Networking BGPVPN Documentation

Release 16.0.1.dev3

OpenStack Foundation

Feb 10, 2024

CONTENTS

BGP-based IP VPNs networks are widely used in the industry especially for enterprises. This project aims at supporting inter-connection between L3VPNs and Neutron resources, i.e. Networks, Routers and Ports.

A typical use-case is the following: a tenant already having a BGP IP VPN (a set of external sites) setup outside the datacenter, and they want to be able to trigger the establishment of connectivity between VMs and these VPN sites.

Another similar need is when E-VPN is used to provide an Ethernet interconnect between multiple sites, and inherits the base protocol architecture from BGP/MPLS IP VPNs.

This service plugin exposes an API to interconnect OpenStack Neutron ports, typically VMs, via the Networks and Routers they are connected to, with a IP VPN as defined by [RFC4364] (BGP/MPLS IP Virtual Private Networks) or with an E-VPN [RFC7432].

Introduction videos:

The following videos are filmed presentations of talks given during the Barcelona OpenStack Summit (Oct 2016). Although they do not cover the work done since, they can be a good introduction to the project:

- https://www.youtube.com/watch?v=kGW5R8mtmRg
- https://www.youtube.com/watch?v=LCDeR7MwTzE

CHAPTER

ONE

USER DOCUMENTATION

1.1 BGP VPN Interconnection Service Overview

BGP-based IP VPNs networks are widely used in the industry especially for enterprises. This project aims at supporting inter-connection between L3VPNs and Neutron resources, i.e. Networks, Routers and Ports.

A typical use-case is the following: a tenant already having a BGP IP VPN (a set of external sites) setup outside the datacenter, and they want to be able to trigger the establishment of connectivity between VMs and these VPN sites.

Another similar need is when E-VPN is used to provide an Ethernet interconnect between multiple sites, and inherits the base protocol architecture from BGP/MPLS IP VPNs.

This service plugin exposes an API to interconnect OpenStack Neutron ports, typically VMs, via the Networks and Routers they are connected to, with a IP VPN as defined by [RFC4364] (BGP/MPLS IP Virtual Private Networks) or with an E-VPN [RFC7432].

Introduction videos:

The following videos are filmed presentations of talks given during the Barcelona OpenStack Summit (Oct 2016). Although they do not cover the work done since, they can be a good introduction to the project:

- https://www.youtube.com/watch?v=kGW5R8mtmRg
- https://www.youtube.com/watch?v=LCDeR7MwTzE

1.1.1 Alternatives and related techniques

Other techniques are available to build VPNs, but the goal of this proposal is to make it possible to create the interconnect we need when the technology of BGP-based VPNs is already used outside an OpenStack cloud.

1.1.2 Reminder on BGP VPNs and Route Targets

BGP-based VPNs allow a network operator to offer a VPN service to a VPN customer, delivering isolating connectivity between multiple sites of this customer.

Unlike IPSec or SSL-based VPNs, for instance, these VPNs are not typically built over the Internet, are most often not encrypted, and their creation is not at the hand of the end-user.

Here is a reminder on how the connectivity is defined between sites of a VPN (VRFs).

In BGP-based VPNs, a set of identifiers called Route Targets are associated with a VPN, and in the typical case identify a VPN ; they can also be used to build other VPN topologies such as hubnspoke.

Each VRF (Virtual Routing and Forwarding) in a PE (Provider Edge router) imports/exports routes from/to Route Targets. If a VRF imports from a Route Target, BGP IP VPN routes will be imported in this VRF. If a VRF exports to a Route Target, the routes in the VRF will be associated to this Route Target and announced by BGP.

1.1.3 Mapping between PEs/CEs and Neutron constructs

As outlined in the overview, how PEs, CEs (Customer Edge router), VRFs map to Neutron constructs will depend on the backend driver used for this service plugin.

For instance, with the current bagpipe driver, the PE and VRF functions are implemented on compute nodes and the VMs are acting as CEs. This PE function will BGP-peer with edge IP/MPLS routers, BGP Route Reflectors or other PEs.

Bagpipe BGP which implements this function could also be instantiated in network nodes, at the l3agent level, with a BGP speaker on each l3agent; router namespaces could then be considered as CEs.

Other backends might want to consider the router as a CE and drive an external PE to peer with the service provider PE, based on information received with this API. Its up to the backend to manage the connection between the CE and the cloud provider PE.

Another typical option is where the driver delegates the work to an SDN controller which drives a BGP implementation advertising/consuming the relevant BGP routes and remotely drives the vswitches to setup the datapath accordingly.

1.1.4 API and Workflows

BGP VPN are deployed, and managed by the operator, in particular to manage Route Target identifiers that control the isolation between the different VPNs. Because of this BGP VPN parameters cannot be chosen by tenants, but only by the admin. In addition, network operators may prefer to not expose actual Route Target values to the users.

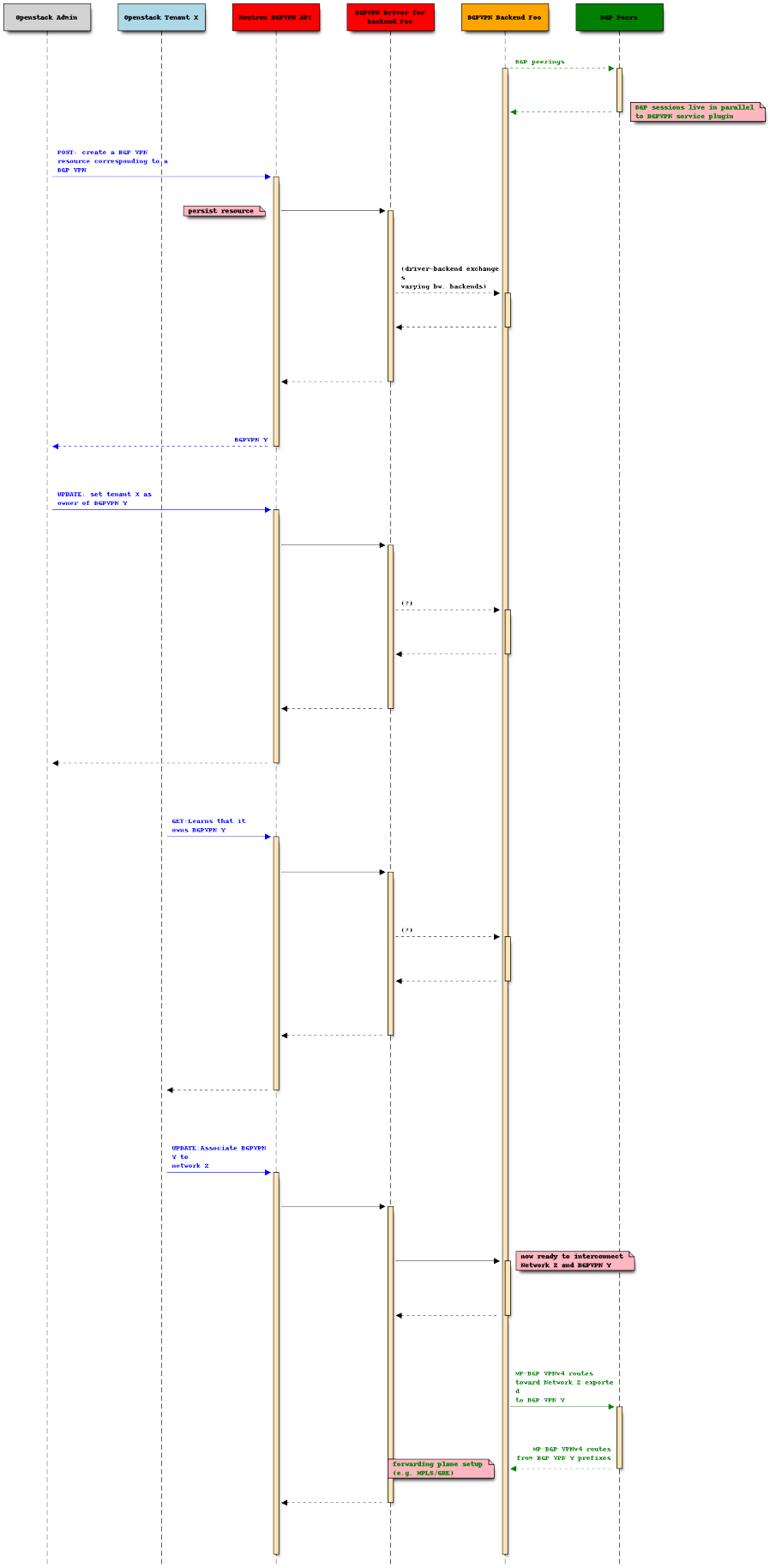
The operation that is let at the hand of a tenant is the association of a BGPVPN resource that it owns with his Neutron Networks or Routers.

