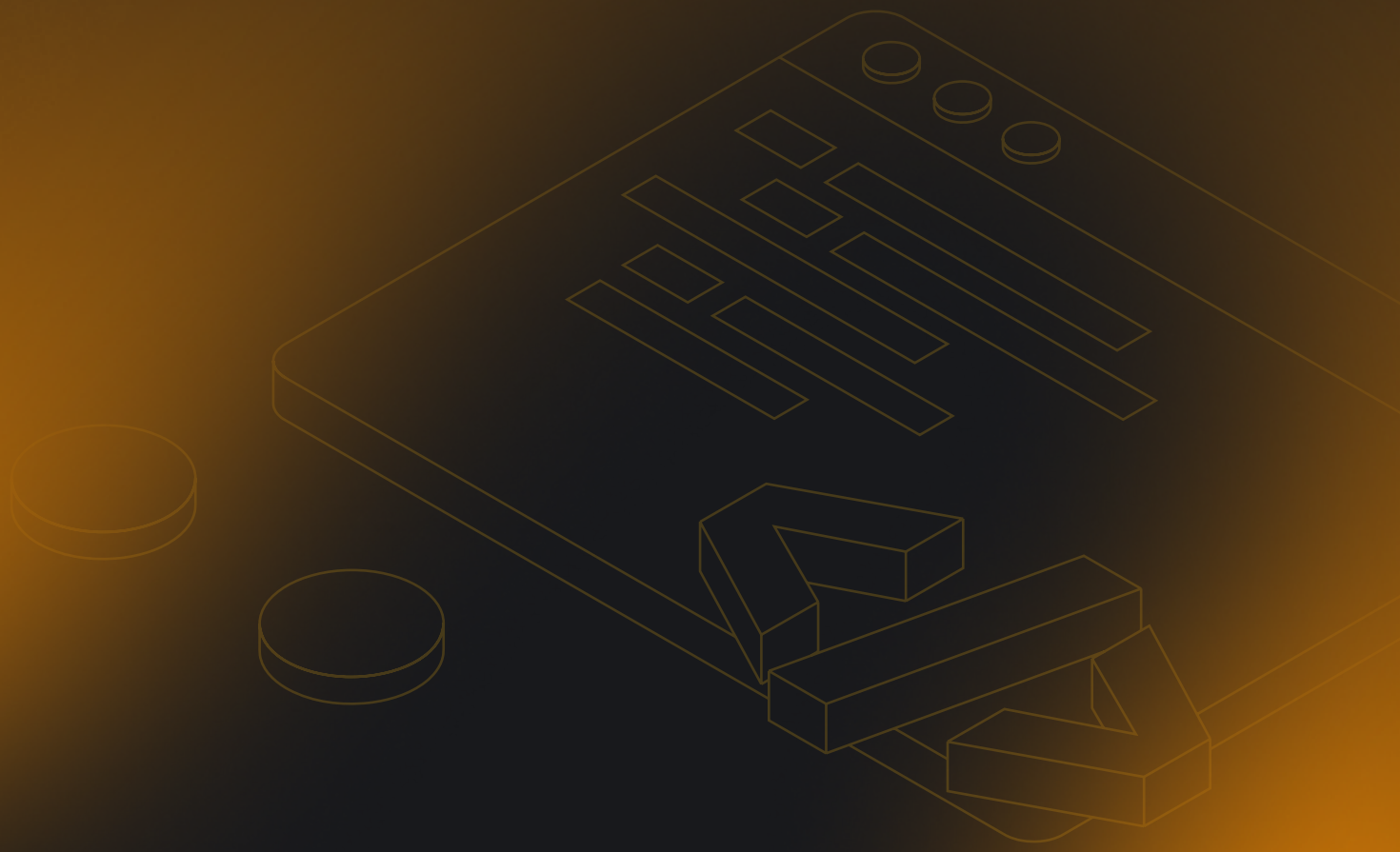
VyOS Networks



Deployment Guide | Technical Doc

L3VPN OVER SEGMENT ROUTING MPLS (SR-MPLS)

2026

Index:

Introduction **3**

Topology **4**

Configuration **5**

- Configuring IGP and enabling Segment Routing 5
- Configuring iBGP for L3VPN control-plane 8
- Configuring L3VPN VRFs on PE nodes 9
- Configuring CE nodes 9

Verification **11**

Packet capture **15**

Introduction

Segment Routing (SR) is a forwarding paradigm closely related to source routing. When a packet enters the network, the ingress router attaches an ordered list of **Segment Identifiers (SIDs)** that represent the sequence of actions or waypoints the packet should traverse.

As the packet moves through the network, each router examines the active SID, performs the corresponding forwarding action, exposing the next SID in the stack. This SID list fully determines the path the packet follows end-to-end.

SR integrates seamlessly with an existing MPLS data plane. In MPLS environments, segments are encoded as MPLS labels and pushed at the ingress. Interior Gateway Protocols (IGPs) such as IS-IS or OSPF advertise and distribute these labels across the network. Segment Routing as defined in **RFC 8667** specifies SR operation over the MPLS data plane, simplifying the control plane while preserving the benefits of MPLS forwarding.

