

Guía de instalación y administración de NSX Container Plug-in para Kubernetes y Cloud Foundry

VMware NSX Container Plug-in 2.4, 2.4.1

VMware NSX-T Data Center 2.4

Puede encontrar la documentación técnica más actualizada en el sitio web de VMware:

<https://docs.vmware.com/es/>

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

VMware Spain, S.L.
Calle Rafael Boti 26
2.ª planta
Madrid 28023
Tel.: +34 914125000
www.vmware.com/es

Copyright © 2017-2019 VMware, Inc. Todos los derechos reservados. [Información sobre el copyright y la marca comercial.](#)

Contenido

Guía de instalación y administración de NSX Container Plug-in para Kubernetes y Cloud Foundry 5

1 Descripción general de NSX Container Plug-in 6

Requisitos de compatibilidad 7

Descripción general de la instalación 8

Actualización de NCP en un entorno de Kubernetes 8

Actualización de NCP en un entorno de Pivotal Cloud Foundry 9

2 Configurar recursos de NSX-T 11

Configurar los recursos de NSX-T 11

3 Instalación de NCP en un entorno de Kubernetes 18

Instalar el complemento CNI de NSX-T Data Center 18

Instalar y configurar OVS 19

Configurar las redes de NSX-T Data Center para los nodos de Kubernetes 21

Instalar el agente del nodo de NSX 22

ConfigMap para ncp.ini in nsx-node-agent-ds.yml 23

Instalar NSX Container Plug-in 26

ConfigMap para ncp.ini en ncp-rc.yml 28

Montar un certificado codificado en formato PEM y una clave privada en el pod de NCP 34

Montar un archivo de certificado en el pod de NCP 34

Configurar syslog 35

Crear un contenedor sidecar para syslog 35

Crear una réplica de DaemonSet para Syslog 37

Ejemplo: Configuración de la rotación de registros y la ejecución de Syslog en un contenedor sidecar 39

Consideraciones de seguridad 46

Consejos sobre la configuración de recursos de red 49

4 Instalación de NCP en un entorno de Pivotal Cloud Foundry 51

Instalación de NCP en un entorno de Pivotal Cloud Foundry 51

5 Equilibrio de carga 54

Configuración del equilibrio de carga 54

Controladores de entrada de terceros 62

6 Administrar NSX Container Plug-in 65

[Visualización de la información de errores almacenada en objetos NSXError de recursos de Kubernetes](#) 65

[Puertos lógicos conectados a CIF](#) 66

[Comandos de la CLI](#) 67

[Códigos de error](#) 86

Guía de instalación y administración de NSX Container Plug-in para Kubernetes y Cloud Foundry

En esta guía se describe cómo instalar y administrar NSX Container Plug-in (NCP) para proporcionar integración entre NSX-T Data Center y Kubernetes, así como entre NSX-T Data Center y Pivotal Cloud Foundry (PCF).

Público objetivo

Esta guía está destinada a administradores de red y de sistemas. Se supone que está familiarizado con la instalación y administración de NSX-T Data Center, Kubernetes y Pivotal Cloud Foundry.

Glosario de publicaciones técnicas de VMware

Publicaciones técnicas de VMware proporciona un glosario de términos que podrían resultarle desconocidos. Si desea ver las definiciones de los términos que se utilizan en la documentación técnica de VMware, acceda a la página <http://www.vmware.com/support/pubs>.

Descripción general de NSX Container Plug-in

1

NSX Container Plug-in (NCP) integra NSX-T Data Center y orquestadores de contenedores, como Kubernetes, así como NSX-T Data Center y productos de PaaS (plataforma como servicio) basados en contenedores, como OpenShift y Pivotal Cloud Foundry. En esta guía se describe cómo configurar NCP con Kubernetes y Pivotal Cloud Foundry.

El componente principal de NCP se ejecuta en un contenedor y se comunica con NSX Manager y con el plano de control de Kubernetes. NCP supervisa los cambios de los contenedores y otros recursos, y administra los recursos de redes, como los puertos lógicos, los conmutadores, los enrutadores y los grupos de seguridad de los contenedores realizando llamadas a NSX API.

El complemento CNI de NSX se ejecuta en cada nodo de Kubernetes. Supervisa los eventos del ciclo de vida de los contenedores, conecta una interfaz de contenedor al vSwitch invitado y programa este vSwitch para que etiquete y reenvíe el tráfico entre las interfaces de contenedor y la VNIC.

NCP ofrece las siguientes funcionalidades:

- Crea de manera automática una topología lógica de NSX-T Data Center para un clúster de Kubernetes y crea una red lógica independiente para cada espacio de nombres de Kubernetes.
- Conecta pods de Kubernetes a la red lógica y asigna direcciones IP y MAC.
- Admite la traducción de direcciones de red (Network Address Translation, NAT) y asigna una dirección IP de SNAT independiente para cada espacio de nombres de Kubernetes.

Nota Al configurar NAT, el número total de direcciones IP traducidas no puede ser superior a 1000.

- Implementa directivas de red de Kubernetes con el firewall distribuido de NSX-T Data Center.
 - Compatibilidad con las directivas de red de entrada y salida.
 - Compatibilidad con el selector IPBlock en las directivas de red.
 - Compatibilidad con `matchLabels` y `matchExpression` al especificar selectores de etiquetas para directivas de redes.
 - Compatibilidad con la selección de pods en otro espacio de nombres.

- Implementa el servicio de Kubernetes de tipo ClusterIP y el servicio de tipo LoadBalancer.
- Implementa la entrada de Kubernetes con el equilibrador de carga de capa 7 de NSX-T.
 - Compatibilidad con las entradas de HTTP y de HTTPS con terminación de Edge de TLS.
 - Compatibilidad con la configuración de back-end predeterminada de entrada.
 - Compatibilidad con la reescritura del URI de entrada.
- Crea etiquetas en el puerto de conmutador lógico de NSX-T Data Center para el espacio de nombres, el nombre de pod y las etiquetas de un pod, y permite al administrador definir directivas y grupos de seguridad de NSX-T con base en etiquetas.

En esta versión, NCP solo admite un clúster de Kubernetes. Puede tener varios clústeres de Kubernetes, cada uno con una instancia NCP particular, con la misma implementación de NSX-T Data Center.

Este capítulo incluye los siguientes temas:

- [Requisitos de compatibilidad](#)
- [Descripción general de la instalación](#)
- [Actualización de NCP en un entorno de Kubernetes](#)
- [Actualización de NCP en un entorno de Pivotal Cloud Foundry](#)

Requisitos de compatibilidad

NSX Container Plug-in (NCP) tiene los siguientes requisitos de compatibilidad para un entorno de Kubernetes y un entorno de Pivotal Cloud Foundry (PCF).

Tabla 1-1. Requisitos de compatibilidad para un entorno de Kubernetes

Productos de software	Versión
NSX-T Data Center	2.3, 2.4
Hipervisor para las máquinas virtuales de los hosts del contenedor	<ul style="list-style-type: none"> ■ Versión de vSphere admitida ■ RHEL KVM 7.4, 7.5, 7.6 ■ Ubuntu KVM 16.04
Sistema operativo del host del contenedor	<ul style="list-style-type: none"> ■ RHEL 7.5, 7.6 ■ Ubuntu 16.04 ■ CentOS 7.4, 7.5
Orquestador del contenedor	Kubernetes 1.12, 1.13
Open vSwitch del host del contenedor	2.9.1 (se incluye con NSX-T Data Center 2.3.x), 2.10.2 (se incluye con NSX-T Data Center 2.4.0)

Tabla 1-2. Requisitos de compatibilidad para un entorno de Cloud Foundry

Productos de software	Versión
Hipervisor para las máquinas virtuales de los hosts del contenedor	■ Versión de vSphere admitida
Orquestador del contenedor	<ul style="list-style-type: none"> ■ Pivotal Application Service 2.3.x y Pivotal Operations Manager 2.3.x ■ Pivotal Application Service 2.4.x (excepto 2.4.0) y Pivotal Operations Manager 2.4.x (excepto 2.4.0)

Descripción general de la instalación

En un entorno con Kubernetes, la instalación y configuración de NCP suele incluir los siguientes pasos. Para realizar los pasos correctamente, debe estar familiarizado con la instalación y administración de NSX-T Data Center y Kubernetes.

- 1 Instale NSX-T Data Center.
- 2 Crear una zona de transporte superpuesta.
- 3 Crear un conmutador lógico superpuesto y conectar los nodos de Kubernetes al conmutador.
- 4 Crear un enrutador lógico de nivel 0.
- 5 Crear bloques de IP para los pods de Kubernetes.
- 6 Crear grupos de IP para la traducción de direcciones de red de origen (Source Network Address Translation, SNAT).
- 7 Instalar el complemento CNI (interfaz de red de los contenedores) de NSX en cada nodo.
- 8 Instalar OVS (Open vSwitch) en cada nodo.
- 9 Configurar las redes de NSX-T para los nodos de Kubernetes.
- 10 Instalar el agente del nodo de NSX como DaemonSet.
- 11 Instalar NCP como ReplicationController.
- 12 Montar certificados de seguridad en el pod de NCP.

Actualización de NCP en un entorno de Kubernetes

Esta sección describe cómo actualizar NCP a la versión 2.4.0 en un entorno de Kubernetes.

Procedimiento

- 1 Actualice el ReplicationController de NCP con el siguiente comando (reemplace <image> con el nombre real de la imagen).

```
kubectl rolling-update nsx-ncp -n nsx-system --image=<image>
```


- 2 Actualice el DaemonSet del agente del nodo de NSX con los siguientes comandos (reemplace <image> con el nombre real de la imagen).

```
kubectl set image ds nsx-node-agent -n nsx-system nsx-node-agent=<image>
kubectl set image ds nsx-node-agent -n nsx-system nsx-kube-proxy=<images>
kubectl rollout status ds/nsx-node-agent -nnsx-system
```

- 3 Actualice el paquete DEB/RPM de CNI a la versión 2.4.0 con el siguiente comando (reemplace <cni deb> y <cni rpm> con el nombre real del paquete).

En Ubuntu:

```
dpkg -i <cni deb>
```

En RHEL o CentOS:

```
rpm -U <cni rpm>
```

- 4 (opcional) Actualice NSX-T Data Center a la versión 2.4.

Si el hipervisor es ESXi, actualícelo al menos a la versión 6.5p03 desde 6.5 o a la versión 6.7ep6 desde 6.7 antes de actualizar NSX-T Data Center.

- 5 (opcional) Actualice el hipervisor (KVM o contenedor sin sistema operativo).
- 6 (opcional) Actualice el host del contenedor (RHEL, Ubuntu o CentOS).
- 7 (opcional) Actualice Kubernetes.
- 8 (opcional) Actualice OVS.

Para un contenedor sin sistema operativo, la actualización de NSX-T Data Center también actualiza OVS, por lo que este paso no es necesario.

Durante este paso, puede que vea errores de comunicación transitorios entre nsx-kube-proxy y nsx-node-agent. Este es el comportamiento esperado y no indica un problema.

Actualización de NCP en un entorno de Pivotal Cloud Foundry

Esta sección describe cómo actualizar NCP a la versión 2.4.0 en un entorno de Pivotal Cloud Foundry.

Procedimiento

- 1 Actualice el mosaico NSX-T o NCP a la versión 2.4.0.
- 2 (opcional) Actualice Pivotal Operations Manager y, a continuación, actualice Pivotal Application Service.
- 3 (opcional) Actualice NSX-T Data Center a la versión 2.4.

Si el hipervisor es ESXi, actualícelo al menos a la versión 6.5p03 desde 6.5 o a la versión 6.7ep6 desde 6.7 antes de actualizar NSX-T Data Center.

- 4 (opcional) Actualice el hipervisor (KVM o contenedor sin sistema operativo).

Configurar recursos de NSX-T

2

Antes de instalar NSX Container Plug-in, debe configurar algunos recursos de NSX-T Data Center.

Este capítulo incluye los siguientes temas:

- [Configurar los recursos de NSX-T](#)

Configurar los recursos de NSX-T

Los recursos de NSX-T Data Center que necesita configurar incluyen una zona de transporte de superposición, un enrutador lógico de nivel 0, un conmutador lógico para conectar las máquinas virtuales del nodo, bloques de IP para los nodos de Kubernetes y un grupo de IP para SNAT.

Importante Si ejecuta NSX-T Data Center 2.4 o una versión posterior, debe configurar los recursos de NSX-T mediante la pestaña **Opciones avanzadas de redes y seguridad**.

En el archivo de configuración de NCP `ncp.ini`, los recursos de NSX-T Data Center se especifican mediante sus UUID o nombres.

