NSX Container Plugin for Kubernetes and Tanzu Application Service - Installation and Administration Guide

Modified on 8 NOV 2022

VMware NSX Container Plugin 4.0

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

https://docs.vmware.com/

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

Copyright © 2017-2022 VMware, Inc. All rights reserved. Copyright and trademark information.

Contents

NSX Container Plugin for Kubernetes and Tanzu Application Service - Installation and Administration Guide

This guide describes how to install and administer NSX Container Plugin (NCP) to provide integration between NSX and Kubernetes, as well as between NSX and Tanzu Application Service (TAS).

Intended Audience

This guide is intended for system and network administrators. A familiarity with the installation and administration of NSX, Kubernetes, and Tanzu Application Service (TAS) 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 https://www.vmware.com/topics/glossary.

Overview of NSX Container Plugin

1

NSX Container Plugin (NCP) provides integration between NSX and container orchestrators such as Kubernetes, as well as integration between NSX and container-based PaaS (platform as a service) products such as OpenShift and Tanzu Application Service (TAS). This guide describes setting up NCP with Kubernetes and TAS.

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

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

NCP provides the following functionalities:

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

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

- Implements Kubernetes network policies with NSX 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.
  - Support for selecting pods in another namespace.
- Implements Kubernetes service of type `ClusterIP` and service of type `LoadBalancer`.
- Implements Kubernetes Ingress with NSX layer 7 load balancer.
  - Support for HTTP Ingress and HTTPS Ingress with TLS edge termination.

  - Support for Ingress default backend configuration.
  - Support for redirect to HTTPS, path rewrite, and path pattern matching.
- Creates tags on the NSX logical switch port for the namespace, pod name, and labels of a pod, and allows the administrator to define NSX security groups and policies based on the tags.
- Multicast is supported between pods in the same namespace, but is not supported between pods in different namespaces.

NCP supports a single Kubernetes cluster. You can have multiple Kubernetes clusters, each with its distinct NCP instance, using the same NSX deployment.

This chapter includes the following topics:

Compatibility Requirements

For compatibility requirements, see the release notes.

NCP 4.0.0 release notes: https://docs.vmware.com/en/VMware-NSX-Container-Plugin/4.0.0/rn/vmware-nsx-container-plugin-400-release-notes/index.html

NCP 4.0.1 release notes: https://docs.vmware.com/en/VMware-NSX-Container-Plugin/4.0.1/rn/vmware-nsx-container-plugin-401-release-notes/index.html

Installation Overview

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

1. Install NSX.
2. Create an overlay transport zone.
3. Create an overlay logical switch and connect the Kubernetes nodes to the switch.
4. Create a tier-0 logical router.
5. Create IP blocks for Kubernetes pods.
6. Create IP pools for SNAT (source network address translation).
7. Install NSX CNI (container network interface) plugin on each node.
8. Install OVS (Open vSwitch) on each node.
9. Configure NSX networking for Kubernetes nodes.
10. Install NSX node agent as a DaemonSet.
11. Install NCP as a ReplicationController.

12 Mount security certificates in the NCP pod.

Configuring Resources 2

Before installing NCP, you need to set up some NSX resources.

Note: When you install NSX, the default license does not allow creating or updating objects that you will need to support NCP. For more information, see the section "Add a License Key and Generate a License Usage Report" in the *NSX Administration Guide*.

The NSX Manager web UI provides 2 methods to configure networking resources: Policy mode and Manager mode. For more information, see the section "NSX Manager" in the *NSX Administration Guide*.

You must use one of the two methods to configure NSX networking for NCP, but not both. You must set the `policy_nsxapi` option in the NCP YAML file to **`True`** if you use the Policy mode, or **`False`** if you use the Manager mode.

The following sections assume that you are familiar with installing and administering NSX. For more information, see the *NSX Installation Guide* and the *NSX Administration Guide*.

This chapter includes the following topics:

Configuring NSX Resources in Policy Mode

There are two methods to configure certain networking resources for NCP. This section describes configuring resources in Policy mode.

In the NCP configuration file `ncp.ini`, you must specify NSX resources using their resource IDs. Usually a resource's name and ID are the same. To be completely sure, on the NSX Manager web UI, click the 3-dot icon that displays options for a resource and select **Copy path to clipboard**. Paste the path to an application such as Notepad. The last part of the path is the resource ID.

Gateways and Segment

1 Create a segment for the Kubernetes nodes, for example, `Segment1`.

2 Create a tier-0 gateway, for example, `T0GW1`. Set the `top_tier_router` option in the `[nsx_v3]` section of `ncp.ini` with the gateway's ID if you do not have a shared tier-1 topology. See below for information on configuring a shared tier-1 topology. Set the HA mode to active-standby if you plan to configure NAT rules on this gateway. Otherwise, set it to active-active. Enable route redistribution. Also configure this gateway for access to the external network.

3 Create a tier-1 gateway, for example, `T1GW1`. Connect this gateway to the tier-0 gateway.

4 Configure router advertisement for `T1GW1`. At the very least, NSX-connected and NAT routes should be enabled.

5 Connect `T1GW1` to `Segment1`. Make sure that the gateway port's IP address does not conflict with the IP addresses of the Kubernetes nodes.

6 For each node VM, make sure that the vNIC for container traffic is attached to the logical switch that is automatically created. You can find it in the **Networking** tab with the same name as the segment, that is, `Segment1`).

NCP must know the VIF ID of the vNIC. You can see Segment1's ports that are automatically created by navigating to Networking > Segments. These ports are not editable except for their tag property. These ports must have the following tags. For one tag, specify the name of the node. For the other tag, specify the name of the cluster. For the scope, specify the appropriate value as indicated below.

Tag	Scope
Node name	`ncp/node_name`
Cluster name	`ncp/cluster`

These tags are automatically propagated to the corresponding logical switch ports. If the node name changes, you must update the tag. To retrieve the node name, you can run the following command:

```
kubectl get nodes
```

If you want to extend the Kubernetes cluster while NCP is running, for example, add more nodes to the cluster, you must add the tags to the corresponding switch ports before running "kubeadm join". If you forget to add the tags before running "kubeadm join", the new nodes will not have connectivity. In this case, you must add the tags and restart NCP to resolve the issue.

To identify the switch port for a node VM, you can make the following API call:

```
/api/v1/fabric/virtual-machines
```

In the response, look for the Node VM and retrieve the value for the ``external_id`` attribute. Or you can make the following API call:

```
/api/v1/search -G --data-urlencode "query=(resource_type:VirtualMachine AND
display_name:<node_vm_name>)"
```

After you have the external ID, you can use it to retrieve the VIFs for the VM with the following API. Note that VIFs are not populated until the VM is started.

```
/api/v1/search -G --data-urlencode \
"query=(resource_type:VirtualNetworkInterface AND external_id:<node_vm_ext_id> AND \
_exists_:lport_attachment_id)"
```

The `lport_attachment_id` attribute is the VIF ID for the node VM. You can then find the logical port for this VIF and add the required tags.

IP Blocks for Kubernetes Pods

Navigate to **Networking > IP Address Management > IP Address Pools > IP Address Blocks** to create one or more IP blocks. Specify the IP block in CIDR format. Set the `container_ip_blocks` option in the `[nsx_v3]` section of `ncp.ini` to the UUIDs of the IP blocks. If you want NCP to automatically create IP blocks, you can set the `container_ip_blocks` option with a comma-separated list of addresses in CIDR format. Note that you cannot set `container_ip_blocks` to both UUIDs and CIDR addresses.

By default, projects share IP blocks specified in `container_ip_blocks`. You can create IP blocks specifically for no-SNAT namespaces (for Kubernetes) or clusters (for TAS) 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.

When you create an IP block, the prefix must not be larger than the value of the `subnet_prefix` option in NCP's configuration file `ncp.ini`. The default is 24.

You must not modify the IP block if NCP has started allocated IP addresses from it. If you want to use a different block, make sure that NCP has not allocated any address from the block.

External IP Pools

An external IP pool is used for allocating IP addresses which will be used for translating pod IPs using SNAT rules, and for exposing Ingress controllers and LoadBalancer-type services using SNAT/DNAT rules, just like Openstack floating IPs. These IP addresses are also referred to as external IPs.

Navigate to **Networking > IP Address Management > IP Address Pools** to create an IP pool. Set the `external_ip_pools` option in the `[nsx_v3]` section of `ncp.ini` to the UUIDs of the IP pools. If you want NCP to automatically create IP pools, you can set the `external_ip_pools` option with a comma-separated list of addresses in CIDR format or IP ranges. Note that you cannot set `external_ip_pools` to both UUIDs and CIDR addresses.

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`.

You can change to a different IP pool by changing the configuration file and restarting NCP.

You must not modify the IP pool if NCP has started allocated IP addresses from it. If you want to use a different pool, make sure that NCP has not allocated any address from the pool.

Shared Tier-1 Topology

To enable a shared tier-1 topology, perform the following configurations:

- Set the `top_tier_router` option to the ID of a tier-1 gateway. Connect the tier-1 gateway to a tier-0 gateway for external connections.
- If SNAT for Pod traffic is enabled, modify the uplink of the segment for Kubernetes nodes to the same tier-0 or tier-1 gateway that is set in `top_tier_router`.
- Set the `single_tier_topology` option to **`True`**. The default value is **`False`**.
- If you want NCP to automatically configure the top tier router as a tier-1 gateway, unset the `top_tier_router` option and set the `tier0_gateway` option. NCP will create a tier-1 gateway and uplink it to the tier-0 gateway specified in the `tier0_gateway` option.

Configuring NSX Resources in Manager Mode

There are two methods to configure certain networking resources for NCP. This section describes configuring resources in Manager mode.

In the NCP configuration file `ncp.ini`, you can specify NSX resources using their UUIDs or names.

Logical Routers and Logical switch

1. Create a logical switch for the Kubernetes nodes, for example, `LS1`.
2. Create a tier-0 logical router, for example, `T0LR1`. Set the `tier0_router` option in the `[nsx_v3]` section of `ncp.ini` with the logical router's ID if you do not have a shared tier-1 topology. See below for information on configuring a shared tier-1 topology. Set the HA mode to active-standby if you plan to configure NAT rules on this logical router. Otherwise, set it to active-active. Enable route redistribution. Also configure this router for access to the external network.

3 Create a tier-1 logical router, for example, `T1LR1`. Connect this logical router to the tier-0 logical router.

4 Configure router advertisement for `T1LR1`. At the very least, NSX-connected and NAT routes should be enabled.

5 Connect `T1LR1` to `LS1`. Make sure that the logical router port's IP address does not conflict with the IP addresses of the Kubernetes nodes.

6 For each node VM, make sure that the vNIC for container traffic is attached to the logical switch that is automatically created. You can find it in the **Networking** tab with the same name as the logical switch, that is, `LS1`).

NCP must know the VIF ID of the vNIC. The corresponding logical switch ports must have the following two tags. For one tag, specify the name of the node. For the other tag, specify the name of the cluster. For the scope, specify the appropriate value as indicated below.

Tag	Scope
Node name	`ncp/node_name`
Cluster name	`ncp/cluster`

If the node name changes, you must update the tag. To retrieve the node name, you can run the following command:

```
kubectl get nodes
```

If you want to extend the Kubernetes cluster while NCP is running, for example, add more nodes to the cluster, you must add the tags to the corresponding switch ports before running "kubeadm join". If you forget to add the tags before running "kubeadm join", the new nodes will not have connectivity. In this case, you must add the tags and restart NCP to resolve the issue.

To identify the switch port for a node VM, you can make the following API call:

```
/api/v1/fabric/virtual-machines
```

In the response, look for the Node VM and retrieve the value for the ``external_id`` attribute. Or you can make the following API call:

```
/api/v1/search -G --data-urlencode "query=(resource_type:VirtualMachine AND
display_name:<node_vm_name>)"
```

After you have the external ID, you can use it to retrieve the VIFs for the VM with the following API. Note that VIFs are not populated until the VM is started.

```
/api/v1/search -G --data-urlencode \
"query=(resource_type:VirtualNetworkInterface AND external_id:<node_vm_ext_id> AND \
_exists_:lport_attachment_id)"
```

The `lport_attachment_id` attribute is the VIF ID for the node VM. You can then find the logical port for this VIF and add the required tags.

IP Blocks for Kubernetes Pods

Navigate to **Networking > IP Address Pools** to create one or more IP blocks. Specify the IP block in CIDR format. Set the `container_ip_blocks` option in the `[nsx_v3]` section of `ncp.ini` to the UUIDs of the IP blocks.

By default, projects share IP blocks specified in `container_ip_blocks`. You can create IP blocks specifically for no-SNAT namespaces (for Kubernetes) or clusters (for TAS) 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.

When you create an IP block, the prefix must not be larger than the value of the `subnet_prefix` option in NCP's configuration file `ncp.ini`. The default is 24.

NCP will allocate additional subnets for a namespace if the originally allocated subnet is exhausted.

External IP Pools

An external IP pool is used for allocating IP addresses which will be used for translating pod IPs using SNAT rules, and for exposing Ingress controllers and LoadBalancer-type services using SNAT/DNAT rules, just like Openstack floating IPs. These IP addresses are also referred to as external IPs.

Navigate to **Networking > IP Address Pools > IP Pools** to create an IP pool. Set the `external_ip_pools` option in the `[nsx_v3]` section of `ncp.ini` to the UUIDs of the IP pools.

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`.

You can change to a different IP pool by changing the configuration file and restarting NCP.

Shared Tier-1 Topology

To enable a shared tier-1 topology, perform the following configurations:

- Set the `top_tier_router` option to the ID of either a tier-0 logical router or a tier-1 logical router. If it is a tier-1 logical router, you need to connect it to a tier-0 logical router for external connections. This option replaces the `tier0_router` option.
- If SNAT for Pod traffic is enabled, disconnect `T1LR1` from `LS1` (the logical switch for the Kubernetes nodes), and connect the tier-0 or tier-1 router set in `top_tier_router` to `LS1`.
- Set the `single_tier_topology` option to **`True`**. The default value is **`False`**.

(Optional) (For Kubernetes only) Firewall Marker Sections

To allow the administrator to create firewall rules and not have them interfere with NCP-created firewall sections based on network policies, navigate to **Security > Distributed 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.

Configuring SNAT

Restricting an SNAT IP Pool to Specific Kubernetes Namespaces or TAS Orgs

You can specify which Kubernetes namespace or TAS org can be allocated IPs from the SNAT IP pool by adding the following tags to the IP pool.

- For a Kubernetes namespace: `scope: ncp/owner, tag: ns:<namespace_UUID>`
- For a TAS org: `scope: ncp/owner, tag: org:<org_UUID>`

You can get the namespace or org UUID with one of the following commands:

```
kubectl get ns -o yaml
cf org <org_name> --guid
```

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 or orgs have been allocated IPs based on the old tags, those IPs will not be reclaimed until the SNAT configurations of the Kubernetes services or TAS apps change or NCP restarts..
- The namespace and TAS org owner tags are optional. Without these tags, any namespace or TAS org can have IPs allocated from the SNAT IP pool.

Configuring an SNAT IP Pool for a Service

You can configure an SNAT IP pool for a 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
...
```

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

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. If an error occurs when NCP configures the SNAT rule, the service will be annotated with `ncp/error.snat`:`<error>`. The possible errors are:

- IP_POOL_NOT_FOUND - The SNAT IP pool is not found in NSX Manager.
- IP_POOL_EXHAUSTED - The SNAT IP pool is exhausted.
- IP_POOL_NOT_UNIQUE - The pool specified by `ncp/snat_pool` refers to multiple pools in NSX Manager.
- SNAT_RULE_OVERLAPPED - A new SNAT rule is created, but the SNAT service's pod also belongs to another SNAT service, that is, there are multiple SNAT rules for the same pod.
- POOL_ACCESS_DENIED - The IP pool specified by `ncp/snat_pool` does not have the tag `scope: ncp/owner, tag: cluster:<cluster_name>`, or the pool's owner tag does not match the namespace of the service that is sending the allocation request. After you fix the error, you must restart NCP, or remove the `ncp/snat_pool` annotation and add it again.

Note the following:

- The pool specified by `ncp/snat_pool` should already exist in NSX before the service is configured.
- In NSX, the priority of the SNAT rule for the service is higher than that for the project.
- If a pod is configured with multiple SNAT rules, only one will work.
- You can change to a different IP pool by changing the annotation and restarting NCP.

Configuring an SNAT IP Pool for a Namespace

You can configure an SNAT IP pool for a namespace by adding an annotation to the namespace. For example,

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns-sample
  annotations:
    ncp/snat_pool: <external IP pool ID or name>
...
```

NCP will configure the SNAT rule for this namespace. 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. If an error occurs when NCP configures the SNAT rule, the namespace will be annotated with `ncp/error.snat`:`<error>`. The possible errors are:

- IP_POOL_NOT_FOUND - The SNAT IP pool is not found in NSX Manager.
- IP_POOL_EXHAUSTED - The SNAT IP pool is exhausted.
- IP_POOL_NOT_UNIQUE - The pool specified by `ncp/snat_pool` refers to multiple pools in NSX Manager.
- POOL_ACCESS_DENIED - The IP pool specified by `ncp/snat_pool` does not have the tag `scope: ncp/owner, tag: cluster:<cluster_name>`, or the pool's owner tag does not match the namespace that is sending the allocation request. After you fix the error, you must restart NCP, or remove the `ncp/snat_pool` annotation and add it again.

Note the following:

- You can specify only one SNAT IP pool in the annotation.
- The SNAT IP pool does not need to be configured in `ncp.ini`.
- The IP pool specified by `ncp/snat_pool` must have the tag `scope: ncp/owner, tag: cluster:<cluster_name>`.
- The IP pool specified by `ncp/snat_pool` can also have a namespace tag `scope: ncp/owner, tag: ns:<namespace_UUID>`.
- If the `ncp/snat_pool` annotation is missing, the namespace will use the SNAT IP pool for the cluster.
- You can change to a different IP pool by changing the annotation and restarting NCP.

Configuring an SNAT IP Address for a Service

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

```
apiVersion: v1
kind: Service
metadata:
  name: svc-example
  annotations:
    ncp/static_snat_ip: "1.2.3.4"
...
```

If the annotation `ncp/snat_pool` is also specified, the SNAT IP address must be in the specified SNAT address pool. Otherwise, it must be in the external IP pool specified in `ncp.ini`. If there are no errors, NCP will create or update the SNAT rule by using the annotated SNAT IP address for this service. The status of configuring the SNAT rule will be annotated with `ncp/snat_ip_status` in the service. The possible values are:

- IP_ALLOCATED_SUCCESSFULLY
- IP_ALREADY_ALLOCATED - The IP address has already been allocated.
- IP_NOT_IN_POOL - The IP address is not in the SNAT IP Pool.
- IP_POOL_EXHAUSTED - The SNAT IP Pool is exhausted.
- SNAT_PROCESS_FAILED - An unknown error occurred.

Configuring an SNAT IP Address for a Namespace

You can configure an SNAT IP address for a namespace by adding an annotation to the namespace. For example,

```
apiVersion: v1
kind: Namespace
metadata:
  name: svc-example
  annotations:
    ncp/static_snat_ip: "1.2.3.4"
...
```

If the annotation `ncp/snat_pool` is also specified, the SNAT IP address must be in the specified SNAT address pool. Otherwise, it must be in the external IP pool specified in `ncp.ini`. If there are no errors, NCP will create or update the SNAT rule by using the annotated SNAT IP address for this namespace. The status of configuring the SNAT rule will be annotated with `ncp/snat_ip_status` in the namespace. The possible values are:

- IP_ALLOCATED_SUCCESSFULLY
- IP_ALREADY_ALLOCATED - The IP address has already been allocated.
- IP_NOT_IN_POOL - The IP address is not in the SNAT IP Pool.

- IP_NOT_REALIZED - An error occurred in NSX.
- IP_POOL_EXHAUSTED - The SNAT IP Pool is exhausted.
- SNAT_PROCESS_FAILED - An unknown error occurred.

Configuring an SNAT Pool for a TAS App

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

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

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

- The pool specified by `NCP_SNAT_POOL` should already exist in NSX before the app is pushed.
- The priority of SNAT rule for an app is higher than that for an org.
- An app can be configured with only one SNAT IP.
- You can change to a different IP pool by changing the configuration and pushing the app again.

Configuring SNAT for TAS version 3

With TAS version 3, you can configure SNAT in one of two ways:

- Configure `NCP_SNAT_POOL` in `manifest.yml` when creating the app.

  For example, the app is called `bread` and the `manifest.yml` has the following information:

  ```
  applications:
  - name: bread
    stack: cflinuxfs2
    random-route: true
    env:
      NCP_SNAT_POOL: AppSnatExternalIppool
    processes:
    - type: web
      disk_quota: 1024M
      instances: 2
      memory: 512M
      health-check-type: port
    - type: worker
  ```

```
    disk_quota: 1024M
    health-check-type: process
    instances: 2
    memory: 256M
    timeout: 15
```

Run the following commands:

```
cf v3-push bread
cf v3-apply-manifest -f manifest.yml
cf v3-apps
cf v3-restart bread
```

- Configure `NCP_SNAT_POOL` using the `cf v3-set-env` command.

