

NSX Container Plug-in for OpenShift : インストー ルおよび管理ガイド

VMware NSX Container Plug-in 2.4
VMware NSX-T Data Center 2.4



vmware®

VMware Web サイトで最新の技術ドキュメントをご確認いただけます。

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

VMware の Web サイトでは、最新の製品アップデートを提供しています。

本書に関するご意見、ご要望をお寄せください。フィードバック送信先：

docfeedback@vmware.com

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

ヴィエムウェア株式会社
105-0013 東京都港区浜松町 1-30-5
浜松町スクエア 13F
www.vmware.com/jp

Copyright © 2017–2019 VMware, Inc. All rights reserved. [著作権および商標](#).

内容

NSX-T Container Plug-in for OpenShift : インストールおよび管理ガイド 4

1 NSX-T Container Plug-in の概要 5

互換性の要件 6

インストールの概要 6

NCP のアップグレード 6

2 NSX-T リソースのセットアップ 8

NSX-T リソースの設定 8

3 NCP のインストール 12

システム要件 12

Ansible の hosts ファイルの準備 13

4 ロード バランシング 18

ロード バランシングの設定 18

5 NSX Container Plug-in の管理 25

NSX Manager グラフィカル ユーザー インターフェイスでの IP ブロックの管理 25

NSX Manager グラフィカル ユーザー インターフェイスでの IP ブロックのサブネットの表示 26

CIF 接続論理ポート 26

CLI コマンド 27

エラー コード 38

NSX-T Container Plug-in for OpenShift : インストールおよび管理ガイド

このガイドでは、NSX Container Plug-in (NCP) をインストールして管理する方法を説明します。また、NSX-T Data Center と OpenShift を統合する方法についても説明します。

対象読者

このガイドは、システム管理者とネットワーク管理者を対象としています。NSX-T Data Center および OpenShift のインストールと管理について理解していることが前提となります。

VMware の技術ドキュメントの用語集

VMware は、新しい用語を集めた用語集を提供しています。VMware の技術ドキュメントで使用されている用語の定義については、<http://www.vmware.com/support/pubs> をご覧ください。

NSX-T Container Plug-in の概要

NSX Container Plug-in(NCP) は、NSX-T Data Center と Kubernetes などのコンテナ オーケストレータとの統合や、NSX-T Data Center と OpenShift などのコンテナベースの PaaS (Platform as a Service) ソフトウェア製品との統合を可能にします。このガイドでは、NCP と OpenShift のセットアップについて説明します。

NCP のメイン コンポーネントはコンテナで実行され、NSX Manager や OpenShift 制御プレーンと通信を行います。NCP は、コンテナや他のリソースに対する変更を監視し、NSX API を呼び出して、コンテナの論理ポート、スイッチ、ルーター、セキュリティ グループなどのネットワーク リソースを管理します。

NSX CNI プラグインは、各 OpenShift ノードで実行されます。コンテナのライフ サイクル イベントを監視し、コンテナ インターフェイスをゲスト vSwitch に接続します。プログラムによってゲスト vSwitch をタグ付けし、コンテナ インターフェイスと vNIC 間でコンテナ トラフィックを転送します。

NCP では、次の機能が提供されます。

- OpenShift クラスタに NSX-T 論理トポロジを自動的に作成し、OpenShift の名前空間に個別に論理ネットワークを作成します。
- OpenShift のポッドを論理ネットワークに接続し、IP アドレスと MAC アドレスを割り当てます。
- ネットワーク アドレス変換 (NAT) をサポートしているため、OpenShift の名前空間に個別の SNAT IP を割り当てることができます。

注: NAT を設定するとき、変換される IP アドレスの合計数は 1,000 を超えることはできません。

- NSX-T 分散ファイアウォールを使用した OpenShift ネットワーク ポリシーを実装。
 - 入力方向および出力方向でネットワーク ポリシーをサポート。
 - ネットワーク ポリシーで **IPBlock** セレクタをサポート。
 - ネットワーク ポリシーでラベル セレクタを指定する際に **matchLabels** と **matchExpression** をサポート。
- NSX-T レイヤー 7 ロード バランサを使用する OpenShift Route を実装。
 - TLS Edge ターミネーションを含む HTTP Route と HTTPS Route をサポート。
 - 代替バックエンドとワイルドカード サブドメインを含むルートをサポート。
- NSX-T 論理スイッチ ポート上に、名前空間、ポッド名、ポッドのラベル用のタグを作成し、管理者がタグ ベースで NSX-T Data Center のセキュリティ グループとポリシーを定義できるようにします。

本リリースでは、NCP は単一の OpenShift クラスタをサポートします。

この章には、次のトピックが含まれています。

- [互換性の要件](#)
- [インストールの概要](#)
- [NCP のアップグレード](#)

互換性の要件

NSX Container Plug-in (NCP) には、次の互換性の要件があります。

ソフトウェア製品	バージョン
NSX-T Data Center	2.3、2.4
コンテナ ホスト仮想マシン向けハイパーバイザー	<ul style="list-style-type: none"> ■ サポート対象の vSphere バージョン ■ RHEL KVM 7.4、7.5、7.6
コンテナ ホストのオペレーティング システム	RHEL 7.4、7.5、7.6
Platform as a Service	OpenShift 3.10、3.11
Container Host Open vSwitch	2.10.2 (NSX-T Data Center 2.4 パッケージに含まれます)

インストールの概要

以下の手順に従って、NCP のインストールと設定を行います。手順を正常に実行するには、NSX-T Data Center および OpenShift のインストールと管理について理解している必要があります。

- 1 NSX-T Data Center をインストールします。
- 2 オーバーレイ トランスポート ゾーンを作成します。
- 3 オーバーレイ 論理スイッチを作成し、ノードをスイッチに接続します。
- 4 Tier-0 論理ルーターを作成します。
- 5 ポッドの IP アドレス ブロックを作成します。
- 6 SNAT (送信元ネットワーク アドレス変換) 用の IP アドレス プールを作成します。
- 7 Ansible の hosts ファイルを準備します。
- 8 単一の Playbook を使用して NCP と OpenShift をインストールします。

NCP のアップグレード

このセクションでは、NSX Container Plug-in (NCP) を 2.4.0 にアップグレードする方法について説明します。

手順

- 1 CNI RPM パッケージ、NSX Node Agent DaemonSet、NCP ReplicationController をアップグレードします。

2 Ansible の hosts ファイルを準備します。

各ノードには、パラメータ **openshift_node_group_name** を指定する必要があります。次はその例です。

```
[nodes]
config-master.example.com openshift_hostname=config-master.example.com
openshift_node_group_name=config-master
```

3 (オプション) ロード バランシングを構成します。

LoadBalancer サービスの外部 IP アドレスに別の IP アドレス プールを指定する手順を追加します。次はその例です。

```
external_ip_pools_lb = <nsx ip pool name>
```

