NSX Container Plug-in pour Kubernetes et Cloud Foundry - Guide d'installation et d'administration

VMware NSX Container Plug-in 2.4, 2.4.1

VMware NSX-T Data Center 2.4

vmware®

Vous trouverez la documentation technique la plus récente sur le site Web de VMware, à l'adresse :

https://docs.vmware.com/fr/

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

VMware France SAS.
Tour Franklin
100-101 Terrasse Boieldieu
92042 Paris La Défense 8 Cedex
France
www.vmware.com/fr

Table des matières

NSX Container Plug-in pour Kubernetes et Cloud Foundry - Guide d'installation et d'administration 5

1 Présentation de NSX Container Plug-in 6
- Conditions de compatibilité 7
- Présentation de l'installation 8
- Mettre à niveau NCP dans un environnement Kubernetes 8
- Mettre à niveau NCP dans un environnement Pivotal Cloud Foundry 9

2 Configuration des ressources de NSX-T 11
- Configuration des ressources de NSX-T 11

3 Installation de NCP dans un environnement Kubernetes 18
- Installer le plug-in CNI NSX-T Data Center 18
- Installer et configurer OVS 19
- Configurer la mise en réseau NSX-T Data Center pour des nœuds Kubernetes 21
- Installer l'agent de nœud NSX 22
- ConfigMap pour ncp.ini dans nsx-node-agent-ds.yml 23
- Installation de NSX Container Plug-in 26
- ConfigMap pour ncp.ini dans ncp-rc.yml 29
- Monter un certificat codé au format PEM et une clé privée dans l'espace NCP 34
- Monter un fichier de certificat dans l'espace NCP 35
- Configuration de Syslog 35
  - Créer un conteneur complémentaire pour Syslog 36
  - Créer un réplica DaemonSet pour Syslog 38
  - Exemple : configuration de la rotation des journaux et de Syslog s'exécutant dans un conteneur complémentaire 39
- Considérations de sécurité 46
- Conseils sur la configuration des ressources réseau 50

4 Installation de NCP dans un environnement Pivotal Cloud Foundry 52
- Installer NCP dans un environnement Pivotal Cloud Foundry 52

5 Équilibrage de charge 55
- Configuration de l'équilibrage de charge 55
- Contrôleurs d'entrée tiers 63

6 Administration de NSX Container Plug-in 66

Affichage des informations d'erreur stockées dans la ressource Kubernetes NSXError 66
Ports logiques attachés par CIF 67
Commandes d'interface de ligne de commande 68
Codes d'erreur 87

NSX Container Plug-in pour Kubernetes et Cloud Foundry - Guide d'installation et d'administration

Ce guide décrit l'installation et l'administration de NSX Container Plug-in (NCP) pour assurer l'intégration entre NSX-T Data Center et Kubernetes, ainsi qu'entre NSX-T Data Center et Pivotal Cloud Foundry (PCF).

Public visé

Ce guide est destiné aux administrateurs système et réseau. Il est impératif de connaître les procédures d'installation et d'administration de NSX-T Data Center Kubernetes et de Pivotal Cloud Foundry.

Glossaire VMware Technical Publications

VMware Technical Publications fournit un glossaire de termes pouvant ne pas vous être familiers. Pour consulter la définition des termes utilisés dans la documentation technique VMware, visitez le site Web http://www.vmware.com/support/pubs.

Présentation de NSX Container Plug-in

1

NSX Container Plug-in (NCP) assure l'intégration entre NSX-T Data Center et les orchestrateurs de conteneur, tels que Kubernetes, ainsi que l'intégration entre NSX-T Data Center et les produits PaaS (plate-forme en tant que service) basés sur un conteneur, tels qu'OpenShift et Pivotal Cloud Foundry. Ce guide décrit comment configurer NCP avec Kubernetes et Pivotal Cloud Foundry.

Le composant principal de NCP s'exécute dans un conteneur et communique avec NSX Manager et avec le plan de contrôle Kubernetes. NCP surveille les modifications apportées aux conteneurs et d'autres ressources et gère des ressources réseau, telles que des ports logiques, des commutateurs, des routeurs et des groupes de sécurité pour les conteneurs en appelant NSX API.

Le plug-in NSX CNI s'exécute sur chaque nœud Kubernetes. Il surveille les événements de cycle de vie des conteneurs, connecte une interface de conteneur au vSwitch invité et programme le vSwitch invité pour marquer et transférer le trafic de conteneur entre les interfaces de conteneur et la carte réseau virtuelle.

NCP fournit les fonctionnalités suivantes :

- Création automatique d'une topologie logique NSX-T Data Center pour un cluster Kubernetes automatiquement et création d'un réseau logique distinct pour chaque espace de noms Kubernetes.
- Connexion des espaces Kubernetes au réseau logique et allocation des adresses IP et MAC.
- Prise en charge de la traduction d'adresse réseau (NAT) et allocation d'une adresse IP SNAT distincte pour chaque espace de noms Kubernetes.

  Note Lors de la configuration de NAT, le nombre total d'adresses IP traduites ne doit pas dépasser 1 000.

- Implémentation de stratégies réseau Kubernetes avec pare-feu distribué NSX-T Data Center.
  - Prise en charge des stratégies réseau d'entrée et de sortie.
  - Prise en charge de sélecteur d'`IPBlock` dans les stratégies réseau.
  - Prise en charge de `matchLabels` et de `matchExpression` lors de la spécification de sélecteurs d'étiquette pour les stratégies réseau.

    - Prise en charge de la sélection des espaces dans un autre espace de noms.
- Implémentation de service Kubernetes de type `ClusterIP` et de service de type `LoadBalancer`.
- Implémentation de Kubernetes Ingress avec un équilibrage de charge de couche 7 NSX-T.
    - Prise en charge du trafic entrant HTTP et HTTPS avec terminaison Edge TLS.
    - Prise en charge de la configuration de serveur principal par défaut pour le trafic entrant.
    - Prise en charge de la réécriture d'URI pour le trafic entrant.
- Création de balises sur le port de commutateur logique NSX-T Data Center pour l'espace de noms, le nom de l'espace et les étiquettes d'un espace, et définition par l'administrateur de groupes de sécurité et de stratégies NSX-T basées sur les balises.

Dans cette version, NCP prend en charge un seul cluster Kubernetes. Vous pouvez avoir plusieurs clusters Kubernetes, chacun avec son instance distincte de NCP, utilisant le même déploiement de NSX-T Data Center.

Ce chapitre contient les rubriques suivantes :

- Conditions de compatibilité
- Présentation de l'installation
- Mettre à niveau NCP dans un environnement Kubernetes
- Mettre à niveau NCP dans un environnement Pivotal Cloud Foundry

Conditions de compatibilité

NSX Container Plug-in (NCP) a les conditions requises de compatibilité suivantes pour un environnement Kubernetes et un environnement Pivotal Cloud Foundry (PCF).

Tableau 1-1. Conditions requises de compatibilité pour un environnement Kubernetes

Produit logiciel	Version
NSX-T Data Center	2.3, 2.4
Hyperviseur pour machines virtuelles d'hôte de conteneur	■ Version de vSphere prise en charge ■ RHEL KVM 7.4, 7.5, 7.6 ■ Ubuntu KVM 16.04
Système d'exploitation d'hôte de conteneur	■ RHEL 7.5, 7.6 ■ Ubuntu 16.04 ■ CentOS 7.4, 7.5
Orchestrateur de conteneur	Kubernetes 1.12, 1.13
Open vSwitch d'hôte de conteneur	2.9.1 (fourni avec NSX-T Data Center 2.3.x), 2.10.2 (fourni avec NSX-T Data Center 2.4.0)

Tableau 1-2. Conditions requises de compatibilité pour un environnement Cloud Foundry

Produit logiciel	Version
Hyperviseur pour machines virtuelles d'hôte de conteneur	■ Version de vSphere prise en charge
Orchestrateur de conteneur	■ Pivotal Application Service 2.3.x et Pivotal Operations Manager 2.3.x ■ Pivotal Application Service 2.4.x (à l'exception de 2.4.0) et Pivotal Operations Manager 2.4.x (à l'exception de 2.4.0)

Présentation de l'installation

Dans un environnement où Kubernetes est déjà installé, l'installation et la configuration de NCP impliquent généralement les étapes suivantes. Pour effectuer la procédure correctement, vous devez être familiarisé avec l'installation et l'administration de NSX-T Data Center et de Kubernetes.

1 Installez NSX-T Data Center.

2 Créez une zone de transport de superposition.

3 Créez un commutateur logique de superposition et connectez les nœuds Kubernetes au commutateur.

4 Créez un routeur logique de niveau 0.

5 Créez des blocs d'adresses IP pour les espaces Kubernetes.

6 Créez des pools d'adresses IP pour SNAT (Source Network Address Translation, traduction d'adresse réseau source).

7 Installez le plug-in CNI (interface réseau de conteneur) NSX sur chaque nœud.

8 Installez OVS (Open vSwitch) sur chaque nœud.

9 Configurez la mise en réseau NSX-T pour les nœuds Kubernetes.

10 Installez l'agent de nœud NSX en tant que DaemonSet.

11 Installez NCP en tant que ReplicationController.

12 Montez des certificats de sécurité dans l'espace NCP.

Mettre à niveau NCP dans un environnement Kubernetes

Cette section décrit comment mettre à niveau NCP vers la version 2.4.0 dans un environnement Kubernetes.

Procédure

1 Mettez à niveau NCP ReplicationController avec la commande suivante (remplacez `<image>` par le nom réel de l'image).

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

2 Mettez à niveau daemonSet de l'agent du nœud NSX avec les commandes suivantes (remplacez <image> par le nom réel de l'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 Mettez à niveau le module DEB/RPM CNI vers la version 2.4.0 avec la commande suivante (remplacez <cni deb> et <cni rpm> par le nom réel du module).

Sous Ubuntu :

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

Sous RHEL ou CentOS :

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

4 (Facultatif) Mettez à niveau NSX-T Data Center vers la version 2.4.

Si l'hyperviseur est ESXi, mettez-le à niveau vers au moins la version 6.5p03 à partir de la version 6.5 ou vers la version 6.7ep6 à partir de la version 6.7 avant la mise à niveau de NSX-T Data Center.

5 (Facultatif) Mettez à niveau l'hyperviseur (KVM ou conteneur bare metal).

6 (Facultatif) Mettez à niveau l'hôte du conteneur (RHEL, Ubuntu ou CentOS).

7 (Facultatif) Mettez à niveau Kubernetes.

8 (Facultatif) Mettez à niveau OVS.

Pour un conteneur bare metal, NSX-T Data Center met également à niveau OVS, cette étape n'est donc pas nécessaire.

Pendant cette étape, vous pouvez voir des échecs de communication transitoires entre nsx-kube-proxy et nsx-node-agent. Ce comportement est normal et n'indique pas de problème.

Mettre à niveau NCP dans un environnement Pivotal Cloud Foundry

Cette section décrit comment mettre à niveau NCP vers la version 2.4.0 dans un environnement Pivotal Cloud Foundry.

Procédure

1 Mettez à niveau NCP ou NSX-T Tile vers la version 2.4.0.

2 (Facultatif) Mettez à niveau Pivotal Operations Manager, puis Pivotal Application Service.

3 (Facultatif) Mettez à niveau NSX-T Data Center vers la version 2.4.

Si l'hyperviseur est ESXi, mettez-le à niveau vers au moins la version 6.5p03 à partir de la version 6.5 ou vers la version 6.7ep6 à partir de la version 6.7 avant la mise à niveau de NSX-T Data Center.

4 (Facultatif) Mettez à niveau l'hyperviseur (KVM ou conteneur bare metal).

Configuration des ressources de NSX-T 2

Avant d'installer NSX Container Plug-in, vous devez configurer certaines ressources de NSX-T Data Center.

Ce chapitre contient les rubriques suivantes :

- Configuration des ressources de NSX-T

Configuration des ressources de NSX-T

Les ressources de NSX-T Data Center que vous devez configurer incluent une zone de transport de superposition, un routeur logique de niveau 0, un commutateur logique pour connecter les machines virtuelles du nœud, des blocs d'adresses IP pour les nœuds Kubernetes et un pool d'adresses IP pour SNAT.

Important Si vous exécutez avec NSX-T Data Center 2.4 ou version ultérieure, vous devez configurer les ressources NSX-T à l'aide de l'onglet **Mise en réseau et sécurité avancées**.

Dans le fichier de configuration NCP `ncp.ini`, les ressources NSX-T Data Center sont spécifiées à l'aide de leurs UUID ou de leurs noms.

Zone de transport de superposition

Connectez-vous à NSX Manager et accédez à **Système > Infrastructure > Zones de transport**. Recherchez la zone de transport de superposition utilisée pour la mise en réseau de conteneur ou créez-en une.

Spécifiez une zone de transport de superposition pour un cluster en définissant l'option `overlay_tz` dans la section `[nsx_v3]` de `ncp.ini`. Cette étape est facultative. Si vous ne définissez pas `overlay_tz`, NCP récupère automatiquement l'ID de zone de transport de superposition à partir du routeur de niveau 0.

Routage logique de niveau 0

Connectez-vous à NSX Manager et accédez à **Mise en réseau et sécurité avancées > Mise en réseau > Routeurs**. Recherchez le routeur utilisé pour la mise en réseau de conteneur ou créez-en un.