Zona de transporte superpuesta

Inicie sesión en NSX Manager y desplácese hasta **Sistema > Tejido > Zonas de transporte**. Encuentre la zona de transporte superpuesta que se usa para las redes de los contenedores o cree una nueva.

Especifique una zona de transporte superpuesta de un clúster configurando la opción `overlay_tz` en la sección `[nsx_v3]` de `ncp.ini`. Este paso es opcional. Si no se configura `overlay_tz`, NCP recuperará automáticamente el identificador de zona de transporte superpuesta del enrutador de nivel 0.

Enrutamiento lógico de nivel 0

Inicie sesión en NSX Manager y desplácese hasta **Opciones avanzadas de redes y seguridad > Redes > Enrutadores**. Encuentre el enrutador que se usa para las redes de los contenedores o cree uno nuevo.

Especifique un enrutador lógico de nivel 0 de un clúster configurando la opción `tier0_router` en la sección `[nsx_v3]` de `ncp.ini`.

Nota El enrutador se debe crear en modo activo-en espera.

Conmutador lógico

Las vNIC que usan el nodo para el tráfico de datos deben estar conectadas a un conmutador lógico superpuesto. No es obligatorio que la interfaz de administración del nodo esté conectada a NSX-T Data Center, aunque hacerlo facilitaría la configuración. Para crear un conmutador lógico, inicie sesión en NSX Manager y desplácese hasta **Opciones avanzadas de redes y seguridad > Redes > Conmutación > Conmutadores**. En el conmutador, cree puertos lógicos y asocie las vNIC del nodo a ellos. Los puertos lógicos deben tener las siguientes etiquetas:

- etiqueta: `<cluster_name>`, ámbito: `ncp/cluster`
- etiqueta: `<node_name>`, ámbito: `ncp/node_name`

El valor de `<cluster_name>` debe coincidir con el valor de la opción `cluster` de la sección `[coe]` de `ncp.ini`.

Bloques de IP para los pods de Kubernetes

Inicie sesión en NSX Manager y desplácese hasta **Opciones avanzadas de redes y seguridad > Redes > IPAM** para crear uno o varios bloques de direcciones IP. Especifique el bloque de IP en formato CIDR

Especifique bloques de IP de pods de Kubernetes configurando la opción `container_ip_blocks` en la sección `[nsx_v3]` de `ncp.ini`.

También puede crear bloques de IP específicos para los espacios de nombres que no sean SNAT (para Kubernetes) o los clústeres (para PCF).

Especifique bloques de IP que no sean SNAT configurando la opción `no_snat_ip_blocks` en la sección `[nsx_v3]` de `ncp.ini`.

Si crea bloques de IP que no sean SNAT mientras NCP se ejecuta, debe reiniciar NCP. De lo contrario, NCP seguirá usando los bloques de IP compartidos hasta que se agoten.

Nota Cuando cree un bloque de IP, el prefijo no debe tener una longitud superior al valor del parámetro `subnet_prefix` en el archivo de configuración de NCP `ncp.ini`. Para obtener más información, consulte [ConfigMap para ncp.ini en ncp-rc.yml](#).

Grupo de IP de SNAT

Un grupo de direcciones IP en NSX Manager se emplea para asignar las direcciones IP que se usarán para traducir IP de pods mediante reglas SNAT y para exponer las controladoras de entrada a través de reglas SNAT/DNAT, igual que las IP flotantes de OpenStack. Estas direcciones IP también se denominan "IP externas".

Varios clústeres de Kubernetes usan el mismo grupo de IP externas. Cada instancia de NCP usa un subgrupo de este grupo para el clúster de Kubernetes que administra. De forma predeterminada, se usará el mismo prefijo de subred para subredes de pods. Para usar un tamaño de subred diferente, actualice la opción `external_subnet_prefix` en la sección `[nsx_v3]` de `ncp.ini`.

Puede especificar grupos de direcciones IP para SNAT configurando la opción `external_ip_pools` en la sección `[nsx_v3]` de `ncp.ini`.

Puede utilizar otro grupo de direcciones IP cambiando el archivo de configuración y reiniciando NCP.

Restringir un grupo de direcciones IP de SNAT a espacios de nombres de Kubernetes u organizaciones de PCF específicos

Puede especificar el espacio de nombres de Kubernetes o la organización de PCF a los que es posible asignar direcciones IP del grupo de direcciones IP de SNAT agregando las siguientes etiquetas al grupo de direcciones IP.

- Para un espacio de nombres de Kubernetes: `scope: ncp/owner, tag: ns:<namespace_UUID>`
- Para una organización de PCF: `scope: ncp/owner, tag: org:<org_UUID>`

Puede obtener el UUID del espacio de nombres o de la organización mediante uno de los siguientes comandos:

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

Tenga en cuenta lo siguiente:

- Cada etiqueta debe especificar un UUID. Puede crear varias etiquetas para el mismo grupo.
- Si se cambian las etiquetas después de que se asignen direcciones IP a algunos espacios de nombres u organizaciones en función de las etiquetas anteriores, no se recuperarán dichas direcciones IP hasta que se cambien los ajustes de SNAT de los servicios de Kubernetes o las aplicaciones de PCF, o bien se reinicie NCP.
- Las etiquetas de propietario del espacio de nombres y de la organización de PCF son opcionales. Sin estas etiquetas, se pueden asignar direcciones IP del grupo de direcciones IP de SNAT a cualquier espacio de nombres u organización de PCF.

Configurar un grupo de direcciones IP de SNAT para un servicio

Puede configurar un grupo de direcciones IP de SNAT para un servicio agregando una anotación al servicio. Por ejemplo,

```
apiVersion: v1
kind: Service
metadata:
  name: svc-example
  annotations:
```

```
ncp/snat_pool: <external IP pool ID or name>
selector:
  app: example
...
```

El grupo de direcciones IP especificado por `ncp/snat_pool` debe tener la etiqueta `scope: ncp/owner`, `tag: cluster:<cluster_name>`.

NCP configurará la regla SNAT para este servicio. La IP de origen de la regla es el conjunto de pods de back-end. La IP de destino es la IP SNAT asignada desde el grupo de IP externo especificado. Si se produce un error cuando NCP configura la regla SNAT, se agregará la nota `ncp/error.snat:<error>` al servicio. Los posibles errores son:

- `IP_POOL_NOT_FOUND`: el grupo de IP de SNAT no se encuentra en NSX Manager.
- `IP_POOL_EXHAUSTED`: el grupo de IP de SNAT se agotó.
- `IP_POOL_NOT_UNIQUE`: el grupo que `ncp/snat_pool` especifica hace referencia a varios grupos en NSX Manager.
- `SNAT_POOL_ACCESS_DENY`: la etiqueta de propietario del grupo no coincide con el espacio de nombres del servicio que envía la solicitud de asignación.
- `SNAT_RULE_OVERLAPPED`: se crea una nueva regla SNAT, pero el pod del servicio SNAT también pertenece a otro servicio SNAT (es decir, hay varias reglas SNAT para el mismo pod).
- `POOL_ACCESS_DENIED`: el grupo de direcciones IP que `ncp/snat_pool` especifica no tiene la etiqueta `scope: ncp/owner`, `tag: cluster:<cluster_name>` o la etiqueta de propietario del grupo no coincide con el espacio de nombres del servicio que envía la solicitud de asignación.

Tenga en cuenta lo siguiente:

- El grupo especificado por `ncp/snat_pool` ya debe existir en NSX-T Data Center antes de configurar el servicio.
- En NSX-T Data Center, la prioridad de la regla SNAT para el servicio es mayor que la del proyecto.
- Si se configura un pod con varias reglas SNAT, solo funcionará una.
- Puede utilizar otro grupo de direcciones IP cambiando la anotación y reiniciando NCP.

Configurar un grupo de direcciones IP de SNAT para un espacio de nombres

Puede configurar un grupo de direcciones IP de SNAT para un espacio de nombres agregando una anotación al espacio de nombres. Por ejemplo,

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

NCP configurará la regla SNAT para este espacio de nombres. La IP de origen de la regla es el conjunto de pods de back-end. La IP de destino es la IP SNAT asignada desde el grupo de IP externo especificado. Si se produce un error cuando NCP configura la regla SNAT, se agregará la nota `ncp/error.snat:<error>` al espacio de nombres. Los posibles errores son:

- `IP_POOL_NOT_FOUND`: el grupo de IP de SNAT no se encuentra en NSX Manager.
- `IP_POOL_EXHAUSTED`: el grupo de IP de SNAT se agotó.
- `IP_POOL_NOT_UNIQUE`: el grupo que `ncp/snat_pool` especifica hace referencia a varios grupos en NSX Manager.
- `POOL_ACCESS_DENIED`: el grupo de direcciones IP que `ncp/snat_pool` especifica no tiene la etiqueta `scope: ncp/owner`, `tag: cluster:<cluster_name>` o la etiqueta de propietario del grupo no coincide con el espacio de nombres que envía la solicitud de asignación.

Tenga en cuenta lo siguiente:

- Solo puede especificar un grupo de direcciones IP de SNAT en la anotación.
- No es necesario que el grupo de direcciones IP de SNAT esté configurado en `ncp.ini`.
- El grupo de direcciones IP especificado por `ncp/snat_pool` debe tener la etiqueta `scope: ncp/owner`, `tag: cluster:<cluster_name>`.
- El grupo de direcciones IP que `ncp/snat_pool` especifica también puede tener una etiqueta de espacio de nombres `scope: ncp/owner`, `tag: ns:<namespace_UUID>`.
- Si falta la anotación `ncp/snat_pool`, el espacio de nombres utilizará el grupo de direcciones IP de SNAT para el clúster.
- Puede utilizar otro grupo de direcciones IP cambiando la anotación y reiniciando NCP.

Configurar un grupo de SNAT para una aplicación PAS

NCP configura la IP de SNAT de una organización de PAS (Pivotal Application Service) de forma predeterminada. Puede configurar una IP de SNAT para una aplicación mediante la creación de una aplicación con un archivo `manifest.xml` que contiene la información del grupo de direcciones IP de SNAT. Por ejemplo,

```
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 configurará la regla SNAT de esta aplicación. La IP de origen de la regla es el conjunto de direcciones IP de las instancias, mientras que la IP de destino es la IP de SNAT que se asigna a partir de un grupo de direcciones IP externo. Tenga en cuenta lo siguiente:

- El grupo que especifica `NCP_SNAT_POOL` ya debe existir en NSX-T Data Center antes de insertar la aplicación.
- La prioridad de la regla SNAT de una aplicación es mayor que la de una organización.
- Una aplicación puede configurarse con solo una IP de SNAT.
- Puede utilizar otro grupo de direcciones IP cambiando la configuración y reiniciando NCP.

Configurar SNAT para la versión 3 de PCF

Con la versión 3 de PCF, puede configurar SNAT de dos formas:

- Configure `NCP_SNAT_POOL` en `manifest.yml` al crear la aplicación.

Por ejemplo, la aplicación se denomina `bread` y el archivo `manifest.yml` tiene la siguiente información:

```
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
```

Ejecute los siguientes comandos:

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

- Configure NCP_SNAT_P00L mediante el comando `cf v3-set-env`.

Ejecute los siguientes comandos (se asume que la aplicación se denomina `app3`):

