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

Modified on 15 MAY 2018
VMware NSX-T 2.1.0.1, 2.1.2, 2.1.3, 2.1.4
VMware NSX-T Data Center 2.1

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

Copyright © 2017, 2018 VMware, Inc. All rights reserved. Copyright and trademark information.

Contents

NSX-T Container Plug-in for Kubernetes and Cloud Foundry - 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, as well as between NSX-T and Pivotal Cloud Foundry (PCF).

Intended Audience

This guide is intended for system and network administrators. A familiarity with the installation and administration of NSX-T, Kubernetes, and Pivotal Cloud Foundry 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) products such as OpenShift and Pivotal Cloud Foundry. This guide describes setting up NCP with Kubernetes and Pivotal Cloud Foundry.

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. You can have multiple Kubernetes clusters, each with its distinct NCP instance, using the same NSX-T deployment.

This chapter includes the following topics:

Compatibility Requirements

NSX-T Container Plug-in has the following compatibility requirements for a Kubernetes environment and a Pivotal Cloud Foundry (PCF) environment..

Table 1-1. Compatibility Requirements for a Kubernetes Environment

Software Product	Version
Hypervisor for Container Host VMs	■ ESXi 6.5 ■ ESXi 6.5 Update 1 ■ RHEL KVM 7.3, 7.4 ■ Ubuntu KVM 16.04
Container Host Operating System	■ RHEL 7.3, 7.4 ■ Ubuntu 16.04

Table 1-1. Compatibility Requirements for a Kubernetes Environment (Continued)

Software Product	Version
Container Runtime	Docker 1.12 (NCP 2.1, 2.1.0.1, 2.1.2) Docker 1.13 (NCP 2.1.2)
Container Orchestrator	Kubernetes 1.7, 1.8 (NCP 2.1, 2.1.0.1, 2.1.2) Kubernetes 1.9, 1.10 (NCP 2.1.2)
Container Host vSwitch	OVS 2.6, 2.7, 2.8

Table 1-2. Compatibility Requirements for a Cloud Foundry Environment

Software Product	Version
Hypervisor for Container Host VMs	■ ESXi 6.5 ■ ESXi 6.5 Update 1 ■ RHEL KVM 7.3, 7.4 ■ Ubuntu KVM 16.04
Container Orchestrator	Pivotal Application Service 2.0 and Pivotal Operations Manager 2.0 (NCP 2.1, 2.1.0.1, 2.1.2) Pivotal Application Service 2.1.1 and Pivotal Operations Manager 2.1 (NCP 2.1.2)

Installation Overview

In an environment with Kubernetes already installed, installing and configuring NCP typically involve the following steps. To perform the steps successfully, you must be familiar with NSX-T and Kubernetes installation and administration.

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 pool for SNAT.

In NCP 2.1 and 2.1.0.1, you must use tags to configure NSX-T resources. Starting with NCP 2.1.2, you must configure NSX-T resources using UUIDs or names in the configuration file `ncp.ini`.

Overlay Transport Zone

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.

(NCP 2.1 and 2.1.0.1) Tag the transport zone with one or more `{'ncp/cluster': '<cluster_name>'}` tags to make it specific to one or more clusters, or tag it with `{'ncp/shared_resource': 'true'}` to make it shared by clusters. If the tag `ncp/cluster` exists, the cluster that is named in the tag will only use the resource with the `ncp/cluster` tag and will not use any resource with the `ncp/shared_resource` tag. The cluster name must match the value of the `cluster` option in the `[coe]` section in `ncp.ini`.

(NCP 2.1.2) Specify an overlay transport zone for a cluster by setting the `overlay_tz` option in the `[nsx_v3]` section of `ncp.ini`.

Tier-0 Logical Routing

Log in to NSX Manager and navigate to **Routing > Routers**. Find the router that is used for container networking or create a new one.

(NCP 2.1 and 2.1.0.1) Tag the router with one or more `{'ncp/cluster': '<cluster_name>'}` tags to make it specific to one or more clusters, or tag it with `{'ncp/shared_resource': 'true'}` to make it shared by clusters. If the tag `ncp/cluster` exists, the cluster that is named in the tag will only use the resource with the `ncp/cluster` tag and will not use any resource with the `ncp/shared_resource` tag. The cluster name must match the value of the `cluster` option in the `[coe]` section in `ncp.ini`.

(NCP 2.1.2) 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, although doing so will make setting up easier. You can create a logical switch by logging in to NSX Manager and navigating to **Switching > Switches**. On the switch, create logical ports and attach the node vNICs to them. The logical ports must be tagged with `{'ncp/cluster': '<cluster_name>'}` and `{'ncp/node_name': '<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 navigate to **DDI > IPAM** to create one or more IP blocks. Specify the IP block in CIDR format.

(NCP 2.1 and 2.1.0.1) Tag the IP block with one or more `{'ncp/cluster': '<cluster_name>'}` tags to make it specific to one or more clusters, or tag it with `{'ncp/shared_resource': 'true'}` to make it shared by clusters. If the tag `ncp/cluster` exists, the cluster that is named in the tag will only use the resource with the `ncp/cluster` tag and will not use any resource with the `ncp/shared_resource` tag. The cluster name must match the value of the `cluster` option in the `[coe]` section in `ncp.ini`.

(NCP 2.1.2) 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 (for Kubernetes) or clusters (for PCF).

(NCP 2.1 and 2.1.0.1) Tag the IP blocks with `{'ncp/no_snat': 'true'}` in addition to the `ncp/cluster` tag. If you are upgrading from a previous release and have the tag `{'ncp/no_snat': '<cluster_name>'}`, you must manually change it to `{'ncp/no_snat': 'true'}`.

(NCP 2.1.2) 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`. For more information, see Configmap for ncp.ini in ncp-rc.yml.

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 navigate to **Inventory > Groups > IP POOL** to create a pool or find an existing pool.

