

NSX Container Plug-In für Kubernetes und Cloud Foundry – Installations- und Administratorhandbuch

VMware NSX Container Plug-In 2.4, 2.4.1

VMware NSX-T Data Center 2.4

Die aktuellste technische Dokumentation finden Sie auf der VMware-Website unter:

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

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

VMware Global, Inc.
Zweigniederlassung Deutschland
Willy-Brandt-Platz 2
81829 München
Germany
Tel.: +49 (0) 89 3706 17 000
Fax: +49 (0) 89 3706 17 333
www.vmware.com/de

Copyright © 2017-2019 VMware, Inc. Alle Rechte vorbehalten. [Urheberrechts- und Markenhinweise](#).

Inhalt

NSX Container Plug-in für Kubernetes und Cloud Foundry – Installations- und Administratorhandbuch 5

- 1 Übersicht über NSX Container Plug-in 6**
 - [Kompatibilitätsanforderungen 7](#)
 - [Überblick über die Installation 8](#)
 - [Upgrade von NCP in einer Kubernetes-Umgebung 8](#)
 - [Upgrade von NCP in einer Pivotal Cloud Foundry\(PCF\)-Umgebung 9](#)
- 2 Einrichten von NSX-T-Ressourcen 11**
 - [Konfigurieren von NSX-T-Ressourcen 11](#)
- 3 Installieren von NCP in einer Kubernetes-Umgebung 18**
 - [Installieren des NSX-T Data Center-CNI-Plug-Ins 18](#)
 - [Installieren und Konfigurieren von OVS 19](#)
 - [Konfigurieren von NSX-T Data Center-Netzwerken für Kubernetes-Knoten 21](#)
 - [Installieren des NSX-Knotenagenten 22](#)
 - [Configmap für ncp.ini in nsx-node-agent-ds.yml 23](#)
 - [Installieren von NSX Container Plug-in 26](#)
 - [Configmap für ncp.ini in ncp-rc.yml 29](#)
 - [Bereitstellen eines PEM-verschlüsselten Zertifikats und privaten Schlüssels im NCP-Pod 34](#)
 - [Bereitstellen einer Zertifikatsdatei im NCP-Pod 35](#)
 - [Konfigurieren von Syslog 35](#)
 - [Erstellen eines Sidecar-Containers für Syslog 36](#)
 - [Erstellen eines DaemonSet-Replikats für Syslog 38](#)
 - [Beispiel: Konfigurieren der Protokollrotation und des in einem Sidecar-Container ausgeführten Syslogs 39](#)
 - [Sicherheitsüberlegungen 46](#)
 - [Tipps zum Konfigurieren von Netzwerkressourcen 49](#)
- 4 Installieren von NCP in einer Pivotal Cloud Foundry(PCF)-Umgebung 51**
 - [Installieren von NCP in einer Pivotal Cloud Foundry\(PCF\)-Umgebung 51](#)
- 5 Load Balancing 54**
 - [Konfigurieren von Load Balancing 54](#)
 - [Ingress-Controller von Drittanbietern 62](#)
- 6 Verwalten von NSX Container Plug-in 66**

Anzeigen von in der Kubernetes-Ressource NSXError gespeicherten Fehlerinformationen	66
CIF-verknüpfte logische Ports	67
CLI-Befehle	68
Fehlercodes	87

NSX Container Plug-in für Kubernetes und Cloud Foundry – Installations- und Administratorhandbuch

In diesem Handbuch wird die Installation und Verwaltung von NSX Container Plug-in (NCP) zur Bereitstellung von Integration zwischen NSX-T Data Center und Kubernetes sowie zwischen NSX-T Data Center und Pivotal Cloud Foundry (PCF) beschrieben.

Zielgruppe

Dieses Handbuch ist für die System- und Netzwerkadministratoren bestimmt. Es wird vorausgesetzt, dass Sie mit der Installation und Verwaltung von NSX-T Data Center, Kubernetes und Pivotal Cloud Foundry vertraut sind.

VMware Technical Publications – Glossar

VMware Technical Publications enthält ein Glossar mit Begriffen, die Ihnen möglicherweise unbekannt sind. Definitionen von Begriffen, die in der technischen Dokumentation von VMware verwendet werden, finden Sie unter <http://www.vmware.com/support/pubs>.

Übersicht über NSX Container Plug-in

1

NSX Container Plug-in (NCP) stellt Integration zwischen NSX-T Data Center und Container-Orchestrierung wie z. B. Kubernetes sowie Integration zwischen NSX-T Data Center und Container-basierten PaaS-Produkten (Platform-as-a-Service), wie z. B. OpenShift und Pivotal Cloud Foundry, bereit. In diesem Handbuch wird die Einrichtung von NCP mit Kubernetes und Pivotal Cloud Foundry beschrieben.

Die Hauptkomponente von NCP wird in einem Container ausgeführt und kommuniziert mit NSX Manager und mit der Kubernetes-Steuerungsebene. NCP überwacht Änderungen an den Containern und anderen Ressourcen und verwaltet Netzwerkressourcen wie logische Ports, Switches, Router und Sicherheitsgruppen für die Container per Aufruf der NSX API.

Das NSX CNI-Plugin wird auf jedem Kubernetes-Knoten ausgeführt. Es überwacht Ereignisse im Container-Lebenszyklus, verbindet eine Containerschnittstelle mit dem Gast-vSwitch und programmiert den Gast-vSwitch für die Kennzeichnung und Weiterleitung des Containerdatenverkehrs zwischen den Containerschnittstellen und der VNIC.

NCP bietet die folgenden Funktionen:

- Es erstellt automatisch eine logische NSX-T Data Center-Topologie für einen Kubernetes-Cluster und erstellt ein separates logisches Netzwerk für jeden Kubernetes-Namespace.
- Es verbindet Kubernetes-Pods mit dem logischen Netzwerk und weist IP- und MAC-Adressen zu.
- Es unterstützt Netzwerkadressübersetzung (Network Address Translation – NAT) und weist eine separate SNAT-IP für jeden Kubernetes-Namespace zu.

Hinweis Bei der Konfiguration von NAT darf die Gesamtzahl der übersetzten IPs 1000 nicht überschreiten.

- Es implementiert Kubernetes-Netzwerkrichtlinien mit verteilter NSX-T Data Center-Firewall.
 - Unterstützung für Ingress- und Egress-Netzwerkrichtlinien.
 - Unterstützung für IPBlock-Selektor in Netzwerkrichtlinien.
 - Unterstützung für `matchLabels` und `matchExpression` beim Angeben von Bezeichnungsselektoren für Netzwerkrichtlinien.

- Unterstützung für die Auswahl von Pods in einem anderen Namespace.
- Es implementiert einen Kubernetes-Dienst des Typs ClusterIP und einen Dienst des Typs LoadBalancer.
- Es implementiert Kubernetes-Ingress mit NSX-T-Load Balancer der Schicht 7.
 - Unterstützung für HTTP-Ingress und HTTPS-Ingress mit TLS-Edge-Beendigung.
 - Unterstützung für Standard-Backend-Konfiguration für den Dateneingang.
 - Unterstützung für Ingress-URI-Umschreibung.
- Es erstellt auf dem logischen NSX-T Data Center-Switch-Port Tags für den Namespace, den Pod-Namen und die Bezeichnungen eines Pods und lässt zu, dass der Administrator die NSX-T-Sicherheitsgruppen und -richtlinien basierend auf den Tags festlegt.

In dieser Version unterstützt NCP einen einzelnen Kubernetes-Cluster. Es besteht die Möglichkeit, für mehrere Kubernetes-Cluster, jeweils mit einer eigenen eindeutigen NCP-Instanz, die gleiche NSX-T Data Center-Bereitstellung zu verwenden.

Dieses Kapitel enthält die folgenden Themen:

- [Kompatibilitätsanforderungen](#)
- [Überblick über die Installation](#)
- [Upgrade von NCP in einer Kubernetes-Umgebung](#)
- [Upgrade von NCP in einer Pivotal Cloud Foundry\(PCF\)-Umgebung](#)

Kompatibilitätsanforderungen

NSX Container Plug-in (NCP) weist folgende Kompatibilitätsanforderungen für eine Kubernetes-Umgebung und eine Pivotal Cloud Foundry(PCF)-Umgebung auf.

Tabelle 1-1. Kompatibilitätsanforderungen für eine Kubernetes-Umgebung

Softwareprodukt	Version
NSX-T Data Center	2.3, 2.4
Hypervisor für Container-Host-VMs	<ul style="list-style-type: none"> ■ Unterstützte vSphere-Version ■ RHEL KVM 7.4, 7.5, 7.6 ■ Ubuntu KVM 16.04
Container-Host-Betriebssystem	<ul style="list-style-type: none"> ■ RHEL 7.5, 7.6 ■ Ubuntu 16.04 ■ CentOS 7.4, 7.5
Container-Orchestrator	Kubernetes 1.12, 1.13
Container-Host-Open vSwitch	2.9.1 (im Lieferumfang von NSX-T Data Center 2.3.x), 2.10.2 (im Lieferumfang von NSX-T Data Center 2.4.0)

Tabelle 1-2. Kompatibilitätsanforderungen für eine Cloud Foundry-Umgebung

Softwareprodukt	Version
Hypervisor für Container-Host-VMs	■ Unterstützte vSphere-Version
Container-Orchestrator	<ul style="list-style-type: none"> ■ Pivotal Application Service 2.3.x und Pivotal Operations Manager 2.3.x ■ Pivotal Application Service 2.4.x (außer 2.4.0) und Pivotal Operations Manager 2.4.x (außer 2.4.0)

Überblick über die Installation

In einer Umgebung, in der Kubernetes bereits installiert ist, beinhaltet die NCP-Installation und -Konfiguration in der Regel die folgenden Schritte. Zur erfolgreichen Durchführung der Schritte müssen Sie mit NSX-T Data Center sowie der Installation und Verwaltung von Kubernetes vertraut sein.

- 1 Installieren Sie NSX-T Data Center.
- 2 Erstellen Sie eine Overlay-Transportzone.
- 3 Erstellen Sie einen logischen Overlay-Switch, und verbinden Sie die Kubernetes-Knoten mit dem Switch.
- 4 Erstellen Sie einen logischen Tier-O-Router.
- 5 Erstellen Sie IP-Blöcke für Kubernetes-Pods.
- 6 Erstellen Sie IP-Pools für SNAT (Source Network Address Translation).
- 7 Installieren Sie das NSX CNI-Plugin (Containernetzwerkschnittstelle) auf jedem Knoten.
- 8 Installieren Sie OVS (Open vSwitch) auf jedem Knoten.
- 9 Konfigurieren Sie das NSX-T-Netzwerk für Kubernetes-Knoten.
- 10 Installieren Sie den NSX-Knotenagent als DaemonSet.
- 11 Installieren Sie NCP als ReplicationController.
- 12 Stellen Sie Sicherheitszertifikate im NCP-Pod bereit.

Upgrade von NCP in einer Kubernetes-Umgebung

In diesem Abschnitt wird beschrieben, wie in einer Kubernetes-Umgebung ein Upgrade von NCP auf 2.4.0 vorgenommen wird.

Verfahren

- 1 Führen Sie das Upgrade von NCP ReplicationController mit dem folgenden Befehl durch (ersetzen Sie <image> durch den tatsächlichen Namen des Image).

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

- 2 Führen Sie das Upgrade des daemonSet des NSX-Knotenagent mit den folgenden Befehlen durch (ersetzen Sie <image> durch den tatsächlichen Namen des Image).

```
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 Führen Sie das Upgrade des CNI DEB/RPM-Pakets auf 2.4.0 mit dem folgenden Befehl durch (ersetzen Sie <cni deb> und <cni rpm> durch den tatsächlichen Namen des Pakets).

Unter Ubuntu:

