

OpenShift용 NSX Container Plug-in - 설치 및 관리 가이드

VMware NSX Container Plug-in 2.3, 2.3.1, 2.3.2

VMware NSX-T Data Center 2.3

VMware NSX-T Data Center 2.3.1



vmware®

다음 VMware 웹 사이트에서 최신 기술 문서를 확인할 수 있습니다.

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

VMware 웹 사이트에서는 최신 제품 업데이트도 제공합니다.

본 문서에 대한 의견이 있으시면 다음 주소로 피드백을 보내주십시오.

docfeedback@vmware.com

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

OpenShift용 NSX-T Container Plug-in - 설치 및 관리 가이드 4

- 1 NSX-T Container Plug-in 개요 5**
 - 호환성 요구 사항 6
 - 설치 개요 6
 - NCP 업그레이드 7
- 2 NSX-T 리소스 설정 8**
 - NSX-T 리소스 구성 8
 - Tier-0 논리적 라우터 생성 및 구성 11
- 3 OpenShift 환경에서 NCP 설치 12**
 - OpenShift VM 배포 12
 - Ansible Hosts 파일 준비 12
 - 단일 플레이북을 사용하여 NCP와 OpenShift 설치 15
 - CNI 플러그인, OVS 및 NCP Docker 이미지 설치 15
 - OpenShift Container Platform 설치 17
 - NCP 및 NSX 노드 에이전트 실행 17
 - 설정 메모 19
- 4 베어메탈 환경에서 NCP 설치 23**
 - NSX-T Data Center CNI 플러그인 설치 23
 - OpenShift 노드에 대한 NSX-T Data Center 네트워킹 구성 23
 - NSX 노드 에이전트 설치 24
 - nsx-node-agent-ds.yml의 ncp.ini에 대한 Configmap 25
 - NSX-T Container Plug-in 설치 28
 - ncp-rc.yml의 ncp.ini에 대한 Configmap 30
- 5 로드 밸런싱 36**
 - 로드 밸런싱 구성 36
- 6 NSX-T Container Plug-in 관리 42**
 - NSX Manager GUI에서 IP 블록 관리 42
 - NSX Manager GUI에서 IP 블록 서브넷 보기 43
 - CIF 연결 논리적 포트 43
 - CLI 명령 44
 - 오류 코드 55

OpenShift용 NSX-T Container Plug-in - 설치 및 관리 가이드

이 가이드에서는 NCP(NSX-T Container Plug-in)를 설치 및 관리하여 NSX-T Data Center 및 OpenShift 간에 통합을 제공하는 방법을 설명합니다.

대상 사용자

이 가이드는 시스템 및 네트워크 관리자를 대상으로 작성되었습니다. 이 가이드에서는 사용자가 NSX-T Data Center 및 OpenShift의 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-T Container Plug-in 개요

1

NCP(NSX-T Container Plug-in)는 NSX-T Data Center와 컨테이너 조정기(예: Kubernetes) 간에 통합을 제공하고, NSX-T Data Center와 컨테이너 기반 PaaS(Platform As a Service) 소프트웨어 제품(예: OpenShift) 간에 통합을 제공합니다. 이 가이드에서는 OpenShift로 NCP를 설정하는 방법에 대해 설명합니다.

NCP의 주요 구성 요소는 컨테이너에서 실행되며 NSX Manager 및 OpenShift 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 OpenShift 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.

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

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

- NSX-T 분산 방화벽으로 OpenShift 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.
- NSX-T 계층 7 로드 밸런서를 사용하여 OpenShift 라우팅을 구현합니다.
 - HTLS Edge 종료로 HTTP 라우팅 및 HTTPS 라우팅 지원.
 - 대체 백엔드와 와일드카드 하위 도메인으로 라우팅 지원.
- 네임스페이스, 포트 이름 및 포트의 레이블에 대한 NSX-T 논리적 스위치 포트에 태그를 생성하고 관리자가 태그를 기반으로 NSX-T Data Center 보안 그룹 및 정책을 정의하도록 허용합니다.

이 릴리스에서 NCP는 단일 OpenShift 클러스터를 지원합니다.

본 장은 다음 항목을 포함합니다.

- [호환성 요구 사항](#)
- [설치 개요](#)
- [NCP 업그레이드](#)

호환성 요구 사항

NSX-T Container Plug-in(NCP)의 호환성 요구 사항은 다음과 같습니다.

소프트웨어 제품	버전
NSX-T Data Center	2.2, 2.3
컨테이너 호스트 VM용 하이퍼바이저	<ul style="list-style-type: none"> ■ 지원되는 vSphere 버전 ■ RHEL KVM 7.4, 7.5
컨테이너 호스트 운영 체제	RHEL 7.4, 7.5
서비스 형태의 플랫폼	OpenShift 3.9, 3.10
컨테이너 호스트 Open vSwitch	2.9.1(NSX-T 2.3 및 2.2와 함께 패키지로 제공)

설치 개요

NCP 설치 및 구성에는 다음 단계가 포함됩니다. 단계를 성공적으로 수행하려면 NSX-T Data Center 및 OpenShift 설치 및 관리에 익숙해야 합니다.

- 1 NSX-T Data Center를 설치합니다.
- 2 오버레이 전송 영역을 생성합니다.
- 3 오버레이 논리적 스위치를 생성하고 노드를 스위치에 연결합니다.
- 4 Tier-0 논리적 라우터를 생성합니다.
- 5 포트에 대한 IP 블록을 생성합니다.
- 6 SNAT(소스 네트워크 주소 변환)를 위한 IP 풀을 생성합니다.
- 7 OpenShift VM을 배포합니다.
- 8 Ansible 호스트 파일을 준비합니다.
- 9 (옵션 1) 단일 플레이북을 사용하여 NCP와 OpenShift를 설치합니다.
(옵션 2) CNI 플러그인, OVS(Open vSwitch) 및 NCP Docker 이미지를 설치합니다. 그런 다음 OpenShift Container Platform을 설치합니다.
- 10 NCP 및 NSX 노드 에이전트를 실행합니다.

제공된 플레이북을 사용하여 NCP를 설치하는 경우에는 2~6단계가 필요하지 않습니다. [단일 플레이북을 사용하여 NCP와 OpenShift 설치](#) 및 [CNI 플러그인, OVS 및 NCP Docker 이미지 설치](#)를 참조하십시오.

NCP 업그레이드

NCP를 2.3.0으로 업그레이드하려면 다음 단계를 수행하십시오.

- 1 CNI RPM 패키지, NSX 노드 에이전트 DaemonSet 및 NCP ReplicationController를 업그레이드합니다.
- 2 (선택 사항) NSX-T Data Center를 2.3으로 업그레이드 합니다.

NCP 2.3.0은 NSX-T 2.2를 지원하지만 NSX-T Data Center 2.3으로 업그레이드할 수도 있습니다.

NSX-T 리소스 설정

OpenShift 노드에 대한 네트워킹을 제공하려면 NSX-T Data Center 리소스를 생성해야 합니다. NSX Manager GUI를 사용하여 이러한 리소스를 수동으로 구성하거나, Aniable 플레이북을 사용하여 프로세스를 자동화할 수 있습니다.