(NCP 2.1 and 2.1.0.1) Tag the pool with `{'ncp/external': 'true'}`. In addition, tag the IP block with one or more `{'ncp/cluster': '<cluster_name>'}` tags to make it specific to one or more clusters, or tag it with `{'ncp/shared_resource': 'true'}` to make it shared by clusters. If the tag `ncp/cluster` exists, the cluster that is named in the tag will only use the resource with the `ncp/cluster` tag and will not use any resource with the `ncp/shared_resource` tag. The cluster name must match the value of the `cluster` option in the `[coe]` section in `ncp.ini`.

(NCP 2.1.2) 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 pool specified by `ncp/snat_pool` should already exist in NSX-T before the service is configured.
- In NSX-T, 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.

(NCP 2.1.4) By default, NCP configures SNAT IP for a PAS (Pivotal Application Service) org. You can configure an SNAT IP for a specific app by creating an app with a `manifest.xml` that contains the SNAT IP pool information. For example,

```
applications:
  - name: frontend
    memory: 32M
    disk_quota: 32M
    buildpack: go_buildpack
    env:
      GOPACKAGENAME: example-apps/cats-and-dogs/frontend
      NCP_SNAT_POOL: <external IP pool ID or name>
...
```

NCP will configure the SNAT rule for this app. The rule's source IP is the set of instances' IPs and its destination IP is the SNAT IP allocated from an external IP pool. Note the following:

- The pool specified by `NCP_SNAT_POOL` should already exist in NSX-T before the app is pushed.
- The priority of SNAT rule for an app is higher than that for an org.
- An app can be configured with only one SNAT IP.

(Optional) (For Kubernetes only) 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, navigate to **Firewall > General** and create two firewall sections.

(NCP 2.1 and 2.1.0.1) Tag one section with `{'ncp/fw_sect_marker': 'bottom'}` and the other tagged with `{'ncp/fw_sect_marker': 'top'}`.

(NCP 2.1.2) 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.

(For Pivotal Cloud Foundry only) SpoofGuard Switching Profile

(NCP 2.1.0.1 and earlier releases) Log in to NSX Manager and navigate to **Switching > Switching Profiles**. Add a SpoofGuard switching profile and add the following tags to the profile:

```
{'ncp/cluster': '<cluster_name>'}
{'ncp/ha': 'true'}
```

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 add 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 in a Kubernetes Environment

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

NSX-T CNI plug-in must be installed on the Kubernetes nodes.

For Ubuntu, installing the NSX-T CNI plug-in will copy the AppArmor profile file `ncp-apparmor` to `/etc/apparmor.d` and load it. Before the install, the AppArmor service must be running and the directory `/etc/apparmor.d` must exist. Otherwise, the install will fail. You can check whether the AppArmor module is enabled with the following command:

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

You can check whether the AppArmor service is started with the following command:

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

If the AppArmor service is not running when you install the NSX-T CNI plug-in, the install will display the following message when it finishes:

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

The message indicates that all the installation steps completed except the loading of the AppArmor profile.

The `ncp-apparmor` profile file provides an AppArmor profile for NSX node agent called `node-agent-apparmor`, which differs from the `docker-default` profile in the following ways:

- The `deny mount` rule is removed.
- The `mount` rule is added.
- Some `network`, `capability`, `file`, and `umount` options are added.

You can replace the `node-agent-apparmor` profile with a different profile. However, the profile name `node-agent-apparmor` is referenced in the file `nsx-node-agent-ds.yml`, which is used in the installation of NSX node agent. If you use a different profile, you must specify the profile name in `nsx-node-agent-ds.yml`, under the section `spec:template:metadata:annotations`, in the following entry:

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

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
```

Note NCP opens persistent HTTP connections to the Kubernetes API server to watch for life cycle events of Kubernetes resources. If an API server failure or a network failure causes NCP's TCP connections to become stale, you must restart NCP so that it can re-establish connections to the API server. Otherwise, NCP will miss the new events.

During a rolling update of the NCP ReplicationController, do not reboot the container host. If the host is rebooted for any reason, you might see two NCP pods running after the reboot. In that case, you should delete one of the NCP pods. It does not matter which one.

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>

# User name for the NSX manager (string value)
#nsx_api_user = <None>

# Password for the NSX manager (string value)
#nsx_api_password = <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

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

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

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

# Max number of virtual servers allowed on a single load balancer service.
```

```
# SMALL LB supports 10 virtual servers. MEDIUM LB supports 100 virtual servers.
# LARGE LB supports 1000 virtual servers'
# Defaults to 10.
#virtual_servers_per_lbs = 10

# Retrieve the node VIF via the display name of the VM, in deployments where
# the display name of the VM is same as the node name
#get_node_vif_by_vm_name = 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: /var/log/nsx-ujo
```

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: Configuring Log Rotation and Syslog Running in a Sidecar Container

The following procedure shows how to configure log rotation and syslog running in a sidecar container.

Creating the Log Directory and Configuring Log Rotation

- Create the log directory on all the nodes, including the master, and change its owner to whatever user has ID 1000.

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

- Configure log rotation on all the nodes for the `/var/log/nsx-ujo` directory.

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

Creating the NCP Replication Controller

- Create the `ncp.ini` file for NCP.

```
cat <<EOF > /tmp/ncp.ini
[DEFAULT]
log_dir = /var/log/nsx-ujo
[coe]
cluster = k8s-cl1
[k8s]
apiserver_host_ip = 10.114.209.77
apiserver_host_port = 6443
ca_file = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
client_token_file = /var/run/secrets/kubernetes.io/serviceaccount/token
```

```
insecure = True
ingress_mode = nat
[nsx_v3]
nsx_api_user = admin
nsx_api_password = Password1!
nsx_api_managers = 10.114.209.68
insecure = True
subnet_prefix = 29
[nsx_node_agent]
[nsx_kube_proxy]
ovs_uplink_port = ens192
EOF
```

- Create the config map from the ini file.

