log/nsx-ujo/ncp.log"
      Tag="ncp"
      Ruleset="remote")
```

- 4 在 NCP 網繭的 `yaml` 檔案中新增 `rsyslog` 容器，然後掛接可讓 `rsyslog` 從其他容器中找到組態資料及讀取記錄的適當磁碟區。例如，

```
spec:
  containers:
  - name: nsx-ncp
    ...
  - name: rsyslog
    image: example/rsyslog
    imagePullPolicy: IfNotPresent
    volumeMounts:
    - name: rsyslog-config-volume
      mountPath: /etc/rsyslog.d
      readOnly: true
    - name: nsx-ujo-log-dir
      mountPath: /var/log/nsx-ujo
  volumes:
  ...
  - name: rsyslog-config-volume
    configMap:
      name: rsyslog-config
  - name: nsx-ujo-log-dir
    hostPath:
      path: <host-filesystem-log-dir-path>
```

建立 Syslog 的 DaemonSet 複本

所有 NCP 元件的記錄皆可透過此方式進行重新導向。應用程式必須設定為記錄至 `stderr` (依預設為啟用)。下列程序假設 Syslog 代理程式映像為 `example/rsyslog`。

程序

- 1 建立 DaemonSet `yaml` 檔案。例如，

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: rsyslog-config
  labels:
    version: v1
data:
  nsx-ncp.conf: |
    module(load="imfile")

    ruleset(name="remote") {
      if $msg contains 'nsx-container' then
        action(type="omfwd"
          Protocol="tcp"
          Target="nsx.example.com"
          Port="514")

      stop
    }
```



```

        input(type="imfile"
              File="/var/log/containers/nsx-node-agent-*.log"
              Tag="nsx-node-agent"
              Ruleset="remote")

        input(type="imfile"
              File="/var/log/containers/nsx-ncp-*.log"
              Tag="nsx-ncp"
              Ruleset="remote")

        input(type="imfile"
              File="/var/log/syslog"
              Tag="nsx-cni"
              Ruleset="remote")
---
# rsyslog DaemonSet
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: rsyslog
  labels:
    component: rsyslog
    version: v1
spec:
  template:
    metadata:
      labels:
        component: rsyslog
        version: v1
  spec:
    hostNetwork: true
    containers:
    - name: rsyslog
      image: example/rsyslog
      imagePullPolicy: IfNotPresent
      volumeMounts:
      - name: rsyslog-config-volume
        mountPath: /etc/rsyslog.d
      - name: log-volume
        mountPath: /var/log
      - name: container-volume
        mountPath: /var/lib/docker/containers
    volumes:
    - name: rsyslog-config-volume
      configMap:
        name: rsyslog-config
    - name: log-volume
      hostPath:
        path: /var/log
    - name: container-volume
      hostPath:
        path: /var/lib/docker/containers

```

2 建立 DaemonSet。

```
kubectl apply -f <daemonset yaml file>
```

範例：設定在 sidecar 容器中執行的記錄輪替和 Syslog

下列程序顯示如何設定在 sidecar 容器中執行的記錄輪替和 Syslog。

建立記錄目錄和設定記錄輪替

- 在包括主節點在內的所有節點上建立記錄目錄，並將其擁有者變更為具備識別碼 1000 的任意使用者。

```
mkdir /var/log/nsx-ujo
chown localadmin:localadmin /var/log/nsx-ujo
```

- 針對 /var/log/nsx-ujo 目錄，在所有節點上設定記錄輪替。

```
cat <<EOF > /etc/logrotate.d/nsx-ujo
/var/log/nsx-ujo/*.log {
    copytruncate
    daily
    size 100M
    rotate 4
    delaycompress
    compress
    notifempty
    missingok
}
EOF
```

建立 NCP 複寫控制站

- 針對 NCP 建立 ncp.ini 檔案。

```
cat <<EOF > /tmp/ncp.ini
[DEFAULT]
log_dir = /var/log/nsx-ujo
[coe]
cluster = k8s-cl1
[k8s]
apiserver_host_ip = 10.114.209.77
apiserver_host_port = 6443
ca_file = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
client_token_file = /var/run/secrets/kubernetes.io/serviceaccount/token
insecure = True
ingress_mode = nat
[nsx_v3]
nsx_api_user = admin
nsx_api_password = Password1!
nsx_api_managers = 10.114.209.68
insecure = True
subnet_prefix = 29
```

```
[nsx_node_agent]
[nsx_kube_proxy]
ovs_uplink_port = ens192
EOF
```

- 從 ini 檔案建立 configmap。

```
kubectl create configmap nsx-ncp-config-with-logging --from-file=/tmp/ncp.ini
```

- 建立 NCP rsyslog 組態。

```
cat <<EOF > /tmp/nsx-ncp-rsyslog.conf
# yaml template for NCP ReplicationController
# Correct kubernetes API and NSX API parameters, and NCP Docker image
# must be specified.
apiVersion: v1
kind: ConfigMap
metadata:
  name: rsyslog-config
  labels:
    version: v1
data:
  ncp.conf: |
    module(load="imfile")

    ruleset(name="remote") {
      action(type="omfwd"
        Protocol="tcp"
        Target="nsx.licf.vmware.com"
        Port="514")

      stop
    }

    input(type="imfile"
      File="/var/log/nsx-ujo/ncp.log"
      Tag="ncp"
      Ruleset="remote")
EOF
```

- 從上述內容建立 configmap。

```
kubectl create -f /tmp/nsx-ncp-rsyslog.conf
```

- 使用 rsyslog sidecar 建立 NCP 複寫控制站。

```
cat <<EOF > /tmp/ncp-rc-with-logging.yml
# Replication Controller yaml for NCP
apiVersion: v1
kind: ReplicationController
metadata:
  # VMware NSX Container Plugin
  name: nsx-ncp
```

```

labels:
  tier: nsx-networking
  component: nsx-ncp
  version: v1
spec:
  # Active-Active/Active-Standby is not supported in current release.
  # so replica *must be* 1.
  replicas: 1
  template:
    metadata:
      labels:
        tier: nsx-networking
        component: nsx-ncp
        version: v1
    spec:
      # NCP shares the host management network.
      hostNetwork: true
      nodeSelector:
        kubernetes.io/hostname: k8s-master
      tolerations:
        - key: "node-role.kubernetes.io/master"
          operator: "Exists"
          effect: "NoSchedule"
      containers:
        - name: nsx-ncp
          # Docker image for NCP
          image: nsx-ujo-docker-local.artifactory.eng.vmware.com/nsx-ncp:ob-6236425
          imagePullPolicy: IfNotPresent
          readinessProbe:
            exec:
              command:
                - cat
                - /tmp/ncp_ready
            initialDelaySeconds: 5
            periodSeconds: 5
            failureThreshold: 5
          securityContext:
            capabilities:
              add:
                - NET_ADMIN
                - SYS_ADMIN
                - SYS_PTRACE
                - DAC_READ_SEARCH
          volumeMounts:
            - name: config-volume
              # NCP expects ncp.ini is present in /etc/nsx-ujo
              mountPath: /etc/nsx-ujo
            - name: log-volume
              mountPath: /var/log/nsx-ujo
        - name: rsyslog
          image: jumanjiman/rsyslog
          imagePullPolicy: IfNotPresent
          volumeMounts:
            - name: rsyslog-config-volume
              mountPath: /etc/rsyslog.d

