neutron-dynamic-routing Documentation

Release 25.1.0.dev12

OpenStack Foundation

Jan 20, 2025

CONTENTS

CHAPTER ONE

INSTALLATION

At the command line:

```
$ pip install neutron-dynamic-routing
```

Or, if you have virtualenv wrapper installed:

```
$ mkvirtualenv neutron-dynamic-routing
$ pip install neutron-dynamic-routing
```

CHAPTER

TWO

ADMINISTRATION GUIDE

2.1 System Design

2.1.1 Introduction

Neutron dynamic routing enables advertisement of self-service (private) network prefixes to physical network devices that support dynamic routing protocols such as routers, thus removing the conventional dependency on static routes.

It advertises three classes of routes:

- Host routes for floating IP addresses hosted on non-DVR routers, the nexthop is the centralized router.
- Host routes for floating IP addresses hosted on DVR routers, the nexthop is the appropriate compute node.
- Prefix routes for directly routable tenant networks with address scopes, the nexthop is the centralized router, the same for DVR and CVR.

For details refer to Route Advertisement.

Neutron dynamic routing consists of service plug-in and agent. The service plug-in implements the Networking service extension and the agent manages dynamic routing protocol peering sessions. The plug-in communicates with the agent through RPC.

2.1.2 Architecture

The following figure shows the architecture of this feature:

```
 Neutron dynamic Routing System Architecture
+----------------------------------------------------------------+
|                  Dynamic Routing plug-in                       |
|  +----------------------------------------------------------+  |
|  |               Dynamic Routing API/Model                  |  |
|  +----------------------------------------------------------+  |
|  |               Dynamic Routing Agent Scheduler            |  |
|  +----------------------------------------------------------+  |
|                              |                                 |
+------------------------------|---------------------------------+
                               |
                               |
                         +-----------+
```

(continues on next page)

(continued from previous page)

```
                         |    RPC    |
                         +-----------+
                               |
                               |
         +---------------------|------------------------+
         |                                              |
         |                                              |
+-------------------------+    +-------------------------+
|   Dynamic Routing Agent1  |    |   Dynamic Routing Agent2  |
|                           |    |                           |
|  +--------------------+   |    |  +--------------------+   |
|  |  Driver Manager    |   |    |  |  Driver Manager    |   |
|  +--------------------+   |    |  +--------------------+   |
|  |  Common Driver API |   |    |  |  Common Driver API |   |
|  +--------------------+   |    |  +--------------------+   |
|            |              |    |            |              |
|  +---------+-----------+  |    |  +---------+-----------+  |
|  | os-ken  |  Other    |  |    |  | os-ken  |  Other    |  |
|  | Driver  |  Drivers  |  |    |  | Driver  |  Drivers  |  |
|  +---------+-----------+  |    |  +---------+-----------+  |
|                           |    |                           |
+-------------------------+    +-------------------------+
```

Dynamic Routing Plug-in

Using dynamic routing plugin one can enable/disable the support of dynamic routing protocols in neutron.

Dynamic Routing API

Dynamic routing API provides APIs to configure dynamic routing. APIs for below mentioned dynamic protocols are supported.

BGP

Three kinds of APIs are available for BGP functionality.For details refer to the API document.

- BGP Speaker APIs to advertise Neutron routes outside the Openstack network.
- BGP Peer APIs to form peers with the remote routers.
- BGP DRAgentScheduler APIs to schedule BGP Speaker(s) to one or more hosts running the dynamic routing agent.

Note

BGP is the only dynamic routing protocol currently supported.

Dynamic Routing Model

Dynamic routing model maintains the database and communicates with the dynamic routing agent.

Dynamic Routing Agent Scheduler

Dynamic routing agent scheduler, is responsible for scheduling a routing entity. For details refer to Agent Scheduler.

Dynamic Routing Agent (DR Agent)

Dynamic routing can reside on hosts with or without other Networking service agents. It manages and configures different dynamic routing stack through Common Driver API.

Note

Currently, only integration with os-ken is supported.

2.2 BGP Speaker

BGP Speaker acts as a route server using BGP routing protocol. It advertises routes to the BGP peers which are added to the BGP Speaker. Now there is a framework that allows different BGP drivers to be plugged into a dynamic routing agent.

Currently, BGP Speaker only advertises routes for a network to which it is associated. A BGP Speaker requires association with a gateway network to determine eligible routes. In Neutron, a gateway network connects Neutron routers to the upstream routers. An external network is best for being used as a gateway network. The association builds a list of all virtual routers with gateways on provider and self-service networks within the same address scope. Hence, the BGP speaker advertises self-service network prefixes with the corresponding router as the next-hop IP address. For details refer to Route advertisement.

2.2.1 Address Scopes

Address scopes provide flexible control as well as decoupling of address overlap from tenancy, so this kind control can provide a routable domain, the domain has itself route and no overlap address, it means an address scope define a L3 routing domain.

BGP Speaker will associate the external networks and advertise the tenants networks routes. Those networks should reside in the same address scope. Neutron can route the tenant network directly without NAT. Then Neutron can host globally routable IPv4 and IPv6 tenant networks. For determining which tenant networks prefixes should be advertised, Neutron will identify all routers with gateway ports on the network which had been bounded with BGP Speaker, check the address scope of the subnets on all connected networks, then begin advertising nexthops for all tenant networks to routers on the bound network.

2.2.2 BGP Peer

BGP peer defined in Neutron represents real BGP infrastructure such as routers, route reflectors and route servers. When a BGP peer is defined and associated with a BGP Speaker, Neutron will attempt to open a BGP peering session with the mentioned remote peer. It is this session, using which Neutron announces its routes.

How to configure a remote peer

A remote peer can be real or virtual e.g. vRouters or real routers. The remote peer should be configured to handle peering with Neutron in passive mode. The peer needs to waits for the Neutron dynamic routing agent to initiate the peering session. Also, the remote peer can be configured in active mode, but it still can speak BGP until the complete initialization of BGP Speaker running on Neutron dynamic routing agent.

Configuring BGP Speaker: One needs to ensure below points for setting a BGP connection.

- Host running Neutron dynamic agent MUST connect to the external router.
- BGP configuration on the router should be proper.

  bgp router-id XX.XX.XX.XX
  : This must be an IP address, the unique identifier of BGP routers actually and can be virtual. If one doesnt configure the router-id, it will be selected automatically as the highest IP address configured for the local interfaces. Just a suggestion, please make sure that it is the same as the `peer_ip` which you configure in Neutron for distinguishing easily.

  local_as
  : Autonomous System number can be same or different from the AS_id of external BGP router. AS_id will be same for iBGP and different for eBGP sessions.

Setting BGP peer:

```
neighbor A.B.C.D remote-as AS_ID
A.B.C.D is the host IP which run Neutron dynamic routing agent.
```

A Sample Quagga router configuration file forming BGP peering with Neutron:

```
!
password zebra
log file /var/log/quagga/bgpd.log
!
debug bgp events
debug bgp keepalives
debug bgp updates
debug bgp fsm
debug bgp filters
!
bgp multiple-instance
!
router bgp <BgpPeer remote_as> view test-as
 bgp router-id <quagga router IP address>
 neighbor <dr_agent IP address> remote-as <BgpSpeaker local_as>
 neighbor <dr_agent IP address> passive
!
line vty
!
```

2.2.3 BGP Speaker Architecture

Dynamic routing project saves BGP Speaker configuration as per the defined data model. and pass on the configuration request to the dynamic routing agent for further processing. The implementation of a BGP Speaker is driver specific. During the driver interface initialization process, needed configurations are read from the configuration file and BGP Speaker object instance is created. For details refer to BGP drivers.

BGP Speaker Life Cycle

Now we support OsKenBgpDriver, BGP Speaker will be processed by Dragent. When associating a BGP Speaker with an active Dragent, the plugin will send an RPC message to the agent for calling driver in order to create a BGP Speaker instance.

In OsKenBgpDriver, the created instance `BGP Speaker` will setup by router-id and ASN, then os-ken will setup new context with speaker configuration and listeners which monitor whether the related peers are alive.

Then the following operation could be done.

- Add peers to BGP Speaker When BGP Speaker is not associated with an active Dragent, there is no real speaker instance, so it will be still the db operation until the speaker is associated with dragent, and all the peers connection before will be setup by `BGP Speaker` creation. If add peers into speaker which is running, Dragent will call driver to add peer dynamically. For OsKenBgpDriver, it will register a new neighbor based on your peer configuration and try to establish a session with the peer.
- Delete peers from BGP Speaker The same logic with below, but it is reverse.

If you dont want use the specific BGP Speaker anymore, you can use CLI: `neutron bgp-speaker-delete <SPEAKER NAME/ID>`

BGP Plugin will find all the associated Dragent and send RPC `bgp_speaker_remove_end` to make the Dragents to clean the `BGP Speaker` instances. This is the same with CLI: `neutron bgp-dragent-speaker-remove <DRAGENT ID> <SPEAKER NAME/ID>` BGP Plugin just send rpc `bgp_speaker_remove_end` to the specific Dragent.

Advertisement

For details refer to Route Advertisement.

2.2.4 How to work

For details refer to Testing.

2.3 Route Advertisement

2.3.1 BGP

This page discusses the behavior of BGP dynamic routing about how to advertise routes and show the routes details in the project.

BGP dynamic routing could advertise 3 classes of routes:

- Host routes for floating IP addresses hosted on non-DVR routers, as floatingip address set on the router namespace, it knows how to route the message to the correct way, so the next-hop should be the IP address of router gateway port.

- Host routes for floating IP addresses hosted on DVR routers. With DVR-enabled routers, the floating IP can be reached directly on the compute node hosting a given instance. As such, host routes for the floating IP address should advertise the FIP agent gateway on the compute node as the next-hop instead of the centralized router. This will keep inbound floating IP traffic from encountering the bottleneck of the centralized router.
- Prefix routes for directly routable tenant networks with address scopes, the nexthop is the centralized router, the same for DVR and CVR. BGP dynamic routing could advertise tenant network prefixes to physical network devices(routers which support BGP protocol), called this `Prefixes advertisement`.

When distributed virtual routing (DVR) is enabled on a router, next-hops for floating IPs and fixed IPs are not advertised as being at the centralized router. Host routes with the next-hop set to the appropriate compute node are advertised.

Logical Model

```
+--------+ 1     N +---------------------+
| Router |---------|  BgpAdvertisedRoute |
+--------+         +---------------------+
                             | N
                             |
                             | 1
+---------+ N       N +------------+ N       N +---------+
| BgpPeer |-----------| BgpSpeaker |-----------| Network |
+---------+           +------------+           +---------+
                             | N
                             |
                             | 1
                      +--------------+
                      | AddressScope |
                      +--------------+
```

Note

A BGP Speaker only supports one address family to speak BGP. A dual-stack IPv4 and IPv6 network needs two BGP Speakers to advertise the routes with BGP, one for IPv4 and the other for IPv6. So A network can have N number of BGP Speakers bound to it.

BgpAdvertisedRoute represents derived data. As the number of BgpAdvertisedRoutes can be quite large, storing in a database table is not feasible. BgpAdvertisedRoute information can be derived by joining data already available in the Neutron database. And now BGP dynamic routing project process the Bgpadvertiseroutes which should be advertised to external Router is basing on the exist Neutron DB tables. Neutron looks on each of the gateway network for any routers with a gateway port on that network. For each router identified, Neutron locates each floating IP and tenant network accessible through the router gateway port. Neutron then advertises each floating IP and tenant network with the IP address of the router gateway port as the next hop.

When BGP Plugin is started, it will register callbacks. All callbacks are used for processing Floating IP, Router Interface and Router Gateway creation or update, this functions listen the events of these resources for calling Dragent to change the advertisement routes.

Now we just focus on the resources which may cause route change, the following callbacks does this work.