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

- Create the NCP rsyslog config.

```
cat <<EOF > /tmp/nsx-ncp-rsyslog.conf
# yaml template for NCP ReplicationController
# Correct kubernetes API and NSX API parameters, and NCP Docker image
# must be specified.
apiVersion: v1
kind: ConfigMap
metadata:
    name: rsyslog-config
    labels:
        version: v1
data:
    ncp.conf: |
        module(load="imfile")

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

            stop
        }

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

- Create the config map from the above.

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

- Create the NCP replication controller with the `rsyslog` sidecar.

```
cat <<EOF > /tmp/ncp-rc-with-logging.yml
# Replication Controller yaml for NCP
apiVersion: v1
kind: ReplicationController
metadata:
  # VMware NSX Container Plugin
  name: nsx-ncp
  labels:
    tier: nsx-networking
    component: nsx-ncp
    version: v1
spec:
  # Active-Active/Active-Standby is not supported in current release.
  # so replica *must be* 1.
  replicas: 1
  template:
    metadata:
      labels:
        tier: nsx-networking
        component: nsx-ncp
        version: v1
    spec:
      # NCP shares the host management network.
      hostNetwork: true
      nodeSelector:
        kubernetes.io/hostname: k8s-master
      tolerations:
      - key: "node-role.kubernetes.io/master"
        operator: "Exists"
        effect: "NoSchedule"
      containers:
        - name: nsx-ncp
          # Docker image for NCP
          image: nsx-ujo-docker-local.artifactory.eng.vmware.com/nsx-ncp:ob-6236425
          imagePullPolicy: IfNotPresent
          readinessProbe:
            exec:
              command:
              - cat
              - /tmp/ncp_ready
            initialDelaySeconds: 5
            periodSeconds: 5
            failureThreshold: 5
          securityContext:
            capabilities:
              add:
                - NET_ADMIN
                - SYS_ADMIN
                - SYS_PTRACE
                - DAC_READ_SEARCH
          volumeMounts:
          - name: config-volume
            # NCP expects ncp.ini is present in /etc/nsx-ujo
```

```
            mountPath: /etc/nsx-ujo
          - name: log-volume
            mountPath: /var/log/nsx-ujo
        - name: rsyslog
          image: jumanjiman/rsyslog
          imagePullPolicy: IfNotPresent
          volumeMounts:
          - name: rsyslog-config-volume
            mountPath: /etc/rsyslog.d
            readOnly: true
          - name: log-volume
            mountPath: /var/log/nsx-ujo
      volumes:
        - name: config-volume
          # ConfigMap nsx-ncp-config is expected to supply ncp.ini
          configMap:
            name: nsx-ncp-config-with-logging
        - name: rsyslog-config-volume
          configMap:
            name: rsyslog-config
        - name: log-volume
          hostPath:
            path: /var/log/nsx-ujo/
EOF
```

- Create NCP with the above specification.

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

Creating the NSX Node Agent Daemon Set

- Create the `rsyslog` configuration for the node agents.

```
cat <<EOF > /tmp/nsx-node-agent-rsyslog.conf
# yaml template for NCP ReplicationController
# Correct kubernetes API and NSX API parameters, and NCP Docker image
# must be specified.
apiVersion: v1
kind: ConfigMap
metadata:
    name: rsyslog-config-node-agent
    labels:
        version: v1
data:
    ncp.conf: |
        module(load="imfile")

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

```
            stop
        }

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

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

- Create the `configmap` from the above.

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

- Create the DaemonSet with the `configmap` sidecar.

```
cat <<EOF > /tmp/nsx-node-agent-rsyslog.yml
# nsx-node-agent DaemonSet
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: nsx-node-agent
  labels:
    tier: nsx-networking
    component: nsx-node-agent
    version: v1
spec:
  template:
    metadata:
      annotations:
        container.apparmor.security.beta.kubernetes.io/nsx-node-agent: localhost/node-agent-
apparmor
      labels:
        tier: nsx-networking
        component: nsx-node-agent
        version: v1
    spec:
      hostNetwork: true
      tolerations:
      - key: "node-role.kubernetes.io/master"
        operator: "Exists"
        effect: "NoSchedule"
      containers:
        - name: nsx-node-agent
          # Docker image for NCP
          image: nsx-ujo-docker-local.artifactory.eng.vmware.com/nsx-ncp:ob-6236425
          imagePullPolicy: IfNotPresent
          # override NCP image entrypoint
          command: ["nsx_node_agent"]
          livenessProbe:
```

```
            exec:
              command:
                - /bin/sh
                - -c
                - ps aux | grep [n]sx_node_agent
            initialDelaySeconds: 5
            periodSeconds: 5
          securityContext:
            capabilities:
              add:
                - NET_ADMIN
                - SYS_ADMIN
                - SYS_PTRACE
                - DAC_READ_SEARCH
          volumeMounts:
          # ncp.ini
          - name: config-volume
            mountPath: /etc/nsx-ujo
          # mount openvswitch dir
          - name: openvswitch
            mountPath: /var/run/openvswitch
          # mount CNI socket path
          - name: cni-sock
            mountPath: /var/run/nsx-ujo
          # mount container namespace
          - name: netns
            mountPath: /var/run/netns
          # mount host proc
          - name: proc
            mountPath: /host/proc
            readOnly: true
          - name: log-volume
            mountPath: /var/log/nsx-ujo
        - name: nsx-kube-proxy
          # Docker image for NCP
          image: nsx-ujo-docker-local.artifactory.eng.vmware.com/nsx-ncp:ob-6236425
          imagePullPolicy: IfNotPresent
          # override NCP image entrypoint
          command: ["nsx_kube_proxy"]
          livenessProbe:
            exec:
              command:
                - /bin/sh
                - -c
                - ps aux | grep [n]sx_kube_proxy
            initialDelaySeconds: 5
            periodSeconds: 5
          securityContext:
            capabilities:
              add:
                - NET_ADMIN
                - SYS_ADMIN
                - SYS_PTRACE
                - DAC_READ_SEARCH
          volumeMounts:
```

