

Kubernetes 및 Cloud Foundry용 NSX Container Plug-in - 설치 및 관리 가이드

VMware NSX Container Plug-in 2.4, 2.4.1

VMware NSX-T Data Center 2.4

다음 VMware 웹 사이트에서 최신 기술 문서를 확인할 수 있습니다.

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

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

VMware 코리아
서울시 강남구
영동대로 517
아셈타워 13층
(우) 06164
전화: +82 2 3016 6500
팩스: +82 2 3016 6501
www.vmware.com/kr

목차

Kubernetes 및 Cloud Foundry용 NSX Container Plug-in - 설치 및 관리 가이드 5

1 NSX Container Plug-in 개요 6

호환성 요구 사항 7

설치 개요 8

Kubernetes 환경에서 NCP 업그레이드 8

Pivotal Cloud Foundry 환경에서 NCP 업그레이드 9

2 NSX-T 리소스 설정 10

NSX-T 리소스 구성 10

3 Kubernetes 환경에서 NCP 설치 16

NSX-T Data Center CNI 플러그인 설치 16

OVS 설치 및 구성 17

Kubernetes 노드에 대한 NSX-T Data Center 네트워킹 구성 18

NSX 노드 에이전트 설치 19

nsx-node-agent-ds.yml의 ncp.ini에 대한 Configmap 20

NSX Container Plug-in 설치 23

ncp-rc.yml의 ncp.ini에 대한 Configmap 26

NCP 포트에 PEM으로 인코딩된 인증서 및 개인 키 마운트 32

NCP 포트에 인증서 파일 마운트 32

Syslog 구성 33

Syslog용 사이드카 컨테이너 생성 33

Syslog용 DaemonSet 복제 생성 35

예: 사이드카 컨테이너에서 실행되는 로그 순환 및 Syslog 구성 37

보안 고려 사항 43

네트워크 리소스 구성 팁 47

4 Pivotal Cloud Foundry 환경에서 NCP 설치 49

Pivotal Cloud Foundry 환경에서 NCP 설치 49

5 로드 밸런싱 52

로드 밸런싱 구성 52

타사 수신 컨트롤러 59

6 NSX Container Plug-in 관리 63

Kubernetes 리소스 NSXError에 저장된 오류 정보 표시 63

CIF 연결 논리적 포트 64

CLI 명령 65

오류 코드 83

Kubernetes 및 Cloud Foundry용 NSX Container Plug-in - 설치 및 관리 가이드

이 가이드에서는 NCP(NSX Container Plug-in)를 설치 및 관리하여 NSX-T Data Center와 Kubernetes 간의 통합은 물론 NSX-T Data Center와 PCF(Pivotal Cloud Foundry) 간의 통합을 제공하는 방법에 대해 설명합니다.

대상 사용자

이 가이드는 시스템 및 네트워크 관리자를 대상으로 작성되었습니다. 이 가이드에서는 사용자가 NSX-T Data Center, Kubernetes 및 Pivotal Cloud Foundry의 설치 및 관리에 익숙하다고 가정합니다.

VMware 기술 자료 용어집

VMware 기술 자료 사이트에서는 새로운 용어를 정리한 용어집을 제공하고 있습니다. VMware 기술 설명서에 사용된 용어에 대한 정의를 보려면 <http://www.vmware.com/support/pubs>를 참조하십시오.

NSX Container Plug-in 개요

1

NCP(NSX Container Plug-in)는 NSX-T Data Center와 컨테이너 조정기(예: Kubernetes) 사이의 통합을 제공하고, NSX-T Data Center와 컨테이너 기반 PaaS(Platform As a Service) 제품(예: OpenShift 및 Pivotal Cloud Foundry) 사이의 통합을 제공합니다. 이 가이드는 Kubernetes 및 Pivotal Cloud Foundry를 사용한 NCP 설정에 대해 설명합니다.

NCP의 주요 구성 요소는 컨테이너에서 실행되며 NSX Manager 및 Kubernetes 제어부와 통신합니다. NCP는 컨테이너 및 기타 리소스에 대한 변경 사항을 모니터링하고, NSX API를 호출하여 컨테이너의 논리적 포트, 스위치, 라우터 및 보안 그룹과 같은 네트워킹 리소스를 관리합니다.

NSX CNI 플러그인은 각 Kubernetes 노드에서 실행됩니다. 컨테이너 수명 주기 이벤트를 모니터링하고, 컨테이너 인터페이스를 게스트 vSwitch에 연결하고, 컨테이너 인터페이스와 VNIC 간에 태그를 지정하고 컨테이너 트래픽을 전달하도록 게스트 vSwitch를 프로그래밍합니다.

NCP는 다음과 같은 기능을 제공합니다.

- Kubernetes 클러스터에 대한 NSX-T Data Center 논리적 토폴로지를 자동으로 생성하고 각 Kubernetes 네임스페이스에 대해 별도의 논리적 네트워크를 생성합니다.
- Kubernetes 포트를 논리적 네트워크에 연결하고 IP 및 MAC 주소를 할당합니다.
- NAT(네트워크 주소 변환)를 지원하고 각 Kubernetes 네임스페이스에 대해 별도의 SNAT IP를 할당합니다.

참고 NAT를 구성할 때 변환된 IP의 총 수는 1000개를 초과할 수 없습니다.

- NSX-T Data Center 분산 방화벽으로 Kubernetes 네트워크 정책을 구현합니다.
 - 수신 및 송신 네트워크 정책에 대한 지원.
 - 네트워크 정책에서 IPBlock 선택기 지원.
 - 네트워크 정책에 레이블 선택기를 지정할 때 matchLabels 및 matchExpression 지원.
 - 다른 네임스페이스에 있는 포트의 선택 지원.
- ClusterIP 유형의 Kubernetes 서비스와 LoadBalancer 유형의 서비스를 구현합니다.
- NSX-T 계층 7 로드 밸런서를 사용하여 Kubernetes 수신을 구현합니다.
 - TLS Edge 종료로 HTTP 수신 및 HTTPS 수신 지원.

- 수신 기본 백엔드 구성 지원.
- 수신 URI 재작성 지원.
- 네임스페이스, 포트 이름 및 포트의 레이블에 대한 NSX-T Data Center 논리적 스위치 포트에 태그를 생성하고 관리자가 태그를 기반으로 NSX-T 보안 그룹 및 정책을 정의하도록 허용합니다.

이 릴리스에서 NCP는 단일 Kubernetes 클러스터를 지원합니다. 각각 고유의 NCP 인스턴스가 있는 여러 개의 Kubernetes 클러스터를 동일한 NSX-T Data Center 배포를 통해 사용할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- [호환성 요구 사항](#)
- [설치 개요](#)
- [Kubernetes 환경에서 NCP 업그레이드](#)
- [Pivotal Cloud Foundry 환경에서 NCP 업그레이드](#)

호환성 요구 사항

Kubernetes 환경 및 PCF(Pivotal Cloud Foundry) 환경에 대한 NSX Container Plug-in(NCP)의 호환성 요구 사항은 다음과 같습니다.

표 1-1. Kubernetes 환경의 호환성 요구 사항

소프트웨어 제품	버전
NSX-T Data Center	2.3, 2.4
컨테이너 호스트 VM용 하이퍼바이저	<ul style="list-style-type: none"> ■ 지원되는 vSphere 버전 ■ RHEL KVM 7.4, 7.5, 7.6 ■ Ubuntu KVM 16.04
컨테이너 호스트 운영 체제	<ul style="list-style-type: none"> ■ RHEL 7.5, 7.6 ■ Ubuntu 16.04 ■ CentOS 7.4, 7.5
컨테이너 조정기	Kubernetes 1.12, 1.13
컨테이너 호스트 Open vSwitch	2.9.1(NSX-T Data Center 2.3.x와 패키지로 제공), 2.10.2(NSX-T Data Center 2.4.0과 패키지로 제공)

표 1-2. Cloud Foundry 환경의 호환성 요구 사항

소프트웨어 제품	버전
컨테이너 호스트 VM용 하이퍼바이저	■ 지원되는 vSphere 버전
컨테이너 조정기	<ul style="list-style-type: none"> ■ Pivotal Application Service 2.3.x 및 Pivotal Operations Manager 2.3.x ■ Pivotal Application Service 2.4.x(2.4.0 제외) 및 Pivotal Operations Manager 2.4.x(2.4.0 제외)

설치 개요

Kubernetes가 이미 설치되어 있는 환경에서 NCP를 설치 및 구성하는 작업에는 일반적으로 다음 단계가 포함됩니다. 단계를 성공적으로 수행하려면 NSX-T Data Center 및 Kubernetes 설치 및 관리에 익숙해야 합니다.

- 1 NSX-T Data Center를 설치합니다.
- 2 오버레이 전송 영역을 생성합니다.
- 3 오버레이 논리적 스위치를 생성하고 Kubernetes 노드를 스위치에 연결합니다.
- 4 Tier-0 논리적 라우터를 생성합니다.
- 5 Kubernetes 포드에 대한 IP 블록을 생성합니다.
- 6 SNAT(소스 네트워크 주소 변환)를 위한 IP 풀을 생성합니다.
- 7 각 노드에 NSX CNI(컨테이너 네트워크 인터페이스) 플러그인을 설치합니다.
- 8 각 노드에 OVS(Open vSwitch)를 설치합니다.
- 9 Kubernetes 노드에 대해 NSX-T 네트워킹을 구성합니다.
- 10 NSX 노드 에이전트를 DaemonSet으로 설치합니다.
- 11 NCP를 ReplicationController로 설치합니다.
- 12 NCP 포드에 보안 인증서를 마운트합니다.

Kubernetes 환경에서 NCP 업그레이드

이 섹션에서는 Kubernetes 환경에서 NCP를 2.4.0으로 업그레이드하는 방법에 대해 설명합니다.

절차

- 1 다음 명령으로 NCP ReplicationController를 업그레이드합니다(<image>를 이미지의 실제 이름으로 대체).

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

- 2 다음 명령으로 NSX 노드 에이전트 daemonSet를 업그레이드합니다(<image>를 이미지의 실제 이름으로 대체).

```
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 -n nsx-system
```

- 3 다음 명령으로 CNI DEB/RPM 패키지를 2.4.0으로 업그레이드합니다(<cni deb> 및 <cni rpm>을 패키지의 실제 이름으로 대체).

Ubuntu에서:

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


RHEL 또는 CentOS에서:

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

- 4 (선택 사항) NSX-T Data Center를 2.4로 업그레이드합니다.

하이퍼바이저가 ESXi인 경우 NSX-T Data Center를 업그레이드하기 전에 6.5에서 6.5p03 이상으로 또는 6.7에서 6.7ep6 이상으로 업그레이드합니다.

- 5 (선택 사항) 하이퍼바이저(KVM 또는 베어메탈 컨테이너)를 업그레이드합니다
- 6 (선택 사항) 컨테이너 호스트(RHEL, Ubuntu 또는 CentOS)를 업그레이드합니다
- 7 (선택 사항) Kubernetes를 업그레이드합니다.
- 8 (선택 사항) OVS를 업그레이드합니다.

베어메탈 컨테이너의 경우 NSX-T Data Center를 업그레이드하면 OVS도 업그레이드되기 때문에 이 단계가 필요하지 않습니다.

이 단계 중에 nsx-kube-proxy와 nsx-node-agent 간에 일시적인 통신 장애가 발생할 수 있습니다. 이는 예상되는 동작이며 문제를 나타내지 않습니다.

Pivotal Cloud Foundry 환경에서 NCP 업그레이드

이 섹션에서는 Pivotal Cloud Foundry 환경에서 NCP를 2.4.0으로 업그레이드하는 방법에 대해 설명합니다.

절차

- 1 NCP 또는 NSX-T 타일을 2.4.0으로 업그레이드합니다.
- 2 (선택 사항) Pivotal Operations Manager를 업그레이드한 다음 Pivotal Application Service를 업그레이드합니다.
- 3 (선택 사항) NSX-T Data Center를 2.4로 업그레이드합니다.
하이퍼바이저가 ESXi인 경우 NSX-T Data Center를 업그레이드하기 전에 6.5에서 6.5p03 이상으로 또는 6.7에서 6.7ep6 이상으로 업그레이드합니다.
- 4 (선택 사항) 하이퍼바이저(KVM 또는 베어메탈 컨테이너)를 업그레이드합니다

NSX-T 리소스 설정

2

NSX Container Plug-in을 설치하기 전에 특정 NSX-T Data Center 리소스를 설정해야 합니다.

본 장은 다음 항목을 포함합니다.

- NSX-T 리소스 구성

NSX-T 리소스 구성

구성해야 하는 NSX-T Data Center 리소스로는 오버레이 전송 영역, Tier-0 논리적 라우터, 노드 VM 연결을 위한 논리적 스위치, Kubernetes 노드용 IP 블록 및 SNAT용 IP 풀 등이 있습니다.

중요 NSX-T Data Center 2.4 이상에서 실행 중인 경우 **고급 네트워킹 및 보안** 탭을 사용하여 NSX-T 리소스를 구성해야 합니다.

NCP 구성 파일 `ncp.ini`에서 NSX-T Data Center 리소스는 UUID 또는 이름을 사용하여 지정됩니다.

오버레이 전송 영역

NSX Manager에 로그인하고 **시스템 > 패브릭 > 전송 영역**으로 이동합니다. 컨테이너 네트워킹에 사용되는 오버레이 전송 영역을 찾거나 새 영역을 생성합니다.

`ncp.ini`의 `[nsx_v3]` 섹션에 `overlay_tz` 옵션을 설정하여 클러스터에 대한 오버레이 전송 영역을 지정합니다. 이 단계는 선택 사항입니다. `overlay_tz`를 설정하지 않으면 NCP가 Tier-0 라우터에서 오버레이 전송 영역 ID를 자동으로 검색합니다.

Tier-0 논리적 라우팅

NSX Manager에 로그인하고 **고급 네트워킹 및 보안 > 네트워킹 > 라우터**로 이동합니다. 컨테이너 네트워킹에 사용되는 라우터를 찾거나 새 라우터를 생성합니다.

`ncp.ini`의 `[nsx_v3]` 섹션에 `tier0_router` 옵션을 설정하여 클러스터에 대한 Tier-0 논리적 라우터를 지정합니다.

참고 라우터는 활성-대기 모드로 생성해야 합니다.

논리적 스위치

노드가 데이터 트래픽에 사용하는 vNIC는 오버레이 논리적 스위치에 연결해야 합니다. 노드의 관리 인터페이스를 NSX-T Data Center에 연결하면 설정이 더 쉬워지지만 반드시 그렇게 할 필요는 없습니다. NSX Manager에 로그인하고 **고급 네트워킹 및 보안 > 네트워킹 > 스위칭 > 스위치**로 이동하여 논리적 스위치를 생성할 수 있습니다. 스위치에서 논리적 포트를 생성하고 노드 vNIC를 논리적 스위치에 연결합니다. 논리적 포트에는 다음과 같은 태그가 있어야 합니다.

