NSX Container Plug-in for OpenShift - Installation and Administration Guide

VMware NSX Container Plug-in 2.4

VMware NSX-T Data Center 2.4

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

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

Contents

NSX-T Container Plug-in for OpenShift - Installation and Administration Guide

This guide describes how to install and administer NSX Container Plug-in (NCP) to provide integration between NSX-T Data Center and OpenShift.

Intended Audience

This guide is intended for system and network administrators. A familiarity with the installation and administration of NSX-T Data Center and OpenShift is assumed.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to http://www.vmware.com/support/pubs.

Overview of NSX-T Container Plug-in

1

NSX Container Plug-in (NCP) provides integration between NSX-T Data Center and container orchestrators such as Kubernetes, as well as integration between NSX-T Data Center and container-based PaaS (platform as a service) software products such as OpenShift. This guide describes setting up NCP with OpenShift.

The main component of NCP runs in a container and communicates with NSX Manager and with the OpenShift control plane. NCP monitors changes to containers and other resources and manages networking resources such as logical ports, switches, routers, and security groups for the containers by calling the NSX API.

The NSX CNI plug-in runs on each OpenShift node. It monitors container life cycle events, connects a container interface to the guest vSwitch, and programs the guest vSwitch to tag and forward container traffic between the container interfaces and the VNIC.

NCP provides the following functionalities:

- Automatically creates an NSX-T logical topology for a OpenShift cluster, and creates a separate logical network for each OpenShift namespace.
- Connects OpenShift pods to the logical network, and allocates IP and MAC addresses.
- Supports network address translation (NAT) and allocates a separate SNAT IP for each OpenShift namespace.

  Note When configuring NAT, the total number of translated IPs cannot exceed 1000.

- Implements OpenShift network policies with NSX-T distributed firewall.
  - Support for ingress and egress network policies.
  - Support for `IPBlock` selector in network policies.
  - Support for `matchLabels` and `matchExpression` when specifying label selectors for network policies.
- Implements OpenShift route with NSX-T layer 7 load balancer.
  - Support for HTTP route and HTTPS route with TLS edge termination.
  - Support for routes with alternate backends and wildcard subdomains.

- Creates tags on the NSX-T logical switch port for the namespace, pod name, and labels of a pod, and allows the administrator to define NSX-T Data Center security groups and policies based on the tags.

In this release, NCP supports a single OpenShift cluster.

This chapter includes the following topics:

Compatibility Requirements

NSX Container Plug-in (NCP) has the following compatibility requirements.

Software Product	Version
NSX-T Data Center	2.3, 2.4
Hypervisor for Container Host VMs	■ Supported vSphere version ■ RHEL KVM 7.4, 7.5, 7.6
Container Host Operating System	RHEL 7.4, 7.5, 7.6
Platform as a Service	OpenShift 3.10, 3.11
Container Host Open vSwitch	2.10.2 (packaged with NSX-T Data Center 2.4)

Installation Overview

Installing and configuring NCP involves the following steps. To perform the steps successfully, you must be familiar with NSX-T Data Center and OpenShift installation and administration.

1 Install NSX-T Data Center.

2 Create an overlay transport zone.

3 Create an overlay logical switch and connect the nodes to the switch.

4 Create a tier-0 logical router.

5 Create IP blocks for the pods.

6 Create IP pools for SNAT (source network address translation).

7 Prepare the Ansible hosts file.

8 Install NCP and OpenShift using a single playbook.

Upgrade NCP

This section describes how to upgrade NCP to 2.4.0.

Procedure

1. Upgrade the CNI RPM package, NSX Node Agent DaemonSet, and NCP ReplicationController.
2. Prepare the Ansible Hosts file.

   Each node must have the parameter `openshift_node_group_name` specified. For example,

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

3. (Optional) Configure load balancing.

   Add a step to specify a different IP pool for external IP addresses for the LoadBalancer service. For example,

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

Setting Up NSX-T Resources

2

NSX-T Data Center resources must be created to provide networking to OpenShift nodes.

This chapter includes the following topics:

Configuring NSX-T Resources

NSX-T Data Center resources that you need to configure include an overlay transport zone, a tier-0 logical router, a logical switch to connect the node VMs, IP blocks for Kubernetes nodes, and an IP pool for SNAT.

Important If you are running with NSX-T Data Center 2.4 or later, you must configure NSX-T resources using the **Advanced Networking & Security** tab.

In the NCP configuration file `ncp.ini`, the NSX-T Data Center resources are specified using their UUIDs or names.

Overlay Transport Zone

Log in to NSX Manager and find the overlay transport zone that is used for container networking or create a new one.

Specify an overlay transport zone for a cluster by setting the `overlay_tz` option in the `[nsx_v3]` section of `ncp.ini`. This step is optional. If you do not set `overlay_tz`, NCP will automatically retrieve the overlay transport zone ID from the tier-0 router.

Tier-0 Logical Routing

Log in to NSX Manager and find the router that is used for container networking or create a new one.

Specify a tier-0 logical router for a cluster by setting the `tier0_router` option in the `[nsx_v3]` section of `ncp.ini`.

Note The router must be created in active-standby mode.