```
dkpg -i <cni deb>
```

Unter RHEL oder CentOS:

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

- 4 (Optional) Führen Sie ein Upgrade von NSX-T Data Center auf 2.4 durch.

Wenn der Hypervisor ESXi ist, führen Sie vor dem Upgrade auf NSX-T Data Center ein Upgrade von ESXi von 6.5 auf mindestens 6.5p03 oder von 6.7 auf mindestens 6.7ep6 durch.

- 5 (Optional) Führen Sie ein Upgrade des Hypervisors durch (KVM oder Bare-Metal-Container).
- 6 (Optional) Führen Sie ein Upgrade des Container-Hosts durch (RHEL, Ubuntu oder CentOS).
- 7 (Optional) Führen Sie ein Upgrade von Kubernetes durch.
- 8 (Optional) Führen Sie ein Upgrade von OVS durch.

Bei einem Bare-Metal-Container wird mit dem Upgrade von NSX-T Data Center auch ein Upgrade von OVS durchgeführt, sodass dieser Schritt nicht notwendig ist.

Während dieses Schritts können vorübergehend Kommunikationsfehler zwischen nsx-kube-proxy und nsx-node-agent auftreten. Dies ist das erwartete Verhalten und weist nicht auf ein Problem hin.

Upgrade von NCP in einer Pivotal Cloud Foundry(PCF)-Umgebung

In diesem Abschnitt wird beschrieben, wie in einer Pivotal Cloud Foundry(PCF)-Umgebung ein Upgrade von NCP auf 2.4.0 vorgenommen wird.

Verfahren

- 1 Führen Sie ein Upgrade der NCP- oder NSX-T-Kachel auf Version 2.4.0 durch.
- 2 (Optional) Führen Sie ein Upgrade von Pivotal Operations Manager und anschließend von Pivotal Application Service durch.
- 3 (Optional) Führen Sie ein Upgrade von NSX-T Data Center auf 2.4 durch.

Wenn der Hypervisor ESXi ist, führen Sie vor dem Upgrade auf NSX-T Data Center ein Upgrade von ESXi von 6.5 auf mindestens 6.5p03 oder von 6.7 auf mindestens 6.7ep6 durch.

- 4 (Optional) Führen Sie ein Upgrade des Hypervisors durch (KVM oder Bare-Metal-Container).

Einrichten von NSX-T-Ressourcen

2

Vor der Installation von NSX Container Plug-in müssen Sie bestimmte NSX-T Data Center-Ressourcen einrichten.

Dieses Kapitel enthält die folgenden Themen:

- [Konfigurieren von NSX-T-Ressourcen](#)

Konfigurieren von NSX-T-Ressourcen

Zu den zu konfigurierenden NSX-T Data Center-Ressourcen gehören eine Overlay-Transportzone, ein logischer Tier-0-Router, ein logischer Switch zum Verbinden der virtuellen Maschinen des Knotens, IP-Blöcke für Kubernetes-Knoten und ein IP-Pool für SNAT.

Wichtig Wenn die Ausführung mit NSX-T Data Center 2.4 oder höher erfolgt, müssen Sie mithilfe der Registerkarte **Netzwerk und Sicherheit – Erweitert** NSX-T-Ressourcen konfigurieren.

In der NCP-Konfigurationsdatei `ncp.ini` sind die NSX-T Data Center-Ressourcen unter Verwendung ihrer UUIDs oder Namen angegeben.

Overlay-Transportzone

Melden Sie sich bei NSX Manager an und navigieren Sie zu **System > Fabric > Transportzonen**. Suchen Sie nach der für Containernetzwerke verwendeten Overlay-Transportzone oder erstellen Sie eine neue.

Geben Sie eine Overlay-Transportzone für einen Cluster an, indem Sie die Option `overlay_tz` im Abschnitt `[nsx_v3]` der Datei `ncp.ini` festlegen. Dieser Schritt ist optional. Wenn Sie `overlay_tz` nicht festlegen, ruft NCP die ID der Overlay-Transportzone automatisch vom Tier-0-Router ab.

Logisches Tier-0-Routing

Melden Sie sich bei NSX Manager an und navigieren Sie zu **Netzwerk und Sicherheit – Erweitert > Netzwerk > Router**. Suchen Sie nach dem für Containernetzwerke verwendeten Router oder erstellen Sie einen neuen.

Geben Sie einen logischen Tier-0-Router für einen Cluster an, indem Sie die Option `tier0_router` im Abschnitt `[nsx_v3]` der Datei `ncp.ini` festlegen.

Hinweis Der Router muss im Aktiv/Standby-Modus erstellt werden.

Logischer Switch

Die vom Knoten für den Netzwerkdatenverkehr verwendeten vNICs müssen mit einem logischen Overlay-Switch verbunden sein. Die Verwaltungsschnittstelle des Knotens muss nicht zwingend mit NSX-T Data Center verbunden sein, obwohl dies die Einrichtung erleichtert. Sie können einen logischen Switch erstellen, indem Sie sich bei NSX Manager anmelden und zu **Netzwerk und Sicherheit – Erweitert > Netzwerk > Switching > Switches** navigieren. Erstellen Sie auf dem Switch logische Ports und hängen Sie die vNICs des Knotens daran an. Die logischen Ports müssen die folgenden Tags aufweisen:

- Tag: `<cluster_name>`, Geltungsbereich: `ncp/cluster`
- Tag: `<node_name>`, Geltungsbereich: `ncp/node_name`

Der Wert für `<cluster_name>` muss mit dem Wert der Option `cluster` im Abschnitt `[coe]` von `ncp.ini` übereinstimmen.

IP-Blöcke für Kubernetes-Pods

Melden Sie sich bei NSX Manager an und navigieren Sie zu **Netzwerk und Sicherheit – Erweitert > Netzwerk > IPAM**, um einen oder mehrere IP-Blöcke zu erstellen. Geben Sie den IP-Block im CIDR-Format an.

Geben Sie IP-Blöcke für Kubernetes-Pods an, indem Sie die Option `container_ip_blocks` im Abschnitt `[nsx_v3]` der Datei `ncp.ini` festlegen.

Sie können IP-Blöcke auch spezifisch für Nicht-SNAT-Namespace (für Kubernetes) oder Cluster (für PCF) erstellen.

Geben Sie Nicht-SNAT-IP-Blöcke ein, indem Sie die Option `no_snat_ip_blocks` im Abschnitt `[nsx_v3]` der Datei `ncp.ini` festlegen.

Wenn Sie Nicht-SNAT-IP-Blöcke erstellen, während NCP ausgeführt wird, müssen Sie NCP neu starten. Andernfalls verwendet NCP weiterhin die freigegebenen IP-Blöcke, bis sie erschöpft sind.

Hinweis Wenn Sie einen IP-Block erstellen, darf das Präfix nicht größer als der Wert des Parameters `subnet_prefix` in der NCP-Konfigurationsdatei `ncp.ini` sein. Weitere Informationen finden Sie unter [Configmap für ncp.ini in ncp-rc.yml](#).

IP-Pool für SNAT

Ein IP-Pool in NSX Manager wird für die Zuteilung von IP-Adressen, die für die Übersetzung von Pod-IPs mithilfe von SNAT-Regeln verwendet werden, und für die Bereitstellung von Ingress-Controllern unter Verwendung von SNAT/DNAT-Regeln verwendet, genau wie OpenStack-Floating-IPs. Diese IP-Adressen werden auch als „externe IP-Adressen“ bezeichnet.

Mehrere Kubernetes-Cluster verwenden denselben externen IP-Pool. Jede NCP-Instanz verwendet einen Teil dieses Pools für den Kubernetes-Cluster, den sie verwaltet. Standardmäßig wird dasselbe Subnetzpräfix für Pod-Subnetze verwendet. Wenn Sie eine andere Subnetzgröße verwenden möchten, aktualisieren Sie die `external_subnet_prefix`-Option im `[nsx_v3]`-Abschnitt in `ncp.ini`.

Sie können IP-Pools für SNAT angeben, indem Sie die Option `external_ip_pools` im Abschnitt `[nsx_v3]` der Datei `ncp.ini` festlegen.

Sie können zu einem anderen IP-Pool wechseln, indem Sie die Konfigurationsdatei ändern und NCP neu starten.

Einschränken eines SNAT-IP-Pools auf bestimmte Kubernetes-Namespace oder PCF-Organisationen

Sie können durch Hinzufügen der folgenden Tags zum IP-Pool festlegen, welchem Kubernetes-Namespace oder welcher PCF Org IPs aus dem SNAT-IP-Pool zugeteilt werden können.

- Für einen Kubernetes-Namespace: `scope: ncp/owner, tag: ns:<namespace_UUID>`
- Für eine PCF Org: `scope: ncp/owner, tag: org:<org_UUID>`

Sie können die Namespace- oder Org-UUID mit einem der folgenden Befehle abrufen:

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

Beachten Sie Folgendes:

- Jedes Tag sollte eine UUID enthalten. Sie können mehrere Tags für denselben Pool erstellen.
- Wenn Sie die Tags ändern, nachdem einigen Namespaces oder Orgs IPs basierend auf den alten Tags zugewiesen wurden, werden diese IPs nicht wiederhergestellt, bis sich die SNAT-Konfigurationen der Kubernetes-Dienste oder PCF-Anwendungen ändern oder NCP neu gestartet wird.
- Die Owner-Tags für Namespace und PCF Org sind optional. Ohne diese Tags können jedem Namespace bzw. jeder PCF Org IPs aus dem SNAT-IP-Pool zugeteilt werden.

Konfigurieren eines SNAT-IP-Pools für einen Dienst

Sie können einen SNAT-IP-Pool für einen Dienst konfigurieren, indem Sie dem Dienst eine Anmerkung hinzufügen. Beispiel:

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

```

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

```

Der von `ncp/snat_pool` angegebene-IP-Pool muss über das Tag `scope: ncp/owner`, `tag: cluster:<cluster_name>` verfügen.

NCP konfiguriert die SNAT-Regel für diesen Dienst. Bei der Quell-IP der Regel handelt es sich um die Gruppe der Backend-Pods. Bei der Ziel-IP handelt es sich um die SNAT-IP, die aus dem angegebenen externen IP-Pool zugeteilt wurde. Falls bei der Konfiguration der SNAT-Regel durch NCP ein Fehler auftritt, wird der Dienst mit der Anmerkung `ncp/error.snat:<error>` versehen. Dies sind die möglichen Fehler:

- `IP_POOL_NOT_FOUND`: Der SNAT-IP-Pool wurde in NSX Manager nicht gefunden.
- `IP_POOL_EXHAUSTED`: Der SNAT-IP-Pool ist ausgeschöpft.
- `IP_POOL_NOT_UNIQUE`: Der durch `ncp/snat_pool` angegebene Pool verweist auf mehrere Pools in NSX Manager.
- `SNAT_POOL_ACCESS_DENY`: Das Owner-Tag des Pools stimmt nicht mit dem Namespace des Diensts überein, der die Zuteilungsanforderung sendet.
- `SNAT_RULE_OVERLAPPED`: Eine neue SNAT-Regel wird erstellt, aber der Pod des SNAT-Diensts gehört auch zu einem anderen SNAT-Dienst, d. h., für denselben Pod sind mehrere SNAT-Regeln vorhanden.
- `POOL_ACCESS_DENIED` – Der von `ncp/snat_pool` angegebene IP-Pool weist das Tag `scope: ncp/owner`, `tag: cluster:<cluster_name>` nicht auf, oder das Owner-Tag des Pools stimmt nicht mit dem Namespace des Diensts überein, der die Zuteilungsanforderung sendet.

Beachten Sie Folgendes:

- Der von `ncp/snat_pool` angegebene Pool sollte bereits in NSX-T Data Center vorhanden sein, bevor der Dienst konfiguriert wird.
- In NSX-T Data Center ist die Priorität der SNAT-Regel für den Dienst höher als die für das Projekt.
- Wenn ein Pod mit mehreren SNAT-Regeln konfiguriert wird, funktioniert nur eine der Regeln.
- Sie können zu einem anderen IP-Pool wechseln, indem Sie die Anmerkung ändern und NCP neu starten.

Konfigurieren eines SNAT-IP-Pools für einen Namespace

Sie können einen SNAT-IP-Pool für einen Namespace konfigurieren, indem Sie dem Namespace eine Anmerkung hinzufügen. Beispiel:

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

NCP konfiguriert die SNAT-Regel für diesen Namespace. Bei der Quell-IP der Regel handelt es sich um die Gruppe der Backend-Pods. Bei der Ziel-IP handelt es sich um die SNAT-IP, die aus dem angegebenen externen IP-Pool zugeteilt wurde. Falls bei der Konfiguration der SNAT-Regel durch NCP ein Fehler auftritt, wird der Namespace mit der Anmerkung `ncp/error.snat:<error>` versehen. Dies sind die möglichen Fehler:

- `IP_POOL_NOT_FOUND`: Der SNAT-IP-Pool wurde in NSX Manager nicht gefunden.
- `IP_POOL_EXHAUSTED`: Der SNAT-IP-Pool ist ausgeschöpft.
- `IP_POOL_NOT_UNIQUE`: Der durch `ncp/snat_pool` angegebene Pool verweist auf mehrere Pools in NSX Manager.
- `POOL_ACCESS_DENIED` – Der von `ncp/snat_pool` angegebene IP-Pool weist das Tag `scope: ncp/owner, tag: cluster:<cluster_name>` nicht auf, oder das Owner-Tag des Pools stimmt nicht mit dem Namespace überein, der die Zuteilungsanforderung sendet.

Beachten Sie Folgendes:

- Sie können nur einen SNAT-IP-Pool in der Anmerkung angeben.
- Der SNAT-IP-Pool muss nicht in `ncp.ini` konfiguriert sein.
- Der von `ncp/snat_pool` angegebene IP-Pool muss über das Tag `scope: ncp/owner, tag: cluster:<cluster_name>` verfügen.
- Der mit `ncp/snat_pool` angegebene IP-Pool kann auch ein Namespace-Tag, `scope: ncp/owner, tag: ns:<namespace_UUID>`, aufweisen.
- Wenn die Anmerkung `ncp/snat_pool` fehlt, verwendet der Namespace den SNAT-IP-Pool für den Cluster.
- Sie können zu einem anderen IP-Pool wechseln, indem Sie die Anmerkung ändern und NCP neu starten.

Konfigurieren eines SNAT-Pools für eine PAS-App

Standardmäßig konfiguriert NCP die SNAT-IP für eine PAS-Organisation (Pivotal Application Service). Sie können eine SNAT-IP für eine App konfigurieren, indem Sie eine App mit einer `manifest.xml` erstellen, die die SNAT-IP-Pool-Informationen enthält. Beispiel:

```
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 konfiguriert die SNAT-Regel für diese Anwendung. Die Quell-IP der Regel ist der Satz von IP-Adressen der Instanzen, und ihre Ziel-IP ist die SNAT-IP, die aus einem externen IP-Pool zugewiesen wurde. Beachten Sie Folgendes:

- Der von `NCP_SNAT_POOL` angegebene Pool sollte bereits in NSX-T Data Center vorhanden sein, bevor die Anwendung mithilfe von Push übertragen wird.
- Die SNAT-Regel für eine Anwendung hat eine höhere Priorität als diejenige für eine Organisation.
- Eine Anwendung kann nur mit einer SNAT-IP konfiguriert werden.
- Sie können zu einem anderen IP-Pool wechseln, indem Sie die Konfiguration ändern und NCP neu starten.

Konfigurieren von SNAT für PCF Version 3

Mit PCF Version 3 können Sie SNAT auf zwei Arten konfigurieren:

- Konfigurieren Sie `NCP_SNAT_POOL` in `manifest.yml`, wenn Sie die App erstellen.

Die App trägt beispielsweise den Namen `bread` und die `manifest.yml` enthält die folgenden Informationen:

```
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
```

Führen Sie die folgenden Befehle aus:

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

- Konfigurieren Sie NCP_SNAT_POOL mithilfe des Befehls `cf v3-set-env`.

Führen Sie die folgenden Befehle aus (unter der Annahme, dass die App den Namen `app3` trägt):

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

(Optional, nur für Kubernetes) Firewall-Markierungsabschnitte

Damit der Administrator Firewallregeln erstellen kann und diese die von NCP erstellten, auf Netzwerkrichtlinien basierenden Firewallabschnitte nicht beeinträchtigen, melden Sie sich bei NSX Manager an, navigieren Sie zu **Sicherheit > Verteilte Firewall > Allgemein** und erstellen Sie zwei Firewallabschnitte.

Geben Sie Firewall-Markierungsabschnitte an, indem Sie die Optionen `bottom_firewall_section_marker` und `top_firewall_section_marker` im Abschnitt `[nsx_v3]` der Datei `ncp.ini` festlegen.

Der untere Firewallabschnitt muss sich unterhalb des oberen Firewallabschnitts befinden. Wenn diese Firewallabschnitte erstellt sind, werden alle von NCP zur Isolierung erstellten Firewallabschnitte oberhalb des unteren Firewallabschnitts und alle von NCP für Richtlinien erstellten Firewallabschnitte unterhalb des oberen Firewallabschnitts erstellt. Wenn diese Markierungsabschnitte nicht erstellt werden, werden alle Isolierungsregeln unten und alle Richtlinienabschnitte oben erstellt. Mehrere markierte Firewallabschnitte mit demselben Wert pro Cluster werden nicht unterstützt und führen zu einem Fehler.

Installieren von NCP in einer Kubernetes-Umgebung

3

Für die Installation von NSX Container Plug-in (NCP) müssen Komponenten auf den Master- und Kubernetes-Knoten installiert werden.

Dieses Kapitel enthält die folgenden Themen:

- [Installieren des NSX-T Data Center-CNI-Plug-Ins](#)
- [Installieren und Konfigurieren von OVS](#)
- [Konfigurieren von NSX-T Data Center-Netzwerken für Kubernetes-Knoten](#)
- [Installieren des NSX-Knotenagenten](#)
- [Configmap für ncp.ini in nsx-node-agent-ds.yml](#)
- [Installieren von NSX Container Plug-in](#)
- [Configmap für ncp.ini in ncp-rc.yml](#)
- [Bereitstellen eines PEM-verschlüsselten Zertifikats und privaten Schlüssels im NCP-Pod](#)
- [Bereitstellen einer Zertifikatdatei im NCP-Pod](#)
- [Konfigurieren von Syslog](#)
- [Sicherheitsüberlegungen](#)
- [Tipps zum Konfigurieren von Netzwerkressourcen](#)

Installieren des NSX-T Data Center-CNI-Plug-Ins

Das NSX-T Data Center-CNI-Plug-In muss auf den Kubernetes-Knoten installiert sein.