NSX-T リソースのセットアップ

OpenShift ノードにネットワーク機能を提供するには、NSX-T Data Center リソースを作成する必要があります。

NSX-T リソースの設定

設定が必要な NSX-T Data Center リソースには、オーバーレイ トランスポート ゾーン、Tier-0 論理ルーター、ノード仮想マシンに接続する論理スイッチ、Kubernetes ノードの IP アドレス ブロック、SNAT の IP アドレス プールなどがあります。

重要: NSX-T Data Center 2.4 以降で実行している場合は、[ネットワークとセキュリティの詳細設定] タブを使用して NSX-T リソースを設定する必要があります。

NSX Container Plug-in (NCP) 設定ファイル **ncp.ini** では、NSX-T Data Center リソースはそれらの UUID または名前を使用して指定されます。

オーバーレイ トランスポート ゾーン

NSX Manager にログインして、コンテナ ネットワークに使用されているオーバーレイ トランスポート ゾーンを検索するか、新規に作成します。

ncp.ini の **[nsx_v3]** セクションで **overlay_tz** オプションを設定して、クラスタのオーバーレイ トランスポート ゾーンを指定します。この手順はオプションです。**overlay_tz** を設定しない場合、NCP は Tier-0 ルーターからオーバーレイ トランスポート ゾーン ID を自動的に取得します。

Tier-0 論理ルーティング

NSX Manager にログインして、コンテナ ネットワークに使用されているルーターを検索するか、新規に作成します。

ncp.ini の **[nsx_v3]** セクションで **tier0_router** オプションを設定して、クラスタの Tier-0 論理ルーターを指定します。

注: ルーターは、アクティブ/スタンバイ モードで作成する必要があります。

論理スイッチ

ノードがデータトラフィック用に使用する vNIC は、オーバーレイ論理スイッチに接続する必要があります。ノードの管理インターフェイスは NSX-T Data Center に接続する必要はありませんが、接続するとセットアップが容易になります。論理スイッチを作成するには、NSX Manager にログインします。スイッチ上で論理ポートを作成し、ノードの vNIC を接続します。論理ポートには次のタグが必要です。

- タグ : `<cluster_name>`、スコープ : `ncp/cluster`
- タグ : `<node_name>`、スコープ : `ncp/node_name`

`<cluster_name>` 値は、`ncp.ini` の `[coe]` セクションにある `cluster` オプションの値と一致させる必要があります。

Kubernetes ポッドの IP アドレス ブロック

NSX Manager にログインして、1 つ以上の IP アドレス ブロックを作成します。IP アドレス ブロックを CIDR 形式で指定します。

`ncp.ini` の `[nsx_v3]` セクションで `container_ip_blocks` オプションを設定して、Kubernetes ポッドの IP アドレス ブロックを指定します。

SNAT 以外の名前空間に IP アドレス ブロックを作成することもできます。

`ncp.ini` の `[nsx_v3]` セクションで `no_snat_ip_blocks` オプションを設定して、SNAT 以外の IP アドレス ブロックを指定します。

NCP の実行中に SNAT 以外の IP アドレス ブロックを作成した場合には、NCP を再起動する必要があります。再起動しない場合、IP アドレス ブロックが枯渇するまで、NCP は共有の IP アドレス ブロックを使用し続けます。

注: IP アドレス ブロックを作成するときに、NCP 構成ファイル `ncp.ini` の `subnet_prefix` パラメータの値より大きいプリフィックスは使用しないでください。

SNAT の IP アドレス プール

SNAT ルールによってポッドの IP アドレスを変換する場合、または SNAT/DNAT ルールによって Ingress Controller を使用する場合、IP アドレス プールは OpenStack のフローティング IP アドレスと同様に、IP アドレスの割り当てに使用されます。これらの IP アドレスは、外部 IP アドレスともいいます。

複数の Kubernetes クラスタが同じ外部 IP アドレス プールを使用します。各 NCP インスタンスは、管理する Kubernetes クラスタにこのプールのサブセットを使用します。デフォルトでは、ポッドのサブネットと同じサブネット プリフィックスが使用されます。異なるサイズのサブネットを使用するには、`ncp.ini` の `[nsx_v3]` セクションにある `external_subnet_prefix` オプションを更新します。

NSX Manager にログインしてプールを作成するか、既存のプールを見つけます。

`ncp.ini` の `[nsx_v3]` セクションで `external_ip_pools` オプションを設定して、SNAT の IP アドレス プールを指定します。

サービスにアノテーションを追加して、特定のサービスの SNAT を設定することもできます。次はその例です。

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

NCP によって、このサービスの SNAT ルールが設定されます。このルールの送信元 IP アドレスは、一連のバックエンド ポッドです。宛先 IP アドレスは、指定した外部 IP アドレス プールから割り当てられた SNAT IP アドレスです。次の点に注意してください。

- **ncp/snat_pool** で指定された IP アドレス プールは、サービスを設定する前に NSX-T Data Center に配置しておく必要があります。IP アドレス プールには、タグ **{"ncp/owner": cluster:<cluster>}** を含める必要があります。
- NSX-T Data Center では、サービスの SNAT ルールの優先順位はプロジェクトよりも高くなります。
- ポッドに複数の SNAT ルールが設定されている場合は、1 つのルールのみが機能します。

次のタグを IP アドレス プールに追加することで、SNAT の IP アドレス プールから IP アドレスを割り当てる名前空間を指定できます。

- スコープ : **ncp/owner**、タグ : **ns:<namespace_UUID>**

名前空間 UUID は、次のいずれかのコマンドで取得できます。

```
oc get ns -o yaml
```

次の点に注意してください。

- 各タグには 1 つの UUID を指定する必要があります。同じプールに対して複数のタグを作成できます。
- 古いタグ ベースで一部の名前空間の IP アドレスが割り当てられた後にタグを変更すると、サービスの SNAT 設定が変更されるか NCP が再起動されるまで、割り当て済みの IP アドレスは再利用されません。
- 名前空間の所有者タグはオプションです。このタグがない場合、任意の名前空間に SNAT IP アドレス プールから IP アドレスを割り当てることができます。

(オプション) ファイアウォールのマーカー セクション

管理者が作成したファイアウォール ルールが、ネットワーク ポリシーに基づいて NCP で作成したファイアウォール セクションに干渉されないようにするには、NSX Manager にログインし、2 つのファイアウォール セクションを作成します。

ncp.ini の **[nsx_v3]** セクションで **bottom_firewall_section_marker** および **top_firewall_section_marker** オプションを設定して、マーカー ファイアウォール セクションを指定します。

最下位のファイアウォール セクションは、最上位のファイアウォール セクションの下に配置される必要があります。これらのファイアウォール セクションを作成することで、NCP が分離用に作成するすべてのファイアウォール セクションは、リストの最下位のファイアウォール セクションの上に作成され、NCP がポリシー用に作成するすべてのファイアウォール セクションは、最上位のファイアウォール セクションの下に作成されるようになります。マーカー セクションを作成しない場合、すべての分離ルールはリストの最下位に作成され、すべてのポリシー セクションは最上位に作成されます。同じ値を持つ複数のマーカー ファイアウォール セクションを 1 つのクラスタに設定することはできないため、エラーが発生します。

