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

Modified on 7 SEP 2017

VMware NSX-T 2.0

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

Contents

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

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

Intended Audience

This guide is intended for system and network administrators. A familiarity with networking and virtualization technology is assumed.

VMware Technical Publications Glossary

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

Overview of NSX-T Container Plug-in

1

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

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

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

In this release, NCP supports a single OpenShift cluster.

This chapter includes the following topics:

Compatibility Requirements

NSX-T Container Plug-in has the following compatibility requirements.

Software Product	Version
Hypervisor for Container Host VMs	■ ESXi 6.5 ■ ESXi 6.5 Update 1 ■ RHEL KVM 7.3
Container Host Operating System	RHEL 7.3
Platform as a Service	OpenShift 3.5, 3.6
Guest vSwitch	OVS 2.6, 2.7

Installation Overview

Installing and configuring NCP involves the following steps.

1 Install NSX-T.

2. Create an overlay transport zone.
3. Create an overlay logical switch and connect the nodes to the switch.
4. Create a tier-0 logical router.
5. Create IP blocks for the pods.
6. Create IP blocks or IP pools for SNAT (source network address translation).
7. Deploy OpenShift VMs.
8. Prepare the Ansible hosts file.
9. Install CNI plug-in and OVS (Open vSwitch) on each node.
10. Install OpenShift Origin.
11. Run NCP and NSX node agent.

Setting Up NSX-T Resources 2

NSX-T resources must be created to provide networking to OpenShift nodes. You can configure these resources manually using the NSX Manager GUI, or automate the process using an Ansible playbook.

This chapter describes creating the NSX-T resources manually. To automate the process, see Install CNI Plug-in and OVS. Note that if you create the resources using a playbook, for SNAT, you can only create an IP block, whereas if you create the resources manually, you can create either an IP block or IP pool for SNAT.

This chapter includes the following topics:

Configuring NSX-T Resources

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

Overlay Transport Zone

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

Tier-0 Logical Routing

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

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

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

Logical Switch

The interface used by the pods for network traffic must be connected to an overlay logical switch. It is not mandatory for the node's management interface to be connected to NSX-T, although doing so will make setting up easier. The switch is identified by the tag `{'ncp/cluster': '<cluster_name>'}`. Log in to NSX Manager and navigate to **Switching > Switches**. You can either create a new switch or use an existing one. After you identify the switch, tag it with `{'ncp/cluster': '<cluster_name>'}`.

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

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

IP Blocks for Kubernetes Pods

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

You can also create IP blocks specifically for no-SNAT namespaces. These IP blocks require the tag `{'ncp/no_snat': '<cluster_name>'}` in addition to the `ncp/cluster` tag. If you create no-SNAT IP blocks while NCP is running, you must restart NCP. Otherwise, NCP will keep using the shared IP blocks until they are exhausted.

IP Block or IP Pool for SNAT

These resources will be used for allocating IP addresses which will be used for translating Pod IPs via SNAT rules, and for exposing ingress controllers via SNAT/DNAT rules - just like Openstack floating IPs. In this guide, these IP addresses are also referred to as *external IPs*. Users can either configure a *global* external IP block or a cluster specific external IP pool.

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

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

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

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

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

(Optional) Firewall Marker Section

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

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

Create and Configure a Tier-0 Logical Router

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

Procedure

1 From a browser, log in to NSX Manager at https://*nsx-manager-ip-address*.

2 Navigate to **Routing > Routers** and click **Add > Tier-0 Router**.

3 Enter a name and optionally a description.

4 Select an existing edge cluster from the drop-down menu to back this tier-0 logical router.

5 Select a high-availability mode.

  Select active-standby.

6 Click **Save**.

  The new logical router appears as a link.

7 Click the logical router link.

8 Click **Routing > Route Redistribution**.

9 Click **Add** to a new redistribution criterion.

  For sources, in a routed (non-NAT) topology, select **NSX Static**. In a NAT topology, select **Tier-0 NAT**.