```
          # ncp.ini
          - name: config-volume
            mountPath: /etc/nsx-ujo
          # mount openvswitch dir
          - name: openvswitch
            mountPath: /var/run/openvswitch
          - name: log-volume
            mountPath: /var/log/nsx-ujo
        - name: rsyslog
          image: jumanjiman/rsyslog
          imagePullPolicy: IfNotPresent
          volumeMounts:
          - name: rsyslog-config-volume
            mountPath: /etc/rsyslog.d
            readOnly: true
          - name: log-volume
            mountPath: /var/log/nsx-ujo
      volumes:
        - name: config-volume
          configMap:
            name: nsx-ncp-config-with-logging
        - name: cni-sock
          hostPath:
            path: /var/run/nsx-ujo
        - name: netns
          hostPath:
            path: /var/run/netns
        - name: proc
          hostPath:
            path: /proc
        - name: openvswitch
          hostPath:
            path: /var/run/openvswitch
        - name: rsyslog-config-volume
          configMap:
            name: rsyslog-config-node-agent
        - name: log-volume
          hostPath:
            path: /var/log/nsx-ujo/
EOF
```

- Create the DaemonSet.

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

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:

```
# Create a ServiceAccount for NCP namespace
apiVersion: v1
kind: ServiceAccount
metadata:
 name: ncp-svc-account
 namespace: nsx-system

---

# Create ClusterRole for NCP
kind: ClusterRole
# Set the apiVersion to v1 while using OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
 name: ncp-cluster-role
rules:
 - apiGroups:
   - ""
   - extensions
   - networking.k8s.io
   resources:
     - deployments
     - endpoints
     - pods
     - pods/log
     - namespaces
     - networkpolicies
     # Move 'nodes' to ncp-patch-role when hyperbus is disabled.
     - nodes
     - replicationcontrollers
     # Remove 'secrets' if not using Native Load Balancer.
     - secrets
   verbs:
     - get
     - watch
     - list

---

# Create ClusterRole for NCP to edit resources
kind: ClusterRole
# Set the apiVersion to v1 while using OpenShift
```

```
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
 name: ncp-patch-role
rules:
 - apiGroups:
   - ""
   - extensions
   resources:
     - ingresses
     - services
   verbs:
     - get
     - watch
     - list
     - update
     - patch
 - apiGroups:
   - ""
   - extensions
   resources:
     - ingresses/status
     - services/status
   verbs:
     - replace
     - update
     - patch
---

# Bind ServiceAccount created for NCP to its ClusterRole
kind: ClusterRoleBinding
# Set the apiVersion to v1 while using OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
 name: ncp-cluster-role-binding
roleRef:
 # Comment out the apiGroup while using OpenShift
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: ncp-cluster-role
subjects:
 - kind: ServiceAccount
   name: ncp-svc-account
   namespace: nsx-system

---

# Bind ServiceAccount created for NCP to the patch ClusterRole
kind: ClusterRoleBinding
# Set the apiVersion to v1 while using OpenShift
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
 name: ncp-patch-role-binding
roleRef:
 # Comment out the apiGroup while using OpenShift
 apiGroup: rbac.authorization.k8s.io
```