```

```

        readOnly: true
      - name: log-volume
        mountPath: /var/log/nsx-ujo
    volumes:
      - name: config-volume
        # ConfigMap nsx-ncp-config is expected to supply ncp.ini
        configMap:
          name: nsx-ncp-config-with-logging
      - name: rsyslog-config-volume
        configMap:
          name: rsyslog-config
      - name: log-volume
        hostPath:
          path: /var/log/nsx-ujo/
EOF

```

- 使用上述規格建立 NCP。

```
kubectl apply -f /tmp/ncp-rc-with-logging.yml
```

建立 NSX 節點代理程式 DaemonSet

- 針對節點代理程式建立 rsyslog 組態。

```

cat <<EOF > /tmp/nsx-node-agent-rsyslog.conf
# yaml template for NCP ReplicationController
# Correct kubernetes API and NSX API parameters, and NCP Docker image
# must be specified.
apiVersion: v1
kind: ConfigMap
metadata:
  name: rsyslog-config-node-agent
  labels:
    version: v1
data:
  ncp.conf: |
    module(load="imfile")

    ruleset(name="remote") {
      action(type="omfwd"
        Protocol="tcp"
        Target="nsx.licf.vmware.com"
        Port="514")

      stop
    }

    input(type="imfile"
      File="/var/log/nsx-ujo/nsx_kube_proxy.log"
      Tag="nsx_kube_proxy"
      Ruleset="remote")

    input(type="imfile"

```

```

        File="/var/log/nsx-ujo/nsx_node_agent.log"
        Tag="nsx_node_agent"
        Ruleset="remote")
EOF

```

- 從上述內容建立 configmap。

```
kubectl create -f /tmp/nsx-node-agent-rsyslog.conf
```

- 使用 configmap sidecar 建立 DaemonSet。

```

cat <<EOF > /tmp/nsx-node-agent-rsyslog.yml
# nsx-node-agent DaemonSet
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: nsx-node-agent
  labels:
    tier: nsx-networking
    component: nsx-node-agent
    version: v1
spec:
  template:
    metadata:
      annotations:
        container.apparmor.security.beta.kubernetes.io/nsx-node-agent: localhost/node-agent-
apparmor
      labels:
        tier: nsx-networking
        component: nsx-node-agent
        version: v1
    spec:
      hostNetwork: true
      tolerations:
        - key: "node-role.kubernetes.io/master"
          operator: "Exists"
          effect: "NoSchedule"
      containers:
        - name: nsx-node-agent
          # Docker image for NCP
          image: nsx-ujo-docker-local.artifactory.eng.vmware.com/nsx-ncp:ob-6236425
          imagePullPolicy: IfNotPresent
          # override NCP image entrypoint
          command: ["nsx_node_agent"]
          livenessProbe:
            exec:
              command:
                - /bin/sh
                - -c
                - ps aux | grep [n]sx_node_agent
            initialDelaySeconds: 5
            periodSeconds: 5
          securityContext:
            capabilities:

```

```

    add:
      - NET_ADMIN
      - SYS_ADMIN
      - SYS_PTRACE
      - DAC_READ_SEARCH
  volumeMounts:
  # ncp.ini
  - name: config-volume
    mountPath: /etc/nsx-ujo
  # mount openvswitch dir
  - name: openvswitch
    mountPath: /var/run/openvswitch
  # mount CNI socket path
  - name: cni-sock
    mountPath: /var/run/nsx-ujo
  # mount container namespace
  - name: netns
    mountPath: /var/run/netns
  # mount host proc
  - name: proc
    mountPath: /host/proc
    readOnly: true
  - name: log-volume
    mountPath: /var/log/nsx-ujo
- name: nsx-kube-proxy
  # Docker image for NCP
  image: nsx-ujo-docker-local.artifactory.eng.vmware.com/nsx-ncp:ob-6236425
  imagePullPolicy: IfNotPresent
  # override NCP image entrypoint
  command: ["nsx_kube_proxy"]
  livenessProbe:
    exec:
      command:
        - /bin/sh
        - -c
        - ps aux | grep [n]sx_kube_proxy
    initialDelaySeconds: 5
    periodSeconds: 5
  securityContext:
    capabilities:
      add:
        - NET_ADMIN
        - SYS_ADMIN
        - SYS_PTRACE
        - DAC_READ_SEARCH
  volumeMounts:
  # ncp.ini
  - name: config-volume
    mountPath: /etc/nsx-ujo
  # mount openvswitch dir
  - name: openvswitch
    mountPath: /var/run/openvswitch
  - name: log-volume
    mountPath: /var/log/nsx-ujo
- name: rsyslog

```

```

    image: jumanjiman/rsyslog
    imagePullPolicy: IfNotPresent
    volumeMounts:
    - name: rsyslog-config-volume
      mountPath: /etc/rsyslog.d
      readOnly: true
    - name: log-volume
      mountPath: /var/log/nsx-ujo
  volumes:
  - name: config-volume
    configMap:
      name: nsx-ncp-config-with-logging
  - name: cni-sock
    hostPath:
      path: /var/run/nsx-ujo
  - name: netns
    hostPath:
      path: /var/run/netns
  - name: proc
    hostPath:
      path: /proc
  - name: openvswitch
    hostPath:
      path: /var/run/openvswitch
  - name: rsyslog-config-volume
    configMap:
      name: rsyslog-config-node-agent
  - name: log-volume
    hostPath:
      path: /var/log/nsx-ujo/
EOF