이 섹션에서는 NSX-T 리소스를 수동으로 생성하는 방법을 설명합니다. 이 프로세스를 자동화하려면 [CNI 플러그인](#), [OVS](#) 및 [NCP Docker 이미지 설치](#) 항목을 참조하십시오.

본 장은 다음 항목을 포함합니다.

- [NSX-T 리소스 구성](#)
- [Tier-0 논리적 라우터 생성 및 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.

구성 파일 ncp.ini의 UUID나 이름을 사용하여 NSX-T Data Center 리소스를 구성합니다.

오버레이 전송 영역

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

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

Tier-0 논리적 라우팅

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

ncp.ini의 [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 **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 **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 네임스페이스에만 사용되는 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.

SNAT용 IP 풀

IP 풀은 SNAT 규칙을 통해 포드 IP를 변환하고 SNAT/DNAT 규칙을 통해 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 포드 서브넷에 대해 동일한 서브넷 접두사가 사용됩니다. 다른 서브넷 크기를 사용하려면 ncp.ini의 [nsx_v3] 섹션에서 external_subnet_prefix 옵션을 업데이트합니다.

NSX Manager에 로그인하고 **인벤토리 > 그룹 > IP 풀**로 이동하여 풀을 생성하거나 기존 풀을 찾습니다.

ncp.ini의 [nsx_v3] 섹션에 external_ip_pools 옵션을 설정하여 SNAT에 대한 IP 풀을 지정합니다.

서비스에 주석을 추가하여 특정 서비스에 대한 SNAT를 구성할 수도 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
...
```

NCP는 이 서비스에 대한 SNAT 규칙을 구성합니다. 규칙의 소스 IP는 백엔드 포트 집합입니다. 대상 IP는 지정된 외부 IP 풀에서 할당된 SNAT IP입니다. 다음에 유의하십시오.

- 서비스를 구성하기 전에 ncp/snat_pool에서 지정한 IP 풀이 NSX-T Data Center에 이미 있어야 합니다. NCP 2.3.1부터는 IP 풀에 태그 {"ncp/owner": cluster:<cluster>}가 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.

다음 태그를 IP 풀에 추가하여 SNAT IP 풀의 IP를 할당할 수 있는 네임스페이스를 지정할 수 있습니다.

- 범위: ncp/owner, 태그: ns:<namespace_UUID>

다음 명령 중 하나를 사용하여 네임스페이스 UUID를 가져올 수 있습니다.

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

다음에 유의하십시오.

- 각 태그는 하나의 UUID를 지정해야 합니다. 동일한 풀에 대해 여러 개의 태그를 생성할 수 있습니다.
- 이전 태그를 기반으로 일부 네임스페이스에 IP를 할당한 후 태그를 변경했다면 이러한 IP는 서비스의 SNAT 구성이 변경되거나 NCP가 다시 시작될 때까지 회수되지 않습니다.
- 네임스페이스 소유자 태그는 선택 사항입니다. 이 태그가 없으면 모든 네임스페이스가 SNAT IP 풀에서 할당된 IP를 가질 수 있습니다.

(선택 사항) 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.ini의 [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.

Tier-0 논리적 라우터 생성 및 구성

Tier-0 논리적 라우터는 Kubernetes 노드를 외부 네트워크에 연결합니다.

절차

- 1 브라우저에서 NSX Manager(<https://nsx-manager-ip-address>)에 로그인합니다.
- 2 **네트워킹 > 라우팅 > 라우터**로 이동한 후 **추가 > Tier-0 라우터**를 클릭합니다.
- 3 이름과 설명(선택 사항)을 입력합니다.
- 4 드롭다운 메뉴에서 이 Tier-0 논리적 라우터를 지원할 기존 Edge 클러스터를 선택합니다.
- 5고가용성 모드를 선택합니다.
활성-대기를 선택합니다.
- 6 **저장**을 클릭합니다.
새 논리적 라우터가 링크로 표시됩니다.
- 7 논리적 라우터 링크를 클릭합니다.
- 8 **라우팅 > 경로 재배포**를 클릭합니다.
- 9 **추가**를 클릭하여 새 재배포 기준을 추가합니다.
소스의 경우 라우팅된(비 NAT) 토폴로지에서 **NSX 정적**을 선택합니다. NAT 토폴로지에서는 **Tier-0 NAT**를 선택합니다.
- 10 **저장**을 클릭합니다.
- 11 새로 생성된 라우터를 클릭합니다.
- 12 **구성 > 라우터 포트**를 클릭합니다.
- 13 **추가**를 클릭하여 업링크 포트를 추가합니다.
- 14 전송 노드를 선택합니다.
- 15 이전에 작성된 논리적 스위치를 선택합니다.
- 16 외부 네트워크의 IP 주소를 지정합니다.
- 17 **저장**을 클릭합니다.
새 논리적 라우터가 링크로 표시됩니다.

OpenShift 환경에서 NCP 설치

이 장에서는 NCP(NSX-T Container Plug-in) 및 OpenShift를 설치 및 구성하는 방법을 설명합니다.

본 장은 다음 항목을 포함합니다.

- [OpenShift VM 배포](#)
- [Ansible Hosts 파일 준비](#)
- 단일 플레이북을 사용하여 NCP와 OpenShift 설치
- CNI 플러그인, OVS 및 NCP Docker 이미지 설치
- OpenShift Container Platform 설치
- NCP 및 NSX 노드 에이전트 실행
- 설정 메모

OpenShift VM 배포

NSX-T Container Plug-in을 설치하려면 먼저 OpenShift가 설치되어 있어야 합니다. 하나 이상의 마스터를 배포해야 합니다.

자세한 내용은 <https://docs.openshift.com> 항목을 참조하십시오.

다음에 수행할 작업

Ansible 호스트 파일을 준비합니다. [Ansible Hosts 파일 준비](#)의 내용을 참조하십시오.

Ansible Hosts 파일 준비

Ansible hosts 파일은 OpenShift 클러스터의 노드를 정의합니다.

절차

- 1 <https://github.com/vmware/nsx-integration-for-openshift>에서 NCP GitHub 저장소를 복제합니다. Hosts 파일은 openshift-ansible-nsx 디렉토리에 있습니다. openshift ansible nsx 디렉토리에 hosts 파일을 유지해야 합니다. 일부 플레이북은 여기가 hosts 파일에 대한 경로라고 가정합니다.

- 2 [masters] 및 [nodes] 섹션에서 OpenShift VM의 호스트 이름과 IP 주소를 지정합니다. 예를 들면 다음과 같습니다.

```
[masters]
admin.rhel.osmaster ansible_ssh_host=101.101.101.4

[single_master]
admin.rhel.osmaster ansible_ssh_host=101.101.101.4

[nodes]
admin.rhel.osmaster ansible_ssh_host=101.101.101.4 openshift_ip=101.101.101.4 openshift_schedulable=true
openshift_hostname=admin.rhel.osmaster
admin.rhel.osnode ansible_ssh_host=101.101.101.5 openshift_ip=101.101.101.5
openshift_hostname=admin.rhel.osnode

[etcd]

[OSEv3:children]
masters
nodes
etcd
```

openshift_ip에서는 사용할 인터페이스가 기본 인터페이스가 아닌 경우 클러스터 내부 IP를 식별하고 설정해야 합니다. single_master 변수는 마스터 노드의 ncp 관련 역할이 특정 작업(예: NSX-T Data Center 관리부 리소스 구성)을 한 번만 수행하는 데 사용됩니다.

- 3 Ansible 역할이 실행되는 노드(일반적으로 마스터 노드)의 암호없이 모든 노드에 액세스할 수 있도록 SSH 액세스를 설정합니다.