```
  kind: ClusterRole
  name: ncp-patch-role
subjects:
  - kind: ServiceAccount
    name: ncp-svc-account
    namespace: nsx-system
```

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.

Tips on Configuring Network Resources

When configuring some network resources, you should be aware of certain restrictions.

NSX-T Tagging Limits

NSX-T has the following limits on tagging an object:

- The scope has a limit of 20 characters.
- The tag has a limit of 40 characters.
- Each object can have no more than 15 tags.

These limits might cause issues when Kubernetes or OpenShift annotations are copied to NSX-T scopes and tags and the limits are exceeded. For example, if a tag is for a switch port and the tag is used in a firewall rule, the rule might not be applied as expected because the annotation key or value was truncated when copied to a scope or tag.

Configuring Network Policies

(NCP 2.1.0.1 and earlier releases) When creating a `NetworkPolicy` resource, in the `podSelector` field, the `matchExpressions` field must be empty. Otherwise, NCP will fail to implement the network policy.

Note Network policies select pods or namespaces using label selectors. In NCP 2.1.0.1 and earlier releases, NCP supports only equality-based label selectors. It does not support inequality-based or set-based label selectors. Starting with NCP 2.1.2, this restriction is removed.

NCP's support for network policies is the same as the support provided by Kubernetes and depends on the Kubernetes version.

- Kubernetes 1.10 and earlier - The rule clauses in the network policy may contain at most one selector from `namespaceSelector`, `podSelector` and `ipBlock`.
- Kubernetes 1.7 - Support for egress rules and `ipBlock` CIDR selector is not available. They are available as beta fields in Kubernetes 1.8 and later.

Installing NCP in a Pivotal Cloud Foundry Environment

4

Pivotal Cloud Foundry (PCF) is an open source platform-as-a-service (PaaS) provider. You can install NSX-T Container Plug-in (NCP) in a PCF environment to provide networking services.

VMs created through the Pivotal Ops Manager must have layer 3 connectivity to the container network to access the NSX-T features.

High availability (HA) is automatically enabled. Before installing NCP, you must create a SpoofGuard switching profile with the tags `{'ncp/cluster': '<cluster_name>'}` and `{'ncp/ha': 'true'}`.

Note When a change is made to a security group, you must re-stage all the applications that the security group applies to. This can happen either because the security group applies to the space where the applications are running, or because the security group is global.

Install NCP in a Pivotal Cloud Foundry Environment

NCP is installed through the Pivotal Ops Manager graphical user interface.

Prerequisites

A fresh installation of Pivotal Ops Manager, NSX-T, and Pivotal Application Service (PAS). Make sure that Ops Manager is installed first, then NSX-T, and then PAS. For more information, see the Pivotal Cloud Foundry documentation.

Procedure

1. Download the NCP installation file for PCF.

   The file name is `VMware-NSX-T.<version>.<build>.pivotal`.
2. Log in to Pivotal Ops Manager as an administrator.
3. Click **Import a Product**.
4. Select the file that was downloaded.
5. Click the **Ops Manager Director for VMware vSphere** tile.
6. In the **Settings** tab for **vCenter Config**, select **NSX Networking** and for **NSX Mode**, select **NSX-T**.
7. In the **NSX CA Cert** field, provide the certificate in PEM format.
8. Click **Save**.

9 Click **Installation Dashboard** in the upper left corner to return to the dashboard.

10 Click the **Pivotal Application Service** tile.

11 In the **Settings** tab, select **Networking** in the navigation pane.

12 Under **Container Network Interface Plugin**, select **External**.

13 Click **Installation Dashboard** in the upper left corner to return to the dashboard.

14 Click **Save**.

15 Click **Installation Dashboard** in the upper left corner to return to the dashboard.

16 Click the **VMware NSX-T** tile.

17 Enter the address of the NSX Manager.

18 Select the method for NSX Manager authentication.

Option	Action
Client Certificate Authentication	Provide the certificate and private key for NSX Manager.
Basic Authentication with Username and Password	Provide the NSX Manager administrator user name and password.

19 In the **NSX Manager CA Cert** field, provide the certificate.

20 Click **Save**.

21 Select **NCP** in the navigation pane.

22 Enter the **PAS Foundation Name**.

This string uniquely identifies a PAS foundation in NSX API. This string is also used as the prefix in the names of NSX resources created by NCP for the PAS foundation.

23 (NCP 2.1.2) Enter the **Overlay Transport Zone**.

24 (NCP 2.1.2) Enter the **Tier-0 Router**.

25 (NCP 2.1.2) Specify one or more **IP Blocks of Container Networks**.

a Click **Add**.

b Enter **IP Block Name**. It can be a new or existing IP block.

c For a new IP block only, specify the block in CIDR format, for example, 10.1.0.0/16.

26 Specify the subnet prefix of the container networks.

27 Click **Enable SNAT for Container Networks** to enable SNAT.

28 (NCP 2.1.2) Specify one or more **IP Pools used to provide External (NAT) IP Address to Org Networks**.

   a Click **Add**.

   b Enter **IP Pool Name**. It can be a new or existing IP pool.

   c For a new IP pool only, specify the IP addresses by providing the CIDR and the IP ranges.

29 (Optional) (NCP 2.1.2) Enter the **Top Firewall Section Marker**.

30 (Optional) (NCP 2.1.2) Enter the **Bottom Firewall Section Marker**.

31 (Optional) Enable or disable the following options.

Option	Default Value
Log Dropped Application Traffic	Disabled. If enabled, traffic that is dropped due to a firewall rule will be logged.
Enable Debug Level for NCP Logging	Enabled.

32 Click **Save**.

33 (Optional) Select **NSX Node Agent** in the navigation pane.

   a Check **Enable Debug Level of Logging for NSX Node Agent** to enable debug level logging.

   b Click **Save**.

34 Click **Installation Dashboard** in the upper left corner to return to the dashboard.

35 Click **Apply Changes**.

Load Balancing

5

You can configure load balancing for a cluster to balance traffic to the pods.

Configuring Load Balancing

Configuring load balancing involves configuring a Kubernetes LoadBalancer service or Ingress resource, and the NCP replication controller.

You can create a layer 4 load balancer by configuring a Kubernetes service of type LoadBalancer. The service 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. Starting with NCP 2.1.4, 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.

You can create a layer 7 load balancer by configuring a Kubernetes Ingress resource. The Ingress resource is allocated an IP address from the external IP block that you configure. The load balancer is exposed on this IP address and the HTTP port 80. Note that a Kubernetes service of type LoadBalancer is not supported as a backend for the Ingress resource. Also, Ingress with a TLS section is not supported because NSX-T load balancer does not have SSL termination capability in this release.

Note You cannot assign a specific IP address for the LoadBalancer service or the Ingress resource. Any address you specify when creating the LoadBalancer service or Ingress resource will be ignored.

To configure load balancing in NCP, in the `ncp_rc.yml` file:

1 Set `use_native_loadbalancer` = **`True`**.

2 (Optional) Set `pool_algorithm` to **`'ROUND_ROBIN'`** or **`'LEAST_CONNECTION/IP_HASH'`**. The default is **`'ROUND_ROBIN'`**.

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

The **`LEAST_CONNECTION/IP_HASH`** algorithm means that traffic from the same source IP address will be sent to the same backend pod.

The small LoadBalancer service supports the following:

- 10 NSX-T virtual servers.
- 10 NSX-T pools.

- 30 NSX-T pool members.
- 8 ports for LoadBalancer services.
- A total of 10 ports defined by the LoadBalancer services and Ingress resources.
- A total of 30 endpoints referenced by the LoadBalancer services and Ingress resources.

The medium LoadBalancer service supports the following:

- 100 NSX-T virtual servers.
- 100 NSX-T pools.
- 300 NSX-T pool members.
- 98 ports for LoadBalancer services.
- A total of 100 ports defined by the LoadBalancer services and Ingress resources.
- A total of 300 endpoints referenced by the LoadBalancer services and Ingress resources.

The large LoadBalancer service supports the following:

- 1000 NSX-T virtual servers.
- 1000 NSX-T pools.
- 3000 NSX-T pool members.
- 998 ports for LoadBalancer services.
- A total of 1000 ports defined by the LoadBalancer services and Ingress resources.
- A total of 3000 endpoints referenced by the LoadBalancer services and Ingress resources.

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.

Ingress

- NSX-T will create one layer 7 load balancer for all Ingresses without TLS data in the specification. Ingresses with TLS data in the specification are not supported because there is no support for HTTPS in this release.
- All Ingresses will get a single IP address.
- Ingresses will be hosted on port 80.
- If there are duplicate rules in Ingress definitions for a single cluster, only the first rule will be applied.
- Only a single Ingress with a default backend is supported per cluster. Traffic not matching any Ingress rule will be forwarded to the default backend.
- If there are multiple Ingresses with a default backend, only the first one will be configured. The others will be annotated with an error.

- (NCP 2.1.0.1) Wildcard URI matching is supported using the regular expression characters "." and "*". For example, the path "/coffee/.*" matches "/coffee/" followed by zero, one or more characters, such as "/coffee/", "/coffee/a", "/coffee/b", but not "/coffee", "/coffeecup" or "/coffeecup/a". Note that if the path contains "/*", for example "/tea/*", it will match "/tea" followed by zero, one or more characters, such as "/tea", "/tea/", "/teacup", "/teacup/", "/tea/a" or "/teacup/b". In this case, the regular expression special character "*" is acting as a wildcard character as well.

  An Ingress specification example:

```
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  rules:
  - http:
      paths:
      - path: /tea/*        #Matches /tea, /tea/, /teacup, /teacup/, /tea/a, /teacup/b, etc.
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee/.*    #Matches /coffee/, /coffee/a but NOT /coffee, /coffeecup, etc.
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

- (NCP 2.1.4) You can configure HTTP URL request rewrite by adding an annotation to the Ingress resource. For example,

```
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    ncp/rewrite_target: "/"
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

  The paths `/tea` and `/coffee` will be rewritten to `/` before the URL is sent to the backend service.

Layer 7 Load Balancer and Network Policy

When traffic is forwarded to the pods from the NSX load balancer virtual server, the source IP is the tier-1 router's uplink port's IP address. This address is on the private tier-1 transit network, and can cause the CIDR-based network policies to disallow traffic that should be allowed. To avoid this issue, the network policy must be configured such that the tier-1 router's uplink port's IP address is part of the allowed CIDR block. This internal IP address will be visible to the user as part of the Ingress specification in the `status.loadbalancer.ingress.ip` field and as an annotation (`ncp/internal_ip_for_policy`) on the Ingress resource.

For example, if the external IP address of the virtual server is 4.4.0.5 and the IP address of the internal tier-1 router's uplink port is 100.64.224.11, the Ingress specification will be:

```
kind: Ingress
...
status:
  loadBalancer:
  ingress:
  - ip: 4.4.0.5
  - ip: 100.64.224.11
```

The annotation on the Ingress resource will be:

```
ncp/internal_ip_for_policy: 100.64.224.11
```

LoadBalancer Service

- NSX-T will create a layer 4 load balancer for each service port.
- Both TCP and UDP are supported.
- Each service will have a unique IP address.

Administering NSX-T Container Plug-in

6

You can administer NSX-T 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 IDs 4093 and 4094 on the hypervisor for PVLAN configuration and the IDs cannot be changed. To avoid any VLAN conflict, do not configure VLAN logical switches or VTEP vmknics with the same VLAN IDs..

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 6-1. CLI Commands for the NCP Container

Type	Command	Note
Status	get ncp-master status	For both Kubernetes and PCF.
Status	get ncp-nsx status	For both Kubernetes and PCF.
Status	get ncp-watcher <watcher-name>	For both Kubernetes and PCF.
Status	get ncp-watchers	For both Kubernetes and PCF.
Status	get ncp-k8s-api-server status	For Kubernetes only.
Status	check projects	For Kubernetes only.
Status	check project <project-name>	For Kubernetes only.
Status	get ncp-bbs status	For PCF only.
Status	get ncp-capi status	For PCF only.
Status	get ncp-policy-server status	For PCF only.
Cache	get project-caches	For Kubernetes only.
Cache	get project-cache <project-name>	For Kubernetes only.
Cache	get namespace-caches	For Kubernetes only.
Cache	get namespace-cache <namespace-name>	For Kubernetes only.
Cache	get pod-caches	For Kubernetes only.
Cache	get pod-cache <pod-name>	For Kubernetes only.

Table 6-1. CLI Commands for the NCP Container (Continued)

Type	Command	Note
Cache	get ingress-caches	For Kubernetes only.
Cache	get ingress-cache <ingress-name>	For Kubernetes only.
Cache	get ingress-controllers	For Kubernetes only.
Cache	get ingress-controller <ingress-controller-name>	For Kubernetes only.
Cache	get network-policy-caches	For Kubernetes only.
Cache	get network-policy-cache <pod-name>	For Kubernetes only.
Cache	get asg-caches	For PCF only.
Cache	get asg-cache <asg-ID>	For PCF only.
Cache	get org-caches	For PCF only.
Cache	get org-cache <org-ID>	For PCF only.
Cache	get space-caches	For PCF only.
Cache	get space-cache <space-ID>	For PCF only.
Cache	get app-caches	For PCF only.
Cache	get app-cache <app-ID>	For PCF only.
Cache	get instance-caches <app-ID>	For PCF only.
Cache	get instance-cache <app-ID> <instance-ID>	For PCF only.
Cache	get policy-caches	For PCF only.
Support	get ncp-log file <filename>	For both Kubernetes and PCF.
Support	get ncp-log-level	For both Kubernetes and PCF.
Support	set ncp-log-level <log-level>	For both Kubernetes and PCF.
Support	get support-bundle file <filename>	For Kubernetes only.
Support	get node-agent-log file <filename>	For Kubernetes only.
Support	get node-agent-log file <filename> <node-name>	For Kubernetes only.

Table 6-2. CLI Commands for the NSX Node Agent Container

Type	Command
Status	get node-agent-hyperbus status
Cache	(NCP 2.1, 2.1.0.1) get app-cache <app-name> (NCP 2.1.2) get container-cache <container-name>
Cache	(NCP 2.1, 2.1.0.1) get app-caches (NCP 2.1.2) get container-caches

Table 6-3. 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-watchers
get ncp-watcher <watcher-name>
```