  Run the following commands (assuming the app is called `app3`):

```
cf v3-set-env app3 NCP_SNAT_POOL AppSnatExternalIppool
(optional) cf v3-stage app3 -package-guid <package-guid> (You can get package-guid with
"cf v3-packages app3".)
cf v3-restart app3
```

Configuring an SNAT IP Pool or IP Address for a TAS Org

You can configure an SNAT IP pool or IP address for a TAS Org using the following annotations:

- `ncp_snat_pool` - The pool must exist and have the tag `scope: ncp/owner, tag: cluster:<cluster_name>`.
- `ncp_snat_ip` - A specific address in an IP pool.

Note the following:

- If both `ncp_snat_pool` and `ncp_snat_ip` are specified, the SNAT IP address must be in the specified SNAT IP pool.
- If only `ncp_snat_ip` is specified, the SNAT IP address must be in the external IP pool specified in `ncp.ini`.
- If only `ncp_snat_pool` is specified, the SNAT IP address will be allocated from the specified pool.

You can configure SNAT IP for an org with the `cf curl` command. For example:

```
cf curl v3/organizations/<org-guid> -X PATCH  -d '{"metadata": {"annotations":
{"ncp_snat_pool": "ann-ip-pool", "ncp_snat_ip": "1.2.3.4"}}}'
```

You can get `org-guid` with the following command:

```
cf org <org-name> --guid
```

You can remove the `ncp_snat_ip` annotation with the following command:

```
cf curl v3/organizations/<org-guid> -X PATCH  -d '{"metadata": {"annotations":
{"ncp_snat_ip": null}}}'
```

You can go to the NSX Manager UI to check if the SNAT rule is successfully created. To check for errors, look at the NCP logs.

If you see the `POOL_ACCESS_DENIED` error in the NCP log, it means that the IP pool specified by `ncp_snat_pool` does not have the tag `scope: ncp/owner, tag: cluster:<cluster_name>`, or the pool's owner tag does not match the org that is sending the allocation request. After you fix the error, you must restart NCP, or remove the `ncp_snat_pool` annotation and add it again.

StatefulSet Persistent IP Allocation

You can specify an IP address range for a StatefulSet with the annotation `ncp/ip_range`. NCP will allocate persistent IP addresses from the range to pods in the StatefulSet based on the subnets of the namespace.

This feature is only supported in Policy mode.

An example of specifyiing the `ncp/ip_range` annotation:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx # has to match .spec.template.metadata.labels
  serviceName: "nginx"
  replicas: 3 # by default is 1
  template:
    metadata:
      labels:
        app: nginx # has to match .spec.selector.matchLabels
      annotations:
        ncp/ip_range: 192.168.0.10-192.168.0.50
    spec:
      ...
```

If an error occurs when NCP checks the IP range or allocates an address from the range, the StatefulSet will be annotated with `ncp/error.ip_range:<error>`. The possible errors are:

- INVALID_IP_RANGE - The IP range is not in any of the namespaces's subnet, or it is in more than one subnets, or an existing pod is already using an IP address in the range.
- IP_RANGE_EXHAUSTED - ncp cannot allocate an IP address from the range because all addresses have been allocated.

Configuring Subnets for a Kubernetes Namespace

When you create a Kubernetes namespace, you can specify subnets for the namespace with the annotation `ncp/subnets` if SNAT is not configured for the namespace. The subnets will be used to allocate IP addresses to the pods in the namespace.

To enable this feature, set `enable_namespace_subnets` to `True` under the `[K8s]` section in `ncp.ini`. The default is `False`. Once set to `True`, you must not set it back to `False`.

Details about this feature:

- The value of the `ncp/subnets` annotation should be a comma-separated list of IP addresses in CIDR format.
- You can update the annotation with additional subnets.
- If a pod is created in the namespace and no IP address from the subnets is available, the namespace is annotated with the error `NAMESPACE_SUBNETS_EXHAUSTED`. You can update the `ncp/subnets` annotation with additional subnets and the pod will be allocated an IP address.
- You can remove a subnet from the annotation if no IP address has been allocated from it.
- This feature is only supported in policy mode.
- This feature is only supported for new namespaces.
- Adding the annotation to an existing namespace is not supported. The namespace will be annotated with the error `SUBNETS_ON_EXISTING_NAMESPACE_NOT_SUPPORTED`.
- Removing the annotation is not supported. If you remove the annotation, NCP will add it back and log a warning in the NCP log.
- This feature is not supported on Tanzu Application Service (TAS), Tanzu Kubernetes Grid Integrated (TKGI), or vSphere with Kubernetes.
- Both IPv4 and IPv6 are supported.
- Removing a subnet from the annotation will be ignored if an IP address from the subnet has been allocated. NCP will add the subnet back and log an error.

Installing NCP in a Kubernetes Environment

3

Installing NSX Container Plugin (NCP) requires installing components on the master and worker nodes.

To set up a Kubernetes cluster with NCP, you can follow this guide, which documents the parameters and features that you can configure. You can also use NCP Operator to automatically set up a Kubernetes cluster with NCP. For more information about NCP Operator, see https://github.com/vmware/nsx-container-plugin-operator/blob/master/README.md.

This chapter includes the following topics:

Download Installation Files

To install NCP, download the NCP installation file bundle from VMware's download page. Unzip the file bundle.

The YAML and image files are located in the `Kubernetes` directory.

Run the following command to load the docker image:

```
docker load -i nsx-ncp-ubuntu-<version>.<build_no>.tar
```

or

```
docker load -i nsx-ncp-rhel-<version>.<build_no>.tar
```

Run the following command to check the name of the loaded image:

```
docker images ls | grep ncp
```

Prepare Kubernetes Nodes

Most of the steps to prepare the Kubernetes nodes are automated by two containers, nsx-ovs and nsx-ncp-bootstrap, that run in the nsx-node-agent and nsx-ncp-bootstrap DaemonSets, respectively.

Before installing NCP, make sure that the Kubernetes nodes have Python installed and accessible through the command line interface. You can use your Linux package manager to install it. For example, on Ubuntu, you can run the command `apt install python`.

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

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

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

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

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

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

You can replace the `node-agent-apparmor` profile with a different profile. If you do, you must change the profile name `node-agent-apparmor` in the NCP YAML file.

The NSX NCP bootstrap container automates the installation and update of of NSX CNI on the host. In previous releases, NSX CNI was installed through a deb/rpm package. In the release, the files are simply copied to the host. The bootstrap container will remove the previously installed NSX CNI components from the package manager's database. The following directories and files will be deleted:

- /etc/cni/net.d

- /etc/apparmor.d/ncp-apparmor
- /opt/cni/bin/nsx

The bootstrap container checks the file `10-nsx.conflist` and looks for the CNI version number in the tag `nsxBuildVersion`. If this version is older than the one in the bootstrap container, the following files are copied to the host:

- /opt/cni/bin/nsx
- /etc/cni/net.d/99-loopback.conf
- /etc/cni/net.d/10-nsx.conflist
- /etc/apparmor.d/ncp-apparmor

If the files `/opt/cni/bin/loopback` and `/etc/cni/net.d/99-loopback.conf` exist, they are not overwritten. If the OS type is Ubuntu, the file `ncp-apparmor` is also copied to the host.

The bootstrap container will move the IP address and routes from `br-int` to `node-if`. It will also stop OVS if it is running on the host because OVS will run inside the nsx-ovs container. The nsx-ovs container will create the `br-int` instance if it does not exist, add the network interface (`node-if`) that is attached to the node logical switch to `br-int`, and make sure that the `br-int` and `node-if` link status is up. It will move the IP address and routes from `node-if` to `br-int`. There will be downtime of a few seconds when the nsx-node-agent pod or nsx-ovs container is restarted.

Note If the nsx-node-agent DaemonSet is removed, OVS is no longer running on the host (in the container or in the host's PID).

Update the network configuration to make the IP address and routes persistent. For example, for Ubuntu, edit `/etc/network/interfaces` (use actual values from your environment where appropriate) to make the IP address and routes persistent:

```
auto eth1
iface eth1 inet static
address 172.16.1.4/24
#persistent static routes
up route add -net 172.16.1.3/24 gw 172.16.1.1 dev eth1
```

Then run the command `ifdown eth1; ifup eth1`.

For RHEL, create and edit `/etc/sysconfig/network-scripts/ifcfg-<node-if>` (use actual values from your environment where appropriate) to make the IP address persistent:

```
HWADDR=00:0C:29:B7:DC:6F
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
IPADDR=172.10.0.2
PREFIX=24
```

```
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV4_DNS_PRIORITY=100
IPV6INIT=no
NAME=eth1
UUID=39317e23-909b-45fc-9951-849ece53cb83
DEVICE=eth1
ONBOOT=yes
```

Then run the command `systemctl restart network.service`.

For information on configuring persistent routes for RHEL, see https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/sec-configuring_static_routes_in_ifcfg_files.

Note IP and static routes must be persisted on the uplink interface (specified by `ovs_uplink_port`) to guarantee that connectivity to the Kubernetes API server is not lost after a VM restart.

Note By default, nsx-ovs has a volume mount with the name `host-original-ovs-db` and path `/etc/openvswitch`. This is the default path that OVS uses to store the file `conf.db`. If OVS was configured to use a different path or if the path is a soft link, you must update the `host-original-ovs-db` mount with the correct path.

If necessary, you can undo the changes made by the bootstrap container. For more information, see Clean Up Kubernetes Nodes.

Create Secrets - Kubernetes

You can create secrets to store NSX client certificate and key or load balancer certificate and key.

The simplest way for creating Secret is to use the kubectl command:

```
kubectl create ns nsx-system
kubectl create secret tls ${CERT_NAME} --key ${KEY_FILE} --cert ${CERT_FILE} -n nsx-system
```

You can also create secrets using the NCP YAML file by uncommenting the the example Secret sections in the file, and fill in the base64-encoded values of the certificate and key. Use the following commands to get the base64-encoded string from the certificate or key file:

```
cat cert_file.crt | base64 -w 0
cat key_file.crt | base64 -w 0
```

Configure NSX Networking for Kubernetes Nodes

This section describes how to configure NSX networking for Kubernetes master and worker nodes.

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

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

The VIF ID of the vNIC used for container traffic in each node must be known to NSX Container Plugin (NCP). The corresponding logical switch port must have the following two tags. For one tag, specify the name of the node. For the other tag, specify the name of the cluster. For the scope, specify the appropriate value as indicated below.

Tag	Scope
Node name	`ncp/node_name`
Cluster name	`ncp/cluster`

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

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

```
kubectl get nodes
```

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

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

Edit the NCP YAML File

The NCP YAML file contains information to configure, install and run all the NCP components.

The NCP YAML file contains the following information:

- RBAC definitions.
- Various CRDs (CustomResourceDefinitions).
- ConfigMap containing ncp.ini dedicated to NCP. Some recommended configuration options are already set.
- NCP Deployment.

- ConfigMap containing ncp.ini deidicated to nsx-node-agent. Some recommended configuration options are already set.
- nsx-node-agent DaemonSet, including nsx-node-agent, nsx-kube-proxy, and nsx-ovs.
- nsx-ncp-bootstrap DaemonSet

The NSX CNI and OpenvSwitch kernel modules are installed automatically by `nsx-ncp-bootstrap` initContainers. The OpenvSwitch userspace daemons are running in nsx-ovs container on each node.

Update the NCP Deployment Specs

Locate the ConfigMap with the name `nsx-ncp-config`. For the complete list of the ConfigMap options, see The nsx-ncp-config ConfigMap. Some options are already configured to recommended values. You can customize all the options for your environment. For example,

- Log level and log directory.
- Kubernetes API server IP, certificate path and client token path. By default, the API server ClusterIP from the environment variable is used, and the certficiate and token are automatically mounted from ServiceAccount. Usually no change is required.
- Kubernetes cluster name.
- NSX Manager IP and credentials.
- NSX resource options such as `overlay_tz`, `top_tier_router`, `container_ip_blocks`, `external_ip_blocks`, and so on.

By default the Kubernetes Service VIP/port and ServiceAccount token and `ca_file` are used for Kubernetes API access. No change is required here, but you need to fill in some NSX API options of ncp.ini.

- Specify the `nsx_api_managers` option. It can be a comma-separated list of NSX Manager IP addresses or URL specifications that are compliant with RFC3896. For example,

  ```
  nsx_api_managers = 192.168.1.181, 192.168.1.182, 192.168.1.183
  ```

- Specify the `nsx_api_user` and `nsx_api_password` options with the user name and password, respectively, if you configure NCP to connect to NSX using basic authentication. This authentication method is not recommended because it is less secure. These options are ignored if NCP is configured to authenticate using client certificates. These options do not appear in the NCP YAML file. You must add them manually.
- Specify the `nsx_api_cert_file` and `nsx_api_private_key_file` options for authentication with NSX. The `nsx_api_cert_file` option is the full path to a client certificate file in PEM format. The contents of this file should look like the following:

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

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

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

By using client certificate authentication, NCP can use its principal identity to create NSX objects. This means that only an identity with the same identity name can modify or delete the objects. It prevents NSX objects created by NCP from being modified or deleted by mistake. Note that an administrator can modify or delete any object. If the object was created with a principal identity, a warning will indicate that.

- (Optional) Specify the `ca_file` option. The value should be a CA bundle file to use in verifying the NSX Manager server certificate. If not set, the system root CAs will be used. If you specify one IP address for `nsx_api_managers`, then specify one CA file. if you specify three IP addresses for nsx_api_managers, you can specify one or three CA files. If you specify one CA file, it will be used for all three managers. If you specify three CA files, each will be used for the corresponding IP address in nsx_api_managers. For example,

```
   nsx_api_managers = 192.168.1.181,192.168.1.182,192.168.1.183
   ca_file = ca_file_for_all_mgrs
or
   nsx_api_managers = 192.168.1.181,192.168.1.182,192.168.1.183
   ca_file = ca_file_for_mgr1,ca_file_for_mgr2,ca_file_for_mgr3
```

- (Optional) Specify the `insecure` option. If set to **`True`**, the NSX Manager server certificate is not verified. The default is **`False`**.

If you want to use a Kubernetes Secret to store the NSX client certificate and load balancer default certificate, you have to first create Secrets using a kubectl command, then update the Deployment spec:

- Add Secret volumes to the NCP pod spec, or uncomment the example Secrete volumes.
- Add volume mounts to the NCP container spec, or uncomment the example volume mounts.
- Update ncp.ini in the ConfigMap to set the certificate file path pointing to the file in the mounted volume.

If you do not have a shared tier-1 topology, you must set the `edge_cluster` option to the edge cluster ID so that NCP will create a tier-1 gateway or router for the Loadbalancer service. You can find the edge cluster ID by navigating to **System > Fabric > Nodes**, selecting the **Edge Clusters** tab and clicking the edge cluster name.

HA (high availability) is enabled by default. In a production environment, it is recommended that you do not disable HA.

Note: kube-scheduler by default will not schedule pods on the master node. In the NCP YAML file, a toleration is added to allow NCP Pod to run on the master node.

Configure SNAT

By default, NCP configures SNAT for every project. SNAT will not be configured for namespaces with the following annotation:

```
ncp/no_snat: True
```

If you do not want SNAT for any namespace in the cluster, configure the following option in `ncp.ini`:

```
[coe]
enable_snat = False
```

Note: Updating an existing namespace SNAT annotation is not supported. If you perform such an action, the topology for the namespace will be in an inconsistent state because a stale SNAT rule might remain. To recover from such an inconsistent state, you must recreate the namespace.

(Policy mode only) If SNAT is configured for a cluster, BGP on the tier-0 router is enabled, and `Connected Interfaces & Segments` is selected under `Advertised tier-1 subnets` when you configure route redistribution for the tier-0 router, you can use the following option to control route redistribution:

```
[nsx_v3]
configure_t0_redistribution = True
```

If `configure_t0_redistribution` is set to `True`, NCP will add a `deny` route map entry in the redistribution rule to stop the tier-0 router from advertising the cluster's internal subnets to BGP neighbors. This is mainly used for vSphere with Kubernetes clusters. If you do not create a route map for the redistribution rule, NCP will create a route map using its principal identity and apply it in the rule. If you want to modify this route map, you must replace it with a new route map, copy the entries from the NCP-created route map, and add new entries. You must manage any potential conflicts between the new entries and NCP-created entries. If you simply unset the NCP-created route map without creating a new route map for the redistribution rule, NCP will apply the previously created route map to the redistribution rule again when NCP restarts.

Configure Firewall Matching for NAT Rules

Starting with NCP 3.2.1, you can use the `natfirewallmatch` option to specify how the NSX gateway firewall behaves with NAT rules created for a Kubernetes namespace. This option applies to newly created Kubernetes namespaces only and not to existing namespaces. This option works in policy mode only.

```
[nsx_v3]
# This parameter indicate how firewall is applied to a traffic packet.
# Firewall can be bypassed, or be applied to external/internal address of
# NAT rule
# Choices: MATCH_EXTERNAL_ADDRESS MATCH_INTERNAL_ADDRESS BYPASS
#natfirewallmatch = MATCH_INTERNAL_ADDRESS
```

Update the nsx-node-agent DaemonSet Specs

Locate the ConfigMap with the name `nsx-node-agent-config`. For the complete list of the ConfigMap options, see The nsx-node-agent-config ConfigMap. Some options are already configured to recommended values. You can customize all the options for your environment. For example,

- Log level and log directory.
- Kubernetes API server IP, certificate path and client token path. By default, the API server ClusterIP from the environment variable is used, and the certficiate and token are automatically mounted from ServiceAccount. Usually no change is required.
- OpenvSwitch uplink port. For example: `ovs_uplink_port=eth1`
- The MTU value for CNI.

To set the MTU value for CNI, modify the `mtu` parameter in the nsx-node-agent ConfigMap and restart the nsx-ncp-bootstrap pods. This will update the pod MTU on every node. You must also update the node MTU accordingly. A mismatch between the node and pod MTU can cause problems for node-pod communication, affecting, for example, TCP liveness and readiness probes.

The nsx-ncp-bootstrap DaemonSet installs CNI and OVS kernel modules on the node. It then shuts down OVS daemons on the node, so that later nsx-ovs container can run OVS daemons inside a Docker container. When CNI is not installed, all the Kubernetes nodes are in "Not Ready" state. There is a toleration on the bootstrap DaemonSet to allow it to run on "Not Ready" nodes. After it installs CNI plugin, the nodes should become "Ready".

If you are not using NSX OVS kernel module, you must update the volume parameter `host-original-ovs-db` with the correct path of where the OpenvSwitch database is configured to be during the compilation of the OVS kernel module. For example, if you specify `--sysconfdir=/var/lib`, then set `host-original-ovs-db` to `/var/lib/openvswitch`. Make sure you use the path of the actual OVS database and not a symbolic link pointing to it.

If you are using the NSX OVS kernel module, you must set `use_nsx_ovs_kernel_module = True` and uncomment the lines about volumes to be mounted:

```
  # Uncomment these mounts if installing NSX-OVS kernel module
#   # mount host lib modules to install OVS kernel module if needed
#   - name: host-modules
#     mountPath: /lib/modules
#   # mount openvswitch database
#   - name: host-config-openvswitch
#     mountPath: /etc/openvswitch
#   - name: dir-tmp-usr-ovs-kmod-backup
#   # we move the usr kmod files to this dir temporarily before
#   # installing new OVS kmod and/or backing up existing OVS kmod backup
#     mountPath: /tmp/nsx_usr_ovs_kmod_backup

#   # mount linux headers for compiling OVS kmod
#   - name: host-usr-src
#     mountPath: /usr/src

...

# Uncomment these volumes if installing NSX-OVS kernel module
# - name: host-modules
#   hostPath:
#     path: /lib/modules
# - name: host-config-openvswitch
#   hostPath:
#     path: /etc/openvswitch
# - name: dir-tmp-usr-ovs-kmod-backup
#   hostPath:
#     path: /tmp/nsx_usr_ovs_kmod_backup

# - name: host-usr-src
#   hostPath:
#     path: /usr/src
```

If you plan to specify `hostPort` for a Pod, set `enable_hostport_snat` to `True` in the `[k8s]` section in the `nsx-node-agent-config` ConfigMap. In the same ConfigMap, `use_ncp_portmap` must be set to `False` (the default value) if you install a CNI plugin. If you do not install a CNI plugin and want to use `portmap` from the NCP image, set `use_ncp_portmap` to `True`.

SNAT uses `hostIP` as the source IP for `hostPort` traffic. If there is a network policy for a Pod and you want to access a Pod's `hostPort`, you must add the worker node IP addresses in the network policy's allow rule. For example, if you have two worker nodes (172.10.0.2 and 172.10.0.3), the Ingress rule must look like:

```
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          project: test
    - podSelector:
        matchLabels:
          app: tea
    - ipBlock:
        cidr: 172.10.0.3/32
    - ipBlock:
        cidr: 172.10.0.2/32
    ...
```

The NSX node agent is a DaemonSet where each pod runs 3 containers:

- `nsx-node-agent` manages container network interfaces. It interacts with the CNI plugin and the Kubernetes API server.
- `nsx-kube-proxy` implements Kubernetes service abstraction by translating cluster IPs into pod IPs. It implements the same functionality as the upstream `kube-proxy`, but is not mutually exclusive with it.

- `nsx-ovs` runs the OpenvSwitch userspace daemons. It will also create the OVS bridge automatically and moves the IP address and routes back from `node-if` to `br-int`. You must add `ovs_uplink_port=ethX` in the `ncp.ini` so that it can use `ethX` as the OVS bridge uplink.

If worker nodes are using Ubuntu, the ncp-ubuntu.yaml assumes AppArmor kernel module is enabled, otherwise Kubelet will refuse to run nsx-node-agent DaemonSet since it's configured with AppArmor option. For Ubuntu and SUSE, it's enabled by default. To check whether the module is enabled, check the `/sys/module/apparmor/parameters/enabled` file.

If AppArmor is disabled intentionally, the following changes need to be applied to the YAML file:

- Remove the AppArmor option:

  ```
  annotations:
      # The following line needs to be removed
      container.apparmor.security.beta.kubernetes.io/nsx-node-agent: localhost/node-agent-
  apparmor
  ```

- Enable the privileged mode for both the nsx-node-agent and nsx-kube-proxy containers

  ```
  securityContext:
      # The following line needs to be appended
      privileged: true
  ```

Note: If kubelet is run inside a container that uses the hyperkube image, kubelet always reports that AppArmor as disabled regardless of the actual state. The same changes above must be made and applied to the YAML file.

Update the NameSpace Name

In the YAML file, all the namespaced objects such as ServiceAccount, ConfigMap, Deployment are created under the `nsx-system` namespace. If you use a different namespace, replace all instances of `nsx-system`.

Enabling Backup and Restore

NCP supports the backup and restore feature in NSX. The supported resources are Namespace, Pod, and Service.

Note: NCP must be configured in policy mode.

To enable this feature, set `enable_restore` to **`True`** in `ncp.ini` and restart NCP.