So there are two workflows, one for the admin, one for a tenant.

- Admin/Operator Workflow: Creation of a BGPVPN
  - the cloud/network admin creates a BGPVPN for a tenant based on contract and OSS information about the VPN for this tenant
  - at this stage, the list of associated Networks and Routers can be empty

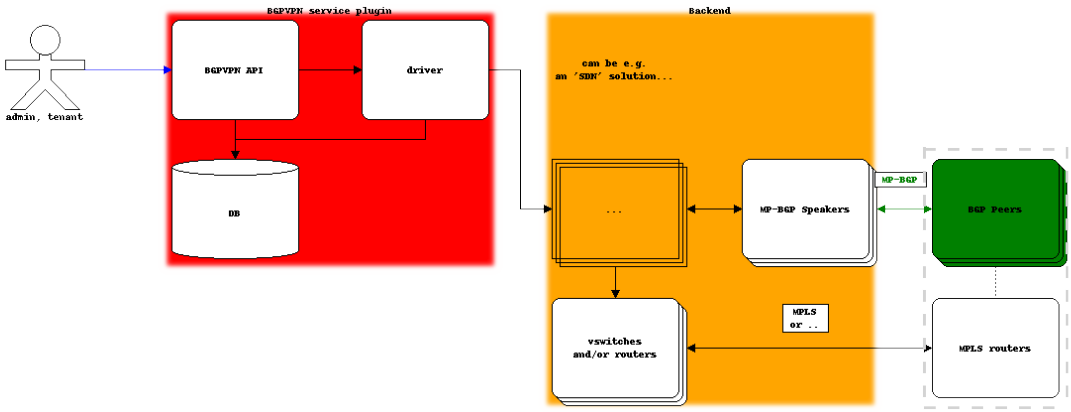
- Tenant Workflow: Association of a BGPVPN to Networks and/or Routers, on-demand
  - the tenant lists the BGPVPNs that it can use
  - the tenant associates a BGPVPN with one or more Networks or Routers.

Sequence diagram summarizing these two workflows:

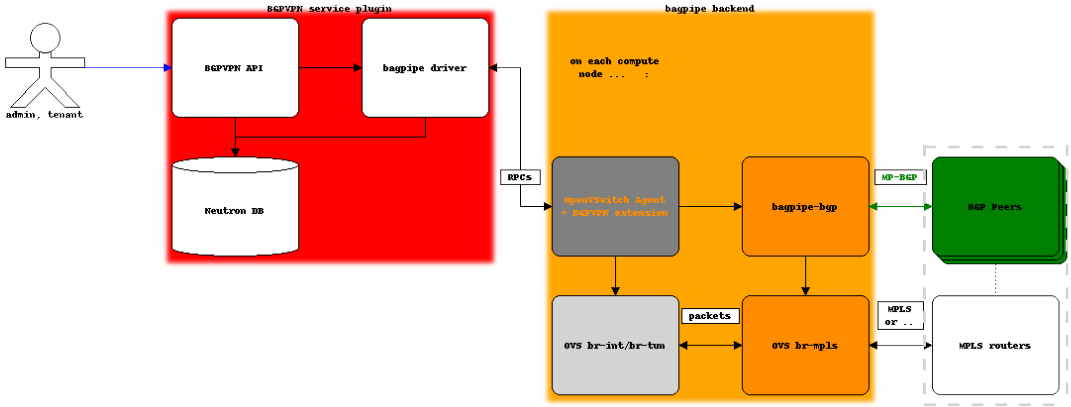

Openstack Admin
Openstack Tenant X
Neutron BGPVPN API
BGPVPN Driver for backend Foo
BGPVPN Backend Foo
BGP Peers
BGP peerings
BGP sessions live in parallel to BGPVPN service plugin
POST: create a BGP VPN resource corresponding to a BGP VPN
persist resource
(driver-backend exchanges varying bw. backends)
BGPVPN Y
UPDATE: set tenant X as owner of BGPVPN Y
(?)
GET:Learns that it owns BGPVPN Y
(?)
UPDATE Associate BGPVPN Y to network Z
now ready to interconnect Network Z and BGPVPN Y
MP-BGP VPNv4 routes toward Network Z exported to BGP VPN Y
MP-BGP VPNv4 routes from BGP VPN Y prefixes
forwarding plane setup (e.g. MPLS/GRE)

1.1.5 Component architecture overview

This diagram gives an overview of the architecture:



This second diagram depicts how the *bagpipe* reference driver implements its backend:



1.1.6 References

1.2 Drivers

The BGPVPN service plugin supports the following drivers:

1.2.1 OVS/linuxbridge driver (bagpipe)

Introduction

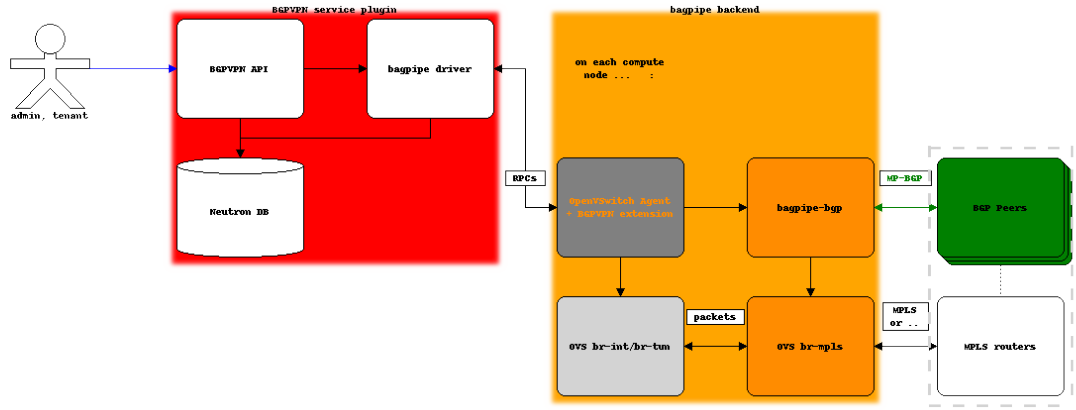
The **BaGPipe** driver for the BGPVPN service plugin is designed to work jointly with the openvswitch and linuxbridge ML2 mechanism drivers.

It relies on the use of the bagpipe-bgp BGP VPN implementation on compute nodes and the MPLS implementation in OpenVSwitch and or linuxbridge.

Architecture overview

The bagpipe driver for the BGPVPN service plugin interacts with the Neutron agent on each compute node, which is extended to support new RPCs to trigger the local configuration on compute nodes of BGP VPN instances and of their MPLS dataplane.

Example with the OpenVSwitch mechanism driver and agent:



Limitations

On DHCP ports, Router interface ports, external network ports, etc.

No connectivity will be setup with BGP VPNs for DHCP ports or Router interface ports, or other network specific ports. This improves the load on network nodes by avoiding them to import/export a significant amount of routes, without limiting BGP VPN deployment scenarios because no useful traffic would be exchanged between a router or DHCP interface of a network associated to a BGP VPN.

Similarly, the driver will not bind a port on an external network. This behavior will be revisited once a use case is well identified.

bagpipe_v2 driver

For Queens release, the mechanism used by this driver for RPCs was changed.

The v1 driver `networking_bgpvpn.neutron.services.service_drivers.bagpipe.bagpipe.BaGPipeBGPVPNDriver` is backwards compatible with pre-Queens neutron agents and can be used during a rolling upgrade, e.g. from Pike to Queens.

The v2 driver `networking_bgpvpn.neutron.services.service_drivers.bagpipe.bagpipe_v2.BaGPipeBGPVPNDriver` does not produce the old RPCs anymore and can be used:

- on a greenfield deployment
- after an upgrade
- during a non-rolling upgrade (some BGPVPN operations would be disrupted during the time where pre-Queens agent still run)

Future developments may happen only on the v2 driver and the v1 driver will be ultimately abandoned.

How to use ?

The steps to take to use this driver are generally:

- install the networking-bagpipe package on both control nodes and compute nodes
- on control node, configure neutron to use bagpipe driver
- on compute nodes, configure the neutron agent to use bagpipe_bgpvpn extension and configure bagpipe-bgp

Of course, the typical way is to have all this taken care of by an automated Openstack installer.