```

- 建立 DaemonSet。

```
kubectl apply -f /tmp/nsx-node-agent-rsyslog.yml
```

安全考量事項

部署 NCP 時，請務必要採取保護 Kubernetes 與 NSX-T Data Center 環境的相關步驟。

將 NCP 限制在指定的節點上執行

NCP 具有 NSX-T Data Center 管理平面的存取權，因此應受到限制而僅能在指定的基礎結構節點上執行。您可以使用適當標籤識別這些節點。接著，應將此標籤的 `nodeSelector` 套用至 NCP ReplicationController 規格。例如，

```

nodeSelector:
  nsx-infra: True

```

您也可以使用其他機制 (例如相似性) 將網繭指派給節點。如需詳細資訊，請參閱 <https://kubernetes.io/docs/concepts/configuration/assign-pod-node>。

確定 Docker Engine 為最新版本

Docker 會定期發行安全性更新。您應實作自動化程序來套用這些更新。

不允許不受信任容器的 NET_ADMIN 和 NET_RAW

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，系統並不會對不具權限的容器授與 NET_ADMIN 功能。請留意網繭規格是否明確加以啟用，或將容器設定為在特殊權限模式中執行。此外，對於不受信任的容器，請在容器規格的 SecurityContext 組態中的捨棄功能清單中指定 NET_RAW 來停用 NET_RAW。例如，

```
securityContext:
  capabilities:
    drop:
      - NET_RAW
      - ...
```

角色型存取控制

Kubernetes 會使用角色型存取控制 (RBAC) API 控制授權決定，讓管理員能夠動態設定原則。如需詳細資訊，請參閱 <https://kubernetes.io/docs/admin/authorization/rbac>。

一般而言，叢集管理員是唯一具有存取權和角色權限的使用者。對於使用者和服務帳戶，授與存取權時必須遵循最低權限的原則。

建議採用下列準則：

- 將 Kubernetes API Token 的存取權限制在需要這些 Token 的網繭。
- 將 NCP ConfigMap 和 NSX API 用戶端憑證 TLS 密碼的存取權限制在 NCP 網繭。
- 封鎖不需要存取 Kubernetes 網路 API 的網繭對此類 API 的存取。
- 新增 Kubernetes RBAC 原則以指定有權存取 Kubernetes API 的網繭。

針對 NCP 網繭而建議的 RBAC 原則

在 ServiceAccount 下建立 NCP 網繭，並給予此帳戶最低的權限集。此外，不要允許其他網繭或 ReplicationController 存取為 NCP ReplicationController 和 NSX 節點代理程式掛接為磁碟區的 ConfigMap 和 TLS 密碼。

下列範例說明如何指定 NCP 的角色和角色繫結：

```
# Create a ServiceAccount for NCP namespace
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ncp-svc-account
  namespace: nsx-system
```

```

---

# Create ClusterRole for NCP
kind: ClusterRole
# Set the apiVersion to v1 while using OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: ncp-cluster-role
rules:
  - apiGroups:
    - ""
    - extensions
    - networking.k8s.io
  resources:
    - deployments
    - endpoints
    - pods
    - pods/log
    - namespaces
    - networkpolicies
    # Move 'nodes' to ncp-patch-role when hyperbus is disabled.
    - nodes
    - replicationcontrollers
    # Remove 'secrets' if not using Native Load Balancer.
    - secrets
  verbs:
    - get
    - watch
    - list

```

```

---

# Create ClusterRole for NCP to edit resources
kind: ClusterRole
# Set the apiVersion to v1 while using OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: ncp-patch-role
rules:
  - apiGroups:
    - ""
    - extensions
  resources:
    - ingresses
    - services
  verbs:
    - get
    - watch
    - list
    - update
    - patch
  - apiGroups:
    - ""
    - extensions
  resources:

```

```

    - ingresses/status
    - services/status
  verbs:
    - replace
    - update
    - patch
---

# Bind ServiceAccount created for NCP to its ClusterRole
kind: ClusterRoleBinding
# Set the apiVersion to v1 while using OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: ncp-cluster-role-binding
roleRef:
  # Comment out the apiGroup while using OpenShift
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ncp-cluster-role
subjects:
  - kind: ServiceAccount
    name: ncp-svc-account
    namespace: nsx-system
---

# Bind ServiceAccount created for NCP to the patch ClusterRole
kind: ClusterRoleBinding
# Set the apiVersion to v1 while using OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: ncp-patch-role-binding
roleRef:
  # Comment out the apiGroup while using OpenShift
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ncp-patch-role
subjects:
  - kind: ServiceAccount
    name: ncp-svc-account
    namespace: nsx-system

```

備註 使用 Kubernetes API 為 NSX-T Data Center 用戶端憑證與私密金鑰配對建立的 TLS 密碼，可供任何有權存取 Kubernetes API 伺服器的網繭存取。同樣地，如果網繭並非使用任何服務帳戶所建立，則該網繭將自動指派至相同命名空間中會自動掛接 Token 以存取 Kubernetes API 的預設服務帳戶。因此，必須限制需要使用的網繭才能存取這些 Token。

設定網路資源的相關提示

設定部分網路資源時，應該注意某些限制。

NSX-T Data Center 標記限制

NSX-T Data Center 對於標記物件具有下列限制：

- 範圍限制為 128 個字元。
- 標記限制為 256 個字元。
- 每個物件最多可以有 30 個標記。

將 Kubernetes 或 OpenShift 註解複製到 NSX-T Data Center 範圍和標記，並且超過限制時，即可能導致問題。例如，如果某個標記適用於交換器連接埠但用於防火牆規則，則規則可能不會如預期般加以套用，因為註解金鑰或值在複製到範圍或標記時已遭到截斷。

設定網路原則

網路原則使用標籤選取器選取網繭或命名空間。

NCP 的網路原則支援與 Kubernetes 所提供的支援相同，視 Kubernetes 版本而定。

- **Kubernetes 1.11** - 您可以指定下列規則選取器：
 - **podSelector**：此選取器會選取建立網路原則所在命名空間中的所有網繭。
 - **namespaceSelector**：此選取器會選取所有命名空間。
 - **podSelector AND namespaceSelector**：此選取器會選取 namespaceSelector 所選取命名空間中的所有網繭。
 - **ipBlockSelector**：如果 ipBlockSelector 與 namespaceSelector 或 podSelector 組合使用，網路原則將無效。ipBlockSelector 必須獨立存在於原則規格中。
- **Kubernetes 1.10** - 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。

