

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

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



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](mailto:docfeedback@vmware.com)

**VMware, Inc.**

3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://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](http://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 6

### 2 NSX-T 리소스 설정 8

NSX-T 리소스 구성 8

### 3 NCP 설치 12

시스템 요구 사항 12

Ansible 호스트 파일 준비 13

### 4 로드 밸런싱 17

로드 밸런싱 구성 17

### 5 NSX Container Plug-in 관리 24

NSX Manager GUI에서 IP 블록 관리 24

NSX Manager GUI에서 IP 블록 서브넷 보기 25

CIF 연결 논리적 포트 25

CLI 명령 26

오류 코드 37

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

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

소프트웨어 제품	버전
NSX-T Data Center	2.3, 2.4
컨테이너 호스트 VM용 하이퍼바이저	<ul style="list-style-type: none"> <li>■ <a href="#">지원되는 vSphere 버전</a></li> <li>■ RHEL KVM 7.4, 7.5, 7.6</li> </ul>
컨테이너 호스트 운영 체제	RHEL 7.4, 7.5, 7.6
서비스 형태의 플랫폼	OpenShift 3.10, 3.11
컨테이너 호스트 Open vSwitch	2.10.2(NSX-T Data Center 2.4와 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 Ansible 호스트 파일을 준비합니다.
- 8 단일 플레이북을 사용하여 NCP와 OpenShift를 설치합니다.

## NCP 업그레이드

이 섹션에서는 NCP를 2.4.0으로 업그레이드하는 방법에 대해 설명합니다.

### 절차

- 1 CNI RPM 패키지, NSX 노드 에이전트 DaemonSet 및 NCP ReplicationController를 업그레이드합니다.

## 2 Ansible 호스트 파일을 준비합니다.

각 노드에는 `openshift_node_group_name` 매개 변수가 지정되어 있어야 합니다. 예를 들면 다음과 같습니다.

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

## 3 (선택 사항) 로드 밸런싱을 구성합니다.

LoadBalancer 서비스를 위한 외부 IP 주소에 대해 다른 IP 풀을 지정하는 단계를 추가합니다. 예를 들면 다음과 같습니다.

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

## NSX-T 리소스 설정

OpenShift 노드에 대한 네트워킹을 제공하려면 NSX-T Data Center 리소스를 생성해야 합니다.

### 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 로그인하여 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 로그인하여 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 로그인하여 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 하나 이상의 IP 블록을 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 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 포드 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. IP 풀에 태그 {"ncp/owner": cluster:<cluster>}가 있어야 합니다.
- NSX-T Data Center에서 서비스에 대한 SNAT 규칙의 우선 순위는 프로젝트의 우선 순위보다 높습니다.
- 포드가 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 방화벽 섹션을 방해하지 않게 하려면 NSX Manager에 로그인하고 2개의 방화벽 섹션을 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.

## NCP 설치

NCP는 OpenShift와 완전하게 통합됩니다. Ansible 호스트 파일에서 필요한 매개 변수를 추가하고 OpenShift를 설치하면 NCP가 자동으로 설치됩니다.

본 장은 다음 항목을 포함합니다.

- [시스템 요구 사항](#)
- [Ansible 호스트 파일 준비](#)

### 시스템 요구 사항

OpenShift를 설치하기 전에 환경이 특정 요구 사항을 충족하는지 확인합니다.

#### 일반 요구 사항

- Ansible 2.4 이상.

#### 가상 시스템 요구 사항

OpenShift 노드 VM에는 다음과 같은 2개의 vNIC가 있어야 합니다.

- 관리 tier-1 라우터에 대한 업링크가 있는 논리적 스위치에 연결된 관리 vNIC.
- 모든 VM의 두 번째 vNIC에는 NCP가 특정 OpenShift 노드에서 실행 중인 모든 POD에 대한 상위 VIF로 사용되는 포트를 알 수 있도록 NSX-T에 다음 태그가 있어야 합니다.