In devstack

- follow the instruction in README.rst
- `local.conf`:
  - add the following to enable the BaGPipe driver for the BGPVPN service plugin:

```
NETWORKING_BGPVPN_DRIVER="BGPVPN:BaGPipe:networking_bgpvpn.neutron.
↪services.service_drivers.bagpipe.bagpipe_v2.
↪BaGPipeBGPVPNDriver:default"
```

  - enable networking-bagpipe, which contains code for agent extensions:

```
enable_plugin networking-bagpipe https://opendev.org/openstack/
↪networking-bagpipe.git
# enable_plugin networking-bagpipe https://opendev.org/openstack/
↪networking-bagpipe.git stable/pike
# enable_plugin networking-bagpipe https://opendev.org/openstack/
↪networking-bagpipe.git stable/queens
```

- on a control node, if you want to run the Fake Route-Reflector there (relevant only for a multinode setup):

```
enable_service b-fakerr
```

- on compute nodes:
  - the compute node Neutron agent is the Neutron openvswitch or linuxbridge agent, with the `bagpipe_bgpvpn` agent extension:
    - install networking-bagpipe (the code to interact with `bagpipe-bgp` comes from there):

```
enable_plugin networking-bagpipe https://opendev.org/openstack/
↪networking-bagpipe.git
# enable_plugin networking-bagpipe https://opendev.org/openstack/
↪networking-bagpipe.git stable/queens
# enable_plugin networking-bagpipe https://opendev.org/openstack/
↪networking-bagpipe.git stable/pike
```

    - the `bagpipe_bgpvpn` agent extension is automatically added to the agent configuration by the devstack plugin

- bagpipe-bgp will be installed automatically (part of networking-bagpipe since Pike, or as a submodule before)
- you need to enable and configure bagpipe-bgp, typically with a peering to a BGP Route-Reflector or BGP router(s):

```
enable_service b-bgp

BAGPIPE_DATAPLANE_DRIVER_IPVPN=ovs
BAGPIPE_DATAPLANE_DRIVER_EVPN=ovs

# IP of your route-reflector or BGP router, or fakeRR
# BAGPIPE_BGP_PEERS defaults to $SERVICE_HOST, which will point to␣
↪the controller in a
# multi-node devstack setup
#BAGPIPE_BGP_PEERS=1.2.3.4,2.3.4.5
```

1.2.2 OpenContrail driver

Introduction

The **OpenContrail** driver for the BGPVPN service plugin is designed to work jointly with the OpenContrail SDN controller (GitHub). The BGP VPN driver can be found in the monolithic Neutron plugin tree[1].

Note: The BGPVPN Contrail driver that was under `networking_bgpvpn` (`networking_bgpvpn.neutron.services.service_drivers.opencontrail.opencontrail.OpenContrailBGPVPNDriver`) has been deprecated in Queens release, and has been completly removed in Stein release. The documentation below refers to the production ready driver under `Juniper/contrail-neutron-plugin`. Be careful, **no** migration path is planned.

---

[1] That driver requires OpenContrail release upper or equal to 4.0

Limitations

Route Distinguishers

The OpenContrail driver for the BGPVPN service plugin does not permit specifying route distinguisher.

Resource Association

The OpenContrail driver for the BGPVPN service plugin does not yet support association with ports. But it supports network associations and router associations.

VPN Type

The OpenContrail driver for the BGPVPN service plugin can create L2 & L3 VPN types for network associations and L3 VPN type for router association.

How to use ?

On an Openstack Installation

[TBC (package installation + config)]

In devstack

A devstack plugin can be used to setup an OpenContrail dev/test platform.

- Clone devstack:

```
git clone git@github.com:openstack-dev/devstack
```

- Here a proposed devstack `local.conf` file which permits to deploy OpenStack keystone, glance, nova, neutron/networking-bgpvpn and compile/install all OpenContrail services and dependencies:

```
[[local|localrc]]
LOG=True
LOGDAYS=1
PASSWORD="secret"
DATABASE_PASSWORD=$PASSWORD
RABBIT_PASSWORD=$PASSWORD
SERVICE_TOKEN=$PASSWORD
SERVICE_PASSWORD=$PASSWORD
ADMIN_PASSWORD=$PASSWORD

# disable some nova services
disable_service n-obj n-novnc n-cauth
# disable cinder
```

(continues on next page)

(continued from previous page)

```
disable_service cinder c-api c-vol c-sch
# disable heat
disable_service h-eng h-api h-api-cfn h-api-cw
# diable horizon
disable_service horizon
# disable swift
disable_service swift s-proxy s-object s-container s-account
# disable some contrail services
#disable_service ui-webs ui-jobs named dns query-engine

DEST=/opt/stack/openstack
CONTRAIL_DEST=/opt/stack/contrail

enable_plugin contrail https://github.com/zioc/contrail-devstack-plugin.git

enable_plugin networking-bgpvpn https://opendev.org/openstack/networking-
↪bgpvpn.git
NETWORKING_BGPVPN_DRIVER="BGPVPN:OpenContrail:neutron_plugin_contrail.plugins.
↪opencontrail.networking_bgpvpn.contrail.ContrailBGPVPNDriver:default"
```

1.2.3 OpenDaylight driver

The **OpenDaylight** driver for the BGPVPN service plugin is designed to work jointly with the OpenDaylight SDN controller.

OpenDaylight driver requires networking-odl plugin which comes with its own devstack scripts. Details on how to configure devstack for OpenDaylight plugin can be found at networking-odl/devstack.

Note: The legacy BGPVPN *v1 driver* for ODL that was hosted in `networking-bgpvpn` tree (`networking_bgpvpn.neutron.services.service_drivers.opendaylight.odl.OpenDaylightBgpvpnDriver`) has been deprecated in Rocky OpenStack release, and removed in Stein OpenStack release. The documentation below refers to the newer *v2 driver* in the `networking-odl` project.

- add the following to local.conf to enable networking-odl plugin:

```
enable_plugin networking-odl http://opendev.org/openstack/networking-odl
```

- add the following to local.conf to enable ODL Driver for BGPVPN service Plugin:

```
NETWORKING_BGPVPN_DRIVER="BGPVPN:OpenDaylight:networking_odl.bgpvpn.odl_
↪v2.OpenDaylightBgpvpnDriver:default"
```

- Run stack.sh:

```
./stack.sh
```

1.2.4 Nuage Networks driver

The Nuage Network driver works jointly with Nuage Networks VSP.

A pre-requisite for the nuage BGPVPN driver is that the Nuage-specific installation and configuration steps have been applied; in particular the installation of the `nuage_neutron` package. Please refer to Nuage Networks documentation.

The driver will be enabled, by specifying in `/etc/neutron/networking_bgpvpn.conf`:

```
[service_providers]
service_provider = BGPVPN:Nuage:nuage_neutron.bgpvpn.services.service_drivers.
↪driver.NuageBGPVPNDriver:default
```

The API is consistent across drivers, but not all drivers support all parts of the API. Refer to the Driver Compatibility Matrix to determine what is supported with each driver.

1.2.5 Driver Compatibility Matrix

API			Driver			
Object	Attribute		Neutron (bagpipe)	OpenContrail[1]	OpenDaylight[2]	Nuage
bgpvpn	base object		✓	✓	✓	✓
	type	L3	✓	✓	✓	✓
		L2	✓	✓	✓	
	route_targets		✓	✓	✓	✓
	import_targets		✓	✓	✓	✓
	export_targets		✓	✓	✓	✓
	route_distinguishers				✓	✓
	vni		✓		✓	
	local_pref		✓			
network_association	base object		✓	✓	✓	
router_association	base object		✓	✓	✓	✓
	advertise_extra_routes				[3]	
port_association	base object		✓			
	advertise_fixed_ips		✓			
	routes:prefix		✓			
	routes:bgpvpn		✓			
	routes:local_pref		✓			

[1] This applies to the current BGPVPN Contrail driver sometimes called *v2 driver*, which is different from the now obsolete *v1 driver* that was under `networking_bgpvpn`.

[2] This applies to the current BGPVPN ODL v2 driver sometimes called *v2 driver*, which is different from the now obsolete *v1 driver* that was under `networking_bgpvpn`.

[3] The behavior corresponding to `advertise_extra_routes: true`, is supported as the default with ODL, without support in the API for turning it off.