10 Click **Save**.

11 Click the newly created router.

12 Click **Configuration > Router Ports**

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

14 Select a transport node.

15 Select the logical switch that was previously created.

16 Specify an IP address in your external network.

17 Click **Save**.

    The new logical router appears as a link.

Setting Up NSX-T Container Plug-in and OpenShift

3

This chapter describes installing and configuring NSX-T Container Plug-in (NCP) and OpenShift.

This chapter includes the following topics:

Deploy OpenShift VMs

Before installing NSX-T Container Plug-in, OpenShift must be installed.

The OpenShift VMs should run RHEL 7.3 or CentOS 7.3 (or later). You must deploy at least one master.

Procedure

1 On the master where Ansible playbooks will be run, run the following install commands:

```
yum install epel-release
yum -y install git ansible httpd-tools
```

2 Install Docker and configure Docker storage. For instructions, see https://docs.openshift.com/container-platform/3.6/install_config/install/host_preparation.html.

What to do next

Prepare the Ansible hosts file. See Prepare the Ansible Hosts File.

Prepare the Ansible Hosts File

The Ansible `hosts` file defines the nodes in the OpenShift cluster.

Procedure

1. Clone the NCP GitHub repository at https://github.com/vmware/nsx-openshift. The `hosts` file is in the `openshift-ansible-nsx` directory.
2. In the [masters] and [nodes] sections, specify the host names and IP addresses of the OpenShift VMs. For example,

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

       [single_master]
       admin.rhel.osmaster ansible_ssh_host=101.101.101.4

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

       [etcd]

       [OSEv3:children]
       masters
       nodes
       etcd
   ```

   Note that `openshift_ip` identifies the cluster internal IP and needs to be set if the interface to be used is not the default one. The `single_master` variable is used by ncp-related roles from a master node to perform certain tasks only once, e.g. NSX-T management plane resource configuration.
3. Set up SSH access so that all the nodes can be accessed without password from the node where the Ansible role is run (typically it is the master node):

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

4. Update the [OSEv3:vars] section. Details about all the parameters can be found in the OpenShift Origin Documentation for the Advanced Installation (https://docs.openshift.com/container-platform/3.6/install_config/install/advanced_install.html). For example,

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

       os_sdn_network_plugin_name=cni
       openshift_use_openshift_sdn=false
       openshift_node_sdn_mtu=1500

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

       deployment_type=origin
   ```

```
    # uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
    openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]
    openshift_master_htpasswd_file=<enter_full_path_here>/openshift-ansible-nsx/files/htpasswd

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

    os_sdn_network_plugin_name
      Set to 'cni' for the NSX Integration

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

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

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

    deployment_type
      Set to origin or openshift-enterprise for the open source or Enterprise version
      of OpenShift respectively

    openshift_master_htpasswd_file
      This file is holding the htpasswd password file. You will need to fix the
      path to it for the deployment to work, so exchange <enter_full_path_here>
      with your 'real path'. You need to install htpasswd and create password
      file with it.
```

5 Check that you have connectivity to all hosts:

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

The results should look like the following. If not, resolve the connectivity problem.

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

What to do next

Install CNI plug-in and OVS. See Install CNI Plug-in and OVS.

Install CNI Plug-in and OVS

The container network interface (CNI) plug-in and Open vSwitch (OVS) must be installed on the OpenShift nodes. The installation is performed by running an Ansible playbook.

The playbook contains instructions to configure NSX-T resources for the nodes. You can also configure the NSX-T resources manually as described in Chapter 2 Setting Up NSX-T Resources. The parameter `perform_nsx_config` indicates whether or not to configure the resources when the playbook is run.