Logical Switch

The vNICs used by the node for data traffic must be connected to an overlay logical switch. It is not mandatory for the node's management interface to be connected to NSX-T Data Center, although doing so will make setting up easier. You can create a logical switch by logging in to NSX Manager. On the switch, create logical ports and attach the node vNICs to them. The logical ports must have the following tags:

- tag: `<cluster_name>`, scope: `ncp/cluster`
- tag: `<node_name>`, scope: `ncp/node_name`

The `<cluster_name>` value must match the value of the `cluster` option in the `[coe]` section in `ncp.ini`.

IP Blocks for Kubernetes Pods

Log in to NSX Manager and create one or more IP blocks. Specify the IP block in CIDR format.

Specify IP blocks for Kubernetes pods by setting the `container_ip_blocks` option in the `[nsx_v3]` section of `ncp.ini`.

You can also create IP blocks specifically for no-SNAT namespaces.

Specify no-SNAT IP blocks by setting the `no_snat_ip_blocks` option in the `[nsx_v3]` section of `ncp.ini`.

If you create no-SNAT IP blocks while NCP is running, you must restart NCP. Otherwise, NCP will keep using the shared IP blocks until they are exhausted.

Note When you create an IP block, the prefix must not be larger than the value of the parameter `subnet_prefix` in NCP's configuration file `ncp.ini`.

IP Pool for SNAT

The IP pool is used for allocating IP addresses which will be used for translating pod IPs via SNAT rules, and for exposing ingress controllers via SNAT/DNAT rules, just like Openstack floating IPs. These IP addresses are also referred to as external IPs.

Multiple Kubernetes clusters use the same external IP pool. Each NCP instance uses a subset of this pool for the Kubernetes cluster that it manages. By default, the same subnet prefix for pod subnets will be used. To use a different subnet size, update the `external_subnet_prefix` option in the `[nsx_v3]` section in `ncp.ini`.

Log in to NSX Manager and create a pool or find an existing pool.

Specify IP pools for SNAT by setting the `external_ip_pools` option in the `[nsx_v3]` section of `ncp.ini`.

You can also configure SNAT for a specific service by adding an annotation to the service. For example,

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

NCP will configure the SNAT rule for this service. The rule's source IP is the set of backend pods. The destination IP is the SNAT IP allocated from the specified external IP pool. Note the following:

- The IP pool specified by `ncp/snat_pool` should already exist in NSX-T Data Center before the service is configured. The IP pool must have the tag `scope: ncp/owner, tag: cluster:<cluster_name>`.
- In NSX-T Data Center, the priority of the SNAT rule for the service is higher than that for the project.
- If a pod is configured with multiple SNAT rules, only one will work.

You can specify which namespace can be allocated IPs from the SNAT IP pool by adding the following tag to the IP pool.

- `scope: ncp/owner, tag: ns:<namespace_UUID>`

You can get the namespace UUID with one of the following command:

```
oc get ns –o yaml
```

Note the following:

- Each tag should specify one UUID. You can create multiple tags for the same pool.
- If you change the tags after some namespaces have been allocated IPs based on the old tags, those IPs will not be reclaimed until the SNAT configurations of the services change or NCP restarts..
- The namespace owner tag is optional. Without this tag, any namespace can have IPs allocated from the SNAT IP pool.

(Optional) Firewall Marker Sections

To allow the administrator to create firewall rules and not have them interfere with NCP-created firewall sections based on network policies, log in to NSX Manager and create two firewall sections.

Specify marker firewall sections by setting the `bottom_firewall_section_marker` and `top_firewall_section_marker` options in the `[nsx_v3]` section of `ncp.ini`.

The bottom firewall section must be below the top firewall section. With these firewall sections created, all firewall sections created by NCP for isolation will be created above the bottom firewall section, and all firewall sections created by NCP for policy will be created below the top firewall section. If these marker sections are not created, all isolation rules will be created at the bottom, and all policy sections will be created at the top. Multiple marker firewall sections with the same value per cluster are not supported and will cause an error.

Installing NCP

3

NCP is fully integrated with OpenShift. When you add the needed parameters in the Ansible hosts file and install OpenShift, NCP is automatically installed.

This chapter includes the following topics:

System Requirements

Before installing OpenShift, ensure that your environment meets certain requirements.

General Requirements

- Ansible 2.4 or later.

Virtual Machine Requirements

Openshift node VMs must have two vNICs:

- A management vNIC connected to the logical switch that has an uplink to the management tier-1 router.
- The second vNIC on all VMs must have the following tags in NSX-T so that NCP knows which port is used as a parent VIF for all PODs running on the particular OenShift node.

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

Bare Metal Machine Requirements

- The OpenShift nodes must be NSX-T transport nodes and the tags mentioned above must be applied on the transport nodes instead of VIFs.
- The Ansible hosts file must have this setting: `nsx_node_type='BAREMETAL'`.

NSX-T Requirements

- A tier-0 router.