```
[k8s]
enable_restore = True
```

You can check the status of a restore with an NSX CLI command. For example,

```
nsxcli
> get ncp-restore status
NCP restore status is INITIAL
```

The status can be INITIAL, RUNNING, or SUCCESS. INITIAL means the backup/restore feature is ready, but no restore is running. RUNNING means the restore process is running in NCP. SUCCESS means a restore completed successfully. If an error occurs during a restore, NCP will restart automatically and retry. If the status is RUNNING for a long time, check the NCP log for error messages.

The nsx-ncp-config ConfigMap

The NCP YAML file includes the nsx-ncp-config ConfigMap. You can update the options for your environment. Below is the nsx-ncp-config ConfigMap from `ncp-ubuntu-policy.yaml` for NCP 3.2.1.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nsx-ncp-config
  namespace: nsx-system
  labels:
    version: v1
data:
  ncp.ini: |
    [DEFAULT]

    # If set to true, the logging level will be set to DEBUG instead of the
    # default INFO level.
    #debug = False

    # If set to true, use syslog for logging.
    #use_syslog = False

    # The base directory used for relative log_file paths.
    #log_dir = <None>

    # Name of log file to send logging output to.
    #log_file = <None>

    # max MB for each compressed file. Defaults to 100 MB.
    #log_rotation_file_max_mb = 100

    # max MB for each compressed file for API logs.Defaults to 10 MB.
    #api_log_rotation_file_max_mb = 10

    # Total number of compressed backup files to store. Defaults to 5.
    #log_rotation_backup_count = 5

    # Total number of compressed backup files to store API logs. Defaults to 5.
    #api_log_rotation_backup_count = 5

    # Log level for the root logger. If debug=True, the default root logger
    # level will be DEBUG regardless of the value of this option. If this
    # option is unset, the default root logger level will be either DEBUG or
    # INFO according to the debug option value
    # Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
    #loglevel = <None>

    [coe]

    # Container orchestrator adaptor to plug in.
    #adaptor = kubernetes

    # Specify cluster for adaptor.
    #cluster = k8scluster

    # Log level for NCP modules (controllers, services, etc.). Ignored if debug
    # is True
    # Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
    #loglevel = <None>

    # Log level for NSX API client operations. Ignored if debug is True
    # Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
    #nsxlib_loglevel = <None>

    # Enable SNAT for all projects in this cluster. Modification of topologies
    # for existing Namespaces is not supported if this option is reset.
    #enable_snat = True

    # The time in seconds for NCP/nsx_node_agent to recover the connection to
    # NSX manager/container orchestrator adaptor/Hyperbus before exiting. If
    # the value is 0, NCP/nsx_node_agent won't exit automatically when the
    # connection check fails
    #connect_retry_timeout = 0

    # Enable system health status report for SHA
    #enable_sha = True

    [ha]

    # Time duration in seconds of mastership timeout. NCP instance will remain
    # master for this duration after elected. Note that the heartbeat period
    # plus the update timeout must not be greater than this period. This is
    # done to ensure that the master instance will either confirm liveness or
    # fail before the timeout.
    #master_timeout = 18

    # Time in seconds between heartbeats for elected leader. Once an NCP
    # instance is elected master, it will periodically confirm liveness based
    # on this value.
    #heartbeat_period = 6

    # Timeout duration in seconds for update to election resource. The default
    # value is calculated by subtracting heartbeat period from master timeout.
    # If the update request does not complete before the timeout it will be
    # aborted. Used for master heartbeats to ensure that the update finishes or
    # is aborted before the master timeout occurs.
    #update_timeout = <None>

    [k8s]
    # Kubernetes API server IP address.
    #apiserver_host_ip = <None>

    # Kubernetes API server port.
    #apiserver_host_port = <None>

    # Full path of the Token file to use for authenticating with the k8s API
    # server.
    client_token_file = /var/run/secrets/kubernetes.io/serviceaccount/token

    # Specify a CA bundle file to use in verifying the k8s API server
    # certificate.
    ca_file = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

    # Specify whether ingress controllers are expected to be deployed in
    # hostnework mode or as regular pods externally accessed via NAT
    # Choices: hostnetwork nat
    #ingress_mode = hostnetwork

    # Log level for the kubernetes adaptor. Ignored if debug is True
    # Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
    #loglevel = <None>

    # Specifying whether the TCP connections between the LoadBalancer service
    # and backend could be reused by multiple client requests.
    #lb_connection_multiplexing_enabled = False

    # Specifying the maximum number of multiplexing connections.
    #lb_connection_multiplexing_number = 6

    # User specified IP address for HTTP and HTTPS ingresses
    #http_and_https_ingress_ip = <None>

    # Set this option to configure the ability to allow a virtual IP that is
    # not in the range of external_ip_pools_lb specified in spec.loadBalancerIP
    # of K8s service of type LoadBalancer to be realized in NSX.When the value
    # is relaxed, any IP specified in spec.loadBalancerIP can be allowed. When
    # the value is strict, only IP within the range of external_ip_pools_lb
    # will be allowed.
    # Choices: relaxed strict
    #lb_ip_allocation = relaxed

    # Set this to True to enable NCP to create LoadBalancer on a Tier-1 for
    # LoadBalancer CRD. This option does not support LB autoscaling.
    #enable_lb_crd = False

    # Option to set the type of baseline cluster policy. ALLOW_CLUSTER creates
    # an explicit baseline policy to allow any pod to communicate any other pod
    # within the cluster. ALLOW_NAMESPACE creates an explicit baseline policy
    # to allow pods within the same namespace to communicate with each other.
    # ALLOW_NAMESPACE_STRICT inherits the behaviors of ALLOW_NAMESPACE, and
    # also restricts service talk to resources outside the cluster. By default,
    # no baseline rule will be created and the cluster will assume the default
    # behavior as specified by the backend.
    # Choices: <None> allow_cluster allow_namespace allow_namespace_strict
    #baseline_policy_type = <None>

    # Set this to True to enable NCP reporting NSX backend error to k8s object
    # using k8s event
    #enable_ncp_event = False

    # Set this to True to enable multus to create multiple interfaces for one
    # pod. Requires policy_nsxapi set to True to take effect. If passthrough
    # interface is used as additional interface, user should deploy the network
    # device plugin to provide device allocation information for NCP. Pod
    # annotations with prefix "k8s.v1.cni.cncf.io" cannot be modified once pod
    # is realized. User defined IP will not be allocated from the Segment
    # IPPool. The "gateway" in NetworkAttachmentDefinition is not used to
    # configure secondary interfaces, as the default gateway of Pod is
    # configured by the primary CNI on the main network interface. User must
    # define IP and/or MAC if no "ipam" is configured. Only available if node
    # type is HOSTVM and not to be leveraged in conjunction with 3rd party CNI
    # plugin
    #enable_multus = False

    # Set this to True to enable NSX restore support (only effective in NSX
    # Policy API mode).
    #enable_restore = False

    # nsx-node-agent will add iptables rules for K8s pod which has hostPort,
    # client packets to hostPort will be SNATed to node IP. We leverage portmap
    # plugin to add iptables DNAT rules for hostPort ingress traffic. This
    # hostPort feature is only supported on K8s Linux node.
    #enable_hostport_snat = False

    # If true, pod ip of statefulset will locate in the ip_range in annotation
    # of statefulset. It only works for policy mode.
    #statefulset_ip_range = False

    # If true, user can set ncp/subnets annotation on namespace to specify the
    # subnets for no-snat namespace. It only works for policy mode.
    #enable_namespace_subnets = False

    # If true, NCP will collect prometheus metrics and export the metrics
    # through the prometheus_metrics_port.On VMC metric monitoring will always
    # be enabled regardless of this option.
    #enable_prometheus_metrics = False

    # The port number for NCP to expose prometheus metrics.
    #prometheus_metrics_port = 8001

    [nsx_v3]

    # Set NSX API adaptor to NSX Policy API adaptor. If unset, NSX adaptor will
    # be set to the NSX Manager based adaptor. If unset or False, the NSX
    # resource ID in other options can be resource name or UUID
    #policy_nsxapi = False

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

    # Path to NSX client private key file. If specified, the nsx_api_user and
    # nsx_api_password options will be ignored. Must be specified along with
    # nsx_api_cert_file option
    #nsx_api_private_key_file = <None>

    # IP address of one or more NSX managers separated by commas. The IP
    # address should be of the form:
    # [<scheme>://]<ip_adress>[:<port>]
    # If scheme is not provided https is used. If port is not provided port 80
    # is used for http and port 443 for https.
    #nsx_api_managers = []

    # If True, skip fatal errors when no endpoint in the NSX management cluster
    # is available to serve a request, and retry the request instead
    #cluster_unavailable_retry = False

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

    # Specify one or a list of CA bundle files to use in verifying the NSX
    # Manager server certificate. This option is ignored if "insecure" is set
    # to True. If "insecure" is set to False and "ca_file" is unset, the
    # "thumbprint" will be used. If "thumbprint" is unset, the system root CAs
    # will be used to verify the server certificate.
    #ca_file = []

    # Specify one or a list of thumbprint strings to use in verifying the NSX
    # Manager server certificate. This option is ignored if "insecure" is set
    # to True or "ca_file" is defined.
    #thumbprint = []

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

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

    # Maximum number of times to retry a HTTP connection.
    #http_retries = 3

    # Maximum concurrent connections to all NSX managers. If multiple NSX
    # managers are configured, connections will be spread evenly across all
    # managers, rounded down to the nearest integer. Each NSX manager will have
    # at least 1 connection. This value should be a multiple of
    # [nsx_v3]nsx_api_managers length.
    #concurrent_connections = 10

    # The amount of time in seconds to wait before ensuring connectivity to the
    # NSX manager if no manager connection has been used.
    #conn_idle_timeout = 10
    # Number of times a HTTP redirect should be followed.
    #redirects = 2

    # Subnet prefix of IP block.
    #subnet_prefix = 24

    # Subnet prefix for v6 IP blocks
    #v6_subnet_prefix = 64

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

    # Indicates whether distributed firewall rules are logged. Option 'ALL'
    # will enable logging for all DFW rules (both DENY and ALLOW), and option
    # 'DENY' will enable logging only for DENY rules. Remove this config if no
    # logging is desired. When IPv6 is enabled this setting will not apply to
    # rules for allowing ND traffic.
    # Choices: ALL DENY <None>
    #log_firewall_traffic = <None>

    # Option to use native load balancer or not
    #use_native_loadbalancer = True

    # Option to auto scale layer 4 load balancer or not. If set to True, NCP
    # will create additional LB when necessary upon K8s Service of type LB
    # creation/update.
    #l4_lb_auto_scaling = True

    # Path to the default certificate file for HTTPS load balancing. Must be
    # specified along with lb_priv_key_path option
    #lb_default_cert_path = <None>

    # Path to the private key file for default certificate for HTTPS load
    # balancing. Must be specified along with lb_default_cert_path option
    #lb_priv_key_path = <None>

    # Option to set load balancing algorithm in load balancer pool object.
    # Choices: ROUND_ROBIN LEAST_CONNECTION IP_HASH WEIGHTED_ROUND_ROBIN
    #pool_algorithm = ROUND_ROBIN

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

    # Option to set load balancer persistence option. If cookie is selected,
    # cookie persistence will be offered.If source_ip is selected, source IP
    # persistence will be offered for ingress traffic through L7 load balancer
    # Choices: <None> cookie source_ip
    #l7_persistence = <None>

    # An integer for LoadBalancer side timeout value in seconds on layer 7
    # persistence profile, if the profile exists.
    #l7_persistence_timeout = 10800

    # Option to set load balancer persistence option. If source_ip is selected,
    # source IP persistence will be offered for ingress traffic through L4 load
    # balancer
    # Choices: <None> source_ip
    #l4_persistence = <None>

    # Name or ID of the container ip blocks that will be used for creating
    # subnets. If name, it must be unique. If policy_nsxapi is enabled, it also
    # support automatically creating the IP blocks. The definition is a comma
    # separated list: CIDR,CIDR,... Mixing different formats (e.g. UUID,CIDR)
    # is not supported.
    #container_ip_blocks = []

    # Resource ID of the container ip blocks that will be used for creating
    # subnets for no-SNAT projects. If specified, no-SNAT projects will use
    # these ip blocks ONLY. Otherwise they will use container_ip_blocks
    #no_snat_ip_blocks = []

    # Name or ID of the external ip pools that will be used for allocating IP
    # addresses which will be used for translating container IPs via SNAT
    # rules. If policy_nsxapi is enabled, it also support automatically
    # creating the ip pools. The definition is a comma separated list:
    # CIDR,IP_1-IP_2,... Mixing different formats (e.g. UUID, CIDR&IP_Range) is
    # not supported.
    #external_ip_pools = []

    # Name or ID of the top-tier router for the container cluster network,
    # which could be either tier0 or tier1. If policy_nsxapi is enabled, should
    # be ID of a tier0/tier1 gateway.
    #top_tier_router = <None>

    # Option to use single-tier router for the container cluster network
    #single_tier_topology = False

    # Resource ID of the external ip pools that will be used only for
    # allocating IP addresses for Ingress controller and LB service. If
    # policy_nsxapi is enabled, it also supports automatically creating the ip
    # pools. The definition is a comma separated list: CIDR,IP_1-IP_2,...
    # Mixing different formats (e.g. UUID, CIDR&IP_Range) is not supported.
    #external_ip_pools_lb = []

    # Name or ID of the NSX overlay transport zone that will be used for
    # creating logical switches for container networking. It must refer to an
    # already existing resource on NSX and every transport node where VMs
    # hosting containers are deployed must be enabled on this transport zone
    #overlay_tz = <None>

    # Resource ID of the lb service that can be attached by virtual servers
    #lb_service = <None>

    # Resource ID of the IPSet containing the IPs of all the virtual servers
    #lb_vs_ip_set = <None>
    # Enable X_forward_for for ingress. Available values are INSERT or REPLACE.
    # When this config is set, if x_forwarded_for is missing, LB will add
    # x_forwarded_for in the request header with value client ip. When
    # x_forwarded_for is present and its set to REPLACE, LB will replace
    # x_forwarded_for in the header to client_ip. When x_forwarded_for is
    # present and its set to INSERT, LB will append client_ip to
    # x_forwarded_for in the header. If not wanting to use x_forwarded_for,
    # remove this config
    # Choices: <None> INSERT REPLACE
    #x_forwarded_for = <None>

    # Resource ID of the firewall section that will be used to create firewall
    # sections below this mark section
    #top_firewall_section_marker = <None>

    # Resource ID of the firewall section that will be used to create firewall
    # sections above this mark section
    #bottom_firewall_section_marker = <None>

    # If this value is not empty, NCP will append it to nameserver list
    #dns_servers = []

    # Set this to True to enable NCP to report errors through NSXError CRD.
    #enable_nsx_err_crd = False

    # Maximum number of virtual servers allowed to create in cluster for
    # LoadBalancer type of services.
    #max_allowed_virtual_servers = 9223372036854775807

    # Edge cluster ID needed when creating Tier1 router for loadbalancer
    # service. Information could be retrieved from Tier0 router
    #edge_cluster = <None>

    # Maximum size of the buffer used to store HTTP request headers for L7
    # virtual servers in cluster. A request with header larger than this value
    # will be processed as best effort whereas a request with header below this
    # value is guaranteed to be processed.
    #lb_http_request_header_size = 1024

    # Maximum size of the buffer used to store HTTP response headers for all L7
    # virtual servers in cluster. A response with header larger than this value
    # will be dropped.
    #lb_http_response_header_size = 4096

    # Maximum server idle time in seconds for L7 virtual servers in cluster. If
    # backend server does not send any packet within this time, the connection
    # is closed.
    #lb_http_response_timeout = 60

    # Determines the behavior when a Tier-1 instance restarts after a failure.
    # If set to PREEMPTIVE, the preferred node will take over, even if it
    # causes another failure. If set to NON_PREEMPTIVE, then the instance that
    # restarted will remain secondary. Applicable to Tier-1 across cluster that
    # was created by NCP and has edge cluster configured.
    # Choices: PREEMPTIVE NON_PREEMPTIVE
    #failover_mode = NON_PREEMPTIVE

    # Set this to ACTIVATE to enable NCP enforced pool member limit for all
    # load balancer servers in cluster. Set this to CRD_LB_ONLY will only
    # enforce the limit for load balancer servers created using lb CRD. Set
    # this to DEACTIVATE to turn off all limit checks. This option requires
    # relax_scale_validation set to True, l4_lb_auto_scaling set to False, and
    # works on Policy API only. When activated, NCP will enforce a pool member
    # limit on LBS to prevent one LBS from using up all resources on edge
    # nodes.
    # Choices: DEACTIVATE ACTIVATE CRD_LB_ONLY
    #ncp_enforced_pool_member_limit = DEACTIVATE

    # Maximum number of pool member allowed for each small load balancer
    # service. Requires ncp_enforced_pool_member_limit set to ACTIVATE or
    # CRD_LB_ONLY to take effect.
    #members_per_small_lbs = 2000

    # Maximum number of pool member allowed for each medium load balancer
    # service. Requires ncp_enforced_pool_member_limit set to ACTIVATE or
    # CRD_LB_ONLY to take effect.
    #members_per_medium_lbs = 2000

    # Resource ID of the client SSL profile which will be used by Loadbalancer
    # while participating in TLS handshake with the client
    #client_ssl_profile = <None>

    # Set this to True to enable rule tag as cluster name in DFW logs for k8s.
    # When IPv6 is enabled, the tag will not be applied to rules created by NCP
    # for allowing ND traffic.
    #enable_rule_tag = True

    # Set this to enable logging for snat rule, supported choices are : None,
    # Basic and Extended.none for no logging, basic for logging for
    # namespacesnat rules, extended for logging for snat rules for
    # allnamespaces and services
    # Choices: none basic extended
    #snat_rule_logging = none

    # Log properties of virtual server for ingress/routeIt maps to two
    # parameters access_log_enabled andlog_significant_event_only of virtual
    # server.It decides whether to log and what events to recordby virtual
    # server.
    # Choices: none all significant
    #vs_access_log = none

    # The time in seconds before a released IP can be reallocated. This value
    # is used to determine if a previously exhuasted logical switch can be used
    # again for creating a new logical port
    #ip_reallocation_time = 60

    # Specify a custom cookie name for NSX default LB when l7_persistence type
    # is set to cookie. It has no effect if l7_persistence is not set.
    #cookie_name = <None>
    # This parameter indicate how firewall is applied to a traffic packet.
    # Firewall can be bypassed, or be applied to external/internal address of
    # NAT rule
    # Choices: MATCH_EXTERNAL_ADDRESS MATCH_INTERNAL_ADDRESS BYPASS
    #natfirewallmatch = MATCH_INTERNAL_ADDRESS

    [vc]

    #[nsx_v3]
    # Deprecated option: tier0_router
    # Replaced by [nsx_v3] top_tier_router

    # Deprecated option: deny_subnets_redistribution
    # Replaced by [nsx_v3] configure_t0_redistribution
```

The nsx-node-agent-config ConfigMap

The NCP YAML file includes the nsx-node-agent-config ConfugMap. You can update the options for your environment. Below is the nsx-node-agent-config ConfigMap from `ncp-ubuntu-policy.yaml`.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nsx-node-agent-config
  namespace: nsx-system
  labels:
    version: v1
data:
  ncp.ini: |
    [DEFAULT]

