BORDER GATEWAY PROTOCOL

March 28, 2019

Border Gateway Protocol (BGP)

Border Gateway Protocol, a.k.a, BGP is a standard exterior gateway protocol developed for exchanging routing and reachability information between Autonomous Systems, a collection of IP routing prefixes managed by a single administrative entity. BGP makes routing decisions based on paths and network policies and is mainly used by Service Providers. BGP can also be used for routing within an autonomous system using its flavor iBGP, which is less popular than other IGP protocols such as OSPF, EIGRP, etc.

Learning routes from BGP simplifies enterprise configuration and integration with Secure Vector Routing. BGP over SVR enables easy distribution of service route information when traffic between 128T peers flows through the internet and adds the security of SVR to the BGP peering.

Prerequisites:

This section presumes that the reader has a running 128T system and wants to add configuration to support and configure BGP. The running 128T system includes configuration for basic platform functionality (e.g., router, node, device-interface, network-interface) and basic routing configuration (e.g., tenants, services, etc.). Refer to the 128T data model and the Configuration Guide for a better understanding about the 128T networking.

BGP Configuration when peer is non-128T:

The BGP configuration exists in the routing configuration container in the 128T data model. For any routing configuration, static or dynamic, a default routing instance called **`default-instance`**, must be defined in the 128T configuration template.

Assuming that the BGP is configured on the peering router with router ID as 1.1.1.1, as AS 6000 and the 128T router as the BGP peer, we can configure BGP on our 128T using the following commands:

```
admin@branchoffice1.seattlesite1# config auth
admin@branchoffice1.seattlesite1 (authority)# router seattlesite1
admin@branchoffice1.seattlesite1 (router[name=seattlesite1])# routing
default-instance
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# routing-
protocol bgp
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# local-as 100
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# router-id
1.1.1.128
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# address-family
ipv4-unicast
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#
exit
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# neighbor
1.1.1.1
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=1.1.1.1])#
neighbor-as 6000
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=1.1.1.1])#
address-family ipv4-unicast
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#
next-hop-self true
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#exit
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=1.1.1.1])# exit
```

Under the default routing instance, specify the routing protocol as BGP using `routing-protocol bgp`. Once inside the routing-protocol element, specify the Local-AS using `local-as <>` and the router-ID using `router-id <>`. Now, specify the address family as IPv4- Unicast.
Next, we configure the neighbor by specifying the neighbor address using `neighbor <>` and the neighbor AS using `neighbor-as <>`. Under the neighbor element configure the address family using `address-family ipv4-unicast` and enable `next-hop-self` as `true`.

Note: *During advertisement, the non-directly connected routers to an external peer need to learn how to reach an advertised route. To provide this information to the non-directly connected as well as iBGP peers,* ***next-hop-self*** *command is used.*

Note: *For BGP neighbor configuration, the* `local-as` *command (under the neighbor instance) can be leveraged when you want to use a specific AS number for your router while peering with that particular neighbor. Note that this AS number* ***cannot*** *be same as the actual AS number that is configured directly under the routing-protocol instance. Example below indicates the difference in red.*

```
admin@branchoffice1.seattlesite1# config auth
admin@branchoffice1.seattlesite1 (authority)# router seattlesite1
admin@branchoffice1.seattlesite1 (router[name=seattlesite1])# routing
default-instance
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# routing-
protocol bgp
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# local-as 100
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# router-id
1.1.1.128
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# address-
family ipv4-unicast
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#
exit
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# neighbor
1.1.1.1
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=1.1.1.1])#
neighbor-as 6000
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=1.1.1.1])#
local-as 62000
```

Go up to the protocol level and issue a `show` command to view the BGP config prior to committing any changes. An ideal BGP configuration must look like:

```
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# show
type            bgp
local-as        100
router-id       1.1.1.128

address-family  ipv4-unicast
    afi-safi  ipv4-unicast

exit

neighbor        1.1.1.1
    neighbor-address  1.1.1.1
    neighbor-as       6000
    shutdown          false


    address-family    ipv4-unicast
        afi-safi      ipv4-unicast
        next-hop-self true
    exit
exit
```

To advertise routes to BGP, configure the network to be advertised under the address-family of the router. You may also apply a policy to the advertised route using `policy <>` command.

