Contents

[Overview 1](#_Toc452041642)

[Neutron 1](#_Toc452041643)

[Neutron ML2 1](#_Toc452041644)

[What ML2 drivers are 1](#_Toc452041645)

[H3C proprietary ML2 drivers 1](#_Toc452041646)

[Preparing for installation 2](#_Toc452041647)

[Hardware requirements 2](#_Toc452041648)

[Software requirements 2](#_Toc452041649)

[LLDP configuration 2](#_Toc452041650)

[Configuring LLDP 2](#_Toc452041651)

[Verifying the LLDP configuration 3](#_Toc452041652)

[Installing or uninstalling the H3C ML2 drivers 4](#_Toc452041653)

[Installing the H3C ML2 drivers 4](#_Toc452041654)

[Uninstalling the H3C ML2 drivers 4](#_Toc452041655)

[Configuring the networking server component 5](#_Toc452041656)

[Tuning SQL connection performance parameters 5](#_Toc452041657)

[Configuring the tenant network as a VLAN network 5](#_Toc452041658)

[Configuring the tenant network as a VXLAN network 6](#_Toc452041659)

[Deploying L2 and L3 agents 6](#_Toc452041660)

[Configuring H3C databases 7](#_Toc452041661)

[Restarting the neutron server 7](#_Toc452041662)

[Troubleshooting 8](#_Toc452041663)

[Layer 3 configuration deployment failure 8](#_Toc452041664)

[L3 agent failure 8](#_Toc452041665)

[Slow deployment of Layer 3 configurations 8](#_Toc452041666)

# Overview

This document describes the installation procedure for H3C ML2 drivers.

## Neutron

Neutron manages the virtual networking infrastructure (VNI) in the OpenStack networking platform. It provides virtual network services for OpenStack computing and management devices. It manages various virtual network resources, including networks, subnets, DHCP, virtual routers, firewalls, and LB devices.

## Neutron ML2

Neutron ML2 is a framework that allows OpenStack networking to use varieties of existing Layer 2 networking technologies available in data centers. This framework is also designed to minimize the effort required to add support for new L2 networking technologies.

### What ML2 drivers are

ML2 drivers include type drivers and mechanism drivers.

* Each type driver manages a network type. A type driver maintains type-specific network state, and performs provider network validation and tenant network allocation. As of the date of this document, the ML2 plugin includes type drivers for the local, flat, VLAN, GRE and VXLAN network types.
* Each mechanism driver manages a networking mechanism. A mechanism driver provides the information established by the type driver to the networking mechanism when it is enabled. As of the date of this document, the mechanism driver interface supports the creation, update, and deletion of network and port resources.

### H3C proprietary ML2 drivers

H3C proprietary ML2 drivers include h3c\_vxlan (type driver) and h3c (mechanism driver).

In VXLAN networking, h3c\_vxlan uses two tiers of virtual network identifiers: Inner VLAN ID and outer VNI (also called VXLAN ID in documentation). The inner VLAN ID is used by OpenvSwitch to encapsulate and decapsulate VLAN packets. The outer VNI is sent in AMQP messages to the network devices for VXLAN packets encapsulation and decapsulation.

# Preparing for installation

## Hardware requirements

shows the hardware requirements for a server or VM to host the H3C ML2 drivers.

* + - * 1. Hardware requirements

|  |  |  |
| --- | --- | --- |
| CPU | Memory size | Disk space |
| One or more cores | 2 GB or above | 5 GB or above |

## Software requirements

shows the software requirements for a server or VM to host the H3C ML2 drivers.

* + - * 1. Software requirements

|  |  |
| --- | --- |
| Item | Requirements |
| OpenStack | OpenStack version:   * + - * Juno 2014.2       * Kilo 2015.1   Operating system version:   * + - * Ubuntu 14.04 LTS       * CentOS 7 |
| LLDP | lldpad software |

## LLDP configuration

Enable LLDP on the network devices and servers for topology data collection. These devices sent their topology data to the controller to form the topology of the entire network. The controller uses the topology data to perform automated VLAN and VXLAN provisioning and configuration.

### Configuring LLDP

LLDP configuration procedure differs depending on whether the leaf devices are directly connected to the server.

#### Configuring LLDP when leaf devices are directly connected to the server

Enable LLDP on the leaf device.

<H3C> system

[H3C] lldp global enable

[H3C] vcf-fabric topology enable

Enable LLDP on the server.

Install and enable lldpad.

Ubuntu:

# apt-get install lldpad

# service lldpad start

CentOS:

# yum install lldpad

# systemctl start lldpad

Configure the TLVs to be carried in LLDP messages on the physical interface connected to the leaf device. This example uses interface em1.

# lldptool –i em1 –L adminstatus=rxtx

# lldptool -T -i em1 -V sysName enableTx=yes

# lldptool -T -i em1 -V portDesc enableTx=yes

#### Configuring LLDP when leaf devices are indirectly connected to the server

On each intermediate device, configure LLDP to operate in service bridge mode. This example uses H3C devices.