    # If set to true, the logging level will be set to DEBUG instead of the
    # default INFO level.
    #debug = False

    # If set to true, use syslog for logging.
    #use_syslog = False

    # The base directory used for relative log_file paths.
    #log_dir = <None>

    # Name of log file to send logging output to.
    #log_file = <None>

    # max MB for each compressed file. Defaults to 100 MB.
    #log_rotation_file_max_mb = 100

    # max MB for each compressed file for API logs.Defaults to 10 MB.
    #api_log_rotation_file_max_mb = 10

    # Total number of compressed backup files to store. Defaults to 5.
```

```
#log_rotation_backup_count = 5

# Total number of compressed backup files to store API logs. Defaults to 5.
#api_log_rotation_backup_count = 5

# Log level for the root logger. If debug=True, the default root logger
# level will be DEBUG regardless of the value of this option. If this
# option is unset, the default root logger level will be either DEBUG or
# INFO according to the debug option value
# Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
#loglevel = <None>

[coe]

# Container orchestrator adaptor to plug in.
#adaptor = kubernetes

# Specify cluster for adaptor.
#cluster = k8scluster

# Log level for NCP modules (controllers, services, etc.). Ignored if debug
# is True
# Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
#loglevel = <None>

# Log level for NSX API client operations. Ignored if debug is True
# Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
#nsxlib_loglevel = <None>

# Enable SNAT for all projects in this cluster. Modification of topologies
# for existing Namespaces is not supported if this option is reset.
#enable_snat = True

# The time in seconds for NCP/nsx_node_agent to recover the connection to
# NSX manager/container orchestrator adaptor/Hyperbus before exiting. If
# the value is 0, NCP/nsx_node_agent won't exit automatically when the
# connection check fails
#connect_retry_timeout = 0

# Enable system health status report for SHA
#enable_sha = True

[k8s]

# Kubernetes API server IP address.
#apiserver_host_ip = <None>

# Kubernetes API server port.
#apiserver_host_port = <None>

# Full path of the Token file to use for authenticating with the k8s API
# server.
client_token_file = /var/run/secrets/kubernetes.io/serviceaccount/token

# Specify a CA bundle file to use in verifying the k8s API server
```

```
    # certificate.
    ca_file = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

    # Specify whether ingress controllers are expected to be deployed in
    # hostnework mode or as regular pods externally accessed via NAT
    # Choices: hostnetwork nat
    #ingress_mode = hostnetwork

    # Log level for the kubernetes adaptor. Ignored if debug is True
    # Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
    #loglevel = <None>

    # Specifying whether the TCP connections between the LoadBalancer service
    # and backend could be reused by multiple client requests.
    #lb_connection_multiplexing_enabled = False

    # Specifying the maximum number of multiplexing connections.
    #lb_connection_multiplexing_number = 6

    # User specified IP address for HTTP and HTTPS ingresses
    #http_and_https_ingress_ip = <None>

    # Set this option to configure the ability to allow a virtual IP that is
    # not in the range of external_ip_pools_lb specified in spec.loadBalancerIP
    # of K8s service of type LoadBalancer to be realized in NSX.When the value
    # is relaxed, any IP specified in spec.loadBalancerIP can be allowed. When
    # the value is strict, only IP within the range of external_ip_pools_lb
    # will be allowed.
    # Choices: relaxed strict
    #lb_ip_allocation = relaxed


    # Set this to True to enable NCP to create LoadBalancer on a Tier-1 for
    # LoadBalancer CRD. This option does not support LB autoscaling.
    #enable_lb_crd = False

    # Option to set the type of baseline cluster policy. ALLOW_CLUSTER creates
    # an explicit baseline policy to allow any pod to communicate any other pod
    # within the cluster. ALLOW_NAMESPACE creates an explicit baseline policy
    # to allow pods within the same namespace to communicate with each other.
    # ALLOW_NAMESPACE_STRICT inherits the behaviors of ALLOW_NAMESPACE, and
    # also restricts service talk to resources outside the cluster. By default,
    # no baseline rule will be created and the cluster will assume the default
    # behavior as specified by the backend.
    # Choices: <None> allow_cluster allow_namespace allow_namespace_strict
    #baseline_policy_type = <None>

    # Set this to True to enable NCP reporting NSX backend error to k8s object
    # using k8s event
    #enable_ncp_event = False

    # Set this to True to enable multus to create multiple interfaces for one
    # pod. Requires policy_nsxapi set to True to take effect. If passthrough
    # interface is used as additional interface, user should deploy the network
    # device plugin to provide device allocation information for NCP. Pod
    # annotations with prefix "k8s.v1.cni.cncf.io" cannot be modified once pod
    # is realized. User defined IP will not be allocated from the Segment
    # IPPool. The "gateway" in NetworkAttachmentDefinition is not used to
    # configure secondary interfaces, as the default gateway of Pod is
    # configured by the primary CNI on the main network interface. User must
    # define IP and/or MAC if no "ipam" is configured. Only available if node
    # type is HOSTVM and not to be leveraged in conjunction with 3rd party CNI
    # plugin
    #enable_multus = False

    # Set this to True to enable NSX restore support (only effective in NSX
    # Policy API mode).
    #enable_restore = False

    # nsx-node-agent will add iptables rules for K8s pod which has hostPort,
    # client packets to hostPort will be SNATed to node IP. We leverage portmap
    # plugin to add iptables DNAT rules for hostPort ingress traffic. This
    # hostPort feature is only supported on K8s Linux node.
    #enable_hostport_snat = False

    # If true, pod ip of statefulset will locate in the ip_range in annotation
    # of statefulset. It only works for policy mode.
    #statefulset_ip_range = False

    # If true, user can set ncp/subnets annotation on namespace to specify the
    # subnets for no-snat namespace. It only works for policy mode.
    #enable_namespace_subnets = False

    [nsx_kube_proxy]

    [nsx_node_agent]

    # The log level of NSX RPC library. Ignored if debug is True
    # Choices: NOTSET DEBUG INFO WARNING ERROR CRITICAL
    #nsxrpc_loglevel = ERROR

    # The time in seconds for nsx_node_agent to backoff before re-using an
    # existing cached CIF to serve CNI request. Must be less than
    # config_retry_timeout.
    #config_reuse_backoff_time = 15

    # The OVS uplink OpenFlow port where to apply the NAT rules to.
    #ovs_uplink_port = <None>

    # Set this to True if you want to install and use the NSX-OVS kernel
    # module. If the host OS is supported, it will be installed by nsx-ncp-
    # bootstrap and used by nsx-ovs container in nsx-node-agent pod. Note that
    # you would have to add (uncomment) the volumes and mounts in the nsx-ncp-
    # bootstrap DS and add SYS_MODULE capability in nsx-ovs container spec in
    # nsx-node-agent DS. Failing to do so will result in failure of
    # installation and/or kernel upgrade of NSX-OVS kernelmodule.
    #use_nsx_ovs_kernel_module = False

    # The time in seconds for nsx_node_agent to call OVS command. Please
    # increase the time if OVS is in heavy load to create/delete ports
    #ovs_operation_timeout = 5

    # Set to true to allow the CNI plugin to enable IPv6 container interfaces
    #enable_ipv6 = False

    # Set to True if DHCP is configured on the "ovs_uplink_port". "auto" will
    # try to automatically infer it but it only works on CoreOS. On other types
    # host OS, it defaults to False
    # Choices: True False auto
    #is_dhcp_configured_on_ovs_uplink_port = auto

    # The MTU value for nsx-cni
    #mtu = 1500

    # The waiting time before nsx-node-agent returns response to CNI plugin,
    # there is a potential timing issue between port creation and related
    # firewall config update on Hypervisor host
    #waiting_before_cni_response = 0

    # If this option is True, nsx-ncp-bootstrap pod will install portmap plugin
    # from nsx-ncp image, nsx-ncp-cleanup pod will remove portmap plugin.
    #use_ncp_portmap = False
```

Apply the NCP YAML File

After you edit the NCP YAML file, you can apply the YAML file to create and run the resources.

Run the following command to apply the NCP YAML file. For example,

```
kubectl apply -f ncp-ubuntu.yaml
```

Run the following command to check the result. For example,

```
~# kubectl get pods -n nsx-system
nsx-ncp-5df7fdf8b9-b6lmx   1/1     Running       0          77s
nsx-ncp-bootstrap-rv6z2    1/1     Running       0          76s
nsx-ncp-bootstrap-tzbv2    1/1     Running       0          76s
nsx-ncp-bootstrap-z4tt5    1/1     Running       0          76s
nsx-node-agent-ghnjk       3/3     Running       0          76s
nsx-node-agent-jkrct       3/3     Running       0          76s
nsx-node-agent-tz6td       3/3     Running       0          76s
```

You can also run the command `kubectl get all -n nsx-system` to see more details.

Mount a Certificate File in the NCP Pod

You need to mount a certificate file in the NCP Pod to configure certificate-based authentication with NSX API, or to configure a default certificate for SSL offloading for the NSXload balancer.

For both cases, do the following:

- Create a secret with a certificate and a private key.

- Attach a secret volume to the NCP pod and mount the volume (see the ConfigMap sample below).

For certificate-based authentication with NSX API, specify the options `nsx_api_cert_file` and `nsx_api_private_key_file` under `[nsx_v3]` in the nsx-ncp-config ConfigMap with the mount path for the certificate and key.

For NSX load balancer SSL offloading, specify the options `lb_default_cert_path` and `lb_priv_key_path` under `[nsx_v3]` in the nsx-ncp-config ConfigMap with the mount path for the certificate and key.

ConfigMap section where you specify the paths to the certificate and key:

```
      volumes:
        - name: projected-volume
          projected:
            sources:
              # ConfigMap nsx-ncp-config is expected to supply ncp.ini
              - configMap:
                  name: nsx-ncp-config
                  items:
                    - key: ncp.ini
                      path: ncp.ini
              # To use cert based auth, uncomment and update the secretName,
              # then update ncp.ini with the mounted cert and key file paths
              #- secret:
              #    name: nsx-secret
              #    items:
              #      - key: tls.crt
              #        path: nsx-cert/tls.crt
              #      - key: tls.key
              #        path: nsx-cert/tls.key
              #- secret:
              #    name: lb-secret
              #    items:
              #      - key: tls.crt
              #        path: lb-cert/tls.crt
              #      - key: tls.key
              #        path: lb-cert/tls.key
              # To use JWT based auth, uncomment and update the secretName.
              #- secret:
              #    name: wcp-cluster-credentials
              #    items:
              #      - key: username
              #        path: vc/username
              #      - key: password
              #        path: vc/password
```

Configuring IPv6

You can configure NCP to support IPv6.

To configure IPv6, note the following:

- Only Policy mode is supported. For more information, see Chapter 2 Configuring Resources.
- Both single-tier and two-tier topologies are supported.
- For north-south traffic to work properly, the tier-0 gateway must have an IPv6 address.
- Kubernetes nodes must have an IPv6 address. Otherwise, there will be no connectivity between the nodes and pods, and TCP and HTTP liveness and readiness probes will not work. Either SLAAC, or static IPs can be used for Kubernetes nodes. The Kubernetes nodes can also be in dual-stack mode. In this case, you must register the node with an IPv6 address in Kubernetes. To do this, specify the IPv6 address with the `-node-ip` option as one of kubelet's startup parameters. Otherwise, kubelet will always prioritize the IPv4 address.
- The Kubernetes cluster must be created with an IPv6 service cluster network CIDR. Note that the maximum size for this subnet is 16 bits.
- In NCP config, you must disable SpoofGuard by setting `enable_spoofguard = False` in the `[nsx_v3]` section.
- In nsx-node-agent config, IPv6 must be enabled to instruct the CNI plugin to enable IPv6 in containers. To do this, set `enable_ipv6 = True` in the `[nsx-node-agent]` section. Make sure to set this configuration option before the bootstrap process for NCP is executed.
- All namespaces will be in no-SNAT mode. SNAT per service as well as any other SNAT capability are not enabled in IPv6.
- Dual stack for containers is not supported. Every container must have only an IPv6 address.
- Mixing IPv4 and IPv6 IP blocks in NCP configuration will result in a startup failure.

NCP with IPv6 has the following limitations:

- Creating NSX load balancers through LoadBalancer CRDs is not supported.
- Automatic scaling of NSX layer-4 load balancers is not supported.

Configuring Syslog

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

You can use one of the following methods to configure logging. For more information about logging in Kubernetes, see https://kubernetes.io/docs/concepts/cluster-administration/logging.

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

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

Create a Sidecar Container for Syslog

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

Procedure

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

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

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

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

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

2 Make sure the host path exists and is writable by the user `nsx-ncp`..

   a Run the following commands.

   ```
       mkdir -p <host-filesystem-log-dir-path>
       chmod +w <host-filesystem-log-dir-path>
   ```

   b Add the user `nsx-ncp` or change the mode of the host path to 777.

   ```
       useradd -s /bin/bash nsx-ncp
       chown nsx-ncp:nsx-ncp <host-filesystem-log-dir-path>
   or
       chmod 777 <host-filesystem-log-dir-path>
   ```

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

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

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

      stop
    }

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

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

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

Create a DaemonSet Replica for Syslog

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

Procedure

1 Create the DaemonSet yaml file. For example,

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

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

        stop
      }

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

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

      input(type="imfile"
        File="/var/log/syslog"
        Tag="nsx-cni"
        Ruleset="remote")
---
# rsyslog DaemonSet
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: rsyslog
  labels:
    component: rsyslog
    version: v1
spec:
  template:
    metadata:
      labels:
      component: rsyslog
      version: v1
spec:
  hostNetwork: true
  containers:
  - name: rsyslog
    image: example/rsyslog
    imagePullPolicy: IfNotPresent
    volumeMounts:
    - name: rsyslog-config-volume
      mountPath: /etc/rsyslog.d
    - name: log-volume
      mountPath: /var/log
    - name: container-volume
      mountPath: /var/lib/docker/containers
  volumes:
  - name: rsyslog-config-volume
    configMap:
    name: rsyslog-config
  - name: log-volume
    hostPath:
      path: /var/log
  - name: container-volume
    hostPath:
      path: /var/lib/docker/containers
```

2 Create the DaemonSet.

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

Example: Configuring Log Rotation and Syslog Running in a Sidecar Container

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

Creating the Log Directory and Configuring Log Rotation

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

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

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

```
cat <<EOF >  /etc/logrotate.d/nsx-ujo
/var/log/nsx-ujo/*.log {
       copytruncate
       daily
```

```
        size 100M
        rotate 4
        delaycompress
        compress
        notifempty
        missingok
}
EOF
```

Creating the NCP Replication Controller

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

```
cat <<EOF > /tmp/ncp.ini
[DEFAULT]
log_dir = /var/log/nsx-ujo
[coe]
cluster = k8s-cl1
[k8s]
apiserver_host_ip = 10.114.209.77
apiserver_host_port = 6443
ca_file = /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
client_token_file = /var/run/secrets/kubernetes.io/serviceaccount/token
insecure = True
ingress_mode = nat
[nsx_v3]
nsx_api_user = admin
nsx_api_password = Password1!
nsx_api_managers = 10.114.209.68
insecure = True
subnet_prefix = 29
[nsx_node_agent]
[nsx_kube_proxy]
ovs_uplink_port = ens192
EOF
```

- Create the config map from the ini file.

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

- Create the NCP rsyslog config.

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

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

            stop
        }

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

- Create the config map from the above.

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

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

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

- Create NCP with the above specification.

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

Creating the NSX Node Agent Daemon Set

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

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

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

            stop
        }

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

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

- Create the `configmap` from the above.

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

- Create the DaemonSet with the `configmap` sidecar.

```
cat <<EOF > /tmp/nsx-node-agent-rsyslog.yml
# nsx-node-agent DaemonSet
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: nsx-node-agent
  labels:
    tier: nsx-networking
    component: nsx-node-agent
    version: v1
```

```
spec:
  template:
    metadata:
      annotations:
        container.apparmor.security.beta.kubernetes.io/nsx-node-agent: localhost/node-
agent-apparmor
      labels:
        tier: nsx-networking
        component: nsx-node-agent
        version: v1
    spec:
      hostNetwork: true
      tolerations:
      - key: "node-role.kubernetes.io/master"
        operator: "Exists"
        effect: "NoSchedule"
      containers:
        - name: nsx-node-agent
          # Docker image for NCP
          image: nsx-ujo-docker-local.artifactory.eng.vmware.com/nsx-ncp:ob-6236425
          imagePullPolicy: IfNotPresent
          # override NCP image entrypoint
          command: ["nsx_node_agent"]
          livenessProbe:
            exec:
              command:
                - /bin/sh
                - -c
                - ps aux | grep [n]sx_node_agent
            initialDelaySeconds: 5
            periodSeconds: 5
          securityContext:
            capabilities:
              add:
                - NET_ADMIN
                - SYS_ADMIN
                - SYS_PTRACE
                - DAC_READ_SEARCH
          volumeMounts:
          # ncp.ini
          - name: config-volume
            mountPath: /etc/nsx-ujo
          # mount openvswitch dir
          - name: openvswitch
            mountPath: /var/run/openvswitch
          # mount CNI socket path
          - name: cni-sock
            mountPath: /var/run/nsx-ujo
          # mount container namespace
          - name: netns
            mountPath: /var/run/netns
          # mount host proc
          - name: proc
            mountPath: /host/proc
            readOnly: true
          - name: log-volume
            mountPath: /var/log/nsx-ujo
        - name: nsx-kube-proxy
          # Docker image for NCP
          image: nsx-ujo-docker-local.artifactory.eng.vmware.com/nsx-ncp:ob-6236425
          imagePullPolicy: IfNotPresent
          # override NCP image entrypoint
          command: ["nsx_kube_proxy"]
          livenessProbe:
            exec:
              command:
                - /bin/sh
                - -c
                - ps aux | grep [n]sx_kube_proxy
            initialDelaySeconds: 5
            periodSeconds: 5
          securityContext:
            capabilities:
              add:
                - NET_ADMIN
                - SYS_ADMIN
                - SYS_PTRACE
                - DAC_READ_SEARCH
          volumeMounts:
          # ncp.ini
          - name: config-volume
            mountPath: /etc/nsx-ujo
          # mount openvswitch dir
          - name: openvswitch
            mountPath: /var/run/openvswitch
          - name: log-volume
            mountPath: /var/log/nsx-ujo
        - name: rsyslog
          image: jumanjiman/rsyslog
          imagePullPolicy: IfNotPresent
          volumeMounts:
          - name: rsyslog-config-volume
            mountPath: /etc/rsyslog.d
            readOnly: true
          - name: log-volume
            mountPath: /var/log/nsx-ujo
      volumes:
        - name: config-volume
          configMap:
            name: nsx-ncp-config-with-logging
        - name: cni-sock
          hostPath:
            path: /var/run/nsx-ujo
        - name: netns
          hostPath:
            path: /var/run/netns
        - name: proc
          hostPath:
            path: /proc
        - name: openvswitch
```

```
          hostPath:
            path: /var/run/openvswitch
        - name: rsyslog-config-volume
          configMap:
            name: rsyslog-config-node-agent
        - name: log-volume
          hostPath:
            path: /var/log/nsx-ujo/
EOF
```

- Create the DaemonSet.

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

Security Considerations

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

Restrict NCP to Run Only on Designated Nodes

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

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

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

Ensure that the Docker Engine is Up To Date

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

Disallow NET_ADMIN and NET_RAW Capabilities of Untrusted Containers

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

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

Role-Based Access Control

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

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

The following guidelines are recommended:

- Restrict access to Kubernetes API tokens to pods which need them.
- Restrict access to NCP ConfigMap and NSX API client certificate's TLS secrets to the NCP pod.
- Block access to Kubernetes networking API from pods that do not require such access.
- Add a Kubernetes RBAC policy to specify which pods can have access to the Kubernetes API.

The recommended RBAC policy is already in the NCP YAML file and will be effective when you install NCP.

Configuring Network Resources

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

Limits on Tags and Labels

Tags have the following limits:

- The tag scope has a limit of 128 characters.
- The tag value has a limit of 256 characters.
- Each object can have a maximum of 30 tags.

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

Labels have the following limits:

- A pod can have no more than 25 labels.
- A namespace can have no more than 27 labels.
- An Ingress controller pod can have no more than 24 labels.

Network Policies

A `NetworkPolicy` resource has the following limitations:

- A `podSelector` or `namespaceSelector` cannot have more than 4 `matchLabels`.
- In an `ingress from` section or `egress to` section, if you have a single element that specifies both `namespaceSelector` and `podSelector`, the `namespaceSelector` must not select more than 5 namespaces. Otherwise, you will get an error. An example of such a specification (notice that there is no hyphen before `podSelector`):

  ```
  - namespaceSelector:
      matchLabels:
        project: myproject
    podSelector:
      matchLabels:
        role: db
  ```

- For any ingress or egress rule, the total number of `namespaceSelector`s plus `podSelector`s cannot be more than 5. For example, the following is not allowed because the rule has a total of 6 selectors:

  ```
  ingress:
      - namespaceSelector:
          matchLabels:
            project: myproject1
      - namespaceSelector:
          matchLabels:
            project: myproject2
      - namespaceSelector:
          matchLabels:
            project: myproject3
      - podSelector:
          matchLabels:
            role: db
      - podSelector:
          matchLabels:
            role: app
      - podSelector:
          matchLabels:
            role: infra
  ```

- Allow-all-egress network policy with `namedPorts` in the `to.ports` section is not supported.