```
ssh-keygen
ssh-copy-id -i ~/.ssh/id_rsa.pub root@admin.rhel.osnode
```

- 4 [OSEv3:vars] 섹션을 업데이트합니다. 모든 매개 변수에 대한 세부 정보는 고급 설치를 위한 OpenShift Container Platform 설명서에서 확인할 수 있습니다 (<https://docs.openshift.com>에서 "고급 설치" 검색). 예를 들면 다음과 같습니다.

```
# Set the default route fqdn
openshift_master_default_subdomain=apps.corp.local

os_sdn_network_plugin_name=cni
openshift_use_openshift_sdn=false
openshift_node_sdn_mtu=1500

# If ansible_ssh_user is not root, ansible_become must be set to true
ansible_become=true

openshift_master_default_subdomain
    This is the default subdomain used in the OpenShift routes for External LB

os_sdn_network_plugin_name
    Set to 'cni' for the NSX Integration
```

```

openshift_use_openshift_sdn
    Set to false to disable the built-in OpenShift SDN solution

openshift_hosted_manage_router
    Set to false to disable creation of router during installation. The router has to be manually started
    after NCP and nsx-node-agent are running.

openshift_hosted_manage_registry
    Set to false to disable creation of registry during installation. The registry has to be manually
    started after NCP and nsx-node-agent are running.

deployment_type
    Set to openshift-enterprise

openshift_hosted_manage_registry
    Set to false to disable auto creation of registry

openshift_hosted_manage_router
    Set to false to disable auto creation of router

openshift_enable_service_catalog
    Set to false to disable service_catalog

(For OpenShift 3.9 only) skip_sanity_checks
    Set to true

(For OpenShift 3.9 only) openshift_web_console_install
    Set to false
    
```

5 모든 호스트에 연결할 수 있는지 확인합니다.

```

ansible OSEv3 -i /PATH/TO/HOSTS/hosts -m ping
    
```

결과는 다음과 같아야 합니다. 그렇지 않은 경우 연결 문제를 해결합니다.

```

openshift-node1 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
openshift-master | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
    
```

다음에 수행할 작업

CNI 플러그인과 OVS를 설치합니다. [CNI 플러그인, OVS 및 NCP Docker 이미지 설치](#)의 내용을 참조하십시오.

단일 플레이북을 사용하여 NCP와 OpenShift 설치

단일 플레이북을 사용하여 NCP와 OpenShift를 설치하거나 별도의 단계로 설치를 수행할 수 있습니다.

단일 Ansible 플레이북 `install.yaml`은 다음 작업을 수행합니다.

- NCP 준비
- OpenShift 설치
- NCP 설치

또는 다음 두 섹션의 지침을 사용하여 NCP와 OpenShift를 설치할 수 있습니다. [CNI 플러그인, OVS 및 NCP Docker 이미지 설치](#) 및 [OpenShift Container Platform 설치](#).

`install.yaml` 플레이북을 실행하기 전에 `ncp_prep` 및 `ncp` 플레이북 역할에 대한 필수 매개 변수 및 선택적 매개 변수를 설정합니다. 매개 변수는 [CNI 플러그인, OVS 및 NCP Docker 이미지 설치](#)에 설명되어 있습니다.

다음 명령은 플레이북을 실행합니다.

```
ansible-playbook -i /PATH/TO/HOSTS/hosts install.yaml
```

CNI 플러그인, OVS 및 NCP Docker 이미지 설치

CNI(컨테이너 네트워크 인터페이스) 플러그인, OVS(Open vSwitch) 및 NCP Docker 이미지는 OpenShift 노드에 설치해야 합니다. 설치하는 Ansible 플레이북을 실행하여 수행됩니다.

참고 단일 플레이북을 사용하여 NCP와 OpenShift를 설치하는 경우에는 이 단계가 필요하지 않습니다. [단일 플레이북을 사용하여 NCP와 OpenShift 설치](#)의 내용을 참조하십시오.

이 플레이북에는 노드에 대해 NSX-T 리소스를 구성하기 위한 지침이 포함되어 있습니다. [장2NSX-T 리소스 설정](#)에 설명된 대로 NSX-T Data Center 리소스를 수동으로 구성할 수도 있습니다.

`perform_nsx_config` 매개 변수는 플레이북이 실행될 때 리소스를 구성할지 여부를 나타냅니다.

절차

- 1 `roles/ncp_prep/default/main.yaml` 및 `roles/nsx_config/default/main.yaml`에서 매개 변수 값을 업데이트합니다. 여기에는 CNI 플러그인 RPM, OVS 및 해당 커널 모듈 RPM을 다운로드할 수 있는 URL이 포함됩니다. 또한 `uplink_port`는 노드 VM에 있는 업링크 포트 VNIC의 이름입니다. 나머지 변수는 NSX-T Data Center 관리부 구성과 관련됩니다.

지정해야 하는 매개 변수:

- `perform_nsx_config`: 리소스 구성을 수행할지 여부입니다. 구성을 수동으로 수행하고 `nsx_config` 스크립트를 실행하지 않으려면 `false`로 설정합니다.
- `nsx_manager_ip`: NSX Manager의 IP입니다.
- `nsx_edge_cluster_name`: Tier-0 라우터에서 사용할 Edge 클러스터의 이름입니다.

- `nsx_transport_zone_name`: 오버레이 전송 영역의 이름입니다.
- `os_node_name_list`: 노드 이름의 쉼표로 구분된 목록입니다.
예를 들어 `node1,node2,node3`입니다.
- `subnet_cidr`: IP 관리자의 CIDR 주소가 노드의 `br-int`에 할당됩니다.
- `vc_host`: vCenter Server의 IP 주소입니다.
- `vc_user`: vCenter Server 관리자의 사용자 이름입니다.
- `vc_password`: vCenter Server 관리자의 암호입니다.
- `vms`: VM 이름의 쉼표로 구분된 목록입니다. 순서는 `os_node_name_list`와 일치해야 합니다.

다음 매개 변수에는 기본값이 있습니다. 필요에 따라 수정할 수 있습니다.

- `nsx_t0_router_name`: 클러스터의 Tier-0 논리적 라우터의 이름입니다. 기본값: **t0**
- `pod_ipblock_name`: 포드의 IP 블록 이름입니다. 기본값: **podIPBlock**
- `pod_ipblock_cidr`: 이 IP 블록의 CIDR 주소입니다. 기본값: **172.20.0.0/16**
- `snat_ippool_name`: SNAT에 대한 IP 블록의 이름입니다. 기본값은 **externalIP**입니다.
- `snat_ippool_cidr`: 이 IP 블록의 CIDR 주소입니다. 기본값: **172.30.0.0/16**
- `start_range`: 이 IP 풀에 대해 지정된 CIDR의 시작 IP 주소입니다. 기본값: **172.30.0.1**
- `end_range`: 이 IP 풀에 대해 지정된 CIDR의 끝 IP 주소입니다. 기본값: **172.30.255.254**
- `os_cluster_name`: OpenShift 클러스터의 이름입니다. 기본값: **occl-one**
- `nsx_node_ls_name`: 노드에 연결된 논리적 스위치의 이름입니다. 기본값: **node_ls**
- `nsx_node_lr_name`: 스위치 **node_ls**에 대한 논리적 라우터의 이름입니다. 기본값: **node_lr**

`nsx-config` 플레이북은 하나의 IP 풀과 하나의 IP 블록만 생성하도록 지원합니다. 더 필요한 경우 수동으로 생성해야 합니다.

2 openshift-ansible-nsx 디렉토리로 변경하고 `ncp_prep` 역할을 실행합니다.

