Border Gateway Protocol - BGP

BGP is the routing protocol that runs the Internet. It is an increasingly popular protocol for use in the data center as it lends itself well to the rich interconnections in a Clos topology. Specifically, BGP:

- Does not require the routing state to be periodically refreshed, unlike OSPF.
- Is less chatty than its link-state siblings. For example, a link or node transition can result in a bestpath change, causing BGP to send updates.
- Is multi-protocol and extensible.
- Has many robust vendor implementations.
- Is very mature as a protocol and comes with many years of operational experience.

RFC 7938 provides further details of the use of BGP within the data center.

Contents

- Autonomous System Number (ASN)
- eBGP and iBGP
- Route Reflectors
- ECMP with BGP
  - Maximum Paths
  - BGP for Both IPv4 and IPv6
- Configure BGP
- BGP Unnumbered Interfaces
  - BGP and Extended Next Hop Encoding
  - Configure BGP Unnumbered Interfaces
  - Manage Unnumbered Interfaces
  - How traceroute Interacts with BGP Unnumbered Interfaces
  - Advanced: How Next Hop Fields Are Set
  - Limitations
- RFC 5549 Support with Global IPv6 Peers (Cumulus Linux 3.7.2 and later)
  - Configure RFC 5549 Support with Global IPv6 Peers
  - Show IPv4 Prefixes Learned with IPv6 Next Hops
- BGP add-path
  - BGP add-path RX
  - BGP add-path TX
- Fast Convergence Design Considerations
- Peer Groups to Simplify Configuration
- Configure BGP Dynamic Neighbors
- Configure BGP Peering Relationships across Switches
- Configure MD5-enabled BGP Neighbors
- Configure eBGP Multihop
- Configure BGP TTL Security
- Configure Graceful BGP Shutdown
- Configuration Tips
  - BGP Advertisement Best Practices
  - Multiple Routing Tables and Forwarding
  - BGP Community Lists
  - Additional Default Settings
  - Configure BGP Neighbor maximum-prefixes
- Troubleshooting
  - Log Neighbor State Changes
  - Troubleshoot Link-local Addresses
- Enable Read-only Mode
- Apply a Route Map for Route Updates
  - Filter Routes from BGP into Zebra
  - Filter Routes from Zebra into the Linux Kernel
- Protocol Tuning
  - Converge Quickly On Link Failures
  - Converge Quickly On Soft Failures
  - Reconnect Quickly
  - Advertisement Interval
- Caveats and Errata
  - ttl-security Issue
  - BGP Dynamic Capabilities not Supported
  - Related Information
Autonomous System Number (ASN)

One of the key concepts in BGP is an autonomous system number or ASN. An autonomous system is defined as a set of routers under a common administration. Because BGP was originally designed to peer between independently managed enterprises and/or service providers, each such enterprise is treated as an autonomous system, responsible for a set of network addresses. Each such autonomous system is given a unique number called its ASN. ASNs are handed out by a central authority (ICANN). However, ASNs between 64512 and 65535 are reserved for private use. Using BGP within the data center relies on either using this number space or using the single ASN you own.

The ASN is central to how BGP builds a forwarding topology. A BGP route advertisement carries with it not only the originator’s ASN, but also the list of ASNs that this route advertisement has passed through. When forwarding a route advertisement, a BGP speaker adds itself to this list. This list of ASNs is called the AS path. BGP uses the AS path to detect and avoid loops.

ASNs were originally 16-bit numbers, but were later modified to be 32-bit. FRRouting supports both 16-bit and 32-bit ASNs, but most implementations still run with 16-bit ASNs.

eBGP and iBGP

When BGP is used to peer between autonomous systems, the peering is referred to as external BGP or eBGP. When BGP is used within an autonomous system, the peering used is referred to as internal BGP or iBGP. eBGP peers have different ASNs while iBGP peers have the same ASN.

The heart of the protocol is the same when used as eBGP or iBGP, however, there is a key difference in the protocol behavior between use as eBGP and iBGP: an iBGP speaker does not forward routing information learned from one iBGP peer to another iBGP peer to prevent loops. eBGP prevents loops using the AS_Path attribute.

All iBGP speakers need to be peered with each other in a full mesh. In a large network, this requirement can quickly become unscalable. The most popular method to scale iBGP networks is to introduce a route reflector.

Route Reflectors

In a two-tier Clos network, the leaf (or tier 1) switches are the only ones connected to end stations. The spines themselves do not have any routes to announce; they are merely reflecting the routes announced by one leaf to the other leaves. Therefore, the spine switches function as route reflectors while the leaf switches serve as route reflector clients.

In a three-tier network, the tier 2 nodes (or mid-tier spines) act as both route reflector servers and route reflector clients. They act as route reflectors because they announce the routes learned from the tier 1 nodes to other tier 1 nodes and to tier 3 nodes. They also act as route reflector clients to the tier 3 nodes, receiving routes learned from other tier 2 nodes. Tier 3 nodes act only as route reflectors.

In the following illustration, tier 2 node 2.1 is acting as a route reflector server, announcing the routes between tier 1 nodes 1.1 and 1.2 to tier 1 node 1.3. It is also a route reflector client, learning the routes between tier 2 nodes 2.2 and 2.3 from the tier 3 node, 3.1.

Configuring route-reflector-client Requires Specific Order

When configuring a router to be a route reflector client, you must specify the FRRouting configuration in a specific order; otherwise, the router will not be a route reflector client.

You must run the net add bgp neighbor <IPv4/IPV6> route-reflector-client command after the net add bgp neighbor <IPv4/IPV6> activate command; otherwise, the route-reflector-client command is ignored. For example:
A cluster consists of route reflectors (RRs) and their clients and is used in iBGP environments where multiple sets of route reflectors and their clients are configured. Configuring a unique ID per cluster (on the route reflector server and clients) prevents looping as a route reflector does not accept routes from another that has the same cluster ID. Additionally, because all route reflectors in the cluster recognize updates from peers in the same cluster, they do not install routes from a route reflector in the same cluster; this reduces the number of updates that need to be stored in BGP routing tables.

To configure a cluster ID on a route reflector, run the `net add bgp cluster-id (<ipv4>|<1-4294967295>)` command. You can enter the cluster ID as an IP address or as a 32-bit quantity.

The following example configures a cluster ID on a route reflector in IP address format:

```
cumulus@switch:~$ net add bgp cluster-id 14.0.0.9
```

The following example configures a cluster ID on a route reflector as a 32-bit quantity:

```
cumulus@switch:~$ net add bgp cluster-id 321
```

**ECMP with BGP**

If a BGP node hears a prefix from multiple peers, it has all the information necessary to program the routing table to forward traffic for that prefix through all of these peers; BGP supports equal-cost multipathing (ECMP).

To perform ECMP in BGP, you may need to configure `net add bgp bestpath as-path multipath-relax` (if you are using eBGP).

**Maximum Paths**

In Cumulus Linux, the BGP `maximum-paths` setting is enabled by default, so multiple routes are already installed. The default setting is 64 paths.

**BGP for Both IPv4 and IPv6**
Unlike OSPF, which has separate versions of the protocol to announce IPv4 and IPv6 routes, BGP is a multi-protocol routing engine, capable of announcing both IPv4 and IPv6 prefixes. It supports announcing IPv4 prefixes over an IPv4 session and IPv6 prefixes over an IPv6 session. It also supports announcing prefixes of both these address families over a single IPv4 session or over a single IPv6 session.

Configure BGP

The following example shows a basic BGP configuration. The rest of this chapter discusses how to configure other BGP features, such as unnumbered interfaces to route maps.

1. Enable the BGP and Zebra daemons (zebra and bgpd), then enable the FRRouting service and start it, as described in Configuring FRRouting.
2. Identify the BGP node by assigning an ASN and router-id:
   
   ```
cumulus@switch:~$ net add bgp autonomous-system 65000
   cumulus@switch:~$ net add bgp router-id 10.0.0.1
   ```
3. Specify where to disseminate routing information:
   
   ```
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 remote-as external
   ```

   For an iBGP session, the `remote-as` is the same as the local AS:
   
   ```
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 remote-as internal
   ```

   Specifying the IP address of the peer allows BGP to set up a TCP socket with this peer, but it does not distribute any prefixes to it, unless it is explicitly told that it must with the `activate` command. You must specify the `activate` command for each address family that is being announced by the BGP session.
   
   ```
cumulus@switch:~$ net add bgp ipv4 unicast neighbor 10.0.0.2 activate
   cumulus@switch:~$ net add bgp ipv6 unicast neighbor 2001:db8:0002::0a00:0002 activate
   ```
4. Specify BGP session properties:
   
   ```
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 next-hop-self
   ```

   If this is a route reflector client, it can be specified as follows:
   
   ```
cumulus@switchRR:~$ net add bgp neighbor 10.0.0.1 route-reflector-client
   ```

   It is node `switchRR`, the route reflector, on which the peer is specified as a client.
5. Specify which prefixes to originate:
BGP Unnumbered Interfaces

Unnumbered interfaces are interfaces without unique IP addresses. In BGP, you configure unnumbered interfaces using extended next hop encoding (ENHE), which is defined by RFC 5549. BGP unnumbered interfaces provide a means of advertising an IPv4 route with an IPv6 next hop. Prior to RFC 5549, an IPv4 route could be advertised only with an IPv4 next hop.

