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

Modified on 18 JULY 2018
VMware NSX-T 2.2.1
VMware NSX-T 2.2

vmware®

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

https://docs.vmware.com/

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

docfeedback@vmware.com

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

Contents

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

This guide describes how to install and administer NSX-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 the installation and administration of NSX-T and OpenShift is assumed.

VMware Technical Publications Glossary

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

Overview of NSX-T Container Plug-in

1

NSX-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.

NCP provides the following functionalities:

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

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

- Implements OpenShift network policies with NSX-T distributed firewall.
  - Support for ingress and egress network policies.
  - Support for `IPBlock` selector in network policies.
  - Support for `matchLabels` and `matchExpression` when specifying label selectors for network policies.
- Implements OpenShift route with NSX-T layer 7 load balancer.
  - Support for HTTP route and HTTPS route with TLS edge termination.
  - Support for routes with alternate backends and wildcard subdomains.
- Creates tags on the NSX-T logical switch port for the namespace, pod name, and labels of a pod, and allows the administrator to define NSX-T security groups and policies based on the tags.

In this release, NCP supports a single OpenShift cluster.

This chapter includes the following topics:

Compatibility Requirements

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

Software Product	Version
NSX-T	2.1, 2.2
Hypervisor for Container Host VMs	■ Supported vSphere version ■ RHEL KVM 7.4, 7.5
Container Host Operating System	RHEL 7.4, 7.5
Platform as a Service	OpenShift 3.7, 3.9
Guest vSwitch	OVS 2.8.1 (packaged with NSX-T 2.1), 2.9.1 (packaged with NSX-T 2.2)

Installation Overview

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

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 pools for SNAT (source network address translation).

7 Deploy OpenShift VMs.

8 Prepare the Ansible hosts file.

9 (Option 1) Install NCP and OpenShift using a single playbook.

(Option 2) Install CNI plug-in, OVS (Open vSwitch), and NCP docker image. Then Install OpenShift Container Platform.

10 Run NCP and NSX node agent.

Steps 2 to 6 are not needed if you install NCP using playbooks that are provided. See Install NCP and OpenShift Using a Single Playbook and Install CNI Plug-in, OVS, and NCP Docker Image.

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 section describes creating the NSX-T resources manually. To automate the process, see Install CNI Plug-in, OVS, and NCP Docker Image.

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.

You 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.

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.

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 have the following tags:

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

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

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

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

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

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

IP Pool for SNAT

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

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

Log in to NSX Manager and navigate to **Inventory > Groups > IP POOL** to create a pool or find an existing pool.

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

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

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

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

- The 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.

In release NCP 2.2.1 and later, you can specify which namespace can be allocated IPs from the SNAT IP pool by adding the following tag to the IP pool.

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

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

```
oc get ns –o yaml
```

Note the following:

- Each tag should specify one UUID. You can create multiple tags for the same pool.
- If you change the tags after some namespaces have been allocated IPs based on the old tags, those IPs will not be reclaimed until the SNAT configurations of the services change or NCP restarts..
- If the IP pool does not have any owner tags, all services can be allocated IPs from the pool.

(Optional) Firewall Marker Sections

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

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

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

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.

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. You must deploy at least one master.

For more information, see https://docs.openshift.com.

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-integration-for-openshift. The `hosts` file is in the `openshift-ansible-nsx` directory. You must keep the `hosts` file in the `openshift-ansible-nsx` directory. Some playbooks assume that this is the path for the `hosts` file.

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 Container Platform Documentation for the Advanced Installation (search for "advanced installation" in https://docs.openshift.com). For example,

```
    # Set the default route fqdn
    openshift_master_default_subdomain=apps.corp.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

    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 openshift-enterprise

    openshift_hosted_manage_registry
      Set to false to disable auto creation of registry

    openshift_hosted_manage_router
      Set to false to disable auto creation of router

    openshift_enable_service_catalog
      Set to false to disable service_catalog

    (For OpenShift 3.9 only) skip_sanity_checks
      Set to true

    (For OpenShift 3.9 only) openshift_web_console_install
      Set to false
```

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, OVS, and NCP Docker Image.

Install NCP and OpenShift Using a Single Playbook

You can install NCP and OpenShift using a single playbook or perform the installation in separate steps.