- 태그: <cluster_name>, 범위: ncp/cluster
- 태그: <node_name>, 범위: ncp/node_name

<cluster_name> 값은 ncp.ini의 [coe] 섹션에 있는 cluster 옵션 값과 일치해야 합니다.

Kubernetes 포드용 IP 블록

NSX Manager에 로그인하고 **고급 네트워킹 및 보안 > 네트워킹 > IPAM**으로 이동하여 IP 블록을 하나 이상 생성합니다. CIDR 형식으로 IP 블록을 지정합니다.

ncp.ini의 [nsx_v3] 섹션에 container_ip_blocks 옵션을 설정하여 Kubernetes 포드에 대한 IP 블록을 지정합니다.

비 SNAT 네임스페이스(Kubernetes의 경우) 또는 클러스터(PCF의 경우)에 대한 IP 블록을 생성할 수도 있습니다.

ncp.ini의 [nsx_v3] 섹션에 no_snat_ip_blocks 옵션을 설정하여 비 SNAT IP 블록을 지정합니다.

NCP가 실행되는 동안 비 SNAT IP 블록을 생성하면 NCP를 다시 시작해야 합니다. 그렇지 않으면, NCP는 고갈될 때까지 공유 IP 블록을 계속 사용합니다.

참고 IP 블록을 생성할 때 접두사는 NCP의 구성 파일 ncp.ini에 있는 subnet_prefix 매개 변수의 값보다 크지 않아야 합니다. 자세한 내용은 [ncp-rc.yml의 ncp.ini에 대한 Configmap](#) 항목을 참조하십시오.

SNAT용 IP 풀

NSX Manager의 IP 풀은 SNAT 규칙을 사용하여 포트 IP를 변환하고 SNAT/DNAT 규칙을 사용하여 수신 컨트롤러를 노출하는 데 사용되는 IP 주소(Openstack 유동 IP와 동일) 할당에 사용됩니다. 이러한 IP 주소를 외부 IP라고도 합니다.

여러 Kubernetes 클러스터가 동일한 외부 IP 풀을 사용합니다. 각 NCP 인스턴스는 관리하는 Kubernetes 클러스터에 대해 이 풀의 하위 집합을 사용합니다. 기본적으로 포트 서브넷에 대해 동일한 서브넷 접두사가 사용됩니다. 다른 서브넷 크기를 사용하려면 ncp.ini의 [nsx_v3] 섹션에서 external_subnet_prefix 옵션을 업데이트합니다.

ncp.ini의 [nsx_v3] 섹션에 external_ip_pools 옵션을 설정하여 SNAT에 대한 IP 풀을 지정할 수 있습니다.

구성 파일을 변경하고 NCP를 다시 시작하여 다른 IP 풀로 변경할 수 있습니다.

SNAT IP 풀을 특정 Kubernetes 네임스페이스 또는 PCF 조직으로 제한

다음 태그를 IP 풀에 추가하여 SNAT IP 풀의 IP를 할당할 수 있는 Kubernetes 네임스페이스 또는 PCF 조직을 지정할 수 있습니다.

- Kubernetes 네임스페이스의 경우: `scope: ncp/owner, tag: ns:<namespace_UUID>`
- PCF 조직의 경우: `scope: ncp/owner, tag: org:<org_UUID>`

다음 명령 중 하나를 사용하여 네임스페이스 또는 조직 UUID를 가져올 수 있습니다.

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

다음에 유의하십시오.

- 각 태그는 하나의 UUID를 지정해야 합니다. 동일한 풀에 대해 여러 개의 태그를 생성할 수 있습니다.
- 이전 태그를 기반으로 일부 네임스페이스 또는 조직에 IP를 할당한 후 태그를 변경했다면 이러한 IP는 Kubernetes 서비스 또는 PCF 애플리케이션의 SNAT 구성이 변경되거나 NCP가 다시 시작될 때까지 회수되지 않습니다.
- 네임스페이스 및 PCF 조직 소유자 태그는 선택 사항입니다. 이러한 태그가 없으면 모든 네임스페이스 또는 PCF 조직이 SNAT IP 풀에서 할당된 IP를 가질 수 있습니다.

서비스에 대한 SNAT IP 풀 구성

서비스에 주석을 추가하여 서비스에 대한 SNAT IP 풀을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

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

`ncp/snat_pool`에서 지정한 IP 풀에는 태그 `scope: ncp/owner, tag: cluster:<cluster_name>`이 있어야 합니다.

NCP는 이 서비스에 대한 SNAT 규칙을 구성합니다. 규칙의 소스 IP는 백엔드 포트 집합입니다. 대상 IP는 지정된 외부 IP 풀에서 할당된 SNAT IP입니다. NCP가 SNAT 규칙을 구성할 때 오류가 발생하면 서비스에 `ncp/error.snat:<error>` 주석이 추가됩니다. 다음과 같은 오류가 발생할 수 있습니다.

- `IP_POOL_NOT_FOUND` - NSX Manager에서 SNAT IP 풀을 찾을 수 없습니다.
- `IP_POOL_EXHAUSTED` - SNAT IP 풀이 모두 사용되었습니다.
- `IP_POOL_NOT_UNIQUE` - `ncp/snat_pool`에서 지정한 풀이 NSX Manager에서 여러 풀을 나타냅니다.

- **SNAT_POOL_ACCESS_DENY** - 풀의 소유자 태그와 할당 요청을 보내는 서비스의 네임스페이스가 일치하지 않습니다.
- **SNAT_RULE_OVERLAPPED** - 새 SNAT 규칙을 생성했지만 SNAT 서비스의 포트가 다른 SNAT 서비스에 속해 있습니다. 즉, 동일한 포트에 대해 SNAT 규칙이 여러 개 있습니다.
- **POOL_ACCESS_DENIED** - `ncp/snat_pool`에서 지정한 IP 풀에 태그 `scope: ncp/owner`, `tag: cluster:<cluster_name>`가 없거나 풀의 소유자 태그가 할당 요청을 전송 중인 서비스의 네임스페이스와 일치하지 않습니다.

다음에 유의하십시오.

- 서비스를 구성하기 전에 `ncp/snat_pool`에 지정된 풀이 NSX-T Data Center에 이미 있어야 합니다.
- NSX-T Data Center에서 서비스에 대한 SNAT 규칙의 우선 순위는 프로젝트의 우선 순위보다 높습니다.
- 포트가 여러 SNAT 규칙으로 구성된 경우 규칙 하나만 작동합니다.
- 주석을 변경하고 NCP를 다시 시작하여 다른 IP 풀로 변경할 수 있습니다.

네임스페이스에 대한 SNAT IP 풀 구성

네임스페이스에 주석을 추가하여 네임스페이스에 대한 SNAT IP 풀을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

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

NCP는 이 네임스페이스에 대한 SNAT 규칙을 구성합니다. 규칙의 소스 IP는 백엔드 포트 집합입니다. 대상 IP는 지정된 외부 IP 풀에서 할당된 SNAT IP입니다. NCP가 SNAT 규칙을 구성할 때 오류가 발생하면 네임스페이스에 `ncp/error.snat:<error>` 주석이 추가됩니다. 다음과 같은 오류가 발생할 수 있습니다.

- **IP_POOL_NOT_FOUND** - NSX Manager에서 SNAT IP 풀을 찾을 수 없습니다.
- **IP_POOL_EXHAUSTED** - SNAT IP 풀이 모두 사용되었습니다.
- **IP_POOL_NOT_UNIQUE** - `ncp/snat_pool`에서 지정한 풀이 NSX Manager에서 여러 풀을 나타냅니다.
- **POOL_ACCESS_DENIED** - `ncp/snat_pool`에서 지정한 IP 풀에 태그 `scope: ncp/owner`, `tag: cluster:<cluster_name>`가 없거나 풀의 소유자 태그가 할당 요청을 전송 중인 네임스페이스와 일치하지 않습니다.

다음에 유의하십시오.

- 주석에 하나의 SNAT IP 풀만 지정할 수 있습니다.

- `ncp.ini`에는 SNAT IP 풀을 지정할 필요가 없습니다.
- `ncp/snat_pool`에서 지정한 IP 풀에는 태그 `scope: ncp/owner`, `tag: cluster:<cluster_name>`이 있어야 합니다.
- `ncp/snat_pool`에서 지정한 IP 풀에 네임스페이스 태그 `scope: ncp/owner`, `tag: ns:<namespace_UUID>`도 있을 수 있습니다.
- `ncp/snat_pool` 주석이 누락된 경우 네임스페이스는 클러스터에 대한 SNAT IP 풀을 사용합니다.
- 주석을 변경하고 NCP를 다시 시작하여 다른 IP 풀로 변경할 수 있습니다.

PAS App에 대한 SNAT 풀 구성

기본적으로 NCP는 PAS(Pivotal Application Service) 조직을 위한 SNAT IP를 구성합니다. SNAT IP 풀 정보가 포함된 `manifest.xml`을 사용하여 애플리케이션을 생성함으로써 애플리케이션을 위한 SNAT IP를 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
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>
...
```

NCP는 이 애플리케이션에 대한 SNAT 규칙을 구성합니다. 규칙의 소스 IP는 인스턴스 IP의 집합이고 대상 IP는 외부 IP 풀에서 할당된 SNAT IP입니다. 다음에 유의하십시오.

- 애플리케이션이 푸시되기 전에 `NCP_SNAT_POOL`에 지정된 풀이 NSX-T Data Center에 이미 있어야 합니다.
- 애플리케이션에 대한 SNAT 규칙의 우선 순위는 조직에 대한 SNAT 규칙의 우선 순위보다 높습니다.
- 애플리케이션은 하나의 SNAT IP만 사용하여 구성할 수 있습니다.
- 구성을 변경하고 NCP를 다시 시작하여 다른 IP 풀로 변경할 수 있습니다.

PCF 버전 3에 대한 SNAT 구성

PCF 버전 3에서는 다음 두 가지 방법 중 하나로 SNAT를 구성할 수 있습니다.

- 애플리케이션을 생성할 때 `manifest.yml`에서 `NCP_SNAT_POOL`을 구성합니다.

예를 들어 애플리케이션의 이름이 `bread`이고 `manifest.yml`에 다음과 같은 정보가 포함되어 있다고 가정합니다.

```
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
```

다음 명령을 실행합니다.

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

- `cf v3-set-env` 명령을 사용하여 `NCP_SNAT_POOL`을 구성합니다.

다음 명령을 실행합니다(애플리케이션의 이름이 `app3`이라고 가정함).

```
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
```

(선택 사항)(Kubernetes만 해당) 방화벽 마커 섹션

관리자가 방화벽 규칙을 생성하고 해당 규칙이 NCP가 생성한 네트워크 정책 기반의 방화벽을 방해하지 않게 하려면 NSX Manager에 로그인하고 **보안 > 분산 방화벽 > 일반**으로 이동한 후 방화벽 섹션을 두 개 생성합니다.

`ncp.ini`의 `[nsx_v3]` 섹션에 `bottom_firewall_section_marker` 옵션과 `top_firewall_section_marker` 옵션을 설정하여 마커 방화벽 섹션을 지정합니다.

하단 방화벽 섹션은 상단 방화벽 섹션보다 아래에 있어야 합니다. 이러한 방화벽 섹션을 생성하면 NCP가 분리를 위해 생성하는 모든 방화벽 섹션이 하단 방화벽 섹션의 위에 생성되고 NCP가 정책을 위해 생성하는 모든 방화벽 섹션이 상단 방화벽 섹션의 아래에 생성됩니다. 이러한 마커 섹션을 생성하지 않으면 모든 분리 규칙이 하단에 생성되고 모든 정책 섹션이 상단에 생성됩니다. 클러스터 하나에 값이 동일한 마커 방화벽 섹션을 여러 개 사용할 수 없으며, 이렇게 할 경우 오류가 발생합니다.

Kubernetes 환경에서 NCP 설치

3

NCP(NSX Container Plug-in)를 설치하려면 마스터 및 Kubernetes 노드에 구성 요소를 설치해야 합니다.

본 장은 다음 항목을 포함합니다.

- NSX-T Data Center CNI 플러그인 설치
- OVS 설치 및 구성
- Kubernetes 노드에 대한 NSX-T Data Center 네트워킹 구성
- NSX 노드 에이전트 설치
- nsx-node-agent-ds.yml의 ncp.ini에 대한 Configmap
- NSX Container Plug-in 설치
- ncp-rc.yml의 ncp.ini에 대한 Configmap
- NCP 포트에 PEM으로 인코딩된 인증서 및 개인 키 마운트
- NCP 포트에 인증서 파일 마운트
- Syslog 구성
- 보안 고려 사항
- 네트워크 리소스 구성 팁

NSX-T Data Center CNI 플러그인 설치

NSX-T Data Center CNI 플러그인은 Kubernetes 노드에 설치해야 합니다.

Ubuntu의 경우 NSX-T CNI 플러그인을 설치하면 AppArmor 프로파일 파일인 `ncp-apparmor`가 `/etc/apparmor.d`에 복사되고 로드됩니다. 설치 전에 AppArmor 서비스를 실행해야 하며 `/etc/apparmor.d` 디렉토리가 있어야 합니다. 없으면 설치가 실패합니다. 다음 명령을 실행하여 AppArmor 모듈이 사용되는지 여부를 확인할 수 있습니다.

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


다음 명령을 실행하여 AppArmor 서비스가 시작되었는지 여부를 확인할 수 있습니다.

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

AppArmor 서비스가 실행되지 않는 상태에서 NSX-T CNI 플러그인을 설치하면 설치가 완료될 때 다음 메시지가 표시됩니다.

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

이 메시지는 AppArmor 프로파일을 로드하는 단계를 제외하고 모든 설치 단계가 완료되었음을 나타냅니다.

ncp-apparmor 프로파일 파일에 포함된 NSX 노드 에이전트용 AppArmor 프로파일인 node-agent-apparmor는 다음과 같은 점에서 docker-default 프로파일과 다릅니다.

- deny mount 규칙이 제거되었습니다.
- mount 규칙이 추가되었습니다.
- 일부 network, capability, file 및 umount 옵션이 추가되었습니다.

node-agent-apparmor 프로파일을 다른 프로파일로 바꿀 수 있습니다. 그러나 프로파일 이름 node-agent-apparmor는 NSX 노드 에이전트의 설치에 사용되는 nsx-node-agent-ds.yml 파일에서 참조됩니다. 다른 프로파일을 사용하는 경우 nsx-node-agent-ds.yml 파일의 spec:template:metadata:annotations 섹션에 있는 다음 항목에 프로파일 이름을 지정해야 합니다.

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

절차

- 1 Linux 배포에 적합한 설치 파일을 다운로드합니다.

파일 이름은 nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm 또는 nsx-cni-1.0.0.0.0.xxxxxxx.deb입니다. 여기서 xxxxxxxx는 빌드 번호입니다.

- 2 1단계에서 다운로드한 rpm 또는 deb 파일을 설치합니다.

플러그인이 /opt/cni/bin에 설치됩니다. CNI 구성 파일 10-nsx.conf가 /etc/cni/net.d에 복사됩니다. rpm은 루프백 플러그인에 대한 구성 파일 /etc/cni/net.d/99-loopback.conf도 설치합니다.

OVS 설치 및 구성

미니언 노드에 OVS(Open vSwitch)를 설치하고 구성합니다.

절차

- 1 Linux 배포용 설치 파일을 다운로드합니다.

파일 이름은 openvswitch-common_2.10.x.xxxxxxx-1_amd64.deb, openvswitch-datapath-dkms_2.10.x.xxxxxxx-1_all.deb 및 openvswitch-switch_2.10.x.xxxxxxx-1_amd64.deb입니다. 여기서 xxxxxxxx는 빌드 번호입니다.

- 2 단계에서 다운로드한 deb 파일을 설치합니다.
- 3 Ubuntu의 경우 다음 명령을 실행하여 OVS 커널 모듈을 다시 로드합니다.

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

- 4 OVS가 실행 중인지 확인합니다.

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

- 5 *br-int* 인스턴스가 아직 생성되지 않은 경우 생성합니다.

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

- 6 노드 논리적 스위치에 연결된 네트워크 인터페이스(*node-if*)를 *br-int*에 추가합니다.

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

다음 명령을 실행하여 **ofport**를 확인합니다. **ofport** 1을 사용할 수 없으면 OVS가 사용 가능한 포트를 할당하기 때문입니다.

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

ofport가 1이 아니면 그에 따라 NSX 노드 에이전트 DaemonSet yaml 파일의 **nsx_kube_proxy** 섹션에서 **ovs_uplink_port** 옵션을 설정합니다.

- 7 *br-int* 및 *node-if link* 상태가 [실행]인지 확인합니다.

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

- 8 네트워크 구성 파일을 업데이트하여 재부팅 후 네트워크 인터페이스가 작동되도록 합니다.

Ubuntu의 경우 **/etc/network/interfaces**를 업데이트하고 다음 줄을 추가합니다.

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

RHEL의 경우 **/etc/sysconfig/network-scripts/ifcfg-<node-if>**를 업데이트하고 다음 줄을 추가합니다.

