### Terminology Reference

This is a glossary of acronyms and terms used throughout this document.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS</td>
<td>Autonomous System is a collection of connected IP routing prefixes under the control of one or more network operators on behalf of a single administrative entity or domain</td>
</tr>
<tr>
<td>BFD</td>
<td>Bidirectional Forwarding Detection is a UDP based protocol that provides fast detection of layer-3 next hop failures; it is used in conjunction with a routing protocol, in this case, BGP</td>
</tr>
<tr>
<td>BGP</td>
<td>Border Gateway Protocol is a standardized routing protocol used to exchange routing and reachability data among autonomous systems</td>
</tr>
<tr>
<td>L2VPN</td>
<td>Layer 2 VPN; in this design guide context, it is a bridge domain extension across multiple dispersed physical locations over a generic IP transport infrastructure</td>
</tr>
<tr>
<td>L3VPN</td>
<td>Layer 3 VPN; in this design guide context, it is an IP domain extension across multiple dispersed physical locations over a generic IP transport infrastructure</td>
</tr>
<tr>
<td>Overlay</td>
<td>In the VXLAN context, this refers to all the elements built on top of the generic IP transport infrastructure in order to offer the L2VPN and L3VPN functionalities</td>
</tr>
<tr>
<td>Underlay</td>
<td>In the VXLAN context, this refers to the generic IP transport infrastructure used to ensure IP communication among all Data Centers</td>
</tr>
<tr>
<td>VIP</td>
<td>Virtual IP is an IP address that does not correspond to an actual physical device; in this design guide context, it is the IP used by the VRRP instances and the VTEPs</td>
</tr>
<tr>
<td>VRRP</td>
<td>Virtual Router Redundancy Protocol is a networking protocol that provides redundancy of routing paths by creation of virtual routers, which are an abstract representation of multiple routers, i.e. master and backup routers, acting as a group</td>
</tr>
<tr>
<td>VXLAN</td>
<td>Virtual Extensible LAN is a Layer 2 overlay scheme over a Layer 3 network. It uses MAC-in-UDP encapsulation to provide extension of Layer 2 segments across IP transport networks.</td>
</tr>
<tr>
<td>VTEP</td>
<td>VXLAN Tunnel Endpoint is the entity responsible for encapsulating / de-encapsulating VXLAN packets</td>
</tr>
<tr>
<td>VTEP HA</td>
<td>VTEP High Availability refers to a mechanism designed to ensure redundancy of the VTEP entity</td>
</tr>
<tr>
<td>Cluster</td>
<td>A pair of adjacent Netvisor powered switches acting as one logical unit for high availability</td>
</tr>
<tr>
<td>Adaptive Cloud Fabric or Fabric</td>
<td>A number of Netvisor powered switches that operate, and are managed, as a single holistic entity</td>
</tr>
<tr>
<td>vRouter</td>
<td>An object used to provide routing between subnets, VLANs and/or vNETs – the vRouter runs in a dedicated network container</td>
</tr>
<tr>
<td>vNET</td>
<td>A Virtual NETwork is a partition of the Fabric. A vNET is defined by a group of network objects that can operate independently and have dedicated resources, providing multi-tenancy and network segmentation.</td>
</tr>
<tr>
<td>Pluribus UNUM</td>
<td>Pluribus UNUM™ Unified Management, Automation and Analytics Platform software</td>
</tr>
<tr>
<td>Insight Analytics</td>
<td>Insight Analytics (IA) is Network Performance Management (NPM) add-on module to the UNUM platform</td>
</tr>
<tr>
<td>vLAG</td>
<td>Virtual Link Aggregation Group is a Netvisor OS technology to connect multiple switches to other devices or to other switches for high availability</td>
</tr>
<tr>
<td>vLE</td>
<td>Virtual Link Extension is a Netvisor technology that allows defining Layer 1 pseudowires that can emulate a direct connection between devices on top of an IP transport network</td>
</tr>
<tr>
<td>Out-of-Band interface</td>
<td>Dedicated out-of-band port on Netvisor powered switches, used either as a management-only interface or as a Fabric-control port to form the Fabric and exchange Fabric information over the out-of-band management network</td>
</tr>
<tr>
<td>In-band interface</td>
<td>Internal management interface used as a Fabric-control port when building a Fabric over any IP network</td>
</tr>
<tr>
<td>VRF</td>
<td>VRF is a technology that allows multiple routing spaces coexist on the same switch; it complements the vRouter construct, offering a highly scalable solution</td>
</tr>
</tbody>
</table>
Table of Contents