在 Pivotal Cloud Foundry 環境中 安裝 NCP

Pivotal Cloud Foundry (PCF) 是開放原始碼的平台即服務 (PaaS) 提供者。您可以在 PCF 環境中安裝 NSX-T Container Plug-in (NCP) 以提供網路服務。

透過 Pivotal Ops Manager 建立的虛擬機器必須具有與容器網路的第 3 層連線，才能存取 NSX-T 功能。

High Availability (HA) 將會自動啟用。安裝 NCP 之前，您必須使用下列標記建立 SpoofGuard 交換設定檔：

- 範圍：ncp/cluster，標記：<cluster_name>
- 範圍：ncp/ha，標記：true

備註 對安全群組做出變更時，您必須重新暫存套用安全群組的所有應用程式。由於安全群組套用至應用程式執行所在的空間，或安全群組為全域群組，因此會出現此情況。

在 Pivotal Cloud Foundry 環境中安裝 NCP

NCP 可透過 Pivotal Ops Manager 圖形化使用者介面安裝。

先決條件

全新安裝 Pivotal Ops Manager、NSX-T Data Center 和 Pivotal Application Service (PAS)。請確定以如下順序進行安裝：Ops Manager、NSX-T Data Center 和 PAS。如需詳細資訊，請參閱 Pivotal Cloud Foundry 說明文件。

程序

- 1 下載適用於 PCF 的 NCP 安裝檔案。
檔案名稱為 VMware-NSX-T.<version>.<build>.pivotal。
- 2 以管理員身分登入 Pivotal Ops Manager。
- 3 按一下匯入產品。
- 4 選取已下載的檔案。
- 5 按一下 **Ops Manager Director for VMware vSphere** 動態磚。
- 6 在設定索引標籤中，針對 **vCenter 組態** 選取 **NSX 網路**，針對 **NSX 模式** 選取 **NSX-T**。
- 7 在 **NSX CA 憑證** 欄位中，提供 PEM 格式的憑證。

- 8 按一下**儲存**。
- 9 在左上角按一下**安裝儀表板**，以返回儀表板。
- 10 按一下 **Pivotal Application Service** 動態磚。
- 11 在**設定索引標籤**的導覽窗格中，選取**網路**。
- 12 在**容器網路**介面外掛程式下，選取**外部**。
- 13 在左上角按一下**安裝儀表板**，以返回儀表板。
- 14 按一下**儲存**。
- 15 在左上角按一下**安裝儀表板**，以返回儀表板。
- 16 按一下 **VMware NSX-T** 動態磚。
- 17 輸入 NSX Manager 的位址。
- 18 選取 NSX Manager 驗證方法。

選項	動作
用戶端憑證驗證	提供 NSX Manager 的憑證和私密金鑰。
透過使用者名稱和密碼進行基本驗證	提供 NSX Manager 管理員使用者名稱和密碼。

- 19 在 **NSX Manager CA 憑證**欄位中，提供憑證。
- 20 按一下**儲存**。
- 21 在導覽窗格中選取 **NCP**。
- 22 輸入 **PAS Foundation** 名稱。
此字串可唯一識別 NSX API 中的 PAS Foundation。為 PAS Foundation 而以 NCP 建立的 NSX 資源，亦可用此字串作為其名稱中的前置字元。
- 23 輸入**覆疊傳輸區域**。
- 24 輸入**第 0 層**路由器。
- 25 指定一或多個**容器網路的 IP 區塊**。
 - a 按一下**新增**。
 - b 輸入 **IP 區塊名稱**。它可以是新的或現有的 IP 區塊。
 - c 僅針對新的 IP 區塊，以 CIDR 格式指定區塊，例如 10.1.0.0/16。
- 26 指定容器網路的子網路首碼。
- 27 按一下**針對容器網路啟用 SNAT**以啟用 SNAT。

28 指定一或多個用於將外部 (NAT) IP 位址提供給組織網路的 IP 集區。

- a 按一下**新增**。
- b 輸入 **IP 集區名稱**。它可以是新的或現有的 IP 集區。
- c 僅針對新的 IP 集區，透過提供 CIDR 和 IP 範圍指定 IP 位址。

29 (可選) 輸入**頂部防火牆區段標記**。

30 (可選) 輸入**底部防火牆區段標記**。

31 (可選) 啟用或停用下列選項。

選項	預設值
記錄捨棄的應用程式流量	已停用。如果啟用，將會記錄因防火牆規則而捨棄的流量。
啟用 NCP 記錄的偵錯層級	已啟用。

32 按一下**儲存**。

33 (可選) 在導覽窗格中選取 **NSX 節點代理程式**。

- a 勾選**針對 NSX 節點代理程式啟用偵錯層級記錄**，以啟用偵錯層級記錄。
- b 按一下**儲存**。

34 在左上角按一下**安裝儀表板**，以返回儀表板。

35 按一下**套用變更**。

負載平衡

NSX-T Data Center 負載平衡器與 Kubernetes 整合。

設定負載平衡

設定負載平衡包括設定 Kubernetes 負載平衡器服務或入口資源以及 NCP 複寫控制站。

您可以透過設定負載平衡器類型的 Kubernetes 服務，建立第 4 層負載平衡器，並透過設定 Kubernetes 入口資源建立第 7 層負載平衡器。

若要在 NCP 中設定負載平衡，請在 `ncp-rc.yml` 檔案中：

- 1 設定 `use_native_loadbalancer = True`。
- 2 (選擇性) 將 `pool_algorithm` 設定為 `ROUND_ROBIN` 或 `LEAST_CONNECTION/IP_HASH`。預設為 `ROUND_ROBIN`。
- 3 (選擇性) 設定 `service_size = SMALL`、`MEDIUM` 或 `LARGE`。預設為 `SMALL`。

`LEAST_CONNECTION/IP_HASH` 演算法表示相同來源 IP 位址中的流量將傳送至相同的後端網繭。

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入口資源定義的總共 10 個連接埠。
- 由負載平衡器服務和入口資源引用的總共 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入口資源定義的總共 100 個連接埠。

- 由負載平衡器服務和入口資源引用的總共 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入口資源定義的總共 1000 個連接埠。
- 由負載平衡器服務和入口資源引用的總共 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。