Bei Ubuntu wird beim Installieren des NSX-T-CNI-Plug-Ins die AppArmor-Profildatei `ncp-apparmor` in `/etc/apparmor.d` kopiert und geladen. Vor der Installation muss der AppArmor-Dienst ausgeführt werden, und das Verzeichnis `/etc/apparmor.d` muss vorhanden sein. Andernfalls schlägt die Installation fehl. Sie können mit dem folgenden Befehl prüfen, ob das AppArmor-Modul aktiviert ist:

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

Sie können mit dem folgenden Befehl prüfen, ob der AppArmor-Dienst gestartet wurde:

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

Wenn beim Installieren des NSX-T-CNI-Plug-Ins der AppArmor-Dienst nicht ausgeführt wird, zeigt der Installationsvorgang folgende Meldung an, wenn er abgeschlossen wird:

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

Die Meldung weist darauf hin, dass alle Installationsschritte außer dem Laden des AppArmor-Profiles abgeschlossen wurden.

Die `ncp-apparmor`-Profildatei stellt ein AppArmor-Profil für den NSX-Knotenagent mit dem Namen `node-agent-apparmor` bereit, das sich vom `docker-default`-Profil wie folgt unterscheidet:

- Die `deny mount`-Regel wird entfernt.
- Die `mount`-Regel wird hinzugefügt.
- Einige `network`-, `capability`-, `file`- und `umount`-Optionen werden hinzugefügt.

Sie können das `node-agent-apparmor`-Profil durch ein anderes Profil ersetzen. Auf den Profilenames `node-agent-apparmor` wird jedoch in der Datei `nsx-node-agent-ds.yml` verwiesen, die bei der Installation des NSX-Knotenagents verwendet wird. Wenn Sie ein anderes Profil verwenden, müssen Sie den Profilenames in `nsx-node-agent-ds.yml` unter dem Abschnitt `spec:template:metadata:annotations` im folgenden Eintrag angeben:

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

Verfahren

- 1 Laden Sie die entsprechende Installationsdatei für Ihre Linux-Bereitstellung herunter.

Der Dateiname lautet `nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm` oder `nsx-cni-1.0.0.0.0.xxxxxxx.deb`, wobei `xxxxxxx` die Build-Nummer ist.

- 2 Installieren Sie die in Schritt 1 heruntergeladene RPM- oder DEB-Datei.

Das Plug-In wird unter `/opt/cni/bin` installiert. Die CNI-Konfigurationsdatei „`10-nsx.conf`“ wird in `/etc/cni/net.d` kopiert. RPM installiert auch die Konfigurationsdatei `/etc/cni/net.d/99-loopback.conf` für das Loopback-Plug-In.

Installieren und Konfigurieren von OVS

Installieren und konfigurieren OVS (Open vSwitch) auf den Minion-Knoten.

Verfahren

- 1 Laden Sie die Installationsdatei für Ihre Linux-Bereitstellung herunter.

Die Dateinamen lauten `openvswitch-common_2.10.x.xxxxxxx-1_amd64.deb`, `openvswitch-datapath-dkms_2.10.x.xxxxxxx-1_all.deb` und `openvswitch-switch_2.10.x.xxxxxxx-1_amd64.deb`, wobei `xxxxxxx` der Build-Nummer entspricht.

- 2 Installieren Sie die in Schritt 1 heruntergeladene DEB-Datei.
- 3 Führen Sie für Ubuntu den folgenden Befehl aus, um das OVS-Kernelmodul neu zu laden.

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

- 4 Stellen Sie sicher, dass OVS ausgeführt wird.

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

- 5 Erstellen Sie die *br-int*-Instanz, sofern dies noch nicht erfolgt ist.

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

- 6 Fügen Sie die Netzwerkschnittstelle (*node-if*) hinzu, die mit dem logischen Switch des Knotens für *br-int* verbunden ist.

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

Führen Sie den folgenden Befehl aus, um anzuzeigen, was `ofport` ist, weil OVS einen verfügbaren Port zuweist, wenn `ofport 1` nicht zur Verfügung steht.

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

Wenn `ofport` nicht 1 ist, legen Sie die Option `ovs_uplink_port` im `nsx_kube_proxy`-Abschnitt der `DaemonSet-YAML-Datei` des NSX-Knotenagenten fest.

- 7 Stellen Sie sicher, dass *br-int* und *node-if link* einen aktiven Status aufweisen.

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

- 8 Aktualisieren Sie die Netzwerkkonfigurationsdatei, um sicherzustellen, dass die Netzwerkschnittstelle nach einem Neustart aktiv ist.

Aktualisieren Sie für Ubuntu den Dateipfad `/etc/network/interfaces`, und fügen Sie die folgenden Zeilen hinzu:

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

Aktualisieren Sie für RHEL den Dateipfad `/etc/sysconfig/network-scripts/ifcfg-<node-if>`, und fügen Sie die folgende Zeile hinzu:

```
ONBOOT=yes
```

Konfigurieren von NSX-T Data Center-Netzwerken für Kubernetes-Knoten

In diesem Abschnitt wird die Konfiguration von NSX-T Data Center-Netzwerken für Kubernetes Master- und Worker-Knoten beschrieben.

Jeder Knoten muss über mindestens zwei Netzwerkschnittstellen verfügen. Die erste ist eine Verwaltungsschnittstelle, die zur NSX-T Data Center-Fabric gehören kann oder nicht. Die anderen Schnittstellen stellen das Netzwerk für die Pods bereit, gehören zur NSX-T Data Center-Fabric und sind mit einem logischen Switch verbunden, der als logischer Switch-Knoten bezeichnet wird. Die IP-Adressen für Verwaltung und Pod müssen routungsfähig sein, damit die Kubernetes-Systemdiagnose funktioniert. Für die Kommunikation zwischen der Verwaltungsschnittstelle und den Pods erstellt NCP automatisch eine DFW-Regel, damit die Systemdiagnose und anderweitiger Datenverkehr für die Verwaltung zulässig ist. Details zu dieser Regel finden Sie auf der NSX Manager-GUI. Diese Regel darf nicht geändert oder gelöscht werden.

Stellen Sie für jeden VM-Knoten sicher, dass die virtuelle Netzwerkkarte, die für das Containernetzwerk vorgesehen ist, mit dem logischen Knoten-Switch verbunden ist.

Die VIF-ID der für den Containerdatenverkehr in den einzelnen Knoten verwendeten virtuellen Netzwerkkarte muss NSX Container Plug-in (NCP) bekannt sein. Die entsprechenden Ports der logischen Switches müssen die folgenden Tags aufweisen: Geben Sie bei einem Tag den Namen des Knotens an. Für das andere Tag geben Sie den Namen des Clusters an. Geben Sie für den Geltungsbereich den entsprechenden Wert wie unten angegeben an.

Tag	Geltungsbereich
Knotenname	<code>ncp/node_name</code>
Clustername	<code>ncp/cluster</code>

Sie können den logischen Switch-Port für eine virtuelle Maschine eines Knotens bestimmen, indem Sie auf der NSX Manager-GUI zu **Bestand > Virtuelle Maschinen** navigieren.

Wenn der Kubernetes-Knotenname geändert wird, müssen Sie das Tag `ncp/node_name` aktualisieren und NCP neu starten. Mithilfe des folgenden Befehls können Sie die Knotennamen abrufen:

```
kubect1 get nodes
```

Wenn Sie einem Cluster einen Knoten hinzufügen, während NCP ausgeführt wird, müssen Sie die Tags dem logischen Switch-Port hinzufügen, bevor Sie den `kubeadm join`-Befehl ausführen. Andernfalls wird der neue Knoten nicht über Netzwerkkonnektivität verfügen. Wenn die Tags falsch oder nicht vorhanden sind, können Sie die folgenden Schritte ausführen, um das Problem zu beheben:

- Wenden Sie auf dem logischen Switch-Port die richtigen Tags an.
- Starten Sie NCP neu.

Installieren des NSX-Knotenagenten

Der NSX-Knotenagent ist ein DaemonSet, wobei von jedem Pod zwei Container ausgeführt werden. Ein Container führt den NSX-Knotenagent aus, der hauptsächlich für die Verwaltung der Container-Netzwerkschnittstellen zuständig ist. Er interagiert mit dem CNI-Plugin und dem Kubernetes-API-Server. Der andere Container führt den NSX-Kube-Proxy aus, der lediglich der Implementierung der Kubernetes-Dienstabstraktion dient, indem Cluster-IP-Adressen in Pod-IP-Adressen übersetzt werden. Er implementiert dieselbe Funktionalität wie der vorgeschaltete Kube-Proxy.

Verfahren

- 1 Laden Sie das NCP-Docker-Image herunter.

Der Dateiname lautet `nsx-ncp-xxxxxxx.tar`, wobei `xxxxxxx` der Build-Nummer entspricht.

- 2 Laden Sie die YAML-Datei für das DaemonSet des NSX-Knotenagents herunter.

Der Dateiname ist `nsx-node-agent-ds.yml`. Sie können diese Datei bearbeiten oder als Beispiel für Ihre eigene Vorlagendatei verwenden.

- 3 Laden Sie das NCP-Docker-Image in Ihre Image-Registrierung.

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

- 4 Bearbeiten Sie `nsx-node-agent-ds.yml`.

Ändern Sie den Image-Namen in den Namen der geladenen Datei.

Für Ubuntu geht die YAML-Datei davon aus, dass AppArmor aktiviert ist. Um zu prüfen, ob AppArmor aktiviert ist, überprüfen Sie die Datei `/sys/module/apparmor/parameters/enabled`. Wenn AppArmor nicht aktiviert ist, nehmen Sie die folgenden Änderungen vor:

- Löschen Sie die folgende Zeile, oder kommentieren Sie sie aus:

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

- Fügen Sie die Zeile `privileged:true` unter `securityContext` für die Container „nsx-node-agent“ und „nsx-kube-proxy“ hinzu. Beispiel:

```
securityContext:
  privileged:true
```

Hinweis Folgendes Problem ist bekannt: Wenn Kubelet innerhalb eines Containers ausgeführt wird, das das Hyperkube-Image verwendet, meldet Kubelet AppArmor unabhängig vom Ist-Zustand immer als deaktiviert. Sie müssen Sie dieselben oben genannten Änderungen an der YAML-Datei vornehmen.

Hinweis In der YAML-Datei müssen Sie angeben, dass das für `ncp.ini` generierte ConfigMap-Objekt als Read-Only-Volume bereitgestellt werden muss. Die heruntergeladene YAML-Datei weist bereits diese Spezifikation auf, die nicht geändert werden sollte.

- 5 Erstellen Sie das NSX-Knotenagent-DaemonSet mit dem folgenden Befehl.

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

Configmap für `ncp.ini` in `nsx-node-agent-ds.yml`

Die yaml-Beispieldatei `nsx-node-agent-ds.yml` enthält ein ConfigMap-Objekt für die Konfigurationsdatei `ncp.ini` für den NSX-Knoten-Agent. Dieser ConfigMap-Abschnitt enthält Parameter, die Sie angeben können, um die Knoten-Agent-Installation anzupassen.

Die von Ihnen heruntergeladene Beispieldatei `nsx-node-agent-ds.yml` enthält die folgenden `ncp.ini`-Informationen:

```
# 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>
```

Installieren von NSX Container Plug-in

NSX Container Plug-in (NCP) wird als Docker-Image bereitgestellt. NCP (NSX Container-Plug-In) sollte auf einem Knoten für Infrastrukturdienste ausgeführt werden. Die NCP-Ausführung auf dem Master-Knoten wird nicht empfohlen.

Verfahren

- 1 Laden Sie das NCP-Docker-Image herunter.

Der Dateiname lautet `nsx-ncp-xxxxxxx.tar`, wobei `xxxxxxx` der Build-Nummer entspricht.

- 2 Laden Sie die YAML-Vorlage für NCP ReplicationController herunter.

Der Dateiname lautet `ncp-rc.yaml`. Sie können diese Datei bearbeiten oder als Beispiel für Ihre eigene Vorlagendatei verwenden.

- 3 Laden Sie das NCP-Docker-Image in Ihre Image-Registrierung.

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

- 4 (Optional) Laden Sie die YAML-Vorlage für die benutzerdefinierte Ressourcendefinition des NSXError-Objekts herunter.

Der Dateiname lautet `nsx-error-crd.yaml`.

- 5 (Optional) Erstellen Sie die benutzerdefinierte Ressource.

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

- 6 Bearbeiten Sie `ncp-rc.yaml`.

Ändern Sie den Image-Namen in den Namen der geladenen Datei.

Geben Sie den `nsx_api_managers`-Parameter an. Sie können die IP-Adresse eines einzelnen NSX Managers, die IP-Adressen (getrennt durch Komma) der drei NSX Manager in einem NSX Manager-Cluster oder die virtuelle IP-Adresse eines NSX Manager-Clusters angeben. Beispiel:

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

(Optional) Geben Sie den Parameter `ca_file` im Abschnitt `[nsx_v3]` an. Der Wert sollte einer CA-Paketdatei entsprechen, die beim Überprüfen des NSX Manager-Serverzertifikats verwendet wird. Wenn Sie keine Festlegung treffen, werden die System-Root-CAs verwendet. Wenn Sie eine IP-Adresse für `nsx_api_managers` angeben, geben Sie eine Zertifizierungsstellendatei an. Wenn Sie drei IP-Adressen für `nsx_api_managers` angeben, können Sie eine oder drei Zertifizierungsstellendatei(en) angeben. Wenn Sie eine Zertifizierungsstellendatei angeben, wird sie für alle drei Manager verwendet. Wenn Sie drei Zertifizierungsstellendateien angeben, wird jede davon für die entsprechende IP-Adresse in `nsx_api_managers` verwendet. Beispiel:

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

Geben Sie die Parameter `nsx_api_cert_file` und `nsx_api_private_key_file` für die Authentifizierung mit NSX-T Data Center an.

`nsx_api_cert_file` ist der vollständige Pfad zu einer Client-Zertifikatdatei im PEM-Format. Der Inhalt dieser Datei sollte wie folgt aussehen:

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

„`nsx_api_private_key_file`“ ist der vollständige Pfad zu einer privaten Client-Schlüsseldatei im PEM-Format. Der Inhalt dieser Datei sollte wie folgt aussehen:

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

Geben Sie den Parameter `ingress_mode = nat` an, wenn der Ingress-Controller für die Ausführung im NAT-Modus konfiguriert ist.