Note: *You can configure the policies directly under the authority level routing configuration, i.e.,* ***config>authority>routing> filters/policies****. You must create a filter to match the desired traffic and then create a policy based on that filter. Refer to the filters/policies guide for detailed information.*

```
admin@branchoffice1.seattlesite1# config auth
admin@branchoffice1.seattlesite1 (authority)# router seattlesite1
admin@branchoffice1.seattlesite1 (router[name=seattlesite1])# routing
default-instance
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# routing-
protocol bgp
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# address-
family ipv4-unicast
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#
network 172.16.255.0/30
admin@branchoffice1.seattlesite1 (network[network-
address=172.16.255.0/30])# policy allow
admin@branchoffice1.seattlesite1 (network[network-
address=172.16.255.0/30])# exit
```

To redistribute connected, static, service and/or OSPF routes specifying `redistribute <connected/static/service/ospf>` within the routing protocol element of the default-routing instance.

```
admin@branchoffice1.seattlesite1# config auth
admin@branchoffice1.seattlesite1 (authority)# router seattlesite1
admin@branchoffice1.seattlesite1 (router[name=seattlesite1])# routing
default-instance
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# routing-
protocol bgp
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# redistribute
connected
```

While configuring iBGP, you may need to enable the ***Route Reflector*** option to facilitate easy learning of routes. Your 128T can be configured as a route reflector client for a particular neighbor under the specific neighbor configuration.

```
admin@branchoffice1.seattlesite1# config auth
admin@branchoffice1.seattlesite1 (authority)# router seattlesite1
admin@branchoffice1.seattlesite1 (router[name=seattlesite1])# routing
default-instance
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# routing-
protocol bgp
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# neighbor
1.1.1.1
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=1.1.1.1])#
address-family ipv4-unicast
dmin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#
route-reflector client true
dmin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])# exit
```

```
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])# show
afi-safi         ipv4-unicast
next-hop-self    true

route-reflector
    client  true
exit
```

When configuring iBGP, the ***Confederation*** feature may come in handy when dealing with an enormous autonomous system. This feature allows you to break up the AS into smaller sub-autonomous systems. Confederation can be directly configured under the routing protocol element. Here, 65535 is the ***confederation identifier AS number*** and, 1100 and 2200 are the ***member AS*** numbers of that confederation AS.

```
admin@branchoffice1.seattlesite1# config auth
admin@branchoffice1.seattlesite1 (authority)# router seattlesite1
admin@branchoffice1.seattlesite1 (router[name=seattlesite1])# routing
default-instance
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# routing-
protocol bgp
*admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])#
confederation identifier 65535
*admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])#
confederation member-as 1100
*admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])#
confederation member-as 2200
*admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# exit
```

```
*admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# show
type              bgp
local-as          100
router-id         10.128.128.2

confederation
    identifier    65535
    member-as     1100
    member-as     2200
exit
```

BGP over SVR:

BGP over SVR can be used when peering with a 128T. This provides the benefit of Secure Vector Routing for all BGP traffic flowing to-and-from the 128T peers.

To begin configuring BGP over SVR, ensure that the running 128T system has configuration for basic platform functionality.

Next, we need to configure a *routing interface*, which is similar to a loopback interface on traditional routers. Unlike normal loopback BGP peering, this IP address does not need to be routable on the transport network as it will never see the wire. A BGP peering will be created on this interface. The conductor will then trigger on this and configure a few more pieces (autogenerated) to activate BGP over SVR, such as:

- _bgp_speaker_ tenant
- Auto-generated BGP services and service routes

- Router Peers

Those peerings will then be protected by SVR.

Note: *One must use a conductor for BGP over SVR as a manual configuration is unsupported.*

Note: *If the WAN interface of your 128T or the interface facing the 128T-BGP peer is already part of a neighborhood, then ensure that their topology type (network interface >neighborhoods >topology) is such that it allows the routers to form peering relationships for the auto-generated peer service routes, e.g., mesh-mesh, mesh-hub, mesh-spoke or hub-spoke. For more information on Neighborhoods, check out Interchange!*

Next, configure a BGP instance with the router's local AS and a router id which matches the self-routing interface IP. For each BGP over SVR peer, use a neighbor address of the neighbor routing interface IP address. Next, configure the normal BGP peer configuration options such as the timers and address families that are needed. In address-family IPv4-unicast `next-hop-self` must be set to `true`.

If BGP over SVR is with eBGP, set the `multihop ttl` to at least 2.

Configuration Template:

```
config
    authority
      router <router name>
        routing default-instance
          type default-instance
          interface rtg-int
            name rtg-int
            ip-address <self rtg-int ip>
          exit
       routing-protocol bgp
          type bgp
          local-as <local as>
          router-id <self rtg-int ip>
          neighbor <neighbor rtg-int ip>
            neighbor-address <neighbor rtg-int ip>
            neighbor-as <neighbor as>
            timers
              hold-time 9
              keepalive-interval 3
            exit
            address-family ipv4-unicast
              afi-safi ipv4-unicast
              next-hop-self true
            exit
            transport
               local-address
                 routing-interface rtg-int
               exit
             exit
             multihop
               ttl 255
```

```
            exit
          exit
        exit
      exit
    exit
  exit
exit
```