```
ansible-playbook -i /PATH/T0/HOSTS/hosts ncp_prep.yaml
```

플레이북에는 다음 작업을 수행하기 위한 지침이 포함되어 있습니다.

- CNI 플러그인 설치 파일을 다운로드합니다.

파일 이름은 `nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm`입니다. 여기서 `xxxxxxx`는 빌드 번호입니다.

- CNI 플러그인 설치 파일을 설치합니다.

플러그인이 `/opt/cni/bin`에 설치됩니다. CNI 구성 파일 `10.net.conf`가 `/etc/cni/net.d`에 복사됩니다. `rpm`은 루프백 플러그인에 대한 구성 파일 `/etc/cni/net.d/99-loopback.conf`도 설치합니다.

- OVS 설치 파일을 다운로드하고 설치합니다.

파일은 `openvswitch-2.9.1.xxxxxxx-1.x86_64.rpm` 및 `openvswitch-kmod-2.9.1.xxxxxxx-1.el7.x86_64.rpm`입니다. 여기서 xxxxxxxx는 빌드 번호입니다.

- br-int 인스턴스가 아직 생성되지 않은 경우 생성합니다.

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

- 노드 논리적 스위치에 연결된 네트워크 인터페이스(node-if)를 br-int에 추가합니다.
- br-int 및 node-if link 상태가 [실행]인지 확인합니다.

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

- 네트워크 구성 파일을 업데이트하여 재부팅 후 네트워크 인터페이스가 작동되도록 합니다.
- NCP tar 파일을 다운로드하고 tar 파일의 Docker 이미지를 로드합니다.
- ncp-rbac yaml 파일을 다운로드하고 apiVersion을 v1으로 변경합니다.
- NSX-T Data Center에서 논리적 토폴로지와 관련 리소스를 생성하고 NCP에서 인식할 수 있도록 태그를 지정합니다.
- ncp.ini를 NSX-T Data Center 리소스 정보로 업데이트합니다.

다음에 수행할 작업

OpenShift Container Platform을 설치합니다. [OpenShift Container Platform 설치](#)의 내용을 참조하십시오.

OpenShift Container Platform 설치

OCP(OpenShift Container Platform)는 Docker 및 Kubernetes와 함께 제공되는 PaaS(Platform as a Service) 제품입니다.

참고 단일 플레이북을 사용하여 NCP와 OpenShift를 설치하는 경우에는 이 단계가 필요하지 않습니다. [단일 플레이북을 사용하여 NCP와 OpenShift 설치](#)의 내용을 참조하십시오.

OCP를 설치하는 방법에 대한 자세한 내용은 <https://docs.openshift.com>을 참조하십시오.

다음에 수행할 작업

NCP 및 NSX 노드 에이전트를 실행합니다. [NCP 및 NSX 노드 에이전트 실행](#)의 내용을 참조하십시오.

NCP 및 NSX 노드 에이전트 실행

NCP 및 NSX 노드 에이전트를 설치하고 실행합니다.

절차

- 1 roles/ncp/defaults/main.yaml을 편집하고 OpenShift API 서버 IP, NSX Manager IP 및 NCP ReplicationController yaml과 nsx-node-agent DaemonSet yaml을 다운로드하기 위한 URL을 지정합니다.
- 2 openshift-ansible-nsx 디렉토리에서 ncp 역할을 실행합니다.

```
ansible-playbook -i /PATH/TO/HOSTS/hosts ncp.yaml
```

ncp 역할은 다음 단계를 수행합니다.

- nsx-system 프로젝트가 있는지 확인하고 그렇지 않은 경우 프로젝트를 생성합니다.

```
oc new-project nsx-system
```

- ncp-rbac yaml 파일을 다운로드하고 apiVersion을 v1으로 변경합니다.
- NCP 포드에 대한 서비스 계정을 생성하고, NCP가 액세스할 수 있는 리소스를 지정한 후 클러스터 역할을 NCP 서비스 계정에 바인딩합니다.
- nsx-node-agent 포드에 대한 서비스 계정을 생성하고, 노드 에이전트가 액세스할 수 있는 리소스를 지정한 후 클러스터 역할을 노드 에이전트 서비스 계정에 바인딩합니다.

```
oc apply -f /tmp/ncp-rbac.yaml
```

- 위의 서비스 계정에 연결된 토큰을 가져와 /etc/nsx-ujo/<service_account>_token 아래에 저장합니다.

```
secret=`kubectll get serviceaccount ncp-svc-account -o yaml | grep -A1 secrets | tail -n1 | awk {'print $3'}`
kubectll get secret $secret -o yaml | grep 'token:' | awk {'print $2'} | base64 -d > /etc/nsx-ujo/ncp_token
secret=`kubectll get serviceaccount nsx-node-agent-svc-account -o yaml | grep -A1 secrets | tail -n1 | awk {'print $3'}`
kubectll get secret $secret -o yaml | grep 'token:' | awk {'print $2'} | base64 -d > /etc/nsx-ujo/node_agent_token
```

- NCP용 SCC(SecurityContextConstraint) yaml 파일인 ncp-os-scc.yaml을 다운로드하고 해당 yaml을 기준으로 SCC를 생성합니다.

```
oc apply -f ncp-os-scc.yaml
```

SCC yaml 파일은 NCP/nsx-node-agent가 SELinux에서 액세스 권한을 갖도록 SELinux 유형을 spc_t로 지정합니다. 즉, 다음과 같이 지정됩니다.

```
seLinuxContext:
  seLinuxOptions:
    type: spc_t
```

SCC yaml 파일에서 `seLinuxContext`의 `seLinuxOptions`에는 `ncp/node-agent` 에이전트가 다양한 파일 범주의 대상에 액세스할 수 있도록 SELinux 레이블 기반 액세스 제어 수준이 `s0:c0:c1023`으로 설정됩니다.

- NCP 및 NSX 노드 에이전트 포드를 생성하는 사용자에게 SCC를 추가합니다. 예를 들어 현재 프로젝트의 기본 사용자에게 SCC를 추가하려면 다음과 같이 합니다.

```
oc adm policy add-scc-to-user ncp-scc -z default
```

- NCP 및 NSX 노드 에이전트 서비스 계정에 SCC를 추가합니다.