- An overlay transport zone.
- An IP block for POD networking.
- (Optional) An IP Block for routed (no NAT) POD networking.
- An IP Pool for SNAT. By default the IP Block for POD networking is routable only inside NSX-T. NCP uses this IP Pool to provide connectivity to the outside.
- (Optional) Top and bottom firewall sections. NCP will place Kubernetes network policy rules between these two sections.
- Open vSwitch and CNI plugin RPMs must be hosted on an HTTP server reachable from the OpenShift node VMs.

NCP Docker Image

Currently the NCP docker image is not publically available. You must have the image `nsx-ncp` in a local private registry, or do the following:

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

On all nodes:

```
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
```

Preparing the Ansible Hosts File

You must specify NCP parameters in the Ansible hosts file for NCP to be integrated with OpenShift.

After you specify the following parameters in the Ansible hosts file, installing OpenShift will nstall NCP automatically.

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

  (Required) This is required because multiple Openshift/Kubernetes clusters can connect to the same NSX Manager.
- `nsx_api_managers='10.10.10.10'`

  (Required) IP address of NSX Manager. For an NSX Manager cluster, specify comma-separated IP addresses.
- `nsx_tier0_router='MyT0Router'`

  (Required) Name or UUID of the tier-0 router that the project's tier-1 routers will connect to.
- `nsx_overlay_transport_zone='my_overlay_tz'`

(Required) Name or UUID of the overlay transport zone that will be used to create logical switches.

- `nsx_container_ip_block='ip_block_for_my_ocp_cluster'`

  (Required) Name or UUID of an IP block configured on NSX-T. There will be a subnet per project out of this IP block. These networks will be behind SNAT and not routable.

- `nsx_ovs_uplink_port='ens224'`

  (Required) If in `HOSTVM` mode. NSX-T needs second a vNIC for POD networking on the OCP nodes, different from the management vNIC. It is highly recomended that both vNICs be connected to NSX-T logical switches. The second (non-management) vNIC must be supplied here. For bare meetal, this parameter is not needed.

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

  (Required) Temporary requirement until NCP can bootstrap the nodes. We need to place `nsx-cni` on an http server.

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

  (Required) Temporary parameters until NCP can bootstrap the nodes. Can be ignored in bare metal setup.

- `nsx_node_type='HOSTVM'`

  (Optional) Defaults to `HOSTVM`. Set to `BAREMETAL` if OpenShift is not running in VMs.

- `nsx_k8s_api_ip=192.168.10.10`

  (Optional) If set, NCP will talk to this IP address, otherwise to Kubernetes service IP.

- `nsx_k8s_api_port=192.168.10.10`

  (Optional) Default to 443 for Kubernetes service. Set to 8443 if you use it in combination with `nsx_k8s_api_ip` to specify master node IP.

- `nsx_insecure_ssl=true`

  (Optional) Default is `true` as NSX Manager comes with untrusted certificate. If you have changed the certificate with a trusted one you can set it to `false`.

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

  (Optional) Defaults to 24. This is the subnet size that will be dedicated per Openshift project. If the number of PODs exceeds the subnet size a new logical switch with the same subnet size will be added to the project.

- `nsx_use_loadbalancer=true`

(Optional) Defaults to `true`. Set to `false` if you do not want to use NSX-T load balancers for OpenShift routes and services of type LoadBalancer.

- `nsx_lb_service_size='SMALL'`

  (Optional) Defaults to `SMALL`. Depending on the NSX Edge size `MEDIUM` or `LARGE` is also possible.

- `nsx_no_snat_ip_block='router_ip_block_for_my_ocp_cluster'`

  (Optional) If the `ncp/no_snat=true` annotation is applied on a project or namespace the subnet will be taken from this IP block and there will be no SNAT for it. It is expected to be routable.

- `nsx_external_ip_pool='external_pool_for_snat'`

  (Requred) IP pool for SNAT and load balancer if `nsx_external_ip_pool_lb` is not defined.

- `nsx_external_ip_pool_lb='my_ip_pool_for_lb'`

  (Optional) Set this if you want a distinct IP pool for `Router` and `SvcTypeLB`.

- `nsx_top_fw_section='top_section'`

  (Optional) Kubernetes network policy rules will be translated to NSX-T firewall rules and placed below this section.

- `nsx_bottom_fw_section='bottom_section'`

  (Optional) Kubernetes network policy rules will be translated to NSX-T firewall rules and placed above this section.

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

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

  (Optional) If set, `nsx_api_user` and `nsx_api_password` will be ignored. The certificate must be uploaded to NSX-T and a Principal Identity user authenticating with this certificate must be manually created.

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

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

  (Optional) NSX-T load balancer requires a default certificate in order to be able to crate SNIs for TLS based Routes. This certificate will be presented only if there is no Route configured. If not provided, a self-signed certificate will be generated.

Sample Ansible Hosts File

```
[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'
```

Load Balancing

4

The NSX-T Data Center load balancer is integrated with OpenShift and acts as the OpenShift Router..

NCP watches OpenShift route and endpoint events and configures load balancing rules on the load balancer based on the route specification. As a result, the NSX-T Data Center load balancer will forward incoming layer 7 traffic to the appropriate backend pods based on the rules.