```
cf v3-set-env app3 NCP_SNAT_P00L AppSnatExternalIppool
(optional) cf v3-stage app3 -package-guid <package-guid> (You can get package-guid with "cf v3-
packages app3".)
cf v3-restart app3
```

(Opcional) (Solo para Kubernetes) Secciones de marcador de firewall

Para que el administrador pueda crear reglas de firewall que no interfieran con las secciones de firewall creadas por NCP en función de las directivas de red, inicie sesión en NSX Manager, desplácese hasta **Seguridad > Firewall distribuido > General** y cree dos secciones de firewall.

Especifique secciones de firewall del marcador configurando las opciones `bottom_firewall_section_marker` y `top_firewall_section_marker` en la sección `[nsx_v3]` de `ncp.ini`.

La sección del firewall inferior debe estar bajo la sección del firewall superior. Con estas secciones del firewall, todas las secciones del firewall creadas por NCP para el aislamiento se crearán sobre la sección del firewall inferior, y todas las secciones del firewall creadas por NCP para la directiva se crearán bajo la sección del firewall superior. Si no se crean estas secciones de marcador, todas las reglas de aislamiento se crearán en la parte inferior, y todas las secciones de la directiva se crearán en la parte superior. No puede haber varias secciones del firewall de marcador con el mismo valor en cada clúster, ya que se producirá un error.

Instalación de NCP en un entorno de Kubernetes

3

Para instalar NSX Container Plug-in (NCP) es necesario instalar componentes en los nodos principal y de Kubernetes.

Este capítulo incluye los siguientes temas:

- [Instalar el complemento CNI de NSX-T Data Center](#)
- [Instalar y configurar OVS](#)
- [Configurar las redes de NSX-T Data Center para los nodos de Kubernetes](#)
- [Instalar el agente del nodo de NSX](#)
- [ConfigMap para ncp.ini in nsx-node-agent-ds.yml](#)
- [Instalar NSX Container Plug-in](#)
- [ConfigMap para ncp.ini en ncp-rc.yml](#)
- [Montar un certificado codificado en formato PEM y una clave privada en el pod de NCP](#)
- [Montar un archivo de certificado en el pod de NCP](#)
- [Configurar syslog](#)
- [Consideraciones de seguridad](#)
- [Consejos sobre la configuración de recursos de red](#)

Instalar el complemento CNI de NSX-T Data Center

El complemento CNI de NSX-T Data Center debe instalarse en los nodos de Kubernetes.

En Ubuntu, al instalar el complemento CNI de NSX-T, se copiará el archivo de perfil de AppArmor `ncp-apparmor` en `/etc/apparmor.d` y se cargará. Antes de la instalación, el servicio de AppArmor debe estar en ejecución y el directorio `/etc/apparmor.d` debe existir. De lo contrario, se producirá un error en la instalación. Puede comprobar si el módulo de AppArmor está habilitado con el siguiente comando:

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

Puede comprobar si el servicio de AppArmor se inició con el siguiente comando:

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

Si el servicio de AppArmor no se está ejecutando al instalar el complemento CNI de NSX-T, la instalación mostrará el siguiente mensaje cuando finalice:

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

El mensaje indica que todos los pasos de instalación se han completado, excepto la carga del perfil de AppArmor.

El archivo de perfil `ncp-apparmor` proporciona un perfil de AppArmor para el agente de nodo NSX llamado `node-agent-apparmor`, que se diferencia del perfil `docker-default` en los siguientes aspectos:

- Se elimina la regla `deny mount`.
- Se agrega la regla `mount`.
- Se agregan algunas opciones de `network`, `capability`, `file` y `umount`.

Puede reemplazar el perfil `node-agent-apparmor` con otro perfil. Sin embargo, en el archivo `nsx-nodo-agent-ds.yml`, que se utiliza en la instalación del agente de nodo NSX, se hace referencia al nombre de perfil `node-agent-apparmor`. Si utiliza otro perfil, deberá especificar el nombre del perfil en `nsx-nodo-agent-ds.yml`, en la sección `spec:template:metadata:annotations`, en la siguiente entrada:

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

Procedimiento

- 1 Descargue el archivo de instalación adecuado para la distribución de Linux.

El nombre de archivo es `1.x86_64.rpm-nsx-cni-1.0.0.0.0.xxxxxxx` o `nsx-cni-1.0.0.0.0.xxxxxxx.deb`, donde `xxxxxxx` es el número de compilación.

- 2 Instale el archivo `rpm` o `deb` que descargó en el paso 1.

El complemento se instala en `/opt/cni/bin`. El archivo de configuración de CNI `10-nsx.conf` se copia en `/etc/cni/net.d`. El `rpm` también instalará el archivo de configuración `/etc/cni/net.d/99-loopback.conf` del complemento de bucle invertido.

Instalar y configurar OVS

Instale y configure OVS (Open vSwitch) en los nodos secundarios.

Procedimiento

- 1 Descargue el archivo de instalación correspondiente a su distribución de Linux.

Los nombres de archivo son `openvswitch-common_2.10.x.xxxxxxx-1_amd64.deb`, `openvswitch-datapath-dkms_2.10.x.xxxxxxx-1_all.deb` y `openvswitch-switch_2.10.x.xxxxxxx-1_amd64.deb`, donde `xxxxxxx` es el número de compilación.

- 2 Instale el archivo deb que descargó en el paso 1.
- 3 En Ubuntu, ejecute el siguiente comando para volver a cargar el módulo kernel de OVS.

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

- 4 Compruebe que OVS se esté ejecutando.

```
# systemctl status openvswitch-switch.service
```

- 5 Cree la instancia *br-int* si aún no se creó.

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

- 6 Agregue la interfaz de red (*node-if*) que está conectada al conmutador lógico del nodo en *br-int*.

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

Ejecute el siguiente comando para comprobar qué valor tiene `ofport`, porque si `ofport 1` no está disponible, OVS asignará un puerto que sí lo esté.

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

Si `ofport` no es 1, configure, según corresponda, la opción `ovs_uplink_port` de la sección `nsx_kube_proxy` del archivo yaml DaemonSet del agente del nodo de NSX.

- 7 Asegúrese de que *br-int* y *node-if link* estén activos.

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

- 8 Actualice el archivo de configuración de red para garantizar que la interfaz de red esté activa después de un reinicio.

En Ubuntu, actualice `/etc/network/interfaces` y agregue las siguientes líneas:

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

En RHEL, actualice `/etc/etere/syconfig/red-scripts/ifcfg-<node-if>` y agregue la siguiente línea:

```
ONBOOT=yes
```

Configurar las redes de NSX-T Data Center para los nodos de Kubernetes

En esta sección se describe cómo configurar las redes de NSX-T Data Center para los nodos principales o de trabajo de Kubernetes.

Cada nodo debe tener al menos dos interfaces de red. La primera es una interfaz de administración que puede que esté o no en el tejido de NSX-T Data Center. Las otras interfaces proporcionan redes para los pods, se encuentran en el tejido de NSX-T Data Center y están conectadas a un conmutador lógico, que se conoce como el conmutador lógico del nodo. Las direcciones IP del pod y de administración se deben poder enrutar para que la comprobación de estado de Kubernetes funcione. En cuanto a la comunicación entre la interfaz de administración y los pods, NCP crea automáticamente una regla DFW para permitir la comprobación de estado y otro tráfico de administración. Puede ver la información de esta regla en la GUI de NSX Manager. Esta regla no se debe cambiar ni eliminar.

En cada máquina virtual del nodo, asegúrese de que la vNIC que se designó como red del contenedor esté asociada al conmutador lógico del nodo.

NSX Container Plug-in (NCP) debe conocer el ID de VIF de la vNIC usada como tráfico del contenedor en cada nodo. El puerto de conmutador lógico debe tener las dos siguientes etiquetas: En una etiqueta, especifique el nombre del nodo. En la otra etiqueta, especifique el nombre del clúster. Para especificar el ámbito, utilice los valores adecuados que se indican a continuación.

Etiqueta	Ámbito
Nombre del nodo	<code>ncp/node_name</code>
Nombre del clúster	<code>ncp/cluster</code>

Puede identificar el puerto de conmutador lógico para una máquina virtual del nodo si accede a **Inventario > Máquinas virtuales** desde la GUI de NSX Manager.

Si el nombre del nodo de Kubernetes cambia, debe actualizar la etiqueta `ncp/node_name` y reiniciar NCP. Puede usar el siguiente comando para obtener los nombres del nodo:

```
kubect1 get nodes
```

Si agrega un nodo a un clúster mientras NCP se está ejecutando, debe agregar las etiquetas al puerto de conmutador lógico antes de ejecutar el comando `kubeadm join`. De lo contrario, el nuevo nodo no tendrá conectividad a la red. Si faltan etiquetas o no son correctas, puede seguir estos pasos para resolver el problema:

- Aplique las etiquetas correctas al puerto de conmutador lógico.

- Reinicie NCP.

Instalar el agente del nodo de NSX

El agente del nodo de NSX es un DaemonSet en el que cada pod ejecuta dos contenedores. Uno de los contenedores ejecuta el agente del nodo de NSX, cuya mayor responsabilidad es administrar las interfaces de red de los contenedores. Interactúa con el complemento CNI y el servidor de la API de Kubernetes. El otro contenedor ejecuta kube-proxy de NSX, cuya única responsabilidad es implementar la abstracción del servicio de Kubernetes traduciendo las IP de los clústeres en IP de pods. Implementa la misma funcionalidad que el kube-proxy ascendente.

Procedimiento

- 1 Descargue la imagen Docker de NCP.

El nombre de archivo es `nsx-ncp-xxxxxxx.tar`, donde `xxxxxxx` es el número de compilación.

- 2 Descargue la plantilla yaml de DaemonSet del agente del nodo de NSX.

El nombre de archivo es `nsx-node-agent-ds.yaml`. Puede editar este archivo o usarlo como ejemplo para su propio archivo de plantilla.

- 3 Cargue la imagen Docker de NCP al registro de imágenes.

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

- 4 Edite `nsx-node-agent-ds.yaml`.

Cambie el nombre de la imagen a la que se cargó.

En Ubuntu, el archivo yaml asume que AppArmor está habilitado. Para ver si AppArmor está habilitado, consulte el archivo `/sys/module/apparmor/parameters/enabled`. Si AppArmor no está habilitado, realice los siguientes cambios:

- Elimine o agregue un comentario a la siguiente línea:

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

- Agregue la línea `privileged:true` en `securityContext` para el contenedor `nsx-node-agent` y el contenedor `nsx-kube-proxy`. Por ejemplo:

```
securityContext:
  privileged:true
```

Nota Si se ejecuta kubelet en un contenedor que usa la imagen `hyperkube`, kubelet siempre notifica que AppArmor está deshabilitado, independientemente del estado real. Este es un problema conocido. Debe realizar los mismos cambios en el archivo `yaml`.

Nota En el archivo `yaml`, es necesario especificar que el `ConfigMap` que se generó para `ncp.ini` debe estar montado como un volumen `ReadOnly`. El archivo descargado `yaml` ya tiene esta especificación y no se debe cambiar.

- 5 Cree el `DaemonSet` del agente del nodo de NSX con el siguiente comando.

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

ConfigMap para `ncp.ini` in `nsx-node-agent-ds.yml`

El archivo de `yaml` de ejemplo `nsx-node-agent-ds.yml` contiene un `ConfigMap` para el archivo de configuración `ncp.ini` del agente del nodo de NSX. Esta sección centrada en `ConfigMap` contiene parámetros que puede especificar para personalizar la instalación del agente del nodo.

El ejemplo `nsx-node-agent-ds.yml` que descargó tiene la siguiente información `ncp.ini`:

```
# 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>
```

Instalar NSX Container Plug-in

NSX Container Plug-in (NCP) se envía como una imagen de Docker. NCP debe ejecutarse en un nodo para los servicios de la infraestructura. No se recomienda ejecutar NCP en el nodo principal.

Procedimiento

- 1 Descargue la imagen Docker de NCP.

El nombre de archivo es `nsx-ncp-xxxxxxx.tar`, donde `xxxxxxx` es el número de compilación.

- 2 Descargue la plantilla yaml ReplicationController de NCP.

El nombre de archivo es `ncp-rc.yaml`. Puede editar este archivo o usarlo como ejemplo para su propio archivo de plantilla.

- 3 Cargue la imagen Docker de NCP al registro de imágenes.

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

- 4 (opcional) Descargue la plantilla yaml para la definición del recurso personalizado del objeto NSXError.

El nombre del archivo es `nsx-error-crd.yaml`.

- 5 (opcional) Cree el recurso personalizado.

```
kubectl create -f nsx-error-crd.yaml
```

- 6 Edite `ncp-rc.yaml`.

Cambie el nombre de la imagen a la que se cargó.

Especifique el parámetro `nsx_api_managers`. Puede especificar la dirección IP de un único NSX Manager, las direcciones IP (separadas por comas) de los tres NSX Managers en un clúster de NSX Manager, o la dirección IP virtual de un clúster de NSX Manager. Por ejemplo:

```
nsx_api_managers = 192.168.1.180
or
nsx_api_managers = 192.168.1.181,192.168.1.182,192.168.1.183
```

(Opcional) Especifique el parámetro `ca_file` en la sección `[nsx_v3]`. El valor debe ser un archivo de paquete de CA que se usa para verificar el certificado del servidor de NSX Manager. Si no se establece, se usarán las CA raíz del sistema. Si especifica una dirección IP para `nsx_api_managers`, especifique un archivo de CA. Si especifica tres direcciones IP para `nsx_api_managers`, puede especificar uno o tres archivos de CA. Si especifica un archivo de CA, se utilizará para los tres administradores. Si especifica tres archivos de CA, cada uno se utilizará para la dirección IP correspondiente en `nsx_api_managers`. Por ejemplo,

```
ca_file = ca_file_for_all_mgrs
or
ca_file = ca_file_for_mgr1,ca_file_for_mgr2,ca_file_for_mgr3
```

Especifique los parámetros `nsx_api_cert_file` y `nsx_api_private_key_file` para la autenticación con NSX-T Data Center.

`nsx_api_cert_file` es la ruta completa a un archivo de certificado cliente en formato PEM. El contenido de este archivo debe ser similar al siguiente:

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

`nsx_api_private_key_file` es la ruta completa a un archivo de clave privada cliente en formato PEM. El contenido de este archivo debe ser similar al siguiente:

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

Especifique el parámetro `ingress_mode = nat` si el controlador de entrada está configurado para ejecutarse en modo NAT.

De forma predeterminada, se utiliza el prefijo 24 para todas las subredes asignadas desde los bloques de IP para los conmutadores lógicos del pod. Para usar un tamaño de subred diferente, actualice la opción `subnet_prefix` en la sección `[nsx_v3]`.

HA (alta disponibilidad) está habilitada de forma predeterminada. Puede deshabilitar HA con la siguiente especificación:

```
[ha]
enable = False
```

(Opcional) Habilita la creación de informes de errores mediante NSXError en `ncp.ini`. Este ajuste está deshabilitado de forma predeterminada.