- In Policy mode, when specifying `matchExpressions` in `podSelector` or `namespaceSelector`, at most one `In` operator is allowed. If you specify both `namespaceSelector` and `podSelector`, you can have at most one `In` operator for one of them. You cannot have an `In` operator for both of them. For example, the following specification is not allowed:

  ```
  - namespaceSelector:
      matchExpressions:
      - {key: key1, operator: In, values: [value1]}
    podSelector:
      matchExpressions:
      - {key: key1, operator: In, values: [value1]}
  ```

- In Manager mode, when specifying `matchExpressions` in `podSelector`, `namespaceSelector`, or both `namespaceSelector` and `podSelector`, there is no restriction on the number of `In` operators.
- Network policy with SCTP in the `protocol` field is not supported.

To work around the limitations, you can create a network policy with `matchLabels` that are more specific, or replace one network policy with several network policies that do not require more than 5 selectors.

If a network policy is not supported by NCP, NCP will annotate it with an error. You can update the network policy to fix the error and NCP will process it again.

If a network policy has only egress specified but not ingress, no firewall rule is created for ingress traffic. Similarly, if a network policy has only ingress specified but not egress, no firewall rule is created for egress traffic.

When NSX App ID firewall rules are applied to resources created by NCP, only the "Segment" and "Segment Port" group membership criteria are supported. Note that the App ID firewall rule feature is supported only if NCP is configured to use NSX policy API.

Clean Up Kubernetes Nodes

You can clean up file system changes made by the bootstrap container.

Note If the nsx-node-agent DaemonSet is removed, OVS is no longer running on the host (in the container or in the host's PID).

You can create the nsx-ncp-cleanup DaemonSet to undo the system changes made by the nsx-ncp-bootstrap DaemonSet. This DaemonSet must only be created if you previously applied the NCP YAML file (`ncp-ubuntu.yaml` or `ncp-rhel.yaml`) and have not deleted them. Note that the nsx-ncp-cleanup DaemonSet will uninstall NSX CNI, which will result in an invalid Kubernetes node state.

To create the DaemonSet, perform the following steps:

- Delete the nsx-ncp-bootstrap and nsx-node-agent DaemonSets. For example, you can run the following commands with the appropriate namespace name:

  ```
  kubectl delete ds nsx-ncp-bootstrap -n <namespace>
  kubectl delete ds nsx-node-agent -n <namespace>
  ```

- Run `kubectl apply -f ncp-cleanup-ubuntu.yaml` or `kubectl apply -f ncp-cleanup-rhel.yaml`, depending on your host OS, from the command line on the Kubernetes master node.

To make the node usable again, run `kubectl apply -f ncp-ubuntu.yaml` or `kubectl apply -f ncp-rhel.yaml`, depending on your host OS.

Installing NCP in a Tanzu Application Service Environment

4

Tanzu Application Service (TAS) is an open source platform-as-a-service (PaaS) provider. You can install NSX Container Plugin (NCP) in a TAS environment to provide networking services.

VMs created through the TAS Ops Manager must have layer 3 connectivity to the container network to access the features.

High availability (HA) is automatically enabled.

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

If you have a default ASG (App Security Group) in your TAS environment, you must create a global firewall section outside of NCP to replace the default ASG.

This chapter includes the following topics:

Install NCP in a Tanzu Application Service Environment

NCP is installed through the Tanzu Application Service (TAS) Ops Manager graphical user interface.

Prerequisites

A fresh installation of Ops Manager, NSX, and TAS. Make sure that Ops Manager is installed first, then NSX, and then TAS. For more information, see the Tanzu Application Service documentation.

Procedure

1 Download the NCP installation file for TAS.

   The file name is `VMware-NSX-T.<version>.<build>.pivotal`.

2 Log in to Ops Manager as an administrator.

3 Click **Import a Product**.

4 Select the file that was downloaded.

5 Click the **Ops Manager Director for VMware vSphere** tile.

6 In the **Settings** tab for **vCenter Config**, select **NSX Networking** and for **NSX Mode**, select **NSX-T**.

7 In the **NSX CA Cert** field, provide the certificate in PEM format.

8 Click **Save**.

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

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

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

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

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

14 Click **Save**.

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

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

17 Enter the address of the NSX Manager.

18 Select the method for NSX Manager authentication.

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

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

20 Click **Save**.

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

22 Enter the **TAS Foundation Name**.

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

23 Enter the **Overlay Transport Zone**.

24 Enter the **Tier-0 Router**.

25 Specify one or more **IP Blocks of Container Networks**.

a Click **Add**.

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

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

26 Specify the subnet prefix of the container networks.

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

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

   a Click **Add**.

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

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

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

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

31 (Optional) Enable or disable the following options.

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

32 Click **Save**.

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

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

   b Click **Save**.

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

35 Click **Apply Changes**.

Handling Custom Labels Created in TAS

NCP can handle custom labels that you create on an app in TAS. NCP will create corresponding tags in NSX for those labels.

In TAS, you can create labels with the following command. For example,

```
cf curl v3/apps/<app-guid> -X PATCH -d '{"metadata": {"labels": {"aaa": "111", "bbb": "222"}}
```

You can also delete a label by setting the value of a label to `null`, For example,

```
cf curl v3/apps/<app-guid> -X PATCH -d '{"metadata": {"labels":{"aaa": null}}}'
```

These commands trigger events that NCP can retrieve. For a new label, for example, `"aaa":"111"`, NCP will create the tag `app_label/aaa:111` for the logical ports of all the app instances. If you delete a label, the corresponding tag will be removed.

Load Balancing

5

The NSX load balancer is integrated with Kubernetes.

This chapter includes the following topics:

Configuring Load Balancing

You can configure NSX load balancer integration with NCP for Kubernetes LoadBalancer services and Ingress resources.

Configuring a Kubernetes service of type LoadBalancer will create a layer 4 load balancer, and configuring a Kubernetes Ingress resource will create a layer 7 load balancer.

To configure load balancing, in the nsx-ncp-config ConfigMap:

1. Set `use_native_loadbalancer` = **`True`**.
2. (Optional) Set `pool_algorithm` to **`ROUND_ROBIN`** or **`LEAST_CONNECTION/IP_HASH`**. The default is **`ROUND_ROBIN`**.
3. (Optional) Set `service_size` = **`SMALL`**, **`MEDIUM`**, or **`LARGE`**. The default is **`SMALL`**. In policy mode, set this value to match the pool allocation size of the tier-1 gateway.

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

For details about what NSX load balancers of different sizes support, see the .

After the load balancer is created, the load balancer size cannot be changed by updating the configuration file. It can be changed through the NSX Manager UI or API.

You can configure an IPSet which will be populated with the IPs of all the virtual servers by NCP. To enable this feature, set the option `lb_vs_ip_set` in the nsx-ncp-config ConfigMap to be the name or UUID of an IPSet. The IPSet can be shared by multiple clusters. The IPs must be unique across all clusters. NCP will manage the allocation of the IPs.

Setting Persistence for Layer 4 and Layer 7 Load Balancer

You can specify a persistence setting with the parameters `l4_persistence` and `l7_persistence` in the NCP ConfigMap.

The available option for layer 4 persistence is source IP. The available options for layer 7 persistence are cookie and source IP. The default is `<None>`. For example,

```
    # Choice of persistence type for ingress traffic through L7 Loadbalancer.
    # Accepted values:
    # 'cookie'
    # 'source_ip'
    l7_persistence = cookie

    # Choice of persistence type for ingress traffic through L4 Loadbalancer.
    # Accepted values:
    # 'source_ip'
    l4_persistence = source_ip
```

For layer-7 persistence, you can also specify the name of the cookie.

```
     # Specify a custom cookie name for NSX default LB when l7_persistence type
     # is set to cookie. It has no effect if l7_persistence is not set.
     #cookie_name = <None>
```

For a Kubernetes LoadBalancer service, you can also specify `sessionAffinity` on the service spec to configure persistence behavior for the service if the global layer 4 persistence is turned off, that is, `l4_persistence` is set to `<None>`. If `l4_persistence` is set to `source_ip`, the `sessionAffinity` on the service spec can be used to customize the persistence timeout for the service. The default layer 4 persistence timeout is 10800 seconds (same as that specified in the Kubernetes documentation for services (https://kubernetes.io/docs/concepts/services-networking/service). All services with default persistence timeout will share the same NSX load balancer persistence profile. A dedicated profile will be created for each service with a non-default persistence timeout.

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

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
-----
apiVersion: v1
kind: Service
metadata:
  name: tea-svc <==== same as the Ingress backend above
  labels:
    app: tea
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
    name: tcp
  selector:
    app: tea
  type: LoadBalancer
```

Ingress

NCP will create one layer 7 load balancer for Ingresses with TLS specification, and one layer 7 load balancer for Ingresses without TLS specification. You can also create CRDs (CustomResourceDefinitions) to handle Ingress scaling.

Note the following:

- All Ingresses will get a single IP address.
- The Ingress resource is allocated an IP address from the external IP pool specified by the `external_ip_pools` option in the `[nsx_v3]` section in `ncp.ini`. The load balancer is exposed on this IP address and the HTTP and HTTPS ports (80 and 443).
- The Ingress resource is allocated an IP address from the external IP pool specified by the `external_ip_pools_lb` option in the `[nsx_v3]` section in `ncp.ini`. If the `external_ip_pools_lb` option does not exist, the pool specified by `external_ip_pools` is used. The load balancer is exposed on this IP address and the HTTP and HTTPS ports (80 and 443).
- You can change to a different IP pool by changing the configuration and restarting NCP.
- You can specify a default certificate for TLS. See below for information about generating a certificate and mounting a certificate into the NCP pod.
- Ingresses without TLS specification will be hosted on HTTP virtual server (port 80).
- Ingresses with TLS specification will be hosted on HTTPS virtual server (port 443). The load balancer will act as an SSL server and terminate the client SSL connection.
- Modification of Ingress by adding or removing the TLS section is supported. When the `tls` key is removed from the Ingress specification, the Ingress rules will be transferred from the HTTPS virtual server (port 443) to the HTTP virtual server (port 80). Similarly, when the `tls` key is added to Ingress specification, the Ingress rules are transferred from the HTTP virtual server (port 80) to the HTTPS virtual server (port 443).
- If there are duplicate rules in Ingress definitions for a single cluster, only the first rule will be applied. The other Ingresses with the duplicate rules will be annotated with error. For example, if you create two Ingresses with the same `host` and `path`, and one Ingress is TLS while and the other is non-TLS, and `kubernetes.io/ingress.allow-http` is `false`, the two rules will be created on different virtual servers and will not conflict with each other. However, if `kubernetes.io/ingress.allow-http` is `true`, the Ingress that is applied later will be annotated with an error. See the "Errors" section below for more information.
- Only a single Ingress with a default backend is supported per cluster. Traffic not matching any Ingress rule will be forwarded to the default backend.
- If there are multiple Ingresses with a default backend, only the first one will be configured. The others will be annotated with an error. See the "Errors" section below for more information.
- The rules are applied in the following order:

  a Rules with both `host` and `path` specified.

    1 Rules with the `Exact` `pathType`.

    2 Rules with the `Prefix` `pathType`.

    3 Rules with the `Regex` `pathType`.

b Rules with `host` or `path` specified.

1. Rules with the `Exact` `pathType`.
2. Rules with the `Prefix` `pathType`.
3. Rules with the `Regex` `pathType`.

Note: If multiple paths match a request, precedence will be given to the longest matching path.

About `pathType`:

- Starting with Kubernetes 1.19 `pathType` is mandatory.
- If `use-regex` is set, it will only take effect if `pathType` is `ImplementationSpecific`.
- If `use-regex` is not set, `ImplementationSpecific` is treated the same as `Exact`. `pathType` takes precedence over other NCP annotations.
- Two matching paths with different `pathType`s can co-exist.
- For the `Prefix` type, `/foo` will match `/foo/`, `/foo/bar` but not `/foo` or `/foobar`. To match `/foo`, add the `Exact` path `/foo` to the Ingress rule.

- Hostname wildcards

  When specifying hostname, you can specify the exact name, such as abc.example.com, or use a wildcard, such as *.example.com. Note that the wildcard will match one or more DNS labels. The following table shows the match results if you specify *.example.com.

Host Header	Match Result
abc.example.com	Match
abc.def.example.com	Match
example.com	No match

- (This is applicable if you use Policy mode.) If a TLS Ingress is created with a default backend, it is recommended that you set up the default backend to respond to both HTTP and HTTPS requests because:
  - The load balancer will terminate TLS and send HTTP requests to the default backend server if there is a TLS Ingress (in the cluster for the Kubernetes/TKGI use case, or in the same namespace for the Project Pacific use case) with host which matches the host in the request.
  - The load balancer will re-encrypt and send HTTPS request to the backend server if there is no TLS Ingress (in the cluster for the Kubernetes/TKGI use case, or in the same namespace for the Project Pacific use case) with host which matches the host in the request.
- The `IngressClass` resource is supported.
- The `pathType` attribute is supported.

- JWT (JSON Web Token) client authentication is supported. This feature requires NSX 3.0.0 or later.

Starting with Kubernetes 1.18, the `kubernetes.io/ingress.class` annotation is deprecated and replaced by the IngressClass resource. To specify NSX as the Ingress controller in the IngressClass resource, set the controller parameter to `k8s.io/nsx`. For example:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: nsx-lb
  annotations:
    ingressclass.kubernetes.io/is-default-class: true
spec:
  controller: k8s.io/nsx
```

To specify a third-party Ingress controller, set the controller parameter to `k8s.io/<controller name>`. For example:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: nginx-lb
  annotations:
    ingressclass.kubernetes.io/is-default-class: false
spec:
  controller: k8s.io/ingress-nginx
```

To set the IngressClass as the default for the cluster, set the annotation `ingressclass.kubernetes.io/is-default-class` to `true`. See the example above.

You must not use both the `kubernetes.io/ingress.class` annotation and the IngressClass resource.

If you manually configure an `Endpoints` object, be sure to set the `targetRef` parameter. The value should be the UID of the backend pod. For example:

```
apiVersion: networking.k8s.io/v1
kind: Endpoints
metadata:
  name: tea-svc
subsets:
  - addresses:
    - ip: 172.26.0.2
      targetRef:
        uid: 4378e0ae-5837-49c6-a0b2-178dced8eb1e
...
```

Feature Annotations

The following table lists annotations that NCP supports:

Annotation	Description	Supported Values	Default Value
kubernetes.io/ingress.allow-http	Enables HTTP requests in addition to HTTPS	true, false	true
ncp/use-regex	Enables path pattern matching	true, false	false
ingress.kubernetes.io/rewrite-target	Rewrites path of incoming request	■ Path starting with '/' ■ Path with a numbered placeholder for a group captured in a regular expression, for example, /$1	No default value
ncp/http-redirect	Redirects HTTP requests to HTTPS	true, false	false
kubernetes.io/ingress.class	Indicates which Ingress controller is responsible for this Ingress	nsx, nginx, etc.	nsx
nsx/loadbalancer	Places an Ingress on a dedicated load balancer	Name of a LoadBalancer CRD	No default value
ncp/allowed-source-range	Limits Ingress access by request source IP	Comma-separated list of CIDRs, IP addresses, or IP ranges.	No default value
ncp/ssl-mode	Selects SSL mode for all the rules in an Ingress	offload, reencrypt, passthrough	offload
ncp/jwt-alg	Determines the algorithm to be used to validate JWT signature. Enables JWT client authentication when set with ncp/jwt-secret.	HS256, RS256	No default value
ncp/jwt-secret	Specifies the name of the Kubernetes secret that contains the JWT secret or public key used for signature validation. Enables JWT client authentication when set with ncp/jwt-alg.	Name of a Kubernetes secret	No default value
ncp/jwt-token	Additional location to search for JWT in the HTTP request.	_arg_<param_name>, _cookie_<cookie_name>	No default value
ncp/jwt-realm	Specifies the Realm header returned with 401 when authentication failed.	String indicating the realm	Hostname of the ingress rule
ncp/jwt-preserve	Option to keep JWT and pass to backend after successful authentication.	true, false	true

Details about the annotations:

- Path Regex (Regular Expression) Matching
  - You can enable or disable regular expression matching of the Ingress `path` (but not `host`) parameter using the annotation `ncp/use-regex`. If set to `false`, exact path matching will be performed by doing the `equals` match. If set to `true`, regular expression matching will be performed by adding the start of string character (^) and end of string character ($) to the path so that the entire request URI matches the pattern. Note that when using the `OR` operator (|), always specify the scope with parentheses so that ^ and $ apply to all the operands. For example, `path: /(tea|coffee|milk)`. An Ingress specification example:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    kubernetes.io/ingress.class: "nsx"
    #/tea/cup will be served by tea-svc:80
    ncp/use-regex: "True"
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      # for this tea and coffee NCP will configure regex rule
      - path: /tea/(.*)
        pathType: ImplementationSpecific
        backend:
          service:
            name: tea-svc
            port:
              number: 80
      - path: /coffee/(.*)
        pathType: ImplementationSpecific
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
      # for juice and alcohol NCP will configure Exact and Prefix rules
      - path: /juice
        pathType: Exact
        backend:
          service:
            name: juice-svc
            port:
              number: 80
      - path: /alcohol
        pathType: Prefix
```

```
          backend:
            service:
              name: bar
                number: 80
```

- Updating Ingresses prior to upgrading NCP

  This is required only if you have Ingresses requiring all sub-path matching using the characters '.' and '*'.

  1. Update the Ingresses to include the annotation `ncp/use-regex: true`.
  2. For all sub-path matching, if you have paths such as `/coffee/*` or `/*`, change them to `/coffee/.*` and `/.*`.

     `/coffee/.*` will match `/coffee/`, `/coffee/a`, `/coffee/b`, `/coffee/a/b`, and so on. `/.*` will match `/coffee`, `/tea`, `/coffee/a`, and so on. Omitting the path will produce the same behavior as `path: /.*`.

- In the following example of the annotation `ingress.kubernetes.io/rewrite-target`, the paths `/tea` and `/coffee` will be rewritten to `/` before the URL is sent to the backend service.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        pathType: ImplementationSpecific
        backend:
          service:
            name: tea-svc
            port:
              number: 80
      - path: /coffee
        pathType: ImplementationSpecific
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
```

If `path` is specified using a regular expression, the captured groups are saved in numbered placeholders in the form $1, $2, and so on. These placeholders can be used as parameters in the `ingress.kubernetes.io/rewrite-target` annotation. Named capture groups and named placeholders are not supported. For example,

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    kubernetes.io/ingress.class: "nsx"
    ncp/use-regex: "true"
    #/tea/cup will be rewritten to /cup before sending request to endpoint
    ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea/(.*)
        pathType: ImplementationSpecific
        backend:
          service:
            name: tea-svc
            port:
              number: 80
      - path: /coffee/(.*)
        pathType: ImplementationSpecific
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
```

- About the annotation `kubernetes.io/ingress.allow-http`:
  - If the annotation is set to `false`, rules will be created for the HTTPS virtual server.
  - If the annotation is set to `true` or missing, rules will created for both HTTP and HTTPS virtual servers. Note that HTTPS rules will be created only if the TLS section is present in the Ingress specification.
- About the annotation `ncp/http-redirect`:
  - If the annotation is set to `false`, Incoming HTTP traffic (port 80) to HTTP Virtual Server will not be redirected to HTTPS Virtual Server.
  - If the annotation is set to `true`, Incoming HTTP traffic (port 80) to HTTP Virtual Server will be redirected to HTTPS Virtual Server (port 443).

  This annotation is only valid if the TLS section is present. This annotation takes precedence over `kubernetes.io/ingress.allow-http`. When set to `true`, it will direct matched HTTP traffic to HTTPS, regardless of how `kubernetes.io/ingress.allow-http` is set.