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

#### 베어메탈 시스템 요구 사항

- OpenShift 노드가 NSX-T 전송 노드여야 하며 위에 설명된 태그가 VIF 대신 전송 노드에 적용되어야 합니다.
- Ansible 호스트 파일에는 nsx\_node\_type='BAREMETAL' 설정이 있어야 합니다.

## NSX-T 요구 사항

- Tier-0 라우터.
- 오버레이 전송 영역.
- POD 네트워킹을 위한 IP 블록.
- (선택 사항) 라우팅(비NAT) POD 네트워킹을 위한 IP 블록.
- SNAT용 IP 풀. 기본적으로 POD 네트워킹을 위한 IP 블록은 NSX-T 내에서만 라우팅할 수 있습니다. NCP는 이 IP 풀을 사용하여 외부에 대한 연결을 제공합니다.
- (선택 사항) 상단 및 하단 방화벽 섹션. NCP는 이러한 두 섹션 사이에 Kubernetes 네트워크 정책 규칙을 배치합니다.
- Open vSwitch 및 CNI 플러그인 RPM을 OpenShift 노드 VM에서 연결 가능한 HTTP 서버에서 호스팅해야 합니다.

## NCP Docker 이미지

현재 NCP Docker 이미지는 공개적으로 사용할 수 없습니다. 로컬 개인 레지스트리에 이미지 nsx-ncp가 있거나 다음을 수행해야 합니다.

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

모든 노드에서:

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

## Ansible 호스트 파일 준비

OpenShift와 통합할 NCP에 대해 Ansible 호스트 파일에서 NCP 매개 변수를 지정해야 합니다.

Ansible 호스트 파일에서 다음 매개 변수를 지정한 후 OpenShift를 설치하면 NCP가 자동으로 설치됩니다.

- openshift\_use\_nsx=True
- openshift\_use\_openshift\_sdn=False
- os\_sdn\_network\_plugin\_name='cni'
- nsx\_openshift\_cluster\_name='ocp-cluster1'

(필수 사항) 이는 여러 OpenShift/Kubernetes 클러스터가 동일한 NSX Manager에 연결될 수 있기 때문에 필요합니다.

- `nsx_api_managers='10.10.10.10'`  
(필수 사항) NSX Manager의 IP 주소입니다. NSX Manager 클러스터의 경우 쉼표로 구분된 IP 주소를 지정합니다.
- `nsx_tier0_router='MyT0Router'`  
(필수 사항) 프로젝트의 Tier-1 라우터가 연결될 Tier-0 라우터의 이름 또는 UUID입니다.
- `nsx_overlay_transport_zone='my_overlay_tz'`  
(필수 사항) 논리적 스위치를 생성하는 데 사용될 오버레이 전송 영역의 이름 또는 UUID입니다.
- `nsx_container_ip_block='ip_block_for_my_ocp_cluster'`  
(필수 사항) NSX-T에 구성된 IP 블록의 이름 또는 UUID입니다. 이 IP 블록 중 프로젝트당 서브넷이 있습니다. 이러한 네트워크는 SNAT 뒤에 위치하며 라우팅할 수 없습니다.
- `nsx_ovs_uplink_port='ens224'`  
(필수 사항) HOSTVM 모드에 있는 경우, NSX-T에는 OCP 노드에서 POD 네트워킹을 위한 관리 vNIC와 다른 두 번째 vNIC가 필요합니다. 두 vNIC를 모두 NSX-T 논리적 스위치에 연결하는 것이 매우 권장됩니다. 두 번째(비관리) vNIC를 여기에 제공해야 합니다. 베어메탈의 경우에는 이 매개 변수가 필요하지 않습니다.
- `nsx_cni_url='http://myserver/nsx-cni.rpm'`  
(필수 사항) NCP가 노드를 부트스트랩할 수 있을 때까지 임시 요구 사항입니다. nsx-cni를 http 서버에 배치해야 합니다.
- `nsx_ovs_url='http://myserver/openvswitch.rpm'`
- `nsx_kmod_ovs_url='http://myserver/kmod-openvswitch.rpm'`  
(필수 사항) NCP가 노드를 부트스트랩할 수 있을 때까지 임시 매개 변수입니다. 베어메탈 설정에서 무시될 수 있습니다.
- `nsx_node_type='HOSTVM'`  
(선택 사항) 기본값은 HOSTVM입니다. OpenShift가 VM에서 실행되고 있지 않으면 BAREMETAL로 설정합니다.
- `nsx_k8s_api_ip=192.168.10.10`  
(선택 사항) 설정하는 경우 NCP가 이 IP 주소 또는 Kubernetes 서비스 IP와 통신합니다.
- `nsx_k8s_api_port=192.168.10.10`  
(선택 사항) Kubernetes 서비스의 경우 기본값은 443입니다. nsx\_k8s\_api\_ip와 함께 사용하여 마스터 노드 IP를 지정하는 경우 8443으로 설정합니다.
- `nsx_insecure_ssl=true`  
(선택 사항) NSX Manager가 신뢰할 수 없는 인증서와 함께 제공될 때 기본값은 true입니다. 인증서를 신뢰할 수 있는 인증서로 변경한 경우 false로 설정할 수 있습니다.
- `nsx_api_user='admin'`