Layer 3 VPN (L3VPN) services over SR-MPLS combine the proven VPN separation of MPLS with the flexibility and efficiency of segment routing. BGP typically handles the VPN control plane, distributing per-VRF routing information across the provider network. Each customer's traffic is logically isolated through Virtual Routing and Forwarding (VRF) instances, while SR-MPLS provides a simplified and deterministic transport layer by steering traffic along explicit segment lists without requiring LDP signaling.

A typical SR-MPLS L3VPN environment consists of:

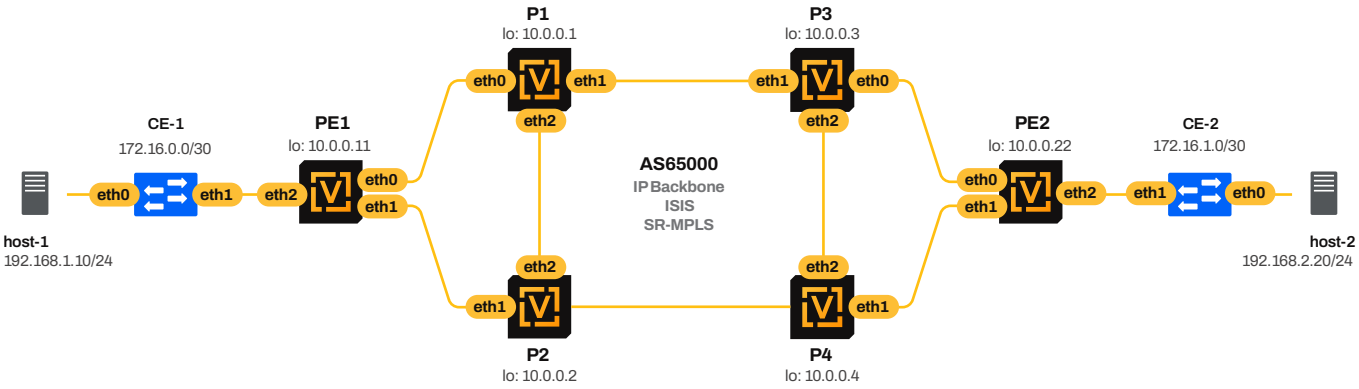
- **Provider Core Routers (P)** running SR-MPLS and segment allocation.
- **Provider Edge Routers (PE)** hosting per-customer VRFs and exchanging VPNv4/VPNv6 routes over BGP.
- **Route Reflectors (RR)** to distribute VPN routes efficiently.
- **Customer Edge Routers (CE)** peering with PEs over standard IP interfaces.

Some tips for a simple SR-MPLS deployment:

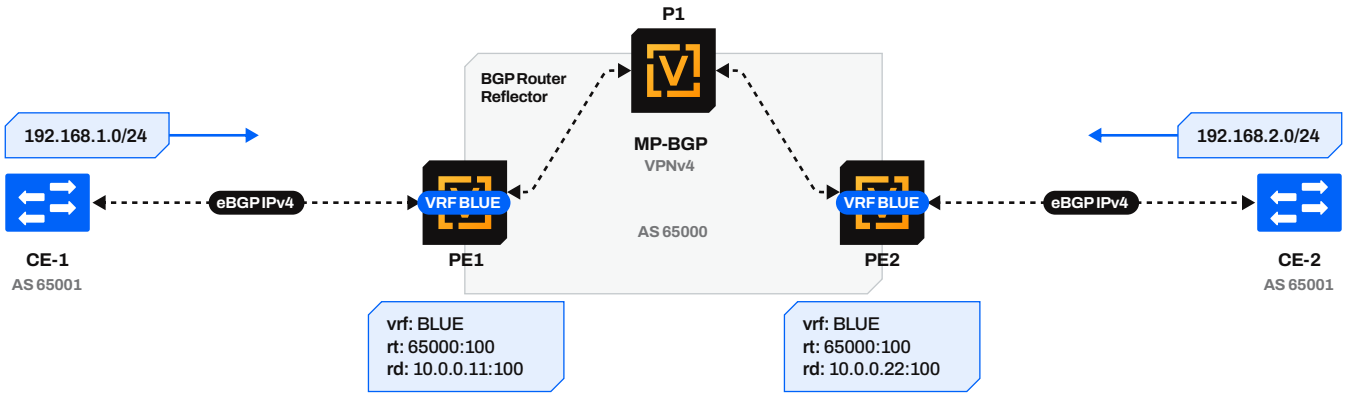
- **Plan the IGP underlay carefully:** Use OSPF or IS-IS with segment routing extensions enabled. Ensure the IGP domain is stable and converges quickly before deploying SR.
- **Define Segment IDs (SIDs) consistently:** Allocate Node-SIDs and Adj-SIDs with a clear numbering plan to avoid overlaps and ease troubleshooting.
- **Deploy a Route Reflector strategy:** Use iBGP with VPNv4/VPNv6 or EVPN for service distribution.
- **Use SRGB planning:** Align the Segment Routing Global Block (SRGB) across routers to simplify operations and reduce label translation issues.
- **Service integration:** Combine SR-MPLS with L3VPN, EVPN, to optimize service delivery and support multiple customer requirements.