[H3C] lldp global enable

[H3C] lldp mode service-bridge

On each leaf device, enable LLDP globally and enable customer bridge mode on the interface that is connected to the intermediate device.

<H3C> system

[H3C] lldp global enable

[H3C] vcf-fabric topology enable

[H3C] interface GigabitEthernet 1/0/1

[H3C]-GigabitEthernet1/0/1]lldp agent nearest-customer admin-status txrx

On the server, enable LLDP and configure the customer bridge mode on the interface that is connected to the intermediate device.

Install and enable lldpad.

Ubuntu:

# apt-get install lldpad

# service lldpad start

CentOS:

# yum install lldpad

# systemctl start lldpad

On the interface that is connected to the intermediate device (in this example, interface em1), enable bridge mode and configure the TLVs to be carried in LLDP messages.

# lldptool –i em1 –g ncb –L adminstatus=rxtx

# lldptool –g ncb -T -i em1 -V sysName enableTx=yes

# lldptool –g ncb -T -i em1 -V portDesc enableTx=yes

### Verifying the LLDP configuration

Execute the following command on the leaf devices to verify that the server information is displayed correctly.

<H3C> display lldp neighbor-information list

System Name Local Interface Chassis ID Port ID

ubuntu136 XGE2/0/5 2880-239d-4664 8cdc-d4b0-c5cd

# Installing or uninstalling the H3C ML2 drivers

H3C ML2 drivers are available in one version: Kilo.

As a best practice, use the Ubuntu14.04 LTS or CentOS 7 operating system for the server to host H3C ML2 driver. Before installing H3C ML2 drivers, make sure the required OpenStack neutron components have been installed.

## Installing the H3C ML2 drivers

H3C ML2 drivers are written in python. The driver installation package is a python egg file. Before installing H3C ML2 drivers, you must install the python setuptools.

To install H3C ML2 drivers:

On the server, install the python setuptools.

Ubuntu:

[user@localhost ~] $ sudo –s apt-get install python-pip python-setuptools

CentOS:

[root@localhost ~] # yum install –y python-pip python-setuptools

On the network device, download the python egg file. Make sure the file is integrated.

If you download the file through FTP, use binary mode.

Install the egg file.

[root@localhost ~]# easy\_install --no-deps h3c-1.0.0-py2.7.egg

Processing h3c-1.0-py2.7.egg

creating /usr/lib/python2.7/site-packages/h3c-1.0-py2.7.egg

Extracting h3c-1.0-py2.7.egg to /usr/lib/python2.7/site-packages

Adding h3c 1.0 to easy-install.pth file

Installing h3c\_config script to /usr/bin

Installed /usr/lib/python2.7/site-packages/h3c-1.0-py2.7.egg

## Uninstalling the H3C ML2 drivers

Remove the H3C database tables.

[root@localhost ~]# h3c\_config uninstall

Remove the H3C ML2 driver installation file.

[root@localhost ~]# pip uninstall h3c

Uninstalling h3c:

/usr/bin/h3c\_config

/usr/lib/python2.7/site-packages/h3c-1.0-py2.7.egg

Proceed (y/n)? y

Successfully uninstalled h3c

To remove the configurations specific to the H3C ML2 drivers, remove them from the ml2\_conf.ini file manually.

Restart the neutron server.

# Configuring the networking server component

## Tuning SQL connection performance parameters

In the /etc/neutron/neutron.conf file, edit the [database] section to set max pool size and max overflow to be identical to the number of leaf devices.

[database]

# Maximum number of SQL connections to keep open in a pool

# max\_pool\_size = 10

max\_pool\_size = 300

# If set, use this value for max\_overflow with sqlalchemy

# max\_overflow = 20

max\_overflow = 300

## Configuring the tenant network as a VLAN network

#### On the control node

Edit the /etc/neutron/plugin/ml2/ml2\_conf.ini file as follows:

Add a vlan type driver.

type\_drivers = vlan

Add h3c to the ML2 driver list.

mechanism\_driver = openvswitch, h3c

Specify vlan as the default tenant network type.

tenant\_network\_types=vlan

In the [ml2\_type\_vlan] section, specify a VLAN ID range for the tenant network. The format is *provide name*:vlan-id1:vlan-id2.

[ml2\_type\_vlan]

network\_vlan\_ranges = vlanphy:1000:2000

|  |  |
| --- | --- |
| IMPORTANT | IMPORTANT:  The provider name must be vlanphy. If other providers exist, make sure their names are not vlanphy.  The VLAN ID range is 0 to 4095, of which 0 and 4095 are reversed and IDs from 1 to 4094 are user configurable. |

#### On all network and compute node

Create br-vlan.

# ovs-vsctl add-br br-vlan

Add the interface that connects the server to the VLAN to br-vlan. This example uses interface em1.

# ovs-vsctl add-port br-vlan em1

Map vlanphy to br-vlan.