Standardmäßig wird das Subnetzpräfix 24 für alle Subnetze verwendet, die von den IP-Blöcken für die logischen Pod-Switches zugeordnet sind. Wenn Sie eine andere Subnetzgröße verwenden möchten, aktualisieren Sie die `subnet_prefix`-Option im Abschnitt `[nsx_v3]`.

HA (Hochverfügbarkeit) ist automatisch aktiviert. Sie können HA mit der folgenden Spezifikation deaktivieren:

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

(Optional) Aktivieren Sie Fehlerberichterstellung mithilfe von NSXError in `ncp.ini`. Diese Einstellung ist standardmäßig deaktiviert.

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

Hinweis In der YAML-Datei müssen Sie angeben, dass das für `ncp.ini` generierte ConfigMap-Objekt als ReadOnly-Volume bereitgestellt wird. Die heruntergeladene YAML-Datei weist bereits diese Spezifikation auf, die nicht geändert werden sollte.

7 Erstellen Sie den NCP ReplicationController.

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

Ergebnisse

Hinweis NCP öffnet dauerhafte HTTP-Verbindungen zum Kubernetes-API-Server, um Lebenszyklusereignisse von Kubernetes-Ressourcen zu überwachen. Wenn ein API-Serverausfall oder ein Netzwerkausfall dazu führt, dass TCP-Verbindungen von NCP verfallen, müssen Sie NCP neu starten, sodass es die Verbindungen zum API-Server wiederherstellen kann. Andernfalls verpasst NCP die neuen Ereignisse.

Während eines fortlaufenden Updates des NCP ReplicationController werden möglicherweise nach dem rollierenden Update zwei NCP-Pods angezeigt, die ausgeführt werden:

- Während des rollierenden Updates starten Sie den Container-Host neu.
- Das rollierende Update schlägt zu Beginn fehl, weil das neue Image auf keinem Kubernetes-Knoten vorhanden ist. Laden Sie das Image herunter und führen Sie das rollierende Update erneut aus, und es verläuft erfolgreich.

Wenn Sie zwei NCP-Pods angezeigt bekommen, die ausgeführt werden, gehen Sie wie folgt vor:

- Löschen Sie einen der NCP-Pods. Dabei spielt es keine Rolle, welchen der beiden Pods Sie löschen. Beispiel:

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

- Löschen Sie den NCP ReplicationController. Beispiel:

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

Configmap für ncp.ini in ncp-rc.yml

Die YAML-Beispieldatei `ncp-rc.yml` enthält ein ConfigMap-Objekt für die Konfigurationsdatei `ncp.ini`. Dieser ConfigMap-Abschnitt enthält Parameter, die Sie angeben müssen, bevor Sie NCP wie im vorherigen Abschnitt beschrieben installieren.

Die von Ihnen heruntergeladene Beispieldatei `ncp-rc.yml` enthält die folgenden `ncp.ini`-Informationen:

```
# 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>
```

Bereitstellen eines PEM-verschlüsselten Zertifikats und privaten Schlüssels im NCP-Pod

Wenn Sie über ein PEM-kodiertes Zertifikat und einen privaten Schlüssel verfügen, können Sie die NCP-Pod-Definition in der YAML-Datei aktualisieren, um die geheimen TLS-Schlüssel im NCP-Pod bereitzustellen.

- 1 Erstellen Sie einen geheimen TLS-Schlüssel für das Zertifikat und den privaten Schlüssel.

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

- 2 Aktualisieren Sie die YAML-Datei der NCP-Pod-Spezifikation, damit der geheime Schlüssel in der NCP-Pod-Spezifikation in Form von Dateien bereitgestellt wird.

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

- 3 Aktualisieren Sie die nsx_v3-Optionen `nsx_api_cert_file` und `nsx_api_private_key_file` in der YAML-Datei.

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

Bereitstellen einer Zertifikatdatei im NCP-Pod

Wenn Sie über eine Zertifikatdatei im Knotendateisystem verfügen, können Sie die NCP-Pod-Spezifikation aktualisieren, um die Datei im NCP-Pod bereitzustellen.

Beispiel:

```
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>
```

Konfigurieren von Syslog

Sie können einen Syslog-Agenten, wie z. B. Rsyslog oder Syslog-ng in einem Container ausführen, um Protokolle aus NCP und den zugehörigen Komponenten an einen Syslog-Server zu senden.

Die folgenden Methoden werden empfohlen. Weitere Informationen zur Protokollierung in Kubernetes finden Sie unter <https://kubernetes.io/docs/concepts/cluster-administration/logging>.

- Erstellen Sie einen Sidecar-Container, der im NCP- oder die NSX-Knoten-Agent-Pod ausgeführt wird.
- Führen Sie auf jedem Knoten ein DaemonSet-Replikat aus.

Hinweis Mit der Sidecar-Containermethode können NSX CNI-Plug-In-Protokolle nicht an den Syslog-Server gesendet werden, da das Plugin nicht in einem Pod ausgeführt wird.

Erstellen eines Sidcar-Containers für Syslog

Sie können einen Sidecar-Container für Syslog für die Ausführung in demselben Pod wie NCP konfigurieren. Bei dem folgenden Verfahren wird davon ausgegangen, dass das Syslog-Agent-Image „example/rsyslog“ ist.

Verfahren

- 1 Konfigurieren Sie NCP und den NSX-Knoten-Agent für die Protokollierung in einer Datei.

Legen Sie in der yaml-Datei für NCP und den NSX-Knoten-Agent den Parameter „log_dir“ fest, und geben Sie bereitzustellende Volume an. Beispiel:

```
[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
```

Sie können den Namen der Protokolldatei ändern, indem Sie den `log_file`-Parameter festlegen. Die Standardnamen sind `ncp.log`, `nsx_node_agent.log` und `nsx_kube_proxy.log`. Wenn die `log_dir`-Option auf einen anderen Pfad als `/var/log/nsx-ujo` festgelegt ist, muss entweder ein `hostPath`- oder `emptyDir`-Volume erstellt und für die entsprechende Pod-Spezifikation bereitgestellt werden.

- 2 Stellen Sie sicher, dass der Host-Pfad existiert und vom Benutzer `nsx-ncp` beschreibbar ist.
 - a Führen Sie die folgenden Befehle aus.

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

- b Fügen Sie den Benutzer `nsx-ncp` hinzu oder ändern Sie den Modus des Hostpfads auf `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** Fügen Sie in der yaml-Spezifikationsdatei des NCP-Pods ein ConfigMap-Objekt für Syslog hinzu. Beispiel:

```
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** Fügen Sie in der yaml-Datei des NCP Pods den rsyslog-Container hinzu, und stellen Sie die entsprechenden Volumes bereit, auf denen Rsyslog Konfigurationsdaten finden und Protokolle von anderen Containern lesen kann. Beispiel:

```
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>
```

Erstellen eines DaemonSet-Replikats für Syslog

Mithilfe dieser Methode können die Protokolle aller NCP-Komponenten umgeleitet werden. Die Anwendungen müssen für die Protokollierung in stderr konfiguriert werden. Dies ist standardmäßig aktiviert. Bei dem folgenden Verfahren wird davon ausgegangen, dass das Syslog-Agent-Image „example/rsyslog“ ist.

Verfahren

- 1 Erstellen Sie die yaml-Datei für DaemonSet. Beispiel:

```
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 Erstellen Sie das DaemonSet-Objekt.

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

Beispiel: Konfigurieren der Protokollrotation und des in einem Sidecar-Container ausgeführten Syslogs

Im folgenden Verfahren wird die Konfiguration der Protokollrotation und des in einem Sidecar-Container ausgeführten Syslogs gezeigt.

Erstellen des Protokollverzeichnisses und Konfigurieren der Protokollrotation

- Erstellen Sie das Protokollverzeichnis auf allen Knoten, einschließlich des Masterknotens, und ändern Sie seinen Besitzer in den Benutzer mit der ID 1000.

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

- Konfigurieren Sie die Protokollrotation auf allen Knoten für das Verzeichnis /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

```

Erstellen des NCP ReplicationController

- Erstellen Sie die Datei `ncp.ini` für 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

```

- Erstellen Sie die Konfigurationszuordnung aus der INI-Datei.

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

- Erstellen Sie die rsyslog-Konfiguration für 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
```

- Erstellen Sie die Konfigurationszuordnung aus dem Obenstehenden.

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

- Erstellen Sie den NCP ReplicationController mit dem 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

- Erstellen Sie NCP mit der oben angegebenen Spezifikation.

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

Erstellen des DaemonSet des NSX-Knotenagents

- Erstellen Sie die rsyslog-Konfiguration für die Knotenagents.

```
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
```

- Erstellen Sie die configmap aus dem Obenstehenden.

```
kubect1 create -f /tmp/nsx-node-agent-rsyslog.conf
```

- Erstellen Sie das DaemonSet mit dem 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

```

- Erstellen Sie das DaemonSet-Objekt.

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

Sicherheitsüberlegungen

Beim Bereitstellen von NCP ist es wichtig, Schritte zum Sichern der Kubernetes- und der NSX-T Data Center-Umgebungen auszuführen.

Beschränken der NCP-Ausführung auf bestimmte Knoten

NCP hat Zugriff auf die NSX-T Data Center-Managementebene und die Ausführung sollte auf bestimmte Infrastrukturknoten beschränkt werden. Sie können diese Knoten mit einer entsprechenden Kennzeichnung identifizieren. Anschließend sollte ein nodeSelector-Objekt für diese Kennzeichnung auf die NCP ReplicationController-Spezifikation angewendet werden.

Beispiel:

```

nodeSelector:
  nsx-infra: True

```

Sie können auch andere Mechanismen wie die Affinität verwenden, um Knoten Pods zuzuweisen. Weitere Informationen finden Sie unter <https://kubernetes.io/docs/concepts/configuration/assign-pod-node>.

Stellen Sie sicher, dass die Docker-Engine aktuell ist

Docker veröffentlicht regelmäßig Sicherheits-Updates. Ein automatisierter Vorgang sollte implementiert werden, um diese Updates anzuwenden.

Nichtzulassen von NET_ADMIN- und NET_RAW-Funktionen nicht vertrauenswürdiger Container

Die Linux-Funktionen NET_ADMIN und NET_RAW können von Angreifern ausgenutzt werden, um das Pod-Netzwerk zu gefährden. Sie sollten diese beiden Funktionen der nicht vertrauenswürdigen Container deaktivieren. Standardmäßig wird die NET_ADMIN-Funktion auf einem nicht Container ohne Berechtigungen nicht gewährt. Seien Sie vorsichtig, wenn sie von

einer Pod-Spezifikation explizit aktiviert oder der Container auf den privilegierten Modus festgelegt wird. Deaktivieren Sie darüber hinaus für nicht vertrauenswürdigen Container NET_RAW, indem Sie NET_RAW in der Liste der verworfenen Funktionen in der SecurityContext-Konfiguration der Container-Spezifikation angeben. Beispiel:

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

Rollenbasierte Zugriffssteuerung

Kubernetes verwendet die APIs für die rollenbasierte Zugriffssteuerung (RBAC) für Autorisierungsfestlegungen und ermöglicht Administratoren damit die dynamische Konfiguration von Richtlinien. Weitere Informationen finden Sie unter <https://kubernetes.io/docs/admin/authorization/rbac>.

Normalerweise ist der Cluster-Administrator der einzige Benutzer mit privilegiertem Zugriff und privilegierten Rollen. Für Benutzer und Dienstkonten muss beim Gewähren des Zugriffs das Prinzip der geringsten Berechtigung befolgt werden.

Die folgenden Richtlinien werden empfohlen:

- Beschränken Sie den Zugriffs auf Kubernetes-API-Token auf Pods, die sie benötigen.
- Beschränken Sie den Zugriff auf die geheimen TLS-Schlüssel des NCP ConfigMap- und NSX-API-Clientzertifikats auf den NCP-Pod.
- Blockieren Sie den Zugriff auf Kubernetes-Netzwerk-API von Pods, die diesen Zugriff nicht benötigen.
- Fügen Sie eine Kubernetes RBAC-Richtlinie hinzu, um anzugeben, welche Pods auf die Kubernetes-API zugreifen können.

Empfohlene RBAC-Richtlinie für den NCP-Pod

Erstellen Sie den NCP-Pod unter einem Dienstkonto, und weisen Sie diesem Konto einen minimale Berechtigungen zu. Gewähren Sie darüber hinaus anderen Pods oder ReplicationController-Objekten keinen Zugriff auf die geheimen ConfigMap- und TLS-Schlüssel, die als Volumes für den NCP ReplicationController- und NSX-Knotenagent bereitgestellt wurden.

Das folgende Beispiel zeigt, wie Sie Rollen und Rollenbindungen für NCP angeben:

```
# 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

```

Hinweis Der geheime TLS-Schlüssel, der mithilfe der Kubernetes-API für das NSX-T Data Center-Clientzertifikat erstellt wird, und das private Schlüsselpaar sind für Pods zugänglich, die Zugriff auf den Kubernetes-API-Server haben. Wird ein Pod ohne Dienstkonto erstellt, wird ihm gleichermaßen automatisch das Standard-Dienstkonto zugewiesen, das den Token automatisch für den Zugriff auf die Kubernetes-API bereitstellt. Aus diesem Grund muss der Zugriff auf diese Token auf Pods beschränkt werden, die sie benötigen.

Tipps zum Konfigurieren von Netzwerkressourcen

Beim Konfigurieren einiger Netzwerkressourcen sollten Sie sich bestimmter Einschränkungen bewusst sein.

Tagging-Grenzwerte in NSX-T Data Center

NSX-T Data Center weist folgende Grenzwerte für das Versehen von Objekten mit Tags auf:

- Der Bereich hat eine Obergrenze von 128 Zeichen.
- Das Tag hat eine Obergrenze von 256 Zeichen.
- Jedes Objekt kann maximal 30 Tags aufweisen.

Diese Grenzwerte können zu Problemen führen, wenn Kubernetes- oder OpenShift-Anmerkungen in NSX-T Data Center-Bereiche und -Tags kopiert und die Grenzwerte überschritten werden.

Wenn ein Tag beispielsweise für einen Switch-Port bestimmt ist und das Tag in einer Firewallregel verwendet wird, wird die Regel möglicherweise nicht erwartungsgemäß angewendet, da der Anmerkungschlüssel oder -wert beim Kopieren in einen Bereich oder ein Tag möglicherweise abgeschnitten wurde.

Konfigurieren von Netzwerkrichtlinien

Netzwerkrichtlinien wählen Pods oder Namespaces mithilfe von Bezeichnungsselektoren aus.

Die NCP-Unterstützung für Netzwerkrichtlinien entspricht der von Kubernetes bereitgestellten Unterstützung und richtet sich nach der Kubernetes-Version.

