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

Modified on 7 SEP 2017
VMware NSX-T Data Center 2.0

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 Kubernetes - Installation and Administration Guide

This guide describes how to install and administer NSX-T Container Plug-in (NCP) to provide integration between NSX-T and Kubernetes, a container-based application environment..

Intended Audience

This guide is intended for system and network administrators. A familiarity with networking and virtualization technology 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-T Container Plug-in (NCP) provides integration between NSX-T and container orchestrators such as Kubernetes, as well as integration between NSX-T and container-based PaaS (platform as a service) software products such as OpenShift. This guide describes setting up NCP with Kubernetes.

The main component of NCP runs in a container and communicates with NSX Manager and with the Kubernetes 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 Kubernetes 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.

In this release, NCP supports a single Kubernetes cluster.

This chapter includes the following topics:

Compatibility Requirements

NSX-T Container Plug-in has the following compatibility requirements related to Kubernetes and Ubuntu.

Software Product	Version
Hypervisor for Container Host VMs	■ ESXi 6.5 ■ ESXi 6.5 Update 1 ■ RHEL KVM 7.3 ■ Ubuntu KVM 16.04
Container Host Operating System	■ RHEL 7.3 ■ Ubuntu 16.04
Container Runtime	Docker 1.12
Container Orchestrator	Kubernetes 1.5, 1.6
Guest vSwitch	OVS 2.6, 2.7

Installation Overview

In an environment with Kubernetes already installed, installing and configuring NCP typically involve the following steps.

1 Install NSX-T.

2 Create an overlay transport zone.

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

4 Create a tier-0 logical router.

5 Create IP blocks for Kubernetes pods.

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

7 Install NSX CNI (container network interface) plug-in on each node.

8 Install OVS (Open vSwitch) on each node.

9 Configure NSX-T networking for Kubernetes nodes.

10 Install NSX node agent as a DaemonSet.

11 Install NCP as a ReplicationController.

12 Mount security certificates in the NCP pod.

Setting Up NSX-T Resources 2

Before installing NSX-T Container Plug-in, you need to set up certain NSX-T resources.

This chapter includes the following topics:

Configuring NSX-T Resources

NSX-T 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 block or pool for SNAT.

Overlay Transport Zone

The overlay Transport Zone for a cluster is identified by the tag `{'ncp/cluster': '<cluster_name>'}`. Log in to NSX Manager and navigate to **Fabric > Transport Zones**. Find the overlay transport zone that is used for container networking, or create a new one. Tag the transport zone with the name of the cluster being configured. Specifically, `<cluster_name>` must match the value of the `cluster` option in the `[coe]` section in `ncp.ini`. You can add more than one tag to the transport zone to make it shared.

Tier-0 Logical Routing

The tier-0 logical router for a cluster is identified by the tag `{'ncp/cluster': '<cluster_name>'}`. Log in to NSX Manager and navigate to **Routing > ROUTERS**. You can either create a new tier-0 logical router for the Kubernetes cluster, or use an existing one. After you identify the router, tag it with `{'ncp/cluster': '<cluster_name>'}`.

The `<cluster_name>` value must match the value of the cluster option in the `[coe]` section in `ncp.ini`. You can add more than one tag to the router to make it shared.

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

Logical Switch

The interface used by the pods for network 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, although doing so will make setting up easier. The switch is identified by the tag `{'ncp/cluster': '<cluster_name>'}`. Log in to NSX Manager and navigate to **Switching > Switches**. You can either create a new switch or use an existing one. After you identify the switch, tag it with `{'ncp/cluster': '<cluster_name>'}`.

The `<cluster_name>` value must match the value of the cluster option in the `[coe]` section in `ncp.ini`. You can add more than one tag to the switch to make it shared.

Connect the node VMs to the switch. For instructions, see "Connecting a VM to a Logical Switch" in the *NSX-T Administration Guide*.

IP Blocks for Kubernetes Pods

Create one or more IP blocks for the Kubernetes pods. You can log in to NSX Manager and navigate to **DDI > IPAM** to create IP blocks. Specify the IP block in CIDR format. Also specify the tag `ncp/cluster` for the block.

You can also create IP blocks specifically for no-SNAT namespaces. These IP blocks require the tag `{'ncp/no_snat': '<cluster_name>'}` in addition to the `ncp/cluster` tag. 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`. For more information, see Configmap for ncp.ini in ncp-rc.yml.