This chapter includes the following topics:

Configuring Load Balancing

Configuring load balancing involves configuring a Kubernetes LoadBalancer service or an OpenShift route. You also need to configure the NCP replication controller. The LoadBalancer service is for layer 4 traffic and the OpenShift route is for layer 7 traffic.

When you configure a Kubernetes LoadBalancer service, it is allocated an IP address from the external IP block that you configure. The load balancer is exposed on this IP address and the service port. You can specify the name or ID of an IP pool using the `loadBalancerIP` spec in the LoadBalancer definition. The Loadbalancer service's IP will be allocated from this IP pool. If the `loadBalancerIP` spec is empty, the IP will be allocated from the external IP block that you configure.

The IP pool specified by `loadBalancerIP` must have the tag `scope: ncp/owner, tag: cluster:<cluster_name>`.

To use the NSX-T Data Center load balancer, you must configure load balancing in NCP. In the `ncp_rc.yml` file, do the following:

1 Set `use_native_loadbalancer` = `True`.

2 Set `pool_algorithm` to `WEIGHTED_ROUND_ROBIN`.

3 Set lb_default_cert_path and lb_priv_key_path to be the full path names of the CA-signed certificate file and the private key file, respectively. See below for a sample script to generate a CA-signed certificate. In addition, mount the default certificate and key into the NCP pod. See below for instructions.

4 (Optional) Specify a persistence setting with the parameters `l4_persistence` and `l7_persistence`. The available option for layer 4 persistence is source IP. The available options for layer 7 persistence are cookie and source IP. The default is `<None>`. For example,

```
# 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 (Optional) Set `service_size` = `SMALL`, `MEDIUM`, or `LARGE`. The default is `SMALL`.

6 If you are running OpenShift 3.11, you must perform the following configuration so that OpenShift will not assign an IP to the LoadBalancer service.

- Set `ingressIPNetworkCIDR` to 0.0.0.0/32 under `networkConfig` in the `/etc/origin/master/master-config.yaml` file.
- Restart the API server and controllers with the following commands:

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

For a Kubernetes LoadBalancer service, you can also specify `sessionAffinity` on the service spec to configure persistence behavior for the service if the global layer 4 persistence is turned off, that is, `l4_persistence` is set to `<None>`. If `l4_persistence` is set to `source_ip`, the `sessionAffinity` on the service spec can be used to customize the persistence timeout for the service. The default layer 4

persistence timeout is 10800 seconds (same as that specified in the Kubernetes documentation for services (https://kubernetes.io/docs/concepts/services-networking/service). All services with default persistence timeout will share the same NSX-T load balancer persistence profile. A dedicated profile will be created for each service with a non-default persistence timeout.

Note If the backend service of an Ingress is a service of type LoadBalancer, then the layer 4 virtual server for the service and the layer 7 virtual server for the Ingress cannot have different persistence settings, for example, `source_ip` for layer 4 and `cookie` for layer 7. In such a scenario, the persistence settings for both virtual servers must be the same (`source_ip`, `cookie`, or `None`), or one of them is `None` (then the other setting can be `source_ip` or `cookie`). An example of such a scenario:

```
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
```

Router Sharding

NCP always handles TLS edge termination and HTTP routes, and skips TLS passthrough routes and TLS re-encrypt routes regardless of their namespaces or namespace labels. To restrict a OpenShift router to handling only TLS re-encrypt and passthrough routes, you must perform the following steps:

- Add a namespace label selector to the Openshift router.
- Add a namespace label to the target namespace.
- Create TLS re-encrypt/passthrough routes in the target namespace.

For example, to configure a router with a namespace label selector, run the following command (assuming the router's service account name is `router`):

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

The router will now handle routes from the selected namespaces. To make this selector match a namespace, run the following command (assuming the namespace is called `ns1`):

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

Layer 7 Load Balancer Example

The following YAML file configures two replication controllers (tea-rc and coffee-rc), two services (tea-svc and coffee-svc), and two routes (cafe-route-multi and cafe-route) to provide layer 7 load balancing.

```
# 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
```

Additional Notes

- Only edge termination is supported for HTTPS traffic.
- Wildcard subdomain is supported. For example, if `wildcardPolicy` is set to **`Subdomain`**, and the host name is set to **`wildcard.example.com`**, any request to **`*.example.com`** will be serviced.
- If NCP throws an error during the processing of a Route event due to misconfiguration, you need to correct the Route YAML file, delete and recreate the Route resource.
- NCP does not enforce hostname ownership by namespaces.
- One Loadbalancer service is supported per Kubernetes cluster.
- NSX-T Data Center will create a layer 4 load balancer virtual server and pool for each LoadBalancer service port. Both TCP and UDP are supported.
- The NSX-T Data Center load balancer comes in different sizes. For information about configuring an NSX-T Data Center load balancer, see the *NSX-T Data Center Administration Guide*.

  After the load balancer is created, the load balancer size cannot be changed by updating the configuration file. It can be changed through the UI or API.
- Automatic scaling of the layer 4 load balancer is supported. If a Kubernetes LoadBalancer service is created or modified so that it requires additional virtual servers and the existing layer 4 load balancer does not have the capacity, a new layer 4 load balancer will be created. NCP will also delete a layer 4 load balancer that no longer has virtual servers attached. This feature is enabled by default. You can disable it by setting `l4_lb_auto_scaling` to **`false`** in the NCP ConfigMap.

Sample Script to Generate a CA-Signed Certificate

The script below generates a CA-signed certificate and a private key stored in the files <filename>.crt and <finename>.key, respectively. The `genrsa` command generates a CA key. The CA key should be encrypted. You can specify an encryption method with the command such as `aes256`.