Spécifiez un routeur logique de niveau 0 pour un cluster en définissant l'option `tier0_router` dans la section `[nsx_v3]` de `ncp.ini`.

Note Le routeur doit être créé en mode actif-en veille.

Commutateur logique

Les cartes réseau virtuelles utilisées par le nœud pour le trafic de données doivent être connectées à un commutateur logique de superposition. Il n'est pas obligatoire que l'interface de gestion du nœud soit connectée à NSX-T Data Center, bien que cela facilite la configuration. Vous pouvez créer un commutateur logique en vous connectant à NSX Manager et en accédant à **Mise en réseau et sécurité avancées > Mise en réseau > Commutation > Commutateurs**. Sur le commutateur, créez des ports logiques et attachez-y les cartes réseau virtuelles du nœud. Les ports logiques doivent avoir les balises suivantes :

- balise : <`cluster_name`>, étendue : `ncp/cluster`
- balise : <`node_name`>, étendue : `ncp/node_name`

La valeur <`cluster_name`> doit correspondre à la valeur de l'option `cluster` dans la section `[coe]` de `ncp.ini`.

Blocs d'adresses IP pour des espaces Kubernetes

Connectez-vous à NSX Manager et accédez à **Mise en réseau et sécurité avancées > Mise en réseau > IPAM** pour créer un ou plusieurs blocs d'adresses IP. Spécifiez le bloc d'adresses IP au format CIDR.

Spécifiez les blocs d'adresses IP pour les espaces Kubernetes en définissant l'option `container_ip_blocks` dans la section `[nsx_v3]` de `ncp.ini`.

Vous pouvez également créer des blocs d'adresses IP spécifiquement pour des espaces de noms non SNAT (pour Kubernetes) ou des clusters (pour PCF).

Spécifiez les blocs d'adresses IP non SNAT en définissant l'option `no_snat_ip_blocks` dans la section `[nsx_v3]` de `ncp.ini`.

Si vous créez des blocs d'adresses IP non SNAT alors que NCP est en cours d'exécution, vous devez redémarrer NCP. Sinon, NCP continuera d'utiliser les blocs d'adresses IP partagés jusqu'à leur épuisement.

Note Lorsque vous créez un bloc d'adresses IP, le préfixe ne doit pas être supérieur à la valeur du paramètre `subnet_prefix` dans le fichier de configuration de NCP `ncp.ini`. Pour plus d'informations, consultez ConfigMap pour ncp.ini dans ncp-rc.yml.

Pool d'adresses IP pour SNAT

Un pool d'adresses IP dans NSX Manager sert à allouer des adresses IP qui seront utilisées pour la traduction d'adresses IP d'espace à l'aide de règles SNAT et pour l'exposition de contrôleurs d'entrée à l'aide de règles SNAT/DNAT, comme des adresses IP flottantes OpenStack. Ces adresses IP sont également appelées adresses IP externes.

Plusieurs clusters Kubernetes utilisent le même pool d'adresses IP externes. Chaque instance NCP utilise un sous-ensemble de ce pool pour le cluster Kubernetes qu'il gère. Par défaut, le même préfixe de sous-réseau pour les sous-réseaux d'espace sera utilisé. Pour utiliser une taille de sous-réseau différente, mettez à jour l'option `external_subnet_prefix` dans la section `[nsx_v3]` dans `ncp.ini`.

Vous pouvez spécifier des pools d'adresses IP pour SNAT en définissant l'option `external_ip_pools` dans la section `[nsx_v3]` de `ncp.ini`.

Vous pouvez spécifier un pool d'adresses IP différent en modifiant le fichier de configuration et en redémarrant NCP.

Restriction d'un pool d'adresses IP SNAT à des espaces de noms Kubernetes ou à des organisations PCF spécifiques

Vous pouvez spécifier l'espace de noms Kubernetes ou l'organisation PCF auquel les adresses IP sont allouées à partir du pool d'adresses IP SNAT en ajoutant les balises suivantes au pool d'adresses IP.

- Pour un espace de noms Kubernetes : `scope: ncp/owner, tag: ns:<namespace_UUID>`
- Pour une organisation PCF : `scope: ncp/owner, tag: org:<org_UUID>`

Vous pouvez obtenir l'UUID de l'espace de noms ou de l'organisation à l'aide de l'une des commandes suivantes :

```
kubectl get ns –o yaml
cf org <org_name> ––guid
```

Notez les points suivants :

- Chaque balise doit spécifier un UUID. Vous pouvez créer plusieurs balises pour le même pool.
- Si vous modifiez les balises après l'allocation d'adresses IP à des espaces de noms ou organisations selon les anciennes balises, ces adresses IP sont récupérées uniquement après la modification des configurations SNAT des services Kubernetes ou des applications PCF, ou le redémarrage de NCP.
- Les balises de propriétaire de l'espace de noms et de l'organisation PCF sont facultatives. Sans ces balises, les adresses IP du pool IP SNAT peuvent être allouées à n'importe quel espace de noms ou organisation PCF.

Configuration d'un pool d'adresses IP SNAT pour un service

Vous pouvez configurer un pool d'adresses IP SNAT pour un service spécifique en ajoutant une annotation au service. Par exemple,

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

Le pool d'adresses IP spécifié par `ncp/snat_pool` doit avoir la balise `scope: ncp/owner, tag: cluster:<cluster_name>`.

NCP configurera la règle SNAT pour ce service. L'adresse IP source de la règle correspond à l'ensemble des espaces de serveur principal. L'adresse IP de destination est l'adresse IP SNAT allouée à partir du pool d'adresses IP externe spécifié. Si une erreur se produit lorsque NCP configure la règle SNAT, le service sera annoté avec `ncp/error.snat` :`<error>`. Les erreurs possibles sont :

- IP_POOL_NOT_FOUND - le pool d'adresses IP SNAT est introuvable dans NSX Manager.
- IP_POOL_EXHAUSTED - le pool d'adresses IP SNAT est saturé.
- IP_POOL_NOT_UNIQUE - le pool spécifié par `ncp/snat_pool` fait référence à plusieurs pools dans NSX Manager.
- SNAT_POOL_ACCESS_DENY - la balise de propriétaire du pool ne correspond pas à l'espace de nom du service qui envoie la demande d'allocation.
- SNAT_RULE_OVERLAPPED - une nouvelle règle SNAT est créée, mais l'espace de service SNAT appartient également à un autre service SNAT, autrement dit, il existe plusieurs règles SNAT pour le même espace.
- POOL_ACCESS_DENIED : le pool d'adresses IP spécifié par `ncp/snat_pool` ne dispose pas de la balise `scope: ncp/owner, tag: cluster:<cluster_name>` ou la balise de propriétaire du pool ne correspond pas à l'espace de noms du service qui envoie la demande d'allocation.

Notez les points suivants :

- Le pool spécifié par `ncp/snat_pool` doit déjà exister dans NSX-T Data Center avant que le service ne soit configuré.
- Dans NSX-T Data Center, la priorité de la règle SNAT pour le service est supérieure à celle du projet.
- Si un espace est configuré avec plusieurs règles SNAT, une seule fonctionnera.
- Vous pouvez spécifier un pool d'adresses IP différent en modifiant l'annotation et en redémarrant NCP.

Configuration d'un pool d'adresses IP SNAT pour un espace de noms

Vous pouvez configurer un pool d'adresses IP SNAT pour un espace de noms en ajoutant une annotation à l'espace de noms. Par exemple,

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

NCP configurera la règle SNAT pour cet espace de noms. L'adresse IP source de la règle correspond à l'ensemble des espaces de serveur principal. L'adresse IP de destination est l'adresse IP SNAT allouée à partir du pool d'adresses IP externe spécifié. Si une erreur se produit lorsque NCP configure la règle SNAT, l'espace de noms sera annoté avec `ncp/error.snat`:<error>. Les erreurs possibles sont :

- IP_POOL_NOT_FOUND - le pool d'adresses IP SNAT est introuvable dans NSX Manager.
- IP_POOL_EXHAUSTED - le pool d'adresses IP SNAT est saturé.
- IP_POOL_NOT_UNIQUE - le pool spécifié par `ncp/snat_pool` fait référence à plusieurs pools dans NSX Manager.
- POOL_ACCESS_DENIED : le pool d'adresses IP spécifié par `ncp/snat_pool` ne dispose pas de la balise `scope: ncp/owner, tag: cluster:`<cluster_name> ou la balise de propriétaire du pool ne correspond pas à l'espace de noms qui envoie la demande d'allocation.

Notez les points suivants :

- Vous pouvez spécifier un seul pool d'adresses IP SNAT dans l'annotation.
- Le pool d'adresses IP SNAT n'a pas besoin d'être configuré dans `ncp.ini`.
- Le pool d'adresses IP spécifié par `ncp/snat_pool` doit avoir la balise `scope: ncp/owner, tag: cluster:`<cluster_name>.
- Le pool d'adresses IP spécifié par `ncp/snat_pool` peut également disposer d'une balise d'espace de noms `scope: ncp/owner, tag: ns:`<namespace_UUID>.
- Si l'annotation `ncp/snat_pool` est manquante, l'espace de noms utilisera le pool d'adresses IP SNAT pour le cluster.
- Vous pouvez spécifier un pool d'adresses IP différent en modifiant l'annotation et en redémarrant NCP.

Configuration d'un pool d'adresses IP SNAT pour une application PAS

Par défaut, NCP configure une adresse IP SNAT pour une organisation PAS (Pivotal Application Service). Vous pouvez configurer une adresse IP SNAT pour une application spécifique en créant une application avec un fichier `manifest.xml` qui contient les informations du pool d'adresses IP SNAT. Par exemple,

```
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 configurera la règle SNAT pour cette application. L'adresse IP source de la règle est l'ensemble des adresses IP des instances et son adresse IP de destination est l'adresse IP SNAT allouée à partir d'un pool d'adresses IP externe. Notez les points suivants :

- Le pool spécifié par `NCP_SNAT_POOL` doit déjà exister dans NSX-T Data Center avant que l'application ne soit transmise.
- La priorité d'une règle SNAT pour une application est supérieure à celle d'une règle SNAT pour une organisation.
- Une application peut être configurée avec une seule adresse IP SNAT.
- Vous pouvez spécifier un pool d'adresses IP différent en modifiant la configuration et en redémarrant NCP.

Configuration SNAT pour PCF version 3

Avec PCF version 3, vous pouvez configurer SNAT de l'une des deux manières suivantes :

- Lors de la création de l'application, configurez `NCP_SNAT_POOL` dans `manifest.yml`.

  Par exemple, l'application est appelée `bread` et `manifest.yml` contient les informations suivantes :