Example:

```
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


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
```

- 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
```

- Check connection status between NCP and PCF BBS

```
get ncp-bbs status
```

Example:

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

- Check connection status between NCP and PCF CAPI

```
get ncp-capi status
```

Example:

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

- Check connection status between NCP and PCF policy server

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

Example:

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

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 all Ingress caches or a specific one

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

Example:

```
kubenode> get ingress-caches
    nsx.default.cafe-ingress:
        ext_pool_id: cc02db70-539a-4934-a938-5b851b3e485b
        lb_virtual_server:
            id: 895c7f43-c56e-4b67-bb4c-09d68459d416
            lb_service_id: 659eefc6-33d1-4672-a419-344b877f528e
            name: dgo2-http
            type: http
        lb_virtual_server_ip: 5.5.0.2
        name: cafe-ingress
        rules:
            host: cafe.example.com
            http:
                paths:
                    path: /coffee
                    backend:
                        serviceName: coffee-svc
                        servicePort: 80
                    lb_rule:
                        id: 4bc16bdd-abd9-47fb-a09e-21e58b2131c3
                        name: dgo2-default-cafe-ingress/coffee
```

```
kubenode> get ingress-cache nsx.default.cafe-ingress
    ext_pool_id: cc02db70-539a-4934-a938-5b851b3e485b
    lb_virtual_server:
        id: 895c7f43-c56e-4b67-bb4c-09d68459d416
        lb_service_id: 659eefc6-33d1-4672-a419-344b877f528e
        name: dgo2-http
        type: http
    lb_virtual_server_ip: 5.5.0.2
    name: cafe-ingress
    rules:
        host: cafe.example.com
        http:
            paths:
                path: /coffee
                    backend:
                        serviceName: coffee-svc
                        servicePort: 80
                    lb_rule:
                        id: 4bc16bdd-abd9-47fb-a09e-21e58b2131c3
                        name: dgo2-default-cafe-ingress/coffee
```

- Get information on all Ingress controllers or a specific one, including controllers that are disabled

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

Example:

```
kubenode> get ingress-controllers
    native-load-balancer:
        ingress_virtual_server:
            http:
                default_backend_tags:
                id: 895c7f43-c56e-4b67-bb4c-09d68459d416
                pool_id: None
            https_terminated:
                default_backend_tags:
                id: 293282eb-f1a0-471c-9e48-ba28d9d89161
                pool_id: None
            lb_ip_pool_id: cc02db70-539a-4934-a938-5b851b3e485b
        loadbalancer_service:
            first_avail_index: 0
            lb_services:
                id: 659eefc6-33d1-4672-a419-344b877f528e
                name: dgo2-bfmxi
                t1_link_port_ip: 100.64.128.5
                t1_router_id: cb50deb2-4460-45f2-879a-1b94592ae886
                virtual_servers:
                    293282eb-f1a0-471c-9e48-ba28d9d89161
                    895c7f43-c56e-4b67-bb4c-09d68459d416
        ssl:
            ssl_client_profile_id: aff205bb-4db8-5a72-8d67-218cdc54d27b
        vip: 5.5.0.2
```

```
    nsx.default.nginx-ingress-rc-host-ed3og
        ip: 10.192.162.201
        mode: hostnetwork
        pool_id: 5813c609-5d3a-4438-b9c3-ea3cd6de52c3


kubenode> get ingress-controller native-load-balancer
    ingress_virtual_server:
        http:
            default_backend_tags:
            id: 895c7f43-c56e-4b67-bb4c-09d68459d416
            pool_id: None
        https_terminated:
            default_backend_tags:
            id: 293282eb-f1a0-471c-9e48-ba28d9d89161
            pool_id: None
    lb_ip_pool_id: cc02db70-539a-4934-a938-5b851b3e485b
        loadbalancer_service:
            first_avail_index: 0
            lb_services:
                id: 659eefc6-33d1-4672-a419-344b877f528e
                name: dgo2-bfmxi
                t1_link_port_ip: 100.64.128.5
                t1_router_id: cb50deb2-4460-45f2-879a-1b94592ae886
                virtual_servers:
                    293282eb-f1a0-471c-9e48-ba28d9d89161
                    895c7f43-c56e-4b67-bb4c-09d68459d416
        ssl:
            ssl_client_profile_id: aff205bb-4db8-5a72-8d67-218cdc54d27b
        vip: 5.5.0.2
```

- 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
```

- Get all ASG caches or a specific one

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

Example:

```
node> get asg-caches
    edc04715-d04c-4e63-abbc-db601a668db6:
        fws_id: 3c66f40a-5378-46d7-a7e2-bee4ba72a4cc
        name: org-85_tcp_80_asg
        rules:
            destinations:
                66.10.10.0/24
            ports:
                80
            protocol: tcp
            rule_id: 4359
        running_default: False
        running_spaces:
            75bc164d-1214-46f9-80bb-456a8fbccbfd
        staging_default: False
        staging_spaces:


node> get asg-cache edc04715-d04c-4e63-abbc-db601a668db6
    fws_id: 3c66f40a-5378-46d7-a7e2-bee4ba72a4cc
    name: org-85_tcp_80_asg
    rules:
        destinations:
            66.10.10.0/24
        ports:
            80
        protocol: tcp
        rule_id: 4359
    running_default: False
    running_spaces:
        75bc164d-1214-46f9-80bb-456a8fbccbfd
    staging_default: False
    staging_spaces:
```