IP Block or IP Pool for SNAT

These resources will be 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. In this guide, these IP addresses are also referred to as *external IPs*. Users can either configure a *global* external IP block or a cluster specific external IP pool.

To set up an external IP block, log in to NSX Manager and navigate to **DDI > IPAM**. Specify a CIDR value with a network address and not a host address. For example, specify 4.3.0.0/16 instead of 4.3.2.1/16. Tag the IP block with the following key and value to indicate that the IP block is for external IP allocation.

```
{'ncp/external': 'true'}
```

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`.

To use a cluster-specific IP pool for allocating external IPs, log in to NSX Manager and navigate to **Inventory > Groups > IP POOL**. Create or use an existing pool. Apply the following tags to the pool.

```
{'ncp/cluster': 'true'}
{'ncp/external': 'true'}
```

(Optional) Firewall Marker Section

To allow the administrator to create firewall rules and not have them interfere with NCP-created firewall sections, log in to NSX Manager, navigate to **Firewall > General** and create an empty firewall section and tag it with `{'ncp/fw_sect_marker': 'true'}`. With this marker firewall section created, all subsequent firewall sections created by NCP for network policies and namespace isolation will be placed above this firewall section, and firewall rules created by the administrator will be placed below this marker firewall section.

If this marker section is not created, all isolation rules will be created at the bottom. Multiple marker firewall sections per cluster is not supported and will cause an error.

Create and Configure a Tier-0 Logical Router

The tier-0 logical router connects the Kubernetes nodes to external networks.

Procedure

1. From a browser, log in to NSX Manager at https://*nsx-manager-ip-address*.
2. Navigate to **Routing > Routers** and click **Add > Tier-0 Router**.
3. Enter a name and optionally a description.
4. Select an existing edge cluster from the drop-down menu to back this tier-0 logical router.
5. Select a high-availability mode.

   Select active-standby.
6. Click **Save**.

   The new logical router appears as a link.
7. Click the logical router link.
8. Click **Routing > Route Redistribution**.
9. Click **Add** to a new redistribution criterion.

   For sources, in a routed (non-NAT) topology, select **NSX Static**. In a NAT topology, select **Tier-0 NAT**.
10. Click **Save**.
11. Click the newly created router.
12. Click **Configuration > Router Ports**

13 Click **Add** to add an uplink port.

14 Select a transport node.

15 Select the logical switch that was previously created.

16 Specify an IP address in your external network.

17 Click **Save**.

The new logical router appears as a link.

Installing NCP and Related Components

3

Installing NSX-T Container Plug-in (NCP) requires installing components on the master and Kubernetes nodes.

This chapter includes the following topics:

Install NSX-T CNI Plug-in

Install NSX-T CNI plug-in on the Kubernetes nodes.

Procedure

1 Download the installation file appropriate to your Linux distribution.

The filename is `nsx-cni-1.0.0.0.0.xxxxxxx-1.x86_64.rpm` or `nsx-cni-1.0.0.0.0.xxxxxxx.deb`, where *xxxxxxx* is the build number.

2 Install the rpm or deb file downloaded in step 1.

The plug-in is installed in `/opt/cni/bin`. The CNI configuration file 10.net.conf is copied to `/etc/cni/net.d`. The rpm will also install the configuration file `/etc/cni/net.d/99-loopback.conf` for the loopback plug-in.

Install and Configure OVS

Install and configure OVS (Open vSwitch) on the minion nodes.

Procedure

1. Download the installation file for your Linux distribution.

   The filenames are `openvswitch-common_2.7.0.6383646-1_amd64.deb`, `openvswitch-datapath-dkms_2.7.0.xxxxxxx-1_all.deb`, and `openvswitch-switch_2.7.0.xxxxxxx-1_amd64.deb`, where *xxxxxxx* is the build number.

2. Install the rpm or deb file downloaded in step 1.

3. For Ubuntu, run the following command to reload the OVS kernel module.

   ```
   service openvswitch-switch force-reload-kmod
   ```

4. Make sure that OVS is running.

   ```
   # service openvswitch-switch status
   ```

5. Create the *br-int* instance if it is not already created.

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

6. Add the network interface (*node-if*) that is attached to the node logical switch to *br-int* .

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

   Run the following command to see what `ofport` is, because if `ofport` 1 is not available, OVS will assign a port that is available.