- floatingip_update_callback This function listens to the Floating IPs AFTER_UPDATE event, it judges whether the associated router is changed, and changes the advertisement routes and nexthop based on that.
- router_interface_callback This function listens to the tenants network routes change, it listens to AFTER_CREATE and AFTER_DELETE events of Router Interface resource. It calls Dragent to advertise or stop the prefix routes after a interface attach into a router.
- router_gateway_callback This function listens to the router gateway port creation or deletion. It also focuses on tenants network routes change.

You could get the advertisement routes of specific BGP Speaker like: `neutron bgp-speaker-advertiseroute-list <created-bgp-speaker>` It does a complicated db query to generate the list of advertised routes. For more details refer to route advertisement db lookup

2.4 Agent

Neutron-dynamic-routing implements a new agent named DRAgent. The agent talks to the neutron-dynamic-routing plugin which resides in the neutron server to get routing entity configuration. DRAgent interacts with the back-end driver to realize the required dynamic routing protocol functionality. For details, please refer to the system design document *System Design*

Note

One DRAgent can support multiple drivers but currently ONLY os-ken is integrated successfully.

2.4.1 Scheduler

Neutron-dynamic-routing scheduler, schedules a routing entity to a proper DRAgent.

BGP Scheduler

BGP Speaker and DRAgent has 1:N association which means one BGP speaker can be scheduled on multiple DRAgents.

There are different options for the scheduling algorithm to be used, these can be selected via the `bgp_drscheduler_driver` configuration option.

StaticScheduler

This is the most simple option, which does no automatic scheduling at all. Instead it relies on API requests to explicitly associate BGP speaker with DRAgents and to disassociate them again.

Sample configuration:

```
bgp_drscheduler_driver = neutron_dynamic_routing.services.bgp.scheduler.bgp_
↪dragent_scheduler.StaticScheduler
```

Here is an example to associate/disassociate a BGP Speaker to/from a DRAgent.

```
(neutron) bgp-speaker-list
+--------------------------------------+------+----------+------------+
| id                                   | name | local_as | ip_version |
+--------------------------------------+------+----------+------------+
| 0967eb04-59e5-4ca6-a0b0-d584d8d4a132 | bgp2 |      200 |          4 |
| a73432c3-a3fc-4b1e-9be2-6c32a61df579 | bgp1 |      100 |          4 |
+--------------------------------------+------+----------+------------+

(neutron) agent-list
+--------------------------------------+---------------------------+----------
↪-----------+-------------------+-------+----------------+------------------
↪--------+
| id                                   | agent_type                | host      ␣
↪          | availability_zone | alive | admin_state_up | binary           ␣
↪         |
+--------------------------------------+---------------------------+----------
↪-----------+-------------------+-------+----------------+------------------
↪--------+
| 0c21a829-4fd6-4375-8e65-36db4dc434ac | DHCP agent                | steve-
↪devstack-test | nova              | :-)   | True           | neutron-dhcp-
↪agent        |
| 0f9d6886-910d-4af4-b248-673b22eb9e78 | Metadata agent            | steve-
↪devstack-test |                   | :-)   | True           | neutron-
↪metadata-agent    |
| 5908a304-b9d9-4e8c-a0af-96a066a7c87e | Open vSwitch agent        | steve-
↪devstack-test |                   | :-)   | True           | neutron-
↪openvswitch-agent |
| ae74e375-6a75-4ebe-b85c-6628d2baf02f | L3 agent                  | steve-
↪devstack-test | nova              | :-)   | True           | neutron-l3-
↪agent          |
| dbd9900e-9d16-444d-afc4-8d0035df5ed5 | BGP dynamic routing agent | steve-
↪devstack-test |                   | :-)   | True           | neutron-bgp-
↪dragent       |
+--------------------------------------+---------------------------+----------
↪-----------+-------------------+-------+----------------+------------------
↪--------+

(neutron) bgp-dragent-speaker-add dbd9900e-9d16-444d-afc4-8d0035df5ed5 bgp1
Associated BGP speaker bgp1 to the Dynamic Routing agent.

(neutron) bgp-speaker-list-on-dragent dbd9900e-9d16-444d-afc4-8d0035df5ed5
+--------------------------------------+------+----------+------------+
| id                                   | name | local_as | ip_version |
+--------------------------------------+------+----------+------------+
| a73432c3-a3fc-4b1e-9be2-6c32a61df579 | bgp1 |      100 |          4 |
+--------------------------------------+------+----------+------------+

(neutron) bgp-dragent-speaker-remove dbd9900e-9d16-444d-afc4-8d0035df5ed5 bgp1
Disassociated BGP speaker bgp1 from the Dynamic Routing agent.
```

(continues on next page)

(continued from previous page)

```
(neutron) bgp-speaker-list-on-dragent dbd9900e-9d16-444d-afc4-8d0035df5ed5

(neutron)
```

ReST APIs for neutron-dynamic-routing scheduler are defined as part of the Neutron API reference.

ChanceScheduler

This is the default option. It will automatically schedule newly created BGP speakers to one of the active DRAgents. When a DRAgent goes down, the BGP speaker will be disassociated from it and an attempt is made to schedule it to a different agent. Note that this action will override any manual associations that have been performed via the API, so you will want to use this scheduler only in very basic deployments.

Sample configuration:

```
bgp_drscheduler_driver = neutron_dynamic_routing.services.bgp.scheduler.bgp_
↪dragent_scheduler.ChanceScheduler
```

CHAPTER

THREE

CONFIGURATION GUIDE

3.1 Configuration

This section provides a list of all possible options for each configuration file.

neutron-dynamic-routing uses the following configuration files for its various services.

3.1.1 bgp_dragent.ini

DEFAULT

debug

Type
boolean

Default
`False`

Mutable
This option can be changed without restarting.

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

log_config_append

Type
string

Default
`<None>`

Mutable
This option can be changed without restarting.

The name of a logging configuration file. This file is appended to any existing logging configuration files. For details about logging configuration files, see the Python logging module documentation. Note that when logging configuration files are used then all logging configuration is set in the configuration file and other logging configuration options are ignored (for example, log-date-format).

Table 1: Deprecated Variations

Group	Name
DEFAULT	log-config
DEFAULT	log_config

`log_date_format`

Type
string

Default
`%Y-%m-%d %H:%M:%S`

Defines the format string for %(asctime)s in log records. Default: the value above . This option is ignored if log_config_append is set.

`log_file`

Type
string

Default
`<None>`

(Optional) Name of log file to send logging output to. If no default is set, logging will go to stderr as defined by use_stderr. This option is ignored if log_config_append is set.

Table 2: Deprecated Variations

Group	Name
DEFAULT	logfile

`log_dir`

Type
string

Default
`<None>`

(Optional) The base directory used for relative log_file paths. This option is ignored if log_config_append is set.

Table 3: Deprecated Variations

Group	Name
DEFAULT	logdir

`watch_log_file`

Type
boolean

Default
`False`

Uses logging handler designed to watch file system. When log file is moved or removed this handler will open a new log file with specified path instantaneously. It makes sense only if log_file option is specified and Linux platform is used. This option is ignored if log_config_append is set.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
This function is known to have bene broken for long time, and depends on the unmaintained library

`use_syslog`

Type
boolean

Default
`False`

Use syslog for logging. Existing syslog format is DEPRECATED and will be changed later to honor RFC5424. This option is ignored if log_config_append is set.

`use_journal`

Type
boolean

Default
`False`

Enable journald for logging. If running in a systemd environment you may wish to enable journal support. Doing so will use the journal native protocol which includes structured metadata in addition to log messages.This option is ignored if log_config_append is set.

`syslog_log_facility`

Type
string

Default
`LOG_USER`

Syslog facility to receive log lines. This option is ignored if log_config_append is set.

`use_json`

Type
boolean

Default
`False`

Use JSON formatting for logging. This option is ignored if log_config_append is set.

`use_stderr`

Type
boolean

Default
`False`

Log output to standard error. This option is ignored if log_config_append is set.

`log_color`

Type
boolean

Default
`False`

(Optional) Set the color key according to log levels. This option takes effect only when logging to stderr or stdout is used. This option is ignored if log_config_append is set.

`log_rotate_interval`

Type
integer

Default
`1`

The amount of time before the log files are rotated. This option is ignored unless log_rotation_type is set to interval.

`log_rotate_interval_type`

Type
string

Default
`days`

Valid Values
Seconds, Minutes, Hours, Days, Weekday, Midnight

Rotation interval type. The time of the last file change (or the time when the service was started) is used when scheduling the next rotation.

`max_logfile_count`

Type
integer

Default
`30`

Maximum number of rotated log files.

`max_logfile_size_mb`

Type
integer

Default
`200`

Log file maximum size in MB. This option is ignored if log_rotation_type is not set to size.

`log_rotation_type`

Type
string

Default
`none`

Valid Values
interval, size, none

Log rotation type.

Possible values

interval
Rotate logs at predefined time intervals.

size
Rotate logs once they reach a predefined size.

none
Do not rotate log files.

`logging_context_format_string`

Type
string

Default
`%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [%(global_request_id)s %(request_id)s %(user_identity)s] %(instance)s%(message)s`

Format string to use for log messages with context. Used by oslo_log.formatters.ContextFormatter

`logging_default_format_string`

Type
string

Default
`%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [-] %(instance)s%(message)s`

Format string to use for log messages when context is undefined. Used by oslo_log.formatters.ContextFormatter

`logging_debug_format_suffix`

Type
string

Default
`%(funcName)s %(pathname)s:%(lineno)d`

Additional data to append to log message when logging level for the message is DEBUG. Used by oslo_log.formatters.ContextFormatter

`logging_exception_prefix`

Type
string

Default
`%(asctime)s.%(msecs)03d %(process)d ERROR %(name)s %(instance)s`

Prefix each line of exception output with this format. Used by oslo_log.formatters.ContextFormatter

`logging_user_identity_format`

Type
string

Default
`%(user)s %(project)s %(domain)s %(system_scope)s %(user_domain)s %(project_domain)s`

Defines the format string for %(user_identity)s that is used in logging_context_format_string. Used by oslo_log.formatters.ContextFormatter

`default_log_levels`

Type
list

Default
`['amqp=WARN', 'amqplib=WARN', 'boto=WARN', 'qpid=WARN', 'sqlalchemy=WARN', 'suds=INFO', 'oslo.messaging=INFO', 'oslo_messaging=INFO', 'iso8601=WARN', 'requests.packages.urllib3.connectionpool=WARN', 'urllib3.connectionpool=WARN', 'websocket=WARN', 'requests.packages.urllib3.util.retry=WARN', 'urllib3.util.retry=WARN', 'keystonemiddleware=WARN', 'routes.middleware=WARN', 'stevedore=WARN', 'taskflow=WARN', 'keystoneauth=WARN', 'oslo.cache=INFO', 'oslo_policy=INFO', 'dogpile.core.dogpile=INFO']`

List of package logging levels in logger=LEVEL pairs. This option is ignored if log_config_append is set.

`publish_errors`

Type
boolean

Default
`False`

Enables or disables publication of error events.

`instance_format`

Type
string

Default
`"[instance:  %(uuid)s] "`

The format for an instance that is passed with the log message.

`instance_uuid_format`

Type
string

Default
`"[instance:  %(uuid)s] "`

The format for an instance UUID that is passed with the log message.

`rate_limit_interval`

Type
integer

Default
`0`

Interval, number of seconds, of log rate limiting.

`rate_limit_burst`

Type
integer

Default
`0`

Maximum number of logged messages per rate_limit_interval.

`rate_limit_except_level`

Type
string

Default
`CRITICAL`

Valid Values
CRITICAL, ERROR, INFO, WARNING, DEBUG,

Log level name used by rate limiting. Logs with level greater or equal to rate_limit_except_level are not filtered. An empty string means that all levels are filtered.

`fatal_deprecations`

Type
boolean

Default
`False`

Enables or disables fatal status of deprecations.

bgp

`bgp_speaker_driver`

Type
string

Default
`<None>`

BGP speaker driver class to be instantiated.

`bgp_router_id`

Type
string

Default
`<None>`

32-bit BGP identifier, typically an IPv4 address owned by the system running the BGP DrAgent.