```
ONBOOT=yes
```

Kubernetes 노드에 대한 NSX-T Data Center 네트워킹 구성

이 섹션에서는 Kubernetes 마스터 및 작업자 노드에 대한 NSX-T Data Center 네트워킹을 구성하는 방법을 설명합니다.

각 노드에는 적어도 두 개의 네트워크 인터페이스가 있어야 합니다. 첫 번째는 관리 인터페이스로 **NSX-T Data Center** 패브릭에 있을 수도 있고 없을 수도 있습니다. 다른 인터페이스는 포트에 대한 네트워킹을 제공하고, **NSX-T Data Center** 패브릭에 있으며, 노드 논리적 스위치라고도 하는 논리적 스위치에 연결됩니다. **Kubernetes** 상태 점검이 작동하려면 관리 및 포트 IP 주소가 라우팅 가능해야 합니다. 관리 인터페이스와 포트 사이의 통신을 위해 **NCP**는 상태 점검 및 기타 관리 트래픽을 허용하는 **DFW** 규칙을 자동으로 생성합니다. **NSX Manager GUI**에서 이 규칙의 세부 정보를 확인할 수 있습니다. 이 규칙은 변경하거나 삭제해서는 안 됩니다.

각 노드 **VM**에 대해 컨테이너 네트워킹용으로 지정된 **vNIC**가 노드 논리적 스위치에 연결되어 있는지 확인하십시오.

각 노드에서 컨테이너 트래픽에 사용되는 **vNIC**의 **VIF ID**를 **NCP(NSX Container Plug-in)**에 알려야 합니다. 해당 논리적 스위치 포트에는 다음과 같은 2개의 태그가 있어야 합니다. 하나의 태그에는 노드의 이름을 지정합니다. 다른 태그에는 클러스터의 이름을 지정합니다. 범위에는 아래에 표시된 대로 적절한 값을 지정합니다.

태그	범위
노드 이름	ncp/node_name
클러스터 이름	ncp/cluster

NSX Manager GUI에서 **인벤토리 > 가상 시스템**으로 이동하여 노드 **VM**의 논리적 스위치 포트를 식별할 수 있습니다.

Kubernetes 노드 이름이 변경되면 **ncp/node_name** 태그를 업데이트하고 **NCP**를 다시 시작해야 합니다. 다음 명령을 사용하여 노드 이름을 가져올 수 있습니다.

```
kubectl get nodes
```

NCP가 실행되는 동안 클러스터에 노드를 추가하는 경우 **kubeadm join** 명령을 실행하기 전에 논리적 스위치 포트에 태그를 추가해야 합니다. 그렇지 않으면 새 노드에 네트워크 연결이 설정되지 않습니다. 태그가 잘못되었거나 누락된 경우 다음 단계에 따라 문제를 해결할 수 있습니다.

- 논리적 스위치 포트에 올바른 태그를 적용합니다.
- **NCP**를 다시 시작합니다.

NSX 노드 에이전트 설치

NSX 노드 에이전트는 각 포트가 두 개의 컨테이너를 실행하는 **DaemonSet**입니다. 하나의 컨테이너는 주로 컨테이너 네트워크 인터페이스를 관리하는 **NSX** 노드 에이전트를 실행합니다. 이 에이전트는 **CNI** 플러그인 및 **Kubernetes API** 서버와 상호 작용합니다. 다른 컨테이너는 클러스터 IP를 포트 IP로 변환하여 **Kubernetes** 서비스 추상화를 구현하는 작업만 담당하는 **NSX kube-proxy**를 실행합니다. 이는 업스트림 **kube-proxy**와 동일한 기능을 구현합니다.

절차

- 1 NCP Docker 이미지를 다운로드합니다.

파일 이름은 `nsx-ncp-xxxxxxx.tar`입니다. 여기서 `xxxxxxx`는 빌드 번호입니다.

- 2 NSX 노드 에이전트 DaemonSet yaml 템플릿을 다운로드합니다.

파일 이름은 `nsx-node-agent-ds.yaml`입니다. 이 파일을 편집하거나 템플릿 파일의 예제로 사용할 수 있습니다.

- 3 NCP Docker 이미지를 이미지 레지스트리에 로드합니다.

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

- 4 `nsx-node-agent-ds.yaml`을 편집합니다.

이미지 이름을 로드된 이미지로 변경합니다.

Ubuntu의 경우 yaml 파일은 AppArmor가 사용되도록 설정되었다고 가정합니다. AppArmor가 사용되도록 설정되어 있는지 여부를 확인하려면 파일 `/sys/module/apparmor/parameters/enabled`를 확인합니다. AppArmor가 사용되도록 설정되어 있지 않으면 다음과 같이 변경하십시오.

- 다음 줄을 삭제하거나 주석 처리합니다.

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

- `nsx-node-agent` 컨테이너 및 `nsx-kube-proxy` 컨테이너에 대해 `privileged:true` 아래에 `securityContext` 줄을 추가합니다. 예:

```
securityContext:
  privileged:true
```

참고 kubelet이 hyperkube 이미지를 사용하는 컨테이너 내에서 실행되는 경우, kubelet이 실제 상태에 관계없이 항상 AppArmor가 사용되지 않도록 설정된 것으로 보고하는 알려진 문제가 있습니다. yaml 파일을 위에서 언급한 것과 동일한 방식으로 변경해야 합니다.

참고 yaml 파일에서 `ncp.ini`에 대해 생성된 ConfigMap이 `ReadOnly` 볼륨으로 마운트되도록 지정해야 합니다. 다운로드한 yaml 파일은 이미 이 사양을 가지며 변경해서는 안 됩니다.

- 5 다음 명령을 사용하여 NSX 노드 에이전트 DaemonSet을 생성합니다.

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

nsx-node-agent-ds.yaml의 ncp.ini에 대한 Configmap

샘플 yaml 파일 `nsx-node-agent-ds.yaml`에는 NSX 노드 에이전트의 구성 파일 `ncp.ini`에 대한 ConfigMap이 포함되어 있습니다. 이 ConfigMap 섹션에는 노드 에이전트 설치의 사용자 지정을 위해 지정할 수 있는 매개 변수가 포함되어 있습니다.

다운로드하는 샘플 `nsx-node-agent-ds.yml`에는 다음 `ncp.ini` 정보가 포함되어 있습니다.

```
# 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>
```

NSX Container Plug-in 설치

NCP(NSX Container Plug-in)는 Docker 이미지로 제공됩니다. NCP는 인프라 서비스용 노드에서 실행되어야 합니다. 마스터 노드에서 NCP를 실행하는 것은 권장되지 않습니다.

절차

- 1 NCP Docker 이미지를 다운로드합니다.

파일 이름은 `nsx-ncp-xxxxxxx.tar`입니다. 여기서 `xxxxxxx`는 빌드 번호입니다.

2 NCP ReplicationController yaml 템플릿을 다운로드합니다.

파일 이름은 `ncp-rc.yaml`입니다. 이 파일을 편집하거나 템플릿 파일의 예제로 사용할 수 있습니다.

3 NCP Docker 이미지를 이미지 레지스트리에 로드합니다.

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

4 (선택 사항) NSXError 개체의 사용자 지정 리소스 정의에 대한 yaml 템플릿을 다운로드합니다.

파일 이름은 `nsx-error-crd.yaml`입니다.

5 (선택 사항) 사용자 지정 리소스를 생성합니다.

```
kubect1 create -f nsx-error-crd.yaml
```

6 `ncp-rc.yaml`을 편집합니다.

이미지 이름을 로드된 이미지로 변경합니다.

`nsx_api_managers` 매개 변수를 지정합니다. 단일 NSX Manager의 IP 주소 또는 NSX Manager 클러스터의 3개 NSX Manager의 IP 주소(쉼표로 구분됨) 또는 NSX Manager 클러스터의 가상 IP 주소를 지정할 수 있습니다. 예:

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

(선택 사항) `[nsx_v3]` 섹션에 매개 변수 `ca_file`을 지정합니다. 이 값은 NSX Manager 서버 인증서를 확인하는 데 사용할 CA 번들 파일이어야 합니다. 설정하지 않으면 시스템 루트 CA가 사용됩니다. `nsx_api_managers`에 대해 하나의 IP 주소를 지정하면 하나의 CA 파일을 지정합니다.

`nsx_api_managers`에 대해 3개의 IP 주소를 지정하면 하나 또는 3개의 CA 파일을 지정할 수 있습니다. 하나의 CA 파일을 지정하면 해당 파일이 모든 3개 관리자에 대해 사용됩니다. 3개의 CA 파일을 지정하면 각각 `nsx_api_managers`의 해당 IP 주소에 대해 사용됩니다. 예를 들면 다음과 같습니다.

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

NSX-T Data Center에 대한 인증을 위해 매개 변수 `nsx_api_cert_file` 및 `nsx_api_private_key_file`을 지정합니다.

`nsx_api_cert_file`은 PEM 형식의 클라이언트 인증서 파일에 대한 전체 경로입니다. 이 파일의 내용은 다음과 같아야 합니다.

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


`nsx_api_private_key_file`은 PEM 형식의 클라이언트 개인 키 파일에 대한 전체 경로입니다. 이 파일의 내용은 다음과 같아야 합니다.

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

수신 컨트롤러가 NAT 모드에서 실행되도록 구성되어 있는 경우 매개 변수 `ingress_mode = nat`를 지정합니다.

기본적으로 서브넷 접두사 **24**는 포트 논리적 스위치의 IP 블록에서 할당된 모든 서브넷에 사용됩니다. 다른 서브넷 크기를 사용하려면 `[nsx_v3]` 섹션에서 `subnet_prefix` 옵션을 업데이트합니다.

HA(고가용성)가 기본적으로 사용되도록 설정됩니다. 다음과 같은 규격으로 HA를 사용하지 않도록 설정할 수 있습니다.

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

(선택 사항) `nep.ini`에서 `NSXError`를 사용하여 오류 보고를 사용하도록 설정합니다. 이 설정은 기본적으로 사용되지 않도록 설정되어 있습니다.

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

참고 `yaml` 파일에서 `nep.ini`에 대해 생성된 `ConfigMap`이 `ReadOnly` 볼륨으로 마운트되도록 지정해야 합니다. 다운로드한 `yaml` 파일은 이미 이 사양을 가지며 변경해서는 안 됩니다.

7 NCP ReplicationController를 생성합니다.

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

결과

참고 NCP가 Kubernetes API 서버에 대한 영구 HTTP 연결을 열어 Kubernetes 리소스의 수명주기 이벤트를 모니터링합니다. API 서버 장애 또는 네트워크 장애로 인해 NCP의 TCP 연결 문제가 발생하는 경우 NCP를 다시 시작하여 API 서버에 대한 연결을 다시 설정해야 합니다. 그러지 않으면 NCP가 새 이벤트를 확인하지 못합니다.

NCP ReplicationController의 롤링 업데이트 중 다음과 같은 상황에서 롤링 업데이트를 수행했을 때 두 개의 NCP 포트가 실행 중일 수 있습니다.

- 롤링 업데이트 중에 컨테이너 호스트를 재부팅했습니다.
- 처음에는 Kubernetes 노드에 새 이미지가 없어서 롤링 업데이트가 실패했지만 이미지 다운로드 후 롤링 업데이트를 다시 실행하여 업데이트가 성공했습니다.

두 개의 NCP 포트가 실행 중이라면 다음을 수행합니다.

- NCP 포트 중 하나를 삭제합니다. 어느 것을 삭제하든 관계없습니다. 예를 들면 다음과 같습니다.

```
kubect1 delete pods <NCP pod name> -n nsx-system
```

- NCP ReplicationController를 삭제합니다. 예를 들면 다음과 같습니다.

```
kubect1 delete -f ncp-rc.yml -n nsx-system
```

ncp-rc.yml의 ncp.ini에 대한 Configmap

샘플 YAML 파일 `ncp-rc.yml`에는 구성 파일 `ncp.ini`에 대한 ConfigMap이 포함되어 있습니다. 이 ConfigMap 섹션에는 이전 섹션에 설명된 대로 NCP를 설치하기 전에 지정해야 하는 매개 변수가 포함되어 있습니다.

다운로드하는 샘플 `ncp-rc.yml`에는 다음 `ncp.ini` 정보가 포함되어 있습니다.