- Kubernetes 1.11 – Sie können folgende Regelselektoren angeben:
 - `podSelector`: Damit werden alle Pods ausgewählt, die sich in dem Namespace befinden, in dem die Netzwerkrichtlinie erstellt wird.
 - `namespaceSelector`: Damit werden alle Namespaces ausgewählt.
 - `podSelector` UND `namespaceSelector`: Damit werden alle Pods ausgewählt, die sich in den von `namespaceSelector` ausgewählten Namespaces befinden.
 - `ipBlockSelector`: Eine Netzwerkrichtlinie ist ungültig, wenn `ipBlockSelector` mit `namespaceSelector` oder `podSelector` kombiniert wird. Ein `ipBlockSelector` muss in der Richtlinienspezifikation allein vorhanden sein.
- Kubernetes 1.10 – Die Regelklauseln in der Netzwerkrichtlinie können höchstens einen der Selektoren `namespaceSelector`, `podSelector` und `ipBlock` enthalten.

Der Kubernetes-API-Server unterzieht eine Netzwerkrichtlinienspezifikation keiner Validierung. Es ist daher möglich, eine Netzwerkrichtlinie zu erstellen, die ungültig ist. NCP lehnt ungültige Netzwerkrichtlinien ab. Falls Sie die Netzwerkrichtlinie aktualisieren, damit sie gültig ist, wird sie von NCP trotzdem nicht verarbeitet. Sie müssen die Netzwerkrichtlinie löschen und mit einer gültigen Spezifikation neu erstellen.

Installieren von NCP in einer Pivotal Cloud Foundry(PCF)-Umgebung

Pivotal Cloud Foundry (PCF) ist ein Open Source-PaaS-Anbieter (Platform-as-a-Service). Sie können das NSX Container Plug-in (NCP) in einer PCF-Umgebung installieren, um Netzwerkdienste bereitzustellen.

Über den Pivotal Ops Manager erstellte VMs müssen Schicht-3-Konnektivität zum Containernetzwerk aufweisen, damit sie auf die NSX-T-Funktionen zugreifen können.

Hochverfügbarkeit (HA) ist automatisch aktiviert.

Hinweis Wenn eine Änderung an einer Sicherheitsgruppe vorgenommen wird, müssen alle Anwendungen, für die die Sicherheitsgruppe gilt, erneut bereitgestellt werden. Dies kann passieren, weil die Sicherheitsgruppe für den Speicherbereich gilt, in dem die Anwendungen ausgeführt werden, oder weil es sich um eine globale Sicherheitsgruppe handelt.

Dieses Kapitel enthält die folgenden Themen:

- [Installieren von NCP in einer Pivotal Cloud Foundry\(PCF\)-Umgebung](#)

Installieren von NCP in einer Pivotal Cloud Foundry(PCF)-Umgebung

NCP wird über die grafische Pivotal Ops Manager-Benutzeroberfläche installiert.

Voraussetzungen

Eine Neuinstallation von Pivotal Ops Manager, NSX-T Data Center und Pivotal Application Service (PAS). Stellen Sie sicher, dass Ops Manager zuerst, dann NSX-T Data Center und dann PAS installiert wird. Weitere Informationen finden Sie in der Pivotal Cloud Foundry-Dokumentation.

Verfahren

- 1 Laden Sie die NCP-Installationsdatei für PCF herunter.
Der Dateiname lautet `VMware-NSX-T.<version>.<build>.pivotal`.
- 2 Melden Sie sich bei Pivotal Ops Manager als Administrator an.
- 3 Klicken Sie auf **Produkt importieren**.

- 4 Wählen Sie die heruntergeladene Datei aus.
- 5 Klicken Sie auf die Kachel **Ops Manager Director für VMware vSphere**.
- 6 Wählen Sie auf der Registerkarte **Einstellungen** für **vCenter-Konfiguration** die Option **NSX-Netzwerk** und für **NSX-Modus** die Option **NSX-T** aus.
- 7 Stellen Sie im Feld **NSX-CA-Zertifikat** das Zertifikat im PEM-Format bereit.
- 8 Klicken Sie auf **Speichern**.
- 9 Klicken Sie in der oberen linken Ecke auf **Installations-Dashboard**, um zum Dashboard zurückzukehren.
- 10 Klicken Sie auf die Kachel **Pivotal Application Service**.
- 11 Wählen Sie auf der Registerkarte **Einstellungen** im Navigationsbereich die Option **Netzwerk** aus.
- 12 Wählen Sie unter **Container-Netzwerkschnittstellen-Plug-In** die Option **Extern** aus.
- 13 Klicken Sie in der oberen linken Ecke auf **Installations-Dashboard**, um zum Dashboard zurückzukehren.
- 14 Klicken Sie auf **Speichern**.
- 15 Klicken Sie in der oberen linken Ecke auf **Installations-Dashboard**, um zum Dashboard zurückzukehren.
- 16 Klicken Sie auf die Kachel **VMware NSX-T**.
- 17 Geben Sie die Adresse des NSX Manager an.
- 18 Wählen Sie die Methode für die NSX Manager-Authentifizierung aus.

Option	Aktion
Clientzertifikat-Authentifizierung	Stellen Sie das Zertifikat und den privaten Schlüssel für NSX Manager bereit.
Standardauthentifizierung mit Benutzername und Kennwort	Geben Sie den Benutzernamen und das Kennwort eines NSX Manager-Administrators ein.

- 19 Stellen Sie im Feld **NSX Manager-CA-Zertifikat** das Zertifikat bereit.
- 20 Klicken Sie auf **Speichern**.
- 21 Wählen Sie im Navigationsbereich **NCP** aus.
- 22 Nehmen Sie eine Eingabe unter **PAS Foundation-Name** vor.
Diese Zeichenfolge dient der eindeutigen Identifizierung einer PAS Foundation in der NSX API. Diese Zeichenfolge wird auch in Namen von NSX-Ressourcen, die von NCP für die PAS Foundation erstellt wurden, als Präfix verwendet.
- 23 Nehmen Sie eine Eingabe unter **Overlay-Transportzone** vor.
- 24 Geben Sie den **Tier-0-Router** ein.

- 25** Geben Sie unter **IP-Blöcke von Containernetzwerken** mindestens einen IP-Block an.
- Klicken Sie auf **Hinzufügen**.
 - Nehmen Sie eine Eingabe unter **Name des IP-Blocks** vor. Hierbei kann es sich um einen neuen oder vorhandenen IP-Block handeln.
 - Geben Sie einen neuen IP-Block im CIDR-Format ein, z. B. 10.1.0.0/16.
- 26** Geben Sie das Subnetzpräfix der Containernetzwerke an.
- 27** Klicken Sie auf **SNAT für Containernetzwerke aktivieren**, um SNAT zu aktivieren.
- 28** Geben Sie unter **IP-Pools, die zum Bereitstellen externer IP-Adressen (NAT) für Organisationsnetzwerke verwendet werden** mindestens einen IP-Pool an.
- Klicken Sie auf **Hinzufügen**.
 - Nehmen Sie eine Eingabe unter **Name des IP-Pools** vor. Hierbei kann es sich um einen neuen oder vorhandenen IP-Pool handeln.
 - Geben Sie nur für einen neuen IP-Pool die IP-Adressen an, indem Sie die CIDR- und IP-Bereiche bereitstellen.
- 29** (Optional) Nehmen Sie eine Eingabe unter **Obere Markierung für Firewallabschnitt** vor.
- 30** (Optional) Nehmen Sie eine Eingabe unter **Untere Markierung für Firewallabschnitt** vor.
- 31** (Optional) Aktivieren oder deaktivieren Sie die folgenden Optionen.

Option	Standardwert
Verloren gegangenen Anwendungsdatenverkehr protokollieren	Deaktiviert. Ist diese Option aktiviert, wird aufgrund einer Firewallregel verloren gegangener Datenverkehr protokolliert.
Debugebene für NCP-Protokollierung aktivieren	Aktiviert.

- 32** Klicken Sie auf **Speichern**.
- 33** (Optional) Wählen Sie im Navigationsbereich **NSX-Knotenagent** aus.
- Aktivieren Sie die Option **Debugebene der Protokollierung für NSX-Knotenagent aktivieren**, um die Protokollierung mit Debugebene zu aktivieren.
 - Klicken Sie auf **Speichern**.
- 34** Klicken Sie in der oberen linken Ecke auf **Installations-Dashboard**, um zum Dashboard zurückzukehren.
- 35** Klicken Sie auf **Änderungen übernehmen**.

Load Balancing

5

Der Load Balancer von NSX-T Data Center ist in Kubernetes integriert.

Dieses Kapitel enthält die folgenden Themen:

- [Konfigurieren von Load Balancing](#)
- [Ingress-Controller von Drittanbietern](#)

Konfigurieren von Load Balancing

Das Konfigurieren des Load Balancing umfasst das Konfigurieren eines Kubernetes-LoadBalancer-Diensts oder einer -Ingress-Ressource und des NCP ReplicationController.

Sie können einen Load Balancer auf Schicht 4 erstellen, indem Sie einen Kubernetes-Dienst des Typs LoadBalancer konfigurieren. Einen Load Balancer auf Schicht 7 erstellen Sie, indem Sie eine Kubernetes-Ingress-Ressource konfigurieren.

Gehen Sie zum Konfigurieren von Load Balancing in NCP in der Datei `ncp-rc.yml` wie folgt vor:

- 1 Legen Sie `use_native_loadbalancer = True` fest.
- 2 (Optional) Legen Sie `pool_algorithm` auf **ROUND_ROBIN** oder **LEAST_CONNECTION/IP_HASH** fest. Die Standardeinstellung ist **ROUND_ROBIN**.
- 3 (Optional) Legen Sie `service_size = SMALL, MEDIUM oder LARGE` fest. Die Standardeinstellung ist **SMALL**.

Der Algorithmus **LEAST_CONNECTION/IP_HASH** bedeutet, dass Datenverkehr von derselben Quell-IP-Adresse zum selben Backend-Pod gesendet wird.

Weitere Informationen dazu, was von NSX-T-Load Balancern unterschiedlicher Größe unterstützt wird, finden Sie im *Administratorhandbuch für NSX-T Data Center*.

Nachdem der Load Balancer erstellt wurde, kann seine Größe nicht durch Aktualisierung der Konfigurationsdatei geändert werden. Die Größe kann über die NSX Manager-Benutzeroberfläche oder -API geändert werden.

Festlegen der Persistenz für Load Balancing auf Schicht 4 und Schicht 7

Mit den Parametern `l4_persistence` und `l7_persistence` können Sie im NCP-ConfigMap-Objekt eine Persistenzeinstellung angeben. Die verfügbare Option für die Schicht-4-Persistenz ist „Quell-IP“. Die verfügbaren Optionen für die Schicht-7-Persistenz sind „Cookie“ und „Quell-IP“. Die Standardeinstellung ist `<None>`. Beispiel:

```
# 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
```

Für einen Kubernetes-LoadBalancer-Dienst können Sie `sessionAffinity` auch in der Dienstspezifikation angeben, um das Persistenzverhalten für den Dienst zu konfigurieren, wenn die globale Schicht-4-Persistenz deaktiviert, also `l4_persistence` auf `<None>` festgelegt ist. Wenn `l4_persistence` auf `source_ip` festgelegt ist, kann die `sessionAffinity` auf der Dienstspezifikation verwendet werden, um die Persistenz-Zeitüberschreitung für den Dienst anzupassen. Die standardmäßige Persistenz-Zeitüberschreitung von Layer 4 beträgt 10.800 Sekunden (wie in der Kubernetes-Dokumentation für Dienste (<https://kubernetes.io/docs/>

[concepts/services-networking/service](#)) angegeben. Alle Dienste mit standardmäßiger Persistenz-Zeitüberschreitung nutzen das gleiche Persistenz-Profil des NSX-T-Load Balancer. Für jeden Dienst mit einer nicht standardmäßigen Persistenz-Zeitüberschreitung wird ein dediziertes Profil erstellt.

Hinweis Wenn der Backend-Dienst eines Ingress ein Dienst des Typs LoadBalancer ist, dürfen der virtuelle Server der Schicht 4 für den Dienst und der virtuelle Server der Schicht 7 für den Ingress nicht unterschiedliche Persistenzeinstellungen aufweisen, beispielsweise `source_ip` für Schicht 4 und `cookie` für Schicht 7. In einem Szenario dieser Art müssen die Persistenzeinstellungen für beide virtuelle Server identisch sein (`source_ip`, `cookie` oder `None`), oder einer davon hat die Einstellung `None` (in diesem Fall kann die andere Einstellung `source_ip` oder `cookie` lauten). Beispiel für ein Szenario dieser Art:

```
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
```

Ingress

- NSX-T Data Center erstellt einen Load Balancer auf Schicht 7 für Ingresses mit TLS-Spezifikation und einen Load Balancer auf Schicht 7 für Ingresses ohne TLS-Spezifikation.
- Alle Ingresses erhalten eine einzelne IP-Adresse.

- Der Ingress-Ressource wird eine IP-Adresse aus dem externen IP-Pool zugeteilt, der durch die Option `external_ip_pools` im Abschnitt `[nsx_v3]` in `ncp.ini` angegeben ist. Der Load Balancer wird unter dieser IP-Adresse und auf den HTTP- und HTTPS-Ports (80 und 443) bereitgestellt.
- Der Ingress-Ressource wird eine IP-Adresse aus dem externen IP-Pool zugeteilt, der durch die Option `external_ip_pools_lb` im Abschnitt `[nsx_v3]` in `ncp.ini` angegeben ist. Wenn die Option `external_ip_pools_lb` nicht vorhanden ist, wird der von `external_ip_pools` angegebene Pool verwendet. Der Load Balancer wird unter dieser IP-Adresse und auf den HTTP- und HTTPS-Ports (80 und 443) bereitgestellt.
- Sie können zu einem anderen IP-Pool wechseln, indem Sie die Konfiguration ändern und NCP neu starten.
- Sie können ein Standardzertifikat für TLS angeben. Informationen zum Erstellen eines Zertifikats sowie zum Mounten eines Zertifikats in den NCP-Pod finden Sie unten.
- Ingresses ohne TLS-Spezifikation werden auf dem virtuellen HTTP-Server (Port 80) gehostet.
- Ingresses mit TLS-Spezifikation werden auf dem virtuellen HTTPS-Server (Port 443) gehostet. Der Load Balancer fungiert als SSL-Server und beendet die SSL-Clientverbindung.
- Die Reihenfolge bei der Erstellung von geheimen Schlüsseln und Ingress spielt keine Rolle. Wenn das Objekt für den geheimen Schlüssel vorhanden ist und ein Ingress darauf verweist, wird das Zertifikat in NSX-T Data Center importiert. Falls der geheime Schlüssel oder der letzte Ingress, der auf den geheimen Schlüssel verweist, gelöscht wird, wird das dem geheimen Schlüssel entsprechende Zertifikat gelöscht.
- Eine Ingress-Änderung durch Hinzufügen oder Entfernen des TLS-Abschnitts wird nicht unterstützt. Nach dem Entfernen des `tls`-Schlüssels aus der Ingress-Spezifikation werden die Ingress-Regeln vom virtuellen HTTPS-Server (Port 443) auf den virtuellen HTTP-Server (Port 80) übertragen. Ebenso werden die Ingress-Regeln vom virtuellen HTTP-Server (Port 80) auf den virtuellen HTTPS-Server (Port 443) übertragen, wenn der `tls`-Schlüssel zur Ingress-Spezifikation hinzugefügt wird.
- Wenn in Ingress-Definitionen für einen einzelnen Cluster doppelte Regeln vorliegen, wird nur die erste Regel angewendet.
- Pro Cluster wird nur ein einzelner Ingress mit einem Standard-Backend unterstützt. Datenverkehr, der mit keiner Ingress-Regel übereinstimmt, wird an das Standard-Backend weitergeleitet.
- Wenn mehrere Ingresses mit einem Standard-Backend vorhanden sind, wird nur der erste konfiguriert. Die anderen erhalten einen Fehlerkommentar.
- Die Ermittlung übereinstimmender URIs mit Platzhaltern wird mit folgenden Zeichen für reguläre Ausdrücke unterstützt: „.“ und „*“. Beispiel: Der Pfad „/coffee/*“ stimmt mit „/coffee/“, gefolgt von null, einem oder mehreren Zeichen wie etwa „/coffee/“, „/coffee/a“, „/coffee/b“ überein, nicht jedoch mit „/coffee“, „/coffeecup“ oder „/coffeecup/a“.