The single Ansible playbook `install.yaml` performs the following tasks:

- NCP preparation
- OpenShift installation
- NCP installation

Alternatively, you can install NCP and OpenShift using the instructions in the following two sections: Install CNI Plug-in, OVS, and NCP Docker Image and Install OpenShift Container Platform.

Before running the `install.yaml` playbook, set the required and optional parameters for the `ncp_prep` and `ncp` plabook roles. The parameters are described in Install CNI Plug-in, OVS, and NCP Docker Image.

The following command runs the playbook:

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

Install CNI Plug-in, OVS, and NCP Docker Image

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

Note This step is not necessary if you install NCP and OpenShift using a single playbook. See Install NCP and OpenShift Using a Single 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-T management plane configuration.

   Parameters that need to be specified:

   - `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_manager_ip`: IP of NSX Manager
   - `nsx_edge_cluster_name`: name of the Edge Cluster to be used by the tier-0 router
   - `nsx_transport_zone_name`: name of the overlay Transport Zone
   - `os_node_name_list`: comma-separated list of node names

     For example, node1,node2,node3

- `subnet_cidr`: CIDR address for IP administrator will assign to br-int on the node
- `vc_host`: IP address of vCenter Server
- `vc_user`: user name of vCenter Server administrator
- `vc_password`: password of vCenter Server administrator
- `vms`: comma-separated list of VM names. The order must match `os_node_name_list`.

The following parameters have default values. You can modify them as needed.

- `nsx_t0_router_name`: name of tier-0 Logical Router for the cluster. Default: **`t0`**
- `pod_ipblock_name`: name of IP block for pods. Default: **`podIPBlock`**
- `pod_ipblock_cidr`: CIDR address for this IP block. Default: **`172.20.0.0/16`**
- `snat_ippool_name`: name of the IP block for SNAT. Default is `externalIP`.
- `snat_ippool_cidr`: CIDR address for this IP block. Default: **`172.30.0.0/16`**
- `start_range`: the start IP address of CIDR specified for this IP pool. Default: **`172.30.0.1`**
- `end_range`: the end IP address of CIDR specified for this IP pool. Default: **`172.30.255.254`**
- `os_cluster_name`: name of the OpenShift cluster. Default: **`occl-one`**
- `nsx_node_ls_name`: name of Llogical switch connected to the nodes. Default: **`node_ls`**
- `nsx_node_lr_name`: name of logical router for the switch **`node_ls`**. Default: **`node_lr`**

The `nsx-config` playbook supports creating only one IP pool and one IP block. If you want more, you must create them manually.

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.
- Download the NCP tar file and load the Docker image from the tar file.
- Download the `ncp-rbac` yaml file and change the `apiVersion` to **v1**.
- Create a logical topology and related resources in NSX-T, and create tags on them so that they can be recognized by NCP.
- Update `ncp.ini` with NSX-T resource information.

What to do next

Install OpenShift Container Platform. See Install OpenShift Container Platform.

Install OpenShift Container Platform

OpenShift Container Platform (OCP) is a platform as a service (PaaS) product that brings together Docker and Kubernetes.

Note This step is not necessary if you install NCP and OpenShift using a single playbook. See Install NCP and OpenShift Using a Single Playbook.

For information about installing OCP, see https://docs.openshift.com.

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
```

- Download the `ncp-rbac` yaml file and change the `apiVersion` to **v1**.
- Create the service account for the NCP pod, create a cluster role that specifies resources that NCP can access and bind the cluster role with the NCP service account.
- Create the service account for the nsx-node-agent pod, create a cluster role that specifies the resources that the node agent can access and bind the cluster role with the node agent service account.

```
    oc apply -f /tmp/ncp-rbac.yml
```

- Obtain the token associated with the above service accounts, and store it under `/etc/nsx-ujo/<service_account>_token`:

```
    secret=`kubectl get serviceaccount ncp-svc-account -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/ncp_token
    secret=`kubectl get serviceaccount nsx-node-agent-svc-account -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/node_agent_token
```

- Download the SecurityContextConstraint (SCC) yaml file for NCP and create the SCC based on the yaml.
- Add the created SCC to the above service accounts:

```
    oadm policy add-scc-to-user ncp-scc -z ncp-svc-account
    oadm policy add-scc-to-user ncp-scc -z nsx-node-agent-svc-account
```

- 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.

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.

Setup Notes

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

- 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.
- There is a defect in the Ansible playbook for upgrading OpenShift 3.6 to 3.7. It ignores the `openshift_use_openshift_sdn=false` setting in the inventory file and tries to install the OpenShift default SDN during the upgrade. To work around this issue, remove the CNI plug-in before the upgrade and re-install it after the upgrade.
- 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 is 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
  ```