Topology

Physical topology



Logical topology



Configuration

Configuring IGP and enabling Segment Routing

Configure the IP/MPLS backbone by deploying an Interior Gateway Protocol (IGP) (ISIS in our example) and enabling Segment Routing for SID distribution. This provides the foundational connectivity across Provider (P), Provider Edge (PE), and Route Reflector (RR) nodes.

Configure the **Segment Routing Global Block (SRGB)**, the MPLS label range reserved for storing Prefix-SIDs in the MPLS forwarding table (FIB).

```
set protocols isis segment-routing global-block high-label-value '1099'
```

```
set protocols isis segment-routing global-block low-label-value '1000'
```

A segment ID that represents the IP address prefix calculated by an IGP in the service provider core network. Prefix SIDs are globally unique.

```
set protocols isis segment-routing prefix <address> index value <0-65535>
```

We have the option to configure the behavior of the penultimate hop. The following flags allow fine-grained control of how labels are handled:

- **No-php-flag:** Disables Penultimate Hop Popping, allowing the SR node to request that its upstream neighbor does not pop the label.
- **explicit-null:** Instructs the SR node to request that the neighbor forward the packet with the Explicit-Null label instead of popping it.
- **n-flag-clear:** Explicitly clears the default Node flag for Prefix-SIDs associated with loopback addresses. This option is required when configuring Anycast-SIDs.

```
set protocols isis segment-routing prefix <address> index <no-php-flag |
explicit-null| n-flag-clear>
```

PE1:

```
# interfaces

set interfaces ethernet eth0 address '10.11.1.11/24'
set interfaces ethernet eth0 description 'to_P1'
set interfaces ethernet eth1 address '10.11.2.11/24'
set interfaces ethernet eth1 description 'to_P2'
set interfaces loopback lo address '10.0.0.11/32'

# protocols ISIS+segment-routing

set protocols isis interface eth0
set protocols isis interface eth1
set protocols isis interface lo
set protocols isis net '49.0001.1000.0000.0011.00'
```

```
set protocols isis segment-routing global-block high-label-value '1099'
set protocols isis segment-routing global-block low-label-value '1000'
set protocols isis segment-routing prefix 10.0.0.11/32 index value '11'
set protocols mpls interface 'eth0'
set protocols mpls interface 'eth1'
```

P1:

```
# interfaces

set interfaces ethernet eth0 address '10.11.1.1/24'
set interfaces ethernet eth0 description 'to_P3'
set interfaces ethernet eth1 address '10.1.3.1/24'
set interfaces ethernet eth1 description 'to_P2'
set interfaces ethernet eth2 address '10.1.2.1/24'
set interfaces ethernet eth2 description 'to_PE1'
set interfaces loopback lo address '10.0.0.1/32'


# protocols ISIS+segment-routing

set protocols isis interface eth0
set protocols isis interface eth1
set protocols isis interface eth2
set protocols isis interface lo
set protocols isis net '49.0001.1000.0000.0001.00'
set protocols isis segment-routing global-block high-label-value '1099'
set protocols isis segment-routing global-block low-label-value '1000'
set protocols isis segment-routing prefix 10.0.0.1/32 index value '1'
set protocols mpls interface 'eth0'
set protocols mpls interface 'eth1'
set protocols mpls interface 'eth2'
```

P2:

```
# interfaces

set interfaces ethernet eth0 address '10.11.2.2/24'
set interfaces ethernet eth0 description 'to_P1'
set interfaces ethernet eth1 address '10.2.4.2/24'
set interfaces ethernet eth1 description 'to_P4'
set interfaces ethernet eth2 address '10.1.2.2/24'
set interfaces ethernet eth2 description 'to_PE1'

# protocols ISIS+segment-routing

set protocols isis interface eth0
set protocols isis interface eth1
set protocols isis interface eth2
set protocols isis interface lo
set protocols isis net '49.0001.1000.0000.0002.00'
set protocols isis segment-routing global-block high-label-value '1099'
set protocols isis segment-routing global-block low-label-value '1000'
set protocols isis segment-routing prefix 10.0.0.2/32 index value '2'
set protocols mpls interface 'eth0'
set protocols mpls interface 'eth1'
set protocols mpls interface 'eth2'
```

P3:

```
# interfaces

set interfaces ethernet eth0 address '10.1.3.3/24'
set interfaces ethernet eth0 description 'to_P1'
set interfaces ethernet eth1 address '10.22.3.3/24'
set interfaces ethernet eth1 description 'to_P4'
set interfaces ethernet eth1 hw-id '0c:80:01:81:00:01'
set interfaces ethernet eth2 address '10.3.4.3/24'
set interfaces ethernet eth2 description 'to_PE2'

# protocols ISIS+segment-routing

set protocols isis interface eth0
set protocols isis interface eth1
set protocols isis interface eth2
set protocols isis interface lo
set protocols isis net '49.0001.1000.0000.0003.00'
set protocols isis segment-routing global-block high-label-value '1099'
set protocols isis segment-routing global-block low-label-value '1000'
set protocols isis segment-routing prefix 10.0.0.3/32 index value '3'
set protocols mpls interface 'eth0'
set protocols mpls interface 'eth1'
set protocols mpls interface 'eth2'
```