```
[nsx_v3]
enable_nsx_err_crd = True
```

Nota En el archivo yaml, debe especificar que el ConfigMap que se generó para `ncp.ini` esté montado como un volumen ReadOnly. El archivo descargado yaml ya tiene esta especificación y no se debe cambiar.

7 Cree ReplicationController de NCP.

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

Resultados

Nota NCP abre conexiones HTTP persistentes con el servidor de API de Kubernetes para inspeccionar los eventos de ciclo de vida de los recursos de Kubernetes. Si un error del servidor de API o un error de red provoca que las conexiones TCP de NCP queden obsoletas, debe reiniciar NCP para que se puedan restablecer las conexiones con el servidor de API. De lo contrario, NCP no detectará los nuevos eventos.

Durante una actualización gradual de ReplicationController de NCP, puede que aparezcan dos pods de NCP que se ejecutan tras la actualización gradual en las siguientes situaciones:

- Se reinicia el host de contenedor durante la actualización gradual.
- Inicialmente se produce un error en la actualización gradual debido a que la nueva imagen no existe en un nodo de Kubernetes. Se descarga la imagen, vuelve a ejecutarse la actualización gradual y esta se realiza correctamente.

Haga lo siguiente si se muestran dos pods de NCP en ejecución:

- Elimine uno de los pods de NCP. No importa cuál. Por ejemplo,

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

- Elimine ReplicationController de NCP. Por ejemplo,

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

ConfigMap para `ncp.ini` en `ncp-rc.yml`

El archivo YAML de ejemplo `ncp-rc.yml` contiene un ConfigMap para el archivo de configuración `ncp.ini`. Esta sección centrada en ConfigMap contiene parámetros que debe especificar antes de instalar NCP, como se describe en la sección anterior.

El ejemplo `ncp-rc.yml` que descargó tiene la siguiente información `ncp.ini`:

```
# 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>

# Name or UUID of the external ip pools that will be used for allocating IP
# addresses for exposing LoadBalancer type service and ingress
#external_ip_pools_lb = <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>

# Option to enabling error reporting through NSXError CRD
#enable_nsx_err_crd = False

# Option for user to define the maximum allowed virtual servers to be created
# for Service of type LoadBalancer in the cluster. The value should be an
# integer greater than zero.
# max_allowed_virtual_servers = <1000>
```

Montar un certificado codificado en formato PEM y una clave privada en el pod de NCP

Si tiene un certificado codificado en formato PEM y una clave privada, puede actualizar la definición del pod de NCP en el archivo yaml para montar los secretos TLS en dicho pod.

- 1 Cree un secreto TLS para el certificado y la clave privada.

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

- 2 Actualice el yaml de especificación del pod de NCP para montar el secreto como archivos en la especificación de dicho pod.

```
spec:
  ...
  containers:
  - name: nsx-ncp
    ...
    volumeMounts:
    ...
    - name: nsx-cert
      mountPath: /etc/nsx-uj0/nsx-cert
      readOnly: true
  volumes:
  ...
  - name: nsx-cert
    secret:
      secretName: SECRET_NAME
```

- 3 Actualice las opciones `nsx_api_cert_file` y `nsx_api_private_key_file` de `nsx_v3` en el archivo yaml.

```
nsx_api_cert_file = /etc/nsx-uj0/nsx-cert/tls.crt
nsx_api_private_key_file = /etc/nsx-uj0/nsx-cert/tls.key
```

Montar un archivo de certificado en el pod de NCP

Si tiene un archivo de certificado en el sistema de archivos del nodo, puede actualizar la especificación del pod de NCP para montar el archivo en dicho pod.

Por ejemplo,

```
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>
```

Configurar syslog

Puede ejecutar un agente syslog, como rsyslog o syslog-ng, en un contenedor para enviar registros desde NCP y los componentes a un servidor syslog.

Se recomiendan los siguientes métodos. Para obtener más información sobre cómo realizar registros en Kubernetes, consulte <https://kubernetes.io/docs/concepts/cluster-administration/logging>.

- Cree un contenedor sidecar que se ejecute en NCP o en el pod nsx-node-agent.
- Ejecute una réplica de DaemonSet en cada nodo.

Nota Con el método del contenedor sidecar, los registros del complemento CNI de NSX no se pueden enviar al servidor syslog porque el complemento no se ejecuta en un pod.

Crear un contenedor sidecar para syslog

Puede configurar un contenedor sidecar para syslog si desea ejecutarlo en el mismo pod que NCP. El siguiente procedimiento asume que la imagen del agente syslog es ejemplo/rsyslog.

Procedimiento

- 1 Configure el agente del nodo de NSX y de NCP para registrarlo en un archivo.

En el archivo yaml para el agente del nodo de NSX y de NCP, establezca el parámetro `log_dir` y especifique el volumen que se montará. Por ejemplo,

```
[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
```

Puede cambiar el nombre del archivo de registro estableciendo el parámetro `log_file`. Los nombres predeterminados son `ncp.log`, `nsx_node_agent.log` y `nsx_kube_proxy.log`. Si la opción `log_dir` se establece en una ruta de acceso que no sea `/var/log/nsx-ujo`, se debe crear un volumen `hostPath` o `emptyDir` y se debe montar en la especificación del pod correspondiente.

- 2 Asegúrese de que la ruta de acceso de host existe y que el usuario `nsx-ncp` puede escribir en ella.
 - a Ejecute los siguientes comandos.

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

- b Agregue el usuario `nsx-ncp` o cambie el modo de la ruta de acceso de host a 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 En el archivo yaml de la especificación del pod de NCP, agregue ConfigMap para syslog. Por ejemplo,

```
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 En el archivo yaml del pod de NCP, agregue el contenedor rsyslog y monte los volúmenes apropiados donde rsyslog pueda encontrar los datos de configuración y leer los registros de otros contenedores. Por ejemplo,

```
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>
```

Crear una réplica de DaemonSet para Syslog

Los registros de todos los componentes de NCP se pueden redireccionar con este método. Es necesario que se configuren las aplicaciones de forma que se registren en stderr, que está habilitado de forma predeterminada. El siguiente procedimiento asume que la imagen del agente syslog es ejemplo/rsyslog.

Procedimiento

- 1 Cree el archivo de yaml de DaemonSet. Por ejemplo,

```

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 Cree DaemonSet.

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

Ejemplo: Configuración de la rotación de registros y la ejecución de Syslog en un contenedor sidecar

El siguiente procedimiento muestra cómo configurar la rotación de registros y la ejecución de Syslog en un contenedor sidecar.

Creación del directorio de registros y configuración de la rotación de registros

- Cree el directorio de registros en todos los nodos, incluido el principal, y cambie su propietario al usuario que tenga el identificador 1000.

```

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

```

- Configure la rotación de registros en todos los nodos del directorio /var/log/nsx-ujo.

```

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

```

```

        notifempty
        missingok
    }
EOF

```

Creación de la controladora de replicación de NCP

- Cree el archivo `ncp.ini` para NCP.

```

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

```

- Cree el mapa de configuración del archivo INI.

```

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

```

- Cree la configuración de rsyslog de NCP.

```

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

```

- Cree el mapa de configuración del elemento anterior.

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

- Cree la controladora de replicación de NCP con el rsyslog sidecar.

```

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

```

- Cree NCP con la especificación anterior.

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

Creación de DaemonSet del agente de nodo de NSX

- Cree la configuración de rsyslog para los agentes de nodo.

```

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
```

- Cree el configmap a partir de los anteriores.

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

- Cree DaemonSet con el configmap sidecar.

```
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

```

- Cree DaemonSet.

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

Consideraciones de seguridad

Cuando implemente NCP, es importante que realice los pasos pertinentes para asegurar los entornos de Kubernetes y de NSX-T Data Center.

Restringir NCP para que solo se ejecute en nodos designados

NCP tiene acceso al plano de administración de NSX-T Data Center y se debe limitar para que solo se ejecute en nodos de infraestructura designados. Puede identificar estos nodos con una etiqueta apropiada. Se debe aplicar un nodeSelector de esta etiqueta a la especificación ReplicationController de NCP. Por ejemplo,

```

nodeSelector:
  nsx-infra: True

```

También puede usar otros mecanismos, como la afinidad, para asignar pods a nodos. Para obtener más información, consulte <https://kubernetes.io/docs/concepts/configuration/assign-pod-node>.

Asegurarse de que Docker Engine esté actualizado

Docker publica periódicamente actualizaciones de seguridad. Se debe implementar un procedimiento automatizado para aplicar esas actualizaciones.

No permitir las funciones NET_ADMIN y NET_RAW de contenedores sin confianza

Los atacantes pueden aprovechar las funciones NET_ADMIN y NET_RAW de Linux para poner en peligro la red de pods. Debe deshabilitar estas dos funciones de contenedores en los que no confía. De forma predeterminada, la función NET_ADMIN no se otorga a un contenedor sin privilegios. Tenga en cuenta que una especificación de pod puede habilitarla o establecer el contenedor en modo privilegiado. Además, en los contenedores que no son de confianza, deshabilite NET_RAW especificando este valor en la lista de funciones descartadas en la configuración SecurityContext de la especificación del contenedor. Por ejemplo,

```

securityContext:
  capabilities:
    drop:

```

```

- NET_RAW
- ...

```

Control de acceso basado en funciones

Kubernetes usa la API del Control de acceso basado en funciones (Role-Based Access Control, RBAC) para tomar decisiones de autorización, permitiendo que los administradores configuren directivas. Para obtener más información, consulte <https://kubernetes.io/docs/admin/authorization/rbac>.

Por lo general, el administrador de clústeres es el único usuario con funciones y acceso privilegiados. Para las cuentas del servicio y de usuarios, se debe seguir el principio de mínimo privilegio al otorgar acceso.

Se recomienda seguir las siguientes instrucciones:

- Restringir el acceso a los tokens de la API de Kubernetes de los pods que los necesiten.
- Restringir el acceso a ConfigMap de NCP y a los secretos TLS del certificado cliente de NSX API al pod de NCP.
- Bloquear el acceso a la API de redes de Kubernetes desde pods que no requieran dicho acceso.
- Agregar una directiva RBAC de Kubernetes para especificar los pods que tienen acceso a la API de Kubernetes.

Directiva de RBAC recomendada para el pod de NCP

Cree el pod de NCP en un ServiceAccount y otorgue un conjunto mínimo de privilegios a esta cuenta. Además, no permita que otros pods ni ReplicationControllers accedan a ConfigMap ni a los secretos TLS que se montan como volúmenes para ReplicationController de NCP y el agente del nodo de NSX.

El siguiente ejemplo muestra cómo se especifican funciones y enlaces de funciones de NCP:

```

# 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
```

Nota Todos los pods que tengan acceso al servidor API de Kubernetes deben tener acceso al secreto TLS que se crea usando la API de Kubernetes para el certificado cliente de NSX-T Data Center y el par de claves privadas. De forma similar, cuando un pod se crea sin cuenta del servicio, se asigna automáticamente la predeterminada en el mismo espacio de nombres que monta automáticamente el token para acceder a la API de Kubernetes. Por ello, el acceso a los tokens debe estar limitado únicamente a los pods que los necesitan.

Consejos sobre la configuración de recursos de red

Al configurar algunos recursos de red, se deben tener en cuenta ciertas restricciones.

Límites de etiquetado de NSX-T Data Center

NSX-T Data Center presenta las siguientes limitaciones al etiquetar un objeto:

- El ámbito tiene un límite de 128 caracteres.
- La etiqueta tiene un límite de 256 caracteres.
- Cada objeto puede tener hasta 30 etiquetas.

Estos límites pueden ocasionar problemas cuando se copian anotaciones de Kubernetes u OpenShift en los ámbitos y las etiquetas de NSX-T Data Center y se superan los límites. Por ejemplo, si una etiqueta de un puerto de conmutador se utiliza en una regla de firewall, es posible que la regla no se aplique según lo previsto debido a que la clave o el valor de anotación se hayan truncado al copiarlos en un ámbito o una etiqueta.

Configuración de las directivas de red

Las directivas de red seleccionan pods o espacios de nombres que emplean selectores de etiquetas.

La compatibilidad de NCP para las directivas de red es la misma que la compatibilidad que proporciona Kubernetes y depende de la versión de Kubernetes.

- En Kubernetes 1.11, puede especificar los siguientes selectores de regla:
 - `podSelector`: selecciona todos los pods que se encuentran en el espacio de nombres en el que se creó la directiva de red.
 - `namespaceSelector`: selecciona todos los espacios de nombres.
 - `podSelector` y `namespaceSelector`: selecciona todos los pods que se encuentran en los espacios de nombres seleccionados por `namespaceSelector`.
 - `ipBlockSelector`: una directiva de red no es válida si `ipBlockSelector` se combina con `namespaceSelector` o `podSelector`. Una instancia de `ipBlockSelector` debe estar sola en la especificación de directiva.
- En Kubernetes 1.10, las cláusulas de reglas en la directiva de red pueden contener como máximo un selector de `namespaceSelector`, `podSelector` y `ipBlock`.

El servidor de la API de Kubernetes no realiza una validación de una especificación de directiva de red. Es posible crear una directiva de red que no sea válida. NCP rechazará dicha directiva de red. Aunque actualice la directiva de red para que sea válida, NCP no la procesará. Debe eliminar la directiva de red y volver a crear una con una especificación válida.

Instalación de NCP en un entorno de Pivotal Cloud Foundry

4

Pivotal Cloud Foundry (PCF) es un proveedor de plataforma como servicio (PaaS) de código abierto. Puede instalar NSX Container Plug-in (NCP) en un entorno de PCF para proporcionar servicios de redes.

Las máquinas virtuales creadas mediante Pivotal Ops Manager deben tener conectividad de capa 3 con la red del contenedor para acceder a las funciones de NSX-T.

La alta disponibilidad (HA) se habilita automáticamente.

Nota Cuando se efectúa un cambio en un grupo de seguridad, se deben volver a realizar copias intermedias de todas las aplicaciones a las que se aplica el grupo de seguridad. Esto puede ocurrir debido a que el grupo de seguridad se aplica al espacio en el que se ejecutan las aplicaciones, o porque el grupo de seguridad es global.

Este capítulo incluye los siguientes temas:

- [Instalación de NCP en un entorno de Pivotal Cloud Foundry](#)

Instalación de NCP en un entorno de Pivotal Cloud Foundry

NCP se instala mediante la interfaz gráfica de usuario de Pivotal Ops Manager.

Requisitos previos

Una instalación nueva de Pivotal Ops Manager, NSX-T Data Center y Pivotal Application Service (PAS). Asegúrese de que Ops Manager se instale primero, luego NSX-T Data Center y, por último, PAS. Para obtener más información, consulte la documentación de Pivotal Cloud Foundry.

Procedimiento

- 1 Descargue el archivo de instalación de NCP para PCF.
El nombre de archivo es `VMware-NSX-T.<versión>.<compilación>.pivotal`.
- 2 Inicie sesión en Pivotal Ops Manager como administrador.
- 3 Haga clic en **Importar un producto**.
- 4 Seleccione el archivo que se descargue.

- 5 Haga clic en el mosaico **Ops Manager Director for VMware vSphere**.
- 6 En la pestaña **Configuración** de **Configuración de vCenter**, seleccione **Redes NSX** y, en **Modo de NSX**, seleccione **NSX-T**.
- 7 En el campo **Certificado de CA de NSX**, proporcione el certificado en formato PEM.
- 8 Haga clic en **Guardar**.
- 9 Haga clic en el **Panel de control de instalación** en la esquina superior izquierda para volver al panel de control.
- 10 Haga clic en el mosaico **Pivotal Application Service**.
- 11 En la pestaña **Configuración**, seleccione **Redes** en el panel de navegación.
- 12 En el **complemento de la interfaz de red del contenedor**, seleccione **Externo**.
- 13 Haga clic en el **Panel de control de instalación** en la esquina superior izquierda para volver al panel de control.
- 14 Haga clic en **Guardar**.
- 15 Haga clic en el **Panel de control de instalación** en la esquina superior izquierda para volver al panel de control.
- 16 Haga clic en el mosaico **VMware NSX-T**.
- 17 Introduzca la dirección de NSX Manager.
- 18 Seleccione el método de autenticación de NSX Manager.

Opción	Acción
Autenticación de certificado de cliente	Proporcione el certificado y la clave privada para NSX Manager.
Autenticación básica con nombre de usuario y contraseña	Proporcione el nombre de usuario y la contraseña del administrador de NSX Manager.

- 19 En el campo **Certificado de CA de NSX Manager**, proporcione el certificado.
- 20 Haga clic en **Guardar**.
- 21 Seleccione **NCP** en el panel de navegación.
- 22 Introduzca el **nombre de la fundación PAS**.

Esta cadena identifica una fundación PAS de forma exclusiva en la API de NSX. Esta cadena también se utiliza como prefijo en los nombres de los recursos de NSX creados por NCP para la fundación PAS.
- 23 Indique el valor de **Zona de transporte superpuesta**.
- 24 Introduzca el valor de **Enrutador de nivel 0**.

25 Especifique uno o varios **bloques de IP de redes del contenedor**.

- a Haga clic en **Agregar**.
- b Introduzca el **nombre del bloque de IP**. Puede ser un bloque de IP nuevo o existente.
- c Solo para un nuevo bloque de IP, especifique el bloque en formato CIDR; por ejemplo, 10.1.0.0/16.

26 Especifique el prefijo de subred de las redes del contenedor.

27 Haga clic en **Habilitar SNAT para redes del contenedor** para habilitar SNAT.

28 Especifique uno o varios **grupos de IP que se utilizan para proporcionar una dirección IP externa (NAT) a redes de organización**.

- a Haga clic en **Agregar**.
- b Introduzca el **nombre del grupo de IP**. Puede ser un grupo de IP nuevo o existente.
- c Solo para un nuevo grupo de IP, especifique las direcciones IP mediante la entrega de CIDR y los rangos de IP.

29 (opcional) Introduzca el **marcador de la sección del firewall superior**.

30 (opcional) Introduzca el **marcador de la sección del firewall inferior**.

31 (opcional) Habilite o deshabilite las siguientes opciones.

Opción	Valor predeterminado
Registrar tráfico perdido de la aplicación	Deshabilitado. Si se habilita, se registrará el tráfico que se pierda debido a una regla de firewall.
Habilitar el nivel de depuración para el registro de NCP	Habilitado.

32 Haga clic en **Guardar**.

33 (opcional) Seleccione **Agente de nodo de NSX** en el panel de navegación.

- a Marque la casilla **Habilitar el nivel de depuración de registro para el agente de nodo de NSX** para habilitar el registro a nivel de depuración.
- b Haga clic en **Guardar**.

34 Haga clic en el **Panel de control de instalación** en la esquina superior izquierda para volver al panel de control.

35 Haga clic en **Aplicar cambios**.

Equilibrio de carga

5

El equilibrador de carga de NSX-T Data Center está integrado en Kubernetes.

Este capítulo incluye los siguientes temas:

- [Configuración del equilibrio de carga](#)
- [Controladores de entrada de terceros](#)

Configuración del equilibrio de carga

La configuración del equilibrio de carga implica la configuración de un servicio de equilibrador de carga de Kubernetes o un recurso de entrada y la controladora de replicación de NCP.

Puede crear un equilibrador de carga de capa 4 mediante la configuración de un servicio de Kubernetes de tipo equilibrador de carga, así como un equilibrador de carga de capa 7 mediante la configuración de un recurso de entrada de Kubernetes.

Para configurar el equilibrio de carga en NCP, en el archivo `ncp-rc.yml`:

- 1 Establezca `use_native_loadbalancer` como **True**.
- 2 (Opcional) Establezca `pool_algorithm` como **ROUND_ROBIN** o **LEAST_CONNECTION/IP_HASH**. El valor predeterminado es **ROUND_ROBIN**.
- 3 (Opcional) Establezca `service_size` como **SMALL**, **MEDIUM** o **LARGE**. El valor predeterminado es **SMALL**.

El algoritmo **LEAST_CONNECTION/IP_HASH** indica que el tráfico de la misma dirección IP de origen se enviará al mismo pod de back-end.

Para obtener detalles sobre lo que admiten los equilibradores de carga de NSX-T de diferentes tamaños, consulte la *Guía de administración de NSX-T Data Center*.

Después de que se crea el equilibrador de carga, no es posible cambiar su tamaño actualizando el archivo de configuración. Se puede cambiar mediante la interfaz de usuario o la API de NSX Manager.

Configuración de persistencia para el equilibrio de carga de capa 4 y capa 7

Puede especificar una configuración de persistencia con los parámetros `l4_persistence` y `l7_persistence` en ConfigMap de NCP. La opción disponible para la persistencia de capa 4 es la IP de origen. Las opciones disponibles para la persistencia de capa 7 son IP de origen y cookie. El valor predeterminado es `<None>`. Por ejemplo,

```
# 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
```

Para los equilibradores de carga de Kubernetes, también puede establecer `sessionAffinity` en la especificación del servicio para configurar su comportamiento de persistencia si la persistencia global de capa 4 está desactivada (es decir, si `l4_persistence` se ha establecido como `<None>`). Si se ha establecido `l4_persistence` como `source_ip`, se puede utilizar `sessionAffinity` en la especificación de servicio para personalizar el tiempo de espera de la persistencia del servicio. El tiempo de espera predeterminado de la persistencia de capa 4 es de 10800 segundos (como se especifica en la documentación de Kubernetes para servicios (<https://kubernetes.io/docs/>

[concepts/services-networking/service](#)). Los servicios con un tiempo de espera de persistencia predeterminado compartirán el mismo perfil de persistencia del equilibrador de carga de NSX-T. Se creará un perfil dedicado para cada servicio con un tiempo de espera de persistencia no predeterminado.

Nota Si el servicio back-end de una entrada es un servicio de tipo equilibrador de carga, el servidor virtual de capa 4 para el servicio y el servidor virtual de capa 7 para la entrada no pueden tener configuraciones de persistencia diferentes (por ejemplo, `source_ip` para la capa 4 y `cookie` para la capa 7). En tal caso, la configuración de persistencia para ambos servidores virtuales debe ser la misma (`source_ip`, `cookie` o `None`) o una de ellas debe ser `None` (y la otra configuración puede ser `source_ip` o `cookie`). A continuación, se muestra un ejemplo de este escenario:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80

-----
apiVersion: v1
kind: Service
metadata:
  name: tea-svc <==== same as the Ingress backend above
  labels:
    app: tea
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
    name: tcp
  selector:
    app: tea
  type: LoadBalancer
```

Entrada

- NSX-T Data Center creará un equilibrador de carga de capa 7 para las entradas con especificación TLS y otro equilibrador de carga de capa 7 para las entradas sin especificación TLS.
- Todas las entradas obtendrán una sola dirección IP.

- Al recurso de entrada se le asigna una dirección IP del grupo de direcciones IP externas que la opción `external_ip_pools` especifica en la sección `[nsx_v3]` de `ncp.ini`. El equilibrador de carga se expone en esta dirección IP y los puertos HTTP y HTTPS (80 y 443).
- Al recurso de entrada se le asigna una dirección IP del grupo de direcciones IP externas que la opción `external_ip_pools_lb` especifica en la sección `[nsx_v3]` de `ncp.ini`. Si la opción `external_ip_pools_lb` no existe, se utiliza el grupo que `external_ip_pools` especifica. El equilibrador de carga se expone en esta dirección IP y los puertos HTTP y HTTPS (80 y 443).
- Puede utilizar otro grupo de direcciones IP cambiando la configuración y reiniciando NCP.
- Puede especificar un certificado predeterminado para TLS. A continuación podrá obtener información sobre la generación de un certificado y el montaje de este en el pod de NCP.
- Las entradas sin especificación TLS se alojarán en el servidor virtual HTTP (puerto 80).
- Las entradas con especificación TLS se alojarán en el servidor virtual HTTPS (puerto 443). El equilibrador de carga actuará como servidor SSL y finalizará la conexión SSL de cliente.
- El orden de creación de los secretos y las entradas es indiferente. Si existe el objeto secreto y hay una entrada que hace referencia a este, el certificado se importará a NSX-T Data Center. Si se eliminan el secreto o la última entrada que hace referencia a este, se eliminará el certificado correspondiente a dicho secreto.
- Se admite la modificación de una entrada agregando o quitando la sección TLS. Cuando la clave `tls` se quita de la especificación de entrada, las reglas de entrada se transferirán del servidor virtual HTTPS (puerto 443) al servidor virtual HTTP (puerto 80). De igual modo, cuando la clave `tls` se agrega a la especificación de entrada, las reglas de entrada se transferirán del servidor virtual HTTP (puerto 80) al servidor virtual HTTPS (puerto 443).
- Si hay reglas duplicadas en las definiciones de entrada de un solo clúster, solo se aplicará la primera regla.
- Solo se admite una entrada con un back-end predeterminado en cada clúster. El tráfico que no coincida con ninguna regla de entrada se reenviará al back-end predeterminado.
- Si hay varias entradas con back-end predeterminados, solo se configurará la primera. Las demás se anotarán con un error.
- La coincidencia de URI de comodines se admite mediante los caracteres de expresión regular `"."` y `"*"`. Por ejemplo, la ruta de acceso `"/coffee/*"` coincide con `"/coffee/"` seguida de uno o varios caracteres (o bien de ninguno), como `"/coffee/"`, `"/coffee/a"` o `"/coffee/b"`, pero no con `"/coffee"`, `"/coffeecup"` ni `"/coffeecup/a"`.