[ovs]

bridge\_mappings = external:br-ex,vlanphy:br-vlan

## Configuring the tenant network as a VXLAN network

#### On the control node

To configure the tenant network as a VXLAN network, edit the /etc/neutron/plugin/ml2/ml2\_conf.ini file as follows:

Add the h3c\_vxlan type driver to the type driver list.

type\_drivers = h3c\_vxlan

Add h3c to the mechanism driver list.

mechanism\_driver = openvswitch, h3c

Specify h3c\_vxlan as the default tenant network type.

tenant\_network\_types=h3c\_vxlan

Add the [ml2\_type\_h3c\_vxlan] section, and specify a VNI range for the VXLAN segments. The format isvxlan-id1:vxlan-id2.

[ml2\_type\_h3c\_vxlan]

vni\_ranges = 10000:60000

|  |  |
| --- | --- |
|  | NOTE:  The VNI range is 0 to 16777215. |

#### On all network and compute node

Create br-vlan.

# ovs-vsctl add-br br-vlan

Add the interface that connects the server to the fabric to br-vlan. This example uses interface em1.

# ovs-vsctl add-port br-vlan em1

Map vlanphy to br-vlan.

[ovs]

bridge\_mappings = external:br-ex,vlanphy:br-vlan

## Deploying L2 and L3 agents

### On the control node

For centralized deployment, deploy L2 and L3 agents as follows:

* Centos:

systemctl stop neutron-l3-agent

* Ubuntu:

service neutron-l3-agent stop

For distributed deployment, edit the /etc/neutron/neutron.conf file as follows:

l3\_ha = True

max\_l3\_agents\_per\_router = 300

min\_l3\_agents\_per\_router = 2

### On the network devices

Deploy L2 and L3 agents depending on the fabric deployment solution. See for centralized deployment. See for distributed deployment.

* + - * 1. L2 and L3 agents in centralized deployment

|  |  |
| --- | --- |
| Network device | Neutron component |
| Spine | Neutron-L2-Agent  Neutron-L3-Agent |
| Leaf | Neutron-L2-Agent |

* + - * 1. L2 and L3 agents in distributed deployment

|  |  |
| --- | --- |
| Network device | Neutron component |
| Spine | N/A |
| Leaf | Neutron-L2-Agent  Neutron-L3-Agent |

## Configuring H3C databases

Before configuring H3C databases, make sure you have finished configuring the neutron server.

[openstack@localhost ~]$ sudo h3c\_config db\_sync

## Restarting the neutron server

* Ubuntu:

[root@localhost ~]# service neutron-server restart

* CentOS:

[root@localhost ~]# systemctl restart neutron-server

# Troubleshooting

## Layer 3 configuration deployment failure

#### Symptom

When a router created through Horizon connects to the network, Layer 3 configurations for the router are not deployed to the network device.

#### Solution

To resolve the problem:

On the network device, execute the display tcp command to verify that the device has a TCP connection to the rabbitMQ server.

On the server, examine the h3c\_device\_topology and h3c\_host\_topology database tables to verify that the topologies are correct.

On the network device, execute the view /var/log/neutron.server.log | include ERROR command in probe view to verify that the commands are issued correctly.

## L3 agent failure

#### Symptom

The neutron agent-list command output of the command on the server shows that a L3 agent has failed.

#### Solution

To resolve the problem:

Use the ping command to verify that the device can ping the server.

Examine the rabbit\_user and rabbit\_password sections in the /etc/neutron/neutron.conf file to verify that these sections contain the same configuration as the template file on the network device.

## Slow deployment of Layer 3 configurations

#### Symptom

Deployment of Layer 3 configurations is slow on a network that uses distributed gateways.

#### Solution

Slow configuration deployment typically occurs when a large number of L3 agents are deployed. This is because the server deploys Layer 3 configurations in unicast mode.

To resolve the problem:

Remove l3\_agent\_scheduler from the neutron/services/l3\_router/l3\_router\_plugin.py file. The server will deploy Layer 3 configurations in broadcast mode.

supported\_extension\_aliases = ["dvr", "router", "ext-gw-mode",

"extraroute", ~~"l3\_agent\_scheduler",~~

"l3-ha"]

Edit the /etc/neutron/neutron.conf file to disable HA for L3 agents.

l3\_ha = False

Remove the interfaces bound to L3 agents from the def sync\_routers method in the neutron/api/rpc/handlers/l3\_rpc.py file.

def sync\_routers(self, context, \*\*kwargs):

…

~~if utils.is\_extension\_supported(~~

~~self.plugin, constants.PORT\_BINDING\_EXT\_ALIAS):~~

~~self.\_ensure\_host\_set\_on\_ports(context, host, routers)~~

LOG.debug("Routers %s returned to l3 agent:\n %s",

routers, utils.DelayedStringRenderer(jsonutils.dumps,

routers, indent=5))

return routers

If the problem persists, contact H3C Support.