  ```
  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
```

Exécutez les commandes suivantes :

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

- Configurez NCP_SNAT_POOL à l'aide de la commande `cf v3-set-env`.

  Exécutez les commandes suivantes (en supposant que l'application est appelée app3) :

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

(Facultatif) (pour Kubernetes uniquement) Sections de marqueur de pare-feu

Pour permettre à l'administrateur de créer des règles de pare-feu et que celles-ci n'interfèrent pas avec les sections de pare-feu créées par NCP et basées sur des stratégies réseau, connectez-vous à NSX Manager, accédez à **Sécurité > Pare-feu distribué > Général** et créez deux sections de pare-feu.

Spécifiez les sections de pare-feu de marqueur en définissant les options `bottom_firewall_section_marker` et `top_firewall_section_marker` dans la section `[nsx_v3]` de `ncp.ini`.

La section de pare-feu inférieure doit se trouver sous la section de pare-feu supérieure. Une fois ces sections de pare-feu créées, toutes les sections de pare-feu créées par NCP pour une isolation sont créées au-dessus de la section de pare-feu inférieure, et toutes les sections de pare-feu créées par NCP pour une stratégie sont créées en dessous de la section de pare-feu supérieure. Si ces sections de marqueur ne sont pas créées, toutes les règles d'isolation sont créées en bas et toutes les sections de stratégie sont créées en haut. L'utilisation de plusieurs sections de pare-feu de marqueur possédant la même valeur par cluster n'est pas prise en charge et provoque une erreur.

Installation de NCP dans un environnement Kubernetes

3

L'installation de NSX Container Plug-in (NCP) requiert l'installation des composants sur les nœuds maître et Kubernetes.

Ce chapitre contient les rubriques suivantes :

- Installer le plug-in CNI NSX-T Data Center
- Installer et configurer OVS
- Configurer la mise en réseau NSX-T Data Center pour des nœuds Kubernetes
- Installer l'agent de nœud NSX
- ConfigMap pour ncp.ini dans nsx-node-agent-ds.yml
- Installation de NSX Container Plug-in
- ConfigMap pour ncp.ini dans ncp-rc.yml
- Monter un certificat codé au format PEM et une clé privée dans l'espace NCP
- Monter un fichier de certificat dans l'espace NCP
- Configuration de Syslog
- Considérations de sécurité
- Conseils sur la configuration des ressources réseau

Installer le plug-in CNI NSX-T Data Center

Le plug-in CNI NSX-T Data Center doit être installé sur les nœuds Kubernetes.

Pour Ubuntu, l'installation du plug-in CNI NSX-T copie le fichier du profil AppArmor `ncp-apparmor` dans `/etc/apparmor.d` et le charge. Avant l'installation, le service AppArmor doit être en cours d'exécution et le répertoire `/etc/apparmor.d` doit exister. Sinon, l'installation va échouer. Vous pouvez vérifier si le module AppArmor est activé avec la commande suivante :

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

Vous pouvez vérifier si le service AppArmor est démarré avec la commande suivante :

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

Si le service AppArmor n'est pas en cours d'exécution lorsque vous installez le plug-in CNI NSX-T, l'installation affiche le message suivant lorsqu'elle se termine :

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

Le message indique que toutes les étapes d'installation sont terminées, sauf le chargement du profil AppArmor.

Le fichier de profil `ncp-apparmor` fournit un profil AppArmor pour l'agent de nœud NSX nommé `node-agent-apparmor`, qui diffère du profil `docker-default` de la manière suivante :

- La règle `deny mount` est retirée.
- La règle `mount` est ajoutée.
- Certaines options `network`, `capability`, `file` et `umount` sont ajoutées.

Vous pouvez remplacer le profil `node-agent-apparmor` par un profil différent. Toutefois, le nom du profil `node-agent-apparmor` est référencé dans le fichier `nsx-node-agent-ds.yml`, qui est utilisé dans l'installation de l'agent de nœud NSX. Si vous utilisez un profil différent, vous devez spécifier le nom du profil dans `nsx-node-agent-ds.yml`, sous la section `spec:template:metadata:annotations`, dans l'entrée suivante :

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

Procédure

1. Téléchargez le fichier d'installation approprié pour votre distribution Linux.

   Le nom de fichier est `nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm` ou `nsx-cni-1.0.0.0.0.xxxxxxx.deb`, où *xxxxxxx* est le numéro de build.

2. Installez le fichier rpm ou deb téléchargé à l'étape 1.

   Le plug-in est installé dans `/opt/cni/bin`. Le fichier de configuration CNI 10-nsx.conf est copié dans `/etc/cni/net.d`. Le RPM installe également le fichier de configuration `/etc/cni/net.d/99-loopback.conf` pour le plug-in de bouclage.

Installer et configurer OVS

Installez et configurez OVS (Open vSwitch) sur les nœuds serviteurs.

Procédure

1 Téléchargez le fichier d'installation de votre distribution Linux.

Les noms de fichier sont `openvswitch-common_2.10.x.xxxxxxx-1_amd64.deb`, `openvswitch-datapath-dkms_2.10.x.xxxxxxx-1_all.deb` et `openvswitch-switch_2.10.x.xxxxxxx-1_amd64.deb`, où *xxxxxxx* est le numéro de build.

2 Installez le fichier deb téléchargé à l'étape 1.

3 Pour Ubuntu, exécutez la commande suivante pour recharger le module de noyau OVS.

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

4 Assurez-vous qu'OVS est en cours d'exécution.

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

5 Créez l'instance de *br-int* si elle n'est pas déjà créée.

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

6 Ajoutez l'interface réseau (*node-if*) qui est attachée au commutateur logique du nœud à *br-int*.

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

Exécutez la commande suivante pour afficher le contenu de `ofport`, car si `ofport` 1 n'est pas disponible, OVS attribuera un port qui est disponible.

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

Si `ofport` n'est pas égal à 1, définissez l'option `ovs_uplink_port` dans la section `nsx_kube_proxy` du fichier yaml DaemonSet de l'agent de nœud NSX en conséquence.

7 Vérifiez que l'état de *br-int* et *node-if link* est actif.

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

8 Mettez à jour le fichier de configuration réseau pour vous assurer que l'interface réseau est active après un redémarrage.

Pour Ubuntu, mettez à jour `/etc/network/interfaces` et ajoutez les lignes suivantes :

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

Pour RHEL, mettez à jour `/etc/sysconfig/network-scripts/ifcfg-<node-if>` et ajoutez la ligne suivante :

```
ONBOOT=yes
```

Configurer la mise en réseau NSX-T Data Center pour des nœuds Kubernetes

Cette section explique comment configurer la mise en réseau NSX-T Data Center pour des nœuds maître et travailleur Kubernetes.

Chaque nœud doit disposer d'au moins deux interfaces réseau. La première est une interface de gestion qui peut ou pas se trouver sur l'infrastructure NSX-T Data Center. Les autres interfaces fournissent une mise en réseau pour les espaces, se trouvent sur l'infrastructure NSX-T Data Center et sont connectées à un commutateur logique, désigné comme commutateur logique du nœud. Les adresses IP de gestion et d'espace doivent être routables pour que la vérification de l'intégrité Kubernetes fonctionne. Pour la communication entre l'interface de gestion et les espaces, NCP crée automatiquement une règle DFW pour permettre la vérification de l'intégrité et tout autre trafic de gestion. Vous pouvez voir les détails de cette règle dans l'interface utilisateur de NSX Manager. Cette règle ne doit pas être modifiée ou supprimée.

Pour chaque machine virtuelle du nœud, assurez-vous que la vNIC désignée pour la mise en réseau du conteneur est attachée au commutateur logique du nœud.

NSX Container Plug-in (NCP) doit connaître l'identifiant de VIF de la vNIC utilisée pour le trafic de conteneur dans chaque nœud. Le port du commutateur logique correspondant doit avoir les deux balises suivantes : Pour une balise, spécifiez le nom du nœud. Pour l'autre balise, spécifiez le nom du cluster. Pour la portée, spécifiez la valeur appropriée comme indiqué ci-dessous.

Balise	Portée
Nom du nœud	`ncp/node_name`
Nom du cluster	`ncp/cluster`

Vous pouvez identifier le port de commutateur logique pour une machine virtuelle du nœud en accédant à **Inventaire > Machines virtuelles** dans l'interface utilisateur de NSX Manager.

Si le nom du nœud Kubernetes change, vous devez mettre à jour la balise `ncp/node_name` et redémarrer NCP. Vous pouvez utiliser la commande suivante pour obtenir les noms de nœud :

```
kubectl get nodes
```

Si vous ajoutez un nœud à un cluster alors que NCP est en cours d'exécution, vous devez ajouter les balises au port de commutateur logique avant d'exécuter la commande `kubeadm join`. Sinon, le nouveau nœud ne disposera pas de connectivité réseau. Si les balises sont incorrectes ou manquantes, vous pouvez procéder comme suit pour résoudre ce problème :

- Appliquez les balises correctes au port de commutateur logique.

- Redémarrez NCP.

Installer l'agent de nœud NSX

L'agent de nœud NSX est un DaemonSet où chaque espace exécute deux conteneurs. Un conteneur exécute l'agent de nœud NSX, dont la responsabilité principale consiste à gérer les interfaces réseau de conteneur. Il interagit avec le plug-in CNI et Kubernetes API Server. L'autre conteneur exécute le proxy kube NSX, dont la seule responsabilité consiste à mettre en œuvre l'abstraction du service Kubernetes en convertissant les adresses IP du cluster en adresses IP d'espace. Il met en œuvre la même fonctionnalité que le proxy kube en amont.

Procédure

1. Téléchargez l'image Docker de NCP.

   Le nom de fichier est `nsx-ncp-xxxxxxx.tar`, où *xxxxxxx* est le numéro de build.

2. Téléchargez le modèle yaml DaemonSet de l'agent de nœud NSX.

   Le nom de fichier est `nsx-node-agent-ds.yml`. Vous pouvez modifier ce fichier ou l'utiliser comme exemple pour votre propre fichier de modèle.

3. Chargez l'image Docker de NCP dans votre registre d'images.

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

4. Modifiez le fichier `nsx-node-agent-ds.yml`.

   Changez le nom de l'image pour celui qui a été chargé.

   Pour Ubuntu, le fichier yaml suppose que AppArmor est activé. Pour voir si AppArmor est activé, vérifiez le fichier `/sys/module/apparmor/parameters/enabled`. Si AppArmor n'est pas activé, apportez les modifications suivantes :

   - Supprimez ou commentez la ligne suivante :

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

- Ajoutez la ligne `privileged:true` sous `securityContext` pour le conteneur nsx-node-agent et le conteneur nsx-kube-proxy. Par exemple :

```
securityContext:
    privileged:true
```

Note Il existe un problème selon lequel kubelet signale toujours AppArmor comme étant désactivé quel que soit l'état réel, si kubelet est exécuté à l'intérieur d'un conteneur qui utilise l'image hyperkube. Vous devez apporter les mêmes modifications mentionnées ci-dessus dans le fichier yaml.

Note Dans le fichier yaml, vous devez spécifier que le ConfigMap généré pour `ncp.ini` doit être monté en tant que volume en lecture seule. Le fichier yaml téléchargé a déjà cette spécification, qui ne doit pas être modifiée.

5 Créez le DaemonSet de l'agent de nœud NSX avec la commande suivante.

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

ConfigMap pour ncp.ini dans nsx-node-agent-ds.yml

L'exemple de fichier yaml `nsx-node-agent-ds.yml` contient un ConfigMap pour le fichier de configuration `ncp.ini` pour l'agent du nœud NSX. Cette section ConfigMap contient des paramètres que vous pouvez spécifier pour personnaliser votre installation de l'agent du nœud.

L'exemple de fichier `nsx-node-agent-ds.yml` que vous téléchargez contient les informations `ncp.ini` suivantes :

```
# 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>
```

Installation de NSX Container Plug-in

NSX Container Plug-in (NCP) est fourni sous forme d'image Docker. NCP doit s'exécuter sur un nœud des services d'infrastructure. Il n'est pas recommandé d'exécuter NCP sur le nœud maître.

Procédure

1. Téléchargez l'image Docker de NCP.

   Le nom de fichier est `nsx-ncp-xxxxxxx.tar`, où *xxxxxxx* est le numéro de build.

2. Téléchargez le modèle yaml ReplicationController de NCP.

   Le nom de fichier est `ncp-rc.yml`. Vous pouvez modifier ce fichier ou l'utiliser comme exemple pour votre propre fichier de modèle.

3. Chargez l'image Docker de NCP dans votre registre d'images.

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

4. (Facultatif) Téléchargez le modèle yaml pour la définition de la ressource personnalisée de l'objet NSXError.

   Le nom de fichier est `nsx-error-crd.yaml`.

5. (Facultatif) Créez la ressource personnalisée.

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

6. Modifiez `ncp-rc.yml`.

   Changez le nom de l'image pour celui qui a été chargé.

Spécifiez le paramètre `nsx_api_managers`. Vous pouvez spécifier l'adresse IP d'une instance unique de NSX Manager, ou les adresses IP (séparées par des virgules) de trois instances de NSX Manager dans un cluster NSX Manager ou l'adresse IP virtuelle d'un cluster NSX Manager. Par exemple :

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

(Facultatif) Spécifiez le paramètre `ca_file` dans la section `[nsx_v3]`. La valeur doit être un fichier de bundle d'autorité de certification à utiliser pour la vérification du certificat de serveur NSX Manager. Si aucune valeur n'est définie, les autorités de certification racines du système seront utilisées. Si vous spécifiez une adresse IP pour `nsx_api_managers`, spécifiez un fichier d'autorité de certification. Si vous spécifiez trois adresses IP pour `nsx_api_managers`, vous pouvez spécifier un ou trois fichiers d'autorité de certification. Si vous spécifiez un fichier d'autorité de certification, il sera utilisé pour les trois gestionnaires. Si vous spécifiez trois fichiers d'autorité de certification, chacun sera utilisé pour l'adresse IP correspondante dans `nsx_api_managers`. Par exemple,

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

Spécifiez les paramètres `nsx_api_cert_file` et `nsx_api_private_key_file` pour l'authentification avec NSX-T Data Center.

`nsx_api_cert_file` est le chemin d'accès complet vers un fichier de certificat client au format PEM. Le contenu de ce fichier devrait ressembler à ce qui suit :

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

nsx_api_private_key_file est le chemin d'accès complet vers un fichier de clé privée client au format PEM. Le contenu de ce fichier devrait ressembler à ce qui suit :

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

Spécifiez le paramètre `ingress_mode = nat` si le contrôleur d'entrée est configuré pour s'exécuter en mode NAT.

Par défaut, le préfixe de sous-réseau 24 est utilisé pour tous les sous-réseaux alloués à partir des blocs d'adresses IP pour les commutateurs logiques d'espace. Pour utiliser une taille de sous-réseau différente, mettez à jour l'option `subnet_prefix` dans la section [nsx_v3].

HA (haute disponibilité) est activée par défaut. Vous pouvez désactiver HA avec la spécification suivante :

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

(Facultatif) Activez la génération de rapports d'erreur à l'aide de NSXError dans `ncp.ini`. Ce paramètre est désactivé par défaut.

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

Note Dans le fichier yaml, vous devez spécifier que le ConfigMap généré pour `ncp.ini` doit être monté en tant que volume en lecture seule. Le fichier yaml téléchargé a déjà cette spécification, qui ne doit pas être modifiée.

7 Créez ReplicationController de NCP.