Un ejemplo de especificación de entrada:

```
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  rules:
  - http:
      paths:
```

```
- path: /coffee/.*    #Matches /coffee/, /coffee/a but NOT /coffee, /coffeecup, etc.
  backend:
    serviceName: coffee-svc
    servicePort: 80
```

- Puede configurar la reescritura de solicitud de la URL agregando una anotación en el recurso de entrada. Por ejemplo,

```
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
```

Las rutas de acceso /tea y /coffee se reescribirán como / antes de enviar la URL al servicio back-end.

- Se admite la anotación de entrada `kubernetes.io/ingress.allow-http`.
 - Si la anotación se establece como **false**, solo se crearán reglas HTTPS.
 - Si falta la anotación o esta se establece como **true**, se crearán reglas HTTP. Asimismo, se crearán reglas HTTPS si la sección TLS está presente en la especificación de entrada.
- Los errores se anotan en el recurso de entrada. La clave de error es `ncp/error.loadbalancer` y la clave de advertencia es `ncp/warning.loadbalancer`. El error y la advertencia posibles son:
 - `ncp/error.loadbalancer: DEFAULT_BACKEND_IN_USE`
 Este error indica que ya existe una entrada con un back-end predeterminado. La entrada estará inactiva. Solo puede haber un back-end predeterminado para un grupo de entradas con y sin TLS. Para solucionar el error, elimine y vuelva a crear la entrada con una especificación correcta.
 - `ncp/warning.loadbalancer: SECRET_NOT_FOUND`
 Este error indica que el secreto especificado en la especificación de entrada no existe. La entrada se activará parcialmente. Para solucionar el error, cree el secreto que falta. Tenga en cuenta que las advertencias incluidas en la anotación no se borrarán durante el ciclo de vida del recurso de entrada.

- `ncp/warning.loadbalancer: INVALID_INGRESS`

Este error indica que se cumple una de las siguientes condiciones. La entrada estará inactiva. Para solucionar el error, elimine y vuelva a crear la entrada con una especificación correcta.

- Una regla de entrada entra en conflicto con otra regla de entrada en el mismo clúster de Kubernetes.
- La anotación `allow-http` se establece como **False** y la entrada no tiene una sección TLS.
- Una regla de entrada no tiene `host` ni `path` especificados. Una regla de entrada de este tipo tiene la misma funcionalidad que el back-end predeterminado de entrada. Utilice el back-end predeterminado de entrada en su lugar.

Servicio de tipo equilibrador de carga

- NSX-T Data Center creará un grupo y un servidor virtual de equilibrador de carga de capa 4 por cada puerto de servicio.
- Se admiten los protocolos TCP y UDP.
- Cada servicio tendrá una dirección IP exclusiva.
- Al servicio se le asigna una dirección IP de un grupo de direcciones IP externas según el campo `loadBalancerIP` de la definición del equilibrador de carga. El campo `loadBalancerIP` puede estar vacío, tener una dirección IP o el nombre o identificador de un grupo de direcciones IP. Si el campo `loadBalancerIP` está vacío, se asignará la dirección IP del grupo de direcciones IP externas especificada en la opción `external_ip_pools_lb` de la sección `[nsx_v3]` de `ncp.ini`. Si la opción `external_ip_pools_lb` no existe, se utiliza el grupo que `external_ip_pools` especifica. El servicio de equilibrador de carga se expone en esta dirección IP y en el puerto del servicio.
- Puede utilizar otro grupo de direcciones IP cambiando la configuración y reiniciando NCP.
- El grupo de direcciones IP especificado por `loadBalancerIP` debe tener la etiqueta `scope: ncp/owner`, `tag: cluster:<cluster_name>`.
- Se anota un error en un servicio. La clave de error es `ncp/error.loadbalancer`. Los posibles errores son:
 - `ncp/error.loadbalancer: IP_POOL_NOT_FOUND`

Este error indica que especifica `loadBalancerIP: <nsx-ip-pool>`, pero `<nsx-ip-pool>` no existe. El servicio estará inactivo. Para solucionar el error, especifique un grupo de direcciones IP válido, y elimine y vuelva a crear el servicio.
 - `ncp/error.loadbalancer: IP_POOL_EXHAUSTED`

Este error indica que especifica `loadBalancerIP: <nsx-ip-pool>`, pero el grupo de direcciones IP ya agotó las direcciones IP. El servicio estará inactivo. Para solucionar el error, especifique un grupo de direcciones IP que tenga direcciones IP disponibles, y elimine y vuelva a crear el servicio.

- `ncp/error.loadbalancer: IP_POOL_NOT_UNIQUE`

Este error indica que varios grupos de direcciones IP tienen el nombre que `loadBalancerIP` especifica: `<nsx-ip-pool>`. El servicio estará inactivo.

- `ncp/error.loadbalancer: POOL_ACCESS_DENIED`

Este error indica que el grupo de direcciones IP que `loadBalancerIP` especifica no tiene la etiqueta `scope: ncp/owner`, `tag: cluster:<cluster_name>` o que el clúster especificado en la etiqueta no coincide con el nombre del clúster de Kubernetes.

- `ncp/error.loadbalancer: LB_VIP_CONFLICT`

Este error indica que la dirección IP especificada en el campo `loadBalancerIP` es la misma que la dirección IP de un servicio activo. El servicio estará inactivo.

- Se admite el ajuste de escala automático del equilibrador de carga de capa 4. Si se crea o se modifica un servicio de equilibrador de carga de Kubernetes de manera que requiera servidores virtuales adicionales y el equilibrador de carga de capa 4 existente no tiene la capacidad, se creará un nuevo equilibrador de carga de capa 4. NCP también eliminará un equilibrador de carga de capa 4 que ya no tenga servidores virtuales asociados. Esta función está habilitada de forma predeterminada. Se puede deshabilitar estableciendo `l4_lb_auto_scaling` como **false** en ConfigMap de NCP.

Equilibrador de carga y directiva de red

Cuando el tráfico se reenvía a los pods desde el servidor virtual del equilibrador de carga de NSX, la IP de origen es la dirección IP del puerto de enlace ascendente del enrutador de nivel 1. Esta dirección se encuentra en la red de tránsito de nivel 1 privada y puede hacer que las directivas de red basadas en CIDR impidan el tráfico que se debería permitir. Para evitar este problema, la directiva de red debe configurarse de manera que la dirección IP del puerto de enlace ascendente del enrutador de nivel 1 sea parte del bloque de CIDR permitido. Esta dirección IP interna estará visible en el campo `status.loadbalancer.ingress.ip` y como una anotación (`ncp/internal_ip_for_policy`) en el recurso de entrada.

Por ejemplo, si la dirección IP externa del servidor virtual es 4.4.0.5 y la dirección IP del puerto de enlace ascendente del enrutador de nivel 1 interno es 100.64.224.11, el estado será:

```
status:
  loadBalancer:
    ingress:
      - ip: 4.4.0.5
      - ip: 100.64.224.11
```

La anotación de la entrada y el servicio del tipo de recurso del equilibrador de carga será:

```
ncp/internal_ip_for_policy: 100.64.224.11
```

La dirección IP 100.64.224.11 debe pertenecer al CIDR permitido en el selector de ipBlock de la directiva de red. Por ejemplo,

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

Script de ejemplo para generar un certificado firmado por CA

Con el siguiente script se genera un certificado firmado por CA y una clave privada que se almacenan en los archivos <nombredearchivo>.crt y <nombredearchivo>.key respectivamente. El comando `genrsa` genera una clave de CA. La clave de CA debe estar cifrada. Se puede especificar un método de cifrado con el comando (p. ej., `aes256`).

```
#!/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
```

Montar la clave y el certificado predeterminados en el pod de NCP

Tras generar la clave privada y el certificado, colóquelos en el directorio `/etc/nsx-ujo` en la máquina virtual del host. Suponiendo que los archivos de clave y de certificado se llaman `lb-default.crt` y `lb-default.key` respectivamente, edite `ncp-rc.yaml` para que estos archivos en el host se monten en el pod. Por ejemplo,

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

```

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

```

Controladores de entrada de terceros

Puede configurar NCP para que admita controladores de entrada de terceros.

Editar el archivo `ncp.ini`

Debe editar el archivo de configuración `/var/vcap/data/jobs/ncp/xxxxxxx/config/ncp.ini` (xxxxxxx es el identificador de implementación de BOSH). Este archivo se copiará en rootfs y NCP lo usará cada vez que se reinicie. El archivo se debe editar en cada nodo principal.

Importante Los cambios en `ncp.ini` no se mantienen en las actualizaciones del clúster de PKS. Si realiza cambios a través del mosaico de PKS y, a continuación, actualiza la implementación de PKS, se perderán los cambios en `ncp.ini`.

Las opciones correspondientes se encuentran en la sección `nsx_v3`.

- `use_native_loadbalancer` - Si se establece como **False**, NCP no procesará ninguna entrada ni servicio del tipo actualizaciones de LoadBalancer, independientemente de sus anotaciones. Este ajuste se aplica a todo el clúster de PKS. El valor predeterminado es **True**.
- `default_ingress_class_nsx` - Si se establece como **True**, NCP se convertirá en el controlador de entrada predeterminado y controlará tanto las entradas anotadas con `kubernetes.io/ingress.class: "nsx"` como las entradas sin ninguna anotación. Si se establece como **False**, NCP solo controlará las entradas anotadas con `kubernetes.io/ingress.class: "nsx"`. El valor predeterminado es **True**.

Si desea que NCP asigne una dirección IP flotante al pod del controlador NGINX y actualice el estado de las entradas con dicha IP flotante, haga lo siguiente:

- En la sección `k8s` de `ncp.ini`, establezca el valor `ingress_mode=nat`.
- Agregue la anotación `ncp/ingress-controller: "True"` al pod del controlador de entrada NGINX.

NCP actualizará el estado de las entradas que incluyan la anotación `kubernetes.io/ingress.class: "nginx"` con la IP flotante del pod del controlador de entrada NGINX. Si `default_ingress_class_nsx=False`, NCP también actualizará el estado de los ingresos sin la anotación `kubernetes.io/ingress.class` con la IP flotante del pod del controlador de entrada NGINX.

Nota: Aunque el pod del controlador de entrada NGINX no incluya la anotación `ncp/ingress-controller: "True"`, NCP actualizará el estado de las entradas mencionadas anteriormente a `loadBalancer: {}`. A continuación, las entradas pueden quedar bloqueadas en un bucle en el que el controlador NGINX actualiza el estado de las entradas a `loadBalancer: {ingress: [{ip: <IP>}]}` y NCP lo actualiza a `loadBalancer: {}`. Para resolver este problema, siga estos pasos:

- Si el controlador de entrada es de <https://github.com/kubernetes/ingress-nginx>:
 - En el controlador de entrada, cambie el valor de `ingress-class` por uno que no sea `"nginx"`.
 - Si hay una entrada con la anotación `kubernetes.io/ingress-class: "nginx"`, cámbiela a un valor diferente.
 - Para obtener más información, consulte <https://kubernetes.github.io/ingress-nginx/user-guide/multiple-ingress>.
- Si el controlador de entrada es de <https://github.com/nginxinc/kubernetes-ingress>:
 - En el controlador de entrada, cambie el valor de `ingress-class` por uno que no sea `"nginx"`.
 - Si hay una entrada con la anotación `kubernetes.io/ingress-class: "nginx"`, cámbiela a un valor diferente.
 - En el pod del controlador de entrada, establezca `use-ingress-class-only` como **True**. Esto evitará que este controlador actualice las entradas sin la anotación `kubernetes.io/ingress-class`.
 - Para obtener más información, consulte <https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/multiple-ingress.md>.

Escenario 1: NCP controla las entradas, pero no es el controlador de entrada predeterminado.

Siga este procedimiento para permitir que NCP controle las entradas de clase `nsx`.