BGP unnumbered interfaces are particularly useful in deployments where IPv4 prefixes are advertised through BGP over a section without any IPv4 address configuration on links. As a result, the routing entries are also IPv4 for destination lookup and have IPv6 next hops for forwarding purposes.

BGP and Extended Next Hop Encoding

When enabled and active, BGP makes use of the available IPv6 next hops for advertising any IPv4 prefixes. BGP learns the prefixes, calculates the routes and installs them in IPv4 AFI to IPv6 AFI format. However, ENHE in Cumulus Linux does not install routes into the kernel in IPv4 prefix to IPv6 next hop format. For link-local peerings enabled by dynamically learning the other end's link-local address using IPv6 neighbor discovery router advertisements, an IPv6 next hop is converted into an IPv4 link-local address and a static neighbor entry is installed for this IPv4 link-local address with the MAC address derived from the link-local address of the other end.

It is assumed that the IPv6 implementation on the peering device uses the MAC address as the interface ID when assigning the IPv6 link-local address, as suggested by RFC 4291.

Cumulus Linux 3.7.2 and later also supports advertising IPv4 prefixes with IPv6 next hop addresses while peering over IPv6 global unicast addresses. See RFC 5549 Support with Global IPv6 Peers below.

Configure BGP Unnumbered Interfaces

To configure a BGP unnumbered interface, you must enable IPv6 neighbor discovery router advertisements. The interval you specify is measured in seconds and defaults to 10 seconds.

In Cumulus Linux 3.7.1 and earlier, extended next hop encoding is sent only for the link-local address peerings (as shown below). In Cumulus Linux 3.7.2 and later, extended next hop encoding can be sent for the both link-local and global unicast address peerings (see RFC 5549 Support with Global IPv6 Peers).
cumulus@switch:~$ net add bgp autonomous-system 65020
cumulus@switch:~$ net add bgp router-id 10.0.0.21
cumulus@switch:~$ net add bgp bestpath as-path multipath-relax
cumulus@switch:~$ net add bgp bestpath compare-routerid
cumulus@switch:~$ net add bgp neighbor fabric peer-group
cumulus@switch:~$ net add bgp neighbor fabric remote-as external
cumulus@switch:~$ net add bgp neighbor fabric description Internal Fabric Network
cumulus@switch:~$ net add bgp neighbor fabric capability extended-nexthop
cumulus@switch:~$ net add bgp neighbor swp1 interface peer-group fabric
cumulus@switch:~$ net add bgp neighbor swp2 interface peer-group fabric
cumulus@switch:~$ net add bgp neighbor swp3 interface peer-group fabric
cumulus@switch:~$ net add bgp neighbor swp4 interface peer-group fabric
cumulus@switch:~$ net add bgp neighbor swp29 interface peer-group fabric
cumulus@switch:~$ net add bgp neighbor swp30 interface peer-group fabric

These commands create the following configuration in the /etc/frr/frr.conf file:

```
router bgp 65020
  bgp router-id 10.0.0.21
  bgp bestpath as-path multipath-relax
  bgp bestpath compare-routerid
  neighbor fabric peer-group
  neighbor fabric remote-as external
  neighbor fabric description Internal Fabric Network
  neighbor fabric capability extended-nexthop
  neighbor swp1 interface peer-group fabric
  neighbor swp2 interface peer-group fabric
  neighbor swp3 interface peer-group fabric
  neighbor swp4 interface peer-group fabric
  neighbor swp29 interface peer-group fabric
  neighbor swp30 interface peer-group fabric
!
```

For an unnumbered configuration, you can use a single command to configure a neighbor and attach it to a peer group (making sure to substitute for the interface and peer group below):

```
cumulus@switch:~$ net add bgp neighbor <swpX> interface peer-group <group name>
```

**Manage Unnumbered Interfaces**

All the relevant BGP commands show IPv6 next hops and/or the interface name for any IPv4 prefix:
cumulus@switch:~$ net show bgp

show bgp ipv4 unicast
=====================
BGP table version is 6, local router ID is 10.0.0.11
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
    i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Hop</th>
<th>Metric</th>
<th>LocPrf</th>
<th>Weight</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>*&gt; 10.0.0.11/32</td>
<td>0.0.0.0</td>
<td>0</td>
<td>32768</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>*&gt; 10.0.0.12/32</td>
<td>swp51</td>
<td>0</td>
<td>65020</td>
<td>65012</td>
<td>i</td>
</tr>
<tr>
<td>*=</td>
<td>swp52</td>
<td>0</td>
<td>65020</td>
<td>65012</td>
<td>i</td>
</tr>
<tr>
<td>*&gt; 10.0.0.21/32</td>
<td>swp51</td>
<td>0</td>
<td>65020</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>*=</td>
<td>swp52</td>
<td>0</td>
<td>65020</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>*&gt; 172.16.1.0/24</td>
<td>0.0.0.0</td>
<td>0</td>
<td>32768</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>*=</td>
<td>swp52</td>
<td>0</td>
<td>65020</td>
<td>65012</td>
<td>i</td>
</tr>
</tbody>
</table>

Total number of prefixes 6

show bgp ipv6 unicast
=====================
No BGP network exists

FRRouting RIB commands are also modified:

cumulus@switch:~$ net show route
RIB entry for route
=====================
Codes: K - kernel route, C - connected, S - static, R - RIP,
    O - OSPF, I - IS-IS, B - BGP, P - PIM, T - Table,
    > - selected route, * - FIB route
K>* 0.0.0.0/0 via 192.168.0.254, eth0
C>* 10.0.0.11/32 is directly connected, lo
B>* 10.0.0.12/32 [20/0] via fe80::4638:3ff:fe00:5c, swp51, 1d01h04m
    * via fe80::4638:3ff:fe00:2b, swp52, 1d01h04m
B>* 10.0.0.21/32 [20/0] via fe80::4638:3ff:fe00:5c, swp51, 1d01h04m
B>* 10.0.0.22/32 [20/0] via fe80::4638:3ff:fe00:2b, swp52, 1d01h04m
C>* 172.16.1.0/24 is directly connected, br0
B>* 172.16.2.0/24 [20/0] via fe80::4638:3ff:fe00:5c, swp51, 1d01h04m
    * via fe80::4638:3ff:fe00:2b, swp52, 1d01h04m
C>* 192.168.0.0/24 is directly connected, eth0

The following commands show how the IPv4 link-local address 169.254.0.1 is used to install the route and static neighbor entry to facilitate proper forwarding without having to install an IPv4 prefix with IPv6 next hop in the kernel:
cumulus@switch:~$ net show route 10.0.0.12
RIB entry for 10.0.0.12
=======================
Routing entry for 10.0.0.12/32
    Known via "bgp", distance 20, metric 0, best
    Last update 1d01h06m ago
* fe80::4638:39ff:fe00:5c, via swp51
* fe80::4638:39ff:fe00:2b, via swp52

FIB entry for 10.0.0.12
=======================
10.0.0.12  proto zebra  metric 20
    nexthop via 169.254.0.1  dev swp51 weight 1 onlink
    nexthop via 169.254.0.1  dev swp52 weight 1 onlink

You can use the following command to display more neighbor information:

cumulus@switch:~$ ip neighbor
192.168.0.254 dev eth0 lladdr 44:38:39:00:00:5f REACHABLE
169.254.0.1 dev swp52 lladdr 44:38:39:00:00:2b PERMANENT
169.254.0.1 dev swp51 lladdr 44:38:39:00:00:5c PERMANENT
fe80::4638:39ff:fe00:2b dev swp52 lladdr 44:38:39:00:00:2b router REACHABLE
fe80::4638:39ff:fe00:5c dev swp51 lladdr 44:38:39:00:00:5c router REACHABLE

How traceroute Interacts with BGP Unnumbered Interfaces

Every router or end host must have an IPv4 address to complete a traceroute of IPv4 addresses. In this case, the IPv4 address used is that of the loopback device.

Even if ENHE is not used in the data center, link addresses are not typically advertised. This is because:

- Link addresses take up valuable FIB resources. In a large Clos environment, the number of such addresses can be quite large.
- Link addresses expose an additional attack vector for intruders to use to either break in or engage in DDoS attacks.

Assigning an IP address to the loopback device is essential.

Advanced: How Next Hop Fields Are Set

This section describes how the IPv6 next hops are set in the MP_REACH_NLRI (multiprotocol reachable NLRI) initiated by the system, which applies whether IPv6 prefixes or IPv4 prefixes are exchanged with ENHE. There are two main aspects to determine — how many IPv6 next hops are included in the MP_REACH_NLRI (since the RFC allows either one or two next hops) and the values of the nexthop(s). This section also describes how a received MP_REACH_NLRI is handled as far as processing IPv6 next hops.