P4:

```
# interfaces

set interfaces ethernet eth0 address '10.22.4.4/24'
set interfaces ethernet eth0 description 'to_P3'
set interfaces ethernet eth1 address '10.2.4.4/24'
set interfaces ethernet eth1 description 'to_P2'
set interfaces ethernet eth2 address '10.3.4.4/24'
set interfaces ethernet eth2 description 'to_PE2'

# protocols ISIS+segment-routing

set protocols isis interface eth0
set protocols isis interface eth1
set protocols isis interface eth2
set protocols isis interface lo
set protocols isis net '49.0001.1000.0000.0004.00'
set protocols isis segment-routing global-block high-label-value '1099'
set protocols isis segment-routing global-block low-label-value '1000'
set protocols isis segment-routing prefix 10.0.0.4/32 index value '4'
set protocols mpls interface 'eth0'
set protocols mpls interface 'eth1'
set protocols mpls interface 'eth2'
```

PE2:

```
# interfaces

set interfaces ethernet eth0 address '10.22.3.22/24'
set interfaces ethernet eth0 description 'to_P3'
set interfaces ethernet eth1 address '10.22.4.22/24'
```

```
set interfaces ethernet eth1 description 'to_P4'

# protocols ISIS+segment-routing

set protocols isis interface eth0
set protocols isis interface eth1
set protocols isis interface lo
set protocols isis net '49.0001.1000.0000.0022.00'
set protocols isis segment-routing global-block high-label-value '1099'
set protocols isis segment-routing global-block low-label-value '1000'
set protocols isis segment-routing prefix 10.0.0.22/32 index value '22'
set protocols mpls interface 'eth0'
set protocols mpls interface 'eth1'
```

Configuring iBGP for L3VPN control-plane

Next, enable **iBGP** on the Provider Edge (PE) routers and the Route Reflector (P1 in this lab; production networks typically deploy two for redundancy) to distribute **IPv4 VPN (L3VPN)** routes across the network.

P1/RR:

```
set protocols bgp neighbor 10.0.0.11 description 'PE1'
set protocols bgp neighbor 10.0.0.11 peer-group 'RR_VPNv4'
set protocols bgp neighbor 10.0.0.22 description 'PE2'
set protocols bgp neighbor 10.0.0.22 peer-group 'RR_VPNv4'
set protocols bgp parameters cluster-id '10.0.0.1'
set protocols bgp parameters log-neighbor-changes
set protocols bgp parameters router-id '10.0.0.1'
set protocols bgp peer-group RR_VPNv4 address-family ipv4-vpn route-reflector-client
set protocols bgp peer-group RR_VPNv4 remote-as '65000'
set protocols bgp peer-group RR_VPNv4 update-source 'lo'
set protocols bgp system-as '65000'
```

PE1:

```
set protocols bgp neighbor 10.0.0.1 description 'RR-P1'
set protocols bgp neighbor 10.0.0.1 peer-group 'RR_VPNv4'
set protocols bgp parameters log-neighbor-changes
set protocols bgp parameters router-id '10.0.0.11'
set protocols bgp peer-group RR_VPNv4 address-family ipv4-vpn nexthop-self
set protocols bgp peer-group RR_VPNv4 remote-as '65000'
set protocols bgp peer-group RR_VPNv4 update-source 'lo'
set protocols bgp system-as '65000'
```

PE2:

```
set protocols bgp neighbor 10.0.0.1 description 'RR-P1'
set protocols bgp neighbor 10.0.0.1 peer-group 'RR_VPNv4'
set protocols bgp parameters log-neighbor-changes
set protocols bgp parameters router-id '10.0.0.22'
set protocols bgp peer-group RR_VPNv4 address-family ipv4-vpn nexthop-self
set protocols bgp peer-group RR_VPNv4 remote-as '65000'
set protocols bgp peer-group RR_VPNv4 update-source 'lo'
set protocols bgp system-as '65000'
```

Configuring L3VPN VRFs on PE nodes

This section provides configuration steps for setting up VRFs on our PE nodes including CE facing interfaces, rd and route-target.