The following are sample configuration files for neutron-dynamic-routing. These are generated from code and reflect the current state of code in the neutron-dynamic-routing repository.

3.1.2 Sample bgp_dragent.ini

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

```
[DEFAULT]

#
# From oslo.log
#

# If set to true, the logging level will be set to DEBUG instead of the␣
↪default
# INFO level. (boolean value)
# Note: This option can be changed without restarting.
#debug = false

# The name of a logging configuration file. This file is appended to any
# existing logging configuration files. For details about logging␣
↪configuration
# files, see the Python logging module documentation. Note that when logging
# configuration files are used then all logging configuration is set in the
# configuration file and other logging configuration options are ignored (for
# example, log-date-format). (string value)
# Note: This option can be changed without restarting.
# Deprecated group/name - [DEFAULT]/log_config
#log_config_append = <None>

# Defines the format string for %%(asctime)s in log records. Default:
# %(default)s . This option is ignored if log_config_append is set. (string
# value)
#log_date_format = %Y-%m-%d %H:%M:%S

# (Optional) Name of log file to send logging output to. If no default is set,
# logging will go to stderr as defined by use_stderr. This option is ignored␣
↪if
# log_config_append is set. (string value)
```

(continues on next page)

(continued from previous page)

```
# Deprecated group/name - [DEFAULT]/logfile
#log_file = <None>

# (Optional) The base directory used for relative log_file  paths. This option
# is ignored if log_config_append is set. (string value)
# Deprecated group/name - [DEFAULT]/logdir
#log_dir = <None>

# DEPRECATED: Uses logging handler designed to watch file system. When log␣
↪file
# is moved or removed this handler will open a new log file with specified␣
↪path
# instantaneously. It makes sense only if log_file option is specified and
# Linux platform is used. This option is ignored if log_config_append is set.
# (boolean value)
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
# Reason: This function is known to have bene broken for long time, and␣
↪depends
# on the unmaintained library
#watch_log_file = false

# Use syslog for logging. Existing syslog format is DEPRECATED and will be
# changed later to honor RFC5424. This option is ignored if log_config_append
# is set. (boolean value)
#use_syslog = false

# Enable journald for logging. If running in a systemd environment you may␣
↪wish
# to enable journal support. Doing so will use the journal native protocol
# which includes structured metadata in addition to log messages.This option␣
↪is
# ignored if log_config_append is set. (boolean value)
#use_journal = false

# Syslog facility to receive log lines. This option is ignored if
# log_config_append is set. (string value)
#syslog_log_facility = LOG_USER

# Use JSON formatting for logging. This option is ignored if log_config_append
# is set. (boolean value)
#use_json = false

# Log output to standard error. This option is ignored if log_config_append is
# set. (boolean value)
#use_stderr = false

# (Optional) Set the 'color' key according to log levels. This option takes
# effect only when logging to stderr or stdout is used. This option is ignored
```

(continues on next page)

(continued from previous page)

```
# if log_config_append is set. (boolean value)
#log_color = false

# The amount of time before the log files are rotated. This option is ignored
# unless log_rotation_type is set to "interval". (integer value)
#log_rotate_interval = 1

# Rotation interval type. The time of the last file change (or the time when
# the service was started) is used when scheduling the next rotation. (string
# value)
# Possible values:
# Seconds - <No description provided>
# Minutes - <No description provided>
# Hours - <No description provided>
# Days - <No description provided>
# Weekday - <No description provided>
# Midnight - <No description provided>
#log_rotate_interval_type = days

# Maximum number of rotated log files. (integer value)
#max_logfile_count = 30

# Log file maximum size in MB. This option is ignored if "log_rotation_type"␣
↪is
# not set to "size". (integer value)
#max_logfile_size_mb = 200

# Log rotation type. (string value)
# Possible values:
# interval - Rotate logs at predefined time intervals.
# size - Rotate logs once they reach a predefined size.
# none - Do not rotate log files.
#log_rotation_type = none

# Format string to use for log messages with context. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_context_format_string = %(asctime)s.%(msecs)03d %(process)d
↪%(levelname)s %(name)s [%(global_request_id)s %(request_id)s %(user_
↪identity)s] %(instance)s%(message)s

# Format string to use for log messages when context is undefined. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_default_format_string = %(asctime)s.%(msecs)03d %(process)d
↪%(levelname)s %(name)s [-] %(instance)s%(message)s

# Additional data to append to log message when logging level for the message
# is DEBUG. Used by oslo_log.formatters.ContextFormatter (string value)
#logging_debug_format_suffix = %(funcName)s %(pathname)s:%(lineno)d
```

(continues on next page)

(continued from previous page)

```
# Prefix each line of exception output with this format. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_exception_prefix = %(asctime)s.%(msecs)03d %(process)d ERROR
↪%(name)s %(instance)s

# Defines the format string for %(user_identity)s that is used in
# logging_context_format_string. Used by oslo_log.formatters.ContextFormatter
# (string value)
#logging_user_identity_format = %(user)s %(project)s %(domain)s %(system_
↪scope)s %(user_domain)s %(project_domain)s

# List of package logging levels in logger=LEVEL pairs. This option is ignored
# if log_config_append is set. (list value)
#default_log_levels = amqp=WARN,amqplib=WARN,boto=WARN,qpid=WARN,
↪sqlalchemy=WARN,suds=INFO,oslo.messaging=INFO,oslo_messaging=INFO,
↪iso8601=WARN,requests.packages.urllib3.connectionpool=WARN,urllib3.
↪connectionpool=WARN,websocket=WARN,requests.packages.urllib3.util.
↪retry=WARN,urllib3.util.retry=WARN,keystonemiddleware=WARN,routes.
↪middleware=WARN,stevedore=WARN,taskflow=WARN,keystoneauth=WARN,oslo.
↪cache=INFO,oslo_policy=INFO,dogpile.core.dogpile=INFO

# Enables or disables publication of error events. (boolean value)
#publish_errors = false

# The format for an instance that is passed with the log message. (string
# value)
#instance_format = "[instance: %(uuid)s] "

# The format for an instance UUID that is passed with the log message. (string
# value)
#instance_uuid_format = "[instance: %(uuid)s] "

# Interval, number of seconds, of log rate limiting. (integer value)
#rate_limit_interval = 0

# Maximum number of logged messages per rate_limit_interval. (integer value)
#rate_limit_burst = 0

# Log level name used by rate limiting. Logs with level greater or equal to
# rate_limit_except_level are not filtered. An empty string means that all
# levels are filtered. (string value)
# Possible values:
# CRITICAL - <No description provided>
# ERROR - <No description provided>
# INFO - <No description provided>
# WARNING - <No description provided>
# DEBUG - <No description provided>
# '' - <No description provided>
#rate_limit_except_level = CRITICAL
```

(continues on next page)

(continued from previous page)

```
# Enables or disables fatal status of deprecations. (boolean value)
#fatal_deprecations = false


[bgp]

#
# From bgp.agent
#

# BGP speaker driver class to be instantiated. (string value)
#bgp_speaker_driver = <None>

# 32-bit BGP identifier, typically an IPv4 address owned by the system running
# the BGP DrAgent. (string value)
#bgp_router_id = <None>
```

3.2 Policy

neutron-dynamic-routing, like most OpenStack projects, uses a policy language to restrict permissions on REST API actions.

3.2.1 neutron-dynamic-routing policies

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

neutron-dynamic-routing

`create_bgp_speaker`

Default
`rule:admin_only`

Operations

- **POST** `/bgp-speakers`

Create a BGP speaker

`update_bgp_speaker`

Default
`rule:admin_only`

Operations

- **PUT** `/bgp-speakers/{id}`

Update a BGP speaker

`delete_bgp_speaker`

Default
`rule:admin_only`

Operations

- **DELETE** `/bgp-speakers/{id}`

Delete a BGP speaker

`get_bgp_speaker`

Default
`rule:admin_only`

Operations

- **GET** `/bgp-speakers`
- **GET** `/bgp-speakers/{id}`

Get BGP speakers

`add_bgp_peer`

Default
`rule:admin_only`

Operations

- **PUT** `/bgp-speakers/{id}/add_bgp_peer`

Add a BGP peer to a BGP speaker

`remove_bgp_peer`

Default
`rule:admin_only`

Operations

- **PUT** `/bgp-speakers/{id}/remove_bgp_peer`

Remove a BGP peer from a BGP speaker

`add_gateway_network`

Default
`rule:admin_only`

Operations

- **PUT** `/bgp-speakers/{id}/add_gateway_network`

Add a gateway network to a BGP speaker

`remove_gateway_network`

Default
`rule:admin_only`

Operations

- **PUT** `/bgp-speakers/{id}/remove_gateway_network`

Remove a gateway network from a BGP speaker

`get_advertised_routes`

Default
`rule:admin_only`

Operations

- **GET** `/bgp-speakers/{id}/get_advertised_routes`

Get advertised routes of a BGP speaker

`create_bgp_peer`

Default
`rule:admin_only`

Operations

- **POST** `/bgp-peers`

Create a BGP peer

`update_bgp_peer`

Default
`rule:admin_only`

Operations

- **PUT** `/bgp-peers/{id}`

Update a BGP peer

`delete_bgp_peer`

Default
`rule:admin_only`

Operations

- **DELETE** `/bgp-peers/{id}`

Delete a BGP peer

`get_bgp_peer`

Default
`rule:admin_only`

Operations

- **GET** `/bgp-peers`
- **GET** `/bgp-peers/{id}`

Get BGP peers

`add_bgp_speaker_to_dragent`

Default
`rule:admin_only`

Operations

- **POST** `/agents/{agent_id}/bgp-drinstances`

Add a BGP speaker to a dynamic routing agent `remove_bgp_speaker_from_dragent`

Default
`rule:admin_only`

Operations

- **DELETE** `/agents/{agent_id}/bgp-drinstances/{bgp_speaker_id}`

Remove a BGP speaker from a dynamic routing agent

`list_bgp_speaker_on_dragent`

Default
`rule:admin_only`

Operations

- **GET** `/agents/{agent_id}/bgp-drinstances`

List BGP speakers hosted by a dynamic routing agent

`list_dragent_hosting_bgp_speaker`

Default
`rule:admin_only`

Operations

- **GET** `/bgp-speakers/{bgp_speaker_id}/bgp-dragents`

List dynamic routing agents hosting a BGP speaker

3.2.2 Sample neutron-dynamic-routing Policy File

The following is a sample neutron-dynamic-routing 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 neutron-dynamic-routing when this documentation is built. You must ensure your version of neutron-dynamic-routing matches the version of this documentation.

```
# Create a BGP speaker
# POST  /bgp-speakers
#"create_bgp_speaker": "rule:admin_only"

# Update a BGP speaker
# PUT  /bgp-speakers/{id}
#"update_bgp_speaker": "rule:admin_only"

# Delete a BGP speaker
# DELETE  /bgp-speakers/{id}
#"delete_bgp_speaker": "rule:admin_only"
```

(continues on next page)

(continued from previous page)

```
# Get BGP speakers
# GET  /bgp-speakers
# GET  /bgp-speakers/{id}
#"get_bgp_speaker": "rule:admin_only"

# Add a BGP peer to a BGP speaker
# PUT  /bgp-speakers/{id}/add_bgp_peer
#"add_bgp_peer": "rule:admin_only"

# Remove a BGP peer from a BGP speaker
# PUT  /bgp-speakers/{id}/remove_bgp_peer
#"remove_bgp_peer": "rule:admin_only"

# Add a gateway network to a BGP speaker
# PUT  /bgp-speakers/{id}/add_gateway_network
#"add_gateway_network": "rule:admin_only"

# Remove a gateway network from a BGP speaker
# PUT  /bgp-speakers/{id}/remove_gateway_network
#"remove_gateway_network": "rule:admin_only"

# Get advertised routes of a BGP speaker
# GET  /bgp-speakers/{id}/get_advertised_routes
#"get_advertised_routes": "rule:admin_only"

# Create a BGP peer
# POST  /bgp-peers
#"create_bgp_peer": "rule:admin_only"

# Update a BGP peer
# PUT  /bgp-peers/{id}
#"update_bgp_peer": "rule:admin_only"

# Delete a BGP peer
# DELETE  /bgp-peers/{id}
#"delete_bgp_peer": "rule:admin_only"

# Get BGP peers
# GET  /bgp-peers
# GET  /bgp-peers/{id}
#"get_bgp_peer": "rule:admin_only"

# Add a BGP speaker to a dynamic routing agent
# POST  /agents/{agent_id}/bgp-drinstances
#"add_bgp_speaker_to_dragent": "rule:admin_only"

# Remove a BGP speaker from a dynamic routing agent
# DELETE  /agents/{agent_id}/bgp-drinstances/{bgp_speaker_id}
```

(continues on next page)

(continued from previous page)

```
#"remove_bgp_speaker_from_dragent": "rule:admin_only"

# List BGP speakers hosted by a dynamic routing agent
# GET  /agents/{agent_id}/bgp-drinstances
#"list_bgp_speaker_on_dragent": "rule:admin_only"

# List dynamic routing agents hosting a BGP speaker
# GET  /bgp-speakers/{bgp_speaker_id}/bgp-dragents
#"list_dragent_hosting_bgp_speaker": "rule:admin_only"
```

CHAPTER FOUR

API

The reference of the OpenStack neutron-dynamic-routing API is found at https://docs.openstack.org/api-ref/network/#bgp-dynamic-routing.