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

   If `ofport` is not 1, set the `ovs_uplink_port` option in the `nsx_kube_proxy` section of the NSX node agent DaemonSet yaml file accordingly.

7. Make sure that the *br-int* and *node-if link* status is up.

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

8 Update the network configuration file to ensure that the network interface is up after a reboot.

For Ubuntu, update `/etc/network/interfaces` and add the following lines:

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

For RHEL, update `/etc/sysconfig/network-scripts/ifcfg-<node-if>` and add the following line:

```
ONBOOT=yes
```

Configure NSX-T Networking for Kubernetes Nodes

This section describes how to configure NSX-T networking for Kubernetes master and minion nodes.

Each node must have at least two network interfaces. The first is a management interface which might or might not be on the NSX-T fabric. The other interfaces provide networking for the pods, are on the NSX-T fabric, and connected to a logical switch which is referred to as the node logical switch. The management and pod IP addresses must be routable for Kubernetes health check to work. For communication between the management interface and the pods, NCP automatically creates a DFW rule to allow health check and other management traffic. You can see details of this rule in the NSX Manager GUI. This rule should not be changed or deleted.

For each node VM, ensure that the vNIC that is designated for container networking is attached to the node logical switch.

The VIF ID of the vNIC used for container traffic in each node must be known to NSX-T Container Plug-in (NCP). The corresponding logical switch port must be tagged in the following way:

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

You can identify the logical switch port for a node VM by navigating to **Inventory > Virtual Machines** from the NSX Manager GUI.

If the Kubernetes node name changes, you must update the tag `ncp/node_name` and restart NCP. You can use the following command to get the node names:

```
kubectl get nodes
```

If you add a node to a cluster while NCP is running, you must add the tags to the logical switch port before you run the `kubeadm join` command. Otherwise, the new node will not have network connectivity. If the tags are incorrect or missing, you can take the following steps to resolve the issue:

- Apply the correct tags to the logical switch port.
- Restart NCP.

Install NSX Node Agent

The NSX node agent is a DaemonSet where each pod runs two containers. One container runs the NSX node agent, whose main responsibility is to manage container network interfaces. It interacts with the CNI plugin and the Kubernetes API server. The other container runs NSX kube-proxy, whose only responsibility is to implement Kubernetes service abstraction by translating cluster IPs into pod IPs. It implements the same functionality as the upstream kube-proxy.

Procedure

1 Download the NCP Docker image.

   The filename is `nsx-ncp-xxxxxxx.tar`, where *xxxxxxx* is the build number.

2 Download the NSX node agent DaemonSet yaml template.

   The filename is `ncp-node-agent-ds.yml`. You can edit this file or use it as an example for your own template file.

3 Load the NCP Docker image to your image registry.

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

4 Edit `ncp-node-agent-ds.yml`.

   Change the image name to the one that was loaded.

   For Ubuntu, the yaml file assumes that AppArmor is enabled. To see whether AppArmor is enabled, check the file `/sys/module/apparmor/parameters/enabled`. If AppArmor is not enabled, make the following changes:

   - Delete or comment out the following line:

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

   Add the line `privileged:true` under `securityContext` for the nsx-node-agent container and the nsx-kube-proxy container. For example:

```
securityContext:
  privileged:true
```

Note There is a known issue where if kubelet is run inside a container that uses the hyperkube image, kubelet always report AppArmor as disabled regardless of the actual state. You must make the same changes mentioned above to the yaml file.

Note In the yaml file, you must specify that the ConfigMap generated for `ncp.ini` must be mounted as a ReadOnly volume. The downloaded yaml file already has this specification, which should not be changed.

5 Create the NSX node agent DaemonSet with the following command.

```
kubectl apply –f ncp–node–agent–ds.yml
```

Configmap for ncp.ini in nsx-node-agent-ds.yml

The sample yaml file `nsx–node–agent–ds.yml` contains a ConfigMap for the configuration file `ncp.ini` for the NSX node agent. This ConfigMap section contains parameters that you can specify to customize your node agent installation.