- About the annotation `kubernetes.io/ingress.class`:
  - If the value is `nsx`, this ingress will be handled by NCP. If any other value is specified, the Ingress will be ignored by NCP. For more info see Third-party Ingress Controllers.
- For more information about the annotation `nsx/loadbalancer`, see LoadBalancer CRDs to Handle Ingress Scaling.
- About the annotation `ncp/allowed-source-range`:
  - When set, an incoming request will only be accepted if its source IP is included in this annotation. Otherwise, traffic will be dropped.
  - You can specify a combination of CIDR, IP addresses, and IP ranges. For example, 1.1.1.1/24, 2.2.2.2, 3.3.3.3-4.4.4.4.
  - Example YAML:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    kubernetes.io/ingress.class: "nsx"
    #Source IP addresses allowed to access ingress URL
    ncp/allowed-source-range: "192.168.128.0/17, 192.168.64.0-192.168.64.255,
192.168.16.32"
spec:
  tls:
  - hosts:
    - cafe.example.com
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        pathType: ImplementationSpecific
        backend:
          service:
            name: tea-svc
            port:
              number: 80
      - path: /coffee
        pathType: ImplementationSpecific
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
```

- About the annotation `ncp/ssl-mode`:
  - Supported in Policy mode only.

  - This annotation applies to all the rules in an Ingress and the load balancer will operate in the selected SSL mode for these rules.

    If `ncp/ssl-mode` is set to `reencrypt` or `passthrough`, `kubernetes.io/ingress.allow-http` must be set to `False`. This annnotation cannot be set to `passthrough` if the `ingress.kubernetes.io/rewrite-target`, `ncp/use-regex` or `ncp/allowed-source-range` annotation is set.

    If `ncp/ssl-mode` is set to `passthrough`, the attribute `path` in the `rules` specification is not supported.

- About JWT client authentication:
  - To enable JWT client authentication for all rules under an Ingress, both `ncp/jwt-alg` and `ncp/jwt-secret` need to be set to a valid value. When enabled, incoming HTTP traffic will only be passed to the backend if bearing valid JSON web token. Otherwise, traffic will be rejected with the 401 status.
  - This feature is incompatible with the following annotations:
    - `kubernetes.io/ingress.allow-http: true`
    - `ncp/http-redirect: true`
    - `ncp/ssl-mode: passthrough`
  - `ncp/jwt-alg`:
    - Supported symmetrical algorithms: HS256
    - Supported asymmetrical algorithms: RS256
  - `ncp/jwt-secret`:
    - A symmetrical key or public certificate must be configured in a Kubernetes secret with the name specified in this annotation under the same namespace as the Ingress.
    - For symmetrical algorithms, the secret must be stored in the `jwt.key` field.
    - For asymmetrical algorithms, the public cert must be stored in the `tls.crt` field.
    - JWT client authentication will be disabled if the symmetrical secret or public certificate is not stored in the locations mentioned above, or if the data stored in the secret is invalid.
  - `ncp/jwt-token`:
    - Only one item can be configured in this annotation.
    - `_arg_<param_name>`: For JWT passed in as a URI parameter. Specify the parameter name that contains JWT.
    - `_cookie_<cookie_name>`: For JWT passed in as a cookie. Specify the cookie name that contains JWT.

Errors

Errors are annotated to the Ingress resource. The error key is `ncp/error.loadbalancer` and the warning key is `ncp/warning.loadbalancer`. The possible error and warning are:

- `ncp/error.loadbalancer`: DEFAULT_BACKEND_IN_USE

  This error indicates that an Ingress with a default backend already exists. The Ingress will be inactive. This error will occur if (1) this Ingress is for HTTP and another Ingress for HTTP with a default backend exists; (2) this Ingress is for HTTPS and another Ingress for HTTPS with a default backend exists; or (3) this Ingress is for HTTP and HTTPS and another Ingress for HTTP and HTTPS with a default backend exists. To fix the error, delete and recreate the Ingress with a correct specification.

- `ncp/warning.loadbalancer`: SECRET_NOT_FOUND

  This error indicates that the secret specified in the Ingress specification does not exist, or if `ncp/jwt-alg` and `ncp/jwt-secret` are anootated but the secret cannot be found under the same namespace as the Ingress. The Ingress will be partially active. To fix the error, create the missing secret. Note that once a warning is in the annotation, it will not be cleared during the life cycle of the Ingress resource.

- `ncp/warning.loadbalancer`: INVALID_INGRESS

  This error indicates that one of the following conditions is true. The Ingress will be inactive. To fix the error, delete and recreate the Ingress with a correct specification.

  - An Ingress rule conflicts with another Ingress rule in the same Kubernetes cluster. Conflicts are determined only for Ingresses with the same match strategy, that is, the same `ncp/use-regex` annotation value.
  - The `kubernetes.io/ingress.allow-http` annotation is set to `false` and the Ingress does not have a TLS section.
  - The `ncp/http-redirect` annotation is set to `true` and the Ingress does not have a TLS section.
  - An Ingress rule does not have `host` and `path` specified. Such an Ingress rule has the same functionality as the Ingress default backend. Use the Ingress default backend instead.
  - The Ingress has JWL annotations that cannot be correctly processed. For example:
    - Either `ncp/jwt-alg` or `ncp/jwt-secret` is missing.
    - `ncp/jwt-alg` is configured with an unsupported algorithm.
    - `ncp/jwt-alg` and `ncp/jwt-secret` are configured with other HTTP enabling annotations, or with ssl passthrough.

LoadBalancer CRDs to Handle Ingress Scaling

You can create CRDs (CustomResourceDefinitions) to monitor the usage of NSX load balancers and to create additional NSX layer 7 load balancers to handle Ingress workloads that the default load balancer cannot handle. These CRDs are not for scaling layer 4 load balancers that are created for Kubernetes LoadBalancer services.

If you have a shared tier-1 topology, you must configure `tier0_gateway` under `[nsx_v3]` in the `nsx-ncp-config` ConfigMap to use this feature.

The CRDs are:

- `NSXLoadBalancerMonitor` - This CRD is used to report usage statistics of the NSX load balancers. In Policy mode, this CRD will only monitor namespace load balancers created using the LoadBalancer CRD.
- `LoadBalancer` - This CRD is used to create new NSX load balancers. The definition of this resource is in the NCP YAML file. In Policy mode and TKGI deployments, this is a namespace resource. In Manager mode deployments, this is a clusterwide resource.

The procedure to enable this feature is the same for Manager mode and Policy mode.

Perform the following steps to enable this feature:

- Set the `enable_lb_crd` option in the `[k8s]` section to `True`.
- Apply the NCP YAML file with the following command:

  `kubectl apply -f ncp-<platform>.yaml`

To create a new NSX load balancer, apply a YAML file that defines a LoadBalancer CRD. For example,

```
apiVersion: vmware.com/v1alpha1
kind: LoadBalancer
metadata:
    name: cluster1-lbs0
spec:
    httpConfig: {}
    size: SMALL
```

This YAML file will create an NSX load balancer of the specified size, and a pair of layer 7 virtual servers without persistence, SSL or X-forward settings. The `size` parameter can be `SMALL`, `MEDIUM`, or `LARGE`. The default value is `SMALL`. After the NSX load balancer is created, the size cannot be changed and any update to the `size` parameter will be ignored. The IP of the virtual server is allocated from the configured default external pool for load balancers. The ports by default are 80 and 443. Non-standard ports are supported if the custom port is included in the HTTP HOST header. Note that custom ports are only supported in non-TLS Ingresses. Also, the virtual servers created by the LoadBalancer CRD do not support the "enable access log" parameter.

To check the creation status of the LoadBalancer CRD, run the following command:

```
kubectl get lb <name of the LoadBalancer> -o yaml
```

The result looks something like the following:

```
status:
 conditions:
 - status: "True"
   type: Ready
 httpVirtualIP: <realized virtual IP>
```

This result indicates that the creation was successful. If the creation failed, `status` will be `False` and there will not be a virtual IP.

You can also customize settings for the NSX load balancer and virtual servers. To configure the IP and port for the virtual server:

```
spec:
    httpConfig:
        virtualIP: <ip address, default to auto-allocate>
        port: <port number, default to 80>
```

Note: You must not configure the same virtualIP for different LoadBalancer CRDs.

To specify the session affinity and X-forwarded-mode:

```
spec:
    httpConfig:
         xForwardedFor: <INSERT or REPLACE, default to None>
         affinity:
             type: <source_ip or cookie, default to None>
             timeout: <timeout number, default to 10800>
```

To configure TLS settings:

```
spec:
    httpConfig:
        tls:
            port: <tls port number, default to 443>
            secretName: <name of secret, default to None>
```

Note that even if you set the HTTP and HTTPS ports to non-default values, the Ingress printer will always display the default port values (80 and 443) when showing the Ingress status because of a Kubernetes limitation. You should still use the configured ports to access Ingress. For example,

```
curl -I -HHost:tea.example.com http://$INGRESS_IP:$CRD_LB_HTTP_PORT/tea
```

You can create the secret before or after the creation of LoadBalancer. To update the certificate, remove the `secretName` from the LoadBalancer spec first, update the data of the secret, then re-attach the same secret using the above configuration. Creating a new secret and updating the `secretName` will also work. Note that sharing the same secret data between different CRD load balancers is not supported. You must configure CRD load balancers with different certificates.

To view the status and statistics of NSX load balancers, run the following command:

```
kubectl get lbm
```

This will list all the NSXLoadBlancerMonitors, one for each NSX load balancer. The following information is displayed:

- Usage - The number of workloads on the NSX load balancer.
- Traffic - The aggregated statistics of each Virtual Server.
- Health - This field has two dimensions:
  - `servicePressureIndex` - This indicates the performance of the load balancer. Two values are provided: score and severity.
  - `infraPressureIndex` - This indicates the performance of the underlying infrastructure components. In NCP 2.5.1, this value is not always accurate.
  - The field `metrics` gives an idea of the parameters that are considered when the health score is calculated.

When the `servicePressureIndex` of a load balancer is `HIGH`, you can migrate the Ingress workload to other load balancers, which must be the default load balancer or load balancers created using the LoadBalancer CRD.

To place an Ingress on a dedicated load balancer, add an annotation to the Ingress specification. For example,

```
 annotations:
   nsx/loadbalancer: <name of the LoadBalancer CRD>
```

If the annotation is missing or set to `null`, the Ingress is placed on the default NSX load balancer.

Service of Type LoadBalancer

NCP will create a layer 4 load balancer virtual server and pool for each service port.

Details about this feature:

- Both TCP and UDP are supported.
- Each service will have a unique IP address.

- The service is allocated an IP address from an external IP pool based on the `loadBalancerIP` field in the LoadBalancer definition. The `loadBalancerIP` field can be empty, have an IP address or the name or ID of an IP pool. If the `loadBalancerIP` field is empty, the IP will be allocated from the external IP pool specified by the `external_ip_pools_lb` option in the `[nsx_v3]` section in `ncp.ini`. If the `external_ip_pools_lb` option does not exist, the pool specified by `external_ip_pools` is used. The LoadBalancer service is exposed on this IP address and the service port.
- You can change to a different IP pool by changing the configuration and restarting NCP.
- The IP pool specified by `loadBalancerIP` must have the tag `scope: ncp/owner, tag: cluster:<cluster_name>`.
- In Policy mode, a service of type LoadBalancer without a selector is supported. For such a service, the NSX load balancer's SNAT IP will be the IP of the service of type LoadBalancer. The NSX load balancer's SNAT IP will be updated if you update the IP of the service of type LoadBalancer. Note that a service without a selector cannot be configured as the endpoint of another service without a selector.
- Error are annotated to a service. The error key is `ncp/error.loadbalancer`. The possible errors are:
  - `ncp/error.loadbalancer`: IP_POOL_NOT_FOUND

    This error indicates that you specify `loadBalancerIP: <nsx-ip-pool>` but `<nsx-ip-pool>` does not exist. The service will be inactive. To fix the error, specify a valid IP pool, delete and recreate the service.
  - `ncp/error.loadbalancer`: IP_POOL_EXHAUSTED

    This error indicates that you specify `loadBalancerIP: <nsx-ip-pool>` but the IP pool has exhausted its IP addresses. The service will be inactive. To fix the error, specify an IP pool that has available IP addresses, delete and recreate the service.
  - `ncp/error.loadbalancer`: IP_POOL_NOT_UNIQUE

    This error indicates that multiple IP pools have the name that is specified by `loadBalancerIP: <nsx-ip-pool>`. The service will be inactive.
  - `ncp/error.loadbalancer`: POOL_ACCESS_DENIED

    This error indicates that the IP pool specified by `loadBalancerIP` does not have the tag `scope: ncp/owner, tag: cluster:<cluster_name>` or the cluster specified in the tag does not match the name of the Kubernetes cluster.
  - `ncp/error.loadbalancer`: LB_VIP_CONFLICT

    This error indicates that the IP in the `loadBalancerIP` field is the same as the IP of an active service. The service will be inactive.

- The layer 4 load balancer supports automatic scaling. If a Kubernetes LoadBalancer service is created or modified so that it requires additional virtual servers and the existing layer 4 load balancer does not have the capacity, a new layer 4 load balancer will be created. NCP will also delete a layer 4 load balancer that no longer has virtual servers attached. This feature is enabled by default. If you want to disable this feature, you must set `l4_lb_auto_scaling` to `false` in the NCP ConfigMap.
- When you create a service of type LoadBalancer, you can specify the parameter `loadBalancerSourceRanges`. For example:

  ```
  kind: Service
  metadata:
    name: nginx-service-lb
    labels:
      app: nginx
  spec:
    type: LoadBalancer
    loadBalancerSourceRanges:
      - "10.30.88.116/24"
    ports:
      - port: 80
        targetPort: 80
        protocol: TCP
        name: tcp
  ...
  ```

  The NSX load balancer that is created will only allow traffic from the specified source IP addresses. You can verify this in NSX Manager by looking at the virtual server's **Access List Control**. It will allow access to a specific group. This group will contain the IP addresses specified in `loadBalancerSourceRanges`. You can look at this group by navigating to **Inventory > Groups**.

Load Balancer and Network Policy

When traffic is forwarded to the pods from the NSX load balancer virtual server, the source IP is the tier-1 router's uplink port's IP address. This address is on the private tier-1 transit network, and can cause the CIDR-based network policies to disallow traffic that should be allowed.

To avoid this issue, the network policy must be configured such that the tier-1 router's uplink port's IP address is part of the allowed CIDR block. This internal IP address will be visible as an annotation (`ncp/internal_ip_for_policy`) on the Ingress and Service resources.

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

```
  status:
    loadBalancer:
    ingress:
    - ip: 4.4.0.5
```

The annotation on the Ingress and service of type LoadBalancer resource will be:

```
    ncp/internal_ip_for_policy: 100.64.224.11
```

The IP address 100.64.224.11 must belong to the allowed CIDR in the `ipBlock` selector of the network policy. For example,

```
    apiVersion: networking.k8s.io/v1
    kind: NetworkPolicy
    ...
    ingress:
    - from:
      - ipBlock:
         cidr: 100.64.224.11/32
```

Sample Script to Generate a CA-Signed Certificate

You can create a script to generate 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`. For example,

```
#!/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
```

Third-party Ingress Controllers

You can configure NCP to support third-party Ingress controllers.

Editing the `ncp.ini` file

You must edit the configuration file `/var/vcap/data/jobs/ncp/xxxxxxxx/config/ncp.ini` (where xxxxxxxx is the BOSH deployment ID). This file will then be copied to `rootfs` and used by NCP every time NCP restarts. The file must be edited on each master node.

Important Changes to `ncp.ini` are not persistent across TKGI cluster updates. If you make changes through the TKGI tile and then update the TKGI deployment, the changes to `ncp.ini` will be lost.

The relevant options are in the `nsx_v3` section.

- `use_native_loadbalancer` - If set to **`False`**, NCP will not process any Ingress or service of type Loadbalancer updates, regardless of its annotations. This setting applies to the whole TKGI cluster. The default is **`True`**.
- `default_ingress_class_nsx` - If set to **`True`**, NCP becomes the default Ingress controller and will handle both Ingresses annotated with `kubernetes.io/ingress.class: "nsx"` and Ingresses without any annotation. If set to **`False`**, NCP will only handle Ingresses annotated with `kubernetes.io/ingress.class: "nsx"`. The default is **`True`**.

  Starting with NCP 3.2.1, `default_ingress_class_nsx` is deprecated. NCP will look at the following when resolving the Ingress class:

  - annotations
  - `ingressClass` objects
  - If no annotation is specified and `use_native_loadbalancer` is True, then NSX-LB will handle the Ingress. Otherwise, NSX-LB will not handle it.

If you want NCP to assign a floating IP to the NGINX controller pod and update the status of Ingresses with the floating IP, do the following:

- In the `k8s` section in `ncp.ini`, set `ingress_mode=nat`.
- Add the annotation `ncp/ingress-controller: "True"` to the NGINX Ingress controller pod.

NCP will update the status of Ingresses that have the annotation `kubernetes.io/ingress.class: "nginx"` with the NGINX Ingress controller pod's floating IP. If `default_ingress_class_nsx=False`, NCP will also update the status of Ingresses without the `kubernetes.io/ingress.class` annotation with the NGINX Ingress controller pod's floating IP.

Note: Even if the NGINX Ingress controller pod does not have the annotation `ncp/ingress-controller: "True"`, NCP will update the status of the Ingresses mentioned above to `loadBalancer: {}`. The Ingresses could then be stuck in a loop where the NGINX controller updates the Ingress status to `loadBalancer: {ingress: [{ip: <IP>}]}` and NCP updates the Ingress status to `loadBalancer: {}`. To avoid this situation, perform the following steps:

- If the Ingress controller is from https://github.com/kubernetes/ingress-nginx,
  - On the Ingress controller, change the `ingress-class` to something other than `"nginx"`.
  - If there is an Ingress with the annotation `kubernetes.io/ingress-class: "nginx"`, change the annotation to a different value.
  - For more information, see https://kubernetes.github.io/ingress-nginx/user-guide/multiple-ingress.
- If the Ingress controller is from https://github.com/nginxinc/kubernetes-ingress,
  - On the Ingress controller, change the `ingress-class` to something other than `"nginx"`.

- If there is an Ingress with the annotation `kubernetes.io/ingress-class: "nginx"`, change the annotation to a different value.
- On the Ingress controller pod, set `use-ingress-class-only` to **`True`**. This will stop this controller from updating Ingresses without the `kubernetes.io/ingress-class` annotation.
- For more information, see https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/multiple-ingress.md.

For third-party Ingress controllers deployed in NAT mode, you can modify the `http_ingress_port` and `https_ingress_port` parameters in the `k8s` section to specify custom ports for the NAT rules exposed for the Ingress controller.

Scenario 1: NCP handles Ingresses but is not the default Ingress controller.

Follow this procedure to let NCP handle `nsx`-class Ingresses.

1 Edit the `nsx_v3` section in `ncp.ini` on each master node.

  a Set `default_ingress_class_nsx` to **`False`**.

  b Leave `use_native_loadbalancer` set to **`True`**, the default value.

2 Restart NCP on each master node. This might cause a master failover.

3 Annotate all the Ingresses that you want NCP to handle with `kubernetes.io/ingress.class: "nsx"`.

Scenario 2: NCP is the default Ingress controller.

Follow this procedure:

1 No need to edit `ncp.ini`, but ensure that every Ingress is annotated.

2 Ingresses to be handled by NCP should be annotated with **kubernetes.io/ingress.class: "nsx"**.

  Although NCP will handle Ingresses without the `kubernetes.io/ingress.class` annotation, in the case of multiple Ingress controllers, the best practice is to always have the `kubernetes.io/ingress.class` annotation and not to rely on the default Ingress controller behavior.

3 Ingresses to be handled by third-party Ingress controllers must be annotated with the value required by those Ingress controllers.

Important Unless the goal is to make NGINX the default Ingress controller, do not use **`nginx`** as the NGINX Ingress controller, because this will make NGINX the default Ingress controller.

Scenario 3: NCP does not handle any Ingress regardless of its annotation.

Follow this procedure:

1 Edit the `nsx_v3` section in `ncp.ini` on each master node.

    a Set `use_native_loadbalancer` to **`False`**. The value of `default_ingress_class_nsx` is now irrelevant.

2 Restart NCP on each master node. This might cause a master failover.

Note that NCP will also not handle services of type LoadBalancer

Upgrade NCP

6

NSX Container Plugin Operator is responsible for managing the life cycle of NCP.

Upgrade of NCP is done by updating the NCP Operator and NCP image. Before performing the upgrade, check the release notes for product compatibility.

Prerequisites

Download the latest nsx-container zip file (`nsx-container-x.x.x.y.zip`) from https://downloads.vmware.com.

The following tar files are needed for the upgrade:

- `nsx-container-x.x.x.y/Kubernetes/nsx-container-plugin-operator-x.x.x.y.tar`
- `nsx-container-x.x.x.y/Kubernetes/nsx-ncp-ubi-x.x.x.y.tar`

The two images must be uploaded to your container registry.

Procedure

1 Edit `operator.yaml`.

   Modify the nsx-ncp-container image:

   ```
   containers:
     - name: nsx-ncp-operator
       image: <URL to the NCP operator in your container registry>
   ```

   Modify the NCP_IMAGE URL:

   ```
     - name: nsx-ncp-operator
       image: <URL to the NCP operator in your container registry>
   ```

2 Update `configmap.yaml` with any new required fields for this release. The information about any new required attributes is available in the release notes.

3 Apply `role.yaml` with the following command.

   ```
   oc apply -f role.yaml -n nsx-system-operator
   ```

4 Apply `operator.yaml` (and `confimap.yaml` if changed) in the `nsx-system-operator` namespace with the following command.