CHAPTER

FIVE

COMMAND-LINE INTERFACE

Neutron client has provided the command-line interfaces (CLI) to realize dynamic routing services supported by neutron-dynamic-routing project.

Current implementation only supports the command line interfaces for BGP functionality. For query on what specific **neutron bgp** commands are supported, enter:

```
$ neutron help | grep bgp
```

5.1 BGP Peer

5.1.1 BGP Peer Create

```
usage: neutron bgp-peer-create [-h]
                               [-f {html,json,json,shell,table,value,yaml,
↪yaml}]
                               [-c COLUMN] [--max-width <integer>]
                               [--noindent] [--prefix PREFIX]
                               [--request-format {json}]
                               [--tenant-id TENANT_ID] --peer-ip
                               PEER_IP_ADDRESS --remote-as PEER_REMOTE_AS
                               [--auth-type PEER_AUTH_TYPE]
                               [--password AUTH_PASSWORD]
                               NAME
```

Create a BGP Peer.

Positional arguments:

NAME
    Name of the BGP peer to create

--peer-ip PEER_IP_ADDRESS
    Peer IP address.

--remote-as PEER_REMOTE_AS
    Peer AS number. (Integer in [1, 65535] is allowed.)

Optional arguments:

-h, --help
    show this help message and exit

--auth-type PEER_AUTH_TYPE
    Authentication algorithm. Supported algorithms: none(default), md5

--password AUTH_PASSWORD
    Authentication password.

5.1.2 BGP Peer List

```
usage: neutron bgp-peer-list [-h]
                             [-f {csv,html,json,json,table,value,yaml,yaml}]
                             [-c COLUMN] [--max-width <integer>] [--noindent]
                             [--quote {all,minimal,none,nonnumeric}]
                             [--request-format {json}] [-D] [-F FIELD]
                             [-P SIZE] [--sort-key FIELD]
                             [--sort-dir {asc,desc}]
```

List BGP peers.

Optional arguments:

-h, --help
    show this help message and exit

-D, --show-details
    Show detailed information.

-F FIELD, --field FIELD
    Specify the field(s) to be returned by server. You can repeat this option.

5.1.3 BGP Peer Show

```
usage: neutron bgp-peer-show [-h]
                             [-f {html,json,json,shell,table,value,yaml,yaml}]
                             [-c COLUMN] [--max-width <integer>] [--noindent]
                             [--prefix PREFIX] [--request-format {json}] [-D]
                             [-F FIELD]
                             BGP_PEER
```

Show information of a given BGP peer.

Positional arguments:

BGP_PEER
    ID or name of the BGP peer to look up.

Optional arguments:

-h, --help
    show this help message and exit

-D, --show-details
    Show detailed information.

-F FIELD, --field FIELD
    Specify the field(s) to be returned by server. You can repeat this option.

5.1.4 BGP Peer Delete

```
usage: neutron bgp-peer-delete [-h] [--request-format {json}] BGP_PEER
```

Delete a BGP peer.

Positional arguments:

BGP_PEER
ID or name of the BGP peer to delete.

Optional arguments:

-h, --help
show this help message and exit

5.1.5 BGP Peer Update

```
usage: neutron bgp-peer-update [-h] [--request-format {json}] [--name NAME]
                               [--password AUTH_PASSWORD]
                               BGP_PEER
```

Update BGP Peers information.

Positional arguments:

BGP_PEER
ID or name of the BGP peer to update.

Optional arguments:

-h, --help
show this help message and exit

--name NAME
Updated name of the BGP peer.

--password AUTH_PASSWORD
Updated authentication password.

5.1.6 Add Peer to BGP Speaker

```
usage: neutron bgp-speaker-peer-add [-h] [--request-format {json}]
                                    BGP_SPEAKER BGP_PEER
```

Add a peer to the BGP speaker.

Positional arguments:

BGP_SPEAKER
ID or name of the BGP speaker.

BGP_PEER
ID or name of the BGP peer to add.

Optional arguments:

-h, --help
show this help message and exit

5.1.7 Delete Peer from BGP Speaker

```
usage: neutron bgp-speaker-peer-remove [-h] [--request-format {json}]
                                       BGP_SPEAKER BGP_PEER
```

Remove a peer from the BGP speaker.

Positional arguments:

BGP_SPEAKER
    ID or name of the BGP speaker.

BGP_PEER
    ID or name of the BGP peer to remove.

Optional arguments:

-h, --help
    show this help message and exit

5.2 BGP Speaker

5.2.1 BGP Speaker Create

```
usage: neutron bgp-speaker-create [-h]
                                  [-f {html,json,json,shell,table,value,yaml,
↪yaml}]
                                  [-c COLUMN] [--max-width <integer>]
                                  [--noindent] [--prefix PREFIX]
                                  [--request-format {json}]
                                  [--tenant-id TENANT_ID] --local-as LOCAL_AS
                                  [--ip-version {4,6}]
                                  [--advertise-floating-ip-host-routes {True,
↪False}]
                                  [--advertise-tenant-networks {True,False}]
                                  NAME
```

Create a BGP Speaker with a specified NAME.

Positional arguments:

NAME
    Name of the BGP speaker to create.

Optional arguments:

-h, --help
    show this help message and exit

--local-as LOCAL_AS
    Local AS number. (Integer in [1, 65535] is allowed.)

--ip-version {4,6}
    IP version for the BGP speaker (default is 4)

--advertise-floating-ip-host-routes {True,False}
    Whether to enable or disable the advertisement of floating-ip host routes by the BGP speaker. By

default floating ip host routes will be advertised by the BGP speaker.

--advertise-tenant-networks {True,False}
Whether to enable or disable the advertisement of tenant network routes by the BGP speaker. By default tenant network routes will be advertised by the BGP speaker.

5.2.2 BGP Speaker List

```
usage: neutron bgp-speaker-list [-h]
                                [-f {csv,html,json,json,table,value,yaml,yaml}
↪]
                                [-c COLUMN] [--max-width <integer>]
                                [--noindent]
                                [--quote {all,minimal,none,nonnumeric}]
                                [--request-format {json}] [-D] [-F FIELD]
                                [-P SIZE] [--sort-key FIELD]
                                [--sort-dir {asc,desc}]
```

List BGP speakers.

Optional arguments:

-h, --help
show this help message and exit

-D, --show-details
Show detailed information.

-F FIELD, --field FIELD
Specify the field(s) to be returned by server. You can repeat this option.

5.2.3 BGP Speaker Show

```
usage: neutron bgp-speaker-show [-h]
                                [-f {html,json,json,shell,table,value,yaml,
↪yaml}]
                                [-c COLUMN] [--max-width <integer>]
                                [--noindent] [--prefix PREFIX]
                                [--request-format {json}] [-D] [-F FIELD]
                                BGP_SPEAKER
```

Show information of a given BGP speaker.

Positional arguments:

BGP_SPEAKER
ID or name of the BGP speaker to look up.

Optional arguments:

-h, --help
show this help message and exit

-D, --show-details
Show detailed information.

-F FIELD, --field FIELD
Specify the field(s) to be returned by server. You can repeat this option.

5.2.4 BGP Speaker Delete

```
usage: neutron bgp-speaker-delete [-h] [--request-format {json}] BGP_SPEAKER
```

Delete a BGP speaker.

Positional arguments:

BGP_SPEAKER
ID or name of the BGP speaker to delete.

Optional arguments:

-h, --help
show this help message and exit

5.2.5 BGP Speaker Update

```
usage: neutron bgp-speaker-update [-h] [--request-format {json}] [--name NAME]
                                  [--advertise-floating-ip-host-routes {True,
↪False}]
                                  [--advertise-tenant-networks {True,False}]
                                  BGP_SPEAKER
```

Update BGP Speakers information.

Positional arguments:

BGP_SPEAKER
ID or name of the BGP speaker to update.

Optional arguments:

-h, --help
show this help message and exit

--name NAME
Name of the BGP speaker to update.

--advertise-floating-ip-host-routes {True,False}
Whether to enable or disable the advertisement of floating-ip host routes by the BGP speaker. By default floating ip host routes will be advertised by the BGP speaker.

--advertise-tenant-networks {True,False}
Whether to enable or disable the advertisement of tenant network routes by the BGP speaker. By default tenant network routes will be advertised by the BGP speaker.

5.2.6 Add Network to BGP Speaker

```
usage: neutron bgp-speaker-network-add [-h] [--request-format {json}]
                                       BGP_SPEAKER NETWORK
```

Add a network to the BGP speaker.

Positional arguments:

`BGP_SPEAKER`
: ID or name of the BGP speaker.

`NETWORK`
: ID or name of the network to add.

Optional arguments:

-h, --help
: show this help message and exit

5.2.7 Delete Network from BGP Speaker

```
usage: neutron bgp-speaker-network-remove [-h] [--request-format {json}]
                                          BGP_SPEAKER NETWORK
```

Remove a network from the BGP speaker.

Positional arguments:

`BGP_SPEAKER`
: ID or name of the BGP speaker.

`NETWORK`
: ID or name of the network to remove.

Optional arguments:

-h, --help
: show this help message and exit

5.2.8 BGP Advertised Routes List

```
usage: neutron bgp-speaker-advertiseroute-list [-h]
                                               [-f {csv,html,json,json,table,
↪value,yaml,yaml}]
                                               [-c COLUMN]
                                               [--max-width <integer>]
                                               [--noindent]
                                               [--quote {all,minimal,none,
↪nonnumeric}]
                                               [--request-format {json}] [-D]
                                               [-F FIELD] [-P SIZE]
                                               [--sort-key FIELD]
                                               [--sort-dir {asc,desc}]
                                               BGP_SPEAKER
```

List routes advertised by a given BGP speaker.

Positional arguments:

`BGP_SPEAKER`
: ID or name of the BGP speaker.

Optional arguments:

-h, --help
: show this help message and exit

-D, --show-details
Show detailed information.

-F FIELD, --field FIELD
Specify the field(s) to be returned by server. You can repeat this option.

5.3 Dynamic Routing Agent

5.3.1 Add BGP Speaker to Dynamic Routing Agent

```
usage: neutron bgp-dragent-speaker-add [-h] [--request-format {json}]
                                       BGP_DRAGENT_ID BGP_SPEAKER
```

Add a BGP speaker to a Dynamic Routing agent.

Positional arguments:

BGP_DRAGENT_ID
ID of the Dynamic Routing agent.

BGP_SPEAKER
ID or name of the BGP speaker.