NCP のインストール

NSX Container Plug-in (NCP) は OpenShift と完全に統合されています。Ansible の hosts ファイルに必要なパラメータを追加して OpenShift をインストールすると、NCP は自動的にインストールされます。

この章には、次のトピックが含まれています。

- システム要件
- Ansible の hosts ファイルの準備

システム要件

OpenShift をインストールする前に、環境が特定の要件を満たしていることを確認します。

一般的な要件

- Ansible 2.4 以降。

仮想マシンの要件

OpenShift ノードの仮想マシンには 2 つの vNIC が必要です。

- 管理用 Tier-1 ルーターへのアップリンクを持つ論理スイッチに接続された管理用 vNIC。
- 特定の OpenShift ノードで実行されているすべてのポッドの親 VIF としてどのポートが使用されているかを NSX Container Plug-in (NCP) が認識できるようにするには、すべての仮想マシンの 2 番目の vNIC で NSX-T の次のタグが必要です。

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

ベア メタルのマシンの要件

- OpenShift ノードは NSX-T トランスポート ノードである必要があり、上記のタグは VIF ではなくトランスポート ノードに適用される必要があります。
- Ansible の hosts ファイルには次の設定が必要です。 `nsx_node_type='BAREMETAL'`

NSX-T の要件

- Tier-0 ルーター。
- オーバーレイ トランスポート ゾーン。
- ポッド ネットワーク用の IP アドレス ブロック。
- (オプション) ルーティングされた (非 NAT) ポッド ネットワーク用の IP アドレス ブロック。
- SNAT の IP アドレス プール。デフォルトでは、ポッド ネットワーク用の IP アドレス ブロックは NSX-T 内でのみルーティング可能です。NCP はこの IP アドレス プールを使用して外部への接続を提供します。
- (オプション) 上部と下部のファイアウォール セクション。NCP はこれら 2 つのセクションの間に Kubernetes ネットワーク ポリシー ルールを配置します。
- Open vSwitch および CNI プラグイン RPM は、OpenShift ノード仮想マシンからアクセス可能な HTTP サーバ上でホストされている必要があります。

NCP Docker イメージ

現在、NCP Docker イメージは公開されていません。ローカルのプライベート レジストリにイメージ **nsx-ncp** を含めるか、次の手順を実行します。

ansible-playbook [-i /path/to/inventory] playbooks/prerequisites.yml

すべてのノード：

```
docker load -i nsx-ncp-rhel-xxx.yyyyyyyy.tar
docker image tag registry.local/xxx.yyyyyyyy/nsx-ncp-rhel nsx-ncp
ansible-playbook [-i /path/to/inventory] playbooks/deploy_cluster.yml
```

Ansible の hosts ファイルの準備

NSX Container Plug-in (NCP) を OpenShift と統合するには、Ansible の hosts ファイルに NCP パラメータを指定する必要があります。

Ansible の hosts ファイルに次のパラメータを指定した後、OpenShift をインストールすると自動的に NCP がインストールされます。

- openshift_use_nsx=True
- openshift_use_openshift_sdn=False
- os_sdn_network_plugin_name='cni'
- nsx_openshift_cluster_name='ocp-cluster1'

(必須) 複数の Openshift/Kubernetes クラスタが同じ NSX Manager に接続できるため、これは必須です。

- **nsx_api_managers='10.10.10.10'**

(必須) NSX Manager の IP アドレス。NSX Manager クラスタの場合は、カンマ区切りの IP アドレスを指定します。

- **nsx_tier0_router='MyT0Router'**

(必須) プロジェクトの Tier-1 ルーターが接続する Tier-0 ルーターの名前または UUID。

- **nsx_overlay_transport_zone='my_overlay_tz'**

(必須) 論理スイッチの作成に使用されるオーバーレイ トランスポート ゾーンの名前または UUID。

- **nsx_container_ip_block='ip_block_for_my_ocp_cluster'**

(必須) NSX-T 上で設定されている IP アドレス ブロックの名前または UUID。この IP アドレス ブロックにプロジェクトごとのサブネットがあります。これらのネットワークは SNAT の背後にあり、ルーティングできません。

- **nsx_ovs_uplink_port='ens224'**

(必須) HOSTVM モードの場合。NSX-T には、OCP ノードでのポッド ネットワーク用に管理 vNIC とは別の vNIC が必要です。両方の vNIC を NSX-T 論理スイッチに接続することを強くお勧めします。2 番目の (非管理用の) vNIC をここに指定する必要があります。ベア メタルの場合は、このパラメータは必要ありません。

- **nsx_cni_url='http://myserver/nsx-cni.rpm'**

(必須) NCP がノードのブートストラップを行えるようになるまでの一時的な要件。**nsx-cni** を http サーバに配置する必要があります。

- **nsx_ovs_url='http://myserver/openvswitch.rpm'**

- **nsx_kmod_ovs_url='http://myserver/kmod-openvswitch.rpm'**

(必須) NCP がノードのブートストラップを行えるようになるまでの一時的なパラメータ。ベア メタルのセットアップでは無視できます。

- **nsx_node_type='HOSTVM'**

(オプション) デフォルトは **HOSTVM** です。OpenShift が仮想マシンで実行されていない場合は、**BAREMETAL** に設定します。

- **nsx_k8s_api_ip=192.168.10.10**

(オプション) 設定すると、NCP はこの IP アドレスと通信します。設定しないと、Kubernetes サービス IP アドレスと通信します。

- **nsx_k8s_api_port=192.168.10.10**

(オプション) Kubernetes サービスのデフォルトは 443 です。**nsx_k8s_api_ip** と組み合わせて使用して、マスター ノード IP アドレスを指定するには、8443 に設定します。

- **nsx_insecure_ssl=true**

(オプション) NSX Manager には信頼されていない証明書が付属しているため、デフォルトは **true** です。信頼されている証明書に変更した場合は **false** に設定できます。

- **nsx_api_user='admin'**
- **nsx_api_password='super_secret_password'**
- **nsx_subnet_prefix=24**

(オプション) デフォルトは 24 です。これは、OpenShift プロジェクトごとに割り当てられるサブネット サイズです。ポッドの数がサブネット サイズを超えると、同じサブネット サイズの新しい論理スイッチがプロジェクトに追加されます。

- **nsx_use_loadbalancer=true**

(オプション) デフォルトは **true** です。OpenShift ルートおよび LoadBalancer タイプのサービスに NSX-T ロード バランサを使用しない場合は、**false** に設定します。

- **nsx_lb_service_size='SMALL'**

(オプション) デフォルトは **SMALL** です。NSX Edge のサイズによっては、**MEDIUM** または **LARGE** も可能です。

- **nsx_no_snat_ip_block='router_ip_block_for_my_ocp_cluster'**

(オプション) **ncp/no_snat=true** アノテーションがプロジェクトまたは名前空間に適用されている場合、サブネットはこの IP アドレス ブロックから取得され、SNAT はありません。ルーティング可能であることが想定されます。