Beispiel für eine Ingress-Spezifikation:

```
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
```

- Sie können das Umschreiben von URL-Anforderungen durch Hinzufügen einer Anmerkung zur Ingress-Ressource konfigurieren. Beispiel:

```
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
```

Die Pfade /tea und /coffee werden in / umgeschrieben, bevor die URL an den Backend-Dienst gesendet wird.

- Die Ingress-Anmerkung `kubernetes.io/ingress.allow-http` wird unterstützt.
 - Wenn die Anmerkung auf **false** festgelegt ist, werden nur HTTPS-Regeln erstellt.
 - Wenn die Anmerkung auf **true** festgelegt ist oder fehlt, werden HTTP-Regeln erstellt. Darüber hinaus werden HTTPS-Regeln erstellt, wenn der TLS-Abschnitt in der Ingress-Spezifikation vorhanden ist.
- Fehler werden in der Ingress-Ressource als Anmerkung dokumentiert. Der Fehlerschlüssel lautet `ncp/error.loadbalancer`, der Schlüssel für Warnungen `ncp/warning.loadbalancer`. Dies sind die möglichen Fehler und Warnungen:
 - `ncp/error.loadbalancer: DEFAULT_BACKEND_IN_USE`

Dieser Fehler weist darauf hin, dass bereits ein Ingress mit einem Standard-Back-End vorhanden ist. Der Ingress ist dann inaktiv. Für eine Gruppe von Ingresses kann nur ein Standard-Back-End mit und ohne TLS vorhanden sein. Um den Fehler zu beheben, löschen Sie den Ingress und erstellen Sie ihn mit einer korrekten Spezifikation neu.

- `ncp/warning.loadbalancer: SECRET_NOT_FOUND`

Dieser Fehler weist darauf hin, dass der in der Ingress-Spezifikation angegebene geheime Schlüssel nicht vorhanden ist. Der Ingress ist dann nur teilweise aktiv. Um den Fehler zu beheben, erstellen Sie den fehlenden geheimen Schlüssel. Beachten Sie, dass eine Warnung in der Anmerkung während des Lebenszyklus der Ingress-Ressource nicht gelöscht wird.

- `ncp/warning.loadbalancer: INVALID_INGRESS`

Dieser Fehler weist darauf hin, dass eine der folgenden Bedingungen wahr ist. Der Ingress ist dann inaktiv. Um den Fehler zu beheben, löschen Sie den Ingress und erstellen Sie ihn mit einer korrekten Spezifikation neu.

- Eine Ingress-Regel steht im Konflikt mit einer anderen Ingress-Regel im selben Kubernetes-Cluster.
- Die Anmerkung `allow-http` wird auf **Falsch** gesetzt, und der Ingress hat keinen TLS-Abschnitt.
- Bei einer Ingress-Regel sind `host` und `path` nicht angegeben. Eine Ingress-Regel dieser Art weist dieselbe Funktionalität wie das Ingress-Standard-Backend auf. Verwenden Sie stattdessen das Ingress-Standard-Backend.

Dienst vom Typ LoadBalancer

- NSX-T Data Center erstellt für jeden Dienst-Port einen virtuellen Server und einen Pool des Load Balancers auf Schicht 4.
- TCP und UDP werden unterstützt.
- Jeder Dienst hat eine eindeutige IP-Adresse.
- Dem Dienst wird eine IP-Adresse aus einem externen IP-Pool zugeteilt, die auf dem Feld "loadBalancerIP" in der LoadBalancer-Definition basiert. Das Feld "loadBalancerIP" kann leer sein, eine IP-Adresse oder den Namen oder die ID eines IP-Pools enthalten. Wenn die Spezifikation `loadBalancerIP` leer ist, wird die IP-Adresse aus dem externen IP-Pool zugeteilt, der durch die `external_ip_pools_lb`-Option im Abschnitt `[nsx_v3]` in `ncp.ini` angegeben ist. Wenn die Option `external_ip_pools_lb` nicht vorhanden ist, wird der von `external_ip_pools` angegebene Pool verwendet. Der LoadBalancer-Dienst wird unter dieser IP-Adresse und auf dem Dienst-Port bereitgestellt.
- Sie können zu einem anderen IP-Pool wechseln, indem Sie die Konfiguration ändern und NCP neu starten.
- Der von `loadBalancerIP` angegebene IP-Pool muss über das Tag `scope: ncp/owner`, `tag: cluster:<cluster_name>` verfügen.

- Fehler werden im Dienst als Anmerkung dokumentiert. Der Fehlerschlüssel lautet `ncp/error.loadbalancer`. Dies sind die möglichen Fehler:
 - `ncp/error.loadbalancer: IP_POOL_NOT_FOUND`

Dieser Fehler weist darauf hin, dass Sie `loadBalancerIP: <nsx-ip-pool>` angeben, `<nsx-ip-pool>` aber nicht vorhanden ist. Der Dienst ist dann inaktiv. Um den Fehler zu beheben, geben Sie einen gültigen IP-Pool ein, löschen Sie den Dienst und erstellen Sie ihn neu.
 - `ncp/error.loadbalancer: IP_POOL_EXHAUSTED`

Dieser Fehler weist darauf hin, dass Sie `loadBalancerIP: <nsx-ip-pool>` angeben, die IP-Adressen im IP-Pool aber ausgeschöpft sind. Der Dienst ist dann inaktiv. Um den Fehler zu beheben, geben Sie einen IP-Pool mit verfügbaren IP-Adressen an, löschen Sie den Dienst und erstellen Sie ihn neu.
 - `ncp/error.loadbalancer: IP_POOL_NOT_UNIQUE`

Dieser Fehler weist darauf hin, dass mehrere IP-Pools den von `loadBalancerIP` angegebenen Namen aufweisen: `<nsx-ip-pool>`. Der Dienst ist dann inaktiv.
 - `ncp/error.loadbalancer: POOL_ACCESS_DENIED`

Dieser Fehler weist darauf hin, dass der von `loadBalancerIP` angegebene IP-Pool das Tag `scope: ncp/owner, tag: cluster:<cluster_name>` nicht aufweist oder der Name des im Tag angegebenen Clusters nicht mit dem Namen des Kubernetes-Clusters übereinstimmt.
 - `ncp/error.loadbalancer: LB_VIP_CONFLICT`

Dieser Fehler weist darauf hin, dass die IP im `loadBalancerIP`-Feld mit der IP eines aktiven Diensts identisch ist. Der Dienst ist dann inaktiv.
- Das automatische Skalieren des Load Balancers auf Schicht 4 wird unterstützt. Wenn ein Kubernetes-LoadBalancer-Dienst erstellt oder geändert wird, sodass er zusätzliche virtuelle Server erfordert, und der vorhandene Load Balancer auf Schicht 4 nicht über ausreichend Kapazität verfügt, wird ein neuer Load Balancer auf Schicht 4 erstellt. NCP löscht einen Load Balancer auf Schicht 4 auch, wenn keine virtuellen Server mehr an ihn angehängt sind. Diese Funktion ist standardmäßig aktiviert. Sie können sie deaktivieren, indem Sie in der NCP-ConfigMap `l4_lb_auto_scaling` auf **false** festlegen.

Load Balancer und Netzwerkrichtlinie

Wenn Datenverkehr über den virtuellen Server des NSX-Load Balancer an die Pods weitergeleitet wird, handelt es sich bei der Quell-IP um die IP-Adresse des Uplink-Ports des Tier-1-Routers. Diese Adresse befindet sich im privaten Tier-1-Transit-Netzwerk und kann dazu führen, dass die CIDR-basierten Netzwerkrichtlinien zulässigen Datenverkehr nicht zulassen. Zur Vermeidung dieses Problems muss die Netzwerkrichtlinie so konfiguriert werden, dass die IP-Adresse des Uplink-Ports des Tier-1-Routers Teil des zulässigen CIDR-Blocks ist. Diese interne IP-Adresse wird im Feld `"status.loadbalancer.ingress.ip"` und als Anmerkung (`ncp/internal_ip_for_policy`) auf der Ingress-Ressource angezeigt.

Lautet die externe IP-Adresse des virtuellen Servers beispielsweise 4.4.0.5 und die IP-Adresse des Uplink-Ports des internen Tier-1-Routers 100.64.224.11, sieht der Status folgendermaßen aus:

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

Die Anmerkung zum Ingress und zum Dienst vom Typ LoadBalancer lautet:

```
ncp/internal_ip_for_policy: 100.64.224.11
```

Die IP-Adresse 100.64.224.11 muss zur zulässigen CIDR im ipBlock-Selektor der Netzwerkrichtlinie gehören. Beispiel:

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

Beispielskript zum Generieren eines von einer Zertifizierungsstelle signierten Zertifikats

Das nachfolgende Skript generiert ein von einer Zertifizierungsstelle signiertes Zertifikat und einen privaten in den Dateien <Dateiname>.crt und <Dateiname>.key gespeicherten privaten Schlüssel. Der Befehl `genrsa` generiert einen Zertifizierungsstellen-Schlüssel. Der Zertifizierungsstellen-Schlüssel muss verschlüsselt werden. Sie können eine Verschlüsselungsmethode mit einem Befehl wie zum Beispiel `aes256` angeben.

```
#!/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
```

Mounten Sie das Standardzertifikat und den Standardschlüssel in den NCP-Pod

Wenn Sie das Zertifikat und den privaten Schlüssel erstellt haben, platzieren Sie sie im Verzeichnis `/etc/nsx-ujo` auf dem VM-Host. Vorausgesetzt, dass die Zertifikats- und Schlüsseldateien die Namen `lb-default.crt` und `lb-default.key` aufweisen, bearbeiten Sie `ncp-rc.yaml`, sodass diese Dateien auf dem Host in den Pod gemountet werden. Beispiel:

```
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
```

Ingress-Controller von Drittanbietern

Sie können NCP für die Unterstützung von Ingress-Controllern von Drittanbietern konfigurieren.

Bearbeiten der Datei `NCP.ini`

Sie müssen die Konfigurationsdatei `/var/vcap/data/jobs/ncp/xxxxxxx/config/ncp.ini` bearbeiten (wobei `xxxxxxx` die BOSH-Bereitstellungs-ID ist). Diese Datei wird dann in `rootfs` kopiert und von NCP jedes Mal verwendet, wenn NCP neu gestartet wird. Diese Datei muss auf jedem Master-Knoten bearbeitet werden.

Wichtig Änderungen an `ncp.ini` sind bei PKS-Cluster-Aktualisierungen nicht persistent. Wenn Sie Änderungen über die PKS-Kachel vornehmen und dann die PKS-Bereitstellung aktualisieren, gehen die Änderungen an `ncp.ini` verloren.

Die relevanten Optionen befinden sich im Abschnitt `nsx_v3`.

- `use_native_loadbalancer`: Wenn dies auf **False** festgelegt ist, verarbeitet NCP unabhängig von den zugehörigen Anmerkungen keine Ingresses oder Dienste vom Typ LoadBalancer-Aktualisierungen. Diese Einstellung gilt für den gesamten PKS-Cluster. Die Standardeinstellung lautet **True**.
- `default_ingress_class_nsx`: Wenn dies auf **True** festgelegt ist, wird NCP zum standardmäßigen Ingress-Controller und verarbeitet sowohl Dateneingänge mit der Anmerkung `kubernetes.io/ingress.class: "nsx"` als auch Dateneingänge ohne Anmerkung. Wenn der Wert auf **False** festgelegt ist, verarbeitet NCP nur Ingresses mit der Anmerkung `kubernetes.io/ingress.class: "nsx"`. Die Standardeinstellung lautet **True**.

Wenn Sie möchten, dass NCP dem NGINX-Controller-Pod eine dynamische IP-Adresse zuweist und den Status von Ingresses mit der dynamischen IP-Adresse aktualisiert, führen Sie die folgenden Schritte aus:

- Legen Sie im Abschnitt `k8s` in `ncp.ini` den Wert `ingress_mode=nat` fest.
- Fügen Sie dem NGINX-Ingress-Controller-Pod die Anmerkung `ncp/ingress-controller: "True"` hinzu.

NCP aktualisiert den Status von Dateneingängen, die die Anmerkung `kubernetes.io/ingress.class: "nginx"` aufweisen, mit der dynamischen IP-Adresse des NGINX-Ingress-Controller-Pods. Wenn `default_ingress_class_nsx=False` festgelegt ist, aktualisiert NCP auch den Status von Dateneingängen ohne die Anmerkung `kubernetes.io/ingress.class` mit der dynamischen IP-Adresse des NGINX-Ingress-Controller-Pods.

Hinweis: Selbst wenn der NGINX-Ingress-Controller-Pod nicht über die Anmerkung `ncp/ingress-controller: "True"` verfügt, aktualisiert NCP den Status der oben erwähnten Dateneingänge auf `loadBalancer: {}`. Die Dateneingänge könnten dann in einer Schleife festgehalten werden, in der der NGINX-Controller den Ingress-Status auf `loadBalancer: {ingress: [{ip: <IP>}]}` aktualisiert und NCP den Ingress-Status auf `loadBalancer: {}` aktualisiert. Um diese Situation zu vermeiden, führen Sie die folgenden Schritte aus:

- Wenn der Ingress-Controller von <https://github.com/kubernetes/ingress-nginx> stammt,
 - Ändern Sie `ingress-class` auf dem Ingress-Controller in einen anderen Wert als `"nginx"`.
 - Wenn ein Ingress mit der Anmerkung `kubernetes.io/ingress-class: "nginx"` vorhanden ist, ändern Sie die Anmerkung in einen anderen Wert.
 - Weitere Informationen finden Sie unter <https://kubernetes.github.io/ingress-nginx/user-guide/multiple-ingress>.
- Wenn der Ingress-Controller von <https://github.com/nginxinc/kubernetes-ingress> stammt,
 - Ändern Sie `ingress-class` auf dem Ingress-Controller in einen anderen Wert als `"nginx"`.
 - Wenn ein Ingress mit der Anmerkung `kubernetes.io/ingress-class: "nginx"` vorhanden ist, ändern Sie die Anmerkung in einen anderen Wert.