```
#!/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
```

Mount the Default Certificate and Key into the NCP Pod

After the certificate and private key have been generated, place them in the directory `/etc/nsx-ujo` on the host VM. Assuming the certificate and key files are named `lb-default.crt` and `lb-default.key`, respectively, edit `ncp-rc.yaml` so that these files on the host are mounted into the pod. For example,

```
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
```

Administering NSX Container Plug-in

5

You can administer NSX Container Plug-in from the NSX Manager GUI or from the command-line interface (CLI).

Note If a container host VM is running on ESXi 6.5 and the VM is migrated through vMotion to another ESXi 6.5 host, containers running on the container host will lose connectivity to containers running on other container hosts. You can resolve the problem by disconnecting and connecting the vNIC of the container host. This issue does not occur with ESXi 6.5 Update 1 or later.

Hyperbus reserves VLAN ID 4094 on the hypervisor for PVLAN configuration and this ID cannot be changed. To avoid any VLAN conflict, do not configure VLAN logical switches or VTEP vmknics with the same VLAN ID.

This chapter includes the following topics:

Manage IP Blocks from the NSX Manager GUI

You can add, delete, edit, view details of, and manage the tags for an IP block from the NSX Manager GUI.

Procedure

1. From a browser, log in to the NSX Manager at `https://<nsx-manager-IP-address-or-domain-name>`.
2. Navigate to **Networking > IPAM**.

   A list of the existing IP blocks is displayed.

3 Perform any of the following actions.

Option	Action
Add an IP block	Click **ADD**.
Delete one or more IP blocks	Select one or more IP blocks and click **DELETE**.
Edit an IP block	Select an IP block and click **EDIT**.
View details about an IP block	Click the IP block name. Click the **Overview** tab to see general information. Click the **Subnets** tab to see this IP block's subnets.
Manage tags for an IP block	Select an IP block and click **ACTIONS > Manage Tags**.

You cannot delete an IP block that has subnets allocated.

View IP Block Subnets from the NSX Manager GUI

You can view the subnets for an IP block from the NSX Manager GUI. Adding or deleting IP block subnets after NCP is installed and running is not recommended.

Procedure

1 From a browser, log in to the NSX Manager at `https://<nsx-manager-IP-address-or-domain-name>`.

2 Navigate to **Networking > IPAM**.

A list of the existing IP blocks is displayed.

3 Click an IP block name

4 Click the **Subnets** tab.

CIF-Attached Logical Ports

CIFs (container interfaces) are network interfaces on containers that are connected to logical ports on a switch. These ports are called CIF-attached logical ports.

You can manage CIF-attached logical ports from the NSX Manager GUI.

Managing CIF-Attached Logical Ports

Navigate to **Networking > Switching > Ports** to see all logical ports, including CIF-attached logical ports. Click the attachment link of a CIF-attached logical port to see the attachment information. Click the logical port link to open a window pane with four tabs: Overview, Monitor, Manage, and Related. Clicking **Related > Logical Ports** shows the related logical port on an uplink switch. For more information about switch ports, see the *NSX-T Administration Guide*.

Network Monitoring Tools

The following tools support CIF-attached logical ports. For more information about these tools, see the *NSX-T Administration Guide*.

- Traceflow
- Port Connection
- IPFIX
- Remote port mirroring using GRE encapsulation of a logical switch port that connects to a container is supported. For more information, see "Understanding Port Mirroring Switching Profile" in the *NSX-T Administration Guide*. However, port mirroring of the CIF to VIF port is not supported via the manager UI.

CLI Commands

To run CLI commands, log in to the NSX Container Plug-in container, open a terminal and run the `nsxcli` command.

You can also get the CLI prompt by running the following command on a node:

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

Table 5-1. CLI Commands for the NCP Container

Type	Command
Status	get ncp-master status
Status	get ncp-nsx status
Status	get ncp-watcher <watcher-name>
Status	get ncp-watchers
Status	get ncp-k8s-api-server status
Status	check projects
Status	check project <project-name>
Cache	get project-cache <project-name>
Cache	get project-caches
Cache	get namespace-cache <namespace-name>
Cache	get namespace-caches
Cache	get pod-cache <pod-name>
Cache	get pod-caches
Cache	get ingress-caches
Cache	get ingress-cache <ingress-name>
Cache	get ingress-controllers
Cache	get ingress-controller <ingress-controller-name>

Table 5-1. CLI Commands for the NCP Container (continued)

Type	Command
Cache	get network-policy-caches
Cache	get network-policy-cache <pod-name>
Support	get ncp-log file <filename>
Support	get ncp-log-level
Support	set ncp-log-level <log-level>
Support	get support-bundle file <filename>
Support	get node-agent-log file <filename>
Support	get node-agent-log file <filename> <node-name>

Table 5-2. CLI Commands for the NSX Node Agent Container

Type	Command
Status	get node-agent-hyperbus status
Cache	get container-cache <container-name>
Cache	get container-caches

Table 5-3. CLI Commands for the NSX Kube Proxy Container

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

Status Commands for the NCP Container

- Show the status of the NCP master