Procedure

1 Update the parameter values in `roles/ncp_prep/default/main.yaml` and `roles/nsx_config/default/main.yaml`, including the URLs where CNI plugin RPM, OVS and its corresponding kernel module RPM can be downloaded. In addition, uplink_port is the name of the uplink port VNIC on the Node VM. The remaining variables pertain to the NSX management plane configuration, including:

   - `perform_nsx_config`: whether to perform the resource configuration. Set it to false if the configuration will be done manually, and nsx_config script will not be run.
   - `nsx_config_script_path`: absolute path of the nsx_config.py script.
   - `nsx_cert_file_path`: absolute path of NSX client certificate file.
   - `nsx_manager_ip`: IP of NSX manager.
   - `nsx_edge_cluster_name`: name of the Edge Cluster to be used by the T0 router
   - `nsx_transport_zone_name`: name of the Overlay Transport Zone
   - `nsx_t0_router_name`: name of Tier-0 Logical Router for the cluster
   - `pod_ipblock_name`: name of IP block for pods.
   - `pod_ipblock_cidr`: CIDR address for this IP block
   - `snat_ipblock_name`: name of the IP block for SNAT
   - `snat_ipblock_cidr`: CIDR address for this IP block
   - `os_cluster_name`: name of the OpenShift cluster
   - `vnic_mac_list`: comma-separated list of MAC addresses the nodes
   - `os_node_name_list`: comma-separated list of node names, the ordering must match the above list's.
   - `nsx_node_ls_name`: name of Logical Switch connected to the nodes

   The names must match the NST-T resources you created. Otherwise the resources will be created with the names specified. The playbook is idempotent, and ensures resources with specified names exist and are tagged accordingly.

2 Change to the `openshift-ansible-nsx` directory and run the `ncp_prep` role.

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

The playbook contains instructions to perform the following actions:

- Download the CNI plug-in installation file.

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

- Install the CNI plug-in installation file.

  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.

- Download and install the OVS installation files.

  The files are `openvswitch-2.7.0.xxxxxxx-1.x86_64.rpm` and `openvswitch-kmod-2.7.0.xxxxxxx-1.el7.x86_64.rpm`, where *xxxxxxx* is the build number.

- Make sure that OVS is running.

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

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

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

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

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

What to do next

Install OpenShift origin. See Install OpenShift Origin.

Install OpenShift Origin

Origin is the upstream community project that powers OpenShift.

Procedure

1 Clone the openshift-ansible repository. For example,

```
git clone https://github.com/openshift/openshift-ansible.git
```

2 Run the `playbooks/byo/config.yml` playbook from the above repository, referencing the hosts file that was created earlier.

```
        ansible-playbook -i /PATH/TO/HOSTS/hosts openshift-ansible/playbooks/byo/config.yml
```

This can take 20 to 30 minutes in a setup with 1 master and 2 nodes. When it completes, anOpenShift cluster will be created with masters and nodes configured as specified in the hosts file.

What to do next

Run NCP and NSX node agent. See Run NCP and NSX Node Agent.

Run NCP and NSX Node Agent

Set up and run NCP and NSX node agent.

Procedure

1 Edit `roles/ncp/defaults/main.yaml` and specify the OpenShift API server IP, NSX manager IP, and URL'sfor downloading NCP ReplicationController yaml and nsx-node-agent DaemonSet yaml.

2 From the openshift-ansible-nsx directory, run the ncp role:

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

The ncp role performs the following steps:

- Check if nsx-system project exists, and create one if it does not.

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

- Check if a default service account exists, and create one if it does not.

```
    oc create serviceaccount default
```

- Add cluster-admin role to the above service account, to be used by user to create NCP and nsx-node-agent pods.

```
    oadm policy add-cluster-role-to-user cluster-admin system:serviceaccount:nsx-system:default
```

- Obtain the token associated with the above service account, and store it under `/etc/nsx-ujo/default_token`.

```
    secret=`kubectl get serviceaccount default -o yaml | grep -A1 secrets | tail -n1 | awk {'print 
$3'}`

    kubectl get secret $secret -o yaml | grep 'token:' | awk {'print $2'} | base64 -d > /etc/nsx-
ujo/default_token
```

- Download the SecurityContextConstraint (SCC) yaml file for NCP and create the SCC based on the yaml.
- Add the created SCC to current user.

  ```
  oadm policy add-scc-to-user ncp-scc -z default
  ```

- Download the YAML files for NCP ReplicationController (RC) and nsx-node-agent DaemonSet (DS) and update the ConfigMap.
- Download and load the NCP image (nsx-node-agent uses the same image).
- Configure the service account and set up the required SecurityContextConstraint for NCP and nsx_node_agent.
- Create the NCP ReplicationController and nsx-node-agent DaemonSet.