- Whether peering to a global IPv6 address or link-local IPv6 address, the determination whether to send one or two next hops is as follows:
  1. If reflecting the route, two next hops are sent only if the peer has nexthop-local unchanged configured and the attribute of the received route has an IPv6 link-local next hop; otherwise, only one next hop is sent.
  2. Otherwise (if it is not reflecting the route), two next hops are sent if explicitly configured (nexthop-local unchanged) or the peer is directly connected (that is, either peering is on link-local address or the global IPv4 or IPv6 address is directly connected) and the route is either a local/self-originated route or the peer is an eBGP peer.
  3. In all other cases, only one next hop gets sent, unless an outbound route map adds another next hop.

route-map can impose two next hops in scenarios where Cumulus Linux only sends one next hop — by specifying set ipv6 nexthop link-local.

For all routes to eBGP peers and self-originated routes to iBGP peers, the global next hop (first value) is the peering address of the local system. If the peering is on the link-local address, this is the global IPv6 address on the peering interface, if present; otherwise, it is the link-local IPv6 address on the peering interface.

For other routes to iBGP peers (eBGP to iBGP or reflected), the global next hop will be the global next hop in the received attribute.
If this address is a link-local IPv6 address, it is reset so that the link-local IPv6 address of the eBGP peer is not passed along to an iBGP peer, which most likely is on a different link.

- route-map and/or the peer configuration can change the above behavior. For example, route-map can set the global IPv6 next hop or the peer configuration can set it to self — which is relevant for iBGP peers. The route map or peer configuration can also set the next hop to unchanged, which ensures the source IPv6 global next hop is passed around — which is relevant for eBGP peers.
- Whenever two next hops are being sent, the link-local next hop (the second value of the two) is the link-local IPv6 address on the peering interface unless it is due to nh-local-unchanged or route-map has set the link-local next hop.
- Network administrators cannot set martian values for IPv6 next hops in route-map. Also, global and link-local next hops are validated to ensure they match the respective address types.
- In a received update, a martian check is imposed for the IPv6 global next hop. If the check fails, it gets treated as an implicit withdraw.
- If two next hops are received in an update and the second next hop is not a link-local address, it gets ignored and the update is treated as if only one next hop was received.
- Whenever two next hops are received in an update, the second next hop is used to install the route into zebra. As per the previous point, it is already assured that this is a link-local IPv6 address. Currently, this is assumed to be reachable and is not registered with NHT.
- When route-map specifies the next hop as peer-address, the global IPv6 next hop as well as the link-local IPv6 next hop (if it's being sent) is set to the peering address. If the peering is on a link-local address, the former could be the link-local address on the peering interface, unless there is a global IPv6 address present on this interface.
- When using iBGP unnumbered with IPv6 Link Local Addresses (the default), FRR rewrites the BGP next hop to be the adjacent link. This is similar behavior to eBGP next hops. However, iBGP route advertisement rules do not change and a full mesh or route reflectors is still required.

The above rules imply that there are scenarios where a generated update has two IPv6 next hops, and both of them are the IPv6 link-local address of the peering interface on the local system. If you are peering with a switch or router that is not running Cumulus Linux and expects the first next hop to be a global IPv6 address, a route map can be used on the sender to specify a global IPv6 address. This conforms with the recommendations in the Internet draft draft-kato-bgp-ipv6-link-local-00.txt, “BGP4+ Peering Using IPv6 Link-local Address”.

Limitations

- Interface-based peering with separate IPv4 and IPv6 sessions is not supported.
- In Cumulus Linux 3.7.1 and earlier, ENHE is sent for IPv6 link-local peerings only. In Cumulus Linux 3.7.2 and later, ENHE can also be sent for IPv6 Unicast peerings (see below).
- If an IPv4 /30 or /31 IP address is assigned to the interface, IPv4 peering is used over IPv6 link-local peering.
- If the default router lifetime in the generated IPv6 route advertisements (RA) is set to 0, the receiving FRRouting instance drops the RA if it is on a Cumulus Linux 2.5.z switch. To work around this issue, either:
  - Explicitly configure the switch to advertise a router lifetime of 0, unless a value is specifically set by the operator — with the assumption that the host is running Cumulus Linux 3.y.z version of FRRouting. When hosts see an IPv6 RA with a router lifetime of 0, they do not make that router a default router.
  - Use the sysctl on the host — net.ipv6.conf.all.accept_ra_defrtr. However, this requires applying this setting on all hosts, which might mean many hosts, especially if FRRouting is run on the hosts.

RFC 5549 Support with Global IPv6 Peers (Cumulus Linux 3.7.2 and later)

RFC 5549 defines the method used for BGP to advertise IPv4 prefixes with IPv6 next hops. The RFC does not make a distinction between whether the IPv6 peering and next hop values should be global unicast addresses (GUA) or link-local addresses. Cumulus Linux 3.7.1 and earlier only supports advertising IPv4 prefixes using link-local IPv6 next hop addresses via BGP unnumbered peering. Cumulus Linux 3.7.2 supports advertising IPv4 prefixes with IPv6 global unicast and link-local next hop addresses, with either unnumbered or numbered BGP.

When BGP peering uses IPv6 global addresses and IPv4 prefixes are being advertised and installed, IPv6 route advertisements are used to derive the MAC address of the peer so that FRR can create an IPv4 route with a link-local IPv4 next hop address (defined by RFC 3927). This is required to install the route into the kernel. These route advertisement settings are configured automatically when FRR receives an update from a BGP peer using IPv6 global addresses that contain an IPv4 prefix with an IPv6 nexthop, and the enhanced-nexthop capability has been negotiated.

Configure RFC 5549 Support with Global IPv6 Peers

To enable advertisement of IPv4 prefixes with IPv6 next hops over IPv6 peerings, add the extended-nexthop capability to the global IPv6 neighbor statements on each end of the BGP sessions.
The above commands create the following configuration in the `/etc/frr/frr.conf` file:

```plaintext
cumulus@switch:~$ net add bgp neighbor 2001:1:1::3 capability extended-nexthop
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

```
router bgp 1
 bgp router-id 10.0.0.11
 neighbor 2001:1:1::3 remote-as external
 neighbor 2001:1:1::3 capability extended-nexthop
!
```

Ensure that the IPv6 peers are activated under the IPv4 unicast address family; otherwise, all peers are activated in the IPv4 unicast address family by default. If `no bgp default ipv4-unicast` is configured, you need to explicitly activate the IPv6 neighbor under the IPv4 unicast address family as shown below:

```
cumulus@switch:~$ net add bgp neighbor 2001:1:1::3 capability extended-nexthop
cumulus@switch:~$ net add bgp ipv4 unicast neighbor 2001:1:1::3 activate
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The above commands create the following configuration in the `/etc/frr/frr.conf` file:

```
router bgp 1 bgp
 router-id 10.0.0.11
 no bgp default ipv4-unicast
 neighbor 2001:1:1::3 remote-as external
 neighbor 2001:1:1::3 capability extended-nexthop
!
 address-family ipv4 unicast
 neighbor 2001:1:1::3 activate
 exit-address-family
```

### Show IPv4 Prefixes Learned with IPv6 Next Hops

To show IPv4 prefixes learned with IPv6 next hops, you can run `net show bgp ipv4 unicast` commands.

The following examples show an IPv4 prefix learned from a BGP peer over an IPv6 session using IPv6 global addresses, but where the next hop installed by BGP is a link-local IPv6 address. This occurs when the session is directly between peers and both link-local and global IPv6 addresses are included as next hops in the BGP update for the prefix. If both global and link-local next hops exist, BGP prefers the link-local address for route installation.
root@Spine01:~# net show bgp ipv4 unicast summary
BGP router identifier 10.0.0.11, local AS number 1 vrf-id 0
BGP table version 3
RIB entries 1, using 152 bytes of memory
Peers 1, using 19 KiB of memory

Neighbor            V AS   MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/PfxRcd
Leaf01(2001:1:1::3) 4 3   6432    6431    0      0 0 05:21:25 1

Total number of neighbors 1

root@Spine01:~# net show bgp ipv4 unicast
BGP table version is 3,
local router ID is 10.0.0.11
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

Network         Next Hop                 Metric   LocPrf Weight Path
*> 172.16.3.0/24 fe80::a00:27ff:fea6:b9fe  0       0       3     i

Displayed 1 routes and 1 total paths

root@Spine01:~# net show bgp ipv4 unicast 172.16.3.0/24
BGP routing table entry for 172.16.3.0/24
Paths: (1 available, best #1, table default)
Advertised to non peer-group peers:
Leaf01(2001:1:1::3)
3
 2001:1:1::3 from Leaf01(2001:1:1::3) (10.0.0.13)
    (fe80::a00:27ff:fea6:b9fe) (used)
    Origin IGP, metric 0, valid, external, bestpath-from-AS 3, best
    AddPath ID: RX 0, TX 3
    Last update: Mon Oct 22 08:09:22 2018

The example output below shows the results of installing the route in the FRR RIB as well as the kernel FIB. Note that the next hop used for installation in the FRR RIB is the link-local IPv6 address, but then it is converted into an IPv4 link-local address as required for installation into the kernel FIB.
If an IPv4 prefix is learned with only an IPv6 global next hop address (for example, when the route is learned through a route reflector), the command output shows the IPv6 global address as the next hop value and shows that it is learned recursively through the link-local address of the route reflector. Note that when a global IPv6 address is used as a next hop for route installation in the FRR RIB, it is still converted into an IPv4 link-local address for installation into the kernel.
root@Leaf01:~# net show bgp ipv4 unicast summary
BGP router identifier 10.0.0.13, local AS number 1 vrf-id 0
BGP table version 1
RIB entries 1, using 152 bytes of memory
Peers 1, using 19 KiB of memory