設定第 4 層與第 7 層負載平衡的持續性

您可以使用 NCP ConfigMap 中的 `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
```

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。

入口

- NSX-T Data Center 將針對含 TLS 規格的入口和不含 TLS 規格的入口，分別建立一個第 7 層負載平衡器。
- 所有入口將取得單一 IP 位址。
- 會從 `ncp.ini` 的 `[nsx_v3]` 區段中的 `external_ip_pools` 選項所指定的外部 IP 集區，為入口資源配置 IP 位址。負載平衡器將在此 IP 位址及 HTTP 和 HTTPS 連接埠 (80 和 443) 上公開。

- 從 NCP 2.3.2 開始，會從 `ncp.ini` 的 `[nsx_v3]` 區段中的 `external_ip_pools_lb` 選項所指定的外部 IP 集區，為入口資源配置 IP 位址。如果 `external_ip_pools_lb` 選項不存在，則會使用 `external_ip_pools` 所指定的集區。負載平衡器將在此 IP 位址及 HTTP 和 HTTPS 連接埠 (80 和 443) 上公開。
- 從 NCP 2.3.2 開始，您可以透過變更組態並重新啟動 NCP，變更為不同的 IP 集區。
- 您可以指定 TLS 的預設憑證。如需產生憑證並將憑證掛接到 NCP 網繭中的相關資訊，請參閱以下內容。
- 不含 TLS 規格的入口將主控於 HTTP 虛擬伺服器 (連接埠 80) 之上。
- 含 TLS 規格的入口將主控於 HTTPS 虛擬伺服器 (連接埠 443) 之上。負載平衡器將充當 SSL 伺服器，並且會終止用戶端 SSL 連線。
- 密碼與入口的建立順序無關緊要。如果該密碼物件已存在，且有入口參考該密碼物件，則憑證將匯入 NSX-T Data Center 中。如果密碼已刪除或參考該密碼的最後一個入口已刪除，將會刪除與密碼對應的憑證。
- 支援透過新增或移除 TLS 區段來修改入口。從入口規格中移除 `tls` 金鑰時，入口規則將從 HTTPS 虛擬伺服器 (連接埠 443) 傳輸至 HTTP 虛擬伺服器 (連接埠 80)。同樣地，將 `tls` 金鑰新增至入口規格時，入口規則將從 HTTP 虛擬伺服器 (連接埠 80) 傳輸至 HTTPS 虛擬伺服器 (連接埠 443)。
- 如果單一叢集的入口定義中有重複規則，則僅套用第一個規則。
- 每個叢集僅支援具有預設後端的單一入口。不符合任何入口規則的流量將會轉送至預設後端。
- 如果有多個入口具有預設後端，則僅設定第一個入口。其他入口將標註為錯誤。
- 支援使用規則運算式字元「`.`」進行萬用字元 URI 比對。「`*`」進行萬用字元 URI 比對。例如，路徑「`/coffee/*`」與後面不跟字元或後跟一或多個字元的「`/coffee/`」(例如，「`/coffee/`」、「`/coffee/a`」、「`/coffee/b`」) 相符，但是與「`/coffee`」、「`/coffeecup`」或「`/coffeecup/a`」等不符。請注意，如果路徑包含「`/`」，例如「`/tea/*`」，則與後面不跟字元或後跟一或多個字元的「`/tea`」(例如，「`/tea`」、「`/tea/`」、「`/teacup`」、「`/teacup/`」、「`/tea/a`」或「`/teacup/b`」) 相符。在此情況下，規則運算式特殊字元「`*`」同樣用做萬用字元。

入口規格範例：

```
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  rules:
  - http:
      paths:
      - path: /tea/*           #Matches /tea, /tea/, /teacup, /teacup/, /tea/a, /teacup/b, etc.
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee/*      #Matches /coffee/, /coffee/a but NOT /coffee, /coffeecup, etc.
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

- 您可以透過新增註解至入口資源，來設定 URL 要求重寫。例如，

```
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

路徑 `/tea` 和 `/coffee` 將重寫為 `/`，然後再將 URL 傳送至後端服務。

- 從 NCP 2.3.2 開始，支援入口註解 `kubernetes.io/ingress.allow-http`。
 - 如果註解設定為 **false**，僅會建立 HTTPS 規則。
 - 如果註解設定為 **true** 或缺少註解，將會建立 HTTP 規則。此外，如果入口規格中存在 TLS 區段，將會建立 HTTPS 規則。
- 已向入口資源標註錯誤。錯誤索引鍵為 `ncp/error.loadbalancer`，警告索引鍵為 `ncp/warning.loadbalancer`。可能的錯誤和警告如下：
 - `ncp/error.loadbalancer: DEFAULT_BACKEND_IN_USE`

這個錯誤指示具有預設後端的入口已存在。入口將處於非作用中狀態。具有和不具有 TLS 的入口群組只能有一個預設後端。若要修正錯誤，請刪除入口，然後重新建立具有正確規格的入口。
 - `ncp/warning.loadbalancer: SECRET_NOT_FOUND`

這個錯誤指示入口規格中指定的密碼不存在。入口將部分處於作用中狀態。若要修正錯誤，請建立遺失的密碼。請注意，註解中一旦出現警告，在入口資源的生命週期內將不會予以清除。

第 7 層負載平衡器和網路原則

當流量從 NSX 負載平衡器虛擬伺服器轉送至網蔴時，來源 IP 為第 1 層路由器上行連接埠的 IP 位址。此位址位於私人第 1 層傳送網路上，並且會導致以 CIDR 為基礎的網路原則禁止應允許的流量。若要避免此問題，必須設定網路原則，以便第 1 層路由器上行連接埠的 IP 位址是允許的 CIDR 區塊的一部分。此內部 IP 位址將做為 `status.loadbalancer.ingress.ip` 欄位中的入口規格的一部分以及做為入口資源上的註解 (`ncp/internal_ip_for_policy`) 向使用者顯示。

例如，如果虛擬伺服器的外部 IP 位址為 4.4.0.5，內部第 1 層路由器上行連接埠的 IP 位址為 100.64.224.11，則入口規格將為：

```
kind: Ingress
...
status:
  loadBalancer:
    ingress:
      - ip: 4.4.0.5
      - ip: 100.64.224.11
```

入口資源上的註解將為：

```
ncp/internal_ip_for_policy: 100.64.224.11
```

IP 位址 100.64.224.11 必須屬於網路原則的 ipBlock 選取器中允許的 CIDR。例如，

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
...
ingress:
- from:
  - ipBlock:
      cidr: 100.64.224.11/32
```