- Get all org caches or a specific one

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

Example:

```
node> get org-caches
    ebb8b4f9-a40f-4122-bf21-65c40f575aca:
        ext_pool_id: 9208a8b8-57d7-4582-9c1f-7a7cefa104f5
        isolation:
            isolation_section_id: d6e7ff95-4737-4e34-91d4-27601897353f
        logical-router: 94a414a2-551e-4444-bae6-3d79901a165f
        logical-switch:
            id: d74807e8-8f74-4575-b26b-87d4fdbafd3c
            ip_pool_id: 1b60f16f-4a30-4a3d-93cc-bfb08a5e3e02
            subnet: 50.0.48.0/24
            subnet_id: a458d3aa-bea9-4684-9957-d0ce80d11788
        name: org-50
        snat_ip: 70.0.0.49
        spaces:
            e8ab7aa0-d4e3-4458-a896-f33177557851


node> get org-cache ebb8b4f9-a40f-4122-bf21-65c40f575aca
    ext_pool_id: 9208a8b8-57d7-4582-9c1f-7a7cefa104f5
    isolation:
        isolation_section_id: d6e7ff95-4737-4e34-91d4-27601897353f
    logical-router: 94a414a2-551e-4444-bae6-3d79901a165f
    logical-switch:
        id: d74807e8-8f74-4575-b26b-87d4fdbafd3c
        ip_pool_id: 1b60f16f-4a30-4a3d-93cc-bfb08a5e3e02
        subnet: 50.0.48.0/24
        subnet_id: a458d3aa-bea9-4684-9957-d0ce80d11788
    name: org-50
    snat_ip: 70.0.0.49
    spaces:
        e8ab7aa0-d4e3-4458-a896-f33177557851
```

- Get all space caches or a specific one

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

Example:

```
node> get space-caches
    global_security_group:
        name: global_security_group
        running_nsgroup: 226d4292-47fb-4c2e-a118-449818d8fa98
        staging_nsgroup: 7ebbf7f5-38c9-43a3-9292-682056722836

    7870d134-7997-4373-b665-b6a910413c47:
        name: test-space1
        org_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
        running_nsgroup: 4a3d9bcc-be36-47ae-bff8-96448fecf307
        running_security_groups:
            aa0c7c3f-a478-4d45-8afa-df5d5d7dc512
```

```
        staging_security_groups:
            aa0c7c3f-a478-4d45-8afa-df5d5d7dc512


node> get space-cache 7870d134-7997-4373-b665-b6a910413c47
    name: test-space1
    org_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
    running_nsgroup: 4a3d9bcc-be36-47ae-bff8-96448fecf307
    running_security_groups:
        aa0c7c3f-a478-4d45-8afa-df5d5d7dc512
    staging_security_groups:
        aa0c7c3f-a478-4d45-8afa-df5d5d7dc512
```

- Get all app caches or a specific one

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

Example:

```
node> get app-caches
     aff2b12b-b425-4d9f-b8e6-b6308644efa8:
         instances:
             b72199cc-e1ab-49bf-506d-478d:
             app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
             cell_id: 0dda88bc-640b-44e7-8cea-20e83e873544
             cif_id: 158a1d7e-6ccc-4027-a773-55bb2618f51b
             gateway_ip: 192.168.5.1
             host_vif: 53475dfd-03e4-4bc6-b8ba-3d803725cbab
             id: b72199cc-e1ab-49bf-506d-478d
             index: 0
             ip: 192.168.5.4/24
             last_updated_time: 1522965828.45
             mac: 02:50:56:00:60:02
             port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
             state: RUNNING
             vlan: 3
         name: hello2
         rg_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
         space_id: 7870d134-7997-4373-b665-b6a910413c47


node> get app-cache aff2b12b-b425-4d9f-b8e6-b6308644efa8
    instances:
        b72199cc-e1ab-49bf-506d-478d:
        app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
        cell_id: 0dda88bc-640b-44e7-8cea-20e83e873544
        cif_id: 158a1d7e-6ccc-4027-a773-55bb2618f51b
        gateway_ip: 192.168.5.1
        host_vif: 53475dfd-03e4-4bc6-b8ba-3d803725cbab
        id: b72199cc-e1ab-49bf-506d-478d
        index: 0
        ip: 192.168.5.4/24
        last_updated_time: 1522965828.45
```

```
        mac: 02:50:56:00:60:02
        port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
        state: RUNNING
        vlan: 3
    name: hello2
    org_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
    space_id: 7870d134-7997-4373-b665-b6a910413c47
```

- Get all instance caches of an app or a specific instance cache

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

Example:

```
node> get instance-caches aff2b12b-b425-4d9f-b8e6-b6308644efa8
    b72199cc-e1ab-49bf-506d-478d:
        app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
        cell_id: 0dda88bc-640b-44e7-8cea-20e83e873544
        cif_id: 158a1d7e-6ccc-4027-a773-55bb2618f51b
        gateway_ip: 192.168.5.1
        host_vif: 53475dfd-03e4-4bc6-b8ba-3d803725cbab
        id: b72199cc-e1ab-49bf-506d-478d
        index: 0
        ip: 192.168.5.4/24
        last_updated_time: 1522965828.45
        mac: 02:50:56:00:60:02
        port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
        state: RUNNING
        vlan: 3


node> get instance-cache aff2b12b-b425-4d9f-b8e6-b6308644efa8 b72199cc-e1ab-49bf-506d-478d
    app_id: aff2b12b-b425-4d9f-b8e6-b6308644efa8
    cell_id: 0dda88bc-640b-44e7-8cea-20e83e873544
    cif_id: 158a1d7e-6ccc-4027-a773-55bb2618f51b
    gateway_ip: 192.168.5.1
    host_vif: 53475dfd-03e4-4bc6-b8ba-3d803725cbab
    id: b72199cc-e1ab-49bf-506d-478d
    index: 0
    ip: 192.168.5.4/24
    last_updated_time: 1522965828.45
    mac: 02:50:56:00:60:02
    port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
    state: RUNNING
    vlan: 3
```

- Get all policy caches

```
get policy-caches
```

Example:

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

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

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.

  ```
  (NCP 2.1, 2.1.0.1) get app-cache <app-name>
  (NCP 2.1, 2.1.0.1) get app-caches
  (NCP 2.1.2) get container-cache <container-name>
  (NCP 2.1.2) get container-caches
  ```

Example:

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


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

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:

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

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

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


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

- 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
```