<table>
<thead>
<tr>
<th>Neighbor</th>
<th>V</th>
<th>AS</th>
<th>MsgRcvd</th>
<th>MsgSent</th>
<th>TblVer</th>
<th>InQ</th>
<th>OutQ</th>
<th>Up/Down</th>
<th>State/PfxRcd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spine01(2001:1:1::1)</td>
<td>4</td>
<td>1</td>
<td>74</td>
<td>68</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00:00:45</td>
<td>1</td>
</tr>
</tbody>
</table>

Total number of neighbors 1

root@Leaf01:~# net show bgp ipv4 unicast
BGP table version is 1, local router ID is 10.0.0.13
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

Network Next Hop Metric LocPrf Weight Path
*>172.16.4.0/24 2001:2::4 0 100 0 i

Displayed 1 routes and 1 total paths

root@Leaf01:~# net show bgp ipv4 unicast 172.16.4.0/24
BGP routing table entry for 172.16.4.0/24
Paths: (1 available, best #1, table default)
Not advertised to any peer
Local
  2001:2::4 from Spine01(2001:1:1::1) (10.0.0.14)
  Origin IGP, metric 0, localpref 100, valid, internal,
  bestpath-from-AS Local, best
  Originator: 10.0.0.14, Cluster list: 10.0.0.11
  AddPath ID: RX 0, TX 5
  Last update: Mon Oct 22 14:25:30 2018

root@Leaf01:~# net show route 172.16.4.0/24
RIB entry for 172.16.4.0/24
===========================
Routing entry for 172.16.4.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 00:01:13 ago
  2001:2::4 (recursive)
  * fe80::a00:27ff:fe5a:84ae, via swp1

FIB entry for 172.16.4.0/24
===========================
172.16.4.0/24 via 169.254.0.1 dev swp1 proto bgp metric 20 onlink
To have only IPv6 global addresses used for route installation into the FRR RIB, you must add an additional route map to the neighbor or peer group statement in the appropriate address family. When the route map command `set ipv6 next-hop prefer-global` is applied to a neighbor, if both a link-local and global IPv6 address are in the BGP update for a prefix, the IPv6 global address is preferred for route installation.

With this additional configuration, the output in the FRR RIB changes in the direct neighbor case, as shown below:

```plaintext
router bgp 1
  bgp router-id 10.0.0.11
  neighbor 2001:2:2::4 remote-as internal
  neighbor 2001:2:2::4 capability extended-nexthop
  !
  address-family ipv4 unicast
  neighbor 2001:2:2::4 route-map GLOBAL in
  exit-address-family
  !
  route-map GLOBAL permit 20
  set ipv6 next-hop prefer-global
  !
```

The resulting FRR RIB output is as follows:

```plaintext
Spine01# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, 
  O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP, 
  T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP, 
  F - PBR, 
  > - selected route, * - FIB route

  B 0.0.0.0/0 [200/0] via 2001:2:2::4, swp2, 00:01:00
  K 0.0.0.0/0 [0/0] via 10.0.2.2, eth0, 1d02h29m
  C>* 10.0.0.9/32 is directly connected, lo, 5d18h32m
  C>* 10.0.2.0/24 is directly connected, eth0, 03:51:31
  B>* 172.16.4.0/24 [200/0] via 2001:2:2::4, swp2, 00:01:00
  C>* 172.16.10.0/24 is directly connected, swp3, 5d18h32m
```

When the route is learned through a route reflector, it appears like this:
BGP add-path

Cumulus Linux supports both BGP add-path RX and BGP add-path TX.

BGP add-path RX

*BGP add-path RX* allows BGP to receive multiple paths for the same prefix. A path identifier is used so that additional paths do not override previously advertised paths. No additional configuration is required for BGP add-path RX.

```
router bgp 1
  bgp router-id 10.0.0.13
  neighbor 2001:1:1::1 remote-as internal
  neighbor 2001:1:1::1 capability extended-nexthop
  address-family ipv6 unicast
    neighbor 2001:1:1::1 activate
    neighbor 2001:1:1::1 route-map GLOBAL in
    exit-address-family
  route-map GLOBAL permit 10
  set ipv6 next-hop prefer-global
Leaf01# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
  O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
  T - Table, V - VNC, V - VNC-Direct, A - Babel, D - SHARP,
  F - PBR,
  > - selected route, * - FIB route
 B  0.0.0.0/0 [200/0] via 2001:2:2::4, 00:00:01
K  0.0.0.0/0 [0/0] via 10.0.2.2, eth0, 3d00h26m
C>* 10.0.0.8/32 is directly connected, lo, 3d00h26m
C>* 10.0.2.0/24 is directly connected, eth0, 03:39:18
C>* 172.16.3.0/24 is directly connected, swp2, 3d00h26m
B>  172.16.4.0/24 [200/0] via 2001:2:2::4 (recursive), 00:00:01
  * via 2001:1:1::1, swp1, 00:00:01
C>* 172.16.10.0/24 is directly connected, swp3, 3d00h26m
```

BGP advertises the add-path RX capability by default. Add-Path TX requires an administrator to enable it. Enabling TX resets the session.

To view the existing capabilities, run `net show bgp neighbor`. The existing capabilities are listed in the subsection *Add Path*, below *Neighbor capabilities*.
The example output above shows that additional BGP paths can be sent and received (TX and RX are advertised). It also shows that the BGP neighbor, fe80::4638:39ff:fe00:5c, supports both.

To view the current additional paths, run `net show bgp <network>`. The example output shows an additional path that has been added by the TX node for receiving. Each path has a unique AddPath ID.
**BGP add-path TX**

AddPath TX allows BGP to advertise more than just the bestpath for a prefix. Consider the following topology:

```
  r8
   | |
   r1 ---- ---- r6
   r2 ---- r7 ---- r5
       | |
       r3 r4
```

In this topology:
- r1 and r2 are in AS 100
- r3 and r4 are in AS 300
- r5 and r6 are in AS 500
- r7 is in AS 700
- r8 is in AS 800
- r7 learns 1.1.1.1/32 from r1, r2, r3, r4, r5, and r6. Among these r7 picks the path from r1 as the bestpath for 1.1.1.1/32

The example below configures the r7 session to advertise the bestpath learned from each AS. In this case, this means a path from AS 100, a path from AS 300, and a path from AS 500. The `net show bgp 1.1.1.1/32` from r7 has "bestpath-from-AS 100" so the user can see what the bestpath is from each AS:

```
cumulus@r7:~$ net add bgp autonomous-system 700
cumulus@r7:~$ net add bgp neighbor 192.0.2.2 addpath-tx-bestpath-per-AS
```

The output below shows the result on r8:
```
cumulus@r8:$ net show bgp 1.1.1.1/32
BGP routing table entry for 1.1.1.1/32
Paths: (3 available, best #3, table Default-IP-Routing-Table)
    Advertised to non peer-group peers:
    r7(10.7.8.1)
    700 100
    10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
        Origin IGP, localpref 100, valid, external
        Community: 1:1
        AddPath ID: RX 2, TX 4
        Last update: Thu Jun 2 00:57:14 2016

    700 300
    10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
        Origin IGP, localpref 100, valid, external
        Community: 3:3
        AddPath ID: RX 4, TX 3
        Last update: Thu Jun 2 00:57:14 2016

    700 500
    10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
        Origin IGP, localpref 100, valid, external, bestpath-from-AS 700,
        best
        Community: 5:5
        AddPath ID: RX 6, TX 2
        Last update: Thu Jun 2 00:57:14 2016
```

The example below shows the results if r7 is configured to advertise all paths to r8:

```
cumulus@r7:$ net add bgp autonomous-system 700
cumulus@r7:$ net add bgp neighbor 192.0.2.2 addpath-tx-all-paths
```

The output below shows the result on r8:
cumulus@r8:~$ net show bgp 1.1.1.1/32
BGP routing table entry for 1.1.1.1/32
Paths: (3 available, best #3, table Default-IP-Routing-Table)
   Advertised to non peer-group peers:
     r7(10.7.8.1)
     700 100
     10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
       Origin IGP, localpref 100, valid, external
       Community: 1:1
       AddPath ID: RX 2, TX 4
       Last update: Thu Jun 2 00:57:14 2016

     700 300
     10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
       Origin IGP, localpref 100, valid, external
       Community: 3:3
       AddPath ID: RX 4, TX 3
       Last update: Thu Jun 2 00:57:14 2016

     700 500
     10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
       Origin IGP, localpref 100, valid, external, bestpath-from-AS 700,
       best
       Community: 5:5
       AddPath ID: RX 6, TX 2
       Last update: Thu Jun 2 00:57:14 2016

**Fast Convergence Design Considerations**

Cumulus Networks strongly recommends the following use of addresses in the design of a BGP-based data center network:

- Set up BGP sessions only using interface-scoped addresses. This allows BGP to react quickly to link failures.
- Use of next hop-self: Every BGP node says that it knows how to forward traffic to the prefixes it is announcing. This reduces the requirement to announce interface-specific addresses and thereby reduces the size of the forwarding table.