- 1 Edite la sección `nsx_v3` del archivo `ncp.ini` de cada nodo principal.
 - a Establezca `default_ingress_class_nsx` como **False**.
 - b Deje `use_native_loadbalancer` establecido como **True**, el valor predeterminado.
- 2 Reinicie NCP en cada nodo principal. Esto puede provocar una conmutación por error generalizada.

- 3 Anote todas las entradas que desea que controle NCP con `kubernetes.io/ingress.class: "nsx"`.

Escenario 2: NCP es el controlador de entrada predeterminado.

Siga este procedimiento:

- 1 No es necesario editar `ncp.ini`, pero asegúrese de que todas las entradas estén anotadas.
- 2 Las entradas que gestionará NCP deben estar anotadas con **`kubernetes.io/ingress.class: "nsx"`**.

Aunque NCP controlará las entradas sin la anotación `kubernetes.io/ingress.class`, en el caso de que haya varios controladores de entrada, se recomienda tener siempre la anotación `kubernetes.io/ingress.class` para no depender del comportamiento del controlador de entrada predeterminado.

- 3 Las entradas que se controlarán mediante controladores de entrada de terceros deben anotarse con el valor que requieran dichos controladores.

Importante A menos que desee que NGINX sea el controlador de entrada predeterminado, no utilice **nginx** como el controlador de entrada NGINX, ya que eso convertirá a NGINX en el controlador de entrada predeterminado.

Escenario 3: NCP no controla ninguna entrada, independientemente de su anotación.

Siga este procedimiento:

- 1 Edite la sección `nsx_v3` del archivo `ncp.ini` de cada nodo principal.
 - a Establezca `use_native_loadbalancer` como **False**. El valor de `default_ingress_class_nsx` ahora es irrelevante.
- 2 Reinicie NCP en cada nodo principal. Esto puede provocar una conmutación por error generalizada.

Tenga en cuenta que NCP tampoco controlará los servicios del tipo LoadBalancer.

Administrar NSX Container Plug-in

6

Puede administrar NSX Container Plug-in desde la GUI de NSX Manager o desde la interfaz de la línea de comandos (command-line interface, CLI).

Nota Si una máquina virtual de host del contenedor se ejecuta en ESXi 6.5 y se migra mediante vMotion a otro host ESXi 6.5, los contenedores que se ejecuten en el host del contenedor perderán la conectividad con los contenedores que se ejecuten en otros hosts del contenedor. Para resolver el problema, desconecte y conecte la vNIC del host del contenedor. Este problema no ocurre con ESXi 6.5 Update 1 o versiones posteriores.

HyperBus reserva el identificador de VLAN 4094 en el hipervisor para la configuración de PVLAN. Este identificador no se puede cambiar. Para evitar conflictos de VLAN, no configure los conmutadores lógicos de VLAN ni las vmknics de VTEP con el mismo identificador de VLAN.

Este capítulo incluye los siguientes temas:

- [Visualización de la información de errores almacenada en objetos NSXError de recursos de Kubernetes](#)
- [Puertos lógicos conectados a CIF](#)
- [Comandos de la CLI](#)
- [Códigos de error](#)

Visualización de la información de errores almacenada en objetos NSXError de recursos de Kubernetes

Para cada objeto de recurso de Kubernetes que tenga errores de back-end NSX, se crea un objeto NSXError con información del error. También existe un objeto de error para todos los errores que se aplican a todo el clúster.

Esta función no está habilitada de forma predeterminada. Para habilitarla, debe establecer `enable_nsx_err_crd` como `True` en `ncp.ini` al instalar NCP.

Nota No debe crear, actualizar ni eliminar objetos NSXError.

Comandos para mostrar los objetos NSXError:

- `kubectl get nsxerrors`
Enumera todos los objetos NSXError.
- `kubectl get nsxerrors -l error-object-type=<type of resource>`
Enumera los objetos NSXError relacionados con un tipo específico de objetos de Kubernetes; por ejemplo, los objetos de tipo `services`.
- `kubectl get nsxerrors <nsxerror name> -o yaml`
Muestra los detalles de un objeto NSXError.
- `kubectl get svc <service name> -o yaml | grep nsxerror`
Busca el objeto NSXError asociado a un servicio específico.

Cuando se muestran los detalles de un objeto NSXError, la sección de especificaciones contiene la siguiente información importante. Por ejemplo,

```
error-object-id: default.svc-1
error-object-name: svc-1
error-object-ns: default
error-object-type: services
message:
- '[2019-01-21 20:25:36]23705: Number of pool members requested exceed LoadBalancerlimit'
```

En este ejemplo, el espacio de nombres es `default`. El nombre del servicio es `svc-1`. El tipo de recurso de Kubernetes es `services`.

En esta versión, el objeto NSXError admite los siguientes errores.

- El ajuste de escala automático no pudo asignar equilibradores de carga adicionales debido a un límite de NSX Edge.
- El número de servidores virtuales del equilibrador de carga supera el límite (el ajuste de escala automático no está habilitado).
- El número de grupos de servidores del equilibrador de carga supera el límite.
- El número de miembros del grupo de servidores del equilibrador de carga supera el límite de equilibrador de carga o el límite de NSX Edge.
- Las direcciones IP flotantes se agotan al procesar un servicio de tipo equilibrador de carga.

Puertos lógicos conectados a CIF

Las CIF (interfaces de los contenedores) son interfaces de red de contenedores conectados a los puertos lógicos de un conmutador. Estos puertos se denominan puertos lógicos conectados a CIF.

Puede administrar los puertos lógicos conectados a CIF desde la GUI de NSX Manager.

Administrar los puertos lógicos conectados a CIF

Desplácese hasta **Redes > Conmutación > Puertos** para ver todos los puertos lógicos, incluidos los puertos lógicos conectados a CIF. Haga clic en el vínculo de conexión de un puerto lógico conectado a CIF para ver la información de la conexión. Haga clic en el vínculo del puerto lógico para abrir un panel de ventana con cuatro pestañas: Información general, Supervisor, Administrar y Relacionado. Al hacer clic en **Relacionado > Puertos lógicos**, aparecen los puertos lógicos relacionados de un conmutador de enlace ascendente. Para obtener más información sobre los puertos de conmutación, consulte la *Guía de administración de NSX-T*.

Herramientas para supervisar la red

Las siguientes herramientas admiten puertos lógicos con conexión a CIF. Para obtener más información sobre estas herramientas, consulte la *Guía de administración de NSX-T*.

- Traceflow
- Conexión de puertos
- IPFIX
- Se admite la creación de reflejo de puertos remotos con la encapsulación GRE de un puerto de conmutador lógico que se conecta a un contenedor. Para obtener más información, consulte el apartado "Información sobre el perfil de conmutación de creación de reflejo del puerto" de la *Guía de administración de NSX-T*. Sin embargo, la creación de reflejo de puerto del puerto CIF a VIF no se admite mediante la interfaz de usuario de administrador.

Comandos de la CLI

Para ejecutar comandos de la CLI, inicie sesión en el contenedor de NSX Container Plug-in, abra un terminal y ejecute el comando `nsxcli`.

También puede obtener avisos de la CLI ejecutando el siguiente comando en un nodo:

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

Tabla 6-1. Comandos de la CLI para el contenedor de NCP

Tipo	Comando	Nota
Estado	<code>get ncp-master status</code>	Para Kubernetes y PCF.
Estado	<code>get ncp-nsx status</code>	Para Kubernetes y PCF.
Estado	<code>get ncp-watcher <nombre-monitor></code>	Para Kubernetes y PCF.
Estado	<code>get ncp-watchers</code>	Para Kubernetes y PCF.
Estado	<code>get ncp-k8s-api-server status</code>	Solo para Kubernetes.
Estado	<code>check projects</code>	Solo para Kubernetes.
Estado	<code>check project <nombre-proyecto></code>	Solo para Kubernetes.

Tabla 6-1. Comandos de la CLI para el contenedor de NCP (continuación)

Tipo	Comando	Nota
Estado	get ncp-bbs status	Solo para PCF.
Estado	get ncp-capi status	Solo para PCF.
Estado	get ncp-policy-server status	Solo para PCF.
Caché	get project-caches	Solo para Kubernetes.
Caché	get project-cache <nombre-proyecto>	Solo para Kubernetes.
Caché	get namespace-caches	Solo para Kubernetes.
Caché	get namespace-cache <nombre-espaciodenombres>	Solo para Kubernetes.
Caché	get pod-caches	Solo para Kubernetes.
Caché	get pod-cache <nombre-pod>	Solo para Kubernetes.
Caché	get ingress-caches	Solo para Kubernetes.
Caché	get ingress-cache <ingress-name>	Solo para Kubernetes.
Caché	get ingress-controllers	Solo para Kubernetes.
Caché	get ingress-controller <nombre-controlador-entrada>	Solo para Kubernetes.
Caché	get network-policy-caches	Solo para Kubernetes.
Caché	get network-policy-cache <nombre-pod>	Solo para Kubernetes.
Caché	get asg-caches	Solo para PCF.
Caché	get asg-cache <ID-asg>	Solo para PCF.
Caché	get org-caches	Solo para PCF.
Caché	get org-cache <ID-org>	Solo para PCF.
Caché	get space-caches	Solo para PCF.
Caché	get space-cache <ID-espacio>	Solo para PCF.
Caché	get app-caches	Solo para PCF.
Caché	get app-cache <<ID-aplicación>	Solo para PCF.
Caché	get instance-caches <ID-aplicación>	Solo para PCF.
Caché	get instance-cache <ID-aplicación> <ID-instancia>	Solo para PCF.
Caché	get policy-caches	Solo para PCF.
Soporte técnico	get ncp-log file <nombredearchivo>	Para Kubernetes y PCF.

Tabla 6-1. Comandos de la CLI para el contenedor de NCP (continuación)

Tipo	Comando	Nota
Soporte técnico	get ncp-log-level	Para Kubernetes y PCF.
Soporte técnico	set ncp-log-level <nivel-registro>	Para Kubernetes y PCF.
Soporte técnico	get support-bundle file <nombredearchivo>	Solo para Kubernetes.
Soporte técnico	get node-agent-log file <nombredearchivo>	Solo para Kubernetes.
Soporte técnico	get node-agent-log file <nombredearchivo> <nombre-nodo>	Solo para Kubernetes.

Tabla 6-2. Comandos de la CLI para el contenedor de agentes del nodo de NSX

Tipo	Comando
Estado	get node-agent-hyperbus status
Caché	get container-cache <nombre-contenedor>
Caché	get container-caches

Tabla 6-3. Comandos de la CLI para el contenedor de Kube Proxy de NSX

Tipo	Comando
Estado	get ncp-k8s-api-server status
Estado	get kube-proxy-watcher <nombre-monitor>
Estado	get kube-proxy-watchers
Estado	dump ovs-flows

Comandos de estado para el contenedor de NCP

■ Mostrar el estado del maestro de NCP

```
get ncp-master status
```

Ejemplo:

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

- Mostrar el estado de la conexión entre NCP y NSX Manager

```
get ncp-nsx status
```

Ejemplo:

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

- Mostrar el estado del monitor para la entrada, el espacio de nombres, el pod y el servicio

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

Ejemplo:

```
kubenode> get ncp-watchers

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

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

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

service:
  Average event processing time: 3 msec (in past 3600-sec window)
  Current watcher started time: Mar 02 2017 10:51:37 PST
  Number of events processed: 1 (in past 3600-sec window)
  Total events processed by current watcher: 1
  Total events processed since watcher thread created: 1
  Total watcher recycle count: 0
```

```
Watcher thread created time: Mar 02 2017 10:51:37 PST
Watcher thread status: Up
```

```
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
```

- Mostrar el estado de conexión entre el servidor de API de NCP y de Kubernetes

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

Ejemplo:

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

- Comprobar todos los proyectos o uno específico

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

Ejemplo:

```
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
```

- Comprobar el estado de conexión entre BBS de NCP y PCF

```
get ncp-bbs status
```

Ejemplo:

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

- Comprobar el estado de conexión entre CAPI de NCP y PCF

```
get ncp-capi status
```

Ejemplo:

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

- Comprobar el estado de conexión entre el servidor de directivas de NCP y PCF

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

Ejemplo:

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

Comandos de caché para el contenedor de NCP

- Obtener la caché interna para los proyectos o los espacios de nombres

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

Ejemplo:

```
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
```

■ Obtener la caché interna para los pods

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

Ejemplo:

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

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

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

- Obtener todas las cachés de entrada o una específica

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

Ejemplo:

```
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
```

- Obtener información sobre todos los controladores de entrada o uno específico, incluidos los controladores que están deshabilitados

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

Ejemplo:

```
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

```

■ Obtener cachés de directiva de red o una específica

```

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

```

Ejemplo:

```

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

```

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

- Obtener todas las cachés de ASG o una específica

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

Ejemplo:

```
node> get asg-caches
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:

node> get asg-cache edc04715-d04c-4e63-abbcb-d601a668db6
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:

```

- Obtener todas las cachés de organización o una específica

```

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

```

Ejemplo:

```

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
```

- Obtener todas las cachés de espacio o una específica

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

Ejemplo:

```
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
```

- Obtener todas las cachés de aplicación o una específica

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

Ejemplo:

```
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

```

- Obtener todas cachés de instancia de una aplicación o una caché de instancia específica

```

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

```

Ejemplo:

```

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
```

- Obtener todas las cachés de directiva

```
get policy-caches
```

Ejemplo:

```
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
```

Comandos de soporte para el contenedor de NCP

- Guardar el paquete de soporte técnico de NCP en el almacén de archivos

El paquete de soporte técnico consta de los archivos de registro de todos los contenedores de los pods con la etiqueta **tier:nsx-networking**. El archivo de paquete tiene formato tgz y se guarda en el directorio del almacén de archivos predeterminado de la CLI `/var/vmware/nsx/file-store`. Puede utilizar el comando del almacén de archivos de la CLI para copiar el archivo de paquete a un sitio remoto.