```
kubectl create –f ncp–rc.yml
```

Résultats

Note NCP ouvre des connexions HTTP persistantes au serveur Kubernetes API pour surveiller les événements de cycle de vie des ressources Kubernetes. Si une défaillance du serveur API ou une panne de réseau provoque l'obsolescence des connexions TCP de NCP, vous devez redémarrer NCP pour lui permettre de rétablir les connexions au serveur API. Sinon, NCP ratera les nouveaux événements.

Pendant une mise à jour continue du ReplicationController NCP, vous pouvez voir deux espaces NCP en cours d'exécution après la mise à jour continue dans les situations suivantes :

- Vous redémarrez l'hôte de conteneur au cours de la mise à jour continue.
- Initialement, la mise à jour continue échoue, car la nouvelle image n'existe pas sur un nœud Kubernetes. Pour réussir la mise à jour, téléchargez l'image et exécutez à nouveau la mise à jour continue.

Si vous voyez deux espaces NCP en cours d'exécution, procédez comme suit :

- Supprimez l'un des espaces NCP. Vous pouvez supprimer l'un ou l'autre indifféremment. Par exemple,

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

- Supprimez NCP ReplicationController. Par exemple,

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

ConfigMap pour ncp.ini dans ncp-rc.yml

L'exemple de fichier YAML `ncp-rc.yml` contient un ConfigMap pour le fichier de configuration `ncp.ini`. Cette section ConfigMap contient les paramètres que vous devez spécifier avant d'installer NCP, comme décrit dans la section précédente.

L'exemple de fichier `ncp-rc.yml` que vous téléchargez contient les informations `ncp.ini` suivantes :

```
# 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 unsest 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_passsword 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_passsword 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>
```

Monter un certificat codé au format PEM et une clé privée dans l'espace NCP

Si vous disposez d'un certificat codé au format PEM et d'une clé privée, vous pouvez mettre à jour la définition d'espace NCP dans le fichier yaml pour monter les secrets TLS dans l'espace NCP.

1 Créez un secret TLS pour le certificat et la clé privée.

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

2 Mettez à jour le fichier yaml de spécification d'espace NCP pour monter le secret sous forme de fichiers dans la spécification d'espace NCP.

```
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 Mettez à jour les options de nsx_v3 `nsx_api_cert_file` et `nsx_api_private_key_file` dans le fichier yaml.

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

Monter un fichier de certificat dans l'espace NCP

Si vous disposez d'un fichier de certificat dans le système de fichiers de nœud, vous pouvez mettre à jour la spécification d'espace NCP pour monter le fichier dans l'espace NCP.

Par exemple,

```
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>
```

Configuration de Syslog

Vous pouvez exécuter un agent Syslog, tel que rsyslog ou syslog-ng, dans un conteneur pour envoyer des journaux NCP et des composants associés à un serveur Syslog.

Les méthodes suivantes sont recommandées. Pour plus d'informations sur la journalisation dans Kubernetes, reportez-vous à la section https://kubernetes.io/docs/concepts/cluster-administration/logging.

- Créez un conteneur complémentaire qui s'exécute dans l'espace NCP ou l'espace nsx-node-agent.
- Exécutez un réplica DaemonSet sur chaque nœud.

Note Avec la méthode de conteneur complémentaire, les journaux du plug-in NSX CNI ne peuvent pas être envoyés au serveur Syslog, car le plug-in ne s'exécute pas dans un espace.

Créer un conteneur complémentaire pour Syslog

Vous pouvez configurer un conteneur complémentaire pour Syslog à exécuter dans le même espace que NCP. La procédure suivante suppose que l'image d'agent Syslog est exemple/rsyslog.

Procédure

1 Configurez des agents de nœud NCP et NSX pour vous connecter à un fichier.

Dans le fichier yaml pour les agents de nœud NCP et NSX, définissez le paramètre log_dir et spécifiez le volume à monter. Par exemple,

```
[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
```

Vous pouvez modifier le nom du fichier journal en définissant le paramètre `log_file`. Les noms par défaut sont `ncp.log`, `nsx_node_agent.log` et `nsx_kube_proxy.log`. Si l'option `log_dir` est définie sur un chemin différent de `/var/log/nsx-ujo`, un volume hostPath ou un volume emptyDir doit être créé et monté sur la spécification d'espace correspondante.

2 Assurez-vous que le chemin d'accès de l'hôte existe et est accessible en écriture par l'utilisateur `nsx-ncp`.

a Exécutez les commandes suivantes.

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

b Ajoutez l'utilisateur `nsx-ncp` ou définissez le mode de chemin d'accès de l'hôte sur 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 Dans le fichier yaml de spécification de l'espace NCP, ajoutez un ConfigMap pour Syslog. Par exemple,

```
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 Dans le fichier yaml de l'espace NCP, ajoutez le conteneur rsyslog et montez les volumes appropriés, où rsyslog peut trouver les données de configuration et lire les journaux d'autres conteneurs. Par exemple,

```
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>
```

Créer un réplica DaemonSet pour Syslog

Les journaux de tous les composants NCP peuvent être redirigés avec cette méthode. Les applications doivent être configurées pour se connecter à stderr, qui est activé par défaut. La procédure suivante suppose que l'image d'agent Syslog est exemple/rsyslog.

Procédure

1 Créez le fichier yaml DaemonSet. Par exemple,

```
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 Créez le DaemonSet.

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

Exemple : configuration de la rotation des journaux et de Syslog s'exécutant dans un conteneur complémentaire

La procédure suivante montre comment configurer la rotation des journaux et Syslog en cours d'exécution dans un conteneur complémentaire.

Création du répertoire de journaux et configuration de la rotation des journaux

- Créez le répertoire des journaux sur tous les nœuds, y compris le nœud master et remplacez son propriétaire par l'utilisateur ayant l'ID 1000.

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

- Configurez la rotation des journaux sur tous les nœuds pour le répertoire `/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
```

Création du contrôleur de réplication NCP

- Créez le fichier `ncp.ini` pour 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
```

- Créez le mappage de configuration à partir du fichier ini.

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

- Créez la configuration de NCP rsyslog.

```
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
```

- Créez le mappage de configuration à partir des éléments ci-dessus.

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

- Créez le contrôleur de réplication NCP avec le conteneur complémentaire `rsyslog`.

```
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
```

- Créez NCP avec la spécification ci-dessus.

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

Création de l'ensemble de démons de l'agent de nœud NSX

- Créez la configuration `rsyslog` pour les agents de nœud.

```
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
```

- Créez le `configmap` à partir de ce qui précède.

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

- Créez le DaemonSet avec le conteneur complémentaire `configmap`.

```
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
```

- Créez le DaemonSet.

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

Considérations de sécurité

Lorsque vous déployez NCP, il est important de prendre des mesures pour sécuriser les environnements Kubernetes et NSX-T Data Center.

Restreindre NCP pour ne fonctionner que sur des nœuds désignés

NCP a accès au plan de gestion de NSX-T Data Center et doit être restreint pour ne fonctionner que sur les nœuds d'infrastructure désignés. Vous pouvez identifier ces nœuds avec une étiquette appropriée. Un nodeSelector pour cette étiquette doit ensuite être appliqué au ReplicationController NCP specification/. Par exemple,

```
nodeSelector:
    nsx-infra: True
```

Vous pouvez également utiliser d'autres mécanismes, tels que l'affinité, pour attribuer des espaces à des nœuds. Pour plus d'informations, consultez https://kubernetes.io/docs/concepts/configuration/assign-pod-node.

Vérifier que le moteur de Docker est à jour

Docker publie périodiquement des mises à jour de sécurité. Une procédure automatique doit être implémentée pour appliquer ces mises à jour.

Interdire les capacités NET_ADMIN et NET_RAW de conteneurs non approuvés

Les capacités NET_ADMIN et NET_RAW de Linux peuvent être exploitées par des personnes mal intentionnées pour compromettre le réseau de l'espace. Vous devez désactiver ces deux capacités de conteneurs non approuvés. Par défaut, la capacité NET_ADMIN n'est pas accordée à un conteneur sans privilège. Faites attention si une spécification d'espace l'active explicitement ou définit le conteneur en mode privilégié. En outre, pour les conteneurs non approuvés, désactivez NET_RAW en spécifiant NET_RAW dans la liste des capacités abandonnées dans la configuration SecurityContext de la spécification du conteneur. Par exemple,

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

Contrôle d'accès basé sur les rôles

Kubernetes utilise des API de contrôle d'accès basé sur les rôles (RBAC) pour prendre les décisions d'autorisation, ce qui permet aux administrateurs de configurer des stratégies de manière dynamique. Pour plus d'informations, consultez https://kubernetes.io/docs/admin/authorization/rbac.

En général, l'administrateur de cluster est le seul utilisateur avec un accès et des rôles avec privilèges. Pour les comptes d'utilisateur et de service, la séparation des privilèges doit être respectée lorsque vous accordez l'accès.

Les directives suivantes sont recommandées :

- Restreindre l'accès aux jetons Kubernetes API aux espaces qui en ont besoin.
- Restreindre l'accès aux secrets TLS du certificat client NCP ConfigMap et NSX API à l'espace NCP.
- Bloquer l'accès à l'API de mise en réseau de Kubernetes depuis les espaces qui ne nécessitent pas ce type d'accès.
- Ajouter une stratégie RBAC Kubernetes pour spécifier les espaces qui peuvent avoir accès à l'API Kubernetes.

Stratégie RBAC recommandée pour l'espace NCP

Créez l'espace NCP sous un ServiceAccount et donnez à ce compte un ensemble de privilèges minimal. De plus, n'autorisez pas les autres espaces ou ReplicationControllers à accéder aux ConfigMap et Secrets TLS qui sont montés en tant que volumes pour ReplicationController NCP et l'agent de nœud NSX.

L'exemple suivant montre comment spécifier des rôles et des liaisons de rôle pour NCP :

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

---

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

---

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

```
      - get
      - watch
      - list
      - update
      - patch
  - apiGroups:
    - ""
    - extensions
    resources:
      - ingresses/status
      - services/status
    verbs:
      - replace
      - update
      - patch
---

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

---

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