```
oc adm policy add-scc-to-user ncp-scc -z ncp-svc-account
oc adm policy add-scc-to-user ncp-scc -z nsx-node-agent-svc-account
```

- NCP RC(ReplicationController) 및 `nsx-node-agent` DS(DaemonSet)용 yaml 파일을 다운로드하고 ConfigMap을 업데이트합니다.
- NCP 이미지를 다운로드하고 로드합니다(`nsx-node-agent`는 동일한 이미지를 사용함).
- 서비스 계정을 구성하고 NCP 및 `nsx_node_agent`에 필요한 SecurityContextConstraint를 설정합니다.
- NCP ReplicationController 및 `nsx-node-agent` DaemonSet을 생성합니다.

참고 NCP가 Kubernetes API 서버에 대한 영구 HTTP 연결을 열어 Kubernetes 리소스의 수명 주기 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. 그렇지 않으면 NCP가 새 이벤트를 확인하지 못합니다.

설정 메모

OpenShift 및 NCP를 설치하기 전에 다음 정보를 기록하십시오.

- 포드에는 11개 이하의 레이블이 있어야 하고 네임스페이스에는 12개 이하의 레이블이 있어야 합니다.
- OpenShift 내부용으로 추가된 레이블, 예를 들어 접두사 `openshift.io`가 키에 포함된 레이블은 NCP에서 무시되므로 관련 NSX 리소스에 생성된 해당 태그가 사용자에게 표시되지 않습니다. 다음은 OpenShift에서 사용하는 레이블 접두사 목록으로, 다음 중 하나로 시작하는 레이블 키는 사용하지 않아야 합니다.

```
openshift.io
pod-template
```

- 노드에는 포드에 대한 액세스 권한이 필요합니다(예: Kubelet 상태 검사). 호스트 관리 인터페이스에서 포드 네트워크에 액세스할 수 있는지 확인합니다.

- 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, 제한된 anyuid SCC를 사용하면 NET_ADMIN이 부여되지 않습니다. NET_ADMIN을 명시적으로 사용하도록 설정하거나 포트가 권한 모드로 실행되도록 하는 SCC에 유의하십시오. 또한 신뢰할 수 없는 컨테이너의 경우, 예를 들어 NET_RAW 기능을 제거한 상태로 anyuid SCC를 기반의 별도 SCC를 생성합니다. 이 작업은 SCC 정의의 `requiredDropCapabilities` 목록에 NET_RAW를 추가하여 수행할 수 있습니다.
- 포트/컨테이너에서 루트 액세스를 허용합니다(테스트 전용). 아래 명령을 실행하려면 현재 로그인된 oc 프로젝트의 모든 포트에서 루트 액세스 권한이 필요합니다.

```
oc new-project test-project
oc project test-project
oc adm policy add-scc-to-user anyuid -z default
```

- OpenShift 레지스트리를 구성(추가)합니다.

```
oc login -u system:admin -n default
oc adm registry --service-account=registry --config=/etc/origin/master/admin.kubeconfig
```

- OpenShift 레지스트리 삭제

```
oc login -u system:admin -n default
oc delete svc/docker-registry dc/docker-registry
```

- Docker 기본 브리지 컨테이너에서 노드의 dnsmasq 프로세스에 대한 DNS 요청을 허용하는 IPtables 방화벽 규칙이 없습니다. 따라서 수동으로 열어야 합니다. /etc/sysconfig/iptables를 편집하고 파일 맨 아래의 COMMIT 앞에 다음 규칙을 추가합니다.

```
-A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A OS_FIREWALL_ALLOW -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
COMMIT
```

- iptables, docker 및 origin-node를 다시 시작합니다(kube-proxy 및 kubelet 다시 시작).

```
systemctl restart iptables
systemctl restart docker
systemctl restart origin-node
```

- OpenShift가 작동하려면 OpenShift의 내부 docker 레지스트리가 비 TLS를 사용하도록 허용되어야 합니다. 일반적으로는 OpenShift Ansible 설치 관리자에 의해 자동으로 추가되지만 현재 작동하지 않는 것 같습니다. /etc/sysconfig/docker를 편집하고 다음을 추가합니다.

```
INSECURE_REGISTRY='--insecure-registry 172.30.0.0/16'
```

- Docker를 다시 시작합니다.

```
systemctl restart docker
```

- NCP에서 지원하는 네트워크 정책은 Kubernetes에서 지원하는 네트워크 정책과 동일하며, 이는 OpenShift에 사용되는 Kubernetes 버전에 따라 다릅니다.
 - OpenShift 3.9 - 네트워크 정책의 규칙 절에 namespaceSelector, podSelector 및 ipBlock 선택기를 최대 하나만 포함할 수 있습니다.
- Kubernetes API 서버는 네트워크 정책 규칙에 대한 검증을 수행하지 않습니다. 따라서 잘못된 네트워크 정책을 생성할 가능성이 있습니다. NCP는 이러한 네트워크 정책을 거부합니다. 네트워크 정책을 업데이트하여 올바르게 만들더라도 NCP가 해당 네트워크 정책을 처리하지 않습니다. 이 경우에는 네트워크 정책을 삭제한 후 규칙이 올바른 네트워크 정책을 다시 생성해야 합니다.
- 특정 버전의 Kubernetes에는 subPath 관련 문제가 있습니다 (<https://github.com/kubernetes/kubernetes/issues/61076> 참조). 이 문제에 대한 수정이 OpenShift 버전에 포함되어 있지 않으면 CreateContainerConfigError: volumeMount를 위한 subPath를 준비하지 못했습니다. 오류가 발생하고 NCP 포트 생성이 실패합니다. 이 문제는 NCP yaml에서 subPath를 사용하지 않는 방법으로 해결할 수 있습니다. 예를 들어 subPath: ncp.ini가 포함된 줄을 제거하고 volumes에 대한 구성을 다음으로 교체합니다.

```
volumes:
- name: config-volume
  # ConfigMap nsx-ncp-config is expected to supply ncp.ini
  configMap:
    name: nsx-ncp-config
    items:
      - key: ncp.ini
        path: ncp.ini
```

이렇게 변경할 경우 전체 /etc/nsx-ujo 디렉토리가 읽기 전용이 된다는 단점이 있습니다. 그 결과, NCP가 인증서와 개인 키를 단일 파일로 이동하기 위한 임시 파일을 /etc/nsx-ujo 아래에 생성할 수 없기 때문에 인증서 및 개인 키를 사용하여 NSX-T에 연결할 수 없게 됩니다.

- OpenShift 3.10 클러스터로 업그레이드하거나 현재 이 버전을 실행 중이면 다음을 참조하십시오.
 - OpenShift 3.10 클러스터에 맞는 노드 그룹 구성을 지정해야 합니다. 인벤토리 호스트 파일에 노드 구성 맵 구성을 제공해야 합니다.
 - 인벤토리 호스트 파일의 [nodes] 그룹에 정의된 모든 호스트를 노드 그룹 이름에 할당해야 합니다.
 - Ansible 플레이북에서 OpenShift 클러스터를 업그레이드하면 네트워크 손실이 발생할 수 있습니다. Open vSwitch 패키지가 중지/제거되지 않도록 Open openshift-ansible 저장소에 패치(<https://github.com/openshift/openshift-ansible/pull/8016/files#diff-2386e21861da3f95091dbb27d72ca366>)를 추가해야 합니다.

- OpenShift 3.10부터는 kube-proxy가 openshift-node 서비스에서 DaemonSet로 이동했습니다. 이제 더 이상 기본적으로 시작되지 않습니다. kube-proxy를 수동으로 시작하려면 다음 단계를 수행합니다(openshift-ansible 리포지토리가 복제되었다고 가정함).
- openshift-ansible 디렉토리로 이동하고 [defaults] 아래에서 다음을 설정합니다.