Next-Generation Data Center Interconnect 3
Design Goals 4
Planning Phase 4
  Step 1 – Design the High-level Network Architecture for the DCI connection 5
  Step 2 – Plan the Fabric layout and identify the Layer 2 extension requirements between data centers 6
  Step 3 – Plan the “underlay” connectivity (Layer 3 » Layer 1 Architecture) 6
  Step 4 – Re-evaluate and re-validate the previous steps based on any possible limitations 8
Implementation Phase 8
  Detailed Underlay Implementation Steps 10
    Step 1 – Choose the “root” node and assign it an in-band IP 11
    Step 2 – Create a vRouter, configure all the required interfaces and ensure IP reachability across the network (BGP) 11
    Step 3 – Create the Fabric and define all the In-band subnets of the nodes that will be part of the Fabric 13
    Step 4 – Configure the next switch with the In-band IP, its own vRouter and join the existing Fabric 14
    Step 5 – Configure the rest of the nodes across data centers to join the existing Fabric 17
    Step 6 – Configure the clusters and the VRRP instances to be used for the HA VTEP 19
    Step 7 – Configure the VXLAN Tunnel End-Points (VTEPs) and implicitly the tunnels between them 22
    Step 8 – Reserve a port for VXLAN loopback / packet recirculation 23
    Step 9 – Configure the vLAGs connecting to edge devices 23
    Step 10 – Validate the underlay configuration 24
  Detailed “Overlay” Implementation Steps 25
    Step 1 – Configure VLANs stretching across data centers and the associated VXLAN IDs 25
    Step 2 – Set the ports facing the edge devices as tagged ports of these new VLANs 25
    Step 3 – Add the VXLAN IDs to the required VTEPs 26
    Step 4 – Create the VRFs that are providing the distributed routing and segmentation to previously created VLANs 27
    Step 5 – Create the Subnets assigned to each of the VLANs as the distributed anycast gateway and attach them to a VRF 27
    Step 6 – Configure the VRF gateways, indicating the northbound next-hops for traffic originating from previously created Subnets 28
    Step 8 – Validate the overlay layer 31
Advanced DCI features 32
  Transparent Pseudowire with Virtual Link Extension 32
Virtual Link Extension Implementation Steps 33
    Step 1 – Define tunnels between the switches where access ports of the vLE are located 33
    Step 2 – Create a transparent bridge domain with associated VLAN/VNI for the client port pair that attaches to the pseudowire 33
    Step 3 – Create the vLE object with port link tracking 34
    Step 4 – Attach the transparent VXLAN VNI to tunnels 34
    Step 5 – Validate the vLE functionality 34
    Step 6 – Repeat previous steps to create redundant vLE groups 35
Conclusion 36
Pluribus Networks offers a unique and highly differentiated approach to software-defined networking and is driving the revolution of the networking market to a more open operating environment. The Adaptive Cloud Fabric™ architecture enables enabling organizations to build scalable private and public clouds that improve service velocity, performance, and reliability. The company’s innovative Netvisor® ONE software virtualizes open networking hardware to build a holistic, distributed network that is more intelligent, automated, and resilient. The Insight Analytics™ platform leverages embedded telemetry and other data sources to enable pervasive visibility across the network to reveal network and application performance that speeds troubleshooting and improves operational and security intelligence.

This Pluribus Validated Design Guide offers reference design architecture for planning and implementing a Data Center Interconnect (DCI) deployment by making the best use of the attributed of the Pluribus Adaptive Cloud Fabric architecture as they apply to network Fabrics that are extended across a generic IP-routed network. This guide is organized to provide a step-by-step process, from the design phase, through implementation, to final deployment verification testing. The intended audience for this document are Network Architects and Engineers involved in designing and configuring the intra- and inter-data center connectivity using the Netvisor ONE OS and Adaptive Cloud Fabric architecture.

Next-Generation Data Center Interconnect

Data Center Interconnect (DCI) is the process of connecting geographically distributed data center locations, in order to form a consistent set of IT resources that can be used for scaling the data center, enabling workload sharing, mobility and establishing high-availability and supporting disaster recovery requirements.