```
    name: ncp-svc-account
    namespace: nsx-system
```

Note Le Secret TLS qui est créé à l'aide de l'API Kubernetes pour le certificat client NSX-T Data Center et la paire de clés privées sont accessibles à tous les espaces qui ont accès au serveur API Kubernetes. De même, quand un espace est créé sans compte de service, il se voit attribuer automatiquement le compte de service par défaut dans le même espace de noms, ce qui monte automatiquement le jeton pour accéder à l'API Kubernetes. Par conséquent, l'accès à ces jetons doit être limité aux espaces qui en ont besoin.

Conseils sur la configuration des ressources réseau

Lors de la configuration de certaines ressources réseau, vous devez connaître certaines restrictions.

Limites de balisage de NSX-T Data Center

NSX-T Data Center présente les limites suivantes pour le balisage d'un objet :

- L'étendue a une limite de 128 caractères.
- La balise a une limite de 256 caractères.
- Chaque objet peut contenir un maximum de 30 balises.

Ces limites peuvent entraîner des problèmes lorsque des annotations Kubernetes ou OpenShift sont copiées vers des étendues et des balises NSX-T Data Center et que les limites sont dépassées. Par exemple, si une balise est destinée à un port de commutateur et que la balise est utilisée dans une règle de pare-feu, la règle risque de ne pas être appliquée comme prévu, car la clé ou la valeur d'annotation a été tronquée lors de sa copie sur une étendue ou une balise.

Configuration de stratégies réseau

Les stratégies réseau sélectionnent les espaces ou les espaces de noms à l'aide de sélecteurs d'étiquette.

La prise en charge par NCP des stratégies réseau est la même que celle fournie par Kubernetes et dépend de la version de Kubernetes.

- Kubernetes 1.11 - vous pouvez spécifier les sélecteurs de règle suivants :
  - `podSelector` : cela permet de sélectionner tous les espaces qui sont dans l'espace de noms où la stratégie réseau est créée.
  - `namespaceSelector` : cela sélectionne tous les espaces de noms.
  - `podSelector` ET `namespaceSelector` : cela sélectionne tous les espaces qui se trouvent dans les espaces de noms sélectionnés par `namespaceSelector`.

    - `ipBlockSelector` : une stratégie réseau n'est pas valide si `ipBlockSelector` est combiné à `namespaceSelector` ou `podSelector`. Un `ipBlockSelector` doit être présent dans la spécification de stratégie elle-même.
- Kubernetes 1.10 - les clauses de règle dans la stratégie réseau peuvent contenir au maximum un sélecteur parmi `namespaceSelector`, `podSelector` et `ipBlock`.

Le serveur API Kubernetes n'effectue pas de validation d'une spécification de la stratégie réseau. Il est possible de créer une stratégie réseau qui n'est pas valide. NCP rejettera cette stratégie réseau. Si vous mettez à jour la stratégie réseau pour la rendre valide, NCP ne sera toujours pas en mesure de traiter la stratégie réseau. Vous devez supprimer la stratégie réseau et en recréer une avec une spécification valide.

Installation de NCP dans un environnement Pivotal Cloud Foundry

4

Pivotal Cloud Foundry (PCF) est un fournisseur de plate-forme en tant que service (PaaS) en libre accès. Vous pouvez installer NSX Container Plug-in (NCP) dans un environnement PCF pour fournir des services de mise en réseau.

Les machines virtuelles créées par le biais de Pivotal Ops Manager doivent avoir une connectivité de couche 3 au réseau de conteneur pour accéder aux fonctionnalités de NSX-T.

High Availability (HA) est automatiquement activé.

Note Lorsqu'une modification est apportée à un groupe de sécurité, vous devez retransférer toutes les applications auxquelles s'applique le groupe de sécurité. Cela peut se produire parce que le groupe de sécurité s'applique à l'espace d'exécution des applications ou que le groupe de sécurité est global.

Ce chapitre contient les rubriques suivantes :

- Installer NCP dans un environnement Pivotal Cloud Foundry

Installer NCP dans un environnement Pivotal Cloud Foundry

NCP est installé via l'interface utilisateur graphique de Pivotal Ops Manager.

Conditions préalables

Une nouvelle installation de Pivotal Ops Manager, NSX-T Data Center et Pivotal Application Service (PAS). Assurez-vous que l'installation est effectuée dans l'ordre suivant : Ops Manager, NSX-T Data Center, puis PAS. Pour plus d'informations, consultez la documentation Pivotal Cloud Foundry.

Procédure

1. Téléchargez le fichier d'installation de NCP pour PCF.

   Le nom de fichier est `VMware-NSX-T.<version>.<build>.pivotal`.

2. Connectez-vous à Manager Ops Pivotal en tant qu'administrateur.

3. Cliquez sur **Importer un produit**.

4 Sélectionnez le fichier qui a été téléchargé.

5 Cliquez sur la vignette **Ops Manager Director pour VMware vSphere**.

6 Dans l'onglet **Paramètres** pour **Configuration de vCenter**, sélectionnez **Mise en réseau NSX** et **Mode NSX**, sélectionnez **NSX-T**.

7 Dans le champ **Certificat CA NSX** , fournissez le certificat au format PEM.

8 Cliquez sur **Enregistrer**.

9 Cliquez sur **Tableau de bord de l'installation** dans le coin supérieur gauche afin de revenir au tableau de bord.

10 Cliquez sur la vignette **Pivotal Application Service**.

11 Dans l'onglet **Paramètres**, sélectionnez **Mise en réseau** dans le volet de navigation.

12 Sous **Container Network Interface Plugin**, sélectionnez **Externe**.

13 Cliquez sur **Tableau de bord de l'installation** dans le coin supérieur gauche afin de revenir au tableau de bord.

14 Cliquez sur **Enregistrer**.

15 Cliquez sur **Tableau de bord de l'installation** dans le coin supérieur gauche afin de revenir au tableau de bord.

16 Cliquez sur la vignette **VMware NSX-T**.

17 Entrez l'adresse de NSX Manager.

18 Sélectionnez la méthode d'authentification pour NSX Manager.

Option	Action
Authentification du certificat client	Fournissez le certificat et la clé privée pour NSX Manager.
Authentification de base avec nom d'utilisateur et mot de passe	Fournissez le nom d'utilisateur et le mot de passe de l'administrateur de NSX Manager.

19 Dans le champ **Certificat CA NSX Manager**, fournissez le certificat.

20 Cliquez sur **Enregistrer**.

21 Sélectionnez **NCP** dans le volet de navigation.

22 Entrez un nom dans le champ **Nom de fondation PAS**.

Cette chaîne identifie de façon unique une fondation PAS dans NSX API. Cette chaîne sert également de préfixe dans les noms des ressources NSX créées par NCP pour établir les fondations PAS.

23 Saisissez la **zone de transport de superposition**.

24 Saisissez le **routeur de niveau 0**.

25 Spécifiez une ou plusieurs valeurs dans le champ **Blocs d'adresses IP des réseaux de conteneurs**.

   a Cliquez sur **Ajouter**.

   b Entrez un nom dans le champ **Nom du bloc d'adresses IP**. Il peut s'agir d'un nouveau bloc d'adresses IP ou d'un bloc existant.

   c Uniquement dans le cas d'un nouveau bloc d'adresses IP, spécifiez le bloc au format CIDR, par exemple 10.1.0.0/16.

26 Spécifiez le préfixe de sous-réseau des réseaux de conteneurs.

27 Cliquez sur **Activer SNAT pour les réseaux de conteneurs** pour activer SNAT.

28 Spécifiez une ou plusieurs valeurs dans le champ **Pools d'adresses IP utilisés pour fournir l'adresse IP externe (NAT) des réseaux d'organisation**.

   a Cliquez sur **Ajouter**.

   b Entrez un nom dans le champ **Nom de pool d'adresses IP**. Il peut s'agir d'un nouveau pool d'adresses IP ou d'un pool existant.

   c Uniquement dans le cas d'un nouveau pool d'adresses IP, spécifiez les adresses IP en fournissant le CIDR et les plages d'adresses IP.

29 (Facultatif) Entrez une valeur dans le champ **Marqueur de section de pare-feu supérieure**.

30 (Facultatif) Entrez une valeur dans le champ **Marqueur de section de pare-feu inférieure**.

31 (Facultatif) Activez ou désactivez les options suivantes.

Option	Valeur par défaut
Consigner le trafic d'application abandonné	Désactivé. Si cette option est activée, le trafic abandonné en raison d'une règle de pare-feu sera consigné.
Activer le niveau de débogage pour la journalisation NCP	Activé.

32 Cliquez sur **Enregistrer**.

33 (Facultatif) Sélectionnez **Agent de nœud NSX** dans le volet de navigation.

   a Cochez **Activer le niveau de débogage de journalisation pour l'agent de nœud NSX** pour activer la journalisation de niveau de débogage.

   b Cliquez sur **Enregistrer**.

34 Cliquez sur **Tableau de bord de l'installation** dans le coin supérieur gauche afin de revenir au tableau de bord.

35 Cliquez sur **Appliquer les modifications**.

Équilibrage de charge

5

L'équilibreur de charge NSX-T Data Center est intégré à Kubernetes.

Ce chapitre contient les rubriques suivantes :

- Configuration de l'équilibrage de charge
- Contrôleurs d'entrée tiers

Configuration de l'équilibrage de charge

La configuration de l'équilibrage de charge implique la configuration d'un service Kubernetes LoadBalancer ou d'une ressource d'entrée et du contrôleur de réplication NCP.

Vous pouvez créer un équilibreur de charge de couche 4 en configurant un service Kubernetes de type équilibreur de charge et un équilibreur de charge de couche 7 en configurant une ressource d'entrée Kubernetes.

Pour configurer l'équilibrage de charge dans NCP, dans le fichier `ncp-rc.yml` :

1 Définissez `use_native_loadbalancer` = **`True`**.

2 (Facultatif) Définissez `pool_algorithm` sur **`ROUND_ROBIN`** ou **`LEAST_CONNECTION/IP_HASH`**. La valeur par défaut est **`ROUND_ROBIN`**.

3 (Facultatif) Définissez `service_size` = **`SMALL`**, **`MEDIUM`** ou **`LARGE`**. La valeur par défaut est **`SMALL`**.

Avec l'algorithme **`LEAST_CONNECTION/IP_HASH`**, le trafic à partir de la même adresse IP source est envoyé au même espace de serveur principal.

Pour plus de détails sur la prise en charge des équilibrages de charge NSX-T de différentes tailles, reportez-vous au *Guide d'administration de NSX-T Data Center*.

Après la création de l'équilibreur de charge, la taille d'équilibreur de charge ne peut pas être modifiée en mettant à jour le fichier de configuration. Elle peut être modifiée au moyen de l'interface utilisateur ou de l'API de NSX Manager.

Configuration de la persistance pour l'équilibrage de charge des couches 4 et 7

Vous pouvez spécifier un paramètre de persistance avec les paramètres `l4_persistence` et `l7_persistence` dans le ConfigMap de NCP. Les options disponibles pour la persistance de la couche 4 est l'adresse IP source. Les options disponibles pour la persistance de la couche 7 sont le cookie et l'adresse IP source. La valeur par défaut est <None>. Par exemple,

```
# 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
```

Pour un service Kubernetes LoadBalancer, vous pouvez également spécifier `sessionAffinity` sur la spécification de service pour configurer le comportement de persistance du service si la persistance de couche 4 globale est désactivée, c'est-à-dire que `l4_persistence` est défini sur <None>. Si `l4_persistence` est défini sur `source_ip`, `sessionAffinity` sur la spécification de service peut être utilisé pour personnaliser le délai d'expiration de la persistance pour le service. Le délai d'expiration de la persistance de couche 4 par défaut est de 10 800 secondes (identique à celui spécifié dans la documentation Kubernetes pour les services) (https://kubernetes.io/docs/

concepts/services-networking/service). Tous les services ayant un délai d'expiration de persistance par défaut partageront le même profil de persistance d'équilibreur de charge NSX-T. Un profil dédié sera créé pour chaque service avec un délai d'expiration de persistance non défini par défaut.

Note Si le service du serveur principal d'une entrée est un service de type LoadBalancer, le serveur virtuel de couche 4 pour le service et le serveur virtuel de couche 7 pour l'entrée ne peuvent pas avoir de paramètres de persistance différents, par exemple, `source_ip` pour la couche 4 et `cookie` pour la couche 7. Dans un tel scénario, les paramètres de persistance des deux serveurs virtuels doivent être les mêmes (`source_ip`, `cookie` ou `None`), ou l'un d'entre eux peut être `None` (l'autre peut alors être `source_ip` ou `cookie`). Exemple d'un tel scénario :

```
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
```

Entrée

- NSX-T Data Center crée un équilibreur de charge de couche 7 pour les entrées avec spécification TLS et un équilibreur de charge de couche 7 pour les entrées sans spécification TLS.
- Tous les entrées obtiennent une adresse IP unique.

- Une adresse IP est allouée à la ressource d'entrée à partir du pool d'adresses IP externe spécifié par l'option `external_ip_pools` dans la section `[nsx_v3]` de `ncp.ini`. L'équilibreur de charge est exposé sur cette adresse IP et les ports HTTP et HTTPS (80 et 443).
- Une adresse IP est allouée à la ressource d'entrée à partir du pool d'adresses IP externe spécifié par l'option `external_ip_pools_lb` dans la section `[nsx_v3]` de `ncp.ini`. Si l'option `external_ip_pools_lb` n'existe pas, le pool spécifié par `external_ip_pools` est utilisé. L'équilibreur de charge est exposé sur cette adresse IP et les ports HTTP et HTTPS (80 et 443).
- Vous pouvez spécifier un pool d'adresses IP différent en modifiant la configuration et en redémarrant NCP.
- Vous pouvez spécifier un certificat par défaut pour TLS. Voir ci-dessous pour plus d'informations sur la création et le placement d'un certificat dans l'espace NCP.
- Les entrées sans spécification TLS sont hébergées sur le serveur virtuel HTTP (port 80).
- Les entrées avec spécification TLS sont hébergées sur le serveur virtuel HTTPS (port 443). L'équilibreur de charge agit comme serveur SSL et termine la connexion SSL client.
- L'ordre de création des secrets et des entrées n'a pas d'importance. Si l'objet secret est présent et qu'une entrée y fait référence, le certificat est importé dans NSX-T Data Center. Si le secret est supprimé ou si la dernière entrée y faisant référence est supprimée, le certificat correspondant au secret est supprimé.
- La modification d'une entrée par l'ajout ou la suppression d'une section TLS n'est pas prise en charge. Lorsque la clé `tls` est supprimée de la spécification des entrées, les règles d'entrée sont transférées vers le serveur virtuel HTTP (port 80) à partir du serveur virtuel HTTPS (port 443). De même, lorsque la clé `tls` est ajoutée à la spécification des entrées, les règles d'entrée sont transférées vers le serveur virtuel HTTPS (port 443) à partir du serveur virtuel HTTP (port 80).
- En présence de règles en double dans les définitions d'entrée pour un cluster unique, seule la première règle s'applique.
- Une seule entrée avec serveur principal par défaut est prise en charge par cluster. Le trafic ne correspondant à aucune règle d'entrée est transféré au serveur principal par défaut.
- En présence de plusieurs entrées avec un serveur principal par défaut, seule la première est configurée. Les autres sont annotées avec une erreur.
- La correspondance d'URI à l'aide de caractère générique est prise en charge grâce à l'utilisation des caractères d'expression régulière « . » et « * ». Par exemple, le chemin d'accès « /coffee/.* » correspond à « /coffee/ » suivi de zéro, un ou plusieurs caractères, tels que « / coffee/ », « /coffee/a », « /coffee/b », mais pas « /coffee », « /coffeecup » ou « /coffeecup/a ».

Exemple de spécification d'entrée :

```
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
```

- Vous pouvez configurer la réécriture de demande d'URL en ajoutant une annotation à la ressource d'entrée. Par exemple,