The sample `nsx–node–agent–ds.yml` that you download has the following `ncp.ini` information:

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

    # Set to True to enable logging to stderr
    use_stderr = True
    # Set to True to send logs to the syslog daemon
    # use_syslog = False
    # Enabler debug–level logging for the root logger. If set to True, the
    # root logger debug level will be DEBUG, otherwise it will be INFO.
    # debug = True

    # Log file path for NCP operations.
    log_dir = /var/log/nsx–ujo/

    [coe]
    #
    # Common options for Container Orchestrators
    #

    # Container orchestrator adaptor to plug in
    # Options: kubernetes (default), cloud–foundry, openshift
    # adaptor = kubernetes

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

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

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

[k8s]
#
# From kubernetes
#

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

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

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

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

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

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

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

[nsx_node_agent]
#
# Configuration for nsx_node_agent
#

# Needs to mount node /proc to container if nsx_node_agent runs in a container.
# By default node /proc will be mounted to /host/proc, the prefix is /host.
# It should be the same setting with mounted path in the daemonset yaml file.
# Set the path to '' if nsx_node_agent is running as a process in minion node.
```

```
#proc_mount_path_prefix = /host

# The IP address for nsx_node_agent to communicate with NSX RPC server.
# The format should be ip/mask.
#nsxrpc_cip = 169.254.1.0/31

# The port for nsx_node_agent to communicate with NSX RPC server.
#nsxrpc_port = 2345

# The vlan id for nsx_node_agent to communicate with NSX RPC server.
#nsxrpc_vlan = 4094

# The interval of NSX RPC keep alive message.
#nsxrpc_keepalive_interval = 3

[nsx_kube_proxy]
#
# Configuration for nsx_kube_proxy
#

# The OVS uplink OpenFlow port where to apply the NAT rules to.
# If not specified, the port that gets assigned ofport=1 is used.
#ovs_uplink_port = <None>
```

Install NSX-T Container Plug-in

NSX-T Container Plug-in (NCP) is delivered as a Docker image. NCP should run on a node for infrastructure services. Running NCP on the master node is not recommended.

Procedure

1 Download the NCP Docker image.

   The filename is `nsx-ncp-xxxxxxx.tar`, where *xxxxxxx* is the build number.

2 Download the NCP ReplicationController yaml template.

   The filename is `ncp-rc.yml`. You can edit this file or use it as an example for your own template file.

3 Load the NCP Docker image to your image registry.

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

4 Edit `ncp-rc.yml`.

   Change the image name to the one that was loaded.

   Specify the `nsx_api_managers` parameter. This release supports a single Kubernetes node cluster and a single NSX Manager instance. For example:

```
nsx_api_managers = 192.168.1.180
```

(Optional) Specify the parameter `ca_file` in the `[nsx_v3]` section. The value should be a CA bundle file to use in verifying the NSX Manager server certificate. If not set, the system root CAs will be used.

Specify the parameters `nsx_api_cert_file` and `nsx_api_private_key_file` for authentication with NSX-T.

`nsx_api_cert_file` is the full path to a client certificate file in PEM format. The contents of this file should look like the following:

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

nsx_api_private_key_file is the full path to a client private key file in PEM format. The contents of this file should look like the following:

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

Specify the parameter `ingress_mode = nat` if the Ingress controller is configured to run in NAT mode.

By default, subnet prefix 24 is used for all subnets allocated from the IP blocks for the pod logical switches. To use a different subnet size, update the `subnet_prefix` option in the [nsx_v3] section.

Note In the yaml file, you must specify that the ConfigMap generated for `ncp.ini` be mounted as a ReadOnly volume. The downloaded yaml file already has this specification, which should not be changed.

5 Create NCP ReplicationController.

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

Configmap for ncp.ini in ncp-rc.yml

The sample YAML file `ncp-rc.yml` contains a ConfigMap for the configuration file `ncp.ini`. This ConfigMap section contains parameters that you must specify before you install NCP, as described in the previous section.

The sample `ncp-rc.yml` that you download has the following `ncp.ini` information:

```
# ConfigMap for ncp.ini
apiVersion: v1
kind: ConfigMap
metadata:
  name: nsx-ncp-config
  labels:
    version: v1
```

```
data:
  ncp.ini: |
    [DEFAULT]

    # Set to True to enable logging to stderr
    use_stderr = True
    # Set to True to send logs to the syslog daemon
    # use_syslog = False
    # Enabler debug-level logging for the root logger. If set to True, the
    # root logger debug level will be DEBUG, otherwise it will be INFO.
    # debug = True

    # Log file path for NCP operations.
    log_dir = /var/log/nsx-ujo/

    [coe]
    #
    # Common options for Container Orchestrators
    #

    # Container orchestrator adaptor to plug in
    # Options: kubernetes (default), cloud-foundry, openshift
    # adaptor = kubernetes