Sample Configuration:

```
admin@branchoffice1.seattlesite1# config auth
admin@branchoffice1.seattlesite1 (authority)# router seattlesite1
admin@branchoffice1.seattlesite1 (router[name=seattlesite1])# routing
default-instance
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# interface
bgp-int-seattle
admin@branchoffice1.seattlesite1 (interface[name=bgp-int-seattle])# ip-
address 10.128.128.2
admin@branchoffice1.seattlesite1 (interface[name=bgp-int-seattle])# exit
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# routing-
protocol bgp
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# local-as 100
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# router-id
10.128.128.2
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# address-
family ipv4-unicast
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#
exit
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# neighbor
10.128.128.1
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=10.128.128.1])#
neighbor-as 300
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=10.128.128.1])#
address-family ipv4-unicast
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#
next-hop-self true
admin@branchoffice1.seattlesite1 (address-family[afi-safi=ipv4-unicast])#
exit
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=10.128.128.1])#
transport
admin@branchoffice1.seattlesite1 (transport)# local-address
admin@branchoffice1.seattlesite1 (local-address)# routing-interface bgp-int-
seattle
admin@branchoffice1.seattlesite1 (local-address)# exit
admin@branchoffice1.seattlesite1 (transport)# exit
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=10.128.128.1])#
multihop
admin@branchoffice1.seattlesite1 (multihop)# ttl 5
admin@branchoffice1.seattlesite1 (multihop)# exit
admin@branchoffice1.seattlesite1 (neighbor[neighbor-address=10.128.128.1])#
exit
admin@branchoffice1.seattlesite1 (routing-protocol[type=bgp])# exit
admin@branchoffice1.seattlesite1 (routing[type=default-instance])# exit
admin@branchoffice1.seattlesite1 (router[name=seattlesite1])# exit
admin@branchoffice1.seattlesite1 (authority)# exit
admin@branchoffice1.seattlesite1#
```

BGP Verification:

Issue the command `show bgp` on your router:

```
admin@branchoffice1.seattlesite1# show bgp
Wed 2019-02-20 23:14:58 UTC
BGP table version is 12, local router ID is 10.128.128.2, vrf id 0
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
   10.128.128.2/32  10.128.128.3                         0 200 300 ?
   10.128.128.3/32  10.128.128.3                         0 200 300 ?
   128.128.128.1/32 10.128.128.3                         0 200 300 ?
   128.128.128.128/32
                    10.128.128.3                         0 200 300 ?
   172.16.128.2/32  10.128.128.3                         0 200 300 ?
   172.16.255.0/30  0.0.0.0                  0         32768 i
   172.26.128.0/30  10.128.128.3                         0 200 300 i

Displayed  7 routes and 7 total paths
```

In addition to the `show bgp` branch of output, you will now see contributions to the RIB from BGP in the output of `show rib`

```
admin@branchoffice1.seattlesite1# show rib
Wed 2018-08-22 16:50:00 UTC
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, T - Table, v - VNC,
       V - VNC-Direct,
       > - selected route, * - FIB route

B   1.1.1.0/24 [20/0] via 1.1.1.1 inactive, 22:28:18
C>* 1.1.1.0/24 is directly connected, g4
B>* 2.2.2.0/24 [20/0] via 1.1.1.1, g4, 22:28:18
B>* 3.3.3.0/24 [20/0] via 1.1.1.1, g4, 22:28:18
C>* 10.0.128.0/31 is directly connected, g1
K>* 128.128.128.1/32 is directly connected, Null0, bh
K>* 128.128.128.128/32 is directly connected, Null0, bh
C>* 169.254.127.126/31 is directly connected, g4294967294
S>* 172.16.128.2/32 [1/0] via 10.0.128.1, g1
K   172.16.128.2/32 is directly connected, Null0, bh
K>* 172.26.128.2/32 is directly connected, Null0, bh
C>* 192.168.64.0/24 is directly connected, g2


Completed in 0.09 seconds
admin@branchoffice1.seattlesite1#
```

BGP Troubleshooting:
Issue the following commands to verify BGP related information:

- Verify BGP router information (`show bgp summary`)
    - Check the AS
    - Check for router ID and peers (up/down and state column)

```
admin@branchoffice1.seattlesite1# show bgp summary
Wed 2019-02-20 23:37:17 UTC

IPv4 Unicast Summary:
BGP router identifier 10.128.128.2, local AS number 100 vrf-id 0
BGP table version 12
RIB entries 13, using 1976 bytes of memory
Peers 2, using 41 KiB of memory

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
10.128.128.1    4        300       0       0        0    0    0    never       Active
10.128.128.3    4        200    8972    8969        0    0    0 3d02h43m            6

Total number of neighbors 2


Completed in 0.29 seconds
```

- Verify OSPF neighbors (`show bgp neighbors`)
    - Check BGP state for each neighbor = established, up for <time>
- Verify RIB (`show rib`)
    - Routes beginning with B are BGP routes
- Verify FIB (`show fib)`
    - FIB entry has the appropriate next hop