```
library = roles/lib_utils/library/
```

- 플레이북 디렉토리에서 다음 항목이 포함된 create_proxy.yaml 파일을 생성합니다.

```
- import_playbook: byo/openshift_facts.yml
- hosts: masters
  run_once: True
  roles:
    - kube_proxy_and_dns
```

- 플레이북을 실행합니다.

```
ansible-playbook -i hosts playbooks/create_proxy.yaml
```

일부 작업이 실패했음을 나타내는 오류 메시지가 표시됩니다. 이러한 메시지는 무시할 수 있습니다. oc get po --all-namespaces 명령을 실행하여 결과를 확인할 수 있습니다.

베어메탈 환경에서 NCP 설치

베어 메탈 환경에서 NCP(NSX-T Container Plug-in)를 설치하는 단계는 비베어메탈 환경에서 NCP를 설치하는 단계와 비슷합니다. 이 섹션에서는 베어메탈 환경에만 해당하는 단계를 설명합니다.

본 장은 다음 항목을 포함합니다.

- NSX-T Data Center CNI 플러그인 설치
- OpenShift 노드에 대한 NSX-T Data Center 네트워킹 구성
- NSX 노드 에이전트 설치
- nsx-node-agent-ds.yml의 ncp.ini에 대한 Configmap
- NSX-T Container Plug-in 설치
- ncp-rc.yml의 ncp.ini에 대한 Configmap

NSX-T Data Center CNI 플러그인 설치

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

절차

- 1 Linux 배포에 적합한 설치 파일을 다운로드합니다.

파일 이름은 nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm입니다. 여기서 xxxxxxxx는 빌드 번호입니다.

- 2 1단계에서 다운로드한 rpm 파일을 설치합니다.

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

OpenShift 노드에 대한 NSX-T Data Center 네트워킹 구성

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

NSX Manager에 각 노드를 RHEL Container OS 유형으로 등록해야 합니다. 이 노드의 관리 인터페이스는 OpenShift 클러스터를 가입시키는 데 사용할 수 있으며 NSX-T Data Center 패브릭에 있거나 없을 수 있습니다. 기타 인터페이스는 포트에 대한 네트워킹을 제공하며 NSX-T Data Center 패브릭에 있어야 합니다.

해당 전송 노드에는 다음과 같은 태그가 있어야 합니다.

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

NSX Manager GUI에서 **패브릭 > 노드**로 이동하여 OpenShift 노드에 대한 전송 노드를 식별할 수 있습니다.

OpenShift 노드 이름이 변경되면 ncp/node_name 태그를 업데이트하고 NCP를 다시 시작해야 합니다. 다음 명령을 사용하여 노드 이름을 가져올 수 있습니다.

```
oc get nodes
```

NCP가 실행되는 동안 클러스터에 노드를 추가하는 경우 oc cluster add 명령을 실행하기 전에 전송 노드에 태그를 추가해야 합니다. 그렇지 않으면 새 노드에 네트워크 연결이 설정되지 않습니다. 태그가 잘못되었거나 누락된 경우 다음 단계에 따라 문제를 해결할 수 있습니다.

- 전송 노드에 올바른 태그를 적용합니다.
- NCP를 다시 시작합니다.

NSX 노드 에이전트 설치

NSX 노드 에이전트는 각 포트가 두 개의 컨테이너를 실행하는 DaemonSet입니다. 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-proxy를 실행합니다. 이는 업스트림 kube-proxy와 동일한 기능을 구현합니다.

절차

- 1 NCP Docker 이미지를 다운로드합니다.

파일 이름은 nsx-ncp-xxxxxxx.tar입니다. 여기서 xxxxxxxx는 빌드 번호입니다.

- 2 NSX 노드 에이전트 DaemonSet yaml 템플릿을 다운로드합니다.

파일 이름은 nsx-node-agent-ds.yml입니다. 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.yml을 편집합니다.

이미지 이름을 로드된 이미지로 변경합니다.

다음과 같이 변경합니다.

```
[coe]
node_type = 'BAREMETAL'
...
[nsx_node_agent]
ovs_bridge = 'nsx-managed'
```

다음 줄의 주석 처리를 해제합니다.

```
securityContext:
  capabilities:
    add:
      - NET_ADMIN
      - SYS_ADMIN
      - SYS_PTRACE
      - DAC_READ_SEARCH
      # For BMC usecase
      - DAC_OVERRIDE
  volumeMounts:
    ...
    # mount nestdb-sock for baremetal node
    - name: nestdb-sock
      mountPath: /var/run/vmware/nestdb/nestdb-server.sock
  volumes:
    ...
    # volume for baremetal node
    - name: nestdb-sock
      hostPath:
        path: /var/run/vmware/nestdb/nestdb-server.sock
        type: Socket
```

참고 yaml 파일에서 ncp.ini에 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.

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

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

샘플 yaml 파일 nsx-node-agent-ds.yml에는 NSX 노드 에이전트의 구성 파일 ncp.ini에 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.yml에는 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 = BAREMETAL

[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-T Container Plug-in 설치

NCP(NSX-T 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.tar입니다. 여기서 xxxxxxxx는 빌드 번호입니다.

- 2 NCP ReplicationController yaml 템플릿을 다운로드합니다.

파일 이름은 ncp-rc.yml입니다. 이 파일을 편집하거나 템플릿 파일의 예제로 사용할 수 있습니다.

3 NCP Docker 이미지를 이미지 레지스트리에 로드합니다.

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

4 ncp-rc.yml을 편집합니다.

노드 유형을 베어메탈로 설정합니다.

```
[coe]
node_type = 'BAREMETAL'
```

이미지 이름을 로드된 이미지로 변경합니다.

nsx_api_managers 매개 변수를 지정합니다. 이 릴리스는 단일 Kubernetes 노드 클러스터 및 단일 NSX Manager 인스턴스를 지원합니다. 예:

```
nsx_api_managers = 192.168.1.180
```

(선택 사항) [nsx_v3] 섹션에 매개 변수 ca_file을 지정합니다. 이 값은 NSX Manager 서버 인증서를 확인하는 데 사용할 CA 번들 파일이어야 합니다. 설정하지 않으면 시스템 루트 CA가 사용됩니다.

NSX-T Data Center에 대한 인증을 위해 매개 변수 nsx_api_cert_file 및 nsx_api_private_key_file을 지정합니다.

nsx_api_cert_file은 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_file은 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 = nat를 지정합니다.

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

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

5 NCP ReplicationController를 생성합니다.

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

참고 NCP가 Kubernetes API 서버에 대한 영구 HTTP 연결을 열어 Kubernetes 리소스의 수명 주기 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 중에는 컨테이너 호스트를 재부팅하지 마십시오. 호스트를 재부팅하면 재부팅 후에 NCP 포트 2개가 실행될 수 있습니다. 이 경우 다음을 수행합니다.

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

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

- 네임스페이스 nsx-system을 삭제합니다. 예를 들면 다음과 같습니다.

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

ncp-rc.yml의 ncp.ini에 대한 Configmap

샘플 YAML 파일 ncp-rc.yml에는 구성 파일 ncp.ini에 대한 ConfigMap이 포함되어 있습니다. 이 ConfigMap 섹션에는 이전 섹션에 설명된 대로 NCP를 설치하기 전에 지정해야 하는 매개 변수가 포함되어 있습니다.

다운로드하는 샘플 ncp-rc.yml에는 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-uj0/'. 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 = BAREMETAL

[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>

# 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>
```

로드 밸런싱

NSX-T Data Center 로드 밸런서는 OpenShift와 통합되어 OpenShift 라우터로 작동합니다.