PE1:

```
# VRF settings

set vrf name BLUE protocols bgp address-family ipv4-unicast export vpn
set vrf name BLUE protocols bgp address-family ipv4-unicast import vpn
set vrf name BLUE protocols bgp address-family ipv4-unicast label vpn export 'auto'
set vrf name BLUE protocols bgp address-family ipv4-unicast network 192.168.2.0/24
set vrf name BLUE protocols bgp address-family ipv4-unicast rd vpn export '10.0.0.11:100'
set vrf name BLUE protocols bgp address-family ipv4-unicast redistribute connected
set vrf name BLUE protocols bgp address-family ipv4-unicast route-target vpn both '65000:100'
set vrf name BLUE protocols bgp parameters router-id '10.0.0.11'
set vrf name BLUE protocols bgp system-as '65000'
set vrf name BLUE table '100'
set vrf name BLUE protocols bgp neighbor 172.16.0.2 address-family ipv4-unicast as-override
set vrf name BLUE protocols bgp neighbor 172.16.0.2 remote-as '65001'

# interfaces

set interfaces ethernet eth2 address '172.16.0.1/30'
set interfaces ethernet eth2 vrf 'BLUE'
```

PE2:

```
# VRF settings

set vrf name BLUE protocols bgp address-family ipv4-unicast export vpn
set vrf name BLUE protocols bgp address-family ipv4-unicast import vpn
set vrf name BLUE protocols bgp address-family ipv4-unicast label vpn export 'auto'
set vrf name BLUE protocols bgp address-family ipv4-unicast network 192.168.1.0/24
set vrf name BLUE protocols bgp address-family ipv4-unicast rd vpn export '10.0.0.22:100'
set vrf name BLUE protocols bgp address-family ipv4-unicast redistribute connected
set vrf name BLUE protocols bgp address-family ipv4-unicast route-target vpn both '65000:100'
set vrf name BLUE protocols bgp parameters router-id '10.0.0.22'
set vrf name BLUE protocols bgp system-as '65000'
set vrf name BLUE table '100'
set vrf name BLUE protocols bgp neighbor 172.16.1.2 address-family ipv4-unicast as-override
set vrf name BLUE protocols bgp neighbor 172.16.1.2 remote-as '65001'

# interfaces

set interfaces ethernet eth2 address '172.16.1.1/30'
set interfaces ethernet eth2 vrf 'BLUE'
```

Configuring CE nodes

Dynamic routing is used between the CE and PE nodes, with eBGP sessions established to exchange routes. The PE routers then import these routes into the L3VPN and advertise them to the Route Reflector (RR). From there, the routes are distributed to the remote PEs, and vice versa, according to the previously defined L3VPN configuration.

CE1:

```
# interfaces

set interfaces ethernet eth0 address '172.16.0.2/30'
set interfaces ethernet eth1 address '192.168.1.254/24'

# BGP for peering with PE

set protocols bgp address-family ipv4-unicast network 192.168.1.0/24
set protocols bgp neighbor 172.16.0.1 address-family ipv4-unicast
set protocols bgp neighbor 172.16.0.1 remote-as '65000'
set protocols bgp neighbor 172.16.0.1 update-source 'eth0'
set protocols bgp parameters log-neighbor-changes
set protocols bgp parameters router-id '192.168.1.1'
set protocols bgp system-as '65001'
```

CE2:

```
# interfaces

set interfaces ethernet eth0 address '172.16.1.2/30'
set interfaces ethernet eth1 address '192.168.2.254/24'

# BGP for peering with PE

set protocols bgp address-family ipv4-unicast network 192.168.2.0/24
set protocols bgp neighbor 172.16.1.1 address-family ipv4-unicast
set protocols bgp neighbor 172.16.1.1 remote-as '65000'
set protocols bgp neighbor 172.16.1.1 update-source 'eth0'
set protocols bgp parameters log-neighbor-changes
set protocols bgp parameters router-id '192.168.2.2'
set protocols bgp system-as '65001'
```

Verification

This section describes verification commands for ISIS/SR/MPLS/BGP protocols and L3VPN related routes as well as diagnosis and reachability checks between remote hosts.

- `show ip isis neighbor` for checking isis relationship

```
vyos@PE1# run show isis neighbor

Area VyOS:
 System Id           Interface   L  State        Holdtime SNPA
 P1                  eth0        1  Up           28       0ca8.ab2d.0000
 P1                  eth0        2  Up           30       0ca8.ab2d.0000
 P2                  eth1        1  Up           30       0cdc.42b4.0000
 P2                  eth1        2  Up           30       0cdc.42b4.0000
```

- `show isis segment-routing node` for checking detailed information about all learned Segment Routing nodes.

```
vyos@PE1# run show isis segment-routing node

Area VyOS:

 IS-IS L1 SR-Nodes:

 System ID       SRGB         SRLB           Algorithm  MSD
 ------------------------------------------------------------
 1000.0000.0001  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0002  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0003  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0004  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0011  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0022  1000 - 1099  15000 - 15999  SPF        0

 IS-IS L2 SR-Nodes:

 System ID       SRGB         SRLB           Algorithm  MSD
 ------------------------------------------------------------
 1000.0000.0001  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0002  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0003  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0004  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0011  1000 - 1099  15000 - 15999  SPF        0
 1000.0000.0022  1000 - 1099  15000 - 15999  SPF        0
```