- `nsx_api_password='super_secret_password'`

- `nsx_subnet_prefix=24`

(선택 사항) 기본값은 24입니다. 이는 OpenShift 프로젝트당 전용으로 사용될 서브넷 크기입니다. POD 수가 서브넷 크기를 초과할 경우 동일한 서브넷 크기의 새 논리적 스위치가 프로젝트에 추가됩니다.

- `nsx_use_loadbalancer=true`

(선택 사항) 기본값은 true입니다. OpenShift 경로 및 LoadBalancer 유형의 서비스에 대해 NSX-T 로드 밸런서를 사용하지 않으려면 false로 설정합니다.

- `nsx_lb_service_size='SMALL'`

(선택 사항) 기본값은 SMALL입니다. NSX Edge 크기에 따라 MEDIUM 또는 LARGE도 가능합니다.

- `nsx_no_snat_ip_block='router_ip_block_for_my_ocp_cluster'`

(선택 사항) `ncp/no_snat=true` 주석이 프로젝트 또는 네임스페이스에 적용되는 경우 서브넷을 이 IP 블록에서 가져오고 관련 SNAT는 없게 됩니다. 이는 라우팅할 수 있어야 합니다.

- `nsx_external_ip_pool='external_pool_for_snat'`

(필수 사항) `nsx_external_ip_pool_lb`가 정의되지 않은 경우 SNAT 및 로드 밸런서에 대한 IP 풀입니다.

- `nsx_external_ip_pool_lb='my_ip_pool_for_lb'`

(선택 사항) Router 및 SvcTypeLB에 대한 고유 IP 풀을 원하는 경우 이를 설정합니다.

- `nsx_top_fw_section='top_section'`

(선택 사항) Kubernetes 네트워크 정책 규칙이 NSX-T 방화벽 규칙으로 변환되고 이 섹션 아래에 배치됩니다.

- `nsx_bottom_fw_section='bottom_section'`

(선택 사항) Kubernetes 네트워크 정책 규칙이 NSX-T 방화벽 규칙으로 변환되고 이 섹션 위에 배치됩니다.

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

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

(선택 사항) 설정하는 경우 `nsx_api_user` 및 `nsx_api_password`가 무시됩니다. 인증서를 NSX-T에 업로드해야 하며 이 인증서를 사용하여 인증하는 주체 ID 사용자를 수동으로 생성해야 합니다.

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

- `nsx_lb_default_key='/path/to/key/nsx.key'`

(선택 사항) TLS 기반 경로에 대한 SNI를 생성할 수 있으려면 NSX-T 로드 밸런서에 기본 인증서가 필요합니다. 이 인증서는 구성된 경로가 없는 경우에만 제공됩니다. 제공되지 않는 경우 자체 서명된 인증서가 생성됩니다.