```
# ConfigMap for ncp.ini
apiVersion: v1
kind: ConfigMap
metadata:
  name: nsx-ncp-config
  labels:
    version: v1
data:
  ncp.ini: |
    [DEFAULT]

    # Set to True to enable logging to stderr
    #use_stderr = True

    # Set to True to send logs to the syslog daemon
    #use_syslog = False

    # Enabler debug-level logging for the root logger. If set to True, the
```

```

# root logger debug level will be DEBUG, otherwise it will be INFO.
#debug = True

# The log file path must be set to something like '/var/log/nsx-ujo/'. By
# default, logging to file is disabled.
#log_dir = None

# Name of log file to send logging output to. If log_dir is set but log_file is
# not, the binary name will be used, i.e., ncp.log, nsx_node_agent.log and
# nsx_kube_proxy.log.
#log_file = None

# max MB for each compressed file. Defaults to 100 MB
#log_rotation_file_max_mb = 100

# Total number of compressed backup files to store. Defaults to 5.
#log_rotation_backup_count = 5
[coe]
#
# Common options for Container Orchestrators
#

# Container orchestrator adaptor to plug in
# Options: kubernetes (default), openshift, pcf.
#adaptor = kubernetes

# Specify cluster for adaptor. It is a prefix of NSX resources name to
# distinguish multiple clusters who are using the same NSX.
# This is also used as the tag of IP blocks for cluster to allocate
# IP addresses. Different clusters should have different IP blocks.
#cluster = k8scluster

# Log level for the NCP operations. If set, overrides the level specified
# for the root logger. Possible values are NOTSET, DEBUG, INFO, WARNING,
# ERROR, CRITICAL
#loglevel=None

# Log level for the NSX API client operations. If set, overrides the level
# specified for the root logger. Possible values are NOTSET, DEBUG, INFO,
# WARNING, ERROR, CRITICAL
nsxlib_loglevel=INFO

# Once enabled, all projects in this cluster will be mapped to a NAT
# topology in NSX backend
#enable_snat = True

# The type of container node. Possible values are HOSTVM, BAREMETAL.
#node_type = HOSTVM

[ha]
#
# NCP High Availability configuration options
#

# Time duration in seconds of mastership timeout. NCP instance will

```

```

# remain master for this duration after elected. Note that the heartbeat
# period plus the update timeout must be less than this period. This
# is done to ensure that the master instance will either confirm
# liveness or fail before the timeout.
#master_timeout = 9

# Time in seconds between heartbeats for elected leader. Once an NCP
# instance is elected master, it will periodically confirm liveness based
# on this value.
#heartbeat_period = 3

# Timeout duration in seconds for update to election resource. If the
# update request does not complete before the timeout it will be
# aborted. Used for master heartbeats to ensure that the update finishes
# or is aborted before the master timeout occurs.
#update_timeout = 3

[k8s]
#
# From kubernetes
#

# IP address of the Kubernetes API Server. If not set, will try to
# read and use the Kubernetes Service IP from environment variable
# KUBERNETES_SERVICE_HOST.
#apiserver_host_ip = <ip_address>

# Port of the Kubernetes API Server.
# Set to 6443 for https. If not set, will try to
# read and use the Kubernetes Service port from environment
# variable KUBERNETES_SERVICE_PORT.
#apiserver_host_port = <port>

# Specify a CA bundle file to use in verifying the Kubernetes API server
# certificate. (string value)
#ca_file = <None>
ca_file = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

# Full path of the Token file to use for authenticating with the k8s API server.
#client_token_file = <None>
client_token_file = /var/run/secrets/kubernetes.io/serviceaccount/token

# Full path of the client certificate file to use for authenticating
# with the k8s API server. It must be specified together with
# "client_private_key_file"
#client_cert_file = <None>

# Full path of the client certificate file to use for authenticating
# with the k8s API server. It must be specified together with
# "client_cert_file"
#client_private_key_file = <None>

# Log level for the kubernetes adaptor. If set, overrides the level specified
# for the root logger. Possible values are NOTSET, DEBUG, INFO, WARNING,
# ERROR, CRITICAL

```

```

#loglevel=None

# Specify how ingress controllers are expected to be deployed. Possible values:
# hostnetwork or nat. NSX will create NAT rules only in the second case.
#ingress_mode = hostnetwork

[nsx_v3]
#
# From nsx
#

# IP address of one or more NSX managers separated by commas. The IP address
# should be of the form (list value):
# <ip_address1>[:<port1>],<ip_address2>[:<port2>],...
# HTTPS will be used for communication with NSX. If port is not provided,
# port 443 will be used.
#nsx_api_managers = <ip_address>

# If true, the NSX Manager server certificate is not verified. If false the CA
# bundle specified via "ca_file" will be used or if unset the default system
# root CAs will be used. (boolean value)
#insecure = False

# Specify one or a list of CA bundle files to use in verifying the NSX Manager
# server certificate. This option is ignored if "insecure" is set to True. If
# "insecure" is set to False and ca_file is unset, the system root CAs will be
# used to verify the server certificate. (list value)
#ca_file = <None>

# Path to NSX client certificate file. If specified, the nsx_api_user and
# nsx_api_password options will be ignored. This option must be specified
# along with "nsx_api_private_key_file" option.
#nsx_api_cert_file = <None>

# Path to NSX client private key file. If specified, the nsx_api_user and
# nsx_api_password options will be ignored. This option must be specified
# along with "nsx_api_cert_file" option.
#nsx_api_private_key_file = <None>

# The time in seconds before aborting a HTTP connection to a NSX manager.
# (integer value)
#http_timeout = 10

# The time in seconds before aborting a HTTP read response from a NSX manager.
# (integer value)
#http_read_timeout = 180

# Maximum number of times to retry a HTTP connection. (integer value)
#http_retries = 3

# Maximum concurrent connections to each NSX manager. (integer value)
#concurrent_connections = 10

# The amount of time in seconds to wait before ensuring connectivity to the NSX
# manager if no manager connection has been used. (integer value)

```

```

#conn_idle_timeout = 10

# Number of times a HTTP redirect should be followed. (integer value)
#redirects = 2

# Maximum number of times to retry API requests upon stale revision errors.
# (integer value)
#retries = 10

# Subnet prefix of IP block. IP block will be retrieved from NSX API and
# recognised by tag 'cluster'.
# Prefix should be less than 31, as two addresses(the first and last addresses)
# need to be network address and broadcast address.
# The prefix is fixed after the first subnet is created. It can be changed only
# if there is no subnets in IP block.
#subnet_prefix = 24

# Indicates whether distributed firewall DENY rules are logged.
#log_dropped_traffic = False

# Option to use native loadbalancer support.
#use_native_loadbalancer = False

# Used when ingress class annotation is missing
# if set to true, the ingress will be handled by nsx lbs
# otherwise will be handled by 3rd party ingress controller (e.g. nginx)
#default_ingress_class_nsx = True

# Path to the default certificate file for HTTPS load balancing
#lb_default_cert_path = <None>

# Path to the private key file for default certificate for HTTPS load balancing
#lb_priv_key_path = <None>

# Option to set load balancing algorithm in load balancer pool object.
# Available choices are
# ROUND_ROBIN/LEAST_CONNECTION/IP_HASH/WEIGHTED_ROUND_ROBIN
#pool_algorithm = 'ROUND_ROBIN'

# Option to set load balancer service size. Available choices are
# SMALL/MEDIUM/LARGE.
# MEDIUM Edge VM (4 vCPU, 8GB) only supports SMALL LB.
# LARGE Edge VM (8 vCPU, 16GB) only supports MEDIUM and SMALL LB.
# Bare Metal Edge (IvyBridge, 2 socket, 128GB) supports LARGE, MEDIUM and
# SMALL LB
#service_size = 'SMALL'

# Choice of persistence type for ingress traffic through L7 Loadbalancer.
# Accepted values:
# 'cookie'
# 'source_ip'
#l7_persistence = <None>

# Choice of persistence type for ingress traffic through L4 Loadbalancer.
# Accepted values:

```

```

# 'source_ip'
#l4_persistence = <None>

# Name or UUID of the tier0 router that project tier1 routers connect to
#tier0_router = <None>

# Name or UUID of the NSX overlay transport zone that will be used for creating
# logical switches for container networking. It must refer to an existing
# transport zone on NSX and every hypervisor that hosts the Kubernetes
# node VMs must join this transport zone
#overlay_tz = <None>

# Name or UUID of the NSX lb service that can be attached by virtual servers
#lb_service = <None>

# Name or UUID of the container ip blocks that will be used for creating
# subnets. If name, it must be unique
#container_ip_blocks = <None>

# Name or UUID of the container ip blocks that will be used for creating
# subnets for no-SNAT projects. If specified, no-SNAT projects will use these
# ip blocks ONLY. Otherwise they will use container_ip_blocks
#no_snat_ip_blocks = <None>

# Name or UUID of the external ip pools that will be used for allocating IP
# addresses which will be used for translating container IPs via SNAT rules
#external_ip_pools = <None>

# Name or UUID of the external ip pools that will be used for allocating IP
# addresses for exposing LoadBalancer type service and ingress
#external_ip_pools_lb = <None>

# Firewall sections for this cluster will be created below this mark section
#top_firewall_section_marker = <None>

# Firewall sections for this cluster will be created above this mark section
#bottom_firewall_section_marker = <None>

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

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

```

NCP 포트에 PEM으로 인코딩된 인증서 및 개인 키 마운트

PEM으로 인코딩된 인증서와 개인 키가 있는 경우 yaml 파일의 NCP 포트 정의를 업데이트하여 NCP 포트에 TLS 암호를 마운트할 수 있습니다.

- 1 인증서 및 개인 키에 대한 TLS 암호를 생성합니다.

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

- 2 NCP 포트 사양 yaml을 업데이트하여 암호를 NCP 포트 사양에 파일로 마운트합니다.

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

- 3 yaml 파일에서 nsx_v3 옵션 nsx_api_cert_file 및 nsx_api_private_key_file을 업데이트합니다.

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

NCP 포트에 인증서 파일 마운트

노드 파일 시스템에 인증서 파일이 있는 경우 NCP 포트 사양을 업데이트하여 해당 파일을 NCP 포트에 마운트할 수 있습니다.

예를 들면 다음과 같습니다.

```
spec:
  ...
  containers:
  - name: nsx-ncp
    ...
    volumeMounts:
    ...
    - name: nsx-cert
      # Mount path must match nsx_v3 option "nsx_api_cert_file"
      mountPath: /etc/nsx-uj0/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>
```

Syslog 구성

컨테이너에서 rsyslog 또는 syslog-ng와 같은 syslog 에이전트를 실행하여 NCP 및 관련 구성 요소의 로그를 syslog 서버로 보낼 수 있습니다.

다음과 같은 방법이 권장됩니다. Kubernetes의 로깅에 대한 자세한 내용은 <https://kubernetes.io/docs/concepts/cluster-administration/logging>을 참조하십시오.

- NCP 또는 nsx-node-agent 포트에서 실행되는 사이드카 컨테이너를 생성합니다.
- 모든 노드에서 DaemonSet 복제를 실행합니다.

참고 사이드카 컨테이너 방법을 사용하면 NSX CNI 플러그인이 포트에서 실행되지 않기 때문에 플러그인 로그를 syslog 서버로 보낼 수 없습니다.

Syslog용 사이드카 컨테이너 생성

NCP와 동일한 포트에서 실행되도록 syslog용 사이드카 컨테이너를 구성할 수 있습니다. 다음 절차에서는 syslog 에이전트 이미지가 example/rsyslog라고 가정합니다.

절차

- 1 파일에 기록하도록 NCP 및 NSX 노드 에이전트를 구성합니다.

NCP 및 NSX 노드 에이전트의 yaml 파일에서 log_dir 매개 변수를 설정하고 마운트할 볼륨을 지정합니다. 예를 들면 다음과 같습니다.

```
[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
```

`log_file` 매개 변수를 설정하여 로그 파일 이름을 변경할 수 있습니다. 기본 이름은 `ncp.log`, `nsx_node_agent.log` 및 `nsx_kube_proxy.log`입니다. `log_dir` 옵션을 `/var/log/nsx-ujo` 경로 이외의 경로로 설정하면 `hostPath` 볼륨 또는 `emptyDir` 볼륨을 생성하여 해당 pod 사양에 마운트해야 합니다.

2 호스트 경로가 있고 사용자 `nsx-ncp`가 호스트 경로를 쓸 수 있는지 확인합니다.

a 다음 명령을 실행합니다.

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

b 사용자 `nsx-ncp`를 추가하거나 호스트 경로의 모드를 777로 변경합니다.

```
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 NCP 포트의 사양 yaml 파일에서 syslog용 ConfigMap을 추가합니다. 예를 들면 다음과 같습니다.

```
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 NCP 포드의 yaml 파일에서 **rsyslog** 컨테이너를 추가하고, **rsyslog**가 구성 데이터를 찾고 다른 컨테이너에서 로그를 읽을 수 있는 적절한 볼륨을 마운트합니다. 예를 들면 다음과 같습니다.

```
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>
```

Syslog용 DaemonSet 복제 생성

다음 방법으로 모든 NCP 구성 요소의 로그를 리디렉션할 수 있습니다. 애플리케이션은 **stderr**에 로깅하도록 구성해야 합니다. 이 구성은 기본적으로 사용되도록 설정됩니다. 다음 절차에서는 **syslog** 에이전트 이미지가 **example/rsyslog**라고 가정합니다.

절차

- 1 DaemonSet yaml 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
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 DaemonSet을 생성합니다.

```
kubect1 apply -f <daemonset yaml file>
```

예: 사이드카 컨테이너에서 실행되는 로그 순환 및 Syslog 구성

다음 절차는 사이드카 컨테이너에서 실행되는 로그 순환 및 syslog를 구성하는 방법을 보여줍니다.

로그 디렉토리 생성 및 로그 순환 구성

- 마스터 노드를 포함한 모든 노드에 로그 디렉토리를 생성하고 ID가 1000인 사용자로 소유자를 변경합니다.

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

- 모든 노드의 /var/log/nsx-ujo 디렉토리에 대한 로그 순환을 구성합니다.

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

NCP 복제 컨트롤러 생성

- NCP에 대한 ncp.ini 파일을 생성합니다.

```
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
```

- ini 파일에서 configmap을 생성합니다.

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

- NCP rsyslog 구성을 생성합니다.

```
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
```

- 위 구성에서 configmap을 생성합니다.

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

- rsyslog 사이드카를 사용하여 NCP 복제 컨트롤러를 생성합니다.

```
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

```

- 위의 규격으로 NCP를 생성합니다.

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

NSX 노드 에이전트 DaemonSet 생성

- 노드 에이전트에 대한 rsyslog 구성을 생성합니다.

```

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

```

- 위의 구성에서 configmap을 생성합니다.

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

- configmap 사이드카를 사용하여 DaemonSet을 생성합니다.

```

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

- DaemonSet을 생성합니다.

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

보안 고려 사항

NCP를 배포할 때에는 Kubernetes와 NSX-T Data Center 환경 둘 다를 보호하기 위한 조치를 취하는 것이 중요합니다.

NCP가 지정된 노드에서만 실행되도록 제한

NCP는 NSX-T Data Center 관리부에 액세스할 수 있어야 하며 지정된 인프라 노드에서만 실행되도록 제한해야 합니다. 적절한 레이블을 사용하여 이러한 노드를 식별할 수 있습니다. 그런 다음 이 레이블의 nodeSelector를 NCP ReplicationController 사양에 적용해야 합니다. 예를 들면 다음과 같습니다.

```

nodeSelector:
  nsx-infra: True

```

선호도와 같은 기타 메커니즘을 사용하여 노드에 포드를 할당할 수도 있습니다. 자세한 내용은 <https://kubernetes.io/docs/concepts/configuration/assign-pod-node> 항목을 참조하십시오.