1.3 Usage

1.3.1 Use from OpenStack CLI

Example commands to use by the admin to create a BGPVPN resource:

```
openstack bgpvpn create --route-target 64512:1 --project␣
↪b954279e1e064dc9b8264474cb3e6bd2
openstack bgpvpn list
openstack bgpvpn set <bgpvpn-uuid> --name myBGPVPN
```

Example commands to use by the tenant owning the BGPVPN to associate a Network to it:

```
openstack bgpvpn network association create myBGPVPN <net-uuid>
# returns <net-assoc-uuid>
openstack bgpvpn network association list myBGPVPN
openstack bgpvpn network association show <net-assoc-uuid> myBGPVPN

openstack bgpvpn network association delete <net-assoc-uuid> myBGPVPN
```

There are more details in the OpenStack Client (OSC) documentation for BGPVPN.

1.3.2 Use from Horizon

See *Horizon*.

1.3.3 Use from Heat

See *Heat*.

1.3.4 Use from Python

The python `neutroclient` library includes support for the BGPVPN API extensions since Ocata release.

Note: For older releases, the dynamic extension of `neutronclient` provided in `networking-bgpvpn` is available. In that case, the methods to list, get, create, delete and update network associations and router associations are different from what is documented here:

- different name: `list_network_associations` instead of *list_bgpvpn_network_assocs*', and same change for all the methods
- order of parameters: BGPVPN UUID as first parameter, association UUID as second parameter

These old methods are deprecated.

Methods

BGPVPN Resources

Table 1: API methods for BGPVPN resources

Method Name	Description	Input parameter(s)	Output
list_bgpvpns()	Get the list of defined BGPVPN resources for the current tenant. An optional list of BGPVPN parameters can be used as filter.	1. Use **kwargs as filter, e.g. list_bgpvpn(param1=val1, param2=val2,) (Optional)	Dictionary of BGPVPN attributes
create_bgpvpn()	Create a BGPVPN resource for the current tenant. Extra information about the BGPVPN resource can be provided as input.	1. Dictionary of BGPVPN attributes (Optional)	Dictionary of BGPVPN attributes
show_bgpvpn()	Get all information for a given BGPVPN.	1. UUID of the said BGPVPN	Dictionary of BGPVPN attributes related to the BGPVPN provided as input
update_bgpvpn()	Update the BGPVPN resource with the parameters provided as input.	1. UUID of the said BGPVPN 2. Dictionary of BGPVPN attributes to be updated	Dictionary of BGPVPN attributes
delete_bgpvpn()	Delete a given BGPVPN resource of which the UUID is provided as input.	1. UUID of the said BGPVPN	Boolean

Network Association Resources

Table 2: API methods for Network association resources

Method Name	Description	Input parameter(s)	Output
list_bgpvpn_network_assocs()	Get the list of defined NETWORK ASSOCIATION resources for a given BGPVPN. An optional list of NETWORK ASSOCIATION parameters can be used as filter.	1. UUID of the BGPVPN 2. Use **kwargs as filter, e.g. list_bgpvpn_network_assocs(BGPVPN UUID, param1=val1, param2=val2,) (Optional)	List of dictionaries of NETWORK ASSOCIATION attributes, one of each related to a given BGPVPN
create_bgpvpn_network_assoc()	Create a NETWORK ASSOCIATION resource for a given BGPVPN. Network UUID must be defined, provided in a NETWORK ASSOCIATION resource as input parameter.	1. UUID of the said BGPVPN 2. Dictionary of NETWORK ASSOCIATION parameters	Dictionary of NETWORK ASSOCIATION attributes
show_bgpvpn_network_assoc()	Get all parameters for a given NETWORK ASSOCIATION.	1. UUID of the BGPVPN resource 2. UUID of the NETWORK ASSOCIATION resource	Dictionary of NETWORK ASSOCIATION parameters
update_bgpvpn_network_assoc()	Update the parameters of the NETWORK ASSOCIATION resource provided as input.	1. UUID of the BGPVPN resource 2. UUID of the NETWORK ASSOCIATION resource, 3. Dictionary of NETWORK ASSOCIATION parameters	Dictionary of NETWORK ASSOCIATION parameters
delete_bgpvpn_network_assoc()	Delete a given NETWORK ASSOCIATION resource of which the UUID is provided as input.	1. UUID of the BGPVPN resource 2. UUID of the NETWORK ASSOCIATION resource	Boolean

Router Association Resources

Table 3: API methods for Router association resources

Method Name	Description	Input parameter(s)	Output
list_bgpvpn_router_assocs()	Get the list of defined ROUTER ASSOCIATION resources for a given BGPVPN. An optional list of ROUTER ASSOCIATION parameters can be used as filter.	1. UUID of the BGPVPN 2. Use **kwargs as filter, e.g. list_bgpvpn_router_assocs(BGPVPN UUID, param1=val1, param2=val2,) (Optional)	List of dictionaries of ROUTER ASSOCIATION attributes, one of each related to a given BGPVPN
create_bgpvpn_router_assocs()	Create a ROUTER ASSOCIATION resource for a given BGPVPN. Router UUID must be defined, provided in a ROUTER ASSOCIATION resource as input parameter.	1. UUID of the said BGPVPN 2. Dictionary of ROUTER ASSOCIATION parameters	Dictionary of ROUTER ASSOCIATION attributes
show_bgpvpn_router_assoc()	Get all parameters for a given ROUTER ASSOCIATION.	1. UUID of the BGPVPN resource 2. UUID of the ROUTER ASSOCIATION resource	Dictionary of ROUTER ASSOCIATION parameters
update_bgpvpn_router_assoc()	Update the parameters of the ROUTER ASSOCIATION resource provided as input.	1. UUID of the BGPVPN resource 2. UUID of the ROUTER ASSOCIATION resource, 3. Dictionary of ROUTER ASSOCIATION parameters	Dictionary of ROUTER ASSOCIATION parameters
delete_bgpvpn_router_assoc()	Delete a given ROUTER ASSOCIATION resource of which the UUID is provided as input.	1. UUID of the BGPVPN resource 2. UUID of the ROUTER ASSOCIATION resource	Boolean

Port Association Resources

Table 4: API methods for Port association resources

Method Name	Description	Input parameter(s)	Output
list_bgpvpn_port_assocs()	Get the list of defined PORT ASSOCIATION resources for a given BGPVPN. An optional list of PORT ASSOCIATION parameters can be used as filter.	1. UUID of the BGPVPN 2. Use **kwargs as filter, e.g. list_bgpvpn_port_assocs(BGPVPN UUID, param1=val1, param2=val2,) (Optional)	List of dictionaries of PORT ASSOCIATION attributes, one of each related to a given BGPVPN
create_bgpvpn_port_assoc()	Create a PORT ASSOCIATION resource for a given BGPVPN. Port UUID must be defined, provided in a PORT ASSOCIATION resource as input parameter.	1. UUID of the said BGPVPN 2. Dictionary of PORT ASSOCIATION parameters	Dictionary of PORT ASSOCIATION attributes
show_bgpvpn_port_assoc()	Get all parameters for a given PORT ASSOCIATION.	1. UUID of the BGPVPN resource 2. UUID of the PORT ASSOCIATION resource	Dictionary of PORT ASSOCIATION parameters
update_bgpvpn_port_assoc()	Update the parameters of the PORT ASSOCIATION resource provided as input.	1. UUID of the BGPVPN resource 2. UUID of the PORT ASSOCIATION resource, 3. Dictionary of PORT ASSOCIATION parameters	Dictionary of PORT ASSOCIATION parameters
delete_bgpvpn_port_assoc()	Delete a given PORT ASSOCIATION resource of which the UUID is provided as input.	1. UUID of the BGPVPN resource 2. UUID of the PORT ASSOCIATION resource	Boolean

Examples

BGPVPN + Network Association Resources

```
# Copyright (c) 2016 Orange.
# All Rights Reserved.
#
#    Licensed under the Apache License, Version 2.0 (the "License"); you may
#    not use this file except in compliance with the License. You may obtain
#    a copy of the License at
#
#         http://www.apache.org/licenses/LICENSE-2.0
#
#    Unless required by applicable law or agreed to in writing, software
#    distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
#    WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
#    License for the specific language governing permissions and limitations
#    under the License.

import os
import sys

from keystoneauth1.identity import v3
from keystoneauth1 import session
from neutronclient.v2_0 import client

# Parameter for subnet neutron object
SUBNET_IP = "192.168.24.0/24"

# Parameters for bgpvpn neutron object
BGPVPN_RT = "64512:2"


# Function to obtain stack parameters from system vars
def get_keystone_creds():
    d = {}
    try:
        d['username'] = os.environ['OS_USERNAME']
        d['password'] = os.environ['OS_PASSWORD']
        d['auth_url'] = os.environ['OS_AUTH_URL']
        d['project_name'] = os.environ['OS_PROJECT_NAME']
        d['project_domain_id'] = os.environ['OS_PROJECT_DOMAIN_ID']
        d['user_domain_id'] = os.environ['OS_USER_DOMAIN_ID']
    except KeyError:
        print("ERROR: Stack environment variables type "
              "OS_* are not properly set")
        sys.exit(1)
    return d
```