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

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

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

    [k8s]
    #
    # From kubernetes
    #

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

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

```
#apiserver_host_port = <port>

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

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

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

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

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

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

[nsx_v3]
#
# From nsx
#

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

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

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

# Path to NSX client private key file. If specified, the nsx_api_user and
# nsx_api_passsword options will be ignored. This option must be specified
```

```
# along with "nsx_api_cert_file" option.
# nsx_api_private_key_file = <None>

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

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

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

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

# The amount of time in seconds to wait before ensuring connectivity to the NSX
# manager if no manager connection has been used. (integer value)
#conn_idle_timeout = 10

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

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

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

# Subnet prefix of external IP block. Use subnet_prefix if not specified.
#external_subnet_prefix = <None>

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

Mount a PEM Encoded Certificate and a Private Key in the NCP Pod

If you have a PEM encoded certificate and a private key, you can update the NCP pod definition in the yaml file to mount the TLS secrets in the NCP Pod.

1 Create a TLS secret for the certificate and private key.

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

2 Update the NCP pod specification yaml to mount the secret as files in the NCP Pod specification.

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

3 Update the nsx_v3 options `nsx_api_cert_file` and `nsx_api_private_key_file` in the yaml file.

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

Mount a Certificate File in the NCP Pod

If you have a certificate file in the node file system, you can update the NCP pod specification to mount the file in the NCP pod.

For example,

```
spec:
  ...
  containers:
  - name: nsx-ncp
    ...
    volumeMounts:
    ...
    - name: nsx-cert
      # Mount path must match nsx_v3 option "nsx_api_cert_file"
      mountPath: /etc/nsx-ujo/nsx_cert
      - name: nsx-priv-key
      # Mount path must match nsx_v3 option "nsx_api_private_key_file"
      mountPath: /etc/nsx-ujo/nsx_priv_key
  volumes:
  ...
  - name: nsx-cert
    hostPath:
      path: <host-filesystem-cert-path>
  - name: nsx-priv-key
    hostPath:
      path: <host-filesystem-priv-key-path>
```

Configuring Syslog

You can run a syslog agent such as rsyslog or syslog-ng in a container to send logs from NCP and related components to a syslog server.

The following methods are recommended. For more information about logging in Kubernetes, see https://kubernetes.io/docs/concepts/cluster-administration/logging.

- Create a sidecar container that runs in the NCP or the nsx-node-agent pod.
- Run a DaemonSet replica on every node.

Note With the sidecar container method, NSX CNI plug-in logs cannot be sent to the syslog server because the plug-in does not run in a pod.

Create a Sidecar Container for Syslog

You can configure a sidecar container for syslog to run in the same pod as NCP. The following procedure assumes that the syslog agent image is example/rsyslog.

Procedure

1 Configure NCP and NSX node agent to log to a file.

In the yaml file for NCP and NSX node agent, set the log_dir parameter and specify the volume to be mounted. For example,

```
[default]
log_dir = /var/log/nsx-ujo/
...

spec:
  ...
  containers:
    - name: nsx-ncp
      ...
      volumeMounts:
      - name: nsx-ujo-log-dir
        # Mount path must match [default] option "log_dir"
        mountPath: /var/log/nsx-ujo
  volumes:
    ...
    - name: nsx-ujo-log-dir
      hostPath:
        path: <host-filesystem-log-dir-path>
```

You can change the log file name by setting the `log_file` parameter. The default names are `ncp.log`, `nsx_node_agent.log`, and `nsx_kube_proxy.log`. If the `log_dir` option is set to a path other than `/var/log/nsx-ujo`, either a hostPath volume or emptyDir volume must be created and mounted to the corresponding pod spec.

2 In the NCP pod's specification yaml file, add a ConfigMap for syslog. For example,

```
kind: ConfigMap
metadata:
  name: rsyslog-config
  labels:
    version: v1
data:
  ncp.conf: |
    module(load="imfile")

    ruleset(name="remote") {
      action(type="omfwd"
      Protocol="tcp"
      Target="nsx.example.com"
      Port="514")

      stop
    }

    input(type="imfile"
      File="/var/log/nsx-ujo/ncp.log"
      Tag="ncp"
      Ruleset="remote"
```

3 In the NCP pod's yaml file, add the rsyslog container and mount the appropriate volumes where rsyslog can find configuration data and read logs from other containers. For example,