NCP는 OpenShift 라우팅 및 끝점 이벤트를 감시하고 라우팅 규칙을 기반으로 로드 밸런서에서 로드 밸런싱 규칙을 구성합니다. 결과적으로 NSX-T Data Center 로드 밸런서는 규칙에 따라 수신 계층 7 트래픽을 적절한 백엔드 포드로 전달합니다.

로드 밸런싱 구성

로드 밸런싱 구성에는 Kubernetes LoadBalancer 서비스 또는 OpenShift 라우팅 구성이 포함됩니다. NCP 복제 컨트롤러도 구성해야 합니다. LoadBalancer 서비스는 계층 4 트래픽용이고 OpenShift 라우팅은 계층 7 트래픽용입니다.

Kubernetes LoadBalancer 서비스를 구성하면 사용자가 구성한 외부 IP 블록의 IP 주소가 할당됩니다. 로드 밸런서는 이 IP 주소와 서비스 포트를 사용합니다. LoadBalancer 정의의 loadBalancerIP 규칙을 사용하여 IP 풀의 이름 또는 ID를 지정할 수 있습니다. 이 IP 풀에서 LoadBalancer 서비스의 IP가 할당됩니다. loadBalancerIP 규칙이 비어 있으면 IP가 사용자가 구성하는 외부 IP 블록에서 할당됩니다.

NCP 2.3.1부터는 loadBalancerIP에서 지정한 IP 풀에 태그 `{"ncp/owner": cluster:<cluster>}`가 있어야 합니다.

NSX-T Data Center 로드 밸런서를 사용하려면 NCP에서 로드 밸런싱을 구성해야 합니다. `ncp_rc.yml` 파일에서 다음을 수행합니다.

- 1 `use_native_loadbalancer`를 True로 설정합니다.
- 2 `pool_algorithm`을 `WEIGHTED_ROUND_ROBIN`으로 설정합니다.
- 3 `lb_default_cert_path` 및 `lb_priv_key_path`를 각각 CA 서명된 인증서 파일 및 개인 키 파일의 전체 경로 이름으로 설정합니다. CA 서명된 인증서를 생성하는 샘플 스크립트는 아래를 참조하십시오. 또한 기본 인증서 및 키를 NCP 포트에 마운트합니다. 자세한 내용은 아래를 참조하십시오.

- 4 (선택 사항) l4_persistence 및 l7_persistence 매개 변수를 사용하여 지속성 설정을 지정합니다. 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
```

- 5 (선택 사항) service_size를 SMALL, MEDIUM 또는 LARGE로 설정합니다. 기본값은 SMALL입니다.
- 6 OpenShift 3.11을 실행하는 경우 OpenShift에서 LoadBalancer 서비스에 IP를 할당하지 않도록 다음 구성을 수행해야 합니다.
- /etc/origin/master/master-config.yaml 파일의 networkConfig 아래에서 ingressIPNetworkCIDR을 0.0.0.0/32로 설정합니다.
 - 다음 명령을 사용하여 API 서버와 컨트롤러를 다시 시작합니다.

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

참고 계층 4 및 계층 7 로드 밸런서를 둘 다 구성하는 경우 l4_persistence 또는 l7_persistence 중 하나 또는 두 가지 모두를 source_ip로 설정할 수 있지만 l4_persistence를 source_ip로 설정하고 l7_persistence를 cookie로 설정할 수는 없습니다. 실수로 l4_persistence를 source_ip로, l7_persistence를 cookie로 설정한 경우 LoadBalancer 서비스는 작동하지 않습니다. 문제를 해결하려면 수신 리소스와 LoadBalancer 서비스를 삭제하고, 지속성 설정을 변경하고, NCP를 다시 시작한 후 수신 리소스와 LoadBalancer 서비스를 다시 생성해야 합니다.

계층 7 로드 밸런서 예

다음 YAML 파일은 계층 7 로드 밸런싱을 제공하는 두 가지 복제 컨트롤러(tea-rc 및 coffee-rc)와 두 가지 서비스(tea-svc 및 coffee-svc) 및 두 가지 라우팅(cafe-route-multi 및 cafe-route)을 구성합니다.

```
# RC
apiVersion: v1
kind: ReplicationController
metadata:
  name: tea-rc
spec:
  replicas: 2
  template:
```

```

    metadata:
      labels:
        app: tea
    spec:
      containers:
      - name: tea
        image: nginxdemos/hello
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: coffee-rc
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
      - name: coffee
        image: nginxdemos/hello
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
---
# Services
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
    app: tea
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
    name: http
  selector:
    app: tea
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
  labels:
    app: coffee
spec:
  ports:
  - port: 80
    targetPort: 80

```

```

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

```

추가 정보

- HTTPS 트래픽에는 Edge 종료만 지원됩니다.
- 와일드카드 하위 도메인이 지원됩니다. 예를 들어 wildcardPolicy가 **하위 도메인**으로 설정되고 호스트 이름이 **wildcard.example.com**으로 설정되면 ***.example.com**에 대한 모든 요청이 처리됩니다.
- 잘못된 구성으로 인해 라우팅 이벤트가 처리되는 동안 NCP에서 오류가 발생하면 라우팅 YAML 파일을 수정하고 라우팅 리소스를 삭제한 다음 다시 생성해야 합니다.
- NCP는 네임스페이스별로 호스트 이름 소유권을 적용하지 않습니다.
- Kubernetes 클러스터당 하나의 Loadbalancer 서비스가 지원됩니다.
- NSX-T Data Center는 각 LoadBalancer 서비스 포트에 대해 계층 4 로드 밸런서 가상 서버와 풀을 생성합니다. TCP와 UDP가 모두 지원됩니다.
- NSX-T Data Center 로드 밸런서는 다양한 크기로 제공됩니다. NSX-T Data Center 로드 밸런서 구성에 대한 자세한 내용은 NSX-T 관리 가이드를 참조하십시오.

소규모 NSX-T Data Center 로드 밸런서는 다음을 지원합니다.

- NSX-T 가상 서버 10개
- NSX-T 풀 10개
- NSX-T 풀 멤버 30개
- LoadBalancer 서비스용 포트 8개
- LoadBalancer 서비스 및 라우팅 리소스에 정의된 포트 총 10개
- LoadBalancer 서비스 및 라우팅 리소스가 참조하는 끝점 총 30개

중간 규모 NSX-T Data Center 로드 밸런서는 다음을 지원합니다.

- NSX-T 가상 서버 100개
- NSX-T 풀 100개
- NSX-T 풀 멤버 300개
- LoadBalancer 서비스용 포트 98개
- LoadBalancer 서비스 및 라우팅 리소스에 정의된 포트 총 100개
- LoadBalancer 서비스 및 라우팅 리소스가 참조하는 끝점 총 300개

대규모 NSX-T Data Center 로드 밸런서는 다음을 지원합니다.

- NSX-T 가상 서버 1000개
- NSX-T 풀 1000개
- NSX-T 풀 멤버 3000개
- LoadBalancer 서비스용 포트 998개
- LoadBalancer 서비스 및 라우팅 리소스에 정의된 포트 총 1000개
- LoadBalancer 서비스 및 라우팅 리소스가 참조하는 끝점 총 3000개

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

- NCP 2.3.1부터는 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 `l4_lb_auto_scaling`을 **false**로 설정하여 이 기능을 사용하지 않도록 설정할 수 있습니다. 이 기능을 사용하려면 NSX-T Data Center 2.3 이상의 릴리스가 필요합니다.

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. aes256과 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.key로 각각 지정되었다고 가정하고 호스트의 해당 파일이 포트에 마운트되도록 ncp-rc.yaml을 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
```