Setup Notes

Before setting up OpenShift and NCP, take note of the following information.

- The names of OpenShift Namespace, Pod and Network Policy resource, OpenShift Cluster (as specified in ncp.ini) and Node, must not be longer than 40 characters.
- Label key and value must not be longer than 20 and 40 characters, respectively.
- A pod must have no more than 11 labels and a namespace must have no more than 12 labels.
- Labels added for OpenShift internal usage, for example, a label with prefix openshift.io in its key, will be disregarded by NCP and thus user won't see the corresponding tags created on the related NSX resources. Here is a list of label prefixes used by OpenShift, and you should avoid using a label key starting with any of the following:

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

- The nodes will need access to the pods, for example, for Kubelet health-checks. Make sure the host management interface is able to access the pod network.
- 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, with restricted and anyuid SCC, NET_ADMIN is not granted. Be wary of any SCC that enables NET_ADMIN explicitly, or enables the pod to run in privileged mode. In addition, for untrusted containers, create a separate SCC based on, for example, anyuid SCC, with NET_RAW capability removed. This can be done by adding NET_RAW to `requiredDropCapabilities` list in the SCC definition.

- Allow root access in PODs/Containers (only for testing). Commands below will require root access in all PODs of the oc project you are currently logged in to.

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

- Configure (add) the OpenShift Registry.

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

- Delete the OpenShift Registry

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

- There's a missing IPtables firewall rule to allow DNS requests from the Docker default bridge containers to the dnsmasq process on the Node. It needs to be opened manually. Edit `/etc/sysconfig/iptables` and add the following Rules at the bottom of the file before COMMIT:

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

- Restart iptables, docker and origin-node (restarts kube-proxy and kubelet).

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

- The internal docker registry of OpenShift needs to be allowed to use non-TLS for OpenShift to work. Normally this should be added automatically by the OpenShift Ansible installer, but it seems that this is currently not working. Edit /etc/sysconfig/docker and add:

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

- Restart Docker.

```
systemctl restart docker
```

- Configure (add) the OpenShift routers (HA-Proxy N/S LBs).

```
    oc login -u system:admin -n default
    oadm router router --replicas=2 --service-account=router
```

- Delete the created routers.

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

- Create a sample Ruby based 2 tier apps.

  ```
  oc login -u system:admin -n default
  oc oc new-project nsx
  oc process -n openshift mysql-ephemeral -v DATABASE_SERVICE_NAME=database | oc create -f -
  oc new-app centos/ruby-22-centos7~https://github.com/openshift/ruby-hello-world.git
  oc expose service ruby-hello-world
  oc env dc database --list | oc env dc ruby-hello-world -e -
  ```

Administering NSX-T Container Plug-in

4

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

This chapter includes the following topics:

Manage IP Blocks from the NSX Manager GUI

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

Procedure

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

2 Select **DDI**.

   A list of the existing IP blocks is displayed.

3 Perform any of the following actions.

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

   You cannot delete an IP block that has subnets allocated.

Manage IP Block Subnets from the NSX Manager GUI

You can add and delete subnets for an IP block from the NSX Manager GUI.

Procedure

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

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

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

CIF-Attached Logical Ports

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

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

Managing CIF-Attached Logical Ports

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

Network Monitoring Tools

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

- Traceflow
- Port Connection
- IPFIX

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

Distributed network encryption is not supported in this release.