(continues on next page)

(continued from previous page)

```
# Main function
def main():
    # Call function that imports (dev)stack vars
    creds = get_keystone_creds()

    # Authentication
    auth = v3.Password(**creds)
    sess = session.Session(auth=auth)

    # Neutron object
    # It dynamically loads the BGPVPN API
    neutron = client.Client(session=sess)

    try:
        # Network object creation. This dummy network will be used to bind the
        # attached subnet to the BGPVPN object.

        # Creation of the Network
        net_obj = neutron.create_network({'network': {'name': "dummyNet"}})
        # Verify creation
        print('Network created\t[network-id:%s]...' %
              net_obj['network']['id'])

        # Creation of the subnet, is attached to the created network
        subnet_obj = neutron.create_subnet(
            {'subnet':
                {'name': "dummySubnet",
                 'cidr': SUBNET_IP,
                 'network_id': net_obj['network']['id'],
                 'ip_version': 4}})
        # Verify
        print("Subnet created\t[subnet-id:%s]..." %
              subnet_obj['subnet']['id'])

        # Creation of a BGPVPN object. This object is created with the
        # required parameter 'routes_targets'.
        # This object can be created with others parameters or be updated with
        # them by calling the update function on the object.

        print("\nBGPVPN object handling.")
        # Creation of the BGPVPN object
        bgpvpn_obj = neutron.create_bgpvpn(
            {'bgpvpn': {'route_targets': [BGPVPN_RT]}})
        print("BGPVPN object created\t[bgpvpn-id:%s]..." %
              bgpvpn_obj['bgpvpn']['id'])
        # Update the BGPVPN object
        bgpvpn_obj = neutron.update_bgpvpn(
            bgpvpn_obj['bgpvpn']['id'], {'bgpvpn': {'name': "dummyBGPVPN"}})
        # List all BGPVPN objects
```

(continues on next page)

(continued from previous page)

```
        list_bgpvpn_obj = neutron.list_bgpvpns()
        print("List of all BGPVPN object\t[%s]" % list_bgpvpn_obj)
        # List of all BGPVPN objects filtered on the type parameter set to l3
        # value
        list_bgpvpn_obj = neutron.list_bgpvpns(type='l3')
        print("List of all BGPVPN object with type=l3\t[%s]" %
              list_bgpvpn_obj)

        # Creation of a BGPVPN Network association.
        print("\nBGPVPN Network Association object handling.")
        # Creation of a Network Association bound on the created BGPVPN object
        bgpvpn_net_assoc_obj = neutron.create_bgpvpn_network_assoc(
            bgpvpn_obj['bgpvpn']['id'],
            {'network_association':
                {'network_id':
                    net_obj['network']['id']}})
        print("BGPVPN Network Association created\t"
              "[network_association:%s]..." %
              bgpvpn_net_assoc_obj['network_association']['id'])
        # List all NETWORK ASSOCIATION object filtered on the network created
        # above
        list_bgpvpn_net_assoc_obj = neutron.list_bgpvpn_network_assocs(
            bgpvpn_obj['bgpvpn']['id'],
            network_id=net_obj['network']['id'])
        print("List of NETWORK ASSOCIATION objects using network_id"
              "[%s]\t[%s]" %
              (net_obj['network']['id'], list_bgpvpn_net_assoc_obj))

        # Deletion of all objects created in this example

        print("\nDeletion of all created objects")
        # First declared associations related of the created BGPVPN object in
        # this example
        neutron.delete_bgpvpn_network_assoc(
            bgpvpn_net_assoc_obj['network_association']['id'],
            bgpvpn_obj['bgpvpn']['id'])
        # Then the BGPVPN object
        neutron.delete_bgpvpn(bgpvpn_obj['bgpvpn']['id'])
        # Subnet
        neutron.delete_subnet(subnet_obj['subnet']['id'])
        # And finally the Network
        neutron.delete_network(net_obj['network']['id'])
    except Exception as e:
        print("[ERROR][%s]" % str(e))
        sys.exit(1)

    print("[Done]")
```

(continues on next page)

(continued from previous page)

```
if __name__ == '__main__':
    main()

__all__ = ['main']
```

1.4 Horizon

1.4.1 General information

Networking-bgpvpn contains the bgpvpn_dashboard plugin for Horizon. It adds a BGPVPN Interconnections panel in the admin section. Admin users can handle BGPVPNs resources through this panel. The operations possible for admin users are:

- listing BGPVPN
- creating a BGPVPN
- editing a BGPVPN
- associating or disassociating a BGPVPN to network(s)
- associating or disassociating a BGPVPN to router(s)
- deleting a BGPVPN

For non admin users the plugin adds a BGPVPN Interconnections panel in the Project section under the Network subsection. The operations possible for non admin users are:

- listing BGPVPN (display only name, type, networks and routers associations)
- editing a BGPVPN (only the name)
- associating or disassociating a BGPVPN to network(s)
- associating or disassociating a BGPVPN to router(s)

1.4.2 Installation and Configuration

Devstack will automatically configure Horizon to enable the Horizon plugin.

For others deployments we assume that Horizon and networking-bgpvpn are already installed. Their installation folders are respectively <horizon> and <networking-bgpvpn>.

Copy configuration file:

```
cp <networking-bgpvpn>/bgpvpn_dashboard/enabled/_[0-9]*.py <horizon>/
↪openstack_dashboard/local/enabled/
```

Configure the policy file for BGPVPN dashboard in OpenStack Dashboard `local_settings.py`. `<bgpvpn-dashboard-dir>` is a directory which contains configurations for BGPVPN dashboard and the location varies across distributions or deployments. `<bgpvpn-dashboard-dir>` can be found with: `dirname $(python -c 'import bgpvpn_dashboard as _; print _.__file__')`

```
POLICY_FILES[' networking-bgpvpn'] = '<bgpvpn-dashboard-dir>/bgpvpn_dashboard/
↪etc/bgpvpn-horizon.conf'
```

Note: If you do not configure `POLICY_FILES` in your `local_settings.py`, you also need to define the default `POLICY_FILES` in `local_settings.py`. If you use the example `local_settings.py` file from horizon, what you need is to uncomment `POLICY_FILES` (which contains the default values).

Restart the web server hosting Horizon.

The BGPVPN Interconnections panels will now be in your Horizon dashboard.

1.5 Heat

1.5.1 Installation and Configuration

Devstack will automatically configure heat to support BGPVPN.

Other deployments need to add the directory for the python networking_bgpvpn_heat module to `plugin_dirs` in the heat config: `/etc/heat/heat.conf`.

This directory can be found out with:

```
dirname $(python -c "import networking_bgpvpn_heat as n;print(n.__file__)")
```

1.5.2 Examples

Heat Orchestration Template (HOT) example 1

This template has to be run with admin rights and will create a BGPVPN for the current tenant, along with a Network associated with it:

```
description: BGPVPN networking example (admin)
heat_template_version: '2013-05-23'

resources:

  BGPVPN1:
    type: OS::Neutron::BGPVPN
    properties:
        import_targets: [ "100:1001"]
        export_targets: [ "100:1002"]
        route_targets: [ "100:1000" ]
        name: "default VPN"

  Net1:
    type: OS::Neutron::Net

  SubNet1:
```

(continues on next page)

(continued from previous page)

```
    type: OS::Neutron::Subnet
    properties:
       network: { get_resource: Net1 }
       cidr: 192.168.10.0/24

  BGPVPN_NET_assoc1:
    type: OS::Neutron::BGPVPN-NET-ASSOCIATION
    properties:
        bgpvpn_id: { get_resource: BGPVPN1 }
        network_id: { get_resource: Net1 }
```

In devstack, this HOT file can be used with cloud admin privileges in the demo project; such privileges can be obtained with the command:

```
source openrc admin demo
```