Traditionally, Layer 3 Internet Protocol-based (IP) connectivity has been used to route traffic between geographically distributed data centers using long-established traffic routing protocols such as the Border Gateway Protocol (BGP). However, in more recent times the development of network virtualization and geo-clustering technologies for the data center has driven the adoption of new interconnection models that require the presence of direct Layer 2 adjacencies between physical and virtual nodes. In contrast to other closed proprietary DCI approaches, which require significant investments in complex and rigid hardware and software combinations.

Pluribus Networks has changed this model to enabling building a flexible distributed interconnect architecture that enables a software-defined unified interconnection architecture that is based upon industry standard Virtual Extensible LAN (VXLAN) encapsulation technologies. Consequently, architecting a DCI deployment with the Pluribus Adaptive Cloud Fabric extends Layer 2 domains across geographically distributed locations leveraging a VXLAN-powered hardware-based data plane coupled with an intelligent and resilient control plane to enable high-speed interconnection with superior scalability and significantly more functionality that traditional approaches.
Design Goals

The main objectives considered in this design guide includes:

- Interoperable and standard based layer-2 extension across data centers
- Anycast gateway distributed across data centers
- Transparency to the Core infrastructure with interoperability with any existing IP transport network
- Deployed on Open Networking switches enabling flexible hardware choices
- Multi-terabit switching capacity with performance and latency predictability
- High Availability with fast re-convergence in-case of a failure event
- Easy to scale to multiple-sites with fault isolation among sites
- Simple to manage with a single management Fabric across multiple datacenters
- End-to-end visibility of network and client-server application traffic
- Advanced DCI features include layer 1 emulation with transparent point-to-point services and multi-tenancy

Planning Phase

During this phase, we want to define the detailed physical connectivity, the Layer 2 and Layer-3 topologies, the prefix schemas and the protocols that will be use for the design in order to achieve the goals as described earlier in this document.

For our reference architecture, we will use a design consisting of three geographically distributed data centers. Figure-1 represents a high-level reference architecture of three Data Center that will be interconnected. The switches represented in grey provide high-speed Layer 3 connectivity among the three sites; they can either be third-party switches or they can run the Pluribus Netvisor ONE operating system.

The green icons represent Pluribus Netvisor ONE powered nodes that are members of the same Pluribus Adaptive Cloud Fabric. In the DCI use case Fabric control plane communications happens over the IP core transport using the in-band interfaces of the switches.

Figure 1:
Building the unified Adaptive Cloud Fabric across multiple data centers over an agnostic IP-routed core.
Planning Steps

The design phase is critical to ensure the architecture will deliver a robust solution fulfilling all the requirements in terms of functionality, redundancy and bandwidth availability. This Design Guide will navigate through this process, offering a comprehensive example of the steps involved. The design phase is also intended to be a dynamic iterative process. For instance, a specific deployment scenario could hit a limitation or constraint, such as port availability, which could trigger a re-evaluation of previous higher-level design decisions.

Using a top-down approach, the design steps are ordered hierarchically from the end goals of the solution down to the details of logical and physical connectivity.

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Design the High-level Network Architecture for the DCI connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2</td>
<td>Plan the Fabric layout and identify the Layer 2 extension requirements between DCs</td>
</tr>
<tr>
<td>Step 3</td>
<td>Plan the “Underlay” connectivity (Layer 3 » Layer 1 Architecture)</td>
</tr>
<tr>
<td>Step 4</td>
<td>Re-evaluate / validate the previous designs based on eventual per-case limitations</td>
</tr>
</tbody>
</table>

Step 1 – Design the High-level Network Architecture for the DCI Connection

Figure 2 below depicts the high-level representation of the set-up that we are going to use throughout this document as an example of the Pluribus DCI implementation.

There are three data centers connected together through a mix of 10/40G redundant links. The core of the network is represented by third-party Layer 3 switches, depicted as grey icons, to demonstrate that the DCI solution is interoperable and transparent to the IP underlay transport connecting the DCI locations. The Netvisor powered switches are depicted with green icons and are placed at the edge connected to hosts, VMs, and third-party switches. Synthetic traffic generator ports are also used to emulate these edge devices for validation purposes.

Figure 2: High-level view of the DCI environment.
Step 2 – Plan the Fabric layout and identify the Layer 2 extension requirements between data centers

All the green switches at the network edge will be part of the same Adaptive Cloud Fabric instance, thus simplifying and centralizing the management and monitoring of all the DCI sites. The Fabric is built by using the in-band management interfaces of the switches. To allow these interfaces to communicate to each other, the underlay layer including the third-party switches in the network core, must provide the IP reachability among all the in-band IP addresses of the Netvisor nodes. By having all the switches in the environment running the Netvisor OS, the user benefits from key features such as end-to-end visibility, single-point-of-management, security and automation. The different VMs located on different hosts throughout the data centers, are represented in different colors as they belong to different Layer 2 VLANs stretched across DCs.