- NCP's support for network policies is the same as the support provided by Kubernetes and depends on the Kubernetes version used by OpenShift.
  - OpenShift 3.7, 3.9 - The rule clauses in the network policy may contain at most one selector from `namespaceSelector`, `podSelector` and `ipBlock`.
  - OpenShift 3.7 - Support for egress rules and `ipBlock` CIDR selector is not available. They are available as beta fields in OpenShift 3.9.
- Certain versions of Kubernetes has a `subPath`-related issue (see https://github.com/kubernetes/kubernetes/issues/61076). If the OpenShift version does not contain a fix for this issue, the creation of the NCP pod will fail with the error `CreateContainerConfigError: failed to prepare subPath for volumeMount`. You can work around this problem by removing the use of `subPath` from the NCP yaml. Specifically, remove the line containing `subPath: ncp.ini` and replace the configuration for `volumes` with the following:

  ```
  volumes:
    - name: config-volume
      # ConfigMap nsx-ncp-config is expected to supply ncp.ini
      configMap:
        name: nsx-ncp-config
        items:
          - key: ncp.ini
            path: ncp.ini
  ```

A side effect of this change is that the entire `/etc/nsx-ujo` directory becomes read-only. As a result, connecting with NSX-T using certificate and private key will not work because NCP will not be able to create a temporary file under `/etc/nsx-ujo` to move both certificate and private key into a single file.

Load Balancing

4

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

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

Configuring Load Balancing

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

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

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

1 Set `use_native_loadbalancer` = `True`.

2 Set `pool_algorithm` to `WEIGHTED_ROUND_ROBIN`.

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

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

Layer 7 Load Balancer Example

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

```
# RC
apiVersion: v1
kind: ReplicationController
metadata:
  name: tea-rc
spec:
  replicas: 2
  template:
    metadata:
       labels:
         app: tea
    spec:
      containers:
      - name: tea
        image: nginxdemos/hello
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: coffee-rc
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
      - name: coffee
        image: nginxdemos/hello
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
---
# Services
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
    app: tea
spec:
  ports:
  - port: 80
    targetPort: 80
```

```
    protocol: TCP
    name: http
  selector:
    app: tea
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
  labels:
    app: coffee
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
    name: http
  selector:
    app: coffee
---
# Routes
apiVersion: v1
kind: Route
metadata:
  name: cafe-route-multi
spec:
  host: www.cafe.com
  path: /drinks
  to:
    kind: Service
    name: tea-svc
    weight: 1
  alternateBackends:
  - kind: Service
    name: coffee-svc
    weight: 2
---
apiVersion: v1
kind: Route
metadata:
  name: cafe-route
spec:
  host: www.cafe.com
  path: /tea-svc
  to:
    kind: Service
    name: tea-svc
    weight: 1
```

Additional Notes

- In this release, load balancing is supported only through the OpenShift route resource, but not the service resource of type `LoadBalancer`.

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

  The small NSX-T load balancer 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 Route resources.
  - A total of 30 endpoints referenced by the LoadBalancer services and Route resources.

  The medium NSX-T load balancer 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 Route resources.
  - A total of 300 endpoints referenced by the LoadBalancer services and Route resources.

  The large NSX-T load balancer 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 Route resources.

- A total of 3000 endpoints referenced by the LoadBalancer services and Route 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.

Sample Script to Generate a CA-Signed Certificate

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

```
#!/bin/bash
host="www.example.com"
filename=server

openssl genrsa -out ca.key 4096
openssl req -key ca.key -new -x509 -days 365 -sha256 -extensions v3_ca -out ca.crt -subj
"/C=US/ST=CA/L=Palo Alto/O=OS3/OU=Eng/CN=${host}"
openssl req -out ${filename}.csr -new -newkey rsa:2048 -nodes -keyout ${filename}.key -subj
"/C=US/ST=CA/L=Palo Alto/O=OS3/OU=Eng/CN=${host}"
openssl x509 -req -days 360 -in ${filename}.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out $
{filename}.crt -sha256
```