This example can then be run:

```
$ heat stack-create networks -f bgpvpn_test-00.yaml
+--------------------------------------+------------+--------------------+----
↪-----------------+--------------+
| id                                   | stack_name | stack_status       |␣
↪creation_time       | updated_time |
+--------------------------------------+------------+--------------------+----
↪-----------------+--------------+
| 5a6c2bf1-c5da-4f8f-9838-4c3e59d13d41 | networks   | CREATE_IN_PROGRESS |␣
↪2016-03-02T08:32:52 | None         |
+--------------------------------------+------------+--------------------+----
↪-----------------+--------------+

$ heat stack-list
+--------------------------------------+------------+-----------------+------
↪--------------+--------------+
| id                                   | stack_name | stack_status    |␣
↪creation_time       | updated_time |
+--------------------------------------+------------+-----------------+------
↪--------------+--------------+
| 5a6c2bf1-c5da-4f8f-9838-4c3e59d13d41 | networks   | CREATE_COMPLETE | 2016-
↪03-02T08:32:52 | None         |
+--------------------------------------+------------+-----------------+------
↪--------------+--------------+
```

Heat Orchestration Template (HOT) example 2

This is a set of two templates:

- one that has to be run with admin rights and will create a BGPVPN for the demo tenant:

```
description: BGPVPN networking example (admin)
heat_template_version: '2013-05-23'

resources:
  BGPVPN1:
    type: OS::Neutron::BGPVPN
    properties:
        import_targets: [ "100:1001"]
        export_targets: [ "100:1002"]
        route_targets: [ "100:1000" ]
        name: "default_vpn"
        tenant_id: "demo"
```

```
$ source openrc admin admin
$ heat stack-create bgpvpn -f bgpvpn_test-04-admin.yaml
```

- one to run as a plain demo tenant user, that will:
  - create a Network and bind it to the default_vpn BGPVPN
  - create a second Network connected to a Router, and bind the Router to the default_vpn

```
description: BGPVPN networking example (tenant)
heat_template_version: '2013-05-23'

resources:
    Net1:
        type: OS::Neutron::Net

    SubNet1:
        type: OS::Neutron::Subnet
        properties:
           network: { get_resource: Net1 }
           cidr: 192.168.10.0/24

    BGPVPN_NET_assoc1:
        type: OS::Neutron::BGPVPN-NET-ASSOCIATION
        properties:
            bgpvpn_id: "default_vpn"
            network_id: { get_resource: Net1 }

    Net2:
        type: OS::Neutron::Net

    SubNet2:
        type: OS::Neutron::Subnet
```

(continues on next page)

(continued from previous page)

```
        properties:
           network: { get_resource: Net2 }
           cidr: 192.168.10.0/24

    Router:
        type: OS::Neutron::Router

    router_interface:
        type: OS::Neutron::RouterInterface
        properties:
           router_id: { get_resource: Router }
           subnet_id: { get_resource: SubNet2 }

    BGPVPN_router_assoc1:
        type: OS::Neutron::BGPVPN-ROUTER-ASSOCIATION
        properties:
            bgpvpn_id: "default_vpn"
            router_id: { get_resource: Router }

    Net3:
        type: OS::Neutron::Net

    SubNet3:
        type: OS::Neutron::Subnet
        properties:
           network: { get_resource: Net3 }
           cidr: 192.168.10.0/24
    Port:
        type: OS::Neutron::Port
        properties:
           network: { get_resource: Net3 }
    BGPVPN_port_assoc1:
        type: OS::Neutron::BGPVPN-PORT-ASSOCIATION
        properties:
            bgpvpn_id: "default_vpn"
            port_id: { get_resource: Port }
```

```
$ source openrc demo demo
$ heat stack-create networks_bgpvpn -f bgpvpn_test-04-tenant.yaml
+--------------------------------------+----------------+---------------
↪----+---------------------+-------------+
| id                                   | stack_name     | stack_status ␣
↪    | creation_time       | updated_time |
+--------------------------------------+----------------+---------------
↪----+---------------------+-------------+
| a3cf1c1b-ac6c-425c-a4b5-d8ca894539f2 | networks_bgpvpn | CREATE_IN_
↪PROGRESS | 2016-03-02T09:16:39 | None         |
+--------------------------------------+----------------+---------------
↪----+---------------------+-------------+
```

(continues on next page)

(continued from previous page)

```
$ openstack bgpvpn list
+--------------------------------------+-------------+------+-------------
↪-----------------------------+----------------------------------------
↪-------+
| id                                   | name        | type | networks
↪                              | routers                                
↪       |
+--------------------------------------+-------------+------+-------------
↪-----------------------------+----------------------------------------
↪-------+
| 473e5218-f4a2-46bd-8086-36d6849ecf8e | default VPN | l3   | [u'5b1af75b-
↪0608-4e03-aac1-2608728be45d'] | [u'cb9c7304-e844-447d-88e9-4a0a2dc14d21
↪']      |
+--------------------------------------+-------------+------+-------------
↪-----------------------------+----------------------------------------
↪-------+
```

1.6 API

This API is documented in the Neutron API Reference.

1.6.1 ADMIN

1.6.2 Configuration

On VXLAN VNI

Note: This feature is under development in the Queens release

VXLAN is one option among others that could be used for BGP E-VPNs. When VXLAN is used on a hardware platform the use of a locally-assigned id may not be always possible which introduces the need to configure a globally-assigned VXLAN VNI.

The optional `vni` attribute is an admin-only parameter and allows the admin to enforce the use of a chosen globally-assigned VXLAN VNI for the said BGPVPN.

The default when no VNI is specified and the VXLAN encapsulation is used, is to let the backend choose the VNI in advertised routes, and use the VNI in received routes for transmitted traffic. The backend will conform to E-VPN overlay specs.

If the `vni` attribute is set for a BGPVPN, the following is enforced:

- the routes announced by the backend will advertise the specified VNI (this relates to traffic sent from this BGP VPN to a Network or Router)
- for the routes received by the backend for this BGPVPN, and that carry a different VNI that the VNI specified for the BGPVPN the behavior may depend on the backend, with the recommended

behavior being to liberally accept such routes.

If a backend does not support the approach recommended above of liberally accepting routes with a different VNI, the check can be implemented as follows:

- when a route is imported, for each BGPVPN associated to the Network or Router and having a VNI defined:
  - the set of Route Targets of the route is intersected with the import_rts of the BGPVPN
  - if this intersection is non-empty the `vni` of the BGPVPN is retained
- the route is used to establish connectivity to the destination in the forwarding plane only if the advertised VNI is equal to all retained VNIs in the previous step

The above check is applied similarly for a Router associated to multiple BGP VPN.

The backend is expected to provide troubleshooting information for the cases when a route ends up not being used because the VNI check failed.

Valid range for the `vni` attribute is [1, 2^{24}-1].

CHAPTER

TWO

INSTALL AND CONFIGURATION

2.1 Installation

The details related to how a package should be installed may depend on your environment.

If possible, you should rely on packages provided by your Linux and/or Openstack distribution.

If you use `pip`, follow these steps to install networking-bgpvpn:

- identify the version of the networking-bgpvpn package that matches your Openstack version:
  - Liberty: most recent of 3.0.x
  - Mitaka: most recent of 4.0.x
  - Newton: most recent of 5.0.x
  - Ocata: most recent of 6.0.x
  - Pike: most recent of 7.0.x
  - (see https://releases.openstack.org/index.html)
- indicate pip to (a) install precisely this version and (b) take into account Openstack upper constraints on package versions for dependencies (example for ocata):

```
$ pip install -c https://releases.openstack.org/constraints/upper/ocata
```

2.2 Configuration

The service plugin is enabled in Neutron, by adding `bgpvpn` to the list of enabled service plugins in `neutron.conf` (typically in `/etc/neutron/` but the location used may depend on your setup or packaging). For instance:

```
service_plugins = router,bgpvpn
```

The BGPVPN driver to use is then specified in the `networking_bgpvpn.conf` file (located by default under `/etc/neutron/`, but in any case in one of the directories specified with `--config-dir` at neutron startup, which may differ from `/etc/neutron` in your setup):

```
[service_providers]
service_provider = BGPVPN:BaGPipe:networking_bgpvpn.neutron.services.service_
↪drivers.bagpipe.bagpipe_v2.BaGPipeBGPVPNDriver:default
#service_provider= BGPVPN:Dummy:networking_bgpvpn.neutron.services.service_
↪drivers.driver_api.BGPVPNDriver:default
```

A given driver may require additional packages to work; the driver section provides detailed installation information for each specific driver.

2.3 Policy

API Policy for the BGPVPN service plugin can be controlled via the standard policy framework.