- `show mpls table` This gives us MPLS segment routing labels for far end loopbacks:

```
vyos@PE1# run show mpls table
 Inbound Label  Type        Nexthop              Outbound Label
 -----------------------------------------------------------------
 80             BGP         BLUE                 -
 1001           SR (IS-IS)  10.11.1.1            implicit-null
 1002           SR (IS-IS)  10.11.1.1            1002
 1003           SR (IS-IS)  10.11.1.1            1003
 1004           SR (IS-IS)  10.11.1.1            1004
 1022           SR (IS-IS)  10.11.1.1            1022
```

```
 15002          SR (IS-IS)  10.11.1.1             implicit-null
 15003          SR (IS-IS)  fe80::ea8:abff:fe2d:0  implicit-null
 15006          SR (IS-IS)  10.11.1.1             implicit-null
 15007          SR (IS-IS)  fe80::ea8:abff:fe2d:0  implicit-null

[edit]

vyos@PE1#
```

- `show ip route isis` Here is the routing tables showing the MPLS segment routing label operations:

```
vyos@PE1# run show ip route isis
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

I>* 10.0.0.1/32 [115/20] via 10.11.1.1, eth0, label implicit-null, weight 1, 00:00:01
I>* 10.0.0.2/32 [115/30] via 10.11.1.1, eth0, label 1002, weight 1, 00:00:01
I>* 10.0.0.3/32 [115/30] via 10.11.1.1, eth0, label 1003, weight 1, 00:00:01
I>* 10.0.0.4/32 [115/40] via 10.11.1.1, eth0, label 1004, weight 1, 00:00:01
I>* 10.0.0.22/32 [115/40] via 10.11.1.1, eth0, label 1022, weight 1, 00:00:01
                          via 10.11.2.2, eth1 inactive, label 1022, weight 1, 00:00:01
I>* 10.1.2.0/24 [115/20] via 10.11.1.1, eth0, weight 1, 00:00:01
                         via 10.11.2.2, eth1 inactive, weight 1, 00:00:01
I>* 10.1.3.0/24 [115/20] via 10.11.1.1, eth0, weight 1, 00:00:01
I>* 10.2.4.0/24 [115/30] via 10.11.1.1, eth0, weight 1, 00:00:01
I>* 10.3.4.0/24 [115/30] via 10.11.1.1, eth0, weight 1, 00:00:01
                         via 10.11.2.2, eth1 inactive, weight 1, 00:00:01
I   10.11.1.0/24 [115/20] via 10.11.1.1, eth0 inactive, weight 1, 00:00:01
I>* 10.11.2.0/24 [115/30] via 10.11.1.1, eth0, weight 1, 01:14:42
I>* 10.22.3.0/24 [115/30] via 10.11.1.1, eth0, weight 1, 00:00:01
I>* 10.22.4.0/24 [115/40] via 10.11.1.1, eth0, weight 1, 00:00:01

[edit]

vyos@PE1#
```

> ⚠ implicit-null: This tells the router not to push a label when forwarding traffic, but instead to **perform Penultimate Hop Popping (PHP)**, removing the MPLS label before sending the packet to the egress router. Advertised by the **egress LSR** for routes it originates or terminates.

- `show ip route forward` shows the routes **installed in the forwarding plane** (kernel FIB) and used for packet forwarding.

```
vyos@PE1# run show ip route forward
default nhid 26 via 192.168.122.1 dev eth4 proto static metric 20
10.0.0.1 nhid 2266 via 10.11.1.1 dev eth0 proto isis metric 20
10.0.0.2 nhid 1997  encap mpls  1002 via 10.11.1.1 dev eth0 proto isis metric 20
10.0.0.3 nhid 53  encap mpls  1003 via 10.11.1.1 dev eth0 proto isis metric 20
10.0.0.4 nhid 1998  encap mpls  1004 via 10.11.1.1 dev eth0 proto isis metric 20
10.0.0.22 nhid 55  encap mpls  1022 via 10.11.1.1 dev eth0 proto isis metric 20
```

```
10.1.2.0/24 nhid 58 via 10.11.1.1 dev eth0 proto isis metric 20
10.1.3.0/24 nhid 59 via 10.11.1.1 dev eth0 proto isis metric 20
10.2.4.0/24 nhid 59 via 10.11.1.1 dev eth0 proto isis metric 20
10.3.4.0/24 nhid 58 via 10.11.1.1 dev eth0 proto isis metric 20
10.11.1.0/24 dev eth0 proto kernel scope link src 10.11.1.11
10.11.2.0/24 dev eth1 proto kernel scope link src 10.11.2.11 dead linkdown
10.11.2.0/24 nhid 59 via 10.11.1.1 dev eth0 proto isis metric 20
10.22.3.0/24 nhid 59 via 10.11.1.1 dev eth0 proto isis metric 20
10.22.4.0/24 nhid 59 via 10.11.1.1 dev eth0 proto isis metric 20
192.168.122.0/24 dev eth4 proto kernel scope link src 192.168.122.167
```

In this output, we can see that PE2 (10.0.0.22) is reachable through interface **eth0**, using MPLS encapsulation with label **1022.**

Now we're checking iBGP status and routes from PE nodes to RR:

- `show bgp ipv4 vpn summary` for checking BGP VPNv4 neighbors:

```
vyos@PE1# run show bgp ipv4 vpn summary
BGP router identifier 10.0.0.11, local AS number 65000 VRF default vrf-id 0
BGP table version 0
RIB entries 3, using 384 bytes of memory
Peers 1, using 24 KiB of memory
Peer groups 1, using 64 bytes of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
10.0.0.1        4      65000       192       190       31    0    0 01:31:27            2        2 RR-P1

Total number of neighbors 1
```

- `show bgp ipv4 vpn` for checking all VPNv4 prefixes information:

```
vyos@PE1# run show bgp ipv4 vpn
BGP table version is 31, local router ID is 10.0.0.11, vrf id 0
Default local pref 100, local AS 65000
Status codes:  s suppressed, d damped, h history, u unsorted, * 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
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 10.0.0.11:100
 *>  172.16.0.0/30    0.0.0.0@8<              0         32768 ?
    UN=0.0.0.0 EC{65000:100} label=80 type=bgp, subtype=5
 *>  192.168.1.0/24   172.16.0.2@8<           0             0 65001 i
    UN=172.16.0.2 EC{65000:100} label=80 type=bgp, subtype=5
Route Distinguisher: 10.0.0.22:100
 *>i 172.16.1.0/30    10.0.0.22               0    100      0 ?
    UN=10.0.0.22 EC{65000:100} label=80 type=bgp, subtype=0
 *>i 192.168.2.0/24   10.0.0.22               0    100      0 65001 i
    UN=10.0.0.22 EC{65000:100} label=80 type=bgp, subtype=0

Displayed 4 routes and 4 total paths
```

The output indicates that the **Host-2** network (192.168.2.0/24) is known via MP-BGP as a VPNv4 prefix, with **PE2** (10.0.0.22) as the next-hop router and MPLS **label 80** assigned for service encapsulation.

- `show bgp ipv4 vpn x.x.x.x/x` for checking best path selected for specific VPNv4 destination

```
vyos@PE1# run show bgp ipv4 vpn 192.168.2.0/24
BGP routing table entry for 10.0.0.22:100:192.168.2.0/24, version 9
not allocated
Paths: (1 available, best #1)
  Not advertised to any peer
  65001
    10.0.0.22 (metric 40) from 10.0.0.1 (10.0.0.22)
      Origin IGP, metric 0, localpref 100, valid, internal, best (First path received)
      Extended Community: RT:65000:100
      Originator: 10.0.0.22, Cluster list: 10.0.0.1
      Remote label: 80
      Last update: Tue Sep  9 16:51:35 2025
[edit]
vyos@PE1#
```

- `show ip route vrf BLUE bgp` or checking all the prefix learning on BGP within VRF BLUE:

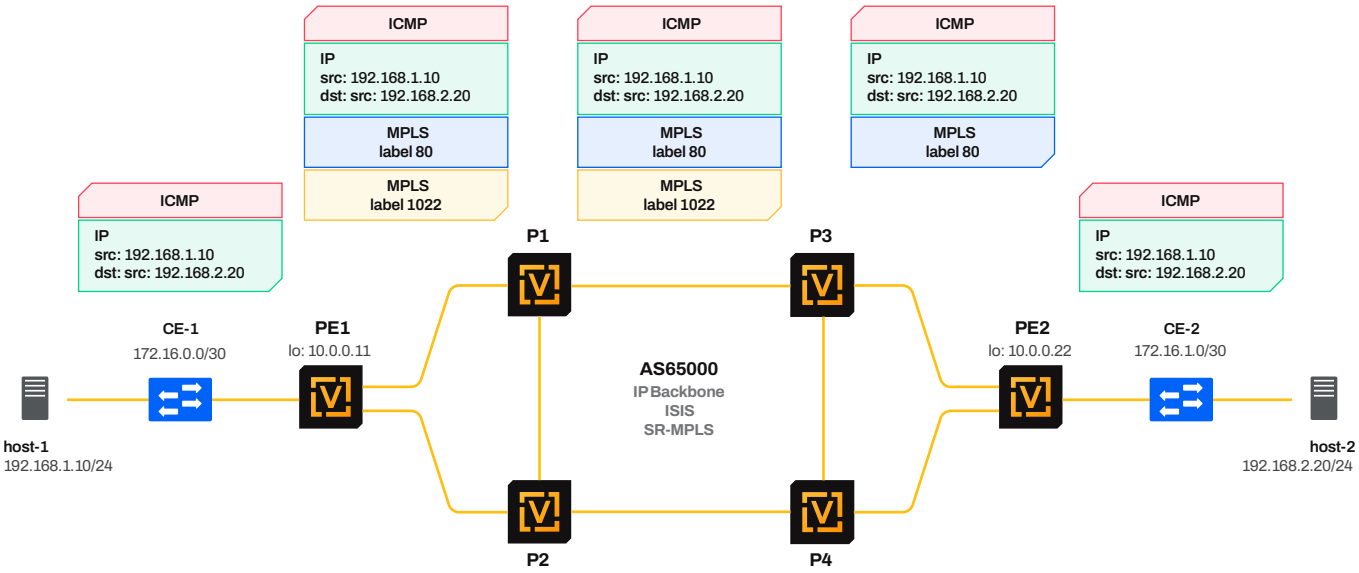
```
vyos@PE1# run show ip route vrf BLUE bgp
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

VRF BLUE:
B>  172.16.1.0/30 [200/0] via 10.0.0.22 (vrf default) (recursive), label 80, weight 1, 00:08:26
  *                         via 10.11.1.1, eth0 (vrf default), label 1022/80, weight 1, 00:08:26
B>* 192.168.1.0/24 [20/0] via 172.16.0.2, eth2, weight 1, 01:38:56
B>  192.168.2.0/24 [200/0] via 10.0.0.22 (vrf default) (recursive), label 80, weight 1, 00:08:26
  *                         via 10.11.1.1, eth0 (vrf default), label 1022/80, weight 1, 00:08:26
[edit]
vyos@PE1#
```