Kubernetes 負載平衡器服務和第 4 層負載平衡器

- NSX-T Data Center 將為每個服務連接埠建立第 4 層負載平衡器虛擬伺服器和集區。
- TCP 和 UDP 均受支援。
- 每個服務都具有唯一的 IP 位址。
- 會從外部 IP 集區為服務配置 IP 位址。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 規格空白，會從 ncp.ini 的 [nsx_v3] 區段中的 external_ip_pools_lb 選項所指定的外部 IP 集區配置 IP。(附註：此選項僅在 NCP 2.3.2 中可用。)如果 external_ip_pools_lb 選項不存在，則會使用 external_ip_pools 所指定的集區。負載平衡器服務將在此 IP 位址和服務連接埠上公開。
- 從 NCP 2.3.2 開始，您可以透過變更組態並重新啟動 NCP，變更為不同的 IP 集區。
- 從 NCP 2.3.1 開始，loadBalancerIP 所指定的 IP 集區必須有標記 {"ncp/owner": cluster:<cluster>}。
- 已向服務標註錯誤。錯誤索引鍵為 ncp/error.loadbalancer。可能的錯誤如下：
 - ncp/error.loadbalancer: IP_POOL_NOT_FOUND

這個錯誤指示您指定 loadBalancerIP: <nsx-ip-pool>，但 <nsx-ip-pool> 不存在。服務將處於非作用中狀態。若要修正錯誤，請指定有效的 IP 集區，刪除服務，然後再重新建立服務。

- `ncp/error.loadbalancer: IP_POOL_EXHAUSTED`

這個錯誤指示您指定 `loadBalancerIP: <nsx-ip-pool>`，但 IP 集區已用盡其 IP 位址。服務將處於非作用中狀態。若要修正錯誤，請指定具有可用 IP 位址的 IP 集區，刪除服務，然後再重新建立服務。

- `ncp/error.loadbalancer: IP_POOL_NOT_UNIQUE`

這個錯誤指出多個 IP 集區具有 `loadBalancerIP: <nsx-ip-pool>` 所指定的名稱。服務將處於非作用中狀態。

- `ncp/error.loadbalancer: POOL_ACCESS_DENIED`

(適用於 NCP 2.3.1 及更新版本) 這個錯誤指出 `loadBalancerIP` 所指定的 IP 集區沒有標記

`{"ncp/owner": cluster:<cluster>}`，或標記中指定的叢集與 Kubernetes 叢集的名稱不相符。

- 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。

備註 無法為負載平衡器服務或入口資源指派特定的 IP 位址。在建立負載平衡器服務或入口資源時，您所指定的任何位址將遭到忽略。

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 `<filename>.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	附註
狀態	<code>get ncp-master status</code>	用於 Kubernetes 和 PCF。
狀態	取得 <code>ncp-nsx</code> 狀態	用於 Kubernetes 和 PCF。
狀態	取得 <code>ncp-watcher <watcher-name></code>	用於 Kubernetes 和 PCF。
狀態	取得 <code>ncp-watchers</code>	用於 Kubernetes 和 PCF。
狀態	取得 <code>ncp-k8s-api-server</code> 狀態	僅用於 Kubernetes。
狀態	<code>check projects</code>	僅用於 Kubernetes。
狀態	<code>check project <project-name></code>	僅用於 Kubernetes。
狀態	<code>get ncp-bbs status</code>	僅用於 PCF。
狀態	<code>get ncp-capi status</code>	僅用於 PCF。
狀態	<code>get ncp-policy-server status</code>	僅用於 PCF。
快取	取得 <code>project-caches</code>	僅用於 Kubernetes。
快取	取得 <code>project-cache <project-name></code>	僅用於 Kubernetes。
快取	取得 <code>namespace-caches</code>	僅用於 Kubernetes。
快取	取得 <code>namespace-cache <namespace-name></code>	僅用於 Kubernetes。
快取	取得 <code>pod-caches</code>	僅用於 Kubernetes。
快取	取得 <code>pod-cache <pod-name></code>	僅用於 Kubernetes。
快取	取得 <code>ingress-caches</code>	僅用於 Kubernetes。
快取	<code>get ingress-cache <ingress-name></code>	僅用於 Kubernetes。
快取	<code>get ingress-controllers</code>	僅用於 Kubernetes。
快取	<code>get ingress-controller <ingress-controller-name></code>	僅用於 Kubernetes。
快取	<code>get network-policy-caches</code>	僅用於 Kubernetes。
快取	<code>get network-policy-cache <pod-name></code>	僅用於 Kubernetes。
快取	<code>get asg-caches</code>	僅用於 PCF。
快取	<code>get asg-cache <asg-ID></code>	僅用於 PCF。
快取	<code>get org-caches</code>	僅用於 PCF。
快取	<code>get org-cache <org-ID></code>	僅用於 PCF。
快取	<code>get space-caches</code>	僅用於 PCF。

表格 6-1. NCP 容器的 CLI 命令 (續)

類型	命令	附註
快取	get space-cache <space-ID>	僅用於 PCF。
快取	取得 app-caches	僅用於 PCF。
快取	get app-cache <app-ID>	僅用於 PCF。
快取	get instance-caches <app-ID>	僅用於 PCF。
快取	get instance-cache <app-ID> <instance-ID>	僅用於 PCF。
快取	get policy-caches	僅用於 PCF。
支援	取得 ncp-log 檔案 <filename>	用於 Kubernetes 和 PCF。
支援	get ncp-log-level	用於 Kubernetes 和 PCF。
支援	set ncp-log-level <log-level>	用於 Kubernetes 和 PCF。
支援	取得 support-bundle 檔案 <filename>	僅用於 Kubernetes。
支援	取得 node-agent-log 檔案 <filename>	僅用於 Kubernetes。
支援	取得 node-agent-log 檔案 <filename> <node-name>	僅用於 Kubernetes。

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

範例：

```
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
```

```
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
```

■ 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
```

- 檢查 NCP 和 PCF BBS 之間的連線狀態

```
get ncp-bbs status
```

範例：

```
node> get ncp-bbs status  
BBS Server status: Healthy
```

- 檢查 NCP 和 PCF CAPI 之間的連線狀態

```
get ncp-capi status
```

範例：

```
node> get ncp-capi status  
CAPI Server status: Healthy
```

- 檢查 NCP 和 PCF 原則伺服器之間的連線狀態

```
get ncp-policy-server status
```

範例：

```
node> get ncp-bbs status  
Policy Server status: Healthy
```

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

```

■ 取得網蔴的內部快取

```

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

```

範例：

```

kubenode> get pod-caches
nsx.default.nginx-rc-ug2lv:
  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-ug2lv
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