  ```
  get ncp-master status
  ```

  Example:

  ```
  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
  ```

- Show the connection status between NCP and NSX Manager

  ```
  get ncp-nsx status
  ```

Example:

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

- Show the watcher status for ingress, namespace, pod, and service

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

Example 1:

```
kubenode> get ncp-watcher pod
    Average event processing time: 1174 msec (in past 3600-sec window)
    Current watcher started time: Mar 02 2017 10:47:35 PST
    Number of events processed: 1 (in past 3600-sec window)
    Total events processed by current watcher: 1
    Total events processed since watcher thread created: 1
    Total watcher recycle count: 0
    Watcher thread created time: Mar 02 2017 10:47:35 PST
    Watcher thread status: Up
```

Example 2:

```
kubenode> get ncp-watchers
    pod:
        Average event processing time: 1145 msec (in past 3600-sec window)
        Current watcher started time: Mar 02 2017 10:51:37 PST
        Number of events processed: 1 (in past 3600-sec window)
        Total events processed by current watcher: 1
        Total events processed since watcher thread created: 1
        Total watcher recycle count: 0
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up

    namespace:
        Average event processing time: 68 msec (in past 3600-sec window)
        Current watcher started time: Mar 02 2017 10:51:37 PST
        Number of events processed: 2 (in past 3600-sec window)
        Total events processed by current watcher: 2
        Total events processed since watcher thread created: 2
        Total watcher recycle count: 0
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up

    ingress:
        Average event processing time: 0 msec (in past 3600-sec window)
        Current watcher started time: Mar 02 2017 10:51:37 PST
        Number of events processed: 0 (in past 3600-sec window)
        Total events processed by current watcher: 0
        Total events processed since watcher thread created: 0
        Total watcher recycle count: 0
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up
```

```
    service:
        Average event processing time: 3 msec (in past 3600-sec window)
        Current watcher started time: Mar 02 2017 10:51:37 PST
        Number of events processed: 1 (in past 3600-sec window)
        Total events processed by current watcher: 1
        Total events processed since watcher thread created: 1
        Total watcher recycle count: 0
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up
```

- Show the connection status between NCP and Kubernetes API server

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

Example:

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

- Check all projects or a specific one

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

Example:

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

    ns1:
        Router 8accc9cd-9883-45f6-81b3-0d1fb2583180 is missing

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

Cache Commands for the NCP Container

- Get the internal cache for projects or namespaces

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

Example:

```
kubenode> get project-caches
    default:
        logical-router: 8accc9cd-9883-45f6-81b3-0d1fb2583180
```

```
        logical-switch:
            id: 9d7da647-27b6-47cf-9cdb-6e4f4d5a356d
            ip_pool_id: 519ff57f-061f-4009-8d92-3e6526e7c17e
            subnet: 10.0.0.0/24
            subnet_id: f75fd64c-c7b0-4b42-9681-fc656ae5e435

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

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

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

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

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

    testns:
        ext_pool_id: 346a0f36-7b5a-4ecc-ad32-338dcb92316f
```

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

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

- Get the internal cache for pods

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

Example:

```
kubenode> get pod-caches
    nsx.default.nginx-rc-uq2lv:
        cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
        gateway_ip: 10.0.0.1
        host_vif: d6210773-5c07-4817-98db-451bd1f01937
        id: 1c8b5c52-3795-11e8-ab42-005056b198fb
        ingress_controller: False
        ip: 10.0.0.2/24
        labels:
            app: nginx
        mac: 02:50:56:00:08:00
        port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
        vlan: 1

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

```
kubenode> get pod-cache nsx.default.nginx-rc-uq2lv
    cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
    gateway_ip: 10.0.0.1
    host_vif: d6210773-5c07-4817-98db-451bd1f01937
    id: 1c8b5c52-3795-11e8-ab42-005056b198fb
    ingress_controller: False
    ip: 10.0.0.2/24
    labels:
        app: nginx
    mac: 02:50:56:00:08:00
    port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
    vlan: 1
```

- Get network policy caches or a specific one

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

Example:

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

```
        src_rules_hash: e4ea7b8d91c1e722670a59f971f8fcc1a5ac51f1


kubenode> get network-policy-cache nsx.testns.allow-tcp-80
    dest_labels: None
    dest_pods:
        50.0.2.3
    match_expressions:
        key: tier
        operator: In
        values:
            cache
    name: allow-tcp-80
    np_dest_ip_set_ids:
        22f82d76-004f-4d12-9504-ce1cb9c8aa00
        np_except_ip_set_ids:
    np_ip_set_ids:
        14f7f825-f1a0-408f-bbd9-bb2f75d44666
    np_isol_section_id: c8d93597-9066-42e3-991c-c550c46b2270
    np_section_id: 04693136-7925-44f2-8616-d809d02cd2a9
    ns_name: testns
    src_egress_rules: None
    src_egress_rules_hash: 97d170e1550eee4afc0af065b78cda302a97674c
    src_pods:
        50.0.2.0/24
    src_rules:
        from:
            namespaceSelector:
                matchExpressions:
                    key: tier
                    operator: DoesNotExist
                matchLabels:
                    ns: myns
        ports:
            port: 80
            protocol: TCP
    src_rules_hash: e4ea7b8d91c1e722670a59f971f8fcc1a5ac51f1
```