Mount the Default Certificate and Key into the NCP Pod

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

```
spec:
  ...
  containers:
  - name: nsx-ncp
    ...
    volumeMounts:
    ...
    - name: lb-default-cert
      # Mount path must match nsx_v3 option "lb_default_cert_path"
      mountPath: /etc/nsx-ujo/lb-default.crt
    - name: lb-priv-key
      # Mount path must match nsx_v3 option "lb_priv_key_path"
      mountPath: /etc/nsx-ujo/lb-default.key
  volumes:
  ...
  - name: lb-default-cert
    hostPath:
      path: /etc/nsx-ujo/lb-default.crt
  - name: lb-priv-key
    hostPath:
      path: /etc/nsx-ujo/lb-default.key
```

Administering NSX-T Container Plug-in

5

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 5-1. CLI Commands for the NCP Container

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

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

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

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

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

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

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

Status Commands for the NCP Container

- Show the status of the NCP master

  ```
  get ncp-master status
  ```

  Example:

  ```
  kubenode> get ncp-master status
  This instance is not the NCP master
  Current NCP Master id is a4h83eh1-b8dd-4e74-c71c-cbb7cc9c4c1c
  Last master update at Wed Oct 25 22:46:40 2017
  ```

- Show the connection status between NCP and NSX Manager

  ```
  get ncp-nsx status
  ```

Example:

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

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

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

Example 1:

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

Example 2:

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

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

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

```
        Watcher thread status: Up

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

- Show the connection status between NCP and Kubernetes API server

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

Example:

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

- Check all projects or a specific one

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

Example:

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

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

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

Cache Commands for the NCP Container

- Get the internal cache for projects or namespaces

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

Example:

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

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

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

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

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

    kube-system:
        logical-router: 5032b299-acad-448e-a521-19d272a08c46
        logical-switch:
            id: 85233651-602d-445d-ab10-1c84096cc22a
            ip_pool_id: ab1c5b09-7004-4206-ac56-85d9d94bffa2
```

```
            subnet: 10.0.1.0/24
            subnet_id: 73e450af-b4b8-4a61-a6e3-c7ddd15ce751

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

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

- Get the internal cache for pods

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

Example:

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

    nsx.testns.web-pod-1:
        cif_id: ce134f21-6be5-43fe-afbf-aaca8c06b5cf
        gateway_ip: 50.0.2.1
        host_vif: d6210773-5c07-4817-98db-451bd1f01937
        id: 3180b521-270e-11e8-ab42-005056b198fb
        ingress_controller: False
        ip: 50.0.2.3/24
        labels:
```

```
            app: nginx-new
            role: db
            tier: cache
        mac: 02:50:56:00:20:02
        port_id: 81bc2b8e-d902-4cad-9fc1-aabdc32ecaf8
        vlan: 3

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

- Get network policy caches or a specific one

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

Example:

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

```
                        key: tier
                        operator: DoesNotExist
                    matchLabels:
                        ns: myns
            ports:
                port: 80
                protocol: TCP
        src_rules_hash: e4ea7b8d91c1e722670a59f971f8fcc1a5ac51f1


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

Support Commands for the NCP Container

- Save the NCP support bundle in the filestore

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

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

Example:

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

- Save the NCP logs in the filestore

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

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

  Example:

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

- Save the node agent logs in the filestore

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

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

  Example:

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

- Get and set the log level

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

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

Example:

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

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

Status Commands for the NSX Node Agent Container

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

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

Example:

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

Cache Commands for the NSX Node Agent Container

- Get the internal cache for NSX node agent containers.

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

Example 1:

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

Example 2:

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

Status Commands for the NSX Kube-Proxy Container

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

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

Example:

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

- Show the Kube Proxy watcher status

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

Example 1:

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

Example 2:

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

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

- Dump OVS flows on a node

```
dump ovs-flows
```

Example:

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

Error Codes

This section lists error codes produced by the various components.