```

■ 取得所有入口快取或特定快取

```

get ingress caches
get ingress-cache <ingress-name>

```

範例：

```

kubenode> get ingress-caches
nsx.default.cafe-ingress:
  ext_pool_id: cc02db70-539a-4934-a938-5b851b3e485b
  lb_virtual_server:
    id: 895c7f43-c56e-4b67-bb4c-09d68459d416
    lb_service_id: 659eefc6-33d1-4672-a419-344b877f528e
    name: dgo2-http
    type: http
  lb_virtual_server_ip: 5.5.0.2
  name: cafe-ingress
  rules:
    host: cafe.example.com
    http:
      paths:
        path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
      lb_rule:
        id: 4bc16bdd-abd9-47fb-a09e-21e58b2131c3
        name: dgo2-default-cafe-ingress/coffee

kubenode> get ingress-cache nsx.default.cafe-ingress
ext_pool_id: cc02db70-539a-4934-a938-5b851b3e485b
lb_virtual_server:
  id: 895c7f43-c56e-4b67-bb4c-09d68459d416
  lb_service_id: 659eefc6-33d1-4672-a419-344b877f528e
  name: dgo2-http
  type: http
lb_virtual_server_ip: 5.5.0.2
name: cafe-ingress
rules:
  host: cafe.example.com
  http:
    paths:
      path: /coffee

```



```

backend:
  serviceName: coffee-svc
  servicePort: 80
lb_rule:
  id: 4bc16bdd-abd9-47fb-a09e-21e58b2131c3
  name: dgo2-default-cafe-ingress/coffee

```

- 取得所有入口控制器或特定控制器的相關資訊，包括已停用的控制器

```

get ingress controllers
get ingress-controller <ingress-controller-name>

```

範例：

```

kubenode> get ingress-controllers
native-load-balancer:
  ingress_virtual_server:
    http:
      default_backend_tags:
        id: 895c7f43-c56e-4b67-bb4c-09d68459d416
        pool_id: None
      https_terminated:
        default_backend_tags:
          id: 293282eb-f1a0-471c-9e48-ba28d9d89161
          pool_id: None
        lb_ip_pool_id: cc02db70-539a-4934-a938-5b851b3e485b
    loadbalancer_service:
      first_avail_index: 0
    lb_services:
      id: 659eefc6-33d1-4672-a419-344b877f528e
      name: dgo2-bfmxi
      t1_link_port_ip: 100.64.128.5
      t1_router_id: cb50deb2-4460-45f2-879a-1b94592ae886
      virtual_servers:
        293282eb-f1a0-471c-9e48-ba28d9d89161
        895c7f43-c56e-4b67-bb4c-09d68459d416
    ssl:
      ssl_client_profile_id: aff205bb-4db8-5a72-8d67-218cdc54d27b
      vip: 5.5.0.2

nsx.default.nginx-ingress-rc-host-ed3og
ip: 10.192.162.201
mode: hostnetwork
pool_id: 5813c609-5d3a-4438-b9c3-ea3cd6de52c3

kubenode> get ingress-controller native-load-balancer
ingress_virtual_server:
  http:
    default_backend_tags:
      id: 895c7f43-c56e-4b67-bb4c-09d68459d416
      pool_id: None
    https_terminated:

```

```

    default_backend_tags:
      id: 293282eb-f1a0-471c-9e48-ba28d9d89161
      pool_id: None
    lb_ip_pool_id: cc02db70-539a-4934-a938-5b851b3e485b
    loadbalancer_service:
      first_avail_index: 0
    lb_services:
      id: 659eefc6-33d1-4672-a419-344b877f528e
      name: dgo2-bfmxi
      t1_link_port_ip: 100.64.128.5
      t1_router_id: cb50deb2-4460-45f2-879a-1b94592ae886
      virtual_servers:
        293282eb-f1a0-471c-9e48-ba28d9d89161
        895c7f43-c56e-4b67-bb4c-09d68459d416
    ssl:
      ssl_client_profile_id: aff205bb-4db8-5a72-8d67-218cdc54d27b
    vip: 5.5.0.2

```

■ 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:
    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

```

■ 取得所有 ASG 快取或特定快取

```

get asg-caches
get asg-cache <asg-ID>

```

範例：

```

node> get asg-caches
edc04715-d04c-4e63-abb0-db601a668db6:
    fws_id: 3c66f40a-5378-46d7-a7e2-bee4ba72a4cc
    name: org-85_tcp_80_asg

```

```

rules:
  destinations:
    66.10.10.0/24
  ports:
    80
  protocol: tcp
  rule_id: 4359
running_default: False
running_spaces:
  75bc164d-1214-46f9-80bb-456a8fbccbfd
staging_default: False
staging_spaces:

node> get asg-cache edc04715-d04c-4e63-abbc-db601a668db6
fws_id: 3c66f40a-5378-46d7-a7e2-bee4ba72a4cc
name: org-85_tcp_80_asg
rules:
  destinations:
    66.10.10.0/24
  ports:
    80
  protocol: tcp
  rule_id: 4359
running_default: False
running_spaces:
  75bc164d-1214-46f9-80bb-456a8fbccbfd
staging_default: False
staging_spaces:

```

■ 取得所有組織快取或特定快取

```

get org-caches
get org-cache <org-ID>

```

範例：

```

node> get org-caches
ebb8b4f9-a40f-4122-bf21-65c40f575aca:
  ext_pool_id: 9208a8b8-57d7-4582-9c1f-7a7cefa104f5
  isolation:
    isolation_section_id: d6e7ff95-4737-4e34-91d4-27601897353f
  logical-router: 94a414a2-551e-4444-bae6-3d79901a165f
  logical-switch:
    id: d74807e8-8f74-4575-b26b-87d4fdbafd3c
    ip_pool_id: 1b60f16f-4a30-4a3d-93cc-bfb08a5e3e02
    subnet: 50.0.48.0/24
    subnet_id: a458d3aa-bea9-4684-9957-d0ce80d11788
  name: org-50
  snat_ip: 70.0.0.49
  spaces:
    e8ab7aa0-d4e3-4458-a896-f33177557851

```

```
node> get org-cache ebb8b4f9-a40f-4122-bf21-65c40f575aca
ext_pool_id: 9208a8b8-57d7-4582-9c1f-7a7cefa104f5
isolation:
  isolation_section_id: d6e7ff95-4737-4e34-91d4-27601897353f
logical-router: 94a414a2-551e-4444-bae6-3d79901a165f
logical-switch:
  id: d74807e8-8f74-4575-b26b-87d4fdbafd3c
  ip_pool_id: 1b60f16f-4a30-4a3d-93cc-bfb08a5e3e02
  subnet: 50.0.48.0/24
  subnet_id: a458d3aa-bea9-4684-9957-d0ce80d11788
name: org-50
snat_ip: 70.0.0.49
spaces:
  e8ab7aa0-d4e3-4458-a896-f33177557851
```

■ 取得所有空間快取或特定快取