```
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
```

  les chemins d'accès `/tea` et `/coffee` seront réécrits en / avant que l'URL ne soit envoyée au service principal.

- L'annotation d'entrée `kubernetes.io/ingress.allow-http` est prise en charge.
  - Si l'annotation est définie sur **`false`**, seules les règles HTTPS sont créées.
  - Si l'annotation est définie sur **`true`** ou si elle est manquante, les règles HTTP sont créées. Les règles HTTPS sont également créées si la section TLS est présente dans la spécification d'entrée.
- Les erreurs sont annotées dans la ressource d'entrée. L'erreur clé est `ncp/error.loadbalancer` et la clé d'avertissement est `ncp/warning.loadbalancer`. L'erreur et l'avertissement possibles sont :
  - `ncp/error.loadbalancer` : DEFAULT_BACKEND_IN_USE

Cette erreur indique qu'il existe déjà une entrée avec un serveur principal par défaut. L'entrée sera inactive. Il ne peut y avoir qu'un seul serveur principal par défaut pour un groupe d'entrées avec et sans TLS. Pour corriger l'erreur, supprimez et recréez l'entrée avec une spécification correcte.

- `ncp/warning.loadbalancer` : SECRET_NOT_FOUND

  Cette erreur indique que le secret indiqué dans la spécification d'entrée n'existe pas. L'entrée sera partiellement active. Pour corriger l'erreur, créez le secret manquant. Notez qu'une fois qu'un avertissement est dans l'annotation, il ne sera pas effacé pendant le cycle de vie de la ressource d'entrée.

- `ncp/warning.loadbalancer` : INVALID_INGRESS

  Cette erreur indique que l'une des conditions suivantes est vraie. L'entrée sera inactive. Pour corriger l'erreur, supprimez et recréez l'entrée avec une spécification correcte.

  - Une règle d'entrée est en conflit avec une autre règle d'entrée dans le même cluster Kubernetes.
  - L'annotation `allow-http` est définie sur **`False`** et l'entrée n'a pas de section TLS.
  - `host` et `path` ne sont pas spécifiés dans une règle d'entrée. Une telle règle d'entrée a la même fonctionnalité que le serveur principal par défaut d'entrée. Utilisez plutôt le serveur principal d'entrée par défaut.

Service de type LoadBalancer

- NSX-T Data Center crée un serveur virtuel et un pool d'équilibreur de charge de couche 4 pour chaque port de service.
- TCP et UDP sont pris en charge.
- Chaque service aura une adresse IP unique.
- Une adresse IP est allouée au service à partir d'un pool d'adresses IP externe basé sur le champ `loadBalancerIP` dans la définition LoadBalancer. Le champ `loadBalancerIP` peut être vide, avoir une adresse IP ou le nom ou l'ID d'un pool d'adresses IP. Si le champ `loadBalancerIP` est vide, l'adresse IP est allouée à partir du pool d'adresses IP externe spécifié par l'option `external_ip_pools_lb` dans la section `[nsx_v3]` de `ncp.ini`. Si l'option `external_ip_pools_lb` n'existe pas, le pool spécifié par `external_ip_pools` est utilisé. Le service de LoadBalancer est exposé sur cette adresse IP et sur le port de service.
- Vous pouvez spécifier un pool d'adresses IP différent en modifiant la configuration et en redémarrant NCP.
- Le pool d'adresses IP spécifié par `loadBalancerIP` doit avoir la balise `scope: ncp/owner, tag: cluster:<cluster_name>`.
- Les erreurs sont annotées dans un service. L'erreur clé est `ncp/error.loadbalancer`. Les erreurs possibles sont :
  - `ncp/error.loadbalancer` : IP_POOL_NOT_FOUND

Cette erreur indique que vous spécifiez `loadBalancerIP` : `<nsx-ip-pool>` mais `<nsx-ip-pool>` n'existe pas. Le service sera inactif. Pour corriger l'erreur, spécifiez un pool d'adresses IP valide, supprimez et recréez le service.

- `ncp/error.loadbalancer` : IP_POOL_EXHAUSTED

  Cette erreur indique que vous spécifiez `loadBalancerIP` : `<nsx-ip-pool>` mais le pool d'adresses IP a épuisé ses adresses IP. Le service sera inactif. Pour corriger l'erreur, spécifiez un pool d'adresses IP ayant des adresses IP disponibles, supprimez et recréez le service.

- `ncp/error.loadbalancer` : IP_POOL_NOT_UNIQUE

  Cette erreur indique que plusieurs pools d'adresses IP portent le nom spécifié par `loadBalancerIP` : `<nsx-ip-pool>`. Le service sera inactif.

- `ncp/error.loadbalancer` : POOL_ACCESS_DENIED

  Cette erreur indique que le pool d'adresses IP spécifié par `loadBalancerIP` n'a pas la balise `scope: ncp/owner, tag: cluster:<cluster_name>` ou le cluster spécifié dans la balise ne correspond pas au nom du cluster Kubernetes.

- `ncp/error.loadbalancer` : LB_VIP_CONFLICT

  Cette erreur indique que l'adresse IP dans le champ `loadBalancerIP` est la même que celle d'un service actif. Le service sera inactif.

- La mise à l'échelle automatique de l'équilibreur de charge de couche 4 est prise en charge. Si un service LoadBalancer Kubernetes est créé ou modifié pour qu'il requiert des serveurs virtuels supplémentaires et que l'équilibreur de charge de couche 4 existant n'a pas la capacité, un nouvel équilibreur de charge de couche 4 sera créé. NCP supprimera également les équilibreurs de charge de couche 4 qui n'ont plus aucun serveur virtuel associé. Cette fonctionnalité est activée par défaut. Vous pouvez la désactiver en définissant `l4_lb_auto_scaling` sur **`false`** dans le ConfigMap NCP.

Équilibreur de charge et stratégie réseau

Lorsque le trafic est transféré vers les espaces à partir du serveur virtuel d'équilibreur de charge NSX, l'adresse IP source est l'adresse IP du port de liaison montante du routeur de niveau 1. Cette adresse se trouve sur le réseau privé de transit de niveau 1. En conséquence, les stratégies réseaux basées sur CIDR peuvent interdire un trafic normalement autorisé. Pour éviter ce problème, la stratégie réseau doit être configurée de telle sorte que l'adresse IP du port de liaison montante du routeur de niveau 1 fasse partie du bloc CIDR autorisé. Cette adresse IP interne sera visible dans le champ `status.loadbalancer.ingress.ip` et sous forme d'annotation (`ncp/internal_ip_for_policy`) sur la ressource d'entrée.

Par exemple, si l'adresse IP externe du serveur virtuel est 4.4.0.5 et que l'adresse IP du port de liaison montante du routeur de niveau 1 interne est 100.64.224.11, l'état sera :

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

L'annotation sur la ressource d'entrée et le service de type LoadBalancer sera :

```
ncp/internal_ip_for_policy: 100.64.224.11
```

L'adresse IP 100.64.224.11 doit appartenir au CIDR autorisé dans le sélecteur `ipBlock` de la stratégie réseau. Par exemple,

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

Exemple de script permettant de générer un certificat signé par une autorité de certification

Le script ci-dessous génère un certificat signé par une autorité de certification et une clé privée stockés dans les fichiers <filename>.crt et <finename>.key, respectivement. La commande `genrsa` génère une clé d'autorité de certification. La clé d'autorité de certification doit être chiffrée. Vous pouvez spécifier une méthode de chiffrement avec la commande `aes256`, par exemple.

```
#!/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
```

Placer le certificat et la clé par défaut dans l'espace NCP

Une fois le certificat et la clé privée générés, placez-les dans le répertoire `/etc/nsx-ujo` sur la machine virtuelle hôte. En supposant que les fichiers de certificat et de clé sont nommés `lb-default.crt` et `lb-default.key`, respectivement, modifiez `ncp-rc.yaml` afin que ces fichiers, stockés sur l'hôte, soient placés dans l'espace. Par exemple,

```
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
```

Contrôleurs d'entrée tiers

Vous pouvez configurer NCP afin qu'il prenne en charge les contrôleurs d'entrée tiers.

Modification du fichier `ncp.ini`

Vous devez modifier le fichier de configuration `/var/vcap/data/jobs/ncp/xxxxxxxx/config/ncp.ini` (où xxxxxxxx est l'ID de déploiement BOSH). Ce fichier sera ensuite copié dans `rootfs` et utilisé par NCP chaque fois qu'il redémarre. Vous devez modifier le fichier sur chaque nœud master.

Important Les modifications apportées à `ncp.ini` ne sont pas persistantes à travers les mises à jour du cluster PKS. En cas de modifications via le fichier PKS et de mise à jour du déploiement de PKS, les modifications effectuées dans `ncp.ini` seront perdues.

Les options appropriées se trouvent dans la section `nsx_v3`.

- `use_native_loadbalancer` : si défini sur **Faux**, NCP ne traitera aucune entrée ni service des mises à jour de type LoadBalancer, indépendamment de ses annotations. Ce paramètre s'applique à l'ensemble du cluster PKS. La valeur par défaut est **Vrai**.

- `default_ingress_class_nsx` : si défini sur **`Vrai`**, NCP devient le contrôleur d'entrée par défaut et traitera les entrées annotées avec `kubernetes.io/ingress.class: "nsx"` et celles sans annotation. Si défini sur **`Faux`**, NCP ne traitera que les entrées annotées avec `kubernetes.io/ingress.class: "nsx"`. La valeur par défaut est **`Vrai`**.

Pour que NCP attribue une adresse IP flottante à l'espace du contrôleur NGINX et mette à jour l'état des entrées avec ce type d'adresse, procédez comme suit :

- Dans la section `k8s` de `ncp.ini`, définissez `ingress_mode=nat`.
- Ajoutez l'annotation `ncp/ingress-controller: "True"` à l'espace du contrôleur d'entrée NGINX.

NCP mettra à jour l'état des entrées comportant l'annotation `kubernetes.io/ingress.class: "nginx"` avec l'adresse IP flottante de l'espace du contrôleur d'entrée NGINX. Si `default_ingress_class_nsx=False`, NCP mettra également à jour l'état des entrées sans l'annotation `kubernetes.io/ingress.class` avec l'adresse IP flottante de l'espace du contrôleur d'entrée NGINX.

Remarque : même si l'espace du contrôleur d'entrée NGINX ne comporte pas l'annotation `ncp/ingress-controller: "True"`, NCP mettra à jour l'état des entrées mentionné ci-dessus sur `loadBalancer: {}`. Les entrées peuvent alors être bloquées dans une boucle dans laquelle le contrôleur NGINX met à jour l'état de l'entrée sur `loadBalancer: {ingress: [{ip: <IP>}]}` et NCP met à jour l'état de l'entrée sur `loadBalancer: {}`. Pour éviter cette situation, suivez les étapes décrites ci-dessous :

- Si le contrôleur d'entrée provient de https://github.com/kubernetes/ingress-nginx,
  - Dans le contrôleur d'entrée, remplacez `ingress-class` par une valeur autre que `"nginx"`.
  - S'il existe une entrée avec l'annotation `kubernetes.io/ingress-class: "nginx"`, remplacez celle-ci par une autre valeur.
  - Pour plus d'informations, reportez-vous à la section https://kubernetes.github.io/ingress-nginx/user-guide/multiple-ingress.
- Si le contrôleur d'entrée provient de https://github.com/nginxinc/kubernetes-ingress,
  - Dans le contrôleur d'entrée, remplacez `ingress-class` par une valeur autre que `"nginx"`.
  - S'il existe une entrée avec l'annotation `kubernetes.io/ingress-class: "nginx"`, remplacez celle-ci par une autre valeur.
  - Dans l'espace du contrôleur d'entrée, définissez `use-ingress-class-only` sur **`Vrai`**. Cela empêchera le contrôleur de mettre à jour les entrées sans l'annotation `kubernetes.io/ingress-class`.
  - Pour plus d'informations, reportez-vous à la section https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/multiple-ingress.md.

Scénario 1 : NCP traite les entrées, mais n'est pas le contrôleur d'entrée par défaut.

Suivez cette procédure pour permettre à NCP de traiter les entrées de classe `nsx`.

1 Modifiez la section `nsx_v3` dans `ncp.ini` sur chaque nœud master.

   a Définissez `default_ingress_class_nsx` sur **`Faux`**.

   b Laissez `use_native_loadbalancer` défini sur **`Vrai`**, la valeur par défaut.

2 Redémarrez NCP sur chaque nœud master. Cela peut entraîner un basculement du maître.

3 Annotez toutes les entrées que vous souhaitez que NCP gère avec `kubernetes.io/ingress.class: "nsx"`.

Scénario 2 : NCP est le contrôleur d'entrée par défaut.

Suivez cette procédure :

1 Il n'est pas nécessaire de modifier `ncp.ini`, mais veillez à ce que chaque entrée soit annotée.

2 Les entrées devant être traitées par NCP doivent être annotées avec **kubernetes.io/ingress.class : « nsx »**.

   Bien que NCP gère les entrées sans l'annotation `kubernetes.io/ingress.class`, dans le cas de plusieurs contrôleurs d'entrée, il est recommandé de toujours disposer de l'annotation `kubernetes.io/ingress.class` et non de s'appuyer sur le comportement par défaut du contrôleur d'entrée.

3 Les entrées devant être traitées par des contrôleurs d'entrée tiers doivent être annotées avec la valeur requise par ces contrôleurs d'entrée.

Important Sauf si l'objectif est de faire de NGINX le contrôleur d'entrée par défaut, n'utilisez pas **`nginx`** comme contrôleur d'entrée NGINX, car cela fera de lui le contrôleur d'entrée par défaut.

Scénario 3 : NCP ne traite aucune entrée, indépendamment de son annotation.

Suivez cette procédure :

1 Modifiez la section `nsx_v3` dans `ncp.ini` sur chaque nœud master.

   a Définissez `use_native_loadbalancer` sur **`Faux`**. La valeur de `default_ingress_class_nsx` est désormais impertinente.

2 Redémarrez NCP sur chaque nœud master. Cela peut entraîner un basculement du maître.

Notez que NCP ne traitera pas non plus les services de type LoadBalancer

Administration de NSX Container Plug-in

6

Vous pouvez administrer NSX Container Plug-in à partir de l'interface utilisateur NSX Manager ou à partir de l'interface de ligne de commande.

Note Si une machine virtuelle d'hôte de conteneur s'exécute sur ESXi 6.5 et si la machine virtuelle est migrée via vMotion vers un autre hôte ESXi 6.5, les conteneurs en cours d'exécution sur l'hôte de conteneur perdent la connectivité aux conteneurs en cours d'exécution sur d'autres hôtes de conteneur. Vous pouvez résoudre ce problème en déconnectant et reconnectant la carte réseau virtuelle de l'hôte de conteneur. Ce problème ne se produit pas avec ESXi 6.5 Update 1 ou version ultérieure.

Hyperbus réserve l'ID de VLAN 4094 sur l'hyperviseur pour la configuration PVLAN et cet ID ne peut pas être modifié. Pour éviter tout conflit de VLAN, ne configurez pas les commutateurs logiques de VLAN ou les vmknics VTEP avec les mêmes ID de VLAN.

Ce chapitre contient les rubriques suivantes :

- Affichage des informations d'erreur stockées dans la ressource Kubernetes NSXError
- Ports logiques attachés par CIF
- Commandes d'interface de ligne de commande
- Codes d'erreur

Affichage des informations d'erreur stockées dans la ressource Kubernetes NSXError

Pour chaque objet de ressource Kubernetes qui présente des pannes de serveur principal NSX, un objet NSXError est créé avec les informations d'erreur. Il y a également un objet d'erreur pour toutes les erreurs à l'échelle du cluster.

Cette fonctionnalité n'est pas activée par défaut. Pour l'activer, vous devez définir `enable_nsx_err_crd` sur True dans `ncp.ini` lorsque vous installez NCP.

Note Vous ne devez pas créer, mettre à jour ou supprimer d'objets NSXError.

Commandes d'affichage d'objets NSXError :

- `kubectl get nsxerrors`

  Répertorier tous les objets NSXError.

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

  Répertorier les objets NSXError liés à un type spécifique d'objets Kubernetes, par exemple, les objets de type `services`.

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

  Afficher les détails d'un objet NSXError.

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

  Permet de trouver l'objet NSXError associé à un service spécifique.

Lorsque vous affichez les détails d'un objet NSXError, la section spécification contient les informations importantes suivantes. Par exemple,