NCP Error Codes

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

Error Code	Description
NCP00019	IP subnet deletion failed
NCP00020	IP pool creation failed
NCP00021	IP pool deletion failed
NCP00022	Logical router creation failed
NCP00023	Logical router update failed
NCP00024	Logical router deletion failed
NCP00025	Logical switch creation failed

Error Code	Description
NCP00026	Logical switch update failed
NCP00027	Logical switch deletion failed
NCP00028	Logical router port creation failed
NCP00029	Logical router port deletion failed
NCP00030	Logical switch port creation failed
NCP00031	Logical switch port update failed
NCP00032	Logical switch port deletion failed
NCP00033	Network policy not found
NCP00034	Firewall creation failed
NCP00035	Firewall read failed
NCP00036	Firewall update failed
NCP00037	Firewall deletion failed
NCP00038	Multiple firewall found
NCP00039	NSGroup creation failed
NCP00040	NSGroup deletion failed
NCP00041	IP set creation failed
NCP00042	IP set update failed
NCP00043	IP set deletion failed
NCP00044	SNAT rule creation failed
NCP00045	SNAT rule deletion failed
NCP00046	Adapter API connection failed
NCP00047	Adapter watcher exception
NCP00048	Load balancer service deletion failed
NCP00049	Load balancer virtual server creation failed
NCP00050	Load balancer virtual server update failed

Error Code	Description
NCP00051	Load balancer virtual server deletion failed
NCP00052	Load balancer pool creation failed
NCP00053	Load balancer pool update failed
NCP00054	Load balancer pool deletion failed
NCP00055	Load balancer rule creation failed
NCP00056	Load balancer rule update failed
NCP00057	Load balancer rule deletion failed
NCP00058	Load balancer pool IP release failed
NCP00059	Load balancer virtual server and service association not found
NCP00060	NSGroup update failed
NCP00061	Firewall rules get failed
NCP00062	NSGroup no criteria
NCP00063	Node VM not found
NCP00064	Node VIF not found
NCP00065	Certificate import failed
NCP00066	Certificate un-import failed
NCP00067	SSL binding update failed
NCP00068	SSL profile not found
NCP00069	IP pool not found
NCP00070	T0 edge cluster not found
NCP00071	IP pool update failed
NCP00072	Dispatcher failed
NCP00073	NAT rule deletion failed
NCP00074	Logical router port get failed
NCP00075	NSX configuration validation failed

Error Code	Description
NCP00076	SNAT rule update failed
NCP00077	SNAT rule overlapped
NCP00078	Load balancer endpoints add failed
NCP00079	Load balancer endpoints update failed
NCP00080	Load balancer rule pool creation failed
NCP00081	Load balancer virtual server not found
NCP00082	IP set read failed
NCP00083	SNAT pool get failed

Error Code	Description
NCP00084	Load balancer service creation failed
NCP00085	Load balancer service update failed
NCP00086	Logical router port update failed
NCP00087	Load balancer init failed
NCP00088	IP pool not unique
NCP00089	Layer 7 load balancer cache sync error
NCP00090	Load balancer pool not exist error
NCP00091	Load balancer rule cache init error
NCP00092	SNAT process failed
NCP00093	Load balancer default certificate error
NCP00094	Load balancer endpoint deletion failed
NCP00095	Project not found

NSX Node Agent Error Codes

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

nsx-kube-proxy Error Codes

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

CLI Error Codes

Error Code	Description
NCP03001	CLI start failed
NCP03002	CLI socket create failed
NCP03003	CLI socket exception
NCP03004	CLI client invalid request
NCP03005	CLI server transmit failed
NCP03006	CLI server receive failed
NCP03007	CLI command execute failed

Kubernetes Error Codes

Error Code	Description
NCP05001	Kubernetes connection failed
NCP05002	Kubernetes invalid configuration
NCP05003	Kubernetes request failed
NCP05004	Kubernetes key not found
NCP05005	Kubernetes type not found
NCP05006	Kubernetes watcher exception
NCP05007	Kubernetes resource invalid length
NCP05008	Kubernetes resource invalid type
NCP05009	Kubernetes resource handle failed
NCP05010	Kubernetes service handle failed
NCP05011	Kubernetes endpoint handle failed
NCP05012	Kubernetes Ingress handle failed
NCP05013	Kubernetes network policy handle failed
NCP05014	Kubernetes node handle failed
NCP05015	Kubernetes namespace handle failed
NCP05016	Kubernetes pod handle failed

Error Code	Description
NCP05017	Kubernetes secret handle failed
NCP05018	Kubernetes default backend failed
NCP05019	Kubernetes unsupported match expression
NCP05020	Kubernetes status update failed
NCP05021	Kubernetes annotation update failed
NCP05022	Kubernetes namespace cache not found
NCP05023	Kubernetes secret not found

OpenShift Error Codes

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