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

Ejemplo:

```
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
```

- Guardar los registros de NCP en el almacén de archivos

El archivo de registro se guarda en formato tgz y en el directorio del almacén de archivos predeterminado de la CLI `/var/vmware/nsx/file-store`. Puede utilizar el comando del almacén de archivos de la CLI para copiar el archivo de paquete a un sitio remoto.

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

Ejemplo:

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

- Guardar los registros del agente del nodo en el almacén de archivos

Guarde los registros del agente del nodo de uno nodo o de todos. Los registros se guardan en formato tgz y en el directorio del almacén de archivos predeterminado de la CLI `/var/vmware/nsx/file-store`. Puede utilizar el comando del almacén de archivos de la CLI para copiar el archivo de paquete a un sitio remoto.

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

Ejemplo:

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

- Obtener y establecer el nivel de registro

Los niveles de registro disponibles son NOTSET, DEBUG, INFO, WARNING, ERROR y CRITICAL.

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

Ejemplo:

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

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

Comandos de estado para el contenedor de agentes del nodo de NSX

- Mostrar el estado de conexión entre HyperBus y el agente de este nodo.

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

Ejemplo:

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

Comandos de caché para el contenedor de agentes del nodo de NSX

- Obtener la caché interna para contenedores de agentes del nodo de NSX.

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

Ejemplo:

```
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
```

Comandos de estado para el contenedor del proxy de NSX Kube

- Mostrar el estado de conexión entre el servidor de Kube Proxy y de Kubernetes

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

Ejemplo:

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

■ Mostrar el estado del monitor de Kube Proxy

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

Ejemplo:

```
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
```

■ Volcar los flujos OVS en un nodo

```
dump ovs-flows
```

Ejemplo:

```
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

```

Códigos de error

En esta sección se muestran los códigos de error que generan los diferentes componentes.

Códigos de error de NCP

Código de error	Descripción
NCP00001	Configuración no válida
NCP00002	Error en la inicialización
NCP00003	Estado no válido
NCP00004	Adaptador no válido
NCP00005	Certificado no encontrado
NCP00006	Token no encontrado
NCP00007	Configuración de NSX no válida
NCP00008	Etiqueta de NSX no válida
NCP00009	Error en la conexión de NSX
NCP00010	Etiqueta de nodo no encontrada
NCP00011	Puerto de conmutador lógico de nodo no válido
NCP00012	Error en la actualización de VIF principal
NCP00013	VLAN agotada
NCP00014	Error en la liberación de VLAN
NCP00015	Grupo de direcciones IP agotado
NCP00016	Error en la liberación de IP
NCP00017	Bloque de direcciones IP agotado
NCP00018	Error en la creación de la subred de direcciones IP

Código de error	Descripción
NCP00019	Error en la eliminación de la subred de direcciones IP
NCP00020	Error en la creación del grupo de direcciones IP
NCP00021	Error en la eliminación del grupo de direcciones IP
NCP00022	Error en la creación del enrutador lógico
NCP00023	Error en la actualización del enrutador lógico
NCP00024	Error en la eliminación del enrutador lógico
NCP00025	Error en la creación del conmutador lógico
NCP00026	Error en la actualización del conmutador lógico
NCP00027	Error en la eliminación del conmutador lógico
NCP00028	Error en la creación del puerto de enrutador lógico
NCP00029	Error en la eliminación del puerto de enrutador lógico
NCP00030	Error en la creación del puerto de conmutador lógico
NCP00031	Error en la actualización del puerto de conmutador lógico
NCP00032	Error en la eliminación del puerto de conmutador lógico
NCP00033	Directiva de red no encontrada
NCP00034	Error en la creación del firewall
NCP00035	Error en la lectura del firewall
NCP00036	Error en la actualización del firewall
NCP00037	Error en la eliminación del firewall
NCP00038	Varias instancias de firewall encontradas
NCP00039	Error en la creación del grupo NSGroup
NCP00040	Error en la eliminación del grupo NSGroup
NCP00041	Error en la creación del conjunto de direcciones IP
NCP00042	Error en la actualización del conjunto de direcciones IP
NCP00043	Error en la eliminación del conjunto de direcciones IP
NCP00044	Error en la creación de reglas SNAT
NCP00045	Error en la eliminación de reglas SNAT

Código de error	Descripción
NCP00046	Error en la conexión de la API de adaptador
NCP00047	Excepción de monitor de adaptador
NCP00048	Error en la eliminación del servicio de equilibrador de carga
NCP00049	Error en la creación del servidor virtual del equilibrador de carga
NCP00050	Error en la actualización del servidor virtual del equilibrador de carga
Código de error	Descripción
NCP00051	Error en la eliminación del servidor virtual del equilibrador de carga
NCP00052	Error en la creación del grupo de equilibradores de carga
NCP00053	Error en la actualización del grupo de equilibradores de carga
NCP00054	Error en la eliminación del grupo de equilibradores de carga
NCP00055	Error en la creación de la regla de equilibrador de carga
NCP00056	Error en la actualización de la regla de equilibrador de carga
NCP00057	Error en la eliminación de la regla de equilibrador de carga
NCP00058	Error en la liberación de IP del grupo de equilibradores de carga
NCP00059	Asociación entre el servicio y el servidor virtual del equilibrador de carga no encontrada
NCP00060	Error en la actualización del grupo NSGroup
NCP00061	Error en la obtención de las reglas de firewall
NCP00062	Grupo NSGroup sin criterios
NCP00063	Máquina virtual de nodo no encontrada
NCP00064	VIF de nodo no encontrado
NCP00065	Error en la importación del certificado
NCP00066	Error en la anulación de la importación del certificado
NCP00067	Error en la actualización del enlace de SSL
NCP00068	Perfil de SSL no encontrado
NCP00069	Grupo de direcciones IP no encontrado
NCP00070	Clúster de Edge T0 no encontrado
NCP00071	Error en la actualización del grupo de direcciones IP
NCP00072	Error en el distribuidor

Código de error	Descripción
NCP00073	Error en la eliminación de reglas NAT
NCP00074	Error en la obtención del puerto de enrutador lógico
NCP00075	Error en la validación de la configuración de NSX

Código de error	Descripción
NCP00076	Error en la actualización de reglas SNAT
NCP00077	Regla SNAT superpuesta
NCP00078	Error en la adición de endpoints de equilibrador de carga
NCP00079	Error en la actualización de endpoints de equilibrador de carga
NCP00080	Error en la creación del grupo de reglas de equilibrador de carga
NCP00081	Servidor virtual del equilibrador de carga no encontrado
NCP00082	Error en la lectura del conjunto de direcciones IP
NCP00083	Error en la obtención del grupo SNAT
NCP00084	Error en la creación del servicio de equilibrador de carga
NCP00085	Error en la actualización del servicio de equilibrador de carga
NCP00086	Error en la actualización del puerto de enrutador lógico
NCP00087	Error en la inicialización del equilibrador de carga
NCP00088	El grupo de direcciones IP no es único
NCP00089	Error en la sincronización de la memoria caché del equilibrador de carga de capa 7
NCP00090	El grupo de equilibradores de carga no existe
NCP00091	Error en la inicialización de la memoria caché de la regla de equilibrador de carga
NCP00092	Error en el proceso SNAT
NCP00093	Error en el certificado predeterminado de equilibrador de carga
NCP00094	Error en la eliminación del endpoint de equilibrador de carga
NCP00095	Proyecto no encontrado
NCP00096	Acceso denegado al grupo
NCP00097	No se pudo obtener un servicio de equilibrador de carga
NCP00098	No se pudo crear un servicio de equilibrador de carga
NCP00099	Error en la sincronización de la memoria caché del grupo de equilibradores de carga

Códigos de error del agente de nodo de NSX

Código de error	Descripción
NCP01001	Vínculo superior de OVS no encontrado
NCP01002	Dirección MAC de host no encontrada
NCP01003	Error en la creación del puerto de OVS
NCP01004	Sin configuración de pod
NCP01005	Error en la configuración de pod
NCP01006	Error en la anulación de la configuración de pod
NCP01007	Socket de CNI no encontrado
NCP01008	Error en la conexión de CNI
NCP01009	Error de coincidencia de la versión de CNI
NCP01010	Error en la recepción del mensaje de CNI
NCP01011	Error en la transmisión del mensaje de CNI
NCP01012	Error en la conexión de HyperBus
NCP01013	Error de coincidencia de la versión de HyperBus
NCP01014	Error en la recepción del mensaje de HyperBus
NCP01015	Error en la transmisión del mensaje de HyperBus
NCP01016	Error en el envío de GARP
NCP01017	Error en la configuración de interfaz

Códigos de error de nsx-kube-proxy

Código de error	Descripción
NCP02001	Puerto de puerta de enlace no válido de proxy
NCP02002	Error en el comando de proxy
NCP02003	Error en la validación de proxy

Códigos de error de la CLI

Código de error	Descripción
NCP03001	Error en el inicio de la CLI
NCP03002	Error en la creación del socket de la CLI

Código de error	Descripción
NCP03003	Excepción de socket de la CLI
NCP03004	Solicitud no válida del cliente de la CLI
NCP03005	Error en la transmisión del servidor de la CLI
NCP03006	Error en la recepción del servidor de la CLI
NCP03007	Error en la ejecución del comando de la CLI

Códigos de error de Kubernetes

Código de error	Descripción
NCP05001	Error en la conexión de Kubernetes
NCP05002	Configuración no válida de Kubernetes
NCP05003	Error en la solicitud de Kubernetes
NCP05004	Clave de Kubernetes no encontrada
NCP05005	Tipo de Kubernetes no encontrado
NCP05006	Excepción de monitor de Kubernetes
NCP05007	Longitud no válida del recurso de Kubernetes
NCP05008	Tipo no válido del recurso de Kubernetes
NCP05009	Error en el identificador del recurso de Kubernetes
NCP05010	Error en el identificador del servicio de Kubernetes
NCP05011	Error en el identificador del endpoint de Kubernetes
NCP05012	Error en el identificador de la entrada de Kubernetes
NCP05013	Error en el identificador de la directiva de red de Kubernetes
NCP05014	Error en el identificador del nodo de Kubernetes
NCP05015	Error en el identificador del espacio de nombres de Kubernetes
NCP05016	Error en el identificador del pod de Kubernetes
NCP05017	Error en el identificador del secreto de Kubernetes
NCP05018	Error en el back-end predeterminado de Kubernetes
NCP05019	Expresión de coincidencia no admitida de Kubernetes
NCP05020	Error en la actualización del estado de Kubernetes
NCP05021	Error en la actualización de la anotación de Kubernetes

Código de error	Descripción
NCP05022	Memoria caché de espacio de nombres de Kubernetes no encontrada
NCP05023	Secreto de Kubernetes no encontrado
NCP05024	El back-end predeterminado de Kubernetes está en uso
NCP05025	Error en el identificador del servicio de equilibrador de carga de Kubernetes

Códigos de error de Pivotal Cloud Foundry

Código de error	Descripción
NCP06001	Error en la conexión de BBS de PCF
NCP06002	Error en la conexión de CAPI de PCF
NCP06006	Memoria caché de PCF no encontrada
NCP06007	Dominio desconocido de PCF
NCP06020	Error en la conexión del servidor de directivas de PCF
NCP06021	Error en el procesamiento de directivas de PCF
NCP06030	Error en el procesamiento de eventos de PCF
NCP06031	Tipo de evento inesperado de PCF
NCP06032	Instancia de evento inesperada de PCF
NCP06033	Error en la eliminación de la tarea de PCF
NCP06034	Error de acceso al archivo de PCF