When you configure BGP for the neighbors of a given interface, you can specify the interface name instead of its IP address. All the other neighbor command options remain the same.

This is equivalent to BGP peering to the link-local IPv6 address of the neighbor on the given interface. The link-local address is learned via IPv6 neighbor discovery router advertisements.

Consider the following example configuration in the /etc/frr/frr.conf file:

```conf
router bgp 65000
  bgp router-id 10.0.0.1
  neighbor swp1 interface
  neighbor swp1 remote-as internal
  neighbor swp1 next-hop-self

  !
  address-family ipv6
  neighbor swp1 activate
  exit-address-family
```
You create the above configuration with the following NCLU commands:

```bash
cumulus@switch:~$ net add bgp autonomous-system 65000
cumulus@switch:~$ net add bgp router-id 10.0.0.1
cumulus@switch:~$ net add bgp neighbor swp1 interface
cumulus@switch:~$ net add bgp neighbor swp1 remote-as internal
cumulus@switch:~$ net add bgp neighbor swp1 next-hop-self
cumulus@switch:~$ net add bgp ipv6 unicast neighbor swp1 activate
```

By default, Cumulus Linux sends IPv6 neighbor discovery router advertisements. Cumulus Networks recommends you adjust the interval of the router advertisement to a shorter value (net add interface <interface> ipv6 nd ra-interval <interval>) to address scenarios when nodes come up and miss router advertisement processing to relay the neighbor’s link-local address to BGP. The interval is measured in seconds and defaults to 10 seconds.

**Peer Groups to Simplify Configuration**

When a switch has many peers to connect to, the amount of redundant configuration becomes overwhelming. For example, repeating the activate and next-hop-self commands for even 60 neighbors makes for a very long configuration file. To address this problem, you can use peer-group.

Instead of specifying properties of each individual peer, FRRouting allows you to define one or more peer groups and associate all the attributes common to that peer session to a peer group. A peer needs to be attached to a peer group only once, when it then inherits all address families activated for that peer group.

After you attach a peer to a peer group, you need to associate an IP address with the peer group. The following example shows how to define and use peer groups:

```bash
cumulus@switch:~$ net add bgp neighbor tier-2 peer-group
cumulus@switch:~$ net add bgp ipv4 unicast
cumulus@switch:~$ net add bgp neighbor tier-2 activate
cumulus@switch:~$ net add bgp neighbor tier-2 next-hop-self
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 peer-group tier-2
cumulus@switch:~$ net add bgp neighbor 192.0.2.2 peer-group tier-2
```

BGP peer-group restrictions have been replaced with update-groups, which dynamically examine all peers and group them if they have the same outbound policy.

**Configure BGP Dynamic Neighbors**

*BGP dynamic neighbor* provides BGP peering to a group of remote neighbors within a specified range of IPv4 or IPv6 addresses for a BGP peer group. You can configure each range as a subnet IP address.

You configure dynamic neighbors using the `bgp listen range <IP address> peer-group <GROUP>` command. After you configure the dynamic neighbors, a BGP speaker can listen for, and form peer relationships with, any neighbor in the IP address range and mapped to a peer group.

```bash
cumulus@switch:~$ net add bgp autonomous-system 65001
cumulus@switch:~$ net add bgp listen range 10.1.1.0/24 peer-group SPINE
```

To limit the number of dynamic peers, specify the limit in the `bgp listen limit` command (the default value is 100):
Collectively, a sample configuration for IPv4 looks like this:

cumulus@switch:~$ net add bgp autonomous-system 65001
cumulus@switch:~$ net add bgp neighbor SPINE peer-group
 cumulus@switch:~$ net add bgp neighbor SPINE remote-as 65000
 cumulus@switch:~$ net add bgp listen limit 5
 cumulus@switch:~$ net add bgp listen range 10.1.1.0/24 peer-group SPINE

These commands produce an IPv4 configuration that looks like this:

```
router bgp 65001
  neighbor SPINE peer-group
  neighbor SPINE remote-as 65000
  bgp listen limit 5
  bgp listen range 10.1.1.0/24 peer-group SPINE
```

### Configure BGP Peering Relationships across Switches

A BGP peering relationship is typically initiated with the `neighbor x.x.x.x remote-as [internal|external]` command.

Specifying `internal` signifies an iBGP peering; that is, the neighbor only creates or accepts a connection with the specified neighbor if the remote peer AS number matches this BGP AS number.

Specifying `external` signifies an eBGP peering; that is, the neighbor will only create a connection with the neighbor if the remote peer AS number does not match this BGP AS number.

You can make this distinction using the `neighbor` command or the `peer-group` command.

In general, use the following syntax with the `neighbor` command:

```
cumulus@switch:~$ net add bgp neighbor [<IP address>|<BGP peer>|<interface>] remote-as [<value>|internal|external]
```

Some example configurations follow.

To connect to the same AS using the `neighbor` command, modify your configuration similar to the following:

```
cumulus@switch:~$ net add bgp autonomous-system 500
 cumulus@switch:~$ net add bgp neighbor 192.168.1.2 remote-as internal
```

These commands create the following configuration snippet:

```
router bgp 500
  neighbor 192.168.1.2 remote-as internal
```
To connect to a **different AS** using the `neighbor` command, modify your configuration similar to the following:

```
$ net add bgp autonomous-system 500
$ net add bgp neighbor 192.168.1.2 remote-as external
```

These commands create the following configuration snippet:

```
router bgp 500
neighbor 192.168.1.2 remote-as external
```

To connect to the **same AS** using the `peer-group` command, modify your configuration similar to the following:

```
$ net add bgp autonomous-system 500
$ net add bgp neighbor swp1 interface
$ net add bgp neighbor IBGP peer-group
$ net add bgp neighbor IBGP remote-as internal
$ net add bgp neighbor swp1 interface peer-group IBGP
$ net add bgp neighbor 192.0.2.3 peer-group IBGP
$ net add bgp neighbor 192.0.2.4 peer-group IBGP
```

These commands create the following configuration snippet:

```
router bgp 500
neighbor swp1 interface
neighbor IBGP peer-group
neighbor IBGP remote-as internal
neighbor swp1 peer-group IBGP
neighbor 192.0.2.3 peer-group IBGP
neighbor 192.0.2.4 peer-group IBGP
```

To connect to **different AS** using the `peer-group` command, modify your configuration similar to the following:

```
$ net add bgp autonomous-system 500
$ net add bgp neighbor swp2 interface
$ net add bgp neighbor EBGP peer-group
$ net add bgp neighbor EBGP remote-as external
$ net add bgp neighbor 192.0.2.2 peer-group EBGP
$ net add bgp neighbor swp2 interface peer-group EBGP
$ net add bgp neighbor 192.0.2.4 peer-group EBGP
```

These commands create the following configuration snippet:
Configure MD5-enabled BGP Neighbors

The following sections outline how to configure an MD5-enabled BGP neighbor. Each process assumes that FRRouting is used as the routing platform, and consists of two switches (AS 65011 and AS 65020), connected by the link 10.0.0.100/30, with the following configurations:

```
router bgp 500
neighbor swp2 interface
neighbor EBGP peer-group
neighbor EBGP remote-as external
neighbor 192.0.2.2 peer-group EBGP
neighbor swp2 peer-group EBGP
neighbor 192.0.2.4 peer-group EBGP
```

```
cumulus@leaf01:~$ net show bgp summary
show bgp ipv4 unicast summary
============================= 
BGP router identifier 10.0.0.11, local AS number 65011 vrf-id 0
BGP table version 6
RIB entries 11, using 1320 bytes of memory
Peers 2, using 36 KiB of memory
Peer groups 1, using 56 bytes of memory
Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down State/PfxRcd
spine01(swp51)  4 65020   93587   93587        0    0    0 1d02h00m 3 3
spine02(swp52)  4 65020   93587   93587        0    0    0 1d02h00m 3 3
Total number of neighbors 2

show bgp ipv6 unicast summary
============================= 
No IPv6 neighbor is configured
```
switch2

cumulus@spine01:~$ net show bgp summary
show bgp ipv4 unicast summary
=============================  
BGP router identifier 10.0.0.21, local AS number 65020 vrf-id 0  
BGP table version 5  
RIB entries 9, using 1080 bytes of memory  
Peers 4, using 73 KiB of memory  
Peer groups 1, using 56 bytes of memory  
Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down          
State/PfxRcd
leaf01(swp1)    4 65011     782     782        0    0    0 00:12:54    
2
leaf02(swp2)    4 65012     781     781        0    0    0 00:12:53    
2
swp3            4     0       0       0        0    0    0 never    Idle
swp4            4     0       0       0        0    0    0 never    Idle

Total number of neighbors 4

show bgp ipv6 unicast summary
=============================  
No IPv6 neighbor is configured

To manually configure an MD5-enabled BGP neighbor:

1. SSH into leaf01.
2. Configure the password for the neighbor:

        cumulus@leaf01:~$ net add bgp neighbor 10.0.0.102 password mypassword

3. Confirm the configuration has been implemented with the net show bgp summary command:
4. SSH into spine01.
5. Configure the password for the neighbor:

    cumulus@spine01:~$ net add bgp neighbor 10.0.0.101 password mypassword

6. Confirm the configuration has been implemented with the `net show bgp summary` command:
cumulus@spine01:~$ net show bgp summary
show bgp ipv4 unicast summary
============================================
BGP router identifier 10.0.0.21, local AS number 65020 vrf-id 0
BGP table version 5
RIB entries 9, using 1080 bytes of memory
Peers 4, using 73 KiB of memory
Peer groups 1, using 56 bytes of memory
Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ
Up/Down State/PfxRcd
leaf01(swp1) 4 65011 782 782 0 0 0 00:12:54
leaf02(swp2) 4 65012 781 781 0 0 0 00:12:53
swp3 4 0 0 0 0 0 never
Idle
swp4 4 0 0 0 0 0 never
Idle
Total number of neighbors 4

show bgp ipv6 unicast summary
============================================
No IPv6 neighbor is configured

The MD5 password configured against a BGP listen-range peer-group (used to accept and create dynamic BGP neighbors) is not enforced. This means that connections are accepted from peers that do not specify a password.