Optional arguments:

-h, --help
show this help message and exit

5.3.2 Delete BGP Speaker from Dynamic Routing Agent

```
usage: neutron bgp-dragent-speaker-remove [-h] [--request-format {json}]
                                          BGP_DRAGENT_ID BGP_SPEAKER
```

Removes a BGP speaker from a Dynamic Routing agent.

Positional arguments:

BGP_DRAGENT_ID
ID of the Dynamic Routing agent.

BGP_SPEAKER
ID or name of the BGP speaker.

Optional arguments:

-h, --help
show this help message and exit

5.3.3 List BGP Speakers hosted by a Dynamic Routing Agent

```
usage: neutron bgp-speaker-list-on-dragent [-h]
                                           [-f {csv,html,json,json,table,
↪value,yaml,yaml}]
                                           [-c COLUMN] [--max-width <integer>]
                                           [--noindent]
                                           [--quote {all,minimal,none,
```

(continues on next page)

(continued from previous page)

```
↪nonnumeric}]
                                                [--request-format {json}] [-D]
                                                [-F FIELD]
                                                BGP_DRAGENT_ID
```

List BGP speakers hosted by a Dynamic Routing agent.

Positional arguments:

BGP_DRAGENT_ID
ID of the Dynamic Routing agent.

Optional arguments:

-h, --help
show this help message and exit

-D, --show-details
Show detailed information.

-F FIELD, --field FIELD
Specify the field(s) to be returned by server. You can repeat this option.

5.3.4 List Dynamic Routing Agents Hosting a BGP Speaker

```
usage: neutron bgp-dragent-list-hosting-speaker [-h]
                                                      [-f {csv,html,json,json,table,
↪value,yaml,yaml}]
                                                      [-c COLUMN]
                                                      [--max-width <integer>]
                                                      [--noindent]
                                                      [--quote {all,minimal,none,
↪nonnumeric}]
                                                      [--request-format {json}] [-D]
                                                      [-F FIELD]
                                                      BGP_SPEAKER
```

List Dynamic Routing agents hosting a BGP speaker.

Positional arguments:

BGP_SPEAKER
ID or name of the BGP speaker.

Optional arguments:

-h, --help
show this help message and exit

-D, --show-details
Show detailed information.

-F FIELD, --field FIELD
Specify the field(s) to be returned by server. You can repeat this option.

CHAPTER

SIX

DEVELOPER GUIDE

In the Developer Guide, you will find information on neutron-dynamic-routing lower level programming APIs. There are sections that cover the core pieces of neutron-dynamic-routing, including its API, command-lines, database, system-design, alembic-migration etc. There are also subsections that describe specific drivers inside neutron-dynamic-routing. Finally, the developer guide includes information about testing and supported functionalities as well. This documentation is generated by the Sphinx toolkit and lives in the source tree.

6.1 Contributing

Please see the Neutron CONTRIBUTING.rst file for how to contribute to neutron-dynamic-routing:

Neutron CONTRIBUTING.rst

6.2 Testing

Dynamic routing enables advertisement of self-service network prefixes to physical network devices that support a dynamic routing protocol, such as routers. The Neutron dynamic routing project consists of a service plugin-in and an agent that can advertise Neutron private network to outside of OpenStack. This document will describe how to test the Dynamic Routing functionalities, introduce what the environment architecture is for dynamic routing test and show how to setup dynamic routing environment using Devstack.

6.2.1 Environment Architecture

Use the following example architecture as a test environment to deploy neutron-dynamic-routing in your environment. The example architecture will deploy an all-in-one OpenStack and connect to an Ubuntu VM running Quagga as a router outside of OpenStack. See following:

```
                                                    +---------------+
                                       10.156.18.20 |               |
                 +----------------------------------|   Quagga      |
                 | BGP Peering Session              |   Router      |
                 |                                  |  172.24.4.3   |
                 |                                  +---------------+
                 |                                          |
                 |10.156.18.21                              |      External␣
↪Network(172.24.4.0/24)
------------------------------------------------------------------------------
↪-------------------
```

(continues on next page)

(continued from previous page)

```
                 |eth0
        +---------------------------------------+
        |        |                              |
        |        |        br-ex                 |
        |        +----------------+             |
        |        |                |172.24.4.1   |
        |  +------------+     +-------+         |
        |  |            |     |Router |         |
        |  |  Dr-Agent  |     |       |         |
        |  |            |     +-------+         |
        |  +------------+         |             |
        |                  ----------------     |
        |                  Tenant Network       |
        |                   (10.0.0.0/24)       |
        |                                       |
        +---------------------------------------+
        All-In-One OpenStack Installation
```

6.2.2 Devstack Setup

1. Download devstack:

```
git clone https://opendev.org/openstack/devstack.git
```

2. Enable neutron-dynamic-routing by including this in your local.conf file:

```
[[local|localrc]]
enable_plugin neutron-dynamic-routing https://opendev.org/openstack/
↪neutron-dynamic-routing
```

3. Run devstack:

```
./stack.sh
```

6.2.3 Quagga Configure

Quagga is a network routing software available in most GNU/Linux, Solaris, FreeBSD, and NetBSD. It provides the implementation of OSPF, RIP, BGP and IS-IS. This section shows you how to install Quagga and then configure it on Ubuntu Linux.

1. Install Quagga using apt-get:

```
$ sudo apt-get install quagga quagga-doc
```

2. Create an empty file (/etc/quagga/zebra.conf) and set permissions.

   The Quagga files and configurations will be stored in /etc/quagga:

```
$ sudo touch /etc/quagga/zebra.conf
$ sudo chown quagga.quagga /etc/quagga/zebra.conf
$ sudo chmod 640 /etc/quagga/zebra.conf
```

3. Update quagga daemon file.

   You can enable/disable the daemons routing in the /etc/quagga/daemons file. Update /etc/quagga/daemons to enable zebra and bgp:

```
zebra=yes
bgpd=yes
ospfd=no
ospf6d=no
ripd=no
ripngd=no
isisd=no
```

4. Update /etc/quagga/zebra.conf:

```
# Zebra configuration
# name of the router
hostname quagga_1
password zebra

# log
log file /var/log/quagga/zebra.log
```

5. Update /etc/quagga/bgpd.conf:

```
# declare a router with local-as 1000
router bgp 1000

# set router-id to the network address we announce
bgp router-id 10.156.18.20

# expose neighbor network which dynamic routing agent is using
neighbor 10.156.18.21 remote-as 12345

# treat neutron dynamic routing agent as a passive peer in case
# quagga keeps making futile connection attempts
neighbor 10.156.18.21 passive

# log
log file /var/log/quagga/bgpd.log

debug bgp events
debug bgp filters
debug bgp fsm
debug bgp keepalives
debug bgp updates
```

6. Restart the Quagga daemon:

```
$ sudo systemcl restart bgpd
```

6.2.4 Service Test

1. As the dynamic routing is only supported by admin, source the devstack admin credentials:

```
$ . devstack/openrc admin admin
```

2. Verify that the neutron dynamic routing agent is running.

```
$ openstack network agent list --agent-type bgp
+---------------------+---------------------+----------------------+-
↪-------------------+-------+-------+---------------------+
| ID                  | Agent Type          | Host                 |␣
↪Availability Zone | Alive | State | Binary              |
+---------------------+---------------------+----------------------+-
↪-------------------+-------+-------+---------------------+
| 69ad386f-e055-4284 | BGP dynamic         | devstack-bgp-dr      |␣
↪                   | :-)   | UP    | neutron-bgp-dragent |
| -8c8e-ef9bd540705c | routing agent       |                      |␣
↪                   |       |       |                     |
+---------------------+---------------------+----------------------+-
↪-------------------+-------+-------+---------------------+
```

3. Create an address scope.

   The provider(external) and tenant networks must belong to the same address scope for the agent to advertise those tenant network prefixes.

```
$ openstack address scope create --ip-version 4 --share public
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| id         | c02c358a-9d35-43ea-8313-986b3e4a91c0 |
| ip_version | 4                                    |
| name       | public                               |
| project_id | b3ac05ef10bf441fbf4aa17f16ae1e6d     |
| shared     | True                                 |
+------------+--------------------------------------+
```

4. Create subnet pools. The provider and tenant networks use different pools.

   - Create the provider network pool.

```
$ openstack subnet pool create --pool-prefix 172.24.4.0/24 \
  --address-scope public provider
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| address_scope_id  | 18f74828-5f38-4d84-b030-ed642f2157c5 |
| created_at        | 2020-08-28T15:12:11Z                 |
| default_prefixlen | 8                                    |
| default_quota     | None                                 |
| description       |                                      |
| id                | d812a10e-5981-4686-90c4-d6fff454b38a |
```

(continues on next page)

(continued from previous page)

```
| ip_version        | 4                                    |
| is_default        | False                                |
| max_prefixlen     | 32                                   |
| min_prefixlen     | 8                                    |
| name              | provider                             |
| prefixes          | 172.24.4.0/24                        |
| project_id        | 17c884da94bc4259b20ace3da6897297     |
| revision_number   | 0                                    |
| shared            | False                                |
| tags              |                                      |
| updated_at        | 2020-08-28T15:12:11Z                 |
+-------------------+--------------------------------------+
```

- Create tenant network pool.

```
$ openstack subnet pool create --pool-prefix 10.0.0.0/16 \
  --address-scope public --share selfservice
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| address_scope_id  | 18f74828-5f38-4d84-b030-ed642f2157c5 |
| created_at        | 2020-08-28T15:15:31Z                 |
| default_prefixlen | 8                                    |
| default_quota     | None                                 |
| description       |                                      |
| id                | 8b9d1c9b-6aba-416f-8d10-1e7a0f6052f6 |
| ip_version        | 4                                    |
| is_default        | False                                |
| max_prefixlen     | 32                                   |
| min_prefixlen     | 8                                    |
| name              | selfservice                          |
| prefixes          | 10.0.0.0/16                          |
| project_id        | 17c884da94bc4259b20ace3da6897297     |
| revision_number   | 0                                    |
| shared            | True                                 |
| tags              |                                      |
| updated_at        | 2020-08-28T15:15:31Z                 |
+-------------------+--------------------------------------+
```

5. Create the provider and tenant networks.

   - Create the provider network.

```
$ openstack network create --external --provider-network-type␣
↪flat \
  --provider-physical-network public provider
+---------------------------+-----------------------------------
↪--+
| Field                     | Value                            ␣
↪  |
```

(continues on next page)

(continued from previous page)

```
+---------------------------+-----------------------------------
↪--+
| admin_state_up            | UP                                ␣
↪  |
| availability_zone_hints   |                                   ␣
↪  |
| availability_zones        |                                   ␣
↪  |
| created_at                | 2020-08-28T15:24:07Z              ␣
↪  |
| description               |                                   ␣
↪  |
| dns_domain                |                                   ␣
↪  |
| id                        | 18f9a9c0-f8f5-4360-822a-
↪b687c1008bf7 |
| ipv4_address_scope        | None                              ␣
↪  |
| ipv6_address_scope        | None                              ␣
↪  |
| is_default                | False                             ␣
↪  |
| is_vlan_transparent       | None                              ␣
↪  |
| mtu                       | 1500                              ␣
↪  |
| name                      | provider                          ␣
↪  |
| port_security_enabled     | True                              ␣
↪  |
| project_id                | 17c884da94bc4259b20ace3da6897297  ␣
↪  |
| provider:network_type     | flat                              ␣
↪  |
| provider:physical_network | public                            ␣
↪  |
| provider:segmentation_id  | None                              ␣
↪  |
| qos_policy_id             | None                              ␣
↪  |
| revision_number           | 1                                 ␣
↪  |
| router:external           | External                          ␣
↪  |
| segments                  | None                              ␣
↪  |
| shared                    | False                             ␣
↪  |
| status                    | ACTIVE                            ␣
```

(continues on next page)

(continued from previous page)

```
↪  |
| subnets                  |                                   ␣
↪  |
| tags                     |                                   ␣
↪  |
| updated_at               | 2020-08-28T15:24:07Z              ␣
↪  |
+--------------------------+-----------------------------------
↪--+
```