When pip is used to install the package, a default policy file is installed at `/etc/neutron/policy.d/bgpvpn.conf`.

2.4 Database setup

The DB tables for networking-bgpvpn are created and upgraded with:

```
neutron-db-manage --config-file /etc/neutron/neutron.conf --subproject␣
↪networking-bgpvpn upgrade
```

2.5 Devstack

You can easily test the bgpvpn service plugin with devstack, by adding the following line to your local.conf:

```
enable_plugin networking-bgpvpn https://git.openstack.org/openstack/
↪networking-bgpvpn.git
```

Or the following if you want a specific branch or version (example for Mitaka):

```
enable_plugin networking-bgpvpn https://git.openstack.org/openstack/
↪networking-bgpvpn.git stable/mitaka
```

By default, the service driver will use a dummy driver, that only responds to API calls, and stores data in the database. If you want to test a fully functional driver with devstack, you can configure the bagpipe driver with its devstack plugin (see *OVS/linuxbridge driver (bagpipe)*).

Detailed information on how to use other drivers is provided in the documentation for each of these drivers.

CHAPTER

THREE

CONFIGURATION GUIDE

3.1 Configuration

This section provides a list of all possible options for each configuration file. networking-bgpvpn uses the following configuration file.

3.1.1 networking-bgpvpn.conf

To use networking-bgpvpn, you need to configure one of valid service providers for `BGPVPN` service in `service_provider` of `[service_providers]` group of the neutron server. Note that you can specify multiple providers for BGPVPN but only one of them can be default.

- Dummy provider: `service_provider = BGPVPN:Dummy:networking_bgpvpn.neutron.services.service_drivers.driver_api.BGPVPNDriver:default`
- BaGPipe provider: `service_provider = BGPVPN:BaGPipe:networking_bgpvpn.neutron.services.service_drivers.bagpipe.bagpipe.BaGPipeBGPVPNDriver:default`

service_providers

`service_provider`

> **Type** multi-valued
>
> **Default** `BGPVPN:Dummy:networking_bgpvpn.neutron.services.service_drivers.driver_api.BGPVPNDriver:default`

Defines providers for advanced services using the format: <service_type>:<name>:<driver>[:default]

The following is a sample configuration file for networking-bgpvpn. It is generated from code and reflect the current state of code in the networking-bgpvpn repository.

3.1.2 Sample networking-bgpvpn.conf

This sample configuration can also be viewed in the raw format.

```
[DEFAULT]


[service_providers]

#
# From networking-bgpvpn.service_provider
#

# Defines providers for advanced services using the format:
# <service_type>:<name>:<driver>[:default] (multi valued)
#service_provider = BGPVPN:Dummy:networking_bgpvpn.neutron.services.service_
↪drivers.driver_api.BGPVPNDriver:default
```

3.2 Policy

networking-bgpvpn, like most OpenStack projects, uses a policy language to restrict permissions on REST API actions.

3.2.1 networking-bgpvpn policies

The following is an overview of all available policies in networking-bgpvpn. For a sample configuration file, refer to *Sample networking-bgpvpn Policy File*.

networking-bgpvpn

create_bgpvpn

Default `rule:admin_only`

Operations

- **POST** `/bgpvpn/bgpvpns`

Create a BGP VPN

update_bgpvpn

Default `rule:admin_or_owner`

Operations

- **PUT** `/bgpvpn/bgpvpns/{id}`

Update a BGP VPN

update_bgpvpn:tenant_id

Default `rule:admin_only`

Operations

- **PUT** /bgpvpn/bgpvpns/{id}

Update tenant_id attribute of a BGP VPN

update_bgpvpn:route_targets

Default rule:admin_only

Operations

- **PUT** /bgpvpn/bgpvpns/{id}

Update route_targets attribute of a BGP VPN

update_bgpvpn:import_targets

Default rule:admin_only

Operations

- **PUT** /bgpvpn/bgpvpns/{id}

Update import_targets attribute of a BGP VPN

update_bgpvpn:export_targets

Default rule:admin_only

Operations

- **PUT** /bgpvpn/bgpvpns/{id}

Update export_targets attribute of a BGP VPN

update_bgpvpn:route_distinguishers

Default rule:admin_only

Operations

- **PUT** /bgpvpn/bgpvpns/{id}

Update route_distinguishers attribute of a BGP VPN

update_bgpvpn:vni

Default rule:admin_only

Operations

- **PUT** /bgpvpn/bgpvpns/{id}

Update vni attribute of a BGP VPN

delete_bgpvpn

Default rule:admin_only

Operations

- **DELETE** /bgpvpn/bgpvpns/{id}

Delete a BGP VPN

get_bgpvpn

Default `rule:admin_or_owner`

Operations

- **GET** `/bgpvpn/bgpvpns`
- **GET** `/bgpvpn/bgpvpns/{id}`

Get BGP VPNs

`get_bgpvpn:tenant_id`

Default `rule:admin_only`

Operations

- **GET** `/bgpvpn/bgpvpns`
- **GET** `/bgpvpn/bgpvpns/{id}`

Get `tenant_id` attributes of BGP VPNs

`get_bgpvpn:route_targets`

Default `rule:admin_only`

Operations

- **GET** `/bgpvpn/bgpvpns`
- **GET** `/bgpvpn/bgpvpns/{id}`

Get `route_targets` attributes of BGP VPNs

`get_bgpvpn:import_targets`

Default `rule:admin_only`

Operations

- **GET** `/bgpvpn/bgpvpns`
- **GET** `/bgpvpn/bgpvpns/{id}`

Get `import_targets` attributes of BGP VPNs

`get_bgpvpn:export_targets`

Default `rule:admin_only`

Operations

- **GET** `/bgpvpn/bgpvpns`
- **GET** `/bgpvpn/bgpvpns/{id}`

Get `export_targets` attributes of BGP VPNs

`get_bgpvpn:route_distinguishers`

Default `rule:admin_only`

Operations

- **GET** `/bgpvpn/bgpvpns`
- **GET** `/bgpvpn/bgpvpns/{id}`

Get route_distinguishers attributes of BGP VPNs

get_bgpvpn:vni

Default rule:admin_only

Operations

- **GET** /bgpvpn/bgpvpns
- **GET** /bgpvpn/bgpvpns/{id}

Get vni attributes of BGP VPNs

create_bgpvpn_network_association

Default rule:admin_or_owner

Operations

- **POST** /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations

Create a network association

update_bgpvpn_network_association

Default rule:admin_or_owner

Operations

- **PUT** /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations/{network_association_id}

Update a network association

delete_bgpvpn_network_association

Default rule:admin_or_owner

Operations

- **DELETE** /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations/{network_association_id}

Delete a network association

get_bgpvpn_network_association

Default rule:admin_or_owner

Operations

- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations
- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations/{network_association_id}

Get network associations

get_bgpvpn_network_association:tenant_id

Default rule:admin_only

Operations

- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations

- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations/{network_association_id}

Get tenant_id attributes of network associations

create_bgpvpn_router_association

Default rule:admin_or_owner

Operations

- **POST** /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations

Create a router association

update_bgpvpn_router_association

Default rule:admin_or_owner

Operations

- **PUT** /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations/{router_association_id}

Update a router association

delete_bgpvpn_router_association

Default rule:admin_or_owner

Operations

- **DELETE** /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations/{router_association_id}

Delete a router association

get_bgpvpn_router_association

Default rule:admin_or_owner

Operations

- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations
- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations/{router_association_id}

Get router associations

get_bgpvpn_router_association:tenant_id

Default rule:admin_only

Operations

- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations
- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations/{router_association_id}

Get tenant_id attributes of router associations

create_bgpvpn_port_association

Default rule:admin_or_owner

Operations

- **POST** /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations

Create a port association

`update_bgpvpn_port_association`

Default rule:admin_or_owner

Operations

- **PUT** /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations/{port_association_id}

Update a port association

`delete_bgpvpn_port_association`

Default rule:admin_or_owner

Operations

- **DELETE** /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations/{port_association_id}

Delete a port association

`get_bgpvpn_port_association`

Default rule:admin_or_owner

Operations

- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations
- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations/{port_association_id}

Get port associations

`get_bgpvpn_port_association:tenant_id`

Default rule:admin_only

Operations

- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations
- **GET** /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations/{port_association_id}

Get tenant_id attributes of port associations

3.2.2 Sample networking-bgpvpn Policy File

The following is a sample networking-bgpvpn policy file for adaptation and use.