- **nsx_external_ip_pool='external_pool_for_snat'**

(必須) **nsx_external_ip_pool_lb** が定義されていない場合の SNAT およびロード バランサの IP アドレス プール。

- **nsx_external_ip_pool_lb='my_ip_pool_for_lb'**

(オプション) **Router** と **SvcTypeLB** に異なる IP アドレス プールが必要な場合に設定します。

- **nsx_top_fw_section='top_section'**

(オプション) Kubernetes ネットワーク ポリシー ルールは NSX-T ファイアウォール ルールに変換され、このセクションの下に配置されます。

- **nsx_bottom_fw_section='bottom_section'**

(オプション) Kubernetes ネットワーク ポリシー ルールは NSX-T ファイアウォール ルールに変換され、このセクションの上に配置されます。

- **nsx_api_cert='/path/to/cert/nsx.crt'**

- **nsx_api_private_key='/path/to/key/nsx.key'**

(オプション) 設定すると、**nsx_api_user** と **nsx_api_password** は無視されます。証明書を NSX-T にアップロードし、この証明書を使用して認証を行うプリンシパル ID ユーザーを手動で作成する必要があります。

- **nsx_lb_default_cert='/path/to/cert/nsx.crt'**

■ nsx_lb_default_key='/path/to/key/nsx.key

(オプション) NSX-T ロード バランサは、TLS ベースのルートの SNI を作成するためにデフォルトの証明書が必要とします。この証明書は、ルートが設定されていない場合にのみ表示されます。指定しない場合、自己署名証明書が生成されます。

Ansible の hosts ファイルのサンプル

```
[OSEv3:children]
masters
nodes
etcd

[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=origin

openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true',
'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]
openshift_master_htpasswd_users={'yasen' : 'password'}

openshift_master_default_subdomain=demo.corp.local
openshift_use_nsx=true
os_sdn_network_plugin_name=cni
openshift_use_openshift_sdn=false
openshift_node_sdn_mtu=1500

# NSX specific configuration
nsx_openshift_cluster_name='ocp-cluster1'
nsx_api_managers='192.168.110.201'
nsx_api_user='admin'
nsx_api_password='VMware1!'
nsx_tier0_router='DefaultT0Router'
nsx_overlay_transport_zone='overlay-tz'
nsx_container_ip_block='ocp-pod-networking'
nsx_no_snat_ip_block='ocp-nonat-pod-networking'
nsx_external_ip_pool='ocp-external'
nsx_top_fw_section='openshift-top'
nsx_bottom_fw_section='openshift-bottom'
nsx_ovs_uplink_port='ens224'
nsx_cni_url='http://1.1.1.1/nsx-cni-2.3.2.x86_64.rpm'
nsx_ovs_url='http://1.1.1.1/openvswitch-2.9.1.rhel75-1.x86_64.rpm'
nsx_kmod_ovs_url='http://1.1.1.1/kmod-openvswitch-2.9.1.rhel75-1.el7.x86_64.rpm'

[masters]
ocp-master.corp.local

[etcd]
ocp-master.corp.local

[nodes]
ocp-master.corp.local ansible_ssh_host=10.1.0.10 openshift_node_group_name='node-config-master'
```



```
ocp-node1.corp.local ansible_ssh_host=10.1.0.11 openshift_node_group_name='node-config-infra'  
ocp-node2.corp.local ansible_ssh_host=10.1.0.12 openshift_node_group_name='node-config-infra'  
ocp-node3.corp.local ansible_ssh_host=10.1.0.13 openshift_node_group_name='node-config-compute'  
ocp-node4.corp.local ansible_ssh_host=10.1.0.14 openshift_node_group_name='node-config-compute'
```

ロード バランシング

NSX-T Data Center ロード バランサは、OpenShift と連携し、OpenShift Router として機能します。

NCP は OpenShift Route とエンドポイントのイベントを監視し、ルートの仕様に基づいて、ロード バランサ上のロード バランシング ルールを設定します。結果的に、NSX-T Data Center ロード バランサはこのルールに基づき、適切なバックエンド ポッドにレイヤー 7 の受信トラフィックを転送します。

ロード バランシングの設定

ロード バランシングの設定には、Kubernetes LoadBalancer サービスや OpenShift Route の設定などが含まれます。また、NCP レプリケーション コントローラの設定も必要です。LoadBalancer サービスは、レイヤー 4 トラフィック向け、OpenShift Route は、レイヤー 7 トラフィック向けです。

Kubernetes LoadBalancer サービスを設定すると、設定した外部の IP アドレス ブロックの IP アドレスが割り当てられます。ロード バランサは、この IP アドレスとサービス ポートを通じて公開されます。LoadBalancer に定義されている **loadBalancerIP** を使用して、IP アドレス プールの名前または ID を指定できます。LoadBalancer サービスの IP アドレスは、この IP アドレス プールから割り当てられます。**loadBalancerIP** が空白の場合、IP アドレスは事前設定した外部 IP アドレス ブロックから割り当てられます。

loadBalancerIP で指定される IP アドレス プールには、タグ **{"ncp/owner": cluster:<cluster>}** を含める必要があります。

NSX-T Data Center のロード バランサを使用するには、ロード バランシングを NCP で設定する必要があります。**ncp_rc.yml** ファイルで次の操作を行います。

- 1 **use_native_loadbalancer** を **True** に設定します。
- 2 **pool_algorithm** を **WEIGHTED_ROUND_ROBIN** に設定します。
- 3 **lb_default_cert_path** と **lb_priv_key_path** が、それぞれ認証局 (CA) 署名証明書ファイルとプライベート キー ファイルの完全パス名になるように設定します。CA 署名証明書を生成するサンプル スクリプトについては、以降を参照してください。また、NCP ポッドに、デフォルトの証明書とキーをマウントします。手順については、以降を参照してください。

- 4 (オプション) パーシステンスの設定にパラメータ **l4_persistence** と **l7_persistence** を指定します。レイヤー 4 パーシステンスに使用可能なオプションは、送信元 IP アドレスです。レイヤー 7 パーシステンスに使用可能なオプションは、Cookie と送信元 IP アドレスです。デフォルトは **<None>** です。次はその例です。

```
# Choice of persistence type for ingress traffic through L7 Loadbalancer.
# Accepted values:
# 'cookie'
# 'source_ip'
l7_persistence = cookie

# Choice of persistence type for ingress traffic through L4 Loadbalancer.
# Accepted values:
# 'source_ip'
l4_persistence = source_ip
```

- 5 (オプション) **service_size** に **SMALL**、**MEDIUM**、または **LARGE** を設定します。デフォルトは **SMALL** です。
- 6 OpenShift 3.11 を実行している場合、次の設定を実行して、OpenShift が LoadBalancer サービスに IP アドレスを割り当てないようにする必要があります。
- **/etc/origin/master/master-config.yaml** ファイル中の **networkConfig** で、**ingressIPNetworkCIDR** を **0.0.0.0/32** に設定します。
 - 次のコマンドを使用して API サーバとコントローラを再起動します。

```
master-restart api
master-restart controllers
```