- Create a subnet on the provider network using an IP address allocation from the provider subnet pool.

```
$ openstack subnet create --network provider --subnet-pool␣
↪provider \
  --prefix-length 24 provider
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| allocation_pools     | 172.24.4.2-172.24.4.254              |
| cidr                 | 172.24.4.0/24                        |
| created_at           | 2020-08-28T15:27:00Z                 |
| description          |                                      |
| dns_nameservers      |                                      |
| dns_publish_fixed_ip | False                                |
| enable_dhcp          | True                                 |
| gateway_ip           | 172.24.4.1                           |
| host_routes          |                                      |
| id                   | 4ed8ac88-2c19-4f94-9362-7b301e743438 |
| ip_version           | 4                                    |
| ipv6_address_mode    | None                                 |
| ipv6_ra_mode         | None                                 |
| name                 | provider                             |
| network_id           | 18f9a9c0-f8f5-4360-822a-b687c1008bf7 |
| prefix_length        | 24                                   |
| project_id           | 17c884da94bc4259b20ace3da6897297     |
| revision_number      | 0                                    |
| segment_id           | None                                 |
| service_types        |                                      |
| subnetpool_id        | a8fecc3d-a489-46ca-87fb-dff4e3371503 |
| tags                 |                                      |
| updated_at           | 2020-08-28T15:27:00Z                 |
+----------------------+--------------------------------------+
```

- Create the tenant network.

```
$ openstack network create private
+--------------------------+-----------------------------------
↪--+
| Field                    | Value                             ␣
```

(continues on next page)

(continued from previous page)

```
↪   |
+---------------------------+-----------------------------------
↪--+
| admin_state_up            | UP                                ␣
↪   |
| availability_zone_hints   |                                   ␣
↪   |
| availability_zones        |                                   ␣
↪   |
| created_at                | 2020-08-28T15:28:06Z              ␣
↪   |
| description               |                                   ␣
↪   |
| dns_domain                |                                   ␣
↪   |
| id                        | 43643543-6edb-4c2b-a087-
↪4553b75b6799 |
| ipv4_address_scope        | None                              ␣
↪   |
| ipv6_address_scope        | None                              ␣
↪   |
| is_default                | False                             ␣
↪   |
| is_vlan_transparent       | None                              ␣
↪   |
| mtu                       | 1442                              ␣
↪   |
| name                      | private                           ␣
↪   |
| port_security_enabled     | True                              ␣
↪   |
| project_id                | 17c884da94bc4259b20ace3da6897297  ␣
↪   |
| provider:network_type     | geneve                            ␣
↪   |
| provider:physical_network | None                              ␣
↪   |
| provider:segmentation_id  | 1                                 ␣
↪   |
| qos_policy_id             | None                              ␣
↪   |
| revision_number           | 1                                 ␣
↪   |
| router:external           | Internal                          ␣
↪   |
| segments                  | None                              ␣
↪   |
| shared                    | False                             ␣
↪   |
```

(continues on next page)

(continued from previous page)

```
| status                     | ACTIVE                                ␣
↪   |
| subnets                    |                                       ␣
↪   |
| tags                       |                                       ␣
↪   |
| updated_at                 | 2020-08-28T15:28:06Z                  ␣
↪   |
+----------------------------+-------------------------------------
↪--+
```

- Create a subnet on the tenant network using an IP address allocation from the private subnet pool.

```
$ openstack subnet create --network private --subnet-pool␣
↪selfservice \
  --prefix-length 24 selfservice
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| allocation_pools      | 10.0.0.2-10.0.0.254                  |
| cidr                  | 10.0.0.0/24                          |
| created_at            | 2020-08-28T15:29:20Z                 |
| description           |                                      |
| dns_nameservers       |                                      |
| dns_publish_fixed_ip  | False                                |
| enable_dhcp           | True                                 |
| gateway_ip            | 10.0.0.1                             |
| host_routes           |                                      |
| id                    | 12eec8cb-8303-4829-8b16-e9a75072fcb0 |
| ip_version            | 4                                    |
| ipv6_address_mode     | None                                 |
| ipv6_ra_mode          | None                                 |
| name                  | selfservice                          |
| network_id            | 43643543-6edb-4c2b-a087-4553b75b6799 |
| prefix_length         | 24                                   |
| project_id            | 17c884da94bc4259b20ace3da6897297     |
| revision_number       | 0                                    |
| segment_id            | None                                 |
| service_types         |                                      |
| subnetpool_id         | 574f9d33-65b6-49a1-ab43-866085d06804 |
| tags                  |                                      |
| updated_at            | 2020-08-28T15:29:20Z                 |
+-----------------------+--------------------------------------+
```

6. Create and configure router

   - Create a router.

```
$ openstack router create router
```

(continues on next page)

(continued from previous page)

```
+-------------------------+-------------------------------------
↪+
| Field                   | Value                               ␣
↪|
+-------------------------+-------------------------------------
↪+
| admin_state_up          | UP                                  ␣
↪|
| availability_zone_hints |                                     ␣
↪|
| availability_zones      |                                     ␣
↪|
| created_at              | 2020-08-28T15:30:09Z                ␣
↪|
| description             |                                     ␣
↪|
| external_gateway_info   | null                                ␣
↪|
| flavor_id               | None                                ␣
↪|
| id                      | 250e5cc1-4cfc-4dff-a3a3-eb206c071621␣
↪|
| name                    | router                              ␣
↪|
| project_id              | 17c884da94bc4259b20ace3da6897297    ␣
↪|
| revision_number         | 1                                   ␣
↪|
| routes                  |                                     ␣
↪|
| status                  | ACTIVE                              ␣
↪|
| tags                    |                                     ␣
↪|
| updated_at              | 2020-08-28T15:30:09Z                ␣
↪|
+-------------------------+-------------------------------------
↪+
```

- Add the private subnet as an interface on the router.

```
$ openstack router add subnet router selfservice
```

- Add the provide network as a gateway on the router

```
$ openstack router set --external-gateway provider router
```

- Verify router ports. Note: from this result, you can see what the advertised routes are.

```
$ openstack port list --router router
+--------------------------------------+------+-----------------
↪-+-----------------------------------------------------------
↪---------------+--------+
| ID                                   | Name | MAC Address      ␣
↪ | Fixed IP Addresses                                          ␣
↪              | Status |
+--------------------------------------+------+-----------------
↪-+-----------------------------------------------------------
↪---------------+--------+
| 218c455f-f565-4e37-a2ac-999da24efa66 |      |␣
↪fa:16:3e:74:d8:61 | ip_address='10.0.0.1', subnet_id='12eec8cb-
↪8303-4829-8b16-e9a75072fcb0'    | ACTIVE |
| 44dcb7d3-b444-4177-82a1-233b1f3bed23 |      |␣
↪fa:16:3e:5b:4b:2d | ip_address='172.24.4.24', subnet_id=
↪'4ed8ac88-2c19-4f94-9362-7b301e743438' | ACTIVE |
+--------------------------------------+------+-----------------
↪-+-----------------------------------------------------------
↪---------------+--------+
```

7. Create and configure the BGP speaker

   The BGP speaker advertised the next-hop IP address for the tenant network prefix.

   - Create the BGP speaker.

   Replace LOCAL_AS with an appropriate local autonomous system number. The example configuration uses AS 12345.

```
$ openstack bgp speaker create  --ip-version 4 \
  --local-as LOCAL_AS bgp-speaker
+-----------------------------------+--------------------------
↪----------+
| Field                             | Value                    ␣
↪          |
+-----------------------------------+--------------------------
↪----------+
| advertise_floating_ip_host_routes | True                     ␣
↪          |
| advertise_tenant_networks         | True                     ␣
↪          |
| id                                | 19cdf669-4d4d-442f-bbf6-
↪510a97ad8cd8 |
| ip_version                        | 4                        ␣
↪          |
| local_as                          | 12345                    ␣
↪          |
| name                              | bgp-speaker              ␣
↪          |
| networks                          | []                       ␣
↪          |
| peers                             | []                       ␣
```

(continues on next page)

(continued from previous page)

```
↪                |
| project_id                        |␣
↪17c884da94bc4259b20ace3da6897297      |
+-----------------------------------+---------------------------
↪----------+
```

- Associate the BGP speaker with the provider network.

A BGP speaker requires association with a provider network to determine eligible prefixes. After the association, the BGP speaker can advertise the tenant network prefixes with the corresponding router as the next-hop IP address.

```
$ openstack bgp speaker add network bgp-speaker provider
```

- Verify the association of the provider network with the BGP speaker.

Checking the `networks` attribute.

```
$ openstack bgp speaker show bgp-speaker
+-----------------------------------+---------------------------
↪--------------+
| Field                             | Value                    ␣
↪              |
+-----------------------------------+---------------------------
↪--------------+
| advertise_floating_ip_host_routes | True                     ␣
↪              |
| advertise_tenant_networks         | True                     ␣
↪              |
| id                                | 19cdf669-4d4d-442f-bbf6-
↪510a97ad8cd8     |
| ip_version                        | 4                        ␣
↪              |
| local_as                          | 12345                    ␣
↪              |
| name                              | bgp-speaker              ␣
↪              |
| networks                          | ['18f9a9c0-f8f5-4360-822a-
↪b687c1008bf7'] |
| peers                             | []                       ␣
↪              |
| project_id                        |␣
↪17c884da94bc4259b20ace3da6897297         |
+-----------------------------------+---------------------------
↪--------------+
```

- Verify the prefixes and next-hop ip addresses that the BGP speaker advertises.

```
$ openstack bgp speaker list advertised routes bgp-speaker
+-------------+------------+
| destination | next_hop   |
```

(continues on next page)

(continued from previous page)

```
+-------------+------------+
| 10.0.0.0/24 | 172.24.4.3 |
+-------------+------------+
```

- Create a BGP peer.

Here the BGP peer is pointed to the quagga VM. Replace REMOTE_AS with an appropriate remote autonomous system number. The example configuration uses AS 12345 which triggers iBGP peering.

```
$ openstack bgp peer create --peer-ip 10.156.18.20 \
  --remote-as REMOTE_AS bgp-peer
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| auth_type  | none                                 |
| id         | 37291604-de77-4333-8f27-4ca336e021f2 |
| name       | bgp-peer                             |
| peer_ip    | 10.156.18.20                         |
| project_id | 17c884da94bc4259b20ace3da6897297     |
| remote_as  | 12345                                |
+------------+--------------------------------------+
```

- Add a BGP peer to the BGP speaker.

```
$ openstack bgp speaker add peer bgp-speaker bgp-peer
```

- Verify the association of the BGP peer with the BGP speaker.

Checking the `peers` attribute.

```
$ openstack bgp speaker show bgp-speaker
+-----------------------------------+-----------------------------
↪--------------+
| Field                             | Value                       ␣
↪              |
+-----------------------------------+-----------------------------
↪--------------+
| advertise_floating_ip_host_routes | True                        ␣
↪              |
| advertise_tenant_networks         | True                        ␣
↪              |
| id                                | 19cdf669-4d4d-442f-bbf6-
↪510a97ad8cd8      |
| ip_version                        | 4                           ␣
↪              |
| local_as                          | 12345                       ␣
↪              |
| name                              | bgp-speaker                 ␣
↪              |
| networks                          | ['18f9a9c0-f8f5-4360-822a-
```

(continues on next page)

(continued from previous page)

```
↪b687c1008bf7'] |
| peers                          | ['37291604-de77-4333-8f27-
↪4ca336e021f2'] |
| project_id                     |␣
↪17c884da94bc4259b20ace3da6897297       |
+--------------------------------+--------------------------
↪--------------+
```

8. Schedule the BGP speaker to an agent.

   - Schedule the BGP speaker to `BGP dynamic routing agent`

   With the default scheduler configuration, the first BGP speaker is scheduled to the first dynamic routing agent automatically. So for a simple setup, there is nothing to be done here.