**Configure eBGP Multihop**

The eBGP Multihop option lets you use BGP to exchange routes with an external peer that is more than one hop away.

1. To establish a connection between two eBGP peers that are not directly connected:

```
cumulus@leaf02:mgmt-vrf:~$ net add bgp neighbor <ip> remote-as external
cumulus@leaf02:mgmt-vrf:~$ net add bgp neighbor <ip> ebgp-multihop
```

2. Confirm the configuration with the `net show bgp neighbor <ip>` command:
Configure BGP TTL Security

The steps below show how to configure BGP TTL security on Cumulus Linux using a leaf (leaf01) and spine (spine01) for the example output:

1. SSH into leaf01 and configure it for TTL security:

```
cumulus@leaf01:~$ net add bgp autonomous-system 65000
```
```
cumulus@leaf01:~$ net add bgp neighbor [spine01-IP] ttl-security hops [value]
```

2. SSH into spine01 and configure it for TTL security:
3. Confirm the configuration with the `show ip bgp neighbor` command:

```
cumulus@spine01:mgmt-vrf:$ net show bgp neighbor swp1
BGP neighbor on swp1: fe80::4638:39ff:fe00:5b, remote AS 65011, local AS 65020, external link
Hostname: leaf01
BGP version 4, remote router ID 10.0.0.11
BGP state = Established, up for 00:10:45
Hold time is 9, keepalive interval is 3 seconds
Neighbor capabilities:
  4 Byte AS: advertised and received
  AddPath:
    IPv4 Unicast: RX advertised IPv4 Unicast and received
  Extended next-hop: advertised and received
    Address families by peer:
      IPv4 Unicast
  Route refresh: advertised and received(old & new)
  Address Family IPv4 Unicast: advertised and received
  Hostname Capability: advertised (name: spine01,domain name: n/a) received (name: leaf01,domain name: n/a)
  Graceful Restart Capability: advertised and received
    Remote Restart timer is 120 seconds
    Address families by peer:
      none
Graceful restart informations:
  End-of-RIB send: IPv4 Unicast
  End-of-RIB received: IPv4 Unicast
Message statistics:
  Inq depth is 0
  Outq depth is 0
    Sent   Rcvd
  Opens:   46    2
  Notifications:  41    0
  Updates:   38    34
  Keepalives: 49334  49331
  Route Refresh:  0    0
  Capability:  0    0
  Total:  49459  49367
Minimum time between advertisement runs is 0 seconds
For address family: IPv4 Unicast
  Update group 1, subgroup 1
  Packet Queue length 0
  Community attribute sent to this neighbor(all)
  3 accepted prefixes
Connections established 2; dropped 1
  Last reset 00:17:37, due to NOTIFICATION sent (Hold Timer Expired)
External BGP neighbor may be up to 1 hops away.
Local host: fe80::4638:39ff:fe00:5c, Local port: 35564
Foreign host: fe80::4638:39ff:fe00:5b, Foreign port: 179
Nexthop: 10.0.0.21
Nexthop global: fe80::4638:39ff:fe00:5c
Nexthop local: fe80::4638:39ff:fe00:5c
BGP connection: shared network
BGP Connect Retry Timer in Seconds: 10
Read thread: on  Write thread: on
```

Configure Graceful BGP Shutdown

To reduce packet loss during planned maintenance of a router or link, you can configure graceful BGP shutdown, which forces traffic to route

```
cumulus@spine01:~$ net add bgp autonomous-system 65001
cumulus@spine01:~$ net add bgp neighbor [leaf01-IP] ttl-security hops [value]
```
around the node.

To configure graceful BGP shutdown for the current node, run the `net add bgp graceful-shutdown` command:

```
cumulus@spine01:~$ net add bgp graceful-shutdown
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```

When configured, the `graceful-shutdown` community is added to all paths from eBGP peers and the `local-pref` for that route is set to 0. An example configuration is shown below:

```
cumulus@switch:~$ show ip bgp 10.1.3.0/24
BGP routing table entry for 10.1.3.0/24
Paths: (2 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
  bottom0(10.1.2.2)
    30 20
    10.1.1.2 (metric 10) from top1(10.1.1.2) (10.1.1.2)
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS 30, best
      Community: 99:1
      AddPath ID: RX 0, TX 52
      Last update: Mon Sep 18 17:01:18 2017

  20
    10.1.2.2 from bottom0(10.1.2.2) (10.1.1.1)
      Origin IGP, metric 0, localpref 0, valid, external, bestpath-from-AS
      Community: 99:1 graceful-shutdown
      AddPath ID: RX 0, TX 2
      Last update: Mon Sep 18 17:01:18 2017
```

To disable graceful shutdown for the current node, run the `net del bgp graceful-shutdown` command:

```
cumulus@spine01:~$ net del bgp graceful-shutdown
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```

**Configuration Tips**

**BGP Advertisement Best Practices**

Limiting the exchange of routing information at various parts in the network is a best practice you should follow. The following image illustrates one way you can do so in a typical Clos architecture:
Multiple Routing Tables and Forwarding

You can run multiple routing tables (one for in-band/data plane traffic and one for out-of-band/management plane traffic) on the same switch using management VRF (multiple routing tables and forwarding).

```
BGP and static routing (IPv4 and IPv6) are supported within a VRF context. For more information, refer to Virtual Routing and Forwarding - VRF.
```

BGP Community Lists

You can use community lists to define a BGP community to tag one or more routes. You can then use the communities to apply route policy on either egress or ingress.

The BGP community list can be either standard or expanded. The standard BGP community list is a pair of values (such as 100:100) that can be tagged on a specific prefix and advertised to other neighbors or applied on route ingress. Alternately, it can be one of four BGP default communities:

- `internet`: a BGP community that matches all routes
- `local-AS`: a BGP community that restrict routes to your confederation's sub-AS
- `no-advertise`: a BGP community that isn't advertised to anyone
- `no-export`: a BGP community that isn't advertised to the eBGP peer

An expanded BGP community list takes a regular expression of communities matches the listed communities.

When the neighbor receives the prefix, it examines the community value and takes action accordingly, such as permitting or denying the community member in the routing policy.

Here is an example of a standard community list filter:

```
cumulus@switch:~$ net add routing community-list standard COMMUNITY1 permit 100:100
```

You can apply the community list to a route map to define the routing policy:
Additional Default Settings

Other settings not discussed in detail in this chapter that are enabled by default, include the following:

- `bgp deterministic-med`, which ensures path ordering no longer impacts bestpath selection.
- `bgp show-hostname`, which displays the hostname in show command output.
- `bgp network import-check`, which enables the advertising of the BGP network in IGP.

Configure BGP Neighbor maximum-prefixes

The maximum number of route announcements, or prefixes, allowed by a BGP neighbor can be configured using the FRR `maximum-prefixes` command:

```
frr(config)# neighbor <peer> maximum-prefix <number>
```

Troubleshooting

To troubleshoot BGP, you can view the summary of neighbors to which the switch is connected and see information about these connections. The following example shows sample command output:

```
cumulus@switch:~$ net show bgp summary
show bgp ipv4 unicast summary
==================================
BGP router identifier 10.0.0.11, local AS number 65011 vrf-id 0
BGP table version 8
RIB entries 11, using 1320 bytes of memory
Peers 2, using 36 KiB of memory
Peer groups 1, using 56 bytes of memory
Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
spine01(swp51)  4 65020     549     551        0    0    0 00:09:03  3
spine02(swp52)  4 65020     548     550        0    0    0 00:09:02  3
Total number of neighbors 2

cumulus@switch:~$ net show bgp ipv6 unicast summary
==================================
No IPv6 neighbor is configured
```

To determine if the sessions above are iBGP or eBGP sessions, look at the ASNs.