The sample policy can also be viewed in `file form`.

Important: The sample policy file is auto-generated from networking-bgpvpn when this documentation is built. You must ensure your version of networking-bgpvpn matches the version of this documentation.

```
# Create a BGP VPN
# POST  /bgpvpn/bgpvpns
#"create_bgpvpn": "rule:admin_only"

# Update a BGP VPN
# PUT  /bgpvpn/bgpvpns/{id}
#"update_bgpvpn": "rule:admin_or_owner"

# Update ``tenant_id`` attribute of a BGP VPN
# PUT  /bgpvpn/bgpvpns/{id}
#"update_bgpvpn:tenant_id": "rule:admin_only"

# Update ``route_targets`` attribute of a BGP VPN
# PUT  /bgpvpn/bgpvpns/{id}
#"update_bgpvpn:route_targets": "rule:admin_only"

# Update ``import_targets`` attribute of a BGP VPN
# PUT  /bgpvpn/bgpvpns/{id}
#"update_bgpvpn:import_targets": "rule:admin_only"

# Update ``export_targets`` attribute of a BGP VPN
# PUT  /bgpvpn/bgpvpns/{id}
#"update_bgpvpn:export_targets": "rule:admin_only"

# Update ``route_distinguishers`` attribute of a BGP VPN
# PUT  /bgpvpn/bgpvpns/{id}
#"update_bgpvpn:route_distinguishers": "rule:admin_only"

# Update ``vni`` attribute of a BGP VPN
# PUT  /bgpvpn/bgpvpns/{id}
#"update_bgpvpn:vni": "rule:admin_only"

# Delete a BGP VPN
# DELETE  /bgpvpn/bgpvpns/{id}
#"delete_bgpvpn": "rule:admin_only"

# Get BGP VPNs
# GET  /bgpvpn/bgpvpns
# GET  /bgpvpn/bgpvpns/{id}
#"get_bgpvpn": "rule:admin_or_owner"
```

(continues on next page)

(continued from previous page)

```
# Get ``tenant_id`` attributes of BGP VPNs
# GET  /bgpvpn/bgpvpns
# GET  /bgpvpn/bgpvpns/{id}
#"get_bgpvpn:tenant_id": "rule:admin_only"

# Get ``route_targets`` attributes of BGP VPNs
# GET  /bgpvpn/bgpvpns
# GET  /bgpvpn/bgpvpns/{id}
#"get_bgpvpn:route_targets": "rule:admin_only"

# Get ``import_targets`` attributes of BGP VPNs
# GET  /bgpvpn/bgpvpns
# GET  /bgpvpn/bgpvpns/{id}
#"get_bgpvpn:import_targets": "rule:admin_only"

# Get ``export_targets`` attributes of  BGP VPNs
# GET  /bgpvpn/bgpvpns
# GET  /bgpvpn/bgpvpns/{id}
#"get_bgpvpn:export_targets": "rule:admin_only"

# Get ``route_distinguishers`` attributes of BGP VPNs
# GET  /bgpvpn/bgpvpns
# GET  /bgpvpn/bgpvpns/{id}
#"get_bgpvpn:route_distinguishers": "rule:admin_only"

# Get ``vni`` attributes of BGP VPNs
# GET  /bgpvpn/bgpvpns
# GET  /bgpvpn/bgpvpns/{id}
#"get_bgpvpn:vni": "rule:admin_only"

# Create a network association
# POST  /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations
#"create_bgpvpn_network_association": "rule:admin_or_owner"

# Update a network association
# PUT  /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations/{network_association_
↪id}
#"update_bgpvpn_network_association": "rule:admin_or_owner"

# Delete a network association
# DELETE  /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations/{network_
↪association_id}
#"delete_bgpvpn_network_association": "rule:admin_or_owner"

# Get network associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations/{network_association_
↪id}
#"get_bgpvpn_network_association": "rule:admin_or_owner"
```

(continues on next page)

(continued from previous page)

```
# Get ``tenant_id`` attributes of network associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/network_associations/{network_association_
↪id}
#"get_bgpvpn_network_association:tenant_id": "rule:admin_only"

# Create a router association
# POST  /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations
#"create_bgpvpn_router_association": "rule:admin_or_owner"

# Update a router association
# PUT  /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations/{router_association_id}
#"update_bgpvpn_router_association": "rule:admin_or_owner"

# Delete a router association
# DELETE  /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations/{router_association_
↪id}
#"delete_bgpvpn_router_association": "rule:admin_or_owner"

# Get router associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations/{router_association_id}
#"get_bgpvpn_router_association": "rule:admin_or_owner"

# Get ``tenant_id`` attributes of router associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/router_associations/{router_association_id}
#"get_bgpvpn_router_association:tenant_id": "rule:admin_only"

# Create a port association
# POST  /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations
#"create_bgpvpn_port_association": "rule:admin_or_owner"

# Update a port association
# PUT  /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations/{port_association_id}
#"update_bgpvpn_port_association": "rule:admin_or_owner"

# Delete a port association
# DELETE  /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations/{port_association_id}
#"delete_bgpvpn_port_association": "rule:admin_or_owner"

# Get port associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations/{port_association_id}
#"get_bgpvpn_port_association": "rule:admin_or_owner"

# Get ``tenant_id`` attributes of port associations
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations
```

(continues on next page)

(continued from previous page)

```
# GET  /bgpvpn/bgpvpns/{bgpvpn_id}/port_associations/{port_association_id}
#"get_bgpvpn_port_association:tenant_id": "rule:admin_only"
```

CHAPTER

FOUR

CONTRIBUTOR GUIDE

4.1 Contributing

If you would like to contribute to the development of OpenStack, you must follow the steps in this page:

https://docs.openstack.org/infra/manual/developers.html

Once those steps have been completed, changes to OpenStack should be submitted for review via the Gerrit tool, following the workflow documented at:

https://docs.openstack.org/infra/manual/developers.html#development-workflow

Pull requests submitted through GitHub will be ignored.

Bugs should be filed on Launchpad, not GitHub:

https://bugs.launchpad.net/bgpvpn

4.2 Specification notes

4.2.1 Database design

The implementation will rely on three tables:

- one for BGPVPN objects
- one to define the n-n relation ship between BGPVPNs and Networks
- one to define the n-n relation ship between BGPVPNs and Routers

The information stored in these tables will reflect what is exposed on the API.

4.3 To be implemented

4.3.1 Future attributes

Specifications for the following attributes have been defined but not implemented yet:

Table 1: Future attributes

Attribute Name	Type	Access	Default Value	Validation/Constraint	Description
technique	string	RW admin only	None	for instance ipvpn or evpn	(optional) selection of the technique used to implement the VPN
auto_aggregate	bool	RW admin only	False	{ True | False }	enable prefix aggregation or not (type l3 only) but no support in any driver
admin_state_up	bool	RW admin only	True	{ True | False }	interconnection with this BGPVPN is enabled by the admin

auto_aggregate attribute

The auto_aggregate flag controls whether or not routes should be automatically aggregated before being advertised outside Neutron. A backend may or may not support this behavior, and its driver should report an API error in the latter case.

technique attribute

The technique attribute is optional and can be used by the admin to select one of multiple techniques when more than one is supported by the driver. When no technique is specified, the driver will use a default value. An API call will be available to let the API user know about the types supported by the driver for a said vpn type.

Currently defined techniques are:

- for l3:
  - ipvpn: this corresponds to RFC4364
  - evpn-prefix: this corresponds to draft-ietf-bess-evpn-prefix-advertisement
- for l2:
  - evpn: this corresponds to RFC7432

API call to list the available techniques, with example answers:

- GET /bgpvpn/techniques:

```
{ "techniques": {
   "l3": [ "ipvpn" ],
   "l2": [ "evpn" ]
} }
```

- GET /bgpvpn/techniques/l3:

```
{ "l3": [ "ipvpn"] }
```

- GET /bgpvpn/techniques/l2:

```
{ "l2": [ "evpn"] }
```

admin_state_up attribute

This is an admin-only attribute allowing the admin to shutdown connectivity to and from a BGP VPN and expose this state to the tenant.

BIBLIOGRAPHY

[RFC4364] BGP/MPLS IP Virtual Private Networks (IP VPNs) http://tools.ietf.org/html/rfc4364

[RFC7432] BGP MPLS-Based Ethernet VPN (Ethernet VPNs, a.k.a E-VPN) http://tools.ietf.org/html/rfc7432