```
get space-caches
get space-cache <space-ID>
```

範例：

```
node> get space-caches
global_security_group:
  name: global_security_group
  running_nsgroup: 226d4292-47fb-4c2e-a118-449818d8fa98
  staging_nsgroup: 7ebbf7f5-38c9-43a3-9292-682056722836

7870d134-7997-4373-b665-b6a910413c47:
  name: test-space1
  org_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
  running_nsgroup: 4a3d9bcc-be36-47ae-bff8-96448fecf307
  running_security_groups:
    aa0c7c3f-a478-4d45-8afa-df5d5d7dc512
  staging_security_groups:
    aa0c7c3f-a478-4d45-8afa-df5d5d7dc512

node> get space-cache 7870d134-7997-4373-b665-b6a910413c47
name: test-space1
org_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
running_nsgroup: 4a3d9bcc-be36-47ae-bff8-96448fecf307
running_security_groups:
  aa0c7c3f-a478-4d45-8afa-df5d5d7dc512
staging_security_groups:
  aa0c7c3f-a478-4d45-8afa-df5d5d7dc512
```

■ 取得所有應用程式快取或特定快取

```
get app-caches
get app-cache <app-ID>
```

範例：

```
node> get app-caches
aff2b12b-b425-4d9f-b8e6-b6308644efa8:
  instances:
    b72199cc-e1ab-49bf-506d-478d:
      app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
      cell_id: 0dda88bc-640b-44e7-8cea-20e83e873544
      cif_id: 158a1d7e-6ccc-4027-a773-55bb2618f51b
      gateway_ip: 192.168.5.1
      host_vif: 53475dfd-03e4-4bc6-b8ba-3d803725cbab
      id: b72199cc-e1ab-49bf-506d-478d
      index: 0
      ip: 192.168.5.4/24
      last_updated_time: 1522965828.45
      mac: 02:50:56:00:60:02
      port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
      state: RUNNING
      vlan: 3
      name: hello2
      rg_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
      space_id: 7870d134-7997-4373-b665-b6a910413c47

node> get app-cache aff2b12b-b425-4d9f-b8e6-b6308644efa8
instances:
  b72199cc-e1ab-49bf-506d-478d:
    app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
    cell_id: 0dda88bc-640b-44e7-8cea-20e83e873544
    cif_id: 158a1d7e-6ccc-4027-a773-55bb2618f51b
    gateway_ip: 192.168.5.1
    host_vif: 53475dfd-03e4-4bc6-b8ba-3d803725cbab
    id: b72199cc-e1ab-49bf-506d-478d
    index: 0
    ip: 192.168.5.4/24
    last_updated_time: 1522965828.45
    mac: 02:50:56:00:60:02
    port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
    state: RUNNING
    vlan: 3
    name: hello2
    org_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
    space_id: 7870d134-7997-4373-b665-b6a910413c47
```

- 取得應用程式的所有執行個體快取或特定執行個體快取

```
get instance-caches <app-ID>
get instance-cache <app-ID> <instance-ID>
```

範例：

```
node> get instance-caches aff2b12b-b425-4d9f-b8e6-b6308644efa8
b72199cc-e1ab-49bf-506d-478d:
  app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
  cell_id: 0dda88bc-640b-44e7-8cea-20e83e873544
  cif_id: 158a1d7e-6ccc-4027-a773-55bb2618f51b
  gateway_ip: 192.168.5.1
  host_vif: 53475dfd-03e4-4bc6-b8ba-3d803725cbab
  id: b72199cc-e1ab-49bf-506d-478d
  index: 0
  ip: 192.168.5.4/24
  last_updated_time: 1522965828.45
  mac: 02:50:56:00:60:02
  port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
  state: RUNNING
  vlan: 3

node> get instance-cache aff2b12b-b425-4d9f-b8e6-b6308644efa8 b72199cc-e1ab-49bf-506d-478d
  app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
  cell_id: 0dda88bc-640b-44e7-8cea-20e83e873544
  cif_id: 158a1d7e-6ccc-4027-a773-55bb2618f51b
  gateway_ip: 192.168.5.1
  host_vif: 53475dfd-03e4-4bc6-b8ba-3d803725cbab
  id: b72199cc-e1ab-49bf-506d-478d
  index: 0
  ip: 192.168.5.4/24
  last_updated_time: 1522965828.45
  mac: 02:50:56:00:60:02
  port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
  state: RUNNING
  vlan: 3
```

■ 取得所有原則快取

```
get policy-caches
```

範例：

```
node> get policy-caches
aff2b12b-b425-4d9f-b8e6-b6308644efa8:
  fws_id: 3fe27725-f139-479a-b83b-8576c9aedbef
  nsg_id: 30583a27-9b56-49c1-a534-4040f91cc333
  rules:
    8272:
      dst_app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
      ports: 8382
      protocol: tcp
      src_app_id: f582ec4d-3a13-440a-afbd-97b7bfae21d1

f582ec4d-3a13-440a-afbd-97b7bfae21d1:
  nsg_id: d24b9f77-e2e0-4fba-b258-893223683aa6
```

```
rules:
  8272:
    dst_app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
    ports: 8382
    protocol: tcp
    src_app_id: f582ec4d-3a13-440a-afbd-97b7bfae21d1
```

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：

```
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

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
```

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:

```
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

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
```

■ 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

錯誤碼	說明
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

錯誤碼	說明
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
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

錯誤碼	說明
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
NCP00091	負載平衡器規則快取初始化錯誤
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

錯誤碼	說明
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

錯誤碼	說明
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
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

Pivotal Cloud Foundry 錯誤碼

錯誤碼	說明
NCP06001	PCF BBS 連線失敗
NCP06002	PCF CAPI 連線失敗
NCP06006	找不到 PCF 快取
NCP06007	PCF 未知的網域
NCP06020	PCF 原則伺服器連線失敗
NCP06021	PCF 原則處理失敗
NCP06030	PCF 事件處理失敗
NCP06031	PCF 未預期的事件類型
NCP06032	PCF 未預期的事件執行個體
NCP06033	PCF 工作刪除失敗
NCP06034	PCF 檔案存取失敗