Kubernetes LoadBalancer サービスの場合は、サービス仕様に **sessionAffinity** を指定して、グローバルレイヤー 4 のパーシステンスがオフの場合、つまり **l4_persistence** が **<None>** に設定されている場合のサービスのパーシステンス動作を設定することもできます。**l4_persistence** が **source_ip** に設定されている場合、サービスのパーシステンス タイムアウトをカスタマイズするために、サービス仕様の **sessionAffinity** を使用できます。デフォルトのレイヤー 4 のパーシステンス タイムアウトは 10,800 秒です。これは、サービスの Kubernetes

ドキュメントに指定されているものと同じです (<https://kubernetes.io/docs/concepts/services-networking/service> を参照)。デフォルトのパーシステンス タイムアウトが設定されているサービスはすべて、同じ NSX-T ロード バランサのパーシステンス プロファイルを共有します。デフォルト以外のパーシステンス タイムアウトを使用して、各サービスに専用プロファイルが作成されます。

注: Ingress のバックエンド サービスが LoadBalancer タイプのサービスである場合、サービスのレイヤー 4 仮想サーバと Ingress のレイヤー 7 仮想サーバに異なるパーシステンスを設定することはできません。たとえば、レイヤー 4 に **source_ip**、レイヤー 7 に **cookie** を設定することはできません。このようなシナリオでは、両方の仮想サーバのパーシステンス設定を同じ (**source_ip**、**cookie**、または **None**) にするか、1 つを **None** (その他の設定は **source_ip** または **cookie**) にする必要があります。シナリオの例：

```
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
```

ルーターのシャーディング

NCP は常に TLS Edge ターミネーションと HTTP ルートを処理し、名前空間や名前空間ラベルに関係なく、TLS パススルー ルートおよび TLS 再暗号化ルートをスキップします。OpenShift ルーターが TLS 再暗号化およびパススルー ルートのみを処理するように制限するには、次の手順を実行する必要があります。

- Openshift ルーターに名前空間ラベル セレクタを追加します。
- ターゲット名前空間に名前空間ラベルを追加します。

- ターゲット名前空間に TLS 再暗号化/パススルー ルートを作成します。

たとえば、名前空間ラベル セレクタを使用してルーターを設定するには、次のコマンドを実行します（ルーターのサービス アカウント名が **router** であると仮定します）。

```
oc set env dc/router NAMESPACE_LABELS="router=r1"
```

ルーターは選択された名前空間からルート进行处理ようになります。このセレクタを名前空間と一致させるには、次のコマンドを実行します（名前空間は **ns1** であると仮定します）。

```
oc label namespace ns1 "router=r1"
```

レイヤー 7 ロード バランサの例

次の YAML ファイルでは、レイヤー 7 ロード バランシングを提供する 2 つのレプリケーション コントローラ（tea-rc および coffee-rc）、2 つのサービス（tea-svc および coffee-svc、2 つのルート（cafe-route-multi および cafe-route））を設定しています。

```
# RC
apiVersion: v1
kind: ReplicationController
metadata:
  name: tea-rc
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
      - name: tea
        image: nginxdemos/hello
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: coffee-rc
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
      - name: coffee
        image: nginxdemos/hello
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
```

```
---
# Services
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
    app: tea
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
      name: http
  selector:
    app: tea
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
  labels:
    app: coffee
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
      name: http
  selector:
    app: coffee
---
# Routes
apiVersion: v1
kind: Route
metadata:
  name: cafe-route-multi
spec:
  host: www.cafe.com
  path: /drinks
  to:
    kind: Service
    name: tea-svc
    weight: 1
  alternateBackends:
    - kind: Service
      name: coffee-svc
      weight: 2
---
apiVersion: v1
kind: Route
metadata:
  name: cafe-route
spec:
  host: www.cafe.com
```

```
path: /tea-svc
to:
  kind: Service
  name: tea-svc
  weight: 1
```

補注

- HTTPS トラフィックでは Edge ターミネーションのみがサポートされます。
- ワイルドカードを使用するサブドメインの指定がサポートされています。たとえば、**Subdomain** に **wildcardPolicy** が設定されていて、ホスト名が **wildcard.example.com** に設定されている場合、***.example.com** へのすべての要求に対してサービスが提供されます。
- 設定の誤りが原因で、Route イベントの処理中に NCP でエラーが発生する場合は、Route の YAML ファイルを修正し、Route リソースを削除して再作成する必要があります。
- NCP では、名前空間によるホスト名の所有は適用されません。
- Kubernetes クラスタあたり 1 つの LoadBalancer サービスがサポートされます。
- NSX-T Data Center は、各 LoadBalancer サービス ポートにレイヤー 4 のロード バランサ仮想サーバおよびプールを作成します。TCP および UDP の両方がサポートされます。
- NSX-T Data Center ロード バランサには、異なるサイズを使用できます。NSX-T Data Center のロード バランサの構成については、『NSX-T Data Center 管理ガイド』を参照してください。
ロード バランサが作成された後に、構成ファイルを更新してロード バランサのサイズを変更することはできません。変更するには、ユーザー インターフェイスまたは API を使用します。
- レイヤー 4 ロード バランサの自動スケーリングがサポートされます。Kubernetes LoadBalancer サービスが追加の仮想サーバを必要とするように作成または変更されており、既存のレイヤー 4 ロード バランサに十分な容量がない場合、新しいレイヤー 4 ロード バランサが作成されます。また、NCP は、仮想サーバが接続されていないレイヤー 4 ロード バランサも削除します。この機能はデフォルトで有効になっています。NCP ConfigMap で **l4_lb_auto_scaling** を **False** に設定することで無効にできます。

CA 署名証明書を作成するサンプル スクリプト

次のスクリプトによって、CA 署名証明書とプライベート キーが生成され、それぞれファイル <filename>.crt および <filename>.key として格納されます。**genrsa** コマンドで CA キーを生成します。CA キーは暗号化する必要があります。**aes256** などのコマンドを使用して、暗号化方式を指定できます。

```
#!/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
```

デフォルトの証明書とキーの NCP ポッドへのマウント

証明書とプライベート キーは生成後、ホスト仮想マシンのディレクトリ `/etc/nsx-ujo` に配置されます。証明書とキー ファイルの名前がそれぞれ `lb-default.crt`、`lb-default.key` の場合、以下のように `nep-rc.yaml` を編集して、ホスト上のこれらのファイルが、ポッドにマウントされるようにします。次はその例です。

```
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
```


NSX Container Plug-in の管理

NSX Container Plug-in は、NSX Manager グラフィカル ユーザー インターフェイスまたはコマンドライン インターフェイス (CLI) から管理できます。

注: ESXi 6.5 で稼働中のコンテナ ホストの仮想マシンを、別の ESXi 6.5 ホストに vMotion で移行する場合、移行元のコンテナ ホスト上のコンテナから、移行先のコンテナ ホスト上のコンテナへの接続は失われます。この問題は、コンテナ ホストの vNIC を切断し、再度接続することで解決できます。この問題は、ESXi 6.5 Update 1 以降では発生しません。