Support Commands for the NCP Container

- Save the NCP support bundle in the filestore

  The support bundle consists of the log files for all the containers in pods with the label **`tier:nsx-networking`**. The bundle file is in the tgz format and saved in the CLI default filestore directory `/var/vmware/nsx/file-store`. You can use the CLI file-store command to copy the bundle file to a remote site.

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

Example:

```
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
```

- Save the NCP logs in the filestore

  The log file is saved in the tgz format in the CLI default filestore directory `/var/vmware/nsx/file-store`. You can use the CLI file-store command to copy the bundle file to a remote site.

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

  Example:

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

- Save the node agent logs in the filestore

  Save the node agent logs from one node or all the nodes. The logs are saved in the tgz format in the CLI default filestore directory `/var/vmware/nsx/file-store`. You can use the CLI file-store command to copy the bundle file to a remote site.

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

  Example:

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

- Get and set the log level

  The available log levels are `NOTSET`, `DEBUG`, `INFO`, `WARNING`, `ERROR`, and `CRITICAL`.

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

  Example:

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

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

Status Commands for the NSX Node Agent Container

- Show the connection status between the node agent and HyperBus on this node.

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

Example:

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

Cache Commands for the NSX Node Agent Container

- Get the internal cache for NSX node agent containers.

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

Example 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
```

Example 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
```

Status Commands for the NSX Kube-Proxy Container

- Show the connection status between Kube Proxy and Kubernetes API Server

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

Example:

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

- Show the Kube Proxy watcher status

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

Example 1:

```
kubenode> get kube-proxy-watcher endpoint
    Average event processing time: 15 msec (in past 3600-sec window)
    Current watcher started time: May 01 2017 15:06:24 PDT
    Number of events processed: 90 (in past 3600-sec window)
    Total events processed by current watcher: 90
```

```
    Total events processed since watcher thread created: 90
    Total watcher recycle count: 0
    Watcher thread created time: May 01 2017 15:06:24 PDT
    Watcher thread status: Up
```

Example 2:

```
kubenode> get kube-proxy-watchers
    endpoint:
        Average event processing time: 15 msec (in past 3600-sec window)
        Current watcher started time: May 01 2017 15:06:24 PDT
        Number of events processed: 90 (in past 3600-sec window)
        Total events processed by current watcher: 90
        Total events processed since watcher thread created: 90
        Total watcher recycle count: 0
        Watcher thread created time: May 01 2017 15:06:24 PDT
        Watcher thread status: Up

     service:
        Average event processing time: 8 msec (in past 3600-sec window)
        Current watcher started time: May 01 2017 15:06:24 PDT
        Number of events processed: 2 (in past 3600-sec window)
        Total events processed by current watcher: 2
        Total events processed since watcher thread created: 2
        Total watcher recycle count: 0
        Watcher thread created time: May 01 2017 15:06:24 PDT
        Watcher thread status: Up
```

- Dump OVS flows on a node

```
dump ovs-flows
```

Example:

```
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
```

Error Codes

This section lists error codes produced by the various components.

NCP Error Codes

Error Code	Description
NCP00001	Invalid configuration
NCP00002	Initialization failed
NCP00003	Invalid state
NCP00004	Invalid adapter
NCP00005	Certificate not found
NCP00006	Token not found
NCP00007	Invalid NSX configuration
NCP00008	Invalid NSX tag
NCP00009	NSX connection failed
NCP00010	Node tag not found
NCP00011	Invalid node logical switch port
NCP00012	Parent VIF update failed
NCP00013	VLAN exhausted
NCP00014	VLAN release failed
NCP00015	IP pool exhausted
NCP00016	IP release failed
NCP00017	IP block exhausted
NCP00018	IP subnet creation failed
NCP00019	IP subnet deletion failed
NCP00020	IP pool creation failed
NCP00021	IP pool deletion failed
NCP00022	Logical router creation failed
NCP00023	Logical router update failed
NCP00024	Logical router deletion failed
NCP00025	Logical switch creation failed

Error Code	Description
NCP00026	Logical switch update failed
NCP00027	Logical switch deletion failed
NCP00028	Logical router port creation failed
NCP00029	Logical router port deletion failed
NCP00030	Logical switch port creation failed
NCP00031	Logical switch port update failed
NCP00032	Logical switch port deletion failed