CLI Commands

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

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

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

Table 4-1. CLI Commands for the NCP Container

Type	Command
Status	get ncp-nsx status
Status	get ncp-k8s-api-server status
Status	get ncp-watcher <watcher-name>
Status	get ncp-watchers
Cache	get project-cache <project-name>
Cache	get project-caches
Cache	get namespace-cache <namespace-name>
Cache	get namespace-caches
Cache	get pod-cache <pod-name>
Cache	get pod-caches
Cache	get ingress-caches
Support	get support-bundle file <filename>
Support	get ncp-log file <filename>
Support	get node-agent-log file <filename>
Support	get node-agent-log file <filename> <node-name>

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

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

Table 4-3. CLI Commands for the NSX Kube Proxy Container

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

Status Commands for the NCP Container

- Show the connection status between NCP and NSX Manager

  ```
  get ncp-nsx status
  ```

  Example:

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

- Show the connection status between NCP and Kubernetes API server

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

  Example:

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

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

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

  Example 1:

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

Example 2:

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

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

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

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

Cache Commands for the NCP Container

- Get the internal cache for projects or namespaces

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

Example 1:

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

Example 2:

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

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

- Get the internal cache for pods

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

Example 1:

```
kubenode> get pod-cache nsx.default.nginx-rc-uq2lv
    cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
    gateway_ip: 10.0.0.1
    ingress_controller: False
    ip: 10.0.0.2/24
    labels:
        app: nginx
    mac: 02:50:56:00:08:00
    port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
    vlan: 1
```

Example 2:

```
kubenode> get pod-caches
    nsx.default.nginx-rc-uq2lv:
        cif_id: 2af9f734-37b1-4072-ba88-abbf935bf3d4
        gateway_ip: 10.0.0.1
        ingress_controller: False
        ip: 10.0.0.2/24
        labels:
            app: nginx
        mac: 02:50:56:00:08:00
        port_id: d52c833a-f531-4bdf-bfa2-e8a084a8d41b
        vlan: 1
```

- Get the internal cache for ingress

```
get ingress caches
```

Example:

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

Support Commands for the NCP Container

- Save the NCP support bundle in the filestore

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

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

Example:

```
kubenode>get support-bundle file foo
Bundle file foo created in tgz format
kubenode>copy file foo url scp://nicira@10.0.0.1:/tmp
```

- Save the NCP logs in the filestore

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

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

Example:

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

- Save the node agent logs in the filestore

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

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

  Example:

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

Status Commands for the NSX Node Agent Container

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

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

  Example:

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

Cache Commands for the NSX Node Agent Container

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

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

  Example 1:

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

Example 2:

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

Status Commands for the NSX Kube-Proxy Container

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

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

  Example:

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

- Show the Kube Proxy watcher status

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

  Example 1:

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

  Example 2:

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

       service:
  ```

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

- Dump OVS flows on a node

  ```
  dump ovs-flows
  ```

  Example:

  ```
  kubenode> dump ovs-flows
      NXST_FLOW reply (xid=0x4):
      cookie=0x0, duration=8.876s, table=0, n_packets=0, n_bytes=0, idle_age=8, priority=100,ip
  actions=ct(table=1)
      cookie=0x0, duration=8.898s, table=0, n_packets=0, n_bytes=0, idle_age=8, priority=0
  actions=NORMAL
      cookie=0x0, duration=8.759s, table=1, n_packets=0, n_bytes=0, idle_age=8,
  priority=100,tcp,nw_dst=10.96.0.1,tp_dst=443 actions=mod_tp_dst:443
      cookie=0x0, duration=8.719s, table=1, n_packets=0, n_bytes=0, idle_age=8,
  priority=100,ip,nw_dst=10.96.0.10 actions=drop
      cookie=0x0, duration=8.819s, table=1, n_packets=0, n_bytes=0, idle_age=8,
  priority=90,ip,in_port=1 actions=ct(table=2,nat)
      cookie=0x0, duration=8.799s, table=1, n_packets=0, n_bytes=0, idle_age=8, priority=80,ip
  actions=NORMAL
      cookie=0x0, duration=8.856s, table=2, n_packets=0, n_bytes=0, idle_age=8, actions=NORMAL
  ```