Docker Engine이 최신 버전인지 확인

Docker는 정기적으로 보안 업데이트를 릴리스합니다. 이러한 업데이트를 적용하기 위해서는 자동화된 절차를 구현해야 합니다.

신뢰할 수 없는 컨테이너의 NET_ADMIN 및 NET_RAW 기능 허용 안 함

Linux 기능인 NET_ADMIN 및 NET_RAW는 공격자가 포트 네트워크를 손상시키는 데 악용될 수 있습니다. 신뢰할 수 없는 컨테이너의 이러한 두 가지 기능을 사용하지 않도록 설정해야 합니다. 기본적으로 NET_ADMIN 기능은 권한 없는 컨테이너에 부여되지 않습니다. 포트 사양에서 명시적으로 사용하도록 설정하거나 컨테이너를 권한 모드로 설정하는 경우 주의해야 합니다. 또한, 신뢰할 수 없는 컨테이너의 경우 컨테이너 사양의 SecurityContext 구성에서 삭제된 기능 목록에 NET_RAW를 지정하여 NET_RAW를 사용하지 않도록 설정합니다. 예를 들면 다음과 같습니다.

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

역할 기반 액세스 제어

Kubernetes는 RBAC(역할 기반 액세스 제어) API를 사용하여 권한 결정을 내리고 관리자가 정책을 동적으로 구성할 수 있도록 합니다. 자세한 내용은 <https://kubernetes.io/docs/admin/authorization/rbac> 항목을 참조하십시오.

일반적으로 클러스터 관리자는 권한 있는 액세스 및 역할을 갖는 유일한 사용자입니다. 사용자 및 서비스 계정의 경우 액세스 권한을 부여할 때 최소 권한 원칙을 따라야 합니다.

다음 지침이 권장됩니다.

- Kubernetes API 토큰에 대한 액세스를 해당 토큰이 필요한 포드로 제한합니다.
- NCP ConfigMap 및 NSX API 클라이언트 인증서의 TLS 암호에 대한 액세스를 NCP 포드로 제한합니다.
- 이러한 액세스가 필요 없는 포드의 Kubernetes 네트워킹 API 액세스를 차단합니다.
- Kubernetes API에 액세스할 수 있는 포드를 지정하는 Kubernetes RBAC 정책을 추가합니다.

NCP 포드에 대한 권장 RBAC 정책

ServiceAccount 아래에 NCP 포드를 생성하고 이 계정에 최소한의 권한 집합을 부여합니다. 또한, 다른 포트 또는 ReplicationController가 NCP ReplicationController 및 NSX 노드 에이전트의 볼륨으로 마운트된 ConfigMap 및 TLS 암호에 액세스하도록 허용하지 마십시오.

다음 예제는 NCP에 대한 역할 및 역할 바인딩을 지정하는 방법을 보여 줍니다.

```
# 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
```

참고 NSX-T Data Center 클라이언트 인증서 및 개인 키 쌍에 대해 Kubernetes API를 사용하여 생성된 TLS 암호를 사용하면 Kubernetes API 서버에 액세스할 수 있는 모든 포트에 액세스할 수 있습니다. 마찬가지로 서비스 계정 없이 생성된 포트에는 Kubernetes API에 액세스할 수 있도록 하기 위해 토큰을 자동으로 마운트하는 동일한 네임스페이스의 기본 서비스 계정이 자동으로 할당됩니다. 따라서 이러한 토큰에 대한 액세스는 해당 토큰이 필요한 포트에 제한해야 합니다.

네트워크 리소스 구성 팁

일부 네트워크 리소스의 경우 특정 제한 사항을 숙지한 후 구성해야 합니다.

NSX-T Data Center 태깅 제한

NSX-T Data Center에서 개체를 태깅할 때는 다음과 같은 제한이 적용됩니다.

- 범위는 128자로 제한됩니다.
- 태그는 256자로 제한됩니다.
- 각 개체는 최대 30개의 태그를 가질 수 있습니다.

Kubernetes 또는 OpenShift 주석을 NSX-T Data Center 범위 및 태그에 복사할 때 이러한 제한이 초과되면 문제가 발생할 수 있습니다. 예를 들어 스위치 포트에 대한 태그가 방화벽 규칙에 사용되는 경우 주석 키나 값을 범위 또는 태그에 복사할 때 해당하는 주석 키나 값이 잘리면 규칙이 예상한 대로 적용되지 않을 수 있습니다.

네트워크 정책 구성

네트워크 정책은 레이블 선택기를 사용하여 포트 또는 네임스페이스를 선택합니다.

NCP에서 지원하는 네트워크 정책은 Kubernetes에서 지원하는 네트워크 정책과 동일하며, 이는 Kubernetes 버전에 따라 다릅니다.

- Kubernetes 1.11 - 다음 규칙 선택기를 지정할 수 있습니다.
 - **podSelector**: 네트워크 정책이 생성된 네임스페이스에 있는 모든 포드를 선택합니다.
 - **namespaceSelector**: 모든 네임스페이스를 선택합니다.
 - **podSelector AND namespaceSelector**: namespaceSelector로 선택한 네임스페이스에 있는 모든 포드를 선택합니다.
 - **ipBlockSelector**: ipBlockSelector를 namespaceSelector 또는 podSelector와 결합하면 네트워크 정책이 올바르지 않습니다. ipBlockSelector는 정책 규칙에 단독으로 사용해야 합니다.
- Kubernetes 1.10 - 네트워크 정책의 규칙 절에 namespaceSelector, podSelector 및 ipBlock 선택기를 최대 하나만 포함할 수 있습니다.

Kubernetes API 서버는 네트워크 정책 규격에 대한 검증을 수행하지 않습니다. 따라서 잘못된 네트워크 정책을 생성할 가능성이 있습니다. **NCP**는 이러한 네트워크 정책을 거부합니다. 네트워크 정책을 업데이트하여 올바르게 만들더라도 **NCP**가 해당 네트워크 정책을 처리하지 않습니다. 이 경우에는 네트워크 정책을 삭제한 후 규격이 올바른 네트워크 정책을 다시 생성해야 합니다.

Pivotal Cloud Foundry 환경에서 NCP 설치

4

PCF(Pivotal Cloud Foundry)는 오픈 소스 PaaS(Platform-as-a-Service) 제공자입니다. PCF 환경에 NCP(NSX Container Plug-in)를 설치하여 네트워킹 서비스를 제공할 수 있습니다.

Pivotal Ops Manager를 통해 생성되는 VM은 컨테이너 네트워크에 대한 계층 3 연결을 통해 NSX-T 기능에 액세스합니다.

HA(고가용성) 기능은 자동으로 사용하도록 설정됩니다.

참고 보안 그룹을 변경하는 경우 보안 그룹이 적용되는 모든 애플리케이션을 다시 스테이징해야 합니다. 이 작업은 애플리케이션이 실행되는 공간에 보안 그룹이 적용되거나 보안 그룹이 글로벌인 경우 필요할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- [Pivotal Cloud Foundry 환경에서 NCP 설치](#)

Pivotal Cloud Foundry 환경에서 NCP 설치

NCP는 Pivotal Ops Manager 그래픽 사용자 인터페이스를 통해 설치됩니다.

사전 요구 사항

Pivotal Ops Manager, NSX-T Data Center 및 Pivotal Application Service(PAS)를 새로 설치합니다. Ops Manager를 먼저 설치한 후 NSX-T Data Center, PAS 순서로 설치해야 합니다. 자세한 내용은 Pivotal Cloud Foundry 설명서를 참조하십시오.

절차

- 1 PCF에 대한 NCP 설치 파일을 다운로드합니다.
파일 이름은 VMware-NSX-T.<version>.<build>.pivotal입니다.
- 2 Pivotal Ops Manager에 관리자로 로그인합니다.
- 3 **제품 가져오기(Import a Product)**를 클릭합니다.
- 4 다운로드한 파일을 선택합니다.
- 5 **Ops Manager Director for VMware vSphere** 타일을 클릭합니다.

- 6 **vCenter 구성의 설정** 탭에서 **NSX 네트워킹**을 선택하고 **NSX 모드**에서 **NSX-T**를 선택합니다.
- 7 **NSX CA 인증서** 필드에 PEM 형식의 인증서를 입력합니다.
- 8 **저장**을 클릭합니다.
- 9 왼쪽 위의 **설치 대시보드**를 클릭하여 대시보드로 돌아갑니다.
- 10 **Pivotal Application Service** 타일을 클릭합니다.
- 11 **설정** 탭의 탐색 창에서 **네트워킹**을 선택합니다.
- 12 **컨테이너 네트워크 인터페이스 플러그인**에서 **외부**를 선택합니다.
- 13 왼쪽 위의 **설치 대시보드**를 클릭하여 대시보드로 돌아갑니다.
- 14 **저장**을 클릭합니다.
- 15 왼쪽 위의 **설치 대시보드**를 클릭하여 대시보드로 돌아갑니다.
- 16 **VMware NSX-T** 타일을 클릭합니다.
- 17 NSX Manager의 주소를 입력합니다.
- 18 NSX Manager 인증 방법을 선택합니다.

옵션	작업
클라이언트 인증서 인증	NSX Manager의 인증서 및 개인 키를 입력합니다.
사용자 이름 및 암호를 사용한 기본 인증	NSX Manager 관리자의 사용자 이름과 암호를 입력합니다.

- 19 **NSX Manager CA 인증서** 필드에 인증서를 입력합니다.
- 20 **저장**을 클릭합니다.
- 21 탐색 창에서 **NCP**를 선택합니다.
- 22 **PAS 기반 이름**을 입력합니다.
이 문자열은 NSX API에서 PAS 기반을 고유하게 식별합니다. 또한 이 문자열은 NCP가 PAS 기반으로 생성한 NSX 리소스 이름의 접두사로 사용됩니다.
- 23 **오버레이 전송 영역**을 입력합니다.
- 24 **Tier-0 라우터**를 입력합니다.
- 25 **컨테이너 네트워크의 IP 블록**을 하나 이상 지정합니다.
 - a **추가**를 클릭합니다.
 - b **IP 블록 이름**을 입력합니다. 새 IP 블록 또는 기존 IP 블록일 수 있습니다.
 - c 새 IP 블록인 경우에만 CIDR 형식으로 블록을 지정합니다(예: 10.1.0.0/16).
- 26 **컨테이너 네트워크의 서브넷 접두사**를 지정합니다.
- 27 **컨테이너 네트워크에 SNAT 사용**을 클릭하여 SNAT를 사용하도록 설정합니다.

28 조직 네트워크에 외부(NAT) IP 주소를 제공하는 데 사용되는 IP 풀을 하나 이상 지정합니다.

- a **추가**를 클릭합니다.
- b **IP 풀 이름**을 입력합니다. 새 IP 풀 또는 기존 IP 풀일 수 있습니다.
- c 새 IP 풀인 경우에만 CIDR 및 IP 범위를 제공하여 IP 주소를 지정합니다.

29 (선택 사항) 상단 방화벽 섹션 마커를 입력합니다.

30 (선택 사항) 하단 방화벽 섹션 마커를 입력합니다.

31 (선택 사항) 다음 옵션을 사용하거나 사용하지 않도록 설정합니다.

옵션	기본값
삭제된 애플리케이션 트래픽 기록	사용 안 함. 사용하도록 설정하면 방화벽 규칙에 의해 삭제된 트래픽이 기록됩니다.
NCP 로깅에 디버그 수준 사용	사용.

32 저장을 클릭합니다.

33 (선택 사항) 탐색 창에서 **NSX 노드 에이전트**를 선택합니다.

- a **NSX 노드 에이전트에 디버그 수준 로깅 사용**을 선택하여 디버그 수준 로깅을 사용하도록 설정합니다.
- b **저장**을 클릭합니다.

34 왼쪽 위의 **설치 대시보드**를 클릭하여 대시보드로 돌아갑니다.

35 변경 내용 적용을 클릭합니다.

로드 밸런싱

5

NSX-T Data Center 로드 밸런서는 Kubernetes와 통합되어 있습니다.

본 장은 다음 항목을 포함합니다.

- 로드 밸런싱 구성
- 타사 수신 컨트롤러

로드 밸런싱 구성

로드 밸런싱을 구성하려면 Kubernetes LoadBalancer 서비스 또는 수신 리소스와 NCP 복제 컨트롤러를 구성해야 합니다.

LoadBalancer 유형의 Kubernetes 서비스를 구성하여 계층 4 로드 밸런서를 생성하고, Kubernetes 수신 리소스를 구성하여 계층 7 로드 밸런서를 생성할 수 있습니다.

NCP에서 로드 밸런싱을 구성하려면 `nep-rc.yml` 파일에서 다음을 수행합니다.

- 1 `use_native_loadbalancer = True`로 설정합니다.
- 2 (선택 사항) `pool_algorithm`을 `ROUND_ROBIN` 또는 `LEAST_CONNECTION/IP_HASH`로 설정합니다. 기본값은 `ROUND_ROBIN`입니다.
- 3 (선택 사항) `service_size` = `SMALL`, `MEDIUM` 또는 `LARGE`로 설정합니다. 기본값은 `SMALL`입니다.

`LEAST_CONNECTION/IP_HASH` 알고리즘은 동일한 소스 IP 주소의 트래픽을 동일한 백엔드 포드로 전송합니다.

다른 크기의 NSX-T 로드 밸런서가 지원하는 항목에 대한 자세한 내용은 "NSX-T Data Center 관리 가이드"를 참조하십시오.

로드 밸런서가 생성된 후 로드 밸런서 크기는 구성 파일을 업데이트하여 변경할 수 없습니다. 대신 NSX Manager UI 또는 API를 통해 변경할 수 있습니다.

계층 4 및 계층 7 로드 밸런싱의 지속성 설정

NCP ConfigMap에서 `l4_persistence` 및 `l7_persistence` 매개 변수를 사용하여 지속성 설정을 지정할 수 있습니다. 계층 4 지속성에 대해서는 소스 IP를 옵션으로 사용할 수 있습니다. 계층 7 지속성에 대해서는 쿠키 및 소스 IP를 옵션으로 사용할 수 있습니다. 기본값은 `<None>`입니다. 예를 들면 다음과 같습니다.

```
# 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
```

Kubernetes LoadBalancer 서비스의 경우 글로벌 계층 4 지속성이 꺼져 있는 경우(`l4_persistence`가 `<None>`으로 설정되어 있음) 서비스 규격에서 `sessionAffinity`를 지정하여 서비스의 지속성 동작을 구성할 수 있습니다. `l4_persistence`를 `source_ip`로 설정한 경우 서비스 규격의 `sessionAffinity`를 사용하여 서비스에 대한 지속성 시간 초과를 사용자 지정할 수 있습니다. 기본 계층 4 지속성 시간 초과는

10800초입니다(서비스에 대한 Kubernetes 설명서 <https://kubernetes.io/docs/concepts/services-networking/service>에 지정된 것과 동일). 기본 지속성 시간 초과가 지정된 모든 서비스는 동일한 NSX-T 로드 밸런서 지속성 프로파일을 공유합니다. 기본값이 아닌 지속성 시간 초과가 지정된 각 서비스에 대해 전용 프로파일이 생성됩니다.