Hyperbus では、ハイパーバイザーの VLAN ID 4094 は PVLAN の設定用に予約されており、ID を変更することはできません。VLAN の競合を避けるため、同じ VLAN ID の VLAN 論理スイッチまたは VTEP vmknics は設定しないでください。

この章には、次のトピックが含まれています。

- [NSX Manager グラフィカル ユーザー インターフェイスでの IP ブロックの管理](#)
- [NSX Manager グラフィカル ユーザー インターフェイスでの IP ブロックのサブネットの表示](#)
- [CIF 接続論理ポート](#)
- [CLI コマンド](#)
- [エラー コード](#)

NSX Manager グラフィカル ユーザー インターフェイスでの IP ブロックの管理

NSX Manager グラフィカル ユーザー インターフェイスで、IP ブロックのタグを追加、削除、編集、表示、管理できます。詳細を確認することもできます。

手順

- 1 ブラウザから NSX Manager (<https://<nsx-manager-IP-address-or-domain-name>>) にログインします。
- 2 [ネットワーク] > [IP アドレス管理] の順に移動します。
既存の IP アドレスのブロックの一覧が表示されます。

3 次のいずれかのアクションを実行します。

オプション	アクション
IP ブロックを追加する	[追加] をクリックします。
1 つ以上の IP ブロックを削除する	1 つ以上の IP ブロックを選択し、[削除] をクリックします。
IP ブロックを編集する	IP ブロックを選択して [編集] をクリックします。
IP ブロックの詳細を表示する	IP ブロック名をクリックします。全般的な情報を表示するには、[概要] タブをクリックします。この IP ブロックのサブネットを表示するには、[サブネット] タブをクリックします。
IP ブロックのタグを管理します。	IP ブロックを選択し、[アクション] - [タグの管理] の順にクリックします。

サブネットが割り当てられた IP ブロックは削除できません。

NSX Manager グラフィカル ユーザー インターフェイスでの IP ブロックのサブネットの表示

NSX Manager グラフィカル ユーザー インターフェイスで IP ブロックのサブネットを表示できます。NCP をインストールして実行した後に IP ブロックを追加または削除することは推奨されません。

手順

- 1 ブラウザから NSX Manager (<https://<nsx-manager-IP-address-or-domain-name>>) にログインします。
- 2 [ネットワーク] > [IP アドレス管理] の順に移動します。
既存の IP アドレスのブロックの一覧が表示されます。
- 3 IP ブロックの名前をクリックします。
- 4 [サブネット] タブをクリックします。

CIF 接続論理ポート

CIF (コンテナ インターフェイス) は、スイッチ上の論理ポートに接続されているコンテナのネットワーク インターフェイスです。これらのポートは CIF 接続論理ポートといいます。

NSX Manager グラフィカル ユーザー インターフェイスから CIF 接続論理ポートを管理できます。

CIF 接続論理ポートの管理

[ネットワーク] - [スイッチング] - [ポート] の順に移動して、CIF 接続論理ポートを含むすべての論理ポートを表示します。CIF 接続論理ポートの接続リンクをクリックして、接続情報を表示します。論理ポートのリンクをクリックすると、ウィンドウ ペインが開き、[概要]、[監視]、[管理]、[関連] の 4 つのタブが表示されます。[関連] - [論理ポート] の順にクリックすると、アップリンク スイッチに関連する論理ポートが表示されます。スイッチ ポートの詳細については、『NSX-T 管理ガイド』を参照してください。

ネットワーク監視ツール

次のツールは、CIF 接続論理ポートをサポートします。これらのツールの詳細については、『NSX-T 管理ガイド』を参照してください。

- トレースフロー
- ポート接続
- IPFIX
- コンテナに接続している論理スイッチ ポートの GRE カプセル化を使用して、リモート ポート ミラーリングを実行できます。詳細については、『NSX-T 管理ガイド』の「ポート ミラーリング スイッチング プロファイルについて」を参照してください。ただし、CIF ポートから VIF ポートへのミラーリングは、マネージャのユーザー インターフェイス経由ではサポートされません。

CLI コマンド

CLI コマンドを実行するには、NSX Container Plug-in コンテナにログインしてターミナルを開き、**nsxcli** コマンドを実行します。

ノードで次のコマンドを実行して、CLI プロンプトを表示することもできます。

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

表 5-1. NCP コンテナの CLI コマンド

タイプ	コマンド
状態	get ncp-master status
状態	get ncp-nsx status
状態	get ncp-watcher <watcher-name>
状態	get ncp-watchers
状態	get ncp-k8s-api-server status
状態	check projects
状態	check project <project-name>
キャッシュ	get project-cache <project-name>
キャッシュ	get project-caches
キャッシュ	get namespace-cache <namespace-name>
キャッシュ	get namespace-caches
キャッシュ	get pod-cache <pod-name>
キャッシュ	get pod-caches
キャッシュ	get ingress-caches
キャッシュ	get ingress-cache <ingress-name>
キャッシュ	get ingress-controllers

表 5-1. NCP コンテナの CLI コマンド (続き)

タイプ	コマンド
キャッシュ	get ingress-controller <ingress-controller-name>
キャッシュ	get network-policy-caches
キャッシュ	get network-policy-cache <pod-name>
サポート	get ncp-log file <filename>
サポート	get ncp-log-level
サポート	set ncp-log-level <log-level>
サポート	get support-bundle file <filename>
サポート	get node-agent-log file <filename>
サポート	get node-agent-log file <filename> <node-name>

表 5-2. NSX Node Agent コンテナの CLI コマンド

タイプ	コマンド
状態	get node-agent-hyperbus status
キャッシュ	get container-cache <container-name>
キャッシュ	get container-caches

表 5-3. NSX Kube-Proxy コンテナの CLI コマンド

タイプ	コマンド
状態	get ncp-k8s-api-server status
状態	get kube-proxy-watcher <watcher-name>
状態	get kube-proxy-watchers
状態	dump ovs-flows

NCP コンテナの状態コマンド

- NCP マスターの状態を表示します。

```
get ncp-master status
```

例 :

```
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
```

- NCP と NSX Manager の間の接続の状態を表示します。

```
get ncp-nsx status
```

例 :

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

- Ingress、名前空間、ポッド、サービスの監視の状態を表示します。

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

例 1 :

```
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
```

例 2 :

```
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
```

- NCP と Kubernetes API サーバ間の接続の状態を表示します。

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

例 :

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

- すべてのプロジェクトまたは特定のプロジェクトを確認します。

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

例 :

```
kubecall> 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

kubecall> check project default
  Tier-1 link port for router 1b90a61f-0f2c-4768-9eb6-ea8954b4f327 is missing
  Switch 40a6829d-c3aa-4e17-ae8a-7f7910fdf2c6 is missing
```

NCP コンテナのキャッシュ コマンド

- プロジェクトまたは名前空間の内部キャッシュを取得します

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

例:

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

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

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

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

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

kube-system:
  logical-router: 5032b299-acad-448e-a521-19d272a08c46
  logical-switch:
    id: 85233651-602d-445d-ab10-1c84096cc22a
    ip_pool_id: ab1c5b09-7004-4206-ac56-85d9d94bffa2
```

```

        subnet: 10.0.1.0/24
        subnet_id: 73e450af-b4b8-4a61-a6e3-c7ddd15ce751

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

kubnode> 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

```

- ポッドの内部キャッシュを取得します

```

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

```

例 :

```

kubnode> get pod-caches
nsx.default.nginx-rc-uw2lv:
  cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
  gateway_ip: 10.0.0.1
  host_vif: d6210773-5c07-4817-98db-451bd1f01937
  id: 1c8b5c52-3795-11e8-ab42-005056b198fb
  ingress_controller: False
  ip: 10.0.0.2/24
  labels:
    app: nginx
  mac: 02:50:56:00:08:00
  port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
  vlan: 1

nsx.testns.web-pod-1:
  cif_id: ce134f21-6be5-43fe-afbf-aaca8c06b5cf
  gateway_ip: 50.0.2.1
  host_vif: d6210773-5c07-4817-98db-451bd1f01937
  id: 3180b521-270e-11e8-ab42-005056b198fb
  ingress_controller: False
  ip: 50.0.2.3/24
  labels:

```



```

    app: nginx-new
    role: db
    tier: cache
  mac: 02:50:56:00:20:02
  port_id: 81bc2b8e-d902-4cad-9fc1-aabdc32ecaf8
  vlan: 3

kubenode> get pod-cache nsx.default.nginx-rc-ug2lv
  cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
  gateway_ip: 10.0.0.1
  host_vif: d6210773-5c07-4817-98db-451bd1f01937
  id: 1c8b5c52-3795-11e8-ab42-005056b198fb
  ingress_controller: False
  ip: 10.0.0.2/24
  labels:
    app: nginx
  mac: 02:50:56:00:08:00
  port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
  vlan: 1

```

- ネットワーク ポリシー キャッシュまたは特定のポリシー キャッシュを取得します。

```

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

```

例 :

```

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

```

```

        key: tier
        operator: DoesNotExist
    matchLabels:
        ns: myns
    ports:
        port: 80
        protocol: TCP
    src_rules_hash: e4ea7b8d91c1e722670a59f971f8fcc1a5ac51f1

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

```

NCP コンテナのサポート コマンド

- ファイルストアに NCP サポート バンドルを保存します。

サポートバンドルには、**tier:nsx-networking** というラベルが付いており、ポッド内のすべてのコンテナのログファイルが含まれます。バンドルファイルはtgz形式で、CLIのデフォルトのファイルストアディレクトリ **/var/vmware/nsx/file-store** に保存されます。CLIのfile-storeコマンドを使用すると、バンドルファイルをリモートサイトにコピーできます。

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

例：

```
kubenode>get support-bundle file foo
Bundle file foo created in tgz format
kubenode>copy file foo url scp://nicira@10.0.0.1:/tmp
```

- NCP ログをファイルストアに保存します。

ログファイルはtgz形式で、CLIのデフォルトのファイルストアディレクトリ **/var/vmware/nsx/file-store** に保存されます。CLIのfile-storeコマンドを使用すると、バンドルファイルをリモートサイトにコピーできます。

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

例：

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

- Node Agent ログをファイルストアに保存します。

1台のノードまたはすべてのノードのNode Agentログを保存します。ログはtgz形式で、CLIのデフォルトのファイルストアディレクトリ **/var/vmware/nsx/file-store** に保存されます。CLIのfile-storeコマンドを使用すると、バンドルファイルをリモートサイトにコピーできます。

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

例：

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

- ログレベルを取得して設定します。

使用可能なログレベルはNOTSET、DEBUG、INFO、WARNING、ERROR、およびCRITICALです。

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

例 :

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

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

NSX Node Agent コンテナの状態コマンド

- このノードの Node Agent と HyperBus 間の接続の状態を表示します。

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

例 :

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

NSX Node Agent コンテナのキャッシュ コマンド

- NSX Node Agent コンテナの内部キャッシュを取得します。

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

例 1 :

```
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
```

例 2 :

```
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
```

NSX Kube-Proxy コンテナの状態コマンド

- Kube プロキシと Kubernetes API サーバ間の接続の状態を表示します。

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

例 :

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

- Kube プロキシの監視の状態を表示します。

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

例 1 :

```
kubepod> 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
```

例 2 :

```
kubepod> 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
```

- ノードの OVS フローのダンプ

```
dump ovs-flows
```

例 :