It is also useful to view the routing table as defined by BGP:

```
cumulus@switch:~$ net show bgp ipv4 unicast route
```
cumulus@switch:~$ net show bgp ipv4
ERROR: Command not found
Use 'net help KEYWORD(s)' to list all options that use KEYWORD(s)
cumulus@leaf01:~$ net show bgp ipv4
          unicast : add help text
cumulus@leaf01:~$ net show bgp ipv4 unicast
BGP table version is 8, local router ID is 10.0.0.11
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Hop</th>
<th>Metric</th>
<th>LocPrf</th>
<th>Weight</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>*&gt; 10.0.0.11/32</td>
<td>0.0.0.0</td>
<td>0</td>
<td></td>
<td>32768</td>
<td></td>
</tr>
<tr>
<td>*= 10.0.0.12/32</td>
<td>swp52</td>
<td>0</td>
<td>65020</td>
<td>65012</td>
<td></td>
</tr>
<tr>
<td>*&gt;</td>
<td>swp51</td>
<td>0</td>
<td>65020</td>
<td>65012</td>
<td></td>
</tr>
<tr>
<td>*&gt; 10.0.0.21/32</td>
<td>swp51</td>
<td>0</td>
<td>65020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*&gt; 10.0.0.22/32</td>
<td>swp52</td>
<td>0</td>
<td>65020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*&gt; 172.16.1.0/24</td>
<td>0.0.0.0</td>
<td>0</td>
<td>32768</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>*= 172.16.2.0/24</td>
<td>swp52</td>
<td>0</td>
<td>65020</td>
<td>65012</td>
<td>i</td>
</tr>
<tr>
<td>*&gt;</td>
<td>swp51</td>
<td>0</td>
<td>65020</td>
<td>65012</td>
<td>i</td>
</tr>
</tbody>
</table>

Total number of prefixes 6

To show a more detailed breakdown of a specific neighbor, run the net show bgp neighbor <neighbor> command:

cumulus@switch:~$ net show bgp neighbor swp51
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020, local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
BGP version 4, remote router ID 10.0.0.21
BGP state = Established, up for 00:11:30
Last read 00:00:00, Last write 00:11:26
Hold time is 3, keepalive interval is 1 seconds
Configured hold time is 3, keepalive interval is 1 seconds
Neighbor capabilities:
  4 Byte AS: advertised and received
AddPath:
  IPv4 Unicast: RX advertised IPv4 Unicast and received
Extended nexthop: advertised and received
  Address families by peer:
    IPv4 Unicast
Route refresh: advertised and received(old & new)
Address family IPv4 Unicast: advertised and received
Hostname Capability: advertised and received
Graceful Restart Capability: advertised and received
  Remote Restart timer is 120 seconds
  Address families by peer:
    none
Graceful restart informations:
  End-of-RIB send: IPv4 Unicast
  End-of-RIB received: IPv4 Unicast
Message statistics:
Inq depth is 0
Outq depth is 0

<table>
<thead>
<tr>
<th></th>
<th>Sent</th>
<th>Rcvd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opens</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Notifications:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Updates:</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Keepalives:</td>
<td>690</td>
<td>689</td>
</tr>
<tr>
<td>Route Refresh:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Capability:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total:</td>
<td>698</td>
<td>696</td>
</tr>
</tbody>
</table>

Minimum time between advertisement runs is 0 seconds
For address family: IPv4 Unicast
fabric peer-group member
Update group 1, subgroup 1
Packet Queue length 0
Community attribute sent to this neighbor (both)
Inbound path policy configured
Outbound path policy configured
Incoming update prefix filter list is *dc-leaf-in
Outgoing update prefix filter list is *dc-leaf-out
3 accepted prefixes
Connections established 1; dropped 0
Last reset never
Local host: fe80::4638:39ff:fe00:5b, Local port: 48424
Foreign host: fe80::4638:39ff:fe00:5c, Foreign port: 179
Nexthop: 10.0.0.11
Nexthop global: fe80::4638:39ff:fe00:5b
Nexthop local: fe80::4638:39ff:fe00:5b
BGP connection: shared network
BGP Connect Retry Timer in Seconds: 3
Estimated round trip time: 3 ms
Read thread: on    Write thread: off

To see details of a specific route, such as from where it is received and where it is sent, run the `net show bgp <ip address/prefix>` command:

```bash
cumulus@leaf01:~$ net show bgp 10.0.0.11/32
BGP routing table entry for 10.0.0.11/32
Paths: (1 available, best #1, table Default-IP-Routing-Table)
    Advertised to non peer-group peers:
        spine01(swp51) spine02(swp52)
    Local
        0.0.0.0 from 0.0.0.0 (10.0.0.11)
        Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, bestpath-from-AS Local, best
        AddPath ID: RX 0, TX 9
```
The above example shows that the routing table prefix seen by BGP is 10.0.0.11/32, that this route is advertised to two neighbors, and that it is not heard by any neighbors.

**Log Neighbor State Changes**

To log the changes that a neighbor goes through so that you can troubleshoot issues associated with that neighbor, run the `log-neighbor-changes` command, which is enabled by default.

The output is sent to the specified log file, usually `/var/log/frr/bgpd.log`, and looks like this:

```plaintext
2016/07/08 10:12:06.572827 BGP: %NOTIFICATION: sent to neighbor 10.0.0.2
   6/3 (Cease/Peer Unconfigured) 0 bytes
2016/07/08 10:12:06.572954 BGP: Notification sent to neighbor 10.0.0.2:
   type 6/3
2016/07/08 10:12:16.682071 BGP: %ADJCHANGE: neighbor 192.0.2.2 Up
2016/07/08 10:12:16.682660 BGP: %ADJCHANGE: neighbor 10.0.0.2 Up
```

**Troubleshoot Link-local Addresses**

To verify that `frr` learned the neighboring link-local IPv6 address via the IPv6 neighbor discovery router advertisements on a given interface, run the `show interface <if-name>` command. If `ipv6 nd suppress-ra` is not enabled on both ends of the interface, then Neighbor address(s): has the other end's link-local address. That is the address that BGP uses when BGP is enabled on that interface.

IPv6 route advertisements (RAs) are automatically enabled on an interface with IPv6 addresses; the `no ipv6 nd suppress-ra` command is not needed for BGP unnumbered.

Use `vtysh` to verify the configuration:
cumulus@switch:~$ sudo vtysh

Hello, this is FRRouting (version 4.0+cl3u8).

R7# show interface swp1
Interface swp1 is up, line protocol is up
Link ups: 0 last: (never)
Link downs: 0 last: (never)
PTM status: disabled
vrf: Default-IP-Routing-Table
index 4 metric 0 mtu 1500
flags: <UP,BROADCAST,RUNNING,MULTICAST>
HWaddr: 44:38:39:00:00:5c
inet6 fe80::4638:39ff:fe00:5c/64
ND advertised reachable time is 0 milliseconds
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 10 seconds
ND router advertisements lifetime tracks ra-interval
ND router advertisement default router preference is medium
Hosts use stateless autoconfig for addresses.
Neighbor address(s):
inet6 fe80::4638:39ff:fe00:5b/128

Instead of the IPv6 address, the peering interface name is displayed in the `show ip bgp summary` command and wherever else applicable:

cumulus@switch:~$ net show bgp summary
BGP router identifier 10.0.0.21, local AS number 65020 vrf-id 0
BGP table version 15
RIB entries 17, using 2040 bytes of memory
Peers 6, using 97 KiB of memory
Peer groups 1, using 56 bytes of memory

<table>
<thead>
<tr>
<th>Neighbor</th>
<th>V</th>
<th>AS</th>
<th>MsgRcvd</th>
<th>MsgSent</th>
<th>TblVer</th>
<th>InQ</th>
<th>OutQ</th>
<th>Up/Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaf01(swpl)</td>
<td>4</td>
<td>65011</td>
<td>2834</td>
<td>2843</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:21:35</td>
</tr>
<tr>
<td>leaf02(swpl)</td>
<td>4</td>
<td>65012</td>
<td>2834</td>
<td>2844</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:21:36</td>
</tr>
<tr>
<td>leaf03(swpl)</td>
<td>4</td>
<td>65013</td>
<td>2834</td>
<td>2843</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:21:35</td>
</tr>
<tr>
<td>leaf04(swpl)</td>
<td>4</td>
<td>65014</td>
<td>2834</td>
<td>2844</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:21:36</td>
</tr>
<tr>
<td>edge01(swpl)</td>
<td>4</td>
<td>65051</td>
<td>8509</td>
<td>8505</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:21:37</td>
</tr>
<tr>
<td>edge01(swpl)</td>
<td>4</td>
<td>65051</td>
<td>8506</td>
<td>8503</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>02:21:35</td>
</tr>
</tbody>
</table>

Total number of neighbors 6
Most of the `net show` commands can take the interface name instead of the IP address.

```bash
cumulus@leaf01:~$ net show bgp neighbor
fabric : BGP neighbor or peer-group
swp51  : BGP neighbor or peer-group
swp52  : BGP neighbor or peer-group
<ENTER>
```

```bash
cumulus@leaf01:~$ net show bgp neighbor swp51
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020, local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Connect
  Last read 20:16:21, Last write 20:55:51
  Hold time is 30, keepalive interval is 10 seconds
  Configured hold time is 30, keepalive interval is 10 seconds
Message statistics:
  Inq depth is 0
  Outq depth is 0
  Sent     Rcvd
  Opens:           1          1
  Notifications:   1          0
  Updates:         7          6
  Keepalives:     2374       2373
  Route Refresh:  0           0
  Capability: 0     0
  Total:          2383       2380
Minimum time between advertisement runs is 5 seconds
For address family: IPv4 Unicast
fabric peer-group member
Not part of any update group
Community attribute sent to this neighbor(both)
Inbound path policy configured
Outbound path policy configured
Incoming update prefix filter list is *dc-leaf-in
Outgoing update prefix filter list is *dc-leaf-out
0 accepted prefixes
Connections established 1; dropped 1
Last reset 20:16:20, due to NOTIFICATION sent (Cease/Other Configuration Change)
BGP Connect Retry Timer in Seconds: 3
Next connect timer due in 1 seconds
Read thread: on  Write thread: on
```

**Enable Read-only Mode**

As BGP peers are established and updates are received, prefixes might be installed in the RIB and advertised to BGP peers even though the information from all peers has not yet been received and processed. Depending on the timing of the updates, prefixes might be installed and...
propagated through BGP, and then immediately withdrawn and replaced with new routing information. Read-only mode minimizes this BGP route churn in both the local RIB and with BGP peers.