Host-1 (192.168.1.0/24) prefix is known by eBGP and reachable via **CE1** on interface eth2. The route to **192.168.2.0/24 (Host-2 prefix)** is reachable through PE2 (10.0.0.22) using MPLS, with label **80** for the VPN service and label **1022** for transport over eth0.

Packet capture

Ping test from **host-1** to **host-2**



CE1 - PE1:

```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface -, id 0
▸ Ethernet II, Src: 0c:c1:82:8e:00:00 (0c:c1:82:8e:00:00), Dst: 0c:c8:e6:b1:00:02 (0c:c8:e6:b1:00:02)
▸ Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.2.20
▸ Internet Control Message Protocol
```

Original ICMP IP packet encapsulation

PE1 - P1:

```
▸ Frame 1: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface -, id 0
▸ Ethernet II, Src: 0c:c8:e6:b1:00:00 (0c:c8:e6:b1:00:00), Dst: 0c:a8:ab:2d:00:00 (0c:a8:ab:2d:00:00)
▾ MultiProtocol Label Switching Header, Label: 1022, Exp: 0, S: 0, TTL: 62
    0000 0000 0011 1111 1110 .... .... .... = MPLS Label: 1022 (0x003fe)
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...0 .... .... = MPLS Bottom Of Label Stack: 0
    .... .... .... .... .... .... 0011 1110 = MPLS TTL: 62
▾ MultiProtocol Label Switching Header, Label: 80, Exp: 0, S: 1, TTL: 62
    0000 0000 0000 0101 0000 .... .... .... = MPLS Label: 80 (0x00050)
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...1 .... .... = MPLS Bottom Of Label Stack: 1
    .... .... .... .... .... .... 0011 1110 = MPLS TTL: 62
▸ Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.2.20
▸ Internet Control Message Protocol
```

Label **1022** serves as the transport encapsulation label.

Label **80** serves as the VPN service encapsulation label. Bottom of Label Stack: 1

In an MPLS (Multiprotocol Label Switching) packet, the "Bottom of Stack" (S) bit is a 1-bit field within the MPLS label header that indicates whether a label is the final one in a stack of multiple labels. When the S bit is set to 1, it signifies that the label is the last one in the stack, and the network device should then look at the network layer header immediately following it.

P1 - P3

```
▸ Frame 3: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface -, id 0
▸ Ethernet II, Src: 0c:a8:ab:2d:00:01 (0c:a8:ab:2d:00:01), Dst: 0c:80:01:81:00:00 (0c:80:01:81:00:00)
▾ MultiProtocol Label Switching Header, Label: 1022, Exp: 0, S: 0, TTL: 61
    0000 0000 0011 1111 1110 .... .... .... = MPLS Label: 1022 (0x003fe)
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...0 .... .... = MPLS Bottom Of Label Stack: 0
    .... .... .... .... .... .... 0011 1101 = MPLS TTL: 61
▾ MultiProtocol Label Switching Header, Label: 80, Exp: 0, S: 1, TTL: 62
    0000 0000 0000 0101 0000 .... .... .... = MPLS Label: 80 (0x00050)
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...1 .... .... = MPLS Bottom Of Label Stack: 1
    .... .... .... .... .... .... 0011 1110 = MPLS TTL: 62
▸ Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.2.20
▸ Internet Control Message Protocol
```

PE1 - P1:

```
▸ Frame 7358: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface -, id 0
▸ Ethernet II, Src: 0c:80:01:81:00:01 (0c:80:01:81:00:01), Dst: 0c:70:fe:eb:00:00 (0c:70:fe:eb:00:00)
▾ MultiProtocol Label Switching Header, Label: 80, Exp: 0, S: 1, TTL: 62
    0000 0000 0000 0101 0000 .... .... .... = MPLS Label: 80 (0x00050)
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...1 .... .... = MPLS Bottom Of Label Stack: 1
    .... .... .... .... .... .... 0011 1110 = MPLS TTL: 62
▸ Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.2.20
▸ Internet Control Message Protocol
```

Only a single MPLS label (VPN service) is present in the encapsulation as a result of **Penultimate Hop Popping** (PHP) implemented by P3