```
spec:
  containers:
  - name: nsx-ncp
    ...
  - name: rsyslog
    image: example/rsyslog
    imagePullPolicy: IfNotPresent
    volumeMounts:
    - name: rsyslog-config-volume
      mountPath: /etc/rsyslog.d
      readOnly: true
    - name: nsx-ujo-log-dir
      mountPath: /var/log/nsx-ujo
  volumes:
    ...
    - name: rsyslog-config-volume
      configMap:
        name: rsyslog-config
    - name: nsx-ujo-log-dir
      hostPath:
        path: <host-filesystem-log-dir-path>
```

Create a DaemonSet Replica for Syslog

The logs of all NCP components can be redirected with this method. The applications need to be configured to log to stderr, which is enabled by default. The following procedure assumes that the syslog agent image is example/rsyslog.

Procedure

1 Create the DaemonSet yaml file. For example,

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: rsyslog-config
  labels:
    version: v1
data:
  nsx-ncp.conf: |
    module(load="imfile")

    ruleset(name="remote") {
      if $msg contains 'nsx-container' then
        action(type="omfwd"
        Protocol="tcp"
        Target="nsx.example.com"
        Port="514")

        stop
      }

      input(type="imfile"
        File="/var/log/containers/nsx-node-agent-*.log"
        Tag="nsx-node-agent"
        Ruleset="remote")

      input(type="imfile"
        File="/var/log/containers/nsx-ncp-*.log"
        Tag="nsx-ncp"
        Ruleset="remote")

      input(type="imfile"
        File="/var/log/syslog"
        Tag="nsx-cni"
        Ruleset="remote")
---
# rsyslog DaemonSet
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: rsyslog
  labels:
    component: rsyslog
    version: v1
```

```
    spec:
      template:
        metadata:
          labels:
          component: rsyslog
          version: v1
    spec:
      hostNetwork: true
      containers:
      - name: rsyslog
        image: example/rsyslog
        imagePullPolicy: IfNotPresent
        volumeMounts:
        - name: rsyslog-config-volume
          mountPath: /etc/rsyslog.d
        - name: log-volume
          mountPath: /var/log
        - name: container-volume
          mountPath: /var/lib/docker/containers
      volumes:
      - name: rsyslog-config-volume
        configMap:
        name: rsyslog-config
      - name: log-volume
        hostPath:
          path: /var/log
      - name: container-volume
        hostPath:
          path: /var/lib/docker/containers
```

2 Create the DaemonSet.

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

Example:

What to do next

Security Considerations

When deploying NCP, it is important to take steps to secure both the Kubernetes and the NSX-T environments.

Restrict NCP to Run Only on Designated Nodes

NCP has access to the NSX-T management plane and should be restricted to run only on designated infrastructure nodes. You can identify these nodes with an appropriate label. A nodeSelector for this label should then be applied to the NCP ReplicationController specification/ For example,

```
nodeSelector:
    nsx-infra: True
```

You can also use other mechanisms, such as affinity, to assign pods to nodes. For more information, see https://kubernetes.io/docs/concepts/configuration/assign-pod-node.

Ensure that the Docker Engine is Up To Date

Docker periodically releases security updates. An automated procedure should be implemented to apply these updates.

Disallow NET_ADMIN and NET_RAW Capabilities of Untrusted Containers

Linux capabilities NET_ADMIN and NET_RAW can be exploited by attackers to compromise the pod network. You should disable these two capabilities of untrusted containers. By default, NET_ADMIN capability is not granted to a non-privileged container. Be wary if a pod specification explicitly enables it or sets the container to be in a privileged mode. In addition, for untrusted containers, disable NET_RAW by specifying NET_RAW in the list of dropped capabilities in the SecurityContext configuration of the container's specification. For example,

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

Role-Based Access Control

Kubernetes uses Role-Based Access Control (RBAC) APIs to drive authorization decisions, allowing administrators to dynamically configure policies. For more information, see https://kubernetes.io/docs/admin/authorization/rbac.

Typically, the cluster administrator is the only user with privileged access and roles. For user and service accounts, the principle of least privilege must be followed when granting access.

The following guidelines are recommended:

- Restrict access to Kubernetes API tokens to pods which need them.
- Restrict access to NCP ConfigMap and NSX API client certificate's TLS secrets to the NCP pod.

- Block access to Kubernetes networking API from pods that do not require such access.
- Add a Kubernetes RBAC policy to specify which pods can have access to the Kubernetes API.

