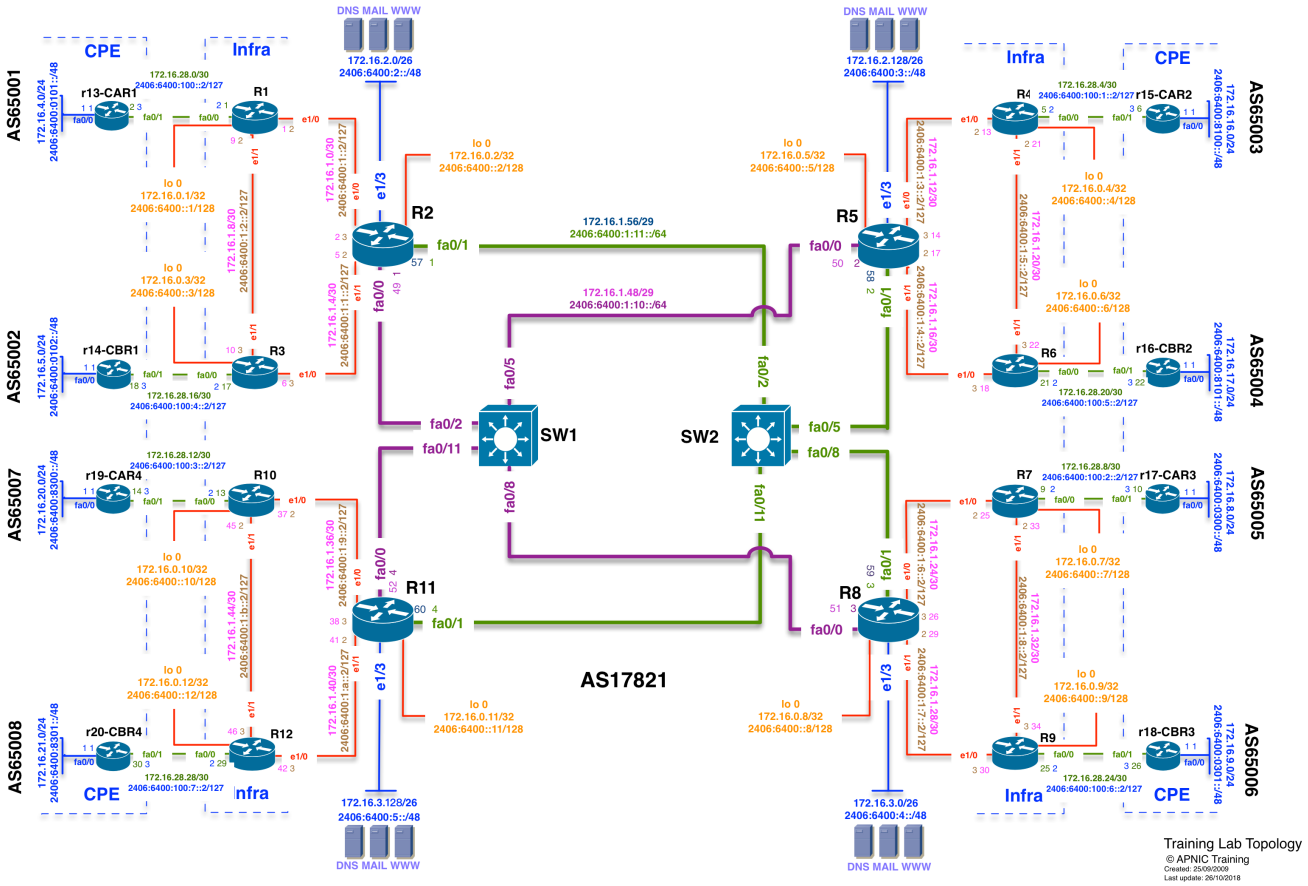
Module 5 - eBGP Configuration (Customer router)

Topology introduction:

- The topology below shows 4 regional networks comprised of a core POP and 2 aggregation POPs (edge routers).
- Edge routers aggregate downstream customers.
- The regional networks are interconnected with redundant transport links.



Lab Tasks

After completing Module-4 (iBGP Lab), your network is setup to propagate externally learned routes to all internal routers, as well as internal routes to external networks (different ASNs).

In this lab, we will configure eBGP with customer routers (R13-R20).

Take note of the following for eBGP:

1. eBGP neighbour relationship is normally established over directly connected interface addresses. Remember, BGP relies on TCP, and that eBGP has a TTL of 1.

2. If you want to use some other interface addresses (loopbacks) for eBGP neighbour session, then you will need to have a static route pointing to each other's loopbacks, as well as modify the eBGP TTL to a value greater than 1 using `e-bgp-multihop <TTL>` command

3. When an eBGP router receives external routes/prefixes from its eBGP peers, the next-hop address is retained when it announces those routes to its iBGP peers. Since the iBGP routers would not have a route to the external next-hop, they will not install those routes in their routing table (next-hop reachability is a necessity). Hence, we need to change this default behaviour using the `next-hop-self` command on the eBGP routers, so that next-hop for those routes points to itself when announcing to iBGP peers.