## 샘플 Ansible 호스트 파일

```
[OSEv3:children]
masters
nodes
etcd

[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=origin

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

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

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

[masters]
ocp-master.corp.local

[etcd]
ocp-master.corp.local

[nodes]
ocp-master.corp.local ansible_ssh_host=10.1.0.10 openshift_node_group_name='node-config-master'
ocp-node1.corp.local ansible_ssh_host=10.1.0.11 openshift_node_group_name='node-config-infra'
ocp-node2.corp.local ansible_ssh_host=10.1.0.12 openshift_node_group_name='node-config-infra'
ocp-node3.corp.local ansible_ssh_host=10.1.0.13 openshift_node_group_name='node-config-compute'
ocp-node4.corp.local ansible_ssh_host=10.1.0.14 openshift_node_group_name='node-config-compute'
```

## 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.

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
```

Kubernetes LoadBalancer 서비스의 경우 글로벌 계층 4 지속성이 꺼져 있는 경우 (l4\_persistence가 <None>으로 설정되어 있음) 서비스 규격에서 sessionAffinity를 지정하여 서비스의 지속성 동작을 구성할 수 있습니다. l4\_persistence를 source\_ip로 설정한 경우 서비스 규격의 sessionAffinity를 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 None여야 합니다(이 경우 기타 설정은 source\_ip 또는 cookie일 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
```

## 라우터 샤딩

NCP는 항상 TLS Edge 종료 및 HTTP 경로를 처리하고 해당 네임스페이스 또는 네임스페이스 레이블에 관계없이 TLS 패스스루 경로 및 TLS 다시 암호화 경로를 건너뛸 수 있습니다. TLS 다시 암호화 및 패스스루 경로만 처리하도록 OpenShift 라우터를 제한하려면 다음과 같은 단계를 수행해야 합니다.

- OpenShift 라우터에 네임스페이스 레이블 선택기를 추가합니다.
- 대상 네임스페이스에 네임스페이스 레이블을 추가합니다.

- 대상 네임스페이스에서 TLS 다시 암호화/패스스루 경로를 생성합니다.

예를 들어 네임스페이스 레이블 선택기를 사용하여 라우터를 구성하려면 다음 명령을 실행합니다(라우터의 서비스 계정 이름이 router라고 가정함).

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

이제 라우터는 선택된 네임스페이스에서 경로를 처리합니다. 이 선택기가 네임스페이스와 일치하도록하려면 다음 명령을 실행합니다(네임스페이스가 ns1이라고 가정함).

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

## 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 Data Center 관리 가이드를 참조하십시오.  
  
로드 밸런서가 생성된 후 로드 밸런서 크기는 구성 파일을 업데이트하여 변경할 수 없습니다. 대신 UI 또는 API를 통해 변경할 수 있습니다.
- 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.

## 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=Palto/0=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 Container Plug-in 관리

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.

- [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 > 포트**로 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 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 5-1. 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 &lt;watcher-name&gt;</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 &lt;project-name&gt;</code>
캐시	<code>get project-cache &lt;project-name&gt;</code>
캐시	<code>get project-caches</code>
캐시	<code>get namespace-cache &lt;namespace-name&gt;</code>
캐시	<code>get namespace-caches</code>
캐시	<code>get pod-cache &lt;pod-name&gt;</code>
캐시	<code>get pod-caches</code>
캐시	<code>get ingress-caches</code>
캐시	<code>get ingress-cache &lt;ingress-name&gt;</code>
캐시	<code>get ingress-controllers</code>
캐시	<code>get ingress-controller &lt;ingress-controller-name&gt;</code>

**표 5-1. 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 5-2. 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 5-3. 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 :

```
kubecall> 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 :

```
kubecall> 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 :

```
kubecall> 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