- Legen Sie `use-ingress-class-only` auf dem Ingress-Controller-Pod auf **True** fest. Dadurch wird verhindert, dass dieser Controller Ingresses ohne die Anmerkung `kubernetes.io/ingress-class` aktualisiert.
- Weitere Informationen finden Sie unter <https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/multiple-ingress.md>.

Szenario 1: NCP verarbeitet Dateneingänge, ist aber nicht der standardmäßige Ingress-Controller.

Befolgen Sie dieses Verfahren, damit NCP die Ingresses von `nsx`-Klassen verarbeiten kann.

- 1 Bearbeiten Sie den Abschnitt `nsx_v3` in `ncp.ini` auf jedem Master-Knoten.
 - a Legen Sie `default_ingress_class_nsx` auf **False** fest.
 - b Lassen Sie `use_native_loadbalancer` auf **True** festgelegt. Dies ist der Standardwert.
- 2 Starten Sie NCP auf jedem Master-Knoten neu. Dies kann zu einem Master-Failover führen.
- 3 Fügen Sie allen Ingresses, die NCP verarbeiten soll, die Anmerkung `kubernetes.io/ingress.class: "nsx"` hinzu.

Szenario 2: NCP ist der standardmäßige Ingress-Controller.

Befolgen Sie dieses Verfahren:

- 1 Es ist nicht erforderlich, `ncp.ini` zu bearbeiten, sondern es muss sichergestellt werden, dass alle Ingresses mit Anmerkungen versehen sind.
- 2 Die von NCP zu verarbeitenden Dateneingänge sollten mit der Anmerkung **`kubernetes.io/ingress.class: "nsx"`** versehen werden.

Obwohl NCP Dateneingänge ohne die Anmerkung `kubernetes.io/ingress.class` verarbeitet, besteht die bewährte Vorgehensweise bei mehreren Ingress-Controllern darin, immer über die Anmerkung `kubernetes.io/ingress.class` zu verfügen und sich nicht auf das Verhalten des standardmäßigen Ingress-Controllers zu verlassen.

- 3 Dateneingänge, die von Drittanbieter-Ingress-Controllern verarbeitet werden, müssen als Anmerkung mit dem Wert versehen werden, der von diesen Ingress-Controllern benötigt wird.

Wichtig Sofern es nicht das Ziel ist, NGINX als standardmäßigen Ingress-Controller festzulegen, verwenden Sie **nginx** nicht als NGINX-Ingress-Controller, da NGINX dadurch zum standardmäßigen Ingress-Controller wird.

Szenario 3: NCP verarbeitet unabhängig von der Anmerkung keinen Ingress.

Befolgen Sie dieses Verfahren:

- 1 Bearbeiten Sie den Abschnitt `nsx_v3` in `ncp.ini` auf jedem Master-Knoten.
 - a Legen Sie `use_native_loadbalancer` auf **False** fest. Der Wert `default_ingress_class_nsx` ist nun irrelevant.
 - 2 Starten Sie NCP auf jedem Master-Knoten neu. Dies kann zu einem Master-Failover führen.
- Beachten Sie, dass NCP auch die Dienste vom Typ „LoadBalancer“ nicht verarbeitet.

Verwalten von NSX Container Plug-in

6

Sie können NSX Container Plug-in über die NSX Manager-GUI oder über die Befehlszeilenschnittstelle (Command Line Interface, CLI) verwalten.

Hinweis Wenn eine Container-Host-VM auf ESXi 6.5 ausgeführt wird und die VM über vMotion auf einen anderen ESXi 6.5-Host migriert wird, geht die Verbindung von Containern, die auf dem Container-Host ausgeführt werden, mit Containern, die auf anderen Container-Hosts ausgeführt werden, verloren. Sie können das Problem lösen, indem Sie die vNIC des Container-Hosts trennen und erneut verbinden. Dieses Problem tritt bei ESXi 6.5 Update 1 oder höher nicht auf.

Hyperbus reserviert die VLAN-ID 4094 auf dem Hypervisor für die PVLAN-Konfiguration, und die ID kann nicht geändert werden. Um VLAN-Konflikte zu vermeiden, konfigurieren Sie logische VLAN-Switches oder VTEP-vmknics mit derselben VLAN-ID.

Dieses Kapitel enthält die folgenden Themen:

- [Anzeigen von in der Kubernetes-Ressource NSXError gespeicherten Fehlerinformationen](#)
- [CIF-verknüpfte logische Ports](#)
- [CLI-Befehle](#)
- [Fehlercodes](#)

Anzeigen von in der Kubernetes-Ressource NSXError gespeicherten Fehlerinformationen

Für jedes Kubernetes-Ressourcenobjekt, das NSX-Backend-Fehler aufweist, wird ein NSXError-Objekt mit Fehlerinformationen erstellt. Es gibt auch ein Fehlerobjekt für alle den gesamten Cluster betreffenden Fehler.

Diese Funktion ist standardmäßig nicht aktiviert. Um sie zu aktivieren, müssen Sie beim Installieren von NCP in `ncp.inienable_nsx_err_crd` auf True festlegen.

Hinweis NSXError-Objekte dürfen nicht erstellt, aktualisiert oder gelöscht werden.

Befehle zum Anzeigen von NSXError-Objekten:

- `kubectl get nsxerrors`

Listet alle NSXError-Objekte auf.

- `kubectl get nsxerrors -l error-object-type=<type of resource>`

Listet alle mit einem bestimmten Typ von Kubernetes-Objekten verbundene NSXError-Objekte auf, beispielsweise Objekte des Typs `services`.

- `kubectl get nsxerrors <nsxerror name> -o yaml`

Zeigt die Details eines NSXError-Objekts an.

- `kubectl get svc <service name> -o yaml | grep nsxerror`

Findet den mit einem bestimmten Dienst verknüpften NSXError.

Wenn Sie die Details eines NSXError-Objekts anzeigen, enthält der Abschnitt mit der Spezifikation die folgenden wichtigen Informationen. Beispiel:

```
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'
```

In diesem Beispiel lautet der Namespace `default`. Der Name des Diensts lautet `svc-1`. Der Typ der Kubernetes-Ressource lautet `services`.

In dieser Version werden die folgenden Fehler vom NSXError-Objekt unterstützt.

- Aufgrund eines NSX Edge-Grenzwerts konnte die automatische Skalierung keine zusätzlichen Load Balancer zuteilen.
- Die Anzahl der virtuellen Load Balancer-Server überschreitet den Grenzwert (automatische Skalierung ist nicht aktiviert).
- Die Anzahl der Load Balancer-Serverpools überschreitet den Grenzwert.
- Die Anzahl der Load Balancer-Serverpoolmitglieder überschreitet den Load Balancer- oder NSX Edge-Grenzwert.
- Beim Verarbeiten eines Diensts des Typs `LoadBalancer` sind die Floating-IP-Adressen ausgeschöpft.

CIF-verknüpfte logische Ports

CIFs (Container Interfaces) sind Netzwerkschnittstellen für Container, die mit logischen Ports auf einem Switch verbunden sind. Diese Ports werden als CIF-verknüpfte logische Ports bezeichnet.

Sie können CIF-verknüpfte logische Ports über die NSX Manager-GUI verwalten.

Verwalten von CIF-verknüpften logischen Ports

Navigieren Sie zu **Netzwerk > Switching > Ports**, um alle logischen Ports anzuzeigen, einschließlich der CIF-verknüpften logischen Ports. Klicken Sie auf die Anhangsverknüpfung eines CIF-verknüpften logischen Ports, um die Anlageninformationen anzuzeigen. Klicken Sie auf den Link des logischen Ports, um einen Fensterbereich mit vier Registerkarten zu öffnen: „Übersicht“, „Überwachen“, „Verwalten“ und „Zugehörig“. Wenn Sie auf **Zugehörig > Logische Ports** klicken, wird der zugehörige logische Port auf einem Uplink-Switch angezeigt. Weitere Informationen zu Switch-Ports finden Sie im *NSX-T-Administratorhandbuch*.

Netzwerküberwachungstools

Die folgenden Tools unterstützen CIF-verknüpfte logische Ports. Weitere Informationen zu diesen Tools finden Sie im *NSX-T-Administratorhandbuch*.

- Traceflow
- Portverbindung
- IPFIX
- Remote-Port-Mirroring mit GRE-Kapselung eines logischen Switch-Ports, der mit einem Container verbunden ist, wird unterstützt. Weitere Informationen finden Sie unter „Grundlegendes zum Switching-Profil für Port-Mirroring“ im *NSX-T-Administratorhandbuch*. Port-Mirroring des CIF-Ports zum VIF-Port wird jedoch über die Manager-Benutzeroberfläche nicht unterstützt.

CLI-Befehle

Um CLI-Befehle auszuführen, melden Sie sich beim NSX Container Plug-in-Container an, öffnen Sie ein Terminal, und führen Sie den Befehl `nsxcli` aus.

Sie können die CLI-Eingabeaufforderung auch aufrufen, indem Sie den folgenden Befehl für einen Knoten ausführen:

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

Tabelle 6-1. CLI-Befehle für den NCP-Container

Typ	Befehl	Hinweis
Status	<code>get ncp-master status</code>	Für Kubernetes und PCF.
Status	<code>get ncp-nsx status</code>	Für Kubernetes und PCF.
Status	<code>get ncp-watcher <Watcher-Name></code>	Für Kubernetes und PCF.
Status	<code>get ncp-watchers</code>	Für Kubernetes und PCF.
Status	<code>get ncp-k8s-api-server status</code>	Für Kubernetes.
Status	<code>check projects</code>	Für Kubernetes.

Tabelle 6-1. CLI-Befehle für den NCP-Container (Fortsetzung)

Typ	Befehl	Hinweis
Status	check project <project-name>	Für Kubernetes.
Status	get ncp-bbs status	Nur für PCF.
Status	get ncp-capi status	Nur für PCF.
Status	get ncp-policy-server status	Nur für PCF.
Cache	get project-caches	Für Kubernetes.
Cache	get project-cache <Projektname>	Für Kubernetes.
Cache	get namespace-caches	Für Kubernetes.
Cache	get namespace-cache <Namensraum-Name>	Für Kubernetes.
Cache	get pod-caches	Für Kubernetes.
Cache	get pod-cache <Pod-Name>	Für Kubernetes.
Cache	get ingress-caches	Für Kubernetes.
Cache	get ingress-cache <ingress-name>	Für Kubernetes.
Cache	get ingress-controllers	Für Kubernetes.
Cache	get ingress-controller <ingress-controller-name>	Für Kubernetes.
Cache	get network-policy-caches	Für Kubernetes.
Cache	get network-policy-cache <pod-name>	Für Kubernetes.
Cache	get asg-caches	Nur für PCF.
Cache	get asg-cache <asg-ID>	Nur für PCF.
Cache	get org-caches	Nur für PCF.
Cache	get org-cache <org-ID>	Nur für PCF.
Cache	get space-caches	Nur für PCF.
Cache	get space-cache <space-ID>	Nur für PCF.
Cache	get app-caches	Nur für PCF.
Cache	get app-cache <app-ID>	Nur für PCF.
Cache	get instance-caches <app-ID>	Nur für PCF.
Cache	get instance-cache <app-ID> <instance-ID>	Nur für PCF.
Cache	get policy-caches	Nur für PCF.
Support	get ncp-log file <Dateiname>	Für Kubernetes und PCF.

Tabelle 6-1. CLI-Befehle für den NCP-Container (Fortsetzung)

Typ	Befehl	Hinweis
Support	get ncp-log-level	Für Kubernetes und PCF.
Support	set ncp-log-level <log-level>	Für Kubernetes und PCF.
Support	get support-bundle file <Dateiname>	Für Kubernetes.
Support	get node-agent-log file <Dateiname>	Für Kubernetes.
Support	get node-agent-log file <Dateiname> <Knotenname>	Für Kubernetes.

Tabelle 6-2. CLI-Befehle für den NSX-Knoten-Agent-Container

Typ	Befehl
Status	get node-agent-hyperbus status
Cache	get container-cache <Containername>
Cache	get container-caches

Tabelle 6-3. CLI-Befehle für den NSX-Kube-Proxy-Container

Typ	Befehl
Status	get ncp-k8s-api-server status
Status	get kube-proxy-watcher <Watcher-Name>
Status	get kube-proxy-watchers
Status	dump ovs-flows

Statusbefehle für den NCP-Container

- Status des NCP-Masters anzeigen

```
get ncp-master status
```

Beispiel:

```
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
```

- Verbindungsstatus zwischen NCP und NSX Manager anzeigen

```
get ncp-nsx status
```

Beispiel:

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

■ Watcher-Status für Ingress, Namensraum, Pod und Dienst anzeigen

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

Beispiel:

```
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
```

- Verbindungsstatus zwischen NCP und Kubernetes-API-Server anzeigen

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

Beispiel:

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

- Alle Projekte oder ein bestimmtes Projekt überprüfen

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

Beispiel:

```
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
```

- Verbindungsstatus zwischen NCP und PCF-BBS überprüfen

```
get ncp-bbs status
```

Beispiel:

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

- Verbindungsstatus zwischen NCP und PCF-CAPI überprüfen

```
get ncp-capi status
```

Beispiel:

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

- Verbindungsstatus zwischen NCP und PCF-Richtlinienserver überprüfen

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

Beispiel:

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

Cachebefehle für den NCP-Container

- Internen Cache für Projekte oder Namensräume abrufen

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

Beispiel:

```
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-ad32-338dc92316f
  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
```

■ Internen Cache für Pods abrufen

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

Beispiel:

```
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
```

■ Alle Ingress-Caches oder einen bestimmten Ingress-Cache abrufen

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

Beispiel:

```
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
```

- Alle Ingress-Caches oder einen bestimmten Ingress-Cache abrufen, einschließlich deaktivierter Controller

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

Beispiel:

```
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

```

- Netzwerkrichtlinien-Caches oder einen bestimmten Netzwerkrichtlinien-Cache abrufen

```

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

```

Beispiel:

```

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

```

- Alle ASG-Caches oder einen bestimmten ASG-Cache abrufen

```

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

```

Beispiel:

```

node> get asg-caches
edc04715-d04c-4e63-abb6-dbb601a668db6:
  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-abb6-dbb601a668db6

```

```
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:
```

- Alle Organisations-Caches oder einen bestimmten Organisations-Cache abrufen

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

Beispiel:

```
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
```

- Alle Speicher-Caches oder einen bestimmten Speicher-Cache abrufen

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

Beispiel:

```
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
```

- Alle App-Caches oder einen bestimmten App-Cache abrufen

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

Beispiel:

```
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

```

- Alle Instanz-Caches einer App oder einen bestimmten Instanz-Cache abrufen

```

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

```

Beispiel:

```

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
```

- Alle Richtlinien-Caches abrufen

```
get policy-caches
```

Beispiel:

```
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
```

Supportbefehle für den NCP-Container

- NCP-Support-Paket im Dateispeicher speichern

Das Support-Paket umfasst die Protokolldateien für alle Container in Pods mit der Bezeichnung **tier:nsx-networking**. Die Paketdatei liegt im TGZ-Format vor und befindet sich im CLI-Standard-Dateispeicherverzeichnis `/var/vmware/nsx/file-store`. Mithilfe des CLI-Befehls „file-store“ können Sie die Paketdatei in eine Remote-Site kopieren.