참고 수신 백엔드 서비스가 LoadBalancer 유형의 서비스인 경우 해당 서비스에 대한 계층 4 가상 서버와 수신에 대한 계층 7 가상 서버에 서로 다른 지속성 설정(예: 계층 4에 대해 **source_ip**, 계층 7에 대해 **cookie**)을 지정할 수 없습니다. 이러한 시나리오에서는 두 가상 서버에 대한 지속성 설정이 동일하거나 (**source_ip**, **cookie** 또는 **None**) 그 중 하나가 **None**여야 합니다(이 경우 기타 설정은 **source_ip** 또는 **cookie**일 수 있음). 이러한 시나리오의 예는 다음과 같습니다.

```
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
```

수신

- NSX-T Data Center는 TLS 규격을 사용하는 수신에 대해 하나의 계층 7 로드 밸런서를 생성하고 TLS 규격을 사용하지 않는 수신에 대해 하나의 계층 7 로드 밸런서를 생성합니다.
- 모든 수신에는 단일 IP 주소가 할당됩니다.

- 수신 리소스에는 `ncp.ini`의 `[nsx_v3]` 섹션에 있는 `external_ip_pools` 옵션에서 지정한 외부 IP 풀의 IP 주소가 할당됩니다. 로드 밸런서는 이 IP 주소와 HTTP 및 HTTPS 포트(80 및 443)를 사용합니다.
- 수신 리소스에는 `ncp.ini`의 `[nsx_v3]` 섹션에 있는 `external_ip_pools_lb` 옵션에서 지정한 외부 IP 풀의 IP 주소가 할당됩니다. `external_ip_pools_lb` 옵션이 없는 경우 `external_ip_pools`에서 지정한 풀이 사용됩니다. 로드 밸런서는 이 IP 주소와 HTTP 및 HTTPS 포트(80 및 443)를 사용합니다.
- 구성을 변경하고 NCP를 다시 시작하여 다른 IP 풀로 변경할 수 있습니다.
- TLS에 대해 기본 인증서를 지정할 수 있습니다. 인증서를 생성하고 NCP 포트에 마운트하는 데 대한 내용은 아래를 참조하십시오.
- TLS 규격을 사용하지 않는 수신은 HTTP 가상 서버(포트 80)에서 호스팅됩니다.
- TLS 규격을 사용하는 수신은 HTTPS 가상 서버(포트 443)에서 호스팅됩니다. 로드 밸런서는 SSL 서버 역할을 하며 클라이언트 SSL 연결을 종료합니다.
- 암호 및 수신 생성 순서는 중요하지 않습니다. 암호 개체가 존재하고 이를 참조하는 수신이 있으면 NSX-T Data Center가 인증서를 가져옵니다. 암호가 삭제되거나 암호를 참조하는 마지막 수신이 삭제되면 암호에 해당하는 인증서가 삭제됩니다.
- TLS 섹션을 추가하거나 제거하여 수신을 수정하는 작업이 지원됩니다. 수신 규격에서 `tls` 키가 제거되면 HTTPS 가상 서버(포트 443)에서 HTTP 가상 서버(포트 80)로 수신 규격이 전송됩니다. 마찬가지로 수신 규격에 `tls` 키가 추가되면 HTTP 가상 서버(포트 80)에서 HTTPS 가상 서버(포트 443)로 수신 규격이 전송됩니다.
- 단일 클러스터에 대한 수신 정의에 중복된 규칙이 있는 경우 첫 번째 규칙만 적용됩니다.
- 각 클러스터당 기본 백엔드가 있는 하나의 수신만 지원됩니다. 수신 규칙과 일치하지 않는 트래픽은 기본 백엔드로 전달됩니다.
- 기본 백엔드가 있는 수신이 여러 개인 경우 첫 번째 수신만 구성됩니다. 다른 수신에는 오류 주석이 추가됩니다.
- 와일드카드 URI 일치가 정규식 문자 “.”를 사용하여 지원됩니다. 및 “*”을 사용하여 지원됩니다. 예를 들어, 경로 “/coffee/.”는 “/coffee/” 뒤에 0 또는 하나 이상의 문자가 나오는 경우와 일치합니다. 예를 들어 “/coffee/”, “/coffee/a”, “/coffee/b”와는 일치하지만 “/coffee”, “/coffeecup” 또는 “/coffeecup/a”와는 일치하지 않습니다.

수신 규격 예:

```
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

```

- 수신 리소스에 주석을 추가하여 URL 요청 재작성을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```

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

```

URL이 백엔드 서비스로 전송되기 전에 경로 `/tea` 및 `/coffee`가 `/`로 재작성됩니다.

- 수신 주석 `kubernetes.io/ingress.allow-http`가 지원됩니다.
 - 주석이 **false**로 설정되어 있으면 HTTPS 규칙만 생성됩니다.
 - 주석이 **true**로 설정되어 있거나 누락된 경우에는 HTTP 규칙이 생성됩니다. 또한 수신 규격에 TLS 섹션이 있는 경우 HTTPS 규칙이 생성됩니다.
- 오류는 수신 리소스에 주석으로 추가됩니다. 오류 키는 `ncp/error.loadbalancer`이고 주의 키는 `ncp/warning.loadbalancer`입니다. 다음과 같은 오류 및 주의가 발생할 수 있습니다.
 - `ncp/error.loadbalancer: DEFAULT_BACKEND_IN_USE`
 이 오류는 기본 백엔드가 있는 수신에 이미 있음을 나타냅니다. 이 경우 수신은 비활성 상태입니다. TLS 사용 여부와 관계없이 수신 그룹에는 기본 백엔드가 하나만 있어야 합니다. 이 오류를 해결하려면 수신을 삭제한 후 올바른 규격을 사용하여 다시 생성하십시오.
 - `ncp/warning.loadbalancer: SECRET_NOT_FOUND`
 이 오류는 수신 규격에 지정된 암호가 없음을 나타냅니다. 이 경우 수신은 부분적으로 활성 상태입니다. 이 오류를 해결하려면 누락된 암호를 생성하십시오. 참고로 주석에 주의가 있으면 해당 주의는 수신 리소스의 수명주기 동안 지워지지 않습니다.
 - `ncp/warning.loadbalancer: INVALID_INGRESS`

이 오류는 다음 조건 중 하나가 **true**임을 나타냅니다. 이 경우 수신은 비활성 상태입니다. 이 오류를 해결하려면 수신을 삭제한 후 올바른 규격을 사용하여 다시 생성하십시오.

- 수신 규칙이 동일한 Kubernetes 클러스터의 다른 수신 규칙과 충돌합니다.
- **allow-http** 주석이 **False**로 설정되고 수신에 TLS 섹션이 포함되지 않습니다.
- 수신 규칙에 **host** 및 **path**가 지정되어 있지 않습니다. 이러한 수신 규칙에 수신 기본 백엔드와 동일한 기능이 있습니다. 대신 수신 기본 백엔드를 사용합니다.

유형 LoadBalancer의 서비스

- NSX-T Data Center는 각 서비스 포트에 대해 계층 4 로드 밸런서 가상 서버와 풀을 생성합니다.
- TCP와 UDP가 모두 지원됩니다.
- 각 서비스에는 고유한 IP 주소가 할당됩니다.
- 이 서비스에는 LoadBalancer 정의의 **loadBalancerIP** 필드를 기준으로 외부 IP 풀의 IP 주소가 할당됩니다. **loadBalancerIP** 필드는 비워 둘 수 있고 IP 풀의 IP 주소, 이름 또는 ID를 포함할 수 있습니다. **loadBalancerIP** 필드가 비어 있으면 **ncp.ini**의 **[nsx_v3]** 섹션에 있는 **external_ip_pools_lb** 옵션에서 지정한 외부 IP 풀에서 IP가 할당됩니다. **external_ip_pools_lb** 옵션이 없는 경우 **external_ip_pools**에서 지정한 풀이 사용됩니다. LoadBalancer 서비스는 이 IP 주소와 서비스 포트를 사용합니다.
- 구성을 변경하고 NCP를 다시 시작하여 다른 IP 풀로 변경할 수 있습니다.
- **loadBalancerIP**에서 지정한 IP 풀에는 태그 **scope: ncp/owner, tag: cluster:<cluster_name>**이 있어야 합니다.
- 오류는 서비스에 주석으로 추가됩니다. 오류 키는 **ncp/error.loadbalancer**입니다. 다음과 같은 오류가 발생할 수 있습니다.
 - **ncp/error.loadbalancer: IP_POOL_NOT_FOUND**

이 오류는 사용자가 **loadBalancerIP: <nsx-ip-pool>**을 지정했지만 **<nsx-ip-pool>**이 없음을 나타냅니다. 이 경우 서비스는 비활성 상태입니다. 이 오류를 해결하려면 올바른 IP 풀을 지정하고, 서비스를 삭제한 후 다시 생성하십시오.
 - **ncp/error.loadbalancer: IP_POOL_EXHAUSTED**

이 오류는 사용자가 **loadBalancerIP: <nsx-ip-pool>**을 지정했지만 해당 IP 풀이 IP 주소를 모두 사용했음을 나타냅니다. 이 경우 서비스는 비활성 상태입니다. 이 오류를 해결하려면 사용 가능한 IP 주소가 있는 IP 풀을 지정하고, 서비스를 삭제한 후 다시 생성하십시오.
 - **ncp/error.loadbalancer: IP_POOL_NOT_UNIQUE**

이 오류는 여러 IP 풀에 **loadBalancerIP: <nsx-ip-pool>**에서 지정한 이름이 있음을 나타냅니다. 이 경우 서비스는 비활성 상태입니다.
 - **ncp/error.loadbalancer: POOL_ACCESS_DENIED**

이 오류는 `loadBalancerIP`에서 지정한 IP 풀에 태그 `scope: ncp/owner`, `tag: cluster:<cluster_name>`가 없거나 태그에 지정된 클러스터가 Kubernetes 클러스터의 이름과 일치하지 않음을 나타냅니다.

■ `ncp/error.loadbalancer: LB_VIP_CONFLICT`

이 오류는 `loadBalancerIP` 필드의 IP가 활성 서비스의 IP와 동일함을 나타냅니다. 이 경우 서비스는 비활성 상태입니다.

- 계층 4 로드 밸런서의 자동 크기 조정이 지원됩니다. Kubernetes LoadBalancer 서비스가 생성 또는 수정되어 추가 가상 서버가 필요하고 기존의 계층 4 로드 밸런서에 용량이 충분하지 않으면 새로운 계층 4 로드 밸런서가 생성됩니다. 또한 NCP는 더 이상 가상 서버가 연결되지 않은 계층 4 로드 밸런서를 삭제합니다. 이 기능은 기본적으로 사용하도록 설정되어 있습니다. NCP ConfigMap에서 `lb_lb_auto_scaling`을 `false`로 설정하여 이 기능을 사용하지 않도록 설정할 수 있습니다.

로드 밸런서 및 네트워크 정책

NSX 로드 밸런서 가상 서버에서 포드로 트래픽을 전달하는 경우 소스 IP는 Tier-1 라우터의 업링크 포트 IP 주소입니다. 이 주소는 개인 Tier-1 전송 네트워크에 있으며, 이 때문에 CIDR 기반 네트워크 정책이 허용해야 할 트래픽을 허용하지 않을 수 있습니다. 이 문제를 방지하려면 Tier-1 라우터의 업링크 포트 IP 주소가 허용되는 CIDR 블록에 포함되도록 네트워크 정책을 구성해야 합니다. 이 내부 IP 주소는 `status.loadbalancer.ingress.ip` 필드와 수신 리소스의 주석(`ncp/internal_ip_for_policy`)으로 표시됩니다.

예를 들어, 가상 서버의 외부 IP 주소가 4.4.0.5이고 내부 Tier-1 라우터의 업링크 포트 IP 주소가 100.64.224.11인 경우 상태는 다음과 같습니다.

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

유형 LoadBalancer 리소스의 수신 및 서비스에 대한 주석은 다음과 같습니다.

```
ncp/internal_ip_for_policy: 100.64.224.11
```

IP 주소 100.64.224.11은 네트워크 정책에서 `ipBlock` 선택기의 허용된 CIDR에 속해 있어야 합니다. 예를 들면 다음과 같습니다.

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

CA 서명된 인증서를 생성하는 샘플 스크립트

아래 스크립트는 <filename>.crt 및 <filename>.key 파일에 각각 저장된 CA 서명된 인증서와 개인 키를 생성합니다. `genrsa` 명령은 CA 키를 생성합니다. CA 키는 암호화해야 합니다. `aes256`과 같은 명령을 사용하여 암호화 방법을 지정할 수 있습니다.

```
#!/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
```

NCP 포트에 기본 인증서 및 키 마운트

인증서와 개인 키가 생성된 후 호스트 VM의 `/etc/nsx-ujo` 디렉토리에 저장합니다. 인증서와 키 파일의 이름이 `lb-default.crt` 및 `lb-default.key`로 각각 지정되었다고 가정하고 호스트의 해당 파일이 포트에 마운트되도록 `ncp-rc.yaml`을 편집합니다. 예를 들면 다음과 같습니다.