```
kubecall> 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
```

エラー コード

このセクションでは、さまざまなコンポーネントによって生成されるエラー コードについて説明します。

NCP (NSX Container Plug-in) エラー コード

エラー コード	説明
NCP00001	設定が無効です
NCP00002	初期化に失敗しました
NCP00003	状態が無効です
NCP00004	アダプタが無効です
NCP00005	証明書が見つかりません
NCP00006	トークンが見つかりません
NCP00007	NSX の設定が無効です
NCP00008	NSX のタグが無効です
NCP00009	NSX の接続に失敗しました
NCP00010	ノード タグが見つかりません
NCP00011	ノードの論理スイッチ ポートが無効です
NCP00012	親 VIF の更新に失敗しました
NCP00013	VLAN が不足しています
NCP00014	VLAN の解放に失敗しました
NCP00015	IP アドレス プールが不足しています
NCP00016	IP アドレスの解放に失敗しました
NCP00017	IP アドレス ブロックが不足しています

エラー コード	説明
NCP00018	IP サブネットの作成に失敗しました
NCP00019	IP サブネットの削除に失敗しました
NCP00020	IP アドレス プールの作成に失敗しました
NCP00021	IP アドレス プールの削除に失敗しました
NCP00022	論理ルーターの作成に失敗しました
NCP00023	論理ルーターの更新に失敗しました
NCP00024	論理ルーターの削除に失敗しました
NCP00025	論理スイッチの作成に失敗しました

エラー コード	説明
NCP00026	論理スイッチの更新に失敗しました
NCP00027	論理スイッチの削除に失敗しました
NCP00028	論理ルーター ポートの作成に失敗しました
NCP00029	論理ルーター ポートの削除に失敗しました
NCP00030	論理スイッチ ポートの作成に失敗しました
NCP00031	論理スイッチ ポートの更新に失敗しました
NCP00032	論理スイッチ ポートの削除に失敗しました
NCP00033	ネットワーク ポリシーが見つかりません
NCP00034	ファイアウォールの作成に失敗しました
NCP00035	ファイアウォールの読み取りに失敗しました
NCP00036	ファイアウォールの更新に失敗しました
NCP00037	ファイアウォールの削除に失敗しました
NCP00038	複数のファイアウォールが見つかりました
NCP00039	NSGroup の作成に失敗しました
NCP00040	NSGroup の削除に失敗しました
NCP00041	IP セットの作成に失敗しました
NCP00042	IP セットの更新に失敗しました
NCP00043	IP セットの削除に失敗しました
NCP00044	SNAT ルールの作成に失敗しました
NCP00045	SNAT ルールの削除に失敗しました
NCP00046	アダプタ API による接続に失敗しました
NCP00047	アダプタ ウォッチャの例外です
NCP00048	ロード バランサ サービスの削除に失敗しました
NCP00049	ロード バランサ仮想サーバの作成に失敗しました
NCP00050	ロード バランサ仮想サーバの更新に失敗しました

エラー コード	説明
NCP00051	ロード バランサ仮想サーバの削除に失敗しました
NCP00052	ロード バランサ プールの作成に失敗しました
NCP00053	ロード バランサ プールの更新に失敗しました
NCP00054	ロード バランサ プールの削除に失敗しました
NCP00055	ロード バランサ ルールの作成に失敗しました
NCP00056	ロード バランサ ルールの更新に失敗しました
NCP00057	ロード バランサ ルールの削除に失敗しました
NCP00058	ロード バランサ プールの IP アドレスの解放に失敗しました
NCP00059	ロード バランサ仮想サーバとサービスの関連付けが見つかりません
NCP00060	NSGroup の更新に失敗しました
NCP00061	ファイアウォール ルールの取得に失敗しました
NCP00062	NSGroup の条件がありません
NCP00063	ノード仮想マシンが見つかりません
NCP00064	ノード VIF が見つかりません
NCP00065	証明書のインポートに失敗しました
NCP00066	証明書のアンインポートに失敗しました
NCP00067	SSL バインドの更新に失敗しました
NCP00068	SSL プロファイルが見つかりません
NCP00069	IP アドレス プールが見つかりません
NCP00070	T0 Edge クラスタが見つかりません
NCP00071	IP アドレス プールの更新に失敗しました
NCP00072	ディスパッチャに失敗しました
NCP00073	NAT ルールの削除に失敗しました
NCP00074	論理ルーター ポートの取得に失敗しました
NCP00075	NSX の設定の確認に失敗しました

エラー コード	説明
NCP00076	SNAT ルールの更新に失敗しました
NCP00077	SNAT ルールが重複しています
NCP00078	ロード バランサのエンドポイントの追加に失敗しました
NCP00079	ロード バランサのエンドポイントの更新に失敗しました
NCP00080	ロード バランサ ルール プールの作成に失敗しました
NCP00081	ロード バランサ仮想サーバが見つかりません
NCP00082	IP セットの読み取りに失敗しました
NCP00083	SNAT プールの取得に失敗しました

エラー コード	説明
NCP00084	ロード バランサ サービスの作成に失敗しました
NCP00085	ロード バランサ サービスの更新に失敗しました
NCP00086	論理ルーター ポートの更新に失敗しました
NCP00087	ロード バランサの初期化に失敗しました
NCP00088	IP アドレス プールが一意ではありません
NCP00089	レイヤー 7 ロード バランサ キャッシュの同期エラー
NCP00090	ロード バランサ プールが配置されていません
NCP00091	ロード バランサ ルール キャッシュの初期化エラー
NCP00092	SNAT プロセスが失敗しました
NCP00093	ロード バランサのデフォルト証明書エラー
NCP00094	ロード バランサのエンドポイントの削除に失敗しました
NCP00095	プロジェクトが見つかりません
NCP00096	プールのアクセスが拒否されました
NCP00097	ロード バランサ サービスの取得に失敗しました
NCP00098	ロード バランサ サービスの作成に失敗しました
NCP00099	ロード バランサのプール キャッシュ同期エラー

NSX ノード エージェントのエラー コード

エラー コード	説明
NCP01001	OVS アップリンクが見つかりません
NCP01002	ホストの MAC アドレスが見つかりません
NCP01003	OVS ポートの作成に失敗しました
NCP01004	ポッドが設定されていません
NCP01005	ポッドの設定に失敗しました
NCP01006	ポッドの設定解除に失敗しました
NCP01007	CNI ソケットが見つかりません
NCP01008	CNI の接続に失敗しました
NCP01009	CNI バージョンが一致しません
NCP01010	CNI メッセージの受信に失敗しました
NCP01011	CNI メッセージの転送に失敗しました
NCP01012	Hyperbus の接続に失敗しました
NCP01013	Hyperbus バージョンが一致しません
NCP01014	Hyperbus メッセージの受信に失敗しました
NCP01015	Hyperbus メッセージの転送に失敗しました

エラー コード	説明
NCP01016	GARP の送信に失敗しました
NCP01017	インターフェイスの設定に失敗しました

nsx-kube-proxy のエラー コード

エラー コード	説明
NCP02001	無効なプロキシのゲートウェイ ポートです
NCP02002	プロキシ コマンドの実行に失敗しました
NCP02003	プロキシの確認に失敗しました

CLI のエラー コード

エラー コード	説明
NCP03001	CLI の起動に失敗しました
NCP03002	CLI ソケットの作成に失敗しました
NCP03003	CLI ソケットの例外です
NCP03004	CLI クライアントの要求が無効です
NCP03005	CLI サーバの転送に失敗しました
NCP03006	CLI サーバの受信に失敗しました
NCP03007	CLI コマンドの実行に失敗しました

Kubernetes のエラー コード

エラー コード	説明
NCP05001	Kubernetes の接続に失敗しました
NCP05002	Kubernetes の設定が無効です
NCP05003	Kubernetes の要求に失敗しました
NCP05004	Kubernetes のキーが見つかりません
NCP05005	Kubernetes のタイプが見つかりません
NCP05006	Kubernetes ウォッチャの例外です
NCP05007	Kubernetes リソースの長さが無効です
NCP05008	Kubernetes リソースのタイプが無効です
NCP05009	Kubernetes リソースの処理に失敗しました
NCP05010	Kubernetes サービスの処理に失敗しました
NCP05011	Kubernetes エンドポイントの処理に失敗しました
NCP05012	Kubernetes Ingress ハンドルの処理に失敗しました

エラー コード	説明
NCP05013	Kubernetes ネットワーク ポリシーの処理に失敗しました
NCP05014	Kubernetes ノードの処理に失敗しました
NCP05015	Kubernetes 名前空間の処理に失敗しました
NCP05016	Kubernetes ポッドの処理に失敗しました
NCP05017	Kubernetes Secret の処理に失敗しました
NCP05018	Kubernetes デフォルト バックエンドが失敗しました
NCP05019	一致式が Kubernetes でサポートされていません
NCP05020	Kubernetes の状態の更新に失敗しました
NCP05021	Kubernetes のアノテーションの更新に失敗しました
NCP05022	Kubernetes の名前空間のキャッシュが見つかりません
NCP05023	Kubernetes Secret が見つかりません
NCP05024	Kubernetes デフォルト バックエンドが使用中です
NCP05025	Kubernetes LoadBalancer サービスの処理に失敗しました

OpenShift のエラー コード

エラー コード	説明
NCP07001	OC ルートの処理に失敗しました
NCP07002	OC ルートの状態の更新に失敗しました