Enable read-only mode to reduce CPU and network usage when you restart the BGP process, or when you issue the `clear ip bgp` command. Because intermediate best paths are possible for the same prefix as peers get established and start receiving updates at different times, read-only mode is particularly useful in topologies where BGP learns a prefix from many peers and the network has a high number of prefixes.

To enable read-only mode, run the `net add bgp update-delay <max-delay in 0-3600 seconds> [establish-wait in 1-3600 seconds>]` command.

The following example command enables read-only mode, sets the `max-delay` timer to 300 seconds and the `establish-wait` timer to 90 seconds.

```
cumulus@switch:$ net add bgp update-delay 300 90
```

The default value for `max-delay` is 0, which disables read-only mode. The `establish-wait` option is optional; however, if specified, the `establish-wait` option must be shorter than the `max-delay`.

Read-only mode begins as soon as the first peer reaches its established state and the `max-delay` timer starts, and continues until either of the following two conditions are met:

- All the configured peers (except the shutdown peers) have sent an explicit EOR (End-Of-RIB) or an implicit EOR. The first keep-alive after BGP has reached the established state is considered an implicit EOR.
  - If you specify the `establish-wait` option, BGP only considers peers that have reached the established state from the moment the `max-delay` timer starts until the `establish-wait` period ends.

  The minimum set of established peers for which EOR is expected are the peers that are established during the `establish-wait` window, not necessarily all the configured neighbors.

- The timer reaches the configured `max-delay`.

While in read-only mode, BGP does not run best-path or generate any updates to its peers.

To show information about the state of the update delay, run the `show bgp summary` command.

### Apply a Route Map for Route Updates

There are two ways you can apply route maps for BGP:

- By filtering routes from BGP into Zebra
- By filtering routes from Zebra into the Linux kernel

### Filter Routes from BGP into Zebra

You can apply a route map on route updates from BGP to Zebra. All the applicable match operations are allowed, such as match on prefix, next hop, communities, and so on. Set operations for this attach-point are limited to metric and next hop only. Any operation of this feature does not affect BGP's internal RIB.

Both IPv4 and IPv6 address families are supported. Route maps work on multi-paths; however, the metric setting is based on the best path only.

To apply a route map to filter route updates from BGP into Zebra, run the following command:

```
cumulus@switch:$ net add bgp table-map <route-map-name>
```

### Filter Routes from Zebra into the Linux Kernel

To apply a route map to filter route updates from Zebra into the Linux kernel, run the following command:
Protocol Tuning

Converge Quickly On Link Failures

In the Clos topology, we recommend that you only use interface addresses to set up peering sessions. This means that when the link fails, the BGP session is torn down immediately, triggering route updates to propagate through the network quickly. This requires the following commands be enabled for all links: `link-detect` and `ttl-security hops <hops>`. `ttl-security hops` specifies how many hops away the neighbor is. For example, in a Clos topology, every peer is at most 1 hop away.

Here is an example:

```shell
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 ttl-security hops 1
```

Converge Quickly On Soft Failures

It is possible that the link is up but the neighboring BGP process is hung or has crashed. If a BGP process hangs or crashes, the FRRouting `watc hrr` daemon, which monitors the various FRRouting daemons, attempts to restart it. BGP itself has a keepalive interval that is exchanged between neighbors. By default, this keepalive interval is set to 3 seconds. You can increase the interval to a higher number, which decreases CPU load, especially in the presence of a lot of neighbors. The keepalive interval is the periodicity with which the keepalive message is sent. The hold time specifies how many keepalive messages can be lost before the connection is considered invalid. It is usually set to three times the keepalive time and defaults to 9 seconds. The following example shows how to change these timers:

```shell
cumulus@switch:~$ net add bgp neighbor swp51 timers 10 30
```

The following snippet shows that the default values have been modified for this neighbor:

```shell
cumulus@switch:~$ net show bgp neighbor swp51
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020, local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Connect
  Last read 00:00:13, Last write 00:39:43
  Hold time is 30, keepalive interval is 10 seconds
  Configured hold time is 30, keepalive interval is 10 seconds
...
```

Reconnect Quickly

A BGP process attempts to connect to a peer after a failure (or on startup) every `connect-time` seconds. By default, this is 10 seconds. To modify this value, run the following command:
You must specify this command for each neighbor.

**Advertisement Interval**

BGP by default chooses stability over fast convergence. This is very useful when routing for the Internet. For example, unlike link-state protocols, BGP typically waits for a duration of `advertisement-interval` seconds between sending consecutive updates to a neighbor. This ensures that an unstable neighbor flapping routes are not propagated throughout the network. By default, this is set to zero seconds for both eBGP and iBGP sessions, which allows for very fast convergence. You can modify this as follows:

```
cumulus@switch:~$ net add bgp neighbor swp51 advertisement-interval 5
```

The following output shows the modified value:

```
cumulus@switch:~$ net show bgp neighbor swp51
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020, local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
BGP version 4, remote router ID 0.0.0.0
BGP state = Connect
Last read 00:04:37, Last write 00:44:07
Hold time is 30, keepalive interval is 10 seconds
Configured hold time is 30, keepalive interval is 10 seconds
Message statistics:
    Inq depth is 0
    Outq depth is 0
    Sent     Rcvd
    Opens: 1       1
    Notifications: 1   0
    Updates: 7       6
    Keepalives: 2374   2373
    Route Refresh: 0    0
    Capability: 0      0
    Total: 2383    2380
Minimum time between advertisement runs is 5 seconds
...
```

This command is not supported with peer-groups.

See this [IETF draft](https://tools.ietf.org/html/draft-ietf-idr-ttl-security-08) for more details on the use of this value.

**Caveats and Errata**

**ttl-security Issue**

Enabling `ttl-security` does not cause the hardware to be programmed with the relevant information. This means that frames will come up to
the CPU and be dropped there. It is recommended that you use the `net add acl` command to explicitly add the relevant entry to hardware.

For example, you can configure a file, such as `/etc/cumulus/acl/policy.d/01control_plane_bgp.rules`, with a rule like this for TTL:

```
INGRESS_INTF = swp1
INGRESS_CHAIN = INPUT, FORWARD

[iptables]
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p tcp --dport bgp -m
  ttl --ttl 255 POLICE --set-mode pkt --set-rate 2000 --set-burst 1000
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p tcp --dport bgp DROP
```

For more information about ACLs, see Netfilter (ACLs).

**BGP Dynamic Capabilities not Supported**

Dynamic capabilities, which enable BGP to renegotiate a new feature for an already established peer, are not supported in Cumulus Linux.

**Related Information**

- Bidirectional forwarding detection (BFD) and BGP
- Wikipedia entry for BGP (includes list of useful RFCs)
- FRR BGP documentation
- IETF draft discussing BGP use within data centers
- RFC 1657, Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2
- RFC 1997, BGP Communities Attribute
- RFC 2385, Protection of BGP Sessions via the TCP MD5 Signature Option
- RFC 2439, BGP Route Flap Damping
- RFC 2545, Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing
- RFC 2918, Route Refresh Capability for BGP-4
- RFC 4271, A Border Gateway Protocol 4 (BGP-4)
- RFC 4360, BGP Extended Communities Attribute
- RFC 4456, BGP Route Reflection – An Alternative to Full Mesh Internal BGP (iBGP)
- RFC 4760, Multiprotocol Extensions for BGP-4
- RFC 5004, Avoid BGP Best Path Transitions from One External to Another
- RFC 5065, Autonomous System Confederations for BGP
- RFC 5291, Outbound Route Filtering Capability for BGP-4
- RFC 5492, Capabilities Advertisement with BGP-4
- RFC 5549, Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop
- RFC 6793, BGP Support for Four-Octet Autonomous System (AS) Number Space
- RFC 7911, Advertisement of Multiple Paths in BGP
- draft-walton-bgp-hostname-capability-02, Hostname Capability for BGP