```
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
```

타사 수신 컨트롤러

타사 수신 컨트롤러를 지원하도록 NCP를 구성할 수 있습니다.

ncp.ini 파일 편집

구성 파일 `/var/vcap/data/jobs/ncp/xxxxxxx/config/ncp.ini`를 편집해야 합니다(여기서 `xxxxxxx`는 BOSH 배포 ID임). 그러면 NCP가 다시 시작될 때마다 이 파일이 `rootfs`에 복사되고 NCP에서 사용됩니다. 이 파일은 각 마스터 노드에서 편집해야 합니다.

중요 `ncp.ini`의 변경 내용은 PKS 클러스터 업데이트 시 지속되지 않습니다. PKS 타일을 통해 변경한 다음, PKS 배포를 업데이트하는 경우 `ncp.ini`의 변경 내용이 손실됩니다.

관련 옵션은 `nsx_v3` 섹션에 있습니다.

- `use_native_loadbalancer` - **False**로 설정하면 NCP는 주석과 관계없이 LoadBalancer 유형 업데이트의 모든 수신 또는 서비스를 처리하지 않습니다. 이 설정은 전체 PKS 클러스터에 적용됩니다. 기본값은 **True**입니다.
- `default_ingress_class_nsx` - **True**로 설정하면 NCP는 기본 수신 컨트롤러가 되고, `kubernetes.io/ingress.class: "nsx"` 주석이 지정된 수신과 어떤 주석도 지정되지 않은 수신을 모두 처리합니다. **False**로 설정하면 NCP는 `kubernetes.io/ingress.class: "nsx"`로 주석이 지정된 수신만 처리합니다. 기본값은 **True**입니다.

NCP가 부동 IP를 NGINX 컨트롤러 포트에 할당하고 부동 IP를 사용하여 수신 상태를 업데이트하도록 하려면 다음을 수행합니다.

- `ncp.ini`의 `k8s` 섹션에서 `ingress_mode=nat`를 설정합니다.
- NGINX 수신 컨트롤러 포트에 주석 `ncp/ingress-controller: "True"`를 추가합니다.

NCP가 NGINX 수신 컨트롤러 포트의 부동 IP를 사용하여 주석 `kubernetes.io/ingress.class: "nginx"`가 지정된 수신 상태를 업데이트합니다. `default_ingress_class_nsx=False`인 경우 NCP가 NGINX 수신 컨트롤러 포트의 부동 IP를 사용하여 주석 `kubernetes.io/ingress.class`가 지정되지 않은 수신 상태도 업데이트합니다.

참고: NGINX 수신 컨트롤러 포트에 `ncp/ingress-controller: "True"` 주석이 지정되지 않은 경우에도 NCP가 위에 언급된 수신 상태를 `loadBalancer: {}`로 업데이트합니다. 그러면 NGINX 컨트롤러가 수신 상태를 `loadBalancer: {ingress: [{ip: <IP>}]}`로 업데이트하고 NCP가 수신 상태를 `loadBalancer: {}`로 업데이트하는 루프에서 수신이 중단될 수 있습니다. 이러한 상황을 피하려면 다음 단계를 수행합니다.

- 수신 컨트롤러를 <https://github.com/kubernetes/ingress-nginx>에서 가져온 경우
 - 수신 컨트롤러에서 `ingress-class`를 "nginx" 이외의 다른 항목으로 변경합니다.
 - `kubernetes.io/ingress-class: "nginx"` 주석이 지정된 수신인 경우 주석을 다른 값으로 변경합니다.
 - 자세한 내용은 <https://kubernetes.github.io/ingress-nginx/user-guide/multiple-ingress>를 참조하십시오.

- 수신 컨트롤러를 <https://github.com/nginxinc/kubernetes-ingress>에서 가져온 경우
 - 수신 컨트롤러에서 `ingress-class`를 "nginx" 이외의 다른 항목으로 변경합니다.
 - `kubernetes.io/ingress-class: "nginx"` 주석이 지정된 수신에 주석을 다른 값으로 변경합니다.
 - 수신 컨트롤러 포트에서 `use-ingress-class-only`를 **True**로 설정합니다. 이렇게 하면 이 컨트롤러는 `kubernetes.io/ingress-class` 주석이 지정되지 않은 수신을 더 이상 업데이트하지 않습니다.
 - 자세한 내용은 <https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/multiple-ingress.md>를 참조하십시오.

시나리오 1: NCP가 수신을 처리하지만 기본 수신 컨트롤러가 아닙니다.

이 절차에 따라 NCP가 `nsx` 클래스 수신을 처리하도록 합니다.

- 1 각 마스터 노드에서 `nep.ini`의 `nsx_v3` 섹션을 편집합니다.
 - a `default_ingress_class_nsx`를 **False**로 설정합니다.
 - b `use_native_loadbalancer`를 기본값인 **True**로 둡니다.
- 2 각 마스터 노드에서 NCP를 다시 시작합니다. 이로 인해 마스터 페일오버가 발생할 수 있습니다.
- 3 NCP가 처리할 모든 수신에 `kubernetes.io/ingress.class: "nsx"` 주석을 지정합니다.

시나리오 2: NCP가 기본 수신 컨트롤러입니다.

다음 절차를 따릅니다.

- 1 `nep.ini`를 편집할 필요가 없지만 모든 수신에 주석이 지정되어 있는지 확인합니다.
- 2 NCP를 통해 처리할 수신에는 `kubernetes.io/ingress.class: "nsx"` 주석을 지정해야 합니다.

NCP는 `kubernetes.io/ingress.class` 주석이 지정되지 않은 수신도 처리하지만 여러 수신 컨트롤러가 있는 경우에는 항상 `kubernetes.io/ingress.class` 주석을 지정하고, 기본 수신 컨트롤러 동작에 의존하지 않는 것이 좋습니다.
- 3 타사 수신 컨트롤러를 통해 처리할 수신에는 해당 수신 컨트롤러에 필요한 값이 주석으로 지정되어야 합니다.

중요 목표가 NGINX를 기본 수신 컨트롤러로 지정하는 것이 아니라면, NGINX를 기본 수신 컨트롤러로 지정하는 `nginx`를 NGINX 수신 컨트롤러로 사용하면 안 됩니다.

시나리오 3: NCP는 주석과 관계없이 어떠한 수신도 처리하지 않습니다.

다음 절차를 따릅니다.

1 각 마스터 노드에서 `nep.ini`의 `nsx_v3` 섹션을 편집합니다.

a `use_native_loadbalancer`를 **False**로 설정합니다. 이제 `default_ingress_class_nsx`의 값은 부적절합니다.

2 각 마스터 노드에서 NCP를 다시 시작합니다. 이로 인해 마스터 페일오버가 발생할 수 있습니다.

NCP는 LoadBalancer 유형의 서비스도 처리하지 않습니다.

NSX Container Plug-in 관리

6

NSX Manager GUI 또는 CLI(명령줄 인터페이스)에서 NSX Container Plug-in을 관리할 수 있습니다.

참고 ESXi 6.5에서 실행 중인 컨테이너 호스트 VM을 vMotion을 통해 다른 ESXi 6.5 호스트로 마이그레이션하는 경우 해당 컨테이너 호스트에서 실행되는 컨테이너와 다른 컨테이너 호스트에서 실행되는 컨테이너 간의 연결이 끊깁니다. 이 문제는 컨테이너 호스트 vNIC의 연결을 끊은 후 다시 연결하여 해결할 수 있습니다. ESXi 6.5 업데이트 1 이상에서는 이 문제가 발생하지 않습니다.

Hyperbus의 경우 하이퍼바이저에 PVLAN 구성을 위한 VLAN ID 4094가 예약되며 이 ID는 변경할 수 없습니다. VLAN 충돌을 방지하려면 VLAN 논리적 스위치 또는 VTEP vmknic를 동일한 VLAN ID로 구성하지 마십시오.

본 장은 다음 항목을 포함합니다.

- [Kubernetes 리소스 NSXError에 저장된 오류 정보 표시](#)
- [CIF 연결 논리적 포트](#)
- [CLI 명령](#)
- [오류 코드](#)

Kubernetes 리소스 NSXError에 저장된 오류 정보 표시

NSX 백엔드 장애가 있는 각 Kubernetes 리소스 개체의 경우 오류 정보와 함께 하나의 NSXError 개체가 생성됩니다. 모든 클러스터 전체 오류에 대한 오류 개체도 있습니다.

이 기능은 기본적으로 사용되도록 설정되어 있지 않습니다. 이 기능을 사용하도록 설정하려면 NCP를 설치할 때 `ncp.ini`에서 `enable_nsx_err_crd`를 True로 설정해야 합니다.

참고 NSXError 개체를 생성, 업데이트 또는 삭제하지 않아야 합니다.

NSXError 개체를 표시하기 위한 명령;

- `kubectl get nsxerrors`

모든 NSXError 개체를 나열합니다.

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

특정 유형의 Kubernetes 개체(예: **services** 유형의 개체)와 관련된 **NSXError** 개체를 나열합니다.

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

NSXError 개체의 세부 정보를 표시합니다.

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

특정 서비스와 연결된 **NSXError**를 찾습니다.

NSXError 개체의 세부 정보를 표시하는 경우 규칙 섹션에 다음과 같은 중요 정보가 포함됩니다. 예를 들면 다음과 같습니다.

```
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'
```

이 예에서 네임스페이스는 **default**입니다. 서비스의 이름은 **svc-1**입니다. Kubernetes 리소스의 유형은 **services**입니다.

이 릴리스에서는 **NSXError** 개체에서 다음 오류가 지원됩니다.

- **NSX Edge** 제한으로 인해 자동 크기 조정이 추가 로드 밸런서를 할당하지 못했습니다.
- 로드 밸런서 가상 서버의 수가 제한을 초과합니다(자동 크기 조정이 사용되도록 설정되지 않음).
- 로드 밸런서 서버 풀의 수가 제한을 초과합니다.
- 로드 밸런서 서버 풀 멤버의 수가 로드 밸런서 제한 또는 **NSX Edge** 제한을 초과합니다.
- **LoadBalancer** 유형 서비스를 처리할 때 부동 IP 주소가 모두 사용됩니다.

CIF 연결 논리적 포트

CIF(컨테이너 인터페이스)는 스위치의 논리적 포트에 연결된 컨테이너의 네트워크 인터페이스입니다. 이러한 포트를 CIF 연결 논리적 포트라고 합니다.

NSX Manager GUI에서 CIF 연결 논리적 포트를 관리할 수 있습니다.

CIF 연결 논리적 포트 관리

네트워킹 > 스위칭 > 포트로 이동하여 CIF 연결 논리적 포트를 비롯한 모든 논리적 포트를 볼 수 있습니다. CIF 연결 논리적 포트의 연결 정보를 보려면 연결 링크를 클릭하십시오. 논리적 포트 링크를 클릭하여 4개의 탭, 즉 [개요], [모니터], [관리] 및 [관련]이 있는 창을 엽니다. **관련 > 논리적 포트**를 클릭하면 업링크 스위치의 관련 논리적 포트가 표시됩니다. 스위치 포트에 대한 자세한 내용은 "**NSX-T 관리 가이드**"를 참조하십시오.

네트워크 모니터링 도구

다음 도구는 CIF 연결 논리적 포트를 지원합니다. 이러한 도구에 대한 자세한 내용은 "NSX-T 관리 가이드"를 참조하십시오.

- Traceflow
- 포트 연결
- IPFIX
- 컨테이너에 연결되는 논리적 스위치 포트의 GRE 캡슐화를 사용한 원격 포트 미러링이 지원됩니다. 자세한 내용은 "NSX-T 관리 가이드"의 "포트 미러링 전환 프로파일 이해"를 참조하십시오. 하지만 CIF를 VIF 포트에 미러링하는 기능은 Manager UI를 통해 지원되지 않습니다.

CLI 명령

CLI 명령을 실행하려면 NSX Container Plug-in 컨테이너에 로그인하고 터미널을 연 후 `nsxcli` 명령을 실행합니다.

노드에서 다음 명령을 실행하여 CLI 프롬프트를 표시할 수도 있습니다.

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

표 6-1. NCP 컨테이너의 CLI 명령

유형	명령	참고
상태	<code>get ncp-master status</code>	Kubernetes와 PCF 모두에 해당
상태	<code>get ncp-nsx status</code>	Kubernetes와 PCF 모두에 해당
상태	<code>get ncp-watcher <watcher-name></code>	Kubernetes와 PCF 모두에 해당
상태	<code>get ncp-watchers</code>	Kubernetes와 PCF 모두에 해당
상태	<code>get ncp-k8s-api-server status</code>	Kubernetes에만 해당
상태	<code>check projects</code>	Kubernetes에만 해당
상태	<code>check project <project-name></code>	Kubernetes에만 해당
상태	<code>get ncp-bbs status</code>	PCF에만 해당
상태	<code>get ncp-capi status</code>	PCF에만 해당
상태	<code>get ncp-policy-server status</code>	PCF에만 해당
캐시	<code>get project-caches</code>	Kubernetes에만 해당
캐시	<code>get project-cache <project-name></code>	Kubernetes에만 해당
캐시	<code>get namespace-caches</code>	Kubernetes에만 해당
캐시	<code>get namespace-cache <namespace-name></code>	Kubernetes에만 해당

표 6-1. NCP 컨테이너의 CLI 명령 (계속)

유형	명령	참고
캐시	get pod-caches	Kubernetes에만 해당
캐시	get pod-cache <pod-name>	Kubernetes에만 해당
캐시	get ingress-caches	Kubernetes에만 해당
캐시	get ingress-cache <ingress-name>	Kubernetes에만 해당
캐시	get ingress-controllers	Kubernetes에만 해당
캐시	get ingress-controller <ingress-controller-name>	Kubernetes에만 해당
캐시	get network-policy-caches	Kubernetes에만 해당
캐시	get network-policy-cache <pod-name>	Kubernetes에만 해당
캐시	get asg-caches	PCF에만 해당
캐시	get asg-cache <asg-ID>	PCF에만 해당
캐시	get org-caches	PCF에만 해당
캐시	get org-cache <org-ID>	PCF에만 해당
캐시	get space-caches	PCF에만 해당
캐시	get space-cache <space-ID>	PCF에만 해당
캐시	get app-caches	PCF에만 해당
캐시	get app-cache <app-ID>	PCF에만 해당
캐시	get instance-caches <app-ID>	PCF에만 해당
캐시	get instance-cache <app-ID> <instance-ID>	PCF에만 해당
캐시	get policy-caches	PCF에만 해당
지원	get ncp-log file <filename>	Kubernetes와 PCF 모두에 해당
지원	get ncp-log-level	Kubernetes와 PCF 모두에 해당
지원	set ncp-log-level <log-level>	Kubernetes와 PCF 모두에 해당
지원	get support-bundle file <filename>	Kubernetes에만 해당
지원	get node-agent-log file <filename>	Kubernetes에만 해당
지원	get node-agent-log file <filename> <node-name>	Kubernetes에만 해당

표 6-2. NSX 노드 에이전트 컨테이너의 CLI 명령

유형	명령
상태	get node-agent-hyperbus status
캐시	get container-cache <container-name>
캐시	get container-caches

표 6-3. NSX Kube Proxy 컨테이너의 CLI 명령

유형	명령
상태	get ncp-k8s-api-server status
상태	get kube-proxy-watcher <watcher-name>
상태	get kube-proxy-watchers
상태	dump ovs-flows

NCP 컨테이너의 상태 명령

■ NCP 마스터의 상태 표시

```
get ncp-master status
```

예:

```
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
```

■ NCP와 NSX Manager 간의 연결 상태 표시

```
get ncp-nsx status
```

예:

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

■ 수신, 네임스페이스, 포트 및 서비스에 대한 감시자 상태 표시

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

예:

```
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

```

kubecode> 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

```

■ NCP와 Kubernetes API 서버 간의 연결 상태 표시

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

예:

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

■ 모든 프로젝트 또는 특정 프로젝트 확인

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

예:

```
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
```

■ NCP와 PCF BBS 간의 연결 상태 확인

```
get ncp-bbs status
```

예:

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

■ NCP와 PCF CAPI 간의 연결 상태 확인

```
get ncp-capi status
```

예:

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

■ NCP와 PCF 정책 서버 간의 연결 상태 확인

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

예:

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

NCP 컨테이너의 캐시 명령

- 프로젝트 또는 네임스페이스에 대한 내부 캐시 가져오기

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

예:

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

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

testns:
  ext_pool_id: 346a0f36-7b5a-4ecc-ad32-338dc92316f
  labels:
    ns: myns
    project: myproject
  logical-router: 4dc8f8a9-69b4-4ff7-8fb7-d2625dc77efa
  logical-switch:
    id: 6111a99a-6e06-4faa-a131-649f10f7c815
    ip_pool_id: 51ca058d-c3dc-41fd-8f2d-e69006ab1b3d
    subnet: 50.0.2.0/24
    subnet_id: 34f79811-bd29-4048-a67d-67ceac97eb98
  project_nsgroup: 9606afee-6348-4780-9dbe-91abfd23e475
  snat_ip: 4.4.0.3

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

kubenode> get namespace-caches
default:
```

```

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

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

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

```

```

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

```

■ 포트에 대한 내부 캐시 가져오기

```

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

```

예:

```

kubenode> get pod-caches
nsx.default.nginx-rc-ug2lv:
  cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
  gateway_ip: 10.0.0.1
  host_vif: d6210773-5c07-4817-98db-451bd1f01937
  id: 1c8b5c52-3795-11e8-ab42-005056b198fb
  ingress_controller: False
  ip: 10.0.0.2/24
  labels:

```

```

    app: nginx
    mac: 02:50:56:00:08:00
    port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
    vlan: 1

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

```

```

kubenode> get pod-cache nsx.default.nginx-rc-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

```

■ 모든 수신 캐시 또는 특정 수신 캐시 가져오기

```

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

```

예:

```

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

kubecall> 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

```

- 모든 수신 컨트롤러 또는 특정 수신 컨트롤러에 대한 정보 가져오기(사용되지 않도록 설정된 컨트롤러 포함)

```

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

```

예:

```

kubecall> 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

kubenset> 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

```

■ 네트워크 정책 캐시 또는 특정 네트워크 정책 캐시 가져오기

```

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

```

예):

```

kubenset> 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
```

■ 모든 ASG 캐시 또는 특정 ASG 캐시 가져오기

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

예:

```
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:
```

■ 모든 조직 캐시 또는 특정 조직 캐시 가져오기

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

예:

```
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
```

■ 모든 공간 캐시 또는 특정 공간 캐시 가져오기

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

예:

```
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

```

- 모든 애플리케이션 캐시 또는 특정 애플리케이션 캐시 가져오기

```

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

```

예:

```

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-bbfb-a4badf21eb09
space_id: 7870d134-7997-4373-b665-b6a910413c47

```

- 애플리케이션의 모든 인스턴스 캐시 또는 특정 인스턴스 캐시 가져오기

```

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

```

예:

```

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

```

- 모든 정책 캐시 가져오기

```

get policy-caches

```

예:

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

f582ec4d-3a13-440a-afbd-97b7bfae21d1:
  nsg_id: d24b9f77-e2e0-4fba-b258-893223683aa6
  rules:
    8272:
      dst_app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
      ports: 8382
      protocol: tcp
      src_app_id: f582ec4d-3a13-440a-afbd-97b7bfae21d1
```

NCP 컨테이너의 지원 명령

■ 파일 저장소에 NCP 지원 번들 저장

지원 번들은 레이블이 **tier:nsx-networking**인 포드의 모든 컨테이너에 대한 로그 파일로 구성됩니다. 번들 파일은 **tgz** 형식으로 되어 있으며 CLI 기본 파일 저장소 디렉토리 **/var/vmware/nsx/file-store**에 저장됩니다. CLI **file-store** 명령을 사용하여 번들 파일을 원격 사이트에 복사할 수 있습니다.

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

예:

```
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
```

■ 파일 저장소에 NCP 로그 저장

로그 파일은 **tgz** 형식으로 CLI 기본 파일 저장소 디렉토리 **/var/vmware/nsx/file-store**에 저장됩니다. CLI **file-store** 명령을 사용하여 번들 파일을 원격 사이트에 복사할 수 있습니다.

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

예:

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

■ 파일 저장소에 노드 에이전트 로그 저장

하나의 노드 또는 모든 노드의 노드 에이전트 로그를 저장합니다. 로그는 **tgz** 형식으로 CLI 기본 파일 저장소 디렉토리 **/var/vmware/nsx/file-store**에 저장됩니다. CLI **file-store** 명령을 사용하여 번들 파일을 원격 사이트에 복사할 수 있습니다.

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

예:

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

■ 로그 수준 가져오기 및 설정

사용 가능한 로그 수준은 NOTSET, DEBUG, INFO, WARNING, ERROR 및 CRITICAL입니다.

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

예:

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

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

NSX 노드 에이전트 컨테이너의 상태 명령

■ 이 노드의 노드 에이전트와 HyperBus 사이의 연결 상태 표시

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

예:

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

NSX 노드 에이전트 컨테이너의 캐시 명령

■ NSX 노드 에이전트 컨테이너의 내부 캐시 가져오기

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

예:

```
kubenode> get container-caches
cif104:
  ip: 192.168.0.14/32
  mac: 50:01:01:01:01:14
```

```
gateway_ip: 169.254.1.254/16
vlan_id: 104
```

```
kubenode> get container-cache cif104
ip: 192.168.0.14/32
mac: 50:01:01:01:01:14
gateway_ip: 169.254.1.254/16
vlan_id: 104
```

NSX Kube Proxy 컨테이너의 상태 명령

- Kube Proxy와 Kubernetes API 서버 간의 연결 상태 표시

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

예:

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

- Kube Proxy 감시자 상태 표시

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

예:

```
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

```

■ 노드에서 OVS 흐름 덤프

```
dump ovs-flows
```

예:

```

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

```

오류 코드

이 섹션에는 다양한 구성 요소에 의해 생성된 오류 코드가 나열되어 있습니다.

NCP 오류 코드

오류 코드	설명
NCP00001	잘못된 구성
NCP00002	초기화 실패
NCP00003	잘못된 상태
NCP00004	잘못된 어댑터
NCP00005	인증서를 찾을 수 없음
NCP00006	토큰을 찾을 수 없음
NCP00007	잘못된 NSX 구성
NCP00008	잘못된 NSX 태그

오류 코드	설명
NCP00009	NSX 연결 실패
NCP00010	노드 태그를 찾을 수 없음
NCP00011	잘못된 노드 논리적 스위치 포트
NCP00012	상위 VIF 업데이트 실패
NCP00013	VLAN이 모두 사용됨
NCP00014	VLAN 릴리스 실패
NCP00015	IP 풀이 모두 사용됨
NCP00016	IP 릴리스 실패
NCP00017	IP 블록이 모두 사용됨
NCP00018	IP 서브넷 생성 실패
NCP00019	IP 서브넷 삭제 실패
NCP00020	IP 풀 생성 실패
NCP00021	IP 풀 삭제 실패
NCP00022	논리적 라우터 생성 실패
NCP00023	논리적 라우터 업데이트 실패
NCP00024	논리적 라우터 삭제 실패
NCP00025	논리적 스위치 생성 실패
오류 코드	설명
NCP00026	논리적 스위치 업데이트 실패
NCP00027	논리적 스위치 삭제 실패
NCP00028	논리적 라우터 포트 생성 실패
NCP00029	논리적 라우터 포트 삭제 실패
NCP00030	논리적 스위치 포트 생성 실패
NCP00031	논리적 스위치 포트 업데이트 실패
NCP00032	논리적 스위치 포트 삭제 실패
NCP00033	네트워크 정책을 찾을 수 없음
NCP00034	방화벽 생성 실패
NCP00035	방화벽 읽기 실패

오류 코드	설명
NCP00036	방화벽 업데이트 실패
NCP00037	방화벽 삭제 실패
NCP00038	여러 개의 방화벽을 찾음
NCP00039	NSGroup 생성 실패
NCP00040	NSGroup 삭제 실패
NCP00041	IP 집합 생성 실패
NCP00042	IP 집합 업데이트 실패
NCP00043	IP 집합 삭제 실패
NCP00044	SNAT 규칙 생성 실패
NCP00045	SNAT 규칙 삭제 실패
NCP00046	어댑터 API 연결 실패
NCP00047	어댑터 감시자 예외
NCP00048	로드 밸런서 서비스 삭제 실패
NCP00049	로드 밸런서 가상 서버 생성 실패
NCP00050	로드 밸런서 가상 서버 업데이트 실패

오류 코드	설명
NCP00051	로드 밸런서 가상 서버 삭제 실패
NCP00052	로드 밸런서 풀 생성 실패
NCP00053	로드 밸런서 풀 업데이트 실패
NCP00054	로드 밸런서 풀 삭제 실패
NCP00055	로드 밸런서 규칙 생성 실패
NCP00056	로드 밸런서 규칙 업데이트 실패
NCP00057	로드 밸런서 규칙 삭제 실패
NCP00058	로드 밸런서 풀 IP 릴리스 실패
NCP00059	로드 밸런서 가상 서버 및 서비스 연결을 찾을 수 없음
NCP00060	NSGroup 업데이트 실패
NCP00061	방화벽 규칙 가져오기 실패
NCP00062	NSGroup 조건 없음

오류 코드	설명
NCP00063	노드 VM을 찾을 수 없음
NCP00064	노드 VIF를 찾을 수 없음
NCP00065	인증서 가져오기 실패
NCP00066	인증서 가져오기 취소 실패
NCP00067	SSL 바인딩 업데이트 실패
NCP00068	SSL 프로파일을 찾을 수 없음
NCP00069	IP 풀을 찾을 수 없음
NCP00070	T0 Edge 클러스터를 찾을 수 없음
NCP00071	IP 풀 업데이트 실패
NCP00072	디스패처 실패
NCP00073	NAT 규칙 삭제 실패
NCP00074	논리적 라우터 포트 가져오기 실패
NCP00075	NSX 구성 검증 실패

오류 코드	설명
NCP00076	SNAT 규칙 업데이트 실패
NCP00077	SNAT 규칙이 겹침
NCP00078	로드 밸런서 끝점 추가 실패
NCP00079	로드 밸런서 끝점 업데이트 실패
NCP00080	로드 밸런서 규칙 풀 생성 실패
NCP00081	로드 밸런서 가상 서버를 찾을 수 없음
NCP00082	IP 집합 읽기 실패
NCP00083	SNAT 풀 가져오기 실패
NCP00084	로드 밸런서 서비스 생성 실패
NCP00085	로드 밸런서 서비스 업데이트 실패
NCP00086	논리적 라우터 포트 업데이트 실패
NCP00087	로드 밸런서 초기화 실패
NCP00088	IP 풀이 고유하지 않음
NCP00089	계층 7 로드 밸런서 캐시 동기화 오류

오류 코드	설명
NCP00090	로드 밸런서 풀이 존재하지 않음
NCP00091	로드 밸런서 규칙 캐시 초기화 오류
NCP00092	SNAT 프로세스 실패
NCP00093	로드 밸런서 기본 인증서 오류
NCP00094	로드 밸런서 끝점 삭제 실패
NCP00095	프로젝트를 찾을 수 없음
NCP00096	풀 액세스가 거부됨
NCP00097	로드 밸런서 서비스를 가져올 수 없음
NCP00098	로드 밸런서 서비스를 생성할 수 없음
NCP00099	로드 밸런서 풀 캐시 동기화 오류

NSX 노드 에이전트 오류 코드

오류 코드	설명
NCP01001	OVS 업링크를 찾을 수 없음
NCP01002	호스트 MAC을 찾을 수 없음
NCP01003	OVS 포트 생성 실패
NCP01004	포트 구성이 없음
NCP01005	포트 구성 실패
NCP01006	포트 구성 취소 실패
NCP01007	CNI 소켓을 찾을 수 없음
NCP01008	CNI 연결 실패
NCP01009	CNI 버전 불일치
NCP01010	CNI 메시지 수신 실패
NCP01011	CNI 메시지 전송 실패
NCP01012	Hyperbus 연결 실패
NCP01013	Hyperbus 버전 불일치
NCP01014	Hyperbus 메시지 수신 실패
NCP01015	Hyperbus 메시지 전송 실패

오류 코드	설명
NCP01016	GARP 전송 실패
NCP01017	인터페이스 구성 실패

nsx-kube-proxy 오류 코드

오류 코드	설명
NCP02001	잘못된 프록시 게이트웨이 포트
NCP02002	프록시 명령 실패
NCP02003	프록시 검증 실패

CLI 오류 코드

오류 코드	설명
NCP03001	CLI 시작 실패
NCP03002	CLI 소켓 생성 실패
NCP03003	CLI 소켓 예외
NCP03004	잘못된 CLI 클라이언트 요청
NCP03005	CLI 서버 전송 실패
NCP03006	CLI 서버 수신 실패
NCP03007	CLI 명령 실행 실패

Kubernetes 오류 코드

오류 코드	설명
NCP05001	Kubernetes 연결 실패
NCP05002	잘못된 Kubernetes 구성
NCP05003	Kubernetes 요청 실패
NCP05004	Kubernetes 키를 찾을 수 없음
NCP05005	Kubernetes 유형을 찾을 수 없음
NCP05006	Kubernetes 감시자 예외
NCP05007	잘못된 Kubernetes 리소스 길이
NCP05008	잘못된 Kubernetes 리소스 유형

오류 코드	설명
NCP05009	Kubernetes 리소스 처리 실패
NCP05010	Kubernetes 서비스 처리 실패
NCP05011	Kubernetes 끝점 처리 실패
NCP05012	Kubernetes 수신 처리 실패
NCP05013	Kubernetes 네트워크 정책 처리 실패
NCP05014	Kubernetes 노드 처리 실패
NCP05015	Kubernetes 네임스페이스 처리 실패
NCP05016	Kubernetes 포트 처리 실패
NCP05017	Kubernetes 암호 처리 실패
NCP05018	Kubernetes 기본 백엔드 실패
NCP05019	지원되지 않는 Kubernetes 일치 식
NCP05020	Kubernetes 상태 업데이트 실패
NCP05021	Kubernetes 주석 업데이트 실패
NCP05022	Kubernetes 네임스페이스 캐시를 찾을 수 없음
NCP05023	Kubernetes 암호를 찾을 수 없음
NCP05024	Kubernetes 기본 백엔드가 사용 중임
NCP05025	Kubernetes LoadBalancer 서비스 처리 실패

Pivotal Cloud Foundry 오류 코드

오류 코드	설명
NCP06001	PCF BBS 연결 실패
NCP06002	PCF CAPI 연결 실패
NCP06006	PCF 캐시를 찾을 수 없음
NCP06007	알 수 없는 PCF 도메인
NCP06020	PCF 정책 서버 연결 실패
NCP06021	PCF 정책 처리 실패
NCP06030	PCF 이벤트 처리 실패
NCP06031	예기치 않은 PCF 이벤트 유형
NCP06032	예기치 않은 PCF 이벤트 인스턴스

오류 코드	설명
NCP06033	PCF 작업 삭제 실패
NCP06034	PCF 파일 액세스 실패