   - Verify scheduling of the BGP speaker to the agent.

```
$ openstack bgp dragent list --bgp-speaker bgp-speaker
+--------------------------------------+-------------------------
↪--+----------------+-------+
| id                                   | host                    ␣
↪  | admin_state_up | alive |
+--------------------------------------+-------------------------
↪--+----------------+-------+
| 239996c8-2d59-4131-98b8-d64372c812cc | devstack-bgp-dr         ␣
↪  | True           | :-)   |
+--------------------------------------+-------------------------
↪--+----------------+-------+
```

6.3 DRAgent Drivers

6.3.1 Introduction

The Neutron dynamic routing drivers are used to support different dynamic routing protocol stacks which implement the dynamic routing functionality.

As shown in the following figure, the drivers are managed by DRAgent through a Driver Manager which provides consistent APIs to realize the functionality of a dynamic routing protocol:

```
 Neutron Dynamic Routing Drivers
+-------------------------------+
|          DRAgent              |
|                               |
|  +-------------------------+  |
|  |    Driver Manager       |  |
|  +-------------------------+  |
|  |    Common Driver API    |  |
|  +-------------------------+  |
|               |               |
|               |               |
```

(continues on next page)

(continued from previous page)

```
|  +------------+------------+  |
|  |  os-ken    |  Other     |  |
|  |  Driver    |  Drivers   |  |
|  +------------+------------+  |
|                               |
+-------------------------------+
```

Note

Currently only the integration with os-ken is supported BGP is the only protocol supported.

6.3.2 Configuration

Driver configurations are done in a separate configuration file.

BGP Driver

There are two configuration parameters related to BGP which are specified in `bgp_dragent.ini`.

- bgp_speaker_driver, to define BGP speaker driver class. Default is os-ken (neutron_dynamic_routing.services.bgp.agent.driver.os_ken.driver.OsKenBgpDriver).
- bgp_router_id, to define BGP identity (typically an IPv4 address). Default is a unique loopback interface IP address.

6.3.3 Common Driver API

Common Driver API is needed to provide a generic and consistent interface to different drivers. Each driver need to implement the provided base driver class.

BGP

Following interfaces need to be implemented by a driver for realizing BGP functionality.

API name	Description
add_bgp_speaker()	Add a BGP Speaker
delete_bgp_speaker()	Delete a BGP speaker
add_bgp_peer()	Add a BGP peer
delete_bgp_peer()	Delete a BGP peer
advertise_route()	Add a new prefix to advertise
withdraw_route()	Withdraw an advertised prefix
get_bgp_speaker_statistics()	Collect BGP Speaker statistics
get_bgp_peer_statistics()	Collect BGP Peer statistics

CHAPTER

SEVEN

USE-CASE: GLOBAL IPV6 CONNECTIVITY FOR TENANT NETWORKS

7.1 Motivation

With IPv6 the use of NAT is strongly deprecated with the intention of allowing direct end-to-end connectivity between hosts. Thus the Neutron implementation does not provide a mechanism to create floating IPv6 addresses.

As a consequence, projects will want globally routed IPv6 adresses directly connected to their instances. As this will be difficult to achieve if you continue to allow your projects to choose their own favorite range of addresses, there are two possible solutions:

1. Set up a shared network to which your instances will be directly connected and configure this network with a public IPv6 prefix (*provider network*). This way instances will get a public IPv6 address that they can use without any restriction.
2. Configure a *subnet pool* containing a range of public IPv6 prefixes, so that projects may configure their own networks by requesting a slice from that subnet pool instead of choosing their own.

Option 1 has some drawbacks, though:

- It places all projects into a single shared network, complicating things like per-project firewall rules or rDNS management.
- If you want to dual-stack your instances (i.e. provide them with IPv4 and IPv6 connectivity at the same time), you either need to use two different networks and attach your instances to both of them, or you will need to also assign a public IPv4 address to every instance, which may be considered a rather wasteful approach to dealing with this scarce resource.

So this document will describe how to set up option 2 by deploying *dynamic routing*.

7.2 Preparation

7.2.1 Address scope

Start by setting up an address scope that will be used by the BGP agent in order to select the set of prefixes to be announced over the BGP sessions. All commands shown in this section will require you to use admin credentials:

```
$ openstack address scope create --share --ip-version 6 ipv6-global
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| id         | ee2ee196-156c-424e-81e5-4d029c66190a |
| ip_version | 6                                    |
| name       | ipv6-global                          |
| project_id | 6de6f29dcf904ab8a12e8ca558f532e9     |
| shared     | True                                 |
+------------+--------------------------------------+
```

7.2.2 Subnet pool

Next create the subnet pool that your projects will use to configure their subnets with. For a real deployment, use a globally routable network prefix instead of the documentation prefix that is used in this example:

```
$ openstack subnet pool create --address-scope ipv6-global --share --default \
>   --pool-prefix 2001:db8:1234::/48 --default-prefix-length 64 \
>   --min-prefix-length 64 --max-prefix-length 124 default-pool-ipv6
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| address_scope_id  | ee2ee196-156c-424e-81e5-4d029c66190a |
| created_at        | 2017-02-24T15:28:27Z                 |
| default_prefixlen | 64                                   |
| default_quota     | None                                 |
| description       |                                      |
| id                | 4c1661ba-b24c-4fda-8815-3f1fd29281af |
| ip_version        | 6                                    |
| is_default        | True                                 |
| max_prefixlen     | 124                                  |
| min_prefixlen     | 64                                   |
| name              | default-pool-ipv6                    |
| prefixes          | 2001:db8:1234::/48                   |
| project_id        | 6de6f29dcf904ab8a12e8ca558f532e9     |
| revision_number   | 1                                    |
| shared            | True                                 |
| updated_at        | 2017-02-24T15:28:27Z                 |
+-------------------+--------------------------------------+
```

7.2.3 Public network

Your public network, the network that the gateway ports of the project routers will be connected to, needs to be configured with the

For an existing deployment, you will usually already have a network defined using public IPv4 addresses for floating IPs and router gateway ports. This is the network that an IPv6 subnet needs to be added to, using same address scope that was used for the subnet pool above.

If you do not yet have such a network, you can create it with:

```
$ openstack network create --provider-network-type flat --provider-physical-
↪network public --external public
<output skipped>
```

The exact parameters that you need to specify depend on your deployment and are out of scope for this document, see the Neutron documentation for more details.

There are now three different options in order to make sure that you have an IPv6 subnet on this public network which is associated with the required address scope:

Using the shared subnet pool

Create the IPv6 subnet on the public network from the shared subnet pool above:

```
$ openstack subnet create --ip-version 6 --use-default-subnet-pool --network␣
↪public public-ip6
+-------------------+--------------------------------------------------------
↪-+
| Field             | Value                                                  ␣
↪ |
+-------------------+--------------------------------------------------------
↪-+
| allocation_pools  | 2001:db8:1234::2-2001:db8:1234:0:ffff:ffff:ffff:ffff   ␣
↪ |
| cidr              | 2001:db8:1234::/64                                     ␣
↪ |
| created_at        | 2017-02-27T10:23:00Z                                   ␣
↪ |
| description       |                                                        ␣
↪ |
| dns_nameservers   |                                                        ␣
↪ |
| enable_dhcp       | True                                                   ␣
↪ |
| gateway_ip        | 2001:db8:1234::1                                       ␣
↪ |
| host_routes       |                                                        ␣
↪ |
| id                | 77551166-fb97-4ea5-912a-c17c75a05eda                   ␣
↪ |
| ip_version        | 6                                                      ␣
↪ |
| ipv6_address_mode | None                                                   ␣
↪ |
| ipv6_ra_mode      | None                                                   ␣
↪ |
| name              | public-ip6                                             ␣
↪ |
| network_id        | 28c08355-cb8f-4b1b-b5fd-f5442e531b28                   ␣
↪ |
| project_id        | 6de6f29dcf904ab8a12e8ca558f532e9                       ␣
↪ |
| revision_number   | 2                                                      ␣
↪ |
| segment_id        | None                                                   ␣
↪ |
```

(continues on next page)

(continued from previous page)

```
| service_types     |                                                           ␣
↪ |
| subnetpool_id     | 56a1b34b-7e0a-4a76-aac9-8893314ee2a4                      ␣
↪ |
| updated_at        | 2017-02-27T10:23:00Z                                      ␣
↪ |
+-------------------+--------------------------------------------------------------
↪-+
```

Using a dedicated subnet pool

Create a second subnet pool containing just the specific prefix that you want to use for your public network:

```
$ openstack subnet pool create --address-scope ipv6-global --pool-prefix␣
↪2001:db8:4321:42::/64 --default-prefix-length 64 public-pool
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| address_scope_id  | ee2ee196-156c-424e-81e5-4d029c66190a |
| created_at        | 2017-02-27T10:13:38Z                 |
| default_prefixlen | 64                                   |
| default_quota     | None                                 |
| description       |                                      |
| id                | 56a1b34b-7e0a-4a76-aac9-8893314ee2a4 |
| ip_version        | 6                                    |
| is_default        | False                                |
| max_prefixlen     | 128                                  |
| min_prefixlen     | 64                                   |
| name              | public-pool                          |
| prefixes          | 2001:db8:4321:42::/64                |
| project_id        | 6de6f29dcf904ab8a12e8ca558f532e9     |
| revision_number   | 1                                    |
| shared            | False                                |
| updated_at        | 2017-02-27T10:13:38Z                 |
+-------------------+--------------------------------------+
```

Create the IPv6 subnet on your public network:

```
$ openstack subnet create --ip-version 6 --subnet-pool public-pool --network␣
↪public public-ip6
+-------------------+--------------------------------------------------------------
↪-+
| Field             | Value                                                     ␣
↪ |
+-------------------+--------------------------------------------------------------
↪-+
| allocation_pools  | 2001:db8:4321:42::2-
↪2001:db8:4321:42:ffff:ffff:ffff:ffff |
| cidr              | 2001:db8:4321:42::/64                                     ␣
```

(continues on next page)

(continued from previous page)

```
↪ |
| created_at        | 2017-02-27T10:23:00Z                                          ␣
↪ |
| description       |                                                               ␣
↪ |
| dns_nameservers   |                                                               ␣
↪ |
| enable_dhcp       | True                                                          ␣
↪ |
| gateway_ip        | 2001:db8:4321:42::1                                           ␣
↪ |
| host_routes       |                                                               ␣
↪ |
| id                | 77551166-fb97-4ea5-912a-c17c75a05eda                          ␣
↪ |
| ip_version        | 6                                                             ␣
↪ |
| ipv6_address_mode | None                                                          ␣
↪ |
| ipv6_ra_mode      | None                                                          ␣
↪ |
| name              | public-ip6                                                    ␣
↪ |
| network_id        | 28c08355-cb8f-4b1b-b5fd-f5442e531b28                          ␣
↪ |
| project_id        | 6de6f29dcf904ab8a12e8ca558f532e9                              ␣
↪ |
| revision_number   | 2                                                             ␣
↪ |
| segment_id        | None                                                          ␣
↪ |
| service_types     |                                                               ␣
↪ |
| subnetpool_id     | 56a1b34b-7e0a-4a76-aac9-8893314ee2a4                          ␣
↪ |
| updated_at        | 2017-02-27T10:23:00Z                                          ␣
↪ |
+-------------------+-------------------------------------------------------------
↪-+
```

Using subnet onboarding

If you have enabled the `subnet_onboard` extension in your Neutron deployment, there is the simpler option of simply onboarding an existing IPv6 subnet on your public network onto the default IPv6 subnet pool created above:

```
$ openstack network onboard subnets public default-pool-ipv6
$ # This command does not generate any output
```

Verifying the address scope

After you have create the public IPv6 subnet using one of the options shown above, verify that the address scope for IPv6 indeed got set for the public network:

```
$ openstack network show public
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2017-02-27T10:21:04Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 28c08355-cb8f-4b1b-b5fd-f5442e531b28 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | ee2ee196-156c-424e-81e5-4d029c66190a |
| is_default                | False                                |
| mtu                       | 1500                                 |
| name                      | public                               |
| port_security_enabled     | True                                 |
| project_id                | 6de6f29dcf904ab8a12e8ca558f532e9     |
| provider:network_type     | flat                                 |
| provider:physical_network | external                             |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 6                                    |
| router:external           | External                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 77551166-fb97-4ea5-912a-c17c75a05eda |
| updated_at                | 2017-02-27T10:23:00Z                 |
+---------------------------+--------------------------------------+
```

Note

It is essential that the same address scope is being used both for the public subnet and the tenant subnets. If there is a mismatch, the BGP announcement will not happen and connectivity will be broken.