4. It is not necessary to send the full-internet route to customers (if they have inexpensive routers with limited memory and CPU). Hence, in this lab, each upstream router will originate a default route towards their downstream customers with `default-originate` command.

5. Each team will need to configure both side – upstream and downstream routers, as follows:

    - a. Steps involved in POP side (edge POP routers) are:

        - i. Customer side interface configuration
        - ii. Connectivity testing
        - iii. eBGP configuration
        - iv. Change the eBGP next hop behaviour using next-hop-self
        - v. Inject a default route towards customer

    - b. Steps involved in CPE side (customer routers) are:

        - i. Basic interface configuration (to the upstream as well as best practices)
        - ii. Connectivity testing
        - iii.eBGP peering with upstream routers
        - iv. Prefix advertisement

6. After finishing eBGP configuration, we should see the following 8 new prefixes on our infrastructure routers:

Customer	AS Number	Customer block
Router13	65001	2406:6400:0101::/48
Router14	65002	2406:6400:0102::/48
Router15	65003	2406:6400:8100::/48
Router16	65004	2406:6400:8101::/48
Router17	65005	2406:6400:0300::/48
Router18	65006	2406:6400:0301::/48
Router19	65007	2406:6400:8300::/48
Router20	65008	2406:6400:8301::/48

Lab Exercise

There are configurations on edge POP routers and customer routers in this module.

1 - POP Router Configuration:

On each of the edge POP routers (that aggregate downstream customers), we will configure eBGP sessions with downstream customers.

Example configuration on Router1:

Step-1 interface configuration:

```
config t
interface fa0/0
description Link to Customer-R13
ipv6 address 2406:6400:100::2/127
no shutdown
```

Step-2 eBGP peering with downstream router:

```
config t
router bgp 17821
address-family ipv6 unicast
neighbor 2406:6400:100::3 remote-as 65001
    !Originate default [::/0] to customer router
neighbor 2406:6400:100::3 description eBGP with R13
neighbor 2406:6400:100::3 default-originate
neighbor 2406:6400:100::3 activate
```

Step-3 configure `next-hop-self` to iBGP neighbor:

In a POP router, next hop of the customer prefix will be point-to-point interface address of the customer router. Since customer prefix is collected by eBGP when POP router forward it to its iBGP peer it will not change the next hop by default unless the above command is executed. After configure the `next-hop-self`, POP router will change the next-hop of customer routes to be its loopback address when the routes are advertised to iBGP peers.

```
config t
router bgp 17821
address-family ipv6 unicast
        !Point next-hop to itself
neighbor RRv6 next-hop-self
```

2 - Downstream Customer Router Configuration:

We will establish eBGP relationship with upstream routers.

Step-1 basic configuration:

Make sure you configure all the best practices you configured earlier. Refer to the lab guide Module-1.

Step-2 interface configuration:

Configure the interface addresses, along with interface best practices as you did earlier for Module-1.

Step-3 eBGP peering with upstream:

Example configuration on Router14:

```
config t
router bgp 65002
no bgp default ipv4-unicast
address-family ipv6 unicast
neighbor 2406:6400:100:4::2 remote-as 17821
neighbor 2406:6400:100:4::2 description eBGP with R3
neighbor 2406:6400:100:4::2 activate
```

Step-4 (announce prefix):

On each customer router, the customer blocks will be announced in BGP.

Example configuration on R16:

```
config t
router bgp 65004
address-family ipv6 unicast
network 2406:6400:8101::/48
exit
exit
ipv6 route 2406:6400:8101::/48 null 0
```

Please remember to save the configuration.

3 - Verify BGP Configuration:

Verify IPv4 BGP configuration:

`show bgp ipv4 unicast summary` ! List IPv4 BGP peers

`show bgp ipv4 unicast` ! List IPv4 routes in BGP Table

`show bgp ipv4 unicast <prefix/length>` ! List IPv4 specific routes

`show ip route bgp` ! Check IPv4 routes learned via BGP

`show ip route` ! Check your IPv4 routing table (best paths)

Verify IPv6 BGP configuration:

`show bgp ipv6 unicast summary` ! List IPv6 BGP peers

`show bgp ipv6 unicast` ! List IPv6 routes in BGP Table

`show bgp ipv6 unicast <prefix/length>` ! List IPv6 specific routes

`show ipv6 route bgp` ! Check IPv6 routes learned via BGP

`show ipv6 route` ! Check your IPv6 routing table (best paths)

`show bgp ipv6 unicast neighbors <neighbour-address> advertised-routes` ! Check routes advertised to your neighbour

`show bgp ipv6 unicast neighbors <neighbour-address> routes` ! Check routes learned from your neighbous