```
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'
```

Dans cet exemple, l'espace de noms est `default`. Le nom du service est `svc–1`. Le type de ressource kubernetes est `services`.

Dans cette version, les erreurs suivantes sont prises en charge par l'objet NSXError.

- La mise à l'échelle automatique n'a pas pu allouer d'équilibrages de charge supplémentaires en raison d'une limite de NSX Edge.
- Le nombre de serveurs virtuels d'équilibrage de charge dépasse la limite (la mise à l'échelle automatique n'est pas activée).
- Le nombre de pools de serveurs d'équilibrage de charge dépasse la limite.
- Le nombre de membres du pool de serveurs d'équilibrage de charge dépasse la limite d'équilibrage de charge ou de la limite de NSX Edge.
- Adresses IP flottantes épuisées lors du traitement d'un service de type LoadBalancer.

Ports logiques attachés par CIF

Les CIF (interfaces de conteneur) sont des interfaces réseau sur des conteneurs qui sont connectés à des ports logiques sur un commutateur. Ces ports sont appelés ports logiques attachés par CIF.

Vous pouvez gérer des ports logiques attachés par CIF depuis l'interface utilisateur de NSX Manager.

Gestion des ports logiques attachés par CIF

Accédez à **Mise en réseau > Commutation > Ports** pour voir tous les ports logiques, y compris les ports logiques attachés par CIF. Cliquez sur le lien d'attachement d'un port logique attaché par CIF pour afficher les informations de l'attachement Cliquez sur le lien du port logique pour ouvrir un volet de fenêtre avec quatre onglets : Présentation, Surveiller, Gérer et Éléments associés. Cliquez sur **Éléments associés > Ports logiques** pour voir le port logique associé sur un commutateur de liaison montante. Pour plus d'informations sur les ports de commutateur, consultez le *Guide d'administration de NSX-T*.

Outils de surveillance du réseau

Les outils suivants prennent en charge les ports logiques attachés par CIF. Pour plus d'informations sur ces outils, consultez le *Guide d'administration de NSX-T*.

- Traceflow
- Connexion de port
- IPFIX
- La mise en miroir de ports distants à l'aide de l'encapsulation GRE d'un port de commutateur logique qui se connecte à un conteneur est prise en charge. Pour plus d'informations, reportez-vous à la section « Comprendre le profil de commutation de mise en miroir de ports » dans le *Guide d'administration de NSX-T*. Toutefois, la mise en miroir de ports CIF-VIF n'est pas prise en charge via l'interface utilisateur du gestionnaire.

Commandes d'interface de ligne de commande

Pour exécuter des commandes d'interface de ligne de commande, connectez-vous au conteneur NSX Container Plug-in, ouvrez un terminal et exécutez la commande `nsxcli`.

Vous pouvez également obtenir l'invite d'interface de ligne de commande en exécutant la commande suivante sur un nœud :

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

Tableau 6-1. Commandes d'interface de ligne de commande pour le conteneur NCP

Type	Commande	Remarque
État	get ncp-master status	Pour Kubernetes et PCF.
État	get ncp-nsx status	Pour Kubernetes et PCF.
État	get ncp-watcher <watcher-name>	Pour Kubernetes et PCF.
État	get ncp-watchers	Pour Kubernetes et PCF.
État	get ncp-k8s-api-server status	Pour Kubernetes seulement.
État	check projects	Pour Kubernetes seulement.

Tableau 6-1. Commandes d'interface de ligne de commande pour le conteneur NCP (suite)

Type	Commande	Remarque
État	check project <nom_projet>	Pour Kubernetes seulement.
État	get ncp-bbs status	Pour PCF seulement.
Statut	get ncp-capi status	Pour PCF seulement.
Statut	get ncp-policy-server status	Pour PCF seulement.
Cache	get project-caches	Pour Kubernetes seulement.
Cache	get project-cache <project-name>	Pour Kubernetes seulement.
Cache	get namespace-caches	Pour Kubernetes seulement.
Cache	get namespace-cache <namespace-name>	Pour Kubernetes seulement.
Cache	get pod-caches	Pour Kubernetes seulement.
Cache	get pod-cache <pod-name>	Pour Kubernetes seulement.
Cache	get ingress-caches	Pour Kubernetes seulement.
Cache	get ingress-cache <nom_entrée>	Pour Kubernetes seulement.
Cache	get ingress-controllers	Pour Kubernetes seulement.
Cache	get ingress-controller <nom_contrôleur_entrée>	Pour Kubernetes seulement.
Cache	get network-policy-caches	Pour Kubernetes seulement.
Cache	get network-policy-cache <nom_espace>	Pour Kubernetes seulement.
Cache	get asg-caches	Pour PCF seulement.
Cache	get asg-cache <ID_ASG>	Pour PCF seulement.
Cache	get org-caches	Pour PCF seulement.
Cache	get org-cache <ID_organisation>	Pour PCF seulement.
Cache	get space-caches	Pour PCF seulement.
Cache	get space-cache <ID-espace>	Pour PCF seulement.
Cache	get app-caches	Pour PCF seulement.
Cache	get app-cache <ID_application>	Pour PCF seulement.
Cache	get instance-caches <ID_application>	Pour PCF seulement.
Cache	get instance-cache <ID_application> <ID_instance>	Pour PCF seulement.
Cache	get policy-caches	Pour PCF seulement.
Support	get ncp-log file <filename>	Pour Kubernetes et PCF.

Tableau 6-1. Commandes d'interface de ligne de commande pour le conteneur NCP (suite)

Type	Commande	Remarque
Support	get ncp-log-level	Pour Kubernetes et PCF.
Support	set ncp-log-level <niveau_journal>	Pour Kubernetes et PCF.
Support	get support-bundle file <filename>	Pour Kubernetes seulement.
Support	get node-agent-log file <filename>	Pour Kubernetes seulement.
Support	get node-agent-log file <filename> <node-name>	Pour Kubernetes seulement.

Tableau 6-2. Commandes d'interface de ligne de commande pour le conteneur de l'agent du nœud NSX

Type	Commande
État	get node-agent-hyperbus status
Cache	get container-cache <nom-conteneur>
Cache	get container-caches

Tableau 6-3. Commandes d'interface de ligne de commande pour le conteneur du proxy Kube NSX

Type	Commande
État	get ncp-k8s-api-server status
État	get kube-proxy-watcher <watcher-name>
État	get kube-proxy-watchers
État	dump ovs-flows

Commandes d'état pour le conteneur NCP

- Afficher l'état du nœud maître NCP

```
get ncp-master status
```

  Exemple :

```
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
```

- Afficher l'état de la connexion entre NCP et NSX Manager

```
get ncp-nsx status
```

Exemple :

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

- Afficher l'état de l'observateur de l'entrée, l'espace de noms, l'espace et le service

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

Exemple :

```
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
```

- Afficher l'état de la connexion entre NCP et Kubernetes API Server

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

  Exemple :

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

- Vérifier tous les projets ou un projet spécifique

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

  Exemple :

  ```
  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
  ```

- Vérifier l'état de connexion entre NCP et PCF BBS

  ```
  get ncp-bbs status
  ```

  Exemple :

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

- Vérifier l'état de connexion entre NCP et PCF CAPI

  ```
  get ncp-capi status
  ```

Exemple :

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

- Vérifier l'état de connexion entre NCP et le serveur de stratégie PCF

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

Exemple :

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

Commandes de cache pour le conteneur NCP

- Obtenir le cache interne pour des projets ou des espaces de noms

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

Exemple :

```
kubenode> get project-caches
    default:
        logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
        logical-switch:
            id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
            ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
            subnet: 10.0.0.0/24
            subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435

    kube-system:
        logical-router: 5032b299-acad-448e-a521-19d272a08c46
        logical-switch:
            id: 85233651-602d-445d-ab10-1c84096cc22a
            ip_pool_id: ab1c5b09-7004-4206-ac56-85d9d94bffa2
            subnet: 10.0.1.0/24
            subnet_id: 73e450af-b4b8-4a61-a6e3-c7ddd15ce751

    testns:
        ext_pool_id: 346a0f36-7b5a-4ecc-ad32-338dcb92316f
        labels:
            ns: myns
            project: myproject
        logical-router: 4dc8f8a9-69b4-4ff7-8fb7-d2625dc77efa
        logical-switch:
            id: 6111a99a-6e06-4faa-a131-649f10f7c815
            ip_pool_id: 51ca058d-c3dc-41fd-8f2d-e69006ab1b3d
            subnet: 50.0.2.0/24
            subnet_id: 34f79811-bd29-4048-a67d-67ceac97eb98
```

```
        project_nsgroup: 9606afee-6348-4780-9dbe-91abfd23e475
        snat_ip: 4.4.0.3


kubenode> get project-cache default
    logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
    logical-switch:
        id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
        ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
        subnet: 10.0.0.0/24
        subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435


kubenode> get namespace-caches
    default:
        logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
        logical-switch:
            id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
            ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
            subnet: 10.0.0.0/24
            subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435

    kube-system:
        logical-router: 5032b299-acad-448e-a521-19d272a08c46
        logical-switch:
            id: 85233651-602d-445d-ab10-1c84096cc22a
            ip_pool_id: ab1c5b09-7004-4206-ac56-85d9d94bffa2
            subnet: 10.0.1.0/24
            subnet_id: 73e450af-b4b8-4a61-a6e3-c7ddd15ce751

    testns:
        ext_pool_id: 346a0f36-7b5a-4ecc-ad32-338dcb92316f
        labels:
            ns: myns
            project: myproject
        logical-router: 4dc8f8a9-69b4-4ff7-8fb7-d2625dc77efa
        logical-switch:
            id: 6111a99a-6e06-4faa-a131-649f10f7c815
            ip_pool_id: 51ca058d-c3dc-41fd-8f2d-e69006ab1b3d
            subnet: 50.0.2.0/24
            subnet_id: 34f79811-bd29-4048-a67d-67ceac97eb98
        project_nsgroup: 9606afee-6348-4780-9dbe-91abfd23e475
        snat_ip: 4.4.0.3


kubenode> get namespace-cache default
    logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
    logical-switch:
        id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
        ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
        subnet: 10.0.0.0/24
        subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435
```

- Obtenir le cache interne pour des espaces

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

Exemple :

```
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
```

- Obtenir tous les caches d'entrée ou un cache spécifique

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

Exemple :

```
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
```

- Obtenir des informations sur tous les contrôleurs d'entrée ou sur un contrôleur spécifique, notamment les contrôleurs qui sont désactivés

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

Exemple :

```
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
```

- Obtenir les caches de stratégie réseau ou un cache spécifique

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

Exemple :

```
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
```

- Obtenir tous les caches ASG ou un cache spécifique

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

Exemple :

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


node> get asg-cache edc04715-d04c-4e63-abbc-db601a668db6
```

```
    fws_id: 3c66f40a-5378-46d7-a7e2-bee4ba72a4cc
    name: org-85_tcp_80_asg
    rules:
        destinations:
            66.10.10.0/24
        ports:
            80
        protocol: tcp
        rule_id: 4359
    running_default: False
    running_spaces:
        75bc164d-1214-46f9-80bb-456a8fbccbfd
    staging_default: False
    staging_spaces:
```

- Obtenir tous les caches d'organisation ou un cache spécifique

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

Exemple :

```
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
```

- Obtenir tous les caches d'espace ou un cache spécifique

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

Exemple :

```
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
```

- Obtenir tous les caches d'application ou un cache spécifique

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

Exemple :

```
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
```

- Obtenir tous les caches d'instance d'une application ou un cache d'instance spécifique

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

Exemple :

```
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
```

- Obtenir tous les caches de stratégie

```
get policy-caches
```

Exemple :

```
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
```

Commandes de support pour le conteneur NCP

- Enregistrer le bundle de support NCP dans le magasin de fichiers

  Le bundle de support est constitué des fichiers journaux de tous les conteneurs d'espaces avec l'étiquette **`tier:nsx-networking`**. Le fichier de bundle est au format tgz et enregistré dans le répertoire du magasin de fichiers par défaut d'interface de ligne de commande `/var/vmware/nsx/file-store`. Vous pouvez utiliser la commande de magasin de fichiers d'interface de ligne de commande pour copier le fichier de bundle sur un site distant.