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

Beispiel:

```
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
```

■ NCP-Protokolle im Dateispeicher speichern

Die Protokolldatei wird im TGZ-Format im CLI-Standard-Dateispeicherverzeichnis `/var/vmware/nsx/file-store` gespeichert. Mithilfe des CLI-Befehls „file-store“ können Sie die Paketdatei in eine Remote-Site kopieren.

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

Beispiel:

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

■ Knoten-Agent-Protokolle im Dateispeicher speichern

Speichern Sie die Knoten-Agent-Protokolle von einem oder allen Knoten. Die Protokolle werden im TGZ-Format im CLI-Standard-Dateispeicherverzeichnis `/var/vmware/nsx/file-store` gespeichert. Mithilfe des CLI-Befehls „file-store“ können Sie die Paketdatei in eine Remote-Site kopieren.

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

Beispiel:

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

■ Protokollebene abrufen und einrichten

Zu den verfügbaren Protokollebenen gehören: NOTSET, DEBUG, INFO, WARNING, ERROR und CRITICAL.

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

Beispiel:

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

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

Status-Befehle für den NSX-Knoten-Agent-Container

- Zeigen Sie den Verbindungsstatus zwischen dem Knoten-Agent und HyperBus auf diesem Knoten an.

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

Beispiel:

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

Cache-Befehle für den NSX-Knoten-Agent-Container

- Internen Cache für NSX-Knoten-Agent-Container abrufen

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

Beispiel:

```
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
```

Status-Befehle für den NSX-Kube-Proxy-Container

- Verbindungsstatus zwischen Kube-Proxy und Kubernetes-API-Server anzeigen

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

Beispiel:

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

- Kube-Proxy-Watcher-Status anzeigen

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

Beispiel:

```
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
```

■ OVS-Flows für Speicherabbild an einem Knoten

```
dump ovs-flows
```

Beispiel:

```
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
```

Fehlercodes

In diesem Abschnitt werden die Fehlercodes der verschiedenen Komponenten aufgelistet.

NCP-Fehlercodes

Fehlercode	Beschreibung
NCP00001	Ungültige Konfiguration
NCP00002	Initialisierung fehlgeschlagen
NCP00003	Ungültiger Zustand
NCP00004	Ungültiger Adapter
NCP00005	Zertifikat wurde nicht gefunden
NCP00006	Token wurde nicht gefunden
NCP00007	Ungültige Konfiguration für NSX
NCP00008	Ungültiges NSX-Tag
NCP00009	NSX-Verbindung fehlgeschlagen
NCP00010	Knoten-Tag nicht gefunden
NCP00011	Ungültiger logischer Switch-Port des Knotens
NCP00012	Aktualisieren der übergeordneten VIF fehlgeschlagen
NCP00013	VLAN ausgeschöpft
NCP00014	VLAN-Version ist fehlgeschlagen
NCP00015	IP-Pool ist ausgeschöpft
NCP00016	IP-Version ist fehlgeschlagen
NCP00017	IP-Block ausgeschöpft
NCP00018	Erstellen des IP-Subnetzes ist fehlgeschlagen
NCP00019	Löschen des IP-Subnetzes ist fehlgeschlagen
NCP00020	Erstellen des IP-Pools ist fehlgeschlagen
NCP00021	Löschen des IP-Pools ist fehlgeschlagen
NCP00022	Erstellen des logischen Routers ist fehlgeschlagen

Fehlercode	Beschreibung
NCP00023	Aktualisieren des logischen Routers ist fehlgeschlagen
NCP00024	Löschen des logischen Routers ist fehlgeschlagen
NCP00025	Erstellen des logischen Switches ist fehlgeschlagen
Fehlercode	Beschreibung
NCP00026	Aktualisieren des logischen Switches ist fehlgeschlagen
NCP00027	Löschen des logischen Switches ist fehlgeschlagen
NCP00028	Erstellen des Ports für den logischen Router ist fehlgeschlagen
NCP00029	Löschen des Ports für den logischen Router ist fehlgeschlagen
NCP00030	Erstellen des Ports für den logischen Switch ist fehlgeschlagen
NCP00031	Aktualisieren des Ports für den logischen Switch ist fehlgeschlagen
NCP00032	Löschen des Ports für den logischen Switch ist fehlgeschlagen
NCP00033	Netzwerkrichtlinie wurde nicht gefunden
NCP00034	Erstellen der Firewall ist fehlgeschlagen
NCP00035	Lesen der Firewall ist fehlgeschlagen
NCP00036	Aktualisieren der Firewall ist fehlgeschlagen
NCP00037	Löschen der Firewall ist fehlgeschlagen
NCP00038	Mehrere Firewalls gefunden
NCP00039	Erstellen der NSGroup ist fehlgeschlagen
NCP00040	Löschen der NSGroup ist fehlgeschlagen
NCP00041	Erstellen von IP Set ist fehlgeschlagen
NCP00042	Aktualisieren von IP Set ist fehlgeschlagen
NCP00043	Löschen von IP Set ist fehlgeschlagen
NCP00044	Erstellen der SNAT-Regel ist fehlgeschlagen
NCP00045	Löschen der SNAT-Regel ist fehlgeschlagen
NCP00046	Adapter-API-Verbindung ist fehlgeschlagen
NCP00047	Adapter-Watcher-Ausnahme
NCP00048	Löschen des Load Balancer-Diensts ist fehlgeschlagen

Fehlercode	Beschreibung
NCP00049	Erstellen des virtuellen Servers für den Load Balancer ist fehlgeschlagen
NCP00050	Aktualisieren des virtuellen Servers für den Load Balancer ist fehlgeschlagen
NCP00051	Löschen des virtuellen Servers für den Load Balancer ist fehlgeschlagen
NCP00052	Erstellen des Load Balancer-Pools ist fehlgeschlagen
NCP00053	Aktualisieren des Load Balancer-Pools ist fehlgeschlagen
NCP00054	Löschen des Load Balancer-Pools ist fehlgeschlagen
NCP00055	Erstellen der Load Balancer-Regel ist fehlgeschlagen
NCP00056	Aktualisieren des Load Balancer-Pools ist fehlgeschlagen
NCP00057	Löschen der Load Balancer-Regel ist fehlgeschlagen
NCP00058	IP-Freigabe für Load Balancer-Pool ist fehlgeschlagen
NCP00059	Zuordnung von virtuellem Server und Dienstzuordnung für Load Balancer nicht gefunden
NCP00060	Aktualisieren der NSGroup ist fehlgeschlagen
NCP00061	Abrufen der Firewallregeln ist fehlgeschlagen
NCP00062	NSGroup – keine Kriterien
NCP00063	Knoten-VM nicht gefunden
NCP00064	Knoten-VIF nicht gefunden
NCP00065	Zertifikatimport ist fehlgeschlagen
NCP00066	Rückgängigmachen des Zertifikats ist fehlgeschlagen
NCP00067	Aktualisieren der SSL-Bindung ist fehlgeschlagen
NCP00068	SSL-Profil nicht gefunden
NCP00069	IP-Pool nicht gefunden
NCP00070	TO-Edge-Cluster nicht gefunden
NCP00071	Aktualisieren des IP-Pools ist fehlgeschlagen
NCP00072	Dispatcher ist fehlgeschlagen
NCP00073	Löschen der NAT-Regel ist fehlgeschlagen
NCP00074	Abrufen des Ports für den logischen Router ist fehlgeschlagen
NCP00075	NSX-Konfigurationsvalidierung ist fehlgeschlagen

Fehlercode	Beschreibung
NCP00076	Aktualisieren der SNAT-Regel ist fehlgeschlagen
NCP00077	SNAT-Regel überlagert
NCP00078	Hinzufügen der Load Balancer-Endpoints ist fehlgeschlagen
NCP00079	Aktualisieren der Load Balancer-Endpoints ist fehlgeschlagen
NCP00080	Erstellen des Load Balancer-Regelpools ist fehlgeschlagen
NCP00081	Virtueller Server für Load Balancer nicht gefunden
NCP00082	Lesen von IP Set ist fehlgeschlagen
NCP00083	Abrufen von SNAT-Pool ist fehlgeschlagen
NCP00084	Erstellen des Load Balancer-Diensts ist fehlgeschlagen
NCP00085	Aktualisieren des Load Balancer-Diensts ist fehlgeschlagen
NCP00086	Aktualisieren des Ports für den logischen Router ist fehlgeschlagen
NCP00087	Load Balancer-Initialisierung ist fehlgeschlagen
NCP00088	IP-Pool nicht eindeutig
NCP00089	Layer 7 des Load Balancers – Cache-Synchronisierungsfehler
NCP00090	Load Balancer-Pool ist nicht vorhanden
NCP00091	Fehler beim Initialisieren des Load Balancer-Regelcaches
NCP00092	SNAT-Prozess ist fehlgeschlagen
NCP00093	Fehler bei Load Balancer-Standardzertifikat
NCP00094	Löschen des Load Balancer-Endpoints ist fehlgeschlagen
NCP00095	Projekt nicht gefunden
NCP00096	Pool-Zugriff verweigert
NCP00097	Fehler beim Abrufen eines Load Balancer-Diensts
NCP00098	Fehler beim Erstellen eines Load Balancer-Diensts
NCP00099	Fehler bei Synchronisierung des Load Balancer-Pool-Caches

Fehlercodes für NSX-Knoten-Agent

Fehlercode	Beschreibung
NCP01001	OVS-Uplink nicht gefunden
NCP01002	Host-MAC nicht gefunden

Fehlercode	Beschreibung
NCP01003	OVS-Porterstellung ist fehlgeschlagen
NCP01004	Keine Pod-Konfiguration
NCP01005	Pod-Konfiguration ist fehlgeschlagen
NCP01006	Aufheben der Pod-Konfiguration ist fehlgeschlagen
NCP01007	CNI-Socket nicht gefunden
NCP01008	CNI-Verbindung ist fehlgeschlagen
NCP01009	CNI-Version stimmt nicht überein
NCP01010	CNI-Nachrichtenempfang ist fehlgeschlagen
NCP01011	CNI-Nachrichtenübertragung ist fehlgeschlagen
NCP01012	Hyperbus-Verbindung ist fehlgeschlagen
NCP01013	Hyperbus-Version stimmt nicht überein
NCP01014	Fehler beim Empfang der Hyperbus-Nachricht
NCP01015	Hyperbus-Nachrichtenübertragung ist fehlgeschlagen
NCP01016	GARP-Senden ist fehlgeschlagen
NCP01017	Schnittstellenkonfiguration ist fehlgeschlagen

Fehlercodes für nsx-kube-proxy

Fehlercode	Beschreibung
NCP02001	Ungültiger Gateway-Port des Proxys
NCP02002	Proxy-Befehl ist fehlgeschlagen
NCP02003	Proxy-Validierung ist fehlgeschlagen

CLI-Fehlercodes

Fehlercode	Beschreibung
NCP03001	CLI-Start ist fehlgeschlagen
NCP03002	Erstellen des CLI-Sockets ist fehlgeschlagen
NCP03003	CLI-Socket-Ausnahme
NCP03004	Ungültige Anforderung von CLI-Client
NCP03005	CLI-Server-Übertragung ist fehlgeschlagen

Fehlercode	Beschreibung
NCP03006	CLI-Server-Empfang ist fehlgeschlagen
NCP03007	CLI-Befehlsausführung ist fehlgeschlagen

Fehlercodes für Kubernetes

Fehlercode	Beschreibung
NCP05001	Kubernetes-Verbindung ist fehlgeschlagen
NCP05002	Ungültige Konfiguration für Kubernetes
NCP05003	Kubernetes-Anforderung ist fehlgeschlagen
NCP05004	Kubernetes-Schlüssel nicht gefunden
NCP05005	Kubernetes-Typ nicht gefunden
NCP05006	Ausnahme bei Kubernetes-Wächter
NCP05007	Kubernetes-Ressource weist ungültige Länge auf
NCP05008	Kubernetes-Ressource weist ungültigen Typ auf
NCP05009	Kubernetes-Ressourcen-Handle ist fehlgeschlagen
NCP05010	Kubernetes-Dienst-Handle ist fehlgeschlagen
NCP05011	Kubernetes-Endpoint-Handle ist fehlgeschlagen
NCP05012	Kubernetes-Ingress-Handle ist fehlgeschlagen
NCP05013	Kubernetes-Netzwerkrichtlinien-Handle ist fehlgeschlagen
NCP05014	Kubernetes-Knoten-Handle ist fehlgeschlagen
NCP05015	Kubernetes-Namespace-Handle ist fehlgeschlagen
NCP05016	Kubernetes-Pod-Handle ist fehlgeschlagen
NCP05017	Kubernetes-Secret-Handle ist fehlgeschlagen
NCP05018	Kubernetes-Standard-Backend ist fehlgeschlagen
NCP05019	Nicht unterstützter Übereinstimmungsausdruck für Kubernetes
NCP05020	Aktualisieren des Kubernetes-Status ist fehlgeschlagen
NCP05021	Aktualisieren des Kubernetes-Kommentars ist fehlgeschlagen
NCP05022	Kubernetes-Namespace-Cache nicht gefunden
NCP05023	Kubernetes-Secret nicht gefunden

Fehlercode	Beschreibung
NCP05024	Kubernetes-Standard-Backend wird verwendet
NCP05025	Kubernetes-LoadBalancer-Dienst-Handle ist fehlgeschlagen

Fehlercodes für Pivotal Cloud Foundry

Fehlercode	Beschreibung
NCP06001	PCF BBS-Verbindung ist fehlgeschlagen
NCP06002	PCF CAPI-Verbindung ist fehlgeschlagen
NCP06006	PCF-Cache nicht gefunden
NCP06007	Unbekannte Domäne für PCF
NCP06020	PCF-Richtlinienserver-Verbindung fehlgeschlagen
NCP06021	PCF-Richtlinienverarbeitung ist fehlgeschlagen
NCP06030	PCF-Ereignisverarbeitung ist fehlgeschlagen
NCP06031	Unerwarteter Ereignistyp für PCF
NCP06032	Unerwartete Ereignisinstanz für PCF
NCP06033	Löschen von PCF-Aufgabe ist fehlgeschlagen
NCP06034	PCF-Dateizugriff ist fehlgeschlagen