```
oc apply -f operator.yaml -n nsx-system-operator
```

There is no need to edit the running config.

Importing Manager Objects to Policy

7

NSX has two methods of configuring networking: Manager mode and Policy mode. If you are upgrading NCP, you have the option of importing Manager objects to Policy. You can use the script `mp_to_policy_importer.py` that is provided for importing Manager objects to Policy.

This feature is supported if you have a Kubernetes environment but not a Tanzu Application Service (TAS) environment.

This chapter includes the following topics:

Importing Manager Objects to Policy Workflow

Importing a Kubernetes cluster can take some time and requires locking down the cluster (no create, update, or delete operations allowed). It is recommended that you import one cluster at a time so that you do not need to lock down all clusters.

Before starting the import process, you must perform a backup of NSX Manager. This ensures that if an unrecoverable failure occurs, you can restore NSX Manager to its state before the import.

The Python3 script that performs the import, `mp_to_policy_importer.py`, is located in the directory `scripts/mp_to_policy_import`. You must run it on the Kubernetes master node that is running the cluster.

For limitations and workarounds, see Limitations and Workarounds. For information on how to handle errors, see Failure and Recovery.

The workflow to import Manager objects to Policy:

1. Upgrade NSX to 3.2 or later.
2. Upgrade NCP to 4.0.
3. In the NCP YAML file, make sure that `policy_nsxapi` is set to `false`.
4. Start the migration coordinator on any of the NSX Managers with the following command:

   ```
   /etc/init.d/migration-coordinator start
   ```

   Verify that it is successfully running with the following command:

   ```
   /etc/init.d/migration-coordinator status
   ```

   Be sure to use this Manager's IP address when importing Manager objects to Policy.
5. Create a backup.
6. Import shared resources. See Importing Shared Resources.
7. Lock the Kubernetes cluster. The Kubernetes API server will be in read-only mode. Do not perform any create, update, or delete operations on Kubernetes resources. Wait at least 10 minutes before proceeding to the next step.
8. Stop NCP.
9. Create a backup.
10. Import the Kubernetes cluster. See Importing a Kubernetes Cluster.
11. Update `ncp.ini` if required for this cluster. See Importing Shared Resources.
12. In the NCP YAML file, make sure that `policy_nsxapi` is set to `true` and start NCP.
13. Unlock the cluster. You can now perform create, update, or delete operations on Kubernetes resources.
14. Repeat steps 4 - 12 for the next cluster.
15. Switch TKGI automation and BOSH CPI to use the Policy API once all clusters in the deployment are imported.

Importing Shared Resources

The first step in the Manager-to-Policy import process is to import shared resources such as routers, IP blocks and pools, NsGroups, and so on.

NCP in Policy mode only accepts the Policy ID of NSX resources in its ncp.ini. Shared Resources are imported to Policy from Manager with a policy ID that is derived from their display names in the following way:

- Each space (' ') literal is replaced with underscore ('_')

- Each forward slash ('/') is replaced with underscore ('_')
- If the display name has only periods (for example, '.', '.....', etc.) it is prepended with one underscore ('_')

Examples:

- "mp display name" becomes policy id: "mp_display_name"
- "mp display/name" becomes policy id: "mp_display_name"
- "....." becomes policy id: "_....."

You must ensure that all the NSX resources that you created have unique display names.

If needed, update the ncp.ini based on the above rules wherever the config reads the ID of the NSX resource.

Edit user-spec.yaml

You must edit user-spec.yaml to specify which resources will be imported. You can specify:

- The resource using either display_name or ID in the Manager API. If the resource is not found in Manager API, it is ignored.
- The IP-Allocations to be imported for any IP Pool in the user-spec.yaml under "ip-allocations". Two scenarios:

  a With custom IpPoolAllocations from this IpPool

     If you have created some IpAllocations manually, please specify them under here. Key is the allocation_id of the IpPoolAllocation and value is its expected policy ID. Do not import any other resource like IpBlock, Tier0, etc. with it. Once they are imported, run the script again to import shared_resources but as specified in step 2 below.

  b Without custom IpPoolAllocations from this IpPool (default)

     Do not edit/specify ip-allocations under any IpPool and add all the other resources like IpBlock, Tier0, etc in the spec to be imported
- static routes and router ports to be imported for a tier 1 router.

  Do not change the `key` and `value` identifier in the spec, but only their assigned values. `Key` is the manager ID and value is the expected policy ID.

Refer to Resource Specification Order to see how shared resources should be specified in `user-spec.yaml`.

Steps to import shared resources

1 Fill in the appropriate information in `config.yaml` and set `import_shared_resources_only` to `True`. See Sample config.yaml.

2 Fill in the shared resources info in `user-spec.yaml`. See Sample user-spec.yaml.

3 Run the `mp_to_policy_importer` using either the config file or command line arguments. For example:

```
python3 mp_to_policy_importer.py --config-file config.yaml
```

Importing a Kubernetes Cluster

After you import shared resources, you can import a Kubernetes cluster.

Edit user-spec.yaml

In `user-spec.yaml`, specify:

- The top tier router ID and type of the cluster
- Any custom resources that need to be imported as a part of any resource importation. For example, you can specify the manager ID of a NAT rule that should be imported as a part of namespace resources. See Custom Resources for more details. You do not need to do anything here unless you have manually created some resources on the resources created by NCP. For example. you added a static route on an NCP-created tier-1 router.
- Manager ID of the lb-service that is created by you as `lb-service-mp-id` to import the lb-service used by default in NCP if configured. This is the same resource as lb_service in the NCP spec (`ncp.ini`). If not used, you do not need to specify it.

Steps to import a Kubernetes cluster

1 Fill in the appropriate information in `config.yaml` and set `import_shared_resources_only` to `False`. See Sample config.yaml.

2 Fill in the Kubernetes cluster information in `user-spec.yaml`. See Sample user-spec.yaml.

3 Run the `mp_to_policy_importer` using either the config file or command line arguments. For example:

```
python3 mp_to_policy_importer.py --config-file config.yaml
```

Note that only the Kubernetes cluster specified in the config.yaml will be imported even if they are mentioned in the user-spec.yaml.

Understanding the Import Process

The import process consists of four phases.

Phase 1

Retrieve all the resources from Manager API. Filter the resources based on Kubernetes cluster tag (ncp/cluster) or shared resources specified in the user-spec.yaml. Start making request bodies to be sent to the migration server. If any request cannot be generated, NCP does not migrate the cluster and exit.

Expected failures and solution:

- Resources cannot be retrieved from Manager API because of connectivity issue

  Solution: Retry after fixing the connectivity issue
- Kubernetes does not contain a resource that is retrieved from Manager API

  Solution: Run NCP in Manager mode again until it achieves an idle state, meaning that is is not performing any CRUD (create, read, update, delete) operations. You should wait at least 10 minutes because that is the maximum time interval until NCP sends retry requests. If no errors in NCP logs, the issue should be fixed.

Phase 2

Start sending the importation requests created in Phase 1 to the migration API. Once a request is processed successfully, record the manager_ids contained in the request on the local disk of the client. If any request fails, rollback the already imported resources using their manager_ids stored on the local disk. If migration API tells it is a duplicate request, the importer will remove its manager_id from the request body and send the request again.

Expected failures and solution:

- Connectivity issue

  Solution: Retry after fixing the connectivity issue
- Migration API returns error

  Solution: Retry after some time as it can be Policy API that's at fault or migration API. If the issue persists, roll back all the imported resources if the importer stops unexpectedly using the `rollback_imported_resources` option in `config.yaml`. By default, the importer will roll back if any issue occurs in this phase. However if there is an issue during rollback, you have to manually try again. If the rollback using `mp_to_policy_importer` is unsuccessful, you must restore the NSX Manager from a backup to the state before importing the Kubernetes cluster.

Note: If DFW Sections and Rules have been imported and the importation requests fails for resources after, you must restore the state of the Manager using the created Backup before initiating the cluster importation again.

Phase 3

Infer the tags that are needed to be added/removed on the resources in Policy for all the imported resources. If any tag cannot be inferred (reason could be missing corresponding Kubernetes resource), the importer will rollback the already imported resources using their manager_ids stored on the local disk. This could happen when NCP in Manager mode was stopped in the middle of transaction. So, you should start NCP in Manager mode again and wait for a while.

Expected failures and solution:

- Kubernetes does not contain a resource that is retrieved from Manager API

Solution: After rollback, run NCP in Manager mode again until it achieves an idle state. You should wait at least 10 minutes because that is the maximum time interval until NCP sends retry requests. If no errors in NCP logs, the issue should be fixed.

Phase 4

This is the most crucial phase. It is highly recommended that no unexpected failure occurs in this phase. In this phase, the importer will update the resources on Policy with new tags and/or additional information (eg, the importer will update display_name of segments). If the resource cannot be updated at the time, the importer will store the updated policy resource body and policy resource URL on the local disk of the client and ask you to try again after fixing the issue (the issue is on the Policy API or a connectivity issue).

Expected failures and solution:

- Connectivity issue

  Solution: Retry after fixing the connectivity issue

In all the four phases, there is also risk of unexpected power failure and other issues which are handled as discussed in Failure and Recovery.

Failure and Recovery

It may happen that the import process is not able to finish importing because of an external issue such as a power failure, disk exhaustion, connectivity issue, and so on. In such scenarios, there are ways to recover.

If the failure occurrs in phase 1, 2, or 3, you can manually set the option `rollback_imported_resources` to `True` in `config.yaml` and run the script again. This will roll back all the imported resources. If you do not set the flag to `True`, NCP retries and continues the import of resources. By default, `rollback_imported_resources` is `False`.

If the failure occurs in phase 4, run `mp_to_policy_importer` again. NCP will attempt to push the cached updated resource bodies of Policy on the NSX Manager. If there was an error and information was not cached, you must roll back using the backup and restore feature to restore the cluster. This is because once the tags are updated on Policy and resources are rollbacked, then all the tags are lost.

Manager to Policy Import Rollback

The `mp_to_policy_importer` process stores the ID of resources on the local disk of the client if they are successfully imported. If a failure occurs while the importer is running, it will immediately roll back all the imported resources and delete them from the file system if the rollback is successful. However, if a failure occurs during rollback, you can set `rollback_imported_resources` to `True` in `config.yaml` to manually retry and rollback the imported resources by rerunning the mp_to_policy_importer. The importer will intimate through the logs if all the resources were successfully rollbacked. If the rollback through the script is unsuccessful, use the backup and restore feature to restore the cluster to its original state.

Custom Resources

You can import custom resources that are not imported by default when a cluster is imported.

The custom resources are not created by NCP. There are methods in the file `nsxt_mp_to_policy_converter.py` (search for the word "`custom`" in the method definitions) which can be overridden to implement behavior expected during the Manager to Policy import. By default NCP takes no actions for these resources and thus no custom behavior is implemented. If you specify any custom resource ID, a simple request to import it will be created (Policy ID is inferred from display_name in this case, no updates are performed in Phase 2 and 3). You can specify the IDs of these resources in the user spec file (see example below) using the `custom_resources` identifier. Below is the syntax:

```
k8s-clusters:
  <k8s cluster name>:
    <NSX Resource Shell Class Name>:
      <NSX Resource - 1 Class Name>:
        custom_resources:
          <NSX Resource - 1A ID>:
            metadata:
              - "key": <metadata key recognized by Migration API>
                "value": <metadata value associated with the key recognized by
                          Migration API>
            linked_ids:
              - "key": <linked_ids key recognized by Migration API>
                "value": <linked_ids value associated with the key recognized by
                          Migration API>
            <NSX Resource - 2 Class Name>: # these resources are imported after NSX Resource
- 1A is
              custom_resources:
                <NSX Resource - 2A ID>:
                  metadata:
                    - "key": <metadata key recognized by Migration API>
                      "value": <metadata value associated with the key
                                recognized by Migration API>
                  linked_ids:
                    - "key": <linked_ids key recognized by Migration API>
                      "value": <linked_ids value associated with the key
                                recognized by Migration API>      <NSX Resource - 1 Class
Name>:
          <NSX Resource - 1B ID>:
            metadata:
              - "key": <metadata key recognized by Migration API>
                "value": <metadata value associated with the key recognized by
                          Migration API>
            linked_ids:
              - "key": <linked_ids key recognized by Migration API>
                "value": <linked_ids value associated with the key recognized by
                          Migration API>
```

Example to specify custom resources in user-spec.yaml

```
k8s-clusters:
   k8scluster:
    # top-tier-router-id is required for each cluster
    top-tier-router-id: t1-router-id
    top-tier-router-type: TIER1
    # Provide custom resources as follow:
    NamespaceResources:
      Tier1Router:
        custom_resources:
          # Custom resources are specified only with MP ID
          6d93a932-87ea-42de-a30c-b39f397322b0:
            metadata:
              # It should be a list
              - key: 'metadata-key'
                value: 'metadata-value'
            linked_ids:
              # It should be a list
              - key: 'linked_id-key'
                value: 'linked_id-value'
      NATRules:
        custom_resources:
          # Custom resources are specified only with MP ID
          536870924:
            metadata:
              # It should be a list
              - key: 'metadata-key'
                value: 'metadata-value'
            linked_ids:
              # It should be a list
              - key: 'linked_id-key'
                value: 'linked_id-value'
```

Refer to Resource Specification Order to see how custom resources should be specified in `user-spec.yaml`.

Limitations and Workarounds

The import process has certain limitations. Some of them have workarounds.

- The Manager to Policy importer cannot roll back completely once the distributed firewall sections and rules are imported in phase 2.

  Workaround: You must use the Backup and Restore feature in this case to restore the cluster to its original configuration in Manager.

- In Policy mode, an NSX load balancer can support a maximum of 255 rules. If a cluster has an Ingress resource that has more than 255 rules, migrating the cluster from Manager mode to Policy mode will fail.

  Workaround: Create LoadBalancer CRDs to distribute the rules across multiple NSX load balancers.

Resource Specification Order

Below is the list of resource names for specifying shared and custom resources in `user-spec.yaml`.

```
SharedResources
- IpBlock
- IpPool
  - IpPoolAllocation (specified as ip-allocations; see the sample user-spec.yaml for an
example)
- Tier0Router
  - Tier0RouterPorts (cannot be specified in user-spec.yaml)
  - Tier0RouterConfig (cannot be specified in user-spec.yaml)
- Tier1Router
- Tier1RouterPortsAndStaticRoutes
- SpoofguradProfile
- NodeLogicalSwitch
- NsGroup
- IpSet
- FirewallSectionsAndRules
- Certificate

ClusterResources
- LogicalPort
- Tier1Router
- Tier1RouterPortsAndStaticRoutes

NamespaceResources
- IpBlockSubnet
  - IpPoolAllocation
- Tier1Router
  - Tier1RouterPorts
- NATRules
- SpoofguradProfile
- NodeLogicalSwitch
- NsGroup

PodResources
- LogicalPort
- NATRules

NetworkPolicyResources
- IpSet
- FirewallSectionsAndRules

SecretResources
```

```
- Certificate

LBClusterWideResources
- IpPool
  - IpPoolAllocation
- NodeLogicalSwitch
- Tier1Router
- Tier1RouterPortsAndStaticRoutes
- Certificate
- ServerPool
- CookiePersistenceProfile
- SourceIPPersistenceProfile
- ApplicationProfile
- VirtualServer

L4ServiceResources
- SourceIPPersistenceProfile
- VirtualServer
- LBService
```

Sample `config.yaml`

You can use the following sample `config.yaml` as a reference.

```
# NSX Manager IP address
mgr_ip: null

# NSX Manager username, ignored if nsx_api_cert_file is set
username: "username"

# NSX Manager password, ignored if nsx_api_cert_file is set
password: "password"

# Principal Identity used to import the NSX resource
principal_identity: "admin"

# Path to NSX client certificate file. If specified, the username and
# password options will be ignored. Must be specified along with
# nsx_api_private_key_file option
# nsx_api_cert_file: /root/nsx-ujo/sample_scripts/com.vmware.nsx.ncp/nsx.crt

# Path to NSX client private key file. If specified, the username and
# password options will be ignored. Must be specified along with
# nsx_api_cert_file option
# nsx_api_private_key_file: /root/nsx-ujo/sample_scripts/com.vmware.nsx.ncp/nsx.key

# Specify one or a list of CA bundle files to use in verifying the NSX
# Manager server certificate. This option is ignored if "allow_insecure"
# is set to True. If "allow_insecure" is set to False and ca_file
# is unset, the system root CAs will be used to verify the server
# certificate
# ca_file: None
# Import only the shared resources
# To import a cluster, this flag must be set to False
import_shared_resources_only: True

# If true, all the imported MP to Policy resources will be rollbacked
# Default=False
rollback_imported_resources: False

# Name of the Kubernetes clusters to be imported, optional.
# Can be specified to import only specific K8s clusters from the
# ones specified under user-spec.yaml
# If not specified, all the clusters will be imported
k8s_cluster_names:
  - k8scluster

# Name of the directory that records the IDs of resources
# that are successfully imported
# Default=successfull_records
# records_dir_name: successfull_records

# Path where the records of successfully imported resources are
# saved. You can use this in conjunction with records_dir_name
# By default, we use the current working directory
# records_dir_base_path: None

# YAML config file that stores the information of resources
# to be imported
user_spec_file: "/root/nsx-ujo/sample_scripts/mp_to_policy_import/user-spec.yaml"

# Disable urllib's insecure request warning
# Default=False
no_warning: False

# If true, the NSX Manager server certificate is not verified. If false the
# CA bundle specified via "ca_file" will be used or if unset the default
# system root CAs will be used
# Default=False
allow_insecure: False

# Enable the debug logs
# Default=False
enable_debug_logs: False

# Do not print the logs on the console.
# Can be used with logging enabled to a file.
# Default=False
# disable_console_logs: False

# Output the logs to this file.
# Default=/var/log/nsx-ujo/mp_to_policy_import.log
# log_file: /var/log/nsx-ujo/mp_to_policy_import.log
```

Sample `user-spec.yaml`

You can use the following sample `user-spec.yaml` as a reference.

```
SharedResources:
  # IMPORTANT: If there are multiple resources with the same
  # display_name, please make the display_names unique before
  # initiating the import
  IpPool:
    resources:
      # Specify the resources to import here with their UUID or
      # display_name
      k8s-snat-pool:
        # Duplicate MP to Policy imports allowed for ip-allocations
        # ip-allocations:
        #   - key: "172.24.4.4"
        #     value: "ip-alloc-1"
      k8s-lb-pool:
  IpBlock:
    resources:
      k8s-container-block:
  IpSet:
    resources:
      vs-ipset-1:
  Tier0Router:
    resources:
      node-t0:
  Tier1Router:
    resources:
      # NOTE: If a Tier1 router is a top-tier router, it should not be imported
      #       as a shared resource
      node-lr:
        is-any-cluster-top-tier: False  # required attribute
  Tier1RouterPortsAndStaticRoutes:
    resources:
      # NOTE: If a Tier1 router is a top-tier router for any cluster,
      #       it should not be imported as a shared resource
      node-lr:
        is-any-cluster-top-tier: False  # required attribute
        # static_routes_import_info:
        #   - key: "a4b04674-feb9-4418-8d5f-ac8bca4665eb"
        #     value: st-r-1
        # router_ports_import_info:
        #   - key: "s2f24456-feb9-4418-8d5f-ar8aqa4665vf"
        #     value: rp-1
  NsGroup:
    # resources:
    #   test-ns-group:
    #     domain: my-domain
  SpoofguradProfile:
    resources:
      nsx-default-spoof-guard-vif-profile:
  NodeLogicalSwitch:
    resources:
      node-ls:
  FirewallSectionsAndRules:
    resources:
      # Make sure to write the DFW Section and Rules in the order
      # in which they should be imported. Otherwise there might be a
      # moment in which the section that is present at lower priority
      # is imported before section that is present above it making the
      # traffic flow inconsistent. The best way to do it is to mention
      # the Sections and Rules with increasing |priority| number. Note
      # that lower priority numbers are present at the top in the Section
      # and Rules order in NSX
      fw-section-1:
        section_info:
          # category is a Security Policy attribute
          category: "Application"
          # You can specify either priority or bool is_top_section.
          # If is_top_section is True, priority is auto-assigned to 5
          # If is_top_section is False, priority is auto-assigned to 95
          # If is_top_section and priority are specified, priority is used
          # If both are not specified, error is thrown
          is_top_section: True
          # priority: 1
          # domain is a Security Policy attribute
          domain: "my-domain"
        rules_info:
          - name: "rule-name"  # name or id must be specified
            priority: 1  # optional. If not specified, FW Rule priority will be
                         # used as sequence number of Policy Rule
          - id: 'rule-id'  # name or id must be specified
  Certificate:
    resources:
      my-cert:
k8s-clusters:
  k8scluster:
    # top-tier-router-id (MP) is required for each cluster
    top-tier-router-id: null
    # top-tier-router-type is required for each cluster
    # choices: TIER0 or TIER1
    top-tier-router-type: TIER0
    # lb-service-mp-id is the same as lb_service in ncp.ini config file
    lb-service-mp-id: null  # optional
    external-ip-pools-lb-mp-id: []  # required. leave empty, [], if not used
    external-ip-pools-mp-id: []  # required. leave empty, [], if not used
    http-and-https-ingress-ip: null
    # NamespaceResources:
    #   Tier1Router:
    #     custom_resources:
    #       6d93a932-87ea-42de-a30c-b39f397322b0:
  k8scluster-2:
    # top-tier-router-id (MP) is required for each cluster
    top-tier-router-id: null
    top-tier-router-type: TIER1
    # lb-service-mp-id is the same as lb_service in ncp.ini config file
    lb-service-mp-id: null  # optional
    external-ip-pools-lb-mp-id: []  # required. leave empty, [], if not used
    external-ip-pools-mp-id: []  # required. leave empty, [], if not used
    http-and-https-ingress-ip: null
    # Provide custom resources as follow:
    NamespaceResources:
      Tier1Router:
        custom_resources:
          # Custom resources are specified only with MP ID
          6d93a932-87ea-42de-a30c-b39f397322b0:
            metadata:
              # It should be a list
              - key: 'metadata-key'
                value: 'metadata-value'
            linked_ids:
              # It should be a list
              - key: 'linked_id-key'
                value: 'linked_id-value'
```

Administering NCP

8

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

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

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

If the nsx-ovs container running in the nsx-node-agent pod is restarted for any reason, Kubernetes services might become unavailable for 2 minutes or more. This is expected behavior.

Do not recreate a pod with the same pod name if the NCP pod is stopped and the node agent pod is running. The new pod will have the wrong network configuration and its network traffic will fail.

This chapter includes the following topics:

Displaying Error Information Stored in the Kubernetes Resource NSXError

For each Kubernetes resource object that has NSX backend failures, one NSXError object is created with error information. There is also an error object for all cluster-wide errors.

This feature is not enabled by default. To enable it, you must set `enable_nsx_err_crd` to True in `ncp.ini` when you install NCP.

Note You must not create, update, or delete NSXError objects.

If you start NCP in policy mode (with the option `policy_nsxapi=true` in the NCP YAML File), the NSXError resource is not supported.

Commands to display NSXError objects:

- `kubectl get nsxerrors`

  List all NSXError objects.

- `kubectl get nsxerrors -l error-object-type=<type of resource>`

  List NSXError objects related to a specific type of Kubernetes objects, for example, objects of type `services`.

- `kubectl get nsxerrors <nsxerror name> -o yaml`

  Display the details of an NSXError object.

- `kubectl get svc <service name> -o yaml | grep nsxerror`

  Find the NSXError associated with a specific service.

When you display the details of an NSXError object, the spec section contains the following important information. For example,