Recommended RBAC Policy for the NCP Pod

Create the NCP pod under a ServiceAccount and give this account a minimum set of privileges. Additionally, do not allow other pods or ReplicationControllers to access the ConfigMap and TLS Secrets that are mounted as volumes for the NCP ReplicationController and NSX node agent.

The following example shows how to specify roles and role bindings for NCP:

```
Cluster wide role to read, watch and get resources
----
kind: ClusterRole
# Set the apiVersion to v1 if running with OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: ncp-cluster-role
rules:
  - apiGroups:
    - ""
    - extensions
    resources:
      - deployments
      - endpoints
      - pods
      - namespaces
      - networkpolicies
      - nodes
      - replicationcontrollers
      - services
    verbs:
      - get
      - watch
      - list
----
Cluster wide role to read, watch, get and modify ingresses
----
kind: ClusterRole
# Set the apiVersion to v1 if running with OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: ncp-ingress-role
rules:
  - apiGroups:
    - ""
    - extensions
    resources:
    - ingresses
    verbs:
      - get
      - watch
      - list
```

```
        - update
        - patch
    - apiGroups:
      - extensions
      resources:
        - ingresses/status
      verbs:
        - replace
        - update
        - patch
---
Bind roles to ServiceAccount belonging to NCP
---
# Set the apiVersion to v1 if running with OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: ncp-cluster-role-binding
roleRef:
  # Comment out the apiGroup if running with OpenShift
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ncp-cluster-role
subjects:
  - kind: ServiceAccount
    name: ncp-svc-account
    namespace: ncp-deployed-ns-name
---
---
# Set the apiVersion to v1 if running with OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: ncp-ingress-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ncp-ingress-role
subjects:
  - kind: ServiceAccount
    name: ncp-svc-account
    namespace: ncp-deployed-ns-name
```

Note The TLS Secret that is created using the Kubernetes API for the NSX-T client certificate and the private key pair are accessible to any pod that has access to the Kubernetes API server. Similarly, when a pod is created with no service account, it is automatically assigned the default service account in the same namespace which auto mounts the token to access Kubernetes API. Therefore, access to these tokens must be restricted to pods which need them.

Administering NSX-T Container Plug-in

4

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

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. Select **DDI**.

   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.

Manage IP Block Subnets from the NSX Manager GUI

You can add and delete subnets 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. Select **DDI**.

   A list of the existing IP blocks is displayed.
3. Click an IP block name
4. Click the **Subnets** tab.
5. Perform any of the following actions..

Option	Action
Add an IP block subnet	Click **ADD**.
Delete one or more IP block subnets	Select one or more subnets and click **DELETE**.

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 **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.

Distributed network encryption is not supported in this release.

CLI Commands

To run CLI commands, log in to the NSX-T 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 4-1. CLI Commands for the NCP Container

Type	Command
Status	get ncp-nsx status
Status	get ncp-k8s-api-server status
Status	get ncp-watcher <watcher-name>
Status	get ncp-watchers
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
Support	get support-bundle file <filename>
Support	get ncp-log file <filename>
Support	get node-agent-log file <filename>
Support	get node-agent-log file <filename> <node-name>

Table 4-2. CLI Commands for the NSX Node Agent Container

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

Table 4-3. 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 connection status between NCP and NSX Manager

  ```
  get ncp-nsx status
  ```

  Example:

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

- 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
  ```

- 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
```

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 1:

```
kubenode> get project-cache default
    isolation:
        is_isolated: False
    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
```

Example 2:

```
kubenode> get project-caches
    default:
        isolation:
            is_isolated: False
        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:
        Isolation:
            is_isolated: False
        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
```

- Get the internal cache for pods

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

Example 1:

```
kubenode> get pod-cache nsx.default.nginx-rc-uq2lv
    cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
    gateway_ip: 10.0.0.1
    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
```

Example 2:

```
kubenode> get pod-caches
    nsx.default.nginx-rc-uq2lv:
        cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
        gateway_ip: 10.0.0.1
        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 the internal cache for ingress

```
get ingress caches
```

Example:

```
kubenode> get ingress-caches
    nsx.default.nginx-ingress-rc-host-ed3og: 10.192.162.201
```

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
  ```

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 applications. Users can retrieve the cache for a specific application or all applications.

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

  Example 1:

  ```
  kubenode> get app-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 app-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
```