7.3 Dynamic routing setup

Now that you have prepared the network configuration, the next step is to configure the dynamic routing part. First add the service plugin to your `neutron.conf` file. Note that depending on your deployment, there may be other plugins already configured, keep those unchanged, just add the BGP plugin:

```
[DEFAULT]
# You may have other plugins enabled here depending on your environment
```

(continues on next page)

(continued from previous page)

```
# Important thing is you add the BgpPlugin to the list
service_plugins = neutron_dynamic_routing.services.bgp.bgp_plugin.BgpPlugin,
↪neutron.services.l3_router.l3_router_plugin.L3RouterPlugin
# In case you run into issues, this will also be helpful
debug = true
```

You need to restart the Neutron service in order activate the plugin.

Now you can create your first BGP speaker. Set the IP version to 6, select some private ASN that can be used for this POC and disable advertising floating IPs:

```
$ openstack bgp speaker create --ip-version 6 --local-as 65000 --no-advertise-
↪floating-ip-host-routes bgp1
+-----------------------------------+--------------------------------------+
| Field                             | Value                                |
+-----------------------------------+--------------------------------------+
| advertise_floating_ip_host_routes | False                                |
| advertise_tenant_networks         | True                                 |
| id                                | b9547458-7bdd-4738-bd57-a985055fc59c |
| ip_version                        | 6                                    |
| local_as                          | 65000                                |
| name                              | bgp1                                 |
| networks                          | []                                   |
| peers                             | []                                   |
| project_id                        | c67d6ba16ea2484597061245e5258c1e     |
+-----------------------------------+--------------------------------------+
```

Add your public network to this speaker, indicating that you want to advertise all those tenant networks here, which have a router with the public network as gateway:

```
$ openstack bgp speaker add network bgp1 public
$ # This command does not generate any output
```

Assuming your external router has the address 2001:db8:4321:e0::1, configure it as BGP peer and add it to the BGP speaker:

```
$ openstack bgp peer create --peer-ip 2001:db8:4321:e0::1 --remote-as 65001␣
↪bgp-peer1
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| auth_type  | none                                 |
| id         | 0183260e-b1d0-40ae-994f-075668b99676 |
| name       | bgp-peer1                            |
| peer_ip    | 2001:db8:4321:e0::1                  |
| project_id | c67d6ba16ea2484597061245e5258c1e     |
| remote_as  | 65001                                |
| tenant_id  | c67d6ba16ea2484597061245e5258c1e     |
+------------+--------------------------------------+
$ openstack bgp speaker add peer bgp1 bgp-peer1
$ # This command does not generate any output
```

Now configure the Neutron BGP agent, add the following data into your `bgp_dragent.ini` configuration file:

```
[BGP]
bgp_speaker_driver = neutron_dynamic_routing.services.bgp.agent.driver.os_ken.
↪driver.OsKenBgpDriver

# 32-bit BGP identifier, typically an IPv4 address owned by the system running
# the BGP DrAgent.
bgp_router_id = 192.0.2.42
```

Again you will have to restart the agent in order for it to pick up the new configuration. If all goes well, the agent should register itself in your setup:

```
$ openstack network agent list --agent-type bgp
+--------------------------------------+---------------------------+----------
↪----+-------------------+-------+-------+---------------------+
| ID                                   | Agent Type                | Host    ␣
↪     | Availability Zone | Alive | State | Binary              |
+--------------------------------------+---------------------------+----------
↪----+-------------------+-------+-------+---------------------+
| 68d6e83c-db04-4711-b031-44cf3fb51bb7 | BGP dynamic routing agent | network-
↪node | None              | :-)   | UP    | neutron-bgp-dragent |
+--------------------------------------+---------------------------+----------
↪----+-------------------+-------+-------+---------------------+
```

As the final step, use the agent ID from above and tell the agent that it should host our BGP speaker:

```
$ openstack bgp dragent add speaker 68d6e83c-db04-4711-b031-44cf3fb51bb7 bgp1
$ # This command does not generate any output
```

7.4 The view from the outside

The final step is to configure your outside router to accept the BGP session from the Neutron dynamic routing agent, so it can receive the prefix announcements and forward traffic accordingly.

In this example we assume a system running [BIRD](http://bird.network.cz/), which we configure to be the remote end of the BGP session like this:

```
protocol bgp bgp1 {
  description "Neutron agent";
  passive on;
  local 2001:db8:4321:e0::1 as 65001;
  neighbor 2001:db8:4321:e0::42 as 65000;
}
```

Verify that the session gets established as expected:

```
bird> show proto bgp1
name     proto    table    state  since       info
bgp1     BGP      master   up     00:01:50    Established
```

7.5 Tenant networks

You have now prepared everything on the admin side of things, so lets have a look at how your users should configure their networking. The commands in this section are meant to be executed with user credentials:

```
$ openstack network create mynet
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2017-02-27T10:26:22Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 1f20da97-ddd4-40f8-b8d3-6321de8671a0 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| mtu                       | 1500                                 |
| name                      | mynet                                |
| port_security_enabled     | True                                 |
| project_id                | 6de6f29dcf904ab8a12e8ca558f532e9     |
| provider:network_type     | None                                 |
| provider:physical_network | None                                 |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 3                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| updated_at                | 2017-02-27T10:26:22Z                 |
+---------------------------+--------------------------------------+
```

In order to add an IPv6 prefix, simply request one from the default pool:

```
$ openstack subnet create --ip-version 6 --use-default-subnet-pool \
> --ipv6-address-mode slaac --ipv6-ra-mode slaac --network mynet mysubnet
+------------------------+----------------------------------------------------
↪----+
| Field                  | Value                                              ␣
↪    |
+------------------------+----------------------------------------------------
↪----+
| allocation_pools       | 2001:db8:1234:1::2-
↪2001:db8:1234:1:ffff:ffff:ffff:ffff |
| cidr                   | 2001:db8:1234:1::/64                               ␣
↪    |
```

(continues on next page)

(continued from previous page)

```
| created_at             | 2017-02-27T11:14:23Z                                     ␣
↪     |
| description            |                                                          ␣
↪     |
| dns_nameservers        |                                                          ␣
↪     |
| enable_dhcp            | True                                                     ␣
↪     |
| gateway_ip             | 2001:db8:1234:1::1                                       ␣
↪     |
| host_routes            |                                                          ␣
↪     |
| id                     | 193f7620-6c4c-4adc-9bb5-ff73c9b08d59                     ␣
↪     |
| ip_version             | 6                                                        ␣
↪     |
| ipv6_address_mode      | slaac                                                    ␣
↪     |
| ipv6_ra_mode           | slaac                                                    ␣
↪     |
| name                   | mysubnet                                                 ␣
↪     |
| network_id             | 1f20da97-ddd4-40f8-b8d3-6321de8671a0                     ␣
↪     |
| project_id             | 6de6f29dcf904ab8a12e8ca558f532e9                         ␣
↪     |
| revision_number        | 2                                                        ␣
↪     |
| segment_id             | None                                                     ␣
↪     |
| service_types          |                                                          ␣
↪     |
| subnetpool_id          | 4c1661ba-b24c-4fda-8815-3f1fd29281af                     ␣
↪     |
| updated_at             | 2017-02-27T11:14:23Z                                     ␣
↪     |
| use_default_subnetpool | true                                                     ␣
↪     |
+------------------------+-----------------------------------------------------
↪----+
```

For outside connectivity create a router, add an interface into your project net and set the gateway to be the public network:

```
$ openstack router create router1
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints |                                      |
```

(continues on next page)

(continued from previous page)

```
| availability_zones      |                                      |
| created_at              | 2017-02-27T12:59:06Z                 |
| description             |                                      |
| distributed             | False                                |
| external_gateway_info   | None                                 |
| flavor_id               | None                                 |
| ha                      | False                                |
| id                      | d2db0603-fda2-4305-a1de-e793a36c0770 |
| name                    | router1                              |
| project_id              | 6de6f29dcf904ab8a12e8ca558f532e9     |
| revision_number         | None                                 |
| routes                  |                                      |
| status                  | ACTIVE                               |
| updated_at              | 2017-02-27T12:59:06Z                 |
+-------------------------+--------------------------------------+
$ openstack router add subnet router1 mysubnet
$ openstack router set --external-gateway public router1
```

Finally boot an instance and verify that it gets an IPv6 address assigned:

```
$ openstack server create --flavor c1 --image cirros vm1
$ openstack server list
+--------------------------------------+------+--------+------------+--------
↪-------------------------------------+------------+
| ID                                   | Name | Status | Power State |␣
↪Networks                                    | Image Name |
+--------------------------------------+------+--------+------------+--------
↪-------------------------------------+------------+
| 17b2ac04-9a17-45ff-be30-401aa8331a66 | vm1  | ACTIVE | Running     |␣
↪mynet=2001:db8:1234:1:f816:3eff:fe53:f89e  | cirros     |
+--------------------------------------+------+--------+------------+--------
↪-------------------------------------+------------+
$ openstack console log show vm1 | grep -A1 -B1 2001
eth0      Link encap:Ethernet  HWaddr FA:16:3E:53:F8:9E
          inet6 addr: 2001:db8:1234:1:f816:3eff:fe53:f89e/64 Scope:Global
          inet6 addr: fe80::f816:3eff:fe53:f89e/64 Scope:Link
```

Note

Most cloud images are built to insist on receiving an IPv4 address via DHCP, so there will be a considerable delay durging booting while waiting for this to run into a timeout. In order to avoid the resulting delay, you could add an IPv4 subnet to your project net.

Note

There is a similar concern regarding access to the metadata provided by the compute service. While Neutron does provide the option to access metadata via IPv6, this is not available in all deployments and not supported by all cloud images. You can work around this either by adding and IPv4 subnet like mentioned above or by using the config drive option to provide metadata to your instance.

7.6 Verification

As an admin you can now verify that the tenant network gets listed for advertisement:

```
$ openstack bgp speaker list advertised routes bgp1
+----+--------------------+---------------------+
| ID | Destination        | Nexthop             |
+----+--------------------+---------------------+
|    | 2001:db8:1234::/64 | 2001:db8:4321:42::c |
+----+--------------------+---------------------+
```

And verify it is being seen on your outside router:

```
bird> show route 2001:db8:1234:1::/64
2001:db8:1234:1::/64 via 2001:db8:4321:2::5 on ens3 [bgp1 12:06:50 from␣
↪2001:db8:4321:e0::42] * (100/0) [i]
```

As extra bonus, verify that the instance is reachable from the router:

```
router01:~$ ping6 -c3  2001:db8:1234:1:f816:3eff:fecd:6bf4
PING 2001:db8:1234:1:f816:3eff:fecd:6bf4(2001:db8:1234:1:f816:3eff:fecd:6bf4)␣
↪56 data bytes
64 bytes from 2001:db8:1234:1:f816:3eff:fecd:6bf4: icmp_seq=1 ttl=63 time=1.
↪80 ms
64 bytes from 2001:db8:1234:1:f816:3eff:fecd:6bf4: icmp_seq=2 ttl=63 time=0.
↪724 ms
64 bytes from 2001:db8:1234:1:f816:3eff:fecd:6bf4: icmp_seq=3 ttl=63 time=1.
↪04 ms

--- 2001:db8:1234:1:f816:3eff:fecd:6bf4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.724/1.190/1.803/0.454 ms
router01:~$ ssh -6 2001:db8:1234:1:f816:3eff:fe58:f80a -l cirros
cirros@2001:db8:1234:1:f816:3eff:fe58:f80a's password:
$
```