```
error-object-id: default.svc-1
error-object-name: svc-1
error-object-ns: default
error-object-type: services
message:
- '[2019-01-21 20:25:36]23705: Number of pool members requested exceed LoadBalancerlimit'
```

In this example, the namespace is `default`. The name of the service is `svc-1`. The type of kubernetes resource is `services`.

In this release, the following errors are supported by the NSXError object.

- Automatic scaling failed to allocate additional load balancers due to an NSX Edge limit.
- The number of load balancer virtual servers exceeds the limit (automatic scaling is not enabled).
- The number of load balancer server pools exceeds the limit.
- The number of load balancer server pool members exceeds the load balancer limit or the NSX Edge limit.
- Floating IP addresses exhausted when processing a LoadBalancer type service.

Changing the NSX Node Agent Configuration after Installation

You can change the NSX node agent configuration after NCP is installed.

Run the following command and modify the appropriate parameters:

```
    kubectl edit cm nsx-node-agent-config -n nsx-system
```

After you make the changes, reboot the nsx-ncp-bootstrap pods and the nsx-node-agent pods.

Cleaning Up the NSX Environment

If necessary, you can run a script to remove all NSX objects created by NCP.

The installation files include the following cleanup scripts:

- `nsx_policy_cleanup.py` - Use this script if the NSX resources were created in Policy mode.
- `nsx_cleanup.py` - Use this script if the NSX resources were created in Manager mode.

Before running the script, perform the following tasks:

- Stop NCP.
- Remove all resources that you created and are associated with the NCP-created objects. The script will fail if you do not delete those objects. For example, if NCP created a segment, and you created a distributed firewall (DFW) rule and group associated with the segment, you must delete the DFW rule and group, or remove the associations. Or if you attached VMs to the segment, you must delete the VMs or detach them from the segment.

Policy Mode

```
Usage: nsx_policy_cleanup.py [options]

Options:
  -h, --help            show this help message and exit
  --mgr-ip=MGR_IP       NSX Manager IP address
  -u USERNAME, --username=USERNAME
                        NSX Manager username, ignored if nsx-cert is set
  -p PASSWORD, --password=PASSWORD
                        NSX Manager password, ignored if nsx-cert is set
  -n NSX_CERT, --nsx-cert=NSX_CERT
                        NSX certificate path
  -k KEY, --key=KEY     NSX client private key path
  --vc-endpoint=VC_ENDPOINT
                        IpAddress or Hostname of VC, ignored if environment
                        variable VC_ENDPOINT is set
  --vc-username=VC_USERNAME
                        Username for the VC ServiceAccount, ignored if
                        environment variable VC_USERNAME is set
  --vc-password=VC_PASSWORD
                        Password for the VC ServiceAccount, ignored if
```

```
                         environment variable VC_PASSWORD is set
  --vc-https-port=VC_HTTPS_PORT
                         HTTPS port of VC, ignored if environment variable
                         VC_HTTPS_PORT is set. If not present, 443 default
                         value will be used
  --vc-sso-domain=VC_SSO_DOMAIN
                         SSO Domain of VC, ignored if environment variable
                         VC_SSO_DOMAIN is set. If not present, local default
                         value will be used
  --vc-ca-cert=VC_CA_CERT
                         Specify a CA bundle to verify the VC server
                         certificate. It will be ignored if environment
                         VC_CA_CERT is set
  --vc-insecure         Not verify VC server certificate
  -c CLUSTER, --cluster=CLUSTER
                         Cluster to be removed
  -r, --remove           CAVEAT: Removes NSX resources. If not set will do dry-
                         run.
  --top-tier-router-id=TOP_TIER_ROUTER_ID
                         Specify the top tier router id. Must be specified if
                         top tier router does not have the cluster tag
  --all-res              Also clean up HA switching profile, ipblock, external
                         ippool. These resources could be created by TAS NSX-T
                         Tile
  --no-warning           Disable urllib's insecure request warning
  --status               Check the deletion status, the exit code can be
                         success(0), in progress(EXIT_CODE_IN_PROGRESS or
                         failure(other non-zerovalues)
  --thumbprint=THUMBPRINT
                         Specify one or a list of thumbprint strings to use in
                         verifying the NSX Manager server certificate
```

For example:

```
python nsx_policy_cleanup.py --mgr-ip={nsx_mngr_ip} -u admin -p {password} -c
{k8s_cluster_name} --no-warning -r
```

In some cases, the `top-tier-router-id` parameter must be be specified.

Manager Mode

```
Usage: nsx_cleanup.py [options]

Options:
  -h, --help            show this help message and exit
  --mgr-ip=MGR_IP       NSX Manager IP address
  -u USERNAME, --username=USERNAME
                         NSX Manager username, ignored if nsx-cert is set
  -p PASSWORD, --password=PASSWORD
                         NSX Manager password, ignored if nsx-cert is set
  -n NSX_CERT, --nsx-cert=NSX_CERT
                         NSX certificate path
  -k KEY, --key=KEY     NSX client private key path
  -c CLUSTER, --cluster=CLUSTER
```

```
                         Cluster to be removed
  -r, --remove           CAVEAT: Removes NSX resources. If not set will do dry-
                         run.
  --top-tier-router-uuid=TOP_TIER_ROUTER_UUID
                         Specify the top tier router uuid. Must be specified if
                         top tier router does not have the cluster tag or for a
                         single-tier1 topology
  --all-res              Also clean up HA switching profile, ipblock, external
                         ippool. These resources could be created by TAS NSX-T
                         Tile
  --no-warning           Disable urllib's insecure request warning
```

For example:

```
python nsx_cleanup.py --mgr-ip={nsx_mngr_ip} -u admin -p {password} -c {k8s_cluster_name} --
top-tier-router-uuid={top_tier_router_uuid} --no-warning -r
```

Changing the Cluster Name of a Running Cluster

Changing the name of a running cluster is not recommended. If you must do it, use the following procedure.

1 Stop NCP.

2 Download the cleanup script to delete NSX resources.

  For resources created using Manager mode, download `nsx_cleanup.py`. For resources created using Policy mode, download `nsx_policy_cleanup.py`.

3 Run the cleanup script.

4 Start NCP with the new cluster name.

5 Re-create the pods.

CLI Commands

To run CLI commands, log in to the NSX Container Plugin 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 8-1. CLI Commands for the NCP Container

Type	Command	Note
Status	get ncp-master status	For both Kubernetes and TAS.
Status	get ncp-nsx status	For both Kubernetes and TAS.
Status	get ncp-watcher <watcher-name>	For both Kubernetes and TAS.

Table 8-1. CLI Commands for the NCP Container (continued)

Type	Command	Note
Status	get ncp-watchers	For both Kubernetes and TAS.
Status	get ncp-k8s-api-server status	For Kubernetes only.
Status	check projects	For Kubernetes only.
Status	check project <project-name>	For Kubernetes only.
Status	get ncp-bbs status	For TAS only.
Status	get ncp-capi status	For TAS only.
Status	get ncp-policy-server status	For TAS only.
Cache	get project-caches	For Kubernetes only.
Cache	get project-cache <project-name>	For Kubernetes only.
Cache	get namespace-caches	For Kubernetes only.
Cache	get namespace-cache <namespace-name>	For Kubernetes only.
Cache	get pod-caches	For Kubernetes only.
Cache	get pod-cache <pod-name>	For Kubernetes only.
Cache	get ingress-caches	For Kubernetes only.
Cache	get ingress-cache <ingress-name>	For Kubernetes only.
Cache	get ingress-controllers	For Kubernetes only.
Cache	get ingress-controller <ingress-controller-name>	For Kubernetes only.
Cache	get network-policy-caches	For Kubernetes only.
Cache	get network-policy-cache <pod-name>	For Kubernetes only.
Cache	get asg-caches	For TAS only.
Cache	get asg-cache <asg-ID>	For TAS only.
Cache	get org-caches	For TAS only.
Cache	get org-cache <org-ID>	For TAS only.
Cache	get space-caches	For TAS only.
Cache	get space-cache <space-ID>	For TAS only.
Cache	get app-caches	For TAS only.
Cache	get app-cache <app-ID>	For TAS only.
Cache	get instance-caches <app-ID>	For TAS only.

Table 8-1. CLI Commands for the NCP Container (continued)

Type	Command	Note
Cache	get instance-cache <app-ID> <instance-ID>	For TAS only.
Cache	get policy-caches	For TAS only.
Support	get ncp-log file <filename>	For both Kubernetes and TAS.
Support	get ncp-log-level [component]	For both Kubernetes and TAS.
Support	set ncp-log-level <log-level> [component]	For both Kubernetes and TAS.
Support	get support-bundle file <filename>	For Kubernetes only.
Support	get node-agent-log file <filename>	For Kubernetes only.
Support	get node-agent-log file <filename> <node-name>	For Kubernetes only.

Table 8-2. 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 8-3. CLI Commands for the NSX Kube Proxy Container

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

Status Commands for the NCP Container

- Show the status of the NCP master

  ```
  get ncp-master status
  ```

  Example:

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

- Show the connection status between NCP and NSX Manager

  ```
  get ncp-nsx status
  ```

  Example:

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

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

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

  Example:

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

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

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

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

```
        Watcher thread created time: Mar 02 2017 10:51:37 PST
        Watcher thread status: Up


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

- Show the connection status between NCP and Kubernetes API server

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

  Example:

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

- Check all projects or a specific one

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

  Example:

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

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

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

- Check connection status between NCP and TAS BBS

  ```
  get ncp-bbs status
  ```

  Example:

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

- Check connection status between NCP and TAS CAPI

  ```
  get ncp-capi status
  ```

  Example:

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

- Check connection status between NCP and TAS policy server

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

  Example:

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

Cache Commands for the NCP Container

- Get the internal cache for projects or namespaces

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

  Example:

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

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

      testns:
          ext_pool_id: 346a0f36-7b5a-4ecc-ad32-338dcb92316f
          labels:
              ns: myns
              project: myproject
          logical-router: 4dc8f8a9-69b4-4ff7-8fb7-d2625dc77efa
          logical-switch:
  ```

```
            id: 6111a99a-6e06-4faa-a131-649f10f7c815
            ip_pool_id: 51ca058d-c3dc-41fd-8f2d-e69006ab1b3d
            subnet: 50.0.2.0/24
            subnet_id: 34f79811-bd29-4048-a67d-67ceac97eb98
        project_nsgroup: 9606afee-6348-4780-9dbe-91abfd23e475
        snat_ip: 4.4.0.3


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


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

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

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


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

- Get the internal cache for pods

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

  Example:

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

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


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

- Get all Ingress caches or a specific one

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

Example:

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


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

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

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

Example:

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

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


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

- Get network policy caches or a specific one

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

Example:

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

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

- Get all ASG caches or a specific one

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

  Example:

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

```
            75bc164d-1214-46f9-80bb-456a8fbccbfd
        staging_default: False
        staging_spaces:


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

- Get all org caches or a specific one

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

Example:

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


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

- Get all space caches or a specific one

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

  Example:

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

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


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

- Get all app caches or a specific one

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

  Example:

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

```
              index: 0
              ip: 192.168.5.4/24
              last_updated_time: 1522965828.45
              mac: 02:50:56:00:60:02
              port_id: a7c6f6bb-c472-4239-a030-bce615d5063e
              state: RUNNING
              vlan: 3
          name: hello2
          rg_id: a8423bc0-4b2b-49fb-bbff-a4badf21eb09
          space_id: 7870d134-7997-4373-b665-b6a910413c47


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

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

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

Example:

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

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

- Get all policy caches

  ```
  get policy-caches
  ```

  Example:

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

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

Support Commands for the NCP Container

- Save the NCP support bundle in the filestore

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

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

Example:

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

- Save the NCP logs in the filestore

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

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

  Example:

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

- Save the node agent logs in the filestore

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

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

  Example:

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

- Get and set the log level globally or for a specific component.

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

  The available components are `nsx_ujo.ncp`, `nsx_ujo.ncp.k8s`, `nsx_ujo.ncp.pcf`, `vmware_nsxlib.v3`, `nsxrpc`, and `nsx_ujo.ncp.nsx`.

  ```
  get ncp-log-level [component]
  set ncp-log-level <log level> [component]
  ```

  Examples:

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

  kubenode> get ncp-log-level nsx_ujo.ncp
  nsx_ujo.ncp log level is INFO

  kubenode>set ncp-log-level DEBUG
  ```

```
NCP log level is changed to DEBUG

kubenode> set ncp-log-level DEBUG nsx_ujo.ncp
nsx_ujo.ncp log level has been 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:

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


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

Status Commands for the NSX Kube-Proxy Container

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

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

  Example:

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

- Show the Kube Proxy watcher status

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

  Example:

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

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


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

- Dump OVS flows on a node

  ```
  dump ovs-flows
  ```

  Example:

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

```
priority=100,ip,nw_dst=10.96.0.10 actions=drop
    cookie=0x0, duration=8.819s, table=1, n_packets=0, n_bytes=0, idle_age=8,
priority=90,ip,in_port=1 actions=ct(table=2,nat)
    cookie=0x0, duration=8.799s, table=1, n_packets=0, n_bytes=0, idle_age=8,
priority=80,ip actions=NORMAL
    cookie=0x0, duration=8.856s, table=2, n_packets=0, n_bytes=0, idle_age=8,
actions=NORMAL
```

Error Codes

This section lists error codes produced by the various components.

NCP Error Codes

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

Error Code	Description
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

Error Code	Description
NCP00048	Load balancer service deletion failed
NCP00049	Load balancer virtual server creation failed
NCP00050	Load balancer virtual server update failed
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

Error Code	Description
NCP00074	Logical router port get failed
NCP00075	NSX configuration validation failed

Error Code	Description
NCP00076	SNAT rule update failed
NCP00077	SNAT rule overlapped
NCP00078	Load balancer endpoints add failed
NCP00079	Load balancer endpoints update failed
NCP00080	Load balancer rule pool creation failed
NCP00081	Load balancer virtual server not found
NCP00082	IP set read failed
NCP00083	SNAT pool get failed
NCP00084	Load balancer service creation failed
NCP00085	Load balancer service update failed
NCP00086	Logical router port update failed
NCP00087	Load balancer init failed
NCP00088	IP pool not unique
NCP00089	Layer 7 load balancer cache sync error
NCP00090	Load balancer pool does not exist
NCP00091	Load balancer rule cache init error
NCP00092	SNAT process failed
NCP00093	Load balancer default certificate error
NCP00094	Load balancer endpoint deletion failed
NCP00095	Project not found
NCP00096	Pool access denied
NCP00097	Failed to get a load balancer service
NCP00098	Failed to create a load balancer service
NCP00099	Load balancer pool cache synchronization error

NSX Node Agent Error Codes

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

nsx-kube-proxy Error Codes

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

CLI Error Codes

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

Error Code	Description
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
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

Error Code	Description
NCP05022	Kubernetes namespace cache not found
NCP05023	Kubernetes secret not found
NCP05024	Kubernetes default backend is in use
NCP05025	Kubernetes LoadBalancer service handle failed

Tanzu Application Service Error Codes

Error Code	Description
NCP06001	TAS BBS connection failed
NCP06002	TAS CAPI connection failed
NCP06006	TAS cache not found
NCP06007	TAS unknown domain
NCP06020	TAS policy server connection failed
NCP06021	TAS policy processing failed
NCP06030	TAS event processing failed
NCP06031	TAS unexpected event type
NCP06032	TAS unexpected event instance
NCP06033	TAS task deletion failed
NCP06034	TAS file access failed

9 Switching Between NSX-OVS and Upstream OVS Kernel Modules

Since NSX-OVS is not supported in the latest kernel version, you can switch the NSX-OVS kernel module to the upstream OVS kernel module before upgrading the kernel to the latest version. If NCP does not work with the latest kernel after a kernel upgrade, you can do a rollback (switch back to NSX-OVS and downgrade the kernel).

The first procedure below describes how to switch the NSX-OVS kernel module to the upstream OVS kernel module when you upgrade the kernel. The second procedure describes how to switch back to the NSX-OVS kernel module when you downgrade the kernel.

Both procedures involve the Kubernetes concepts `taints` and `tolerations`. For more information about these concepts, see https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration.

Switch to the upstream OVS kernel module

1 Modify the `tolerations` of both `daemonset.apps/nsx-ncp-bootstrap` and `daemonset.apps/nsx-node-agent`. Change the following:

```
      - effect: NoExecute
        operator: Exists
```

to:

```
      - effect: NoExecute
        key: evict-user-pods
```

2 Modify the nsx-node-agent configmap. Change `use_nsx_ovs_kernel_module` to `False`.

3 `Taint` worker-node1 "evict-user-pods:NoExecute" to evict all user pods in this node to other nodes:

```
kubectl taint nodes worker-node1 evict-user-pods:NoExecute
```

4 `Taint` worker-node1 "evict-ncp-pods:NoExecute" to evict nsx-node-agent and nsx-ncp-bootstrap pods in this node to other nodes:

```
kubectl taint nodes worker-node1 evict-ncp-pods:NoExecute
```

5 Uninstall the ovs-kernel module and restore the upstream OVS kernel module on worker-node1.

   a Delete kmod files `vport-geneve.ko`, `vport-gre.ko`, `vport-lisp.ko`, `vport-stt.ko`, `vport-vxlan.ko`, `openvswitch.ko` in directory `/lib/modules/$(uname -r)/weak-updates/openvswitch`.

   b If there are `vport-geneve.ko`, `vport-gre.ko`, `vport-lisp.ko`, `vport-stt.ko`, `vport-vxlan.ko`, `openvswitch.ko` files in directory `/lib/modules/$(uname -r)/nsx/usr-ovs-kmod-backup`, move them to directory `/lib/modules/$(uname -r)/weak-updates/openvswitch`.

   c Delete directory `/lib/modules/$(uname -r)/nsx`.

6 Upgrade the kernel of worker-node1 to the latest version and reboot it.

   Note: Set SELinux to Permissive mode on worker-node1 if containerd and kubelet cannot be running.

7 Restart kubelet.

8 Remove `taint` "evict-ncp-pods:NoExecute" from worker-node1. Verify that bootstrap and node-agent can start.

9 Remove `taint` "evict-user-pods:NoExecute" from worker-node1. Verify that all pods in this node are running.

10 Repeat steps 3-9 for other nodes.

11 Recover the `tolerations` of both nsx-ncp-bootstrap and nsx-node-agent DaemonSets in step 1.

Switch back to the NSX-OVS kernel module

1 Modify the `tolerations` of both `daemonset.apps/nsx-ncp-bootstrap` and `daemonset.apps/nsx-node-agent`. Change the following:

```
      - effect: NoExecute
        operator: Exists
```

   to:

```
      - effect: NoExecute
        key: evict-user-pods
```

2 Modify the nsx-node-agent configmap. Change `use_nsx_ovs_kernel_module` to `True`.

3 `Taint` worker-node1 "evict-user-pods:NoExecute" to evict all user pods in this node to other nodes:

```
kubectl taint nodes worker-node1 evict-user-pods:NoExecute
```

4. `Taint` worker-node1 "evict-ncp-pods:NoExecute" to evict nsx-node-agent and nsx-ncp-bootstrap pods in this node to other nodes:

   ```
   kubectl taint nodes worker-node1 evict-ncp-pods:NoExecute
   ```

5. Downgrade the kernel of worker-node1 to a supported version and reboot it.

   Note: Set SELinux to Permissive mode on worker-node1 if containerd and kubelet cannot be running.

6. Restart kubelet.
7. Remove `taint` "evict-ncp-pods:NoExecute" from worker-node1. Verify that bootstrap and node-agent can start.
8. Remove `taint` "evict-user-pods:NoExecute" from worker-node1. Verify that all pods in this node are running.
9. Repeat steps 3-8 for other nodes.
10. Recover the `tolerations` of both nsx-ncp-bootstrap and nsx-node-agent DaemonSets in step 1.