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

Exemple :

```
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
```

- Enregistrer les journaux NCP dans le magasin de fichiers

  Le fichier journal est enregistré au format tgz dans le répertoire du magasin de fichiers par défaut d'interface de ligne de commande `/var/vmware/nsx/file-store`. Vous pouvez utiliser la commande de magasin de fichiers d'interface de ligne de commande pour copier le fichier de bundle sur un site distant.

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

  Exemple :

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

- Enregistrer les journaux de l'agent de nœud dans le magasin de fichiers

  Enregistrez les journaux de l'agent de nœud d'un seul nœud ou de tous les nœuds. Les journaux sont enregistrés au format tgz dans le répertoire du magasin de fichiers par défaut d'interface de ligne de commande `/var/vmware/nsx/file-store`. Vous pouvez utiliser la commande de magasin de fichiers d'interface de ligne de commande pour copier le fichier de bundle sur un site distant.

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

  Exemple :

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

- Obtenir et définir le niveau de journalisation

  Les niveaux de journalisation disponibles sont `NOTSET`, `DEBUG`, `INFO`, `WARNING`, `ERROR` et `CRITICAL`.

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

  Exemple :

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

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

Commandes d'état pour le conteneur de l'agent du nœud NSX

- Affichez l'état de la connexion entre l'agent de nœud et HyperBus sur ce nœud.

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

  Exemple :

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

Commandes de cache pour le conteneur de l'agent du nœud NSX

- Obtenir le cache interne pour les conteneurs d'agents du nœud NSX.

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

  Exemple :

  ```
  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
  ```

Commandes d'état pour le conteneur du proxy Kube NSX

- Afficher l'état de la connexion entre Kube Proxy et Kubernetes API Server

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

  Exemple :

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

- Afficher l'état de l'observateur Kube Proxy

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

Exemple :

```
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
```

- Vider les flux OVS sur un nœud

```
dump ovs-flows
```

Exemple :

```
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
```

Codes d'erreur

Cette section répertorie les codes d'erreur renvoyés par les différents composants.

Codes d'erreur NCP

Code d'erreur	Description
NCP00001	Configuration non valide
NCP00002	Échec de l'initialisation
NCP00003	État non valide
NCP00004	Adaptateur non valide
NCP00005	Certificat introuvable
NCP00006	Jeton introuvable
NCP00007	Configuration de NSX non valide
NCP00008	Balise NSX non valide
NCP00009	Échec de la connexion à NSX
NCP00010	Balise de nœud introuvable
NCP00011	Port de commutateur logique du nœud non valide
NCP00012	Échec de la mise à jour de la VIF parent
NCP00013	VLAN épuisé
NCP00014	Échec de la libération du VLAN
NCP00015	Pool d'adresses IP épuisé
NCP00016	Échec de la libération de l'adresse IP
NCP00017	Bloc d'adresses IP épuisé
NCP00018	Échec de la création du sous-réseau IP
NCP00019	Échec de la suppression du sous-réseau IP
NCP00020	Échec de la création du pool d'adresses IP
NCP00021	Échec de la suppression du pool d'adresses IP
NCP00022	Échec de la création du routeur logique

Code d'erreur	Description
NCP00023	Échec de la mise à jour du routeur logique
NCP00024	Échec de la suppression du routeur logique
NCP00025	Échec de la création du commutateur logique

Code d'erreur	Description
NCP00026	Échec de la mise à jour du commutateur logique
NCP00027	Échec de la suppression du commutateur logique
NCP00028	Échec de la création du port de routeur logique
NCP00029	Échec de la suppression du port de routeur logique
NCP00030	Échec de la création du port de commutateur logique
NCP00031	Échec de la mise à jour du port de commutateur logique
NCP00032	Échec de la suppression du port de commutateur logique
NCP00033	Stratégie réseau introuvable
NCP00034	Échec de la création du pare-feu
NCP00035	Échec de la lecture du pare-feu
NCP00036	Échec de la mise à jour du pare-feu
NCP00037	Échec de la suppression du pare-feu
NCP00038	Plusieurs pare-feu trouvés
NCP00039	Échec de la création du NSGroup
NCP00040	Échec de la suppression du NSGroup
NCP00041	Échec de la création de l'ensemble d'adresses IP
NCP00042	Échec de la mise à jour de l'ensemble d'adresses IP
NCP00043	Échec de la suppression de l'ensemble d'adresses IP
NCP00044	Échec de la création de la règle SNAT
NCP00045	Échec de la suppression de la règle SNAT
NCP00046	Échec de la connexion à l'API d'adaptateur
NCP00047	Exception de l'observateur d'adaptateur
NCP00048	Échec de la suppression du service d'équilibreur de charge

Code d'erreur	Description
NCP00049	Échec de la création du serveur virtuel d'équilibreur de charge
NCP00050	Échec de la mise à jour du serveur virtuel d'équilibreur de charge

Code d'erreur	Description
NCP00051	Échec de la suppression du serveur virtuel d'équilibreur de charge
NCP00052	Échec de la création du pool d'équilibreur de charge
NCP00053	Échec de la mise à jour du pool d'équilibreur de charge
NCP00054	Échec de la suppression du pool d'équilibreur de charge
NCP00055	Échec de la création de la règle d'équilibreur de charge
NCP00056	Échec de la mise à jour de la règle d'équilibreur de charge
NCP00057	Échec de la suppression de la règle d'équilibreur de charge
NCP00058	Échec de la libération de l'adresse IP de pool d'équilibreur de charge
NCP00059	Serveur virtuel d'équilibreur de charge et association de service introuvables
NCP00060	Échec de la mise à jour du NSGroup
NCP00061	Échec de l'obtention des règles de pare-feu
NCP00062	Aucun critère de NSGroup
NCP00063	VM de nœud introuvable
NCP00064	VIF de nœud introuvable
NCP00065	Échec de l'importation de certificat
NCP00066	Échec de l'annulation de l'importation de certificat
NCP00067	Échec de la mise à jour de la liaison SSL
NCP00068	Profil SSL introuvable
NCP00069	Pool d'adresses IP introuvable
NCP00070	Cluster Edge T0 introuvable
NCP00071	Échec de la mise à jour du pool d'adresses IP
NCP00072	Échec du répartiteur
NCP00073	Échec de la suppression de la règle NAT
NCP00074	Échec de l'obtention du port de routeur logique
NCP00075	Échec de la validation de la configuration de NSX

Code d'erreur	Description
NCP00076	Échec de la mise à jour de la règle SNAT
NCP00077	Chevauchement de la règle SNAT
NCP00078	Échec de l'ajout des points de terminaison d'équilibreur de charge
NCP00079	Échec de la mise à jour des points de terminaison d'équilibreur de charge
NCP00080	Échec de la création du pool de règles d'équilibreur de charge
NCP00081	Serveur virtuel d'équilibreur de charge introuvable
NCP00082	Échec de la lecture de l'ensemble d'adresses IP
NCP00083	Échec de l'obtention du pool SNAT
NCP00084	Échec de la création du service d'équilibreur de charge
NCP00085	Échec de la mise à jour du service d'équilibreur de charge
NCP00086	Échec de la mise à jour du port de routeur logique
NCP00087	Échec de l'initialisation de l'équilibreur de charge
NCP00088	Pool d'adresses IP non unique
NCP00089	Erreur de synchronisation du cache d'équilibreur de charge de couche 7
NCP00090	Le pool d'équilibreur de charge n'existe pas
NCP00091	Erreur d'initialisation du cache de la règle d'équilibreur de charge
NCP00092	Échec du processus SNAT
NCP00093	Erreur de certificat par défaut d'équilibreur de charge
NCP00094	Échec de la suppression du point de terminaison d'équilibreur de charge
NCP00095	Projet introuvable
NCP00096	Accès au pool refusé
NCP00097	Échec de l'obtention d'un service d'équilibreur de charge
NCP00098	Échec de la création d'un service d'équilibreur de charge
NCP00099	Erreur de synchronisation du cache de pool d'équilibreur de charge

Codes d'erreur de l'agent de nœud NSX

Code d'erreur	Description
NCP01001	Liaison montante OVS introuvable
NCP01002	Adresse MAC de l'hôte introuvable

Code d'erreur	Description
NCP01003	Échec de la création du port OVS
NCP01004	Aucune configuration de l'espace
NCP01005	Échec de la configuration de l'espace
NCP01006	Échec de l'annulation de la configuration de l'espace
NCP01007	Socket CNI introuvable
NCP01008	Échec de la connexion à CNI
NCP01009	Incompatibilité de la version de CNI
NCP01010	Échec de la réception du message CNI
NCP01011	Échec de la transmission du message CNI
NCP01012	Échec de la connexion à Hyperbus
NCP01013	Incompatibilité de la version d'Hyperbus
NCP01014	Échec de la réception du message Hyperbus
NCP01015	Échec de la transmission du message Hyperbus
NCP01016	Échec de l'envoi du paquet GARP
NCP01017	Échec de la configuration de l'interface

Codes d'erreur de nsx-kube-proxy

Code d'erreur	Description
NCP02001	Port de la passerelle de proxy non valide
NCP02002	Échec de la commande de proxy
NCP02003	Échec de la validation du proxy

Codes d'erreur de la CLI

Code d'erreur	Description
NCP03001	Échec du démarrage de la CLI
NCP03002	Échec de la création du socket de la CLI
NCP03003	Exception de socket de la CLI
NCP03004	Demande du client de CLI non valide
NCP03005	Échec de la transmission du serveur de CLI

Code d'erreur	Description
NCP03006	Échec de la réception du serveur de CLI
NCP03007	Échec de l'exécution de la commande de CLI

Codes d'erreur Kubernetes

Code d'erreur	Description
NCP05001	Échec de la connexion à Kubernetes
NCP05002	Configuration de Kubernetes non valide
NCP05003	Échec de la demande Kubernetes
NCP05004	Clé Kubernetes introuvable
NCP05005	Type Kubernetes introuvable
NCP05006	Exception de l'observateur Kubernetes
NCP05007	Longueur de la ressource Kubernetes non valide
NCP05008	Type de ressource Kubernetes non valide
NCP05009	Échec du handle de ressource Kubernetes
NCP05010	Échec du handle de service Kubernetes
NCP05011	Échec du handle de point de terminaison Kubernetes
NCP05012	Échec du handle d'entrée Kubernetes
NCP05013	Échec du handle de stratégie réseau Kubernetes
NCP05014	Échec du handle de nœud Kubernetes
NCP05015	Échec du handle d'espace de noms Kubernetes
NCP05016	Échec du handle d'espace Kubernetes
NCP05017	Échec du handle de secret Kubernetes
NCP05018	Échec du serveur principal Kubernetes par défaut
NCP05019	Expression de correspondance Kubernetes non prise en charge
NCP05020	Échec de la mise à jour de l'état Kubernetes
NCP05021	Échec de la mise à jour de l'annotation Kubernetes
NCP05022	Cache d'espace de noms Kubernetes introuvable
NCP05023	Secret Kubernetes introuvable

Code d'erreur	Description
NCP05024	Serveur principal Kubernetes par défaut en cours d'utilisation
NCP05025	Échec du handle de service Kubernetes LoadBalancer

Codes d'erreur Pivotal Cloud Foundry

Code d'erreur	Description
NCP06001	Échec de la connexion à PCF BBS
NCP06002	Échec de la connexion à PCF CAPI
NCP06006	Cache PCF introuvable
NCP06007	Domaine PCF inconnu
NCP06020	Échec de la connexion au serveur de la stratégie PCF
NCP06021	Échec du traitement de la stratégie PCF
NCP06030	Échec du traitement de l'événement PCF
NCP06031	Type d'événement PCF inattendu
NCP06032	Instance d'événement PCF inattendue
NCP06033	Échec de la suppression de la tâche PCF
NCP06034	Échec de l'accès au fichier PCF