NSX-T Container Plug-in 관리

6

NSX Manager GUI 또는 CLI(명령줄 인터페이스)에서 NSX-T 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.

- [NSX Manager GUI에서 IP 블록 관리](#)
- [NSX Manager GUI에서 IP 블록 서브넷 보기](#)
- [CIF 연결 논리적 포트](#)
- [CLI 명령](#)
- [오류 코드](#)

NSX Manager GUI에서 IP 블록 관리

NSX Manager GUI에서 IP 블록을 추가, 삭제, 편집하고, 세부 정보를 보고, 태그를 관리할 수 있습니다.

절차

- 1 브라우저에서 NSX Manager(<https://<nsx-manager-IP-address-or-domain-name>>)에 로그인합니다.
- 2 **네트워킹 > IPAM**으로 이동합니다.
기존 IP 블록 목록이 표시됩니다.

3 다음 작업 중 하나를 수행합니다.

옵션	작업
IP 블록 추가	추가를 클릭합니다.
하나 이상의 IP 블록 삭제	하나 이상의 IP 블록을 선택하고 삭제를 클릭합니다.
IP 블록 편집	IP 블록을 선택하고 편집을 클릭합니다.
IP 블록에 대한 세부 정보 보기	IP 블록 이름을 클릭합니다. 개요 탭을 클릭하여 일반 정보를 확인합니다. 서브넷 탭을 클릭하여 이 IP 블록의 서브넷을 확인합니다.
IP 블록에 대한 태그 관리	IP 블록을 선택하고 작업 > 태그 관리를 클릭합니다.

서브넷이 할당된 IP 블록은 삭제할 수 없습니다.

NSX Manager GUI에서 IP 블록 서브넷 보기

NSX Manager GUI에서 IP 블록의 서브넷을 볼 수 있습니다. NCP를 설치하고 실행한 이후에는 IP 블록 서브넷을 추가하거나 삭제하지 않는 것이 좋습니다.

절차

- 1 브라우저에서 NSX Manager(<https://<nsx-manager-IP-address-or-domain-name>>)에 로그인합니다.
- 2 **네트워킹 > IPAM**으로 이동합니다.
기존 IP 블록 목록이 표시됩니다.
- 3 IP 블록 이름을 클릭합니다.
- 4 **서브넷** 탭을 클릭합니다.

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 가이드를 참조하십시오.

네트워크 모니터링 도구

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

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

CLI 명령

CLI 명령을 실행하려면 NSX-T 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
상태	<code>get ncp-master status</code>
상태	<code>get ncp-nsx status</code>
상태	<code>get ncp-watcher <watcher-name></code>
상태	<code>get ncp-watchers</code>
상태	<code>get ncp-k8s-api-server status</code>
상태	<code>check projects</code>
상태	<code>check project <project-name></code>
캐시	<code>get project-cache <project-name></code>
캐시	<code>get project-caches</code>
캐시	<code>get namespace-cache <namespace-name></code>
캐시	<code>get namespace-caches</code>
캐시	<code>get pod-cache <pod-name></code>
캐시	<code>get pod-caches</code>
캐시	<code>get ingress-caches</code>
캐시	<code>get ingress-cache <ingress-name></code>
캐시	<code>get ingress-controllers</code>
캐시	<code>get ingress-controller <ingress-controller-name></code>

표 6-1. NCP 컨테이너의 CLI 명령 (계속)

유형	명령
캐시	get network-policy-caches
캐시	get network-policy-cache <pod-name>
지원	get ncp-log file <filename>
지원	get ncp-log-level
지원	set ncp-log-level <log-level>
지원	get support-bundle file <filename>
지원	get node-agent-log file <filename>
지원	get node-agent-log file <filename> <node-name>

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 :

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

■ 수신, 네임스페이스, 포트 및 서비스에 대한 감시자 상태 표시

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

예 1 :

```
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
```

예 2 :

```
kubecode> 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
```

■ 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 8acc9cd-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
```

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-ad32-338dcb92316f
  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-8fd2-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

```

■ 네트워크 정책 캐시 또는 특정 네트워크 정책 캐시 가져오기

```

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

```

예 :

```

kubenode> get network-policy-caches
nsx.testns.allow-tcp-80:
  dest_labels: None
  dest_pods:
    50.0.2.3
  match_expressions:
    key: tier
    operator: In
    values:
      cache
  name: allow-tcp-80
  np_dest_ip_set_ids:
    22f82d76-004f-4d12-9504-ce1cb9c8aa00
  np_except_ip_set_ids:
    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

kubecall> 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:
  14f7f825-f1a0-408f-bbd9-bb2f75d44666
np_ip_set_ids:
  14f7f825-f1a0-408f-bbd9-bb2f75d44666
np_isol_section_id: c8d93597-9066-42e3-991c-c550c46b2270
np_section_id: 04693136-7925-44f2-8616-d809d02cd2a9
ns_name: testns
src_egress_rules: None
src_egress_rules_hash: 97d170e1550eee4afc0af065b78cda302a97674c
src_pods:
  50.0.2.0/24
src_rules:
  from:
    namespaceSelector:
      matchExpressions:
        key: tier
        operator: DoesNotExist
      matchLabels:
        ns: myns
    ports:
      port: 80
      protocol: TCP
  src_rules_hash: e4ea7b8d91c1e722670a59f971f8fcc1a5ac51f1

```

NCP 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 **CRITICAL**입니다.

```
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 1 :

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

예 2 :

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

NSX Kube Proxy 컨테이너의 상태 명령

- Kube Proxy와 Kubernetes API 서버 간의 연결 상태 표시

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

예 :

```
kubecall> 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 1 :

```
kubecall> 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
```

예 2 :

```
kubecall> 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
```

■ 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
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

오류 코드	설명
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
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
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

오류 코드	설명
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
NCP00090	로드 밸런서 풀 없음 오류
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 엠퍼링크를 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
NCP05009	Kubernetes 리소스 처리 실패
NCP05010	Kubernetes 서비스 처리 실패
NCP05011	Kubernetes 끝점 처리 실패
NCP05012	Kubernetes 수신 처리 실패

오류 코드	설명
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 주식 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

OpenShift 오류 코드

오류 코드	설명
NCP07001	OC 경로 처리 실패
NCP07002	OC 경로 상태 업데이트 실패