Error Code	Description
NCP00033	Network policy not found
NCP00034	Firewall creation failed
NCP00035	Firewall read failed
NCP00036	Firewall update failed
NCP00037	Firewall deletion failed
NCP00038	Multiple firewall found
NCP00039	NSGroup creation failed
NCP00040	NSGroup deletion failed
NCP00041	IP set creation failed
NCP00042	IP set update failed
NCP00043	IP set deletion failed
NCP00044	SNAT rule creation failed
NCP00045	SNAT rule deletion failed
NCP00046	Adapter API connection failed
NCP00047	Adapter watcher exception
NCP00048	Load balancer service deletion failed
NCP00049	Load balancer virtual server creation failed
NCP00050	Load balancer virtual server update failed

Error Code	Description
NCP00051	Load balancer virtual server deletion failed
NCP00052	Load balancer pool creation failed
NCP00053	Load balancer pool update failed
NCP00054	Load balancer pool deletion failed
NCP00055	Load balancer rule creation failed
NCP00056	Load balancer rule update failed
NCP00057	Load balancer rule deletion failed
NCP00058	Load balancer pool IP release failed
NCP00059	Load balancer virtual server and service association not found
NCP00060	NSGroup update failed
NCP00061	Firewall rules get failed
NCP00062	NSGroup no criteria
NCP00063	Node VM not found
NCP00064	Node VIF not found
NCP00065	Certificate import failed

Error Code	Description
NCP00066	Certificate un-import failed
NCP00067	SSL binding update failed
NCP00068	SSL profile not found
NCP00069	IP pool not found
NCP00070	T0 edge cluster not found
NCP00071	IP pool update failed
NCP00072	Dispatcher failed
NCP00073	NAT rule deletion failed
NCP00074	Logical router port get failed
NCP00075	NSX configuration validation failed

Error Code	Description
NCP00076	SNAT rule update failed
NCP00077	SNAT rule overlapped
NCP00078	Load balancer endpoints add failed
NCP00079	Load balancer endpoints update failed
NCP00080	Load balancer rule pool creation failed
NCP00081	Load balancer virtual server not found
NCP00082	IP set read failed
NCP00083	SNAT pool get failed
NCP00084	Load balancer service creation failed
NCP00085	Load balancer service update failed
NCP00086	Logical router port update failed
NCP00087	Load balancer init failed
NCP00088	IP pool not unique
NCP00089	Layer 7 load balancer cache sync error
NCP00090	Load balancer pool not exist error
NCP00091	Load balancer rule cache init error
NCP00092	SNAT process failed
NCP00093	Load balancer default certificate error
NCP00094	Load balancer endpoint deletion failed
NCP00095	Project not found
NCP00096	Pool access denied
NCP00097	Failed to get a load balancer service

Error Code	Description
NCP00098	Failed to create a load balancer service
NCP00099	Load balancer pool cache synchronization error

NSX Node Agent Error Codes

Error Code	Description
NCP01001	OVS uplink not found
NCP01002	Host MAC not found
NCP01003	OVS port creation failed
NCP01004	No pod configuration
NCP01005	Pod configuration failed
NCP01006	Pod un-configuration failed
NCP01007	CNI socket not found
NCP01008	CNI connection failed
NCP01009	CNI version mismatch
NCP01010	CNI message receive failed
NCP01011	CNI message transmit failed
NCP01012	Hyperbus connection failed
NCP01013	Hyperbus version mismatch
NCP01014	Hyperbus message receive failed
NCP01015	Hyperbus message transmit failed
NCP01016	GARP send failed
NCP01017	Interface configuration failed

nsx-kube-proxy Error Codes

Error Code	Description
NCP02001	Proxy invalid gateway port
NCP02002	Proxy command failed
NCP02003	Proxy validate failed

CLI Error Codes

Error Code	Description
NCP03001	CLI start failed
NCP03002	CLI socket create failed
NCP03003	CLI socket exception

Error Code	Description
NCP03004	CLI client invalid request
NCP03005	CLI server transmit failed
NCP03006	CLI server receive failed
NCP03007	CLI command execute failed

Kubernetes Error Codes

Error Code	Description
NCP05001	Kubernetes connection failed
NCP05002	Kubernetes invalid configuration
NCP05003	Kubernetes request failed
NCP05004	Kubernetes key not found
NCP05005	Kubernetes type not found
NCP05006	Kubernetes watcher exception
NCP05007	Kubernetes resource invalid length
NCP05008	Kubernetes resource invalid type
NCP05009	Kubernetes resource handle failed
NCP05010	Kubernetes service handle failed
NCP05011	Kubernetes endpoint handle failed
NCP05012	Kubernetes Ingress handle failed
NCP05013	Kubernetes network policy handle failed
NCP05014	Kubernetes node handle failed
NCP05015	Kubernetes namespace handle failed
NCP05016	Kubernetes pod handle failed
NCP05017	Kubernetes secret handle failed
NCP05018	Kubernetes default backend failed
NCP05019	Kubernetes unsupported match expression
NCP05020	Kubernetes status update failed
NCP05021	Kubernetes annotation update failed
NCP05022	Kubernetes namespace cache not found
NCP05023	Kubernetes secret not found
NCP05024	Kubernetes default backend is in use
NCP05025	Kubernetes LoadBalancer service handle failed

OpenShift Error Codes

Error Code	Description
NCP07001	OC route handle failed
NCP07002	OC route status update failed