Step 3 – Plan the “underlay” connectivity (Layer 3 » Layer 1 Architecture)

The Layer 3 topology used in the current design guide is represented in Figure 4. This Layer 3 infrastructure is used to carry both the user data traffic, such as traffic in between VMs across data centers, and control traffic exchanged by the nodes in order to operate the Fabric.

We will configure each Netvisor powered switch with both an in-band interface, used to exchange Fabric related information with the other nodes, and a loopback interface for validating and troubleshooting potential connectivity issues.

The comprehensive IP address plan used in the topology has the following attributes:

- For simplicity all prefix masks used are either /24 for subnets and /32 for loopback addresses
- Each device has its own ID (represented in the white circle)
- Each subnet used to interconnect two Layer 3 speakers is using the following IP convention:
  - 10.<SW1_ID>.<SW2_ID>.0/24, the last octet of the IP is either <SW1_ID> or <SW2_ID>
- Each device uses three local interfaces:
  - Loopback interface (used for routing): 10.10.<SW_ID>.<SW_ID>/32
  - In-band interface (used to build the Fabric): 10.20.<SW_ID>.<SW_ID>/24
  - VTEP interface (used for VXLAN tunnel termination): 10.40.<SW1SW2_IDs>.<SW_ID>/24
- A local subnet is created on each site to validate Layer 3 inter-site traffic in the underlay network using the following convention:
  - Subnet: 172.20. <SW1SW2_IDs>.0/24
  - Routers: using either <SW1_ID> or <SW2_ID> as last octet for the Primary IP or Physical IP (PIP)
  - VRRP: a Virtual IP (VIP) is used in each subnet to provide redundancy to the devices connected using <SW1SW2_IDs> as the last octet for the VIP
The Layer 3 routing at the underlay level is done by BGP; the details on the AS numbering are presented in Figure 5 below. The design includes the use of ECMP in order to load-balance the traffic on all the available connections within the setup. BFD is also enabled to enhance the failover times in case of a link-down/node-down event.

NOTE: The use of BGP is not mandatory; OSPF or Static Routes could be also used as long as the IP reachability across the network is ensured. BGP gives more scalability and flexibility and thus is the preferred option for this Design Guide.
From a Layer 1 perspective, the physical connectivity and the detailed port numbers used in the current setup are documented in Figure:

**Figure 6:** The physical DCI connectivity.

**Step 4 – Re-evaluate and re-validate the previous steps based on any possible limitations**

As mentioned above, the design phase is usually an iterative process. Once all design steps are complete, it is a good time to revisit the design goals of the architecture and make sure all the requirements are fulfilled. The sooner we identify the need for a change in the design, the easier is to manage and mitigate the implications.

**Implementation Phase**

Once the design phase is completed and the desired configuration is known, the implementation phase can start. For simplicity, we assume that the IP core transport is already operational: this is indeed the case in many real-life scenarios, where the IP connectivity between different data centers are already in place, before the different services are extended. Hence this guide does not cover the Generic IP Transport configuration and assumes it is implemented using BGP as routing protocol.

The provisioning of the underlay is the initial building block of the implementation that provides reachability between the in-band interface of all Netvisor powered nodes and acts as an IP infrastructure layer on top of which it is possible to create the Fabric. We then create the cluster pairs and the VRRP instances which will be used in the subsequent step to configure VTEP high availability (VTEP HA). The provisioning of VXLAN tunnels between the VTEPs is fully automated and does not require any operational effort. At the end of this phase, the system will be in a state represented in Figure 7 below.
Then we will continue with the first component of the overlay configuration, where we create the overlay VLANs, define the VLAN IDs ↔ VXLAN VNIs mapping and assign the VXLAN VNIs to the previously created VTEPs. This will enable the extension of the Layer 2 VLANs across the different datacenters. At the end of this phase, the virtual machines connected in the different datacenters will be able to have mutual reachability within the same VLANs.

The second element of the overlay configuration enables distributed inter-VLAN routing for the different extended VLANs, by creating distributed VRFs. Pluribus defines the VRF as a Fabric scope object that acts a distributed anycast gateway, allowing hosts within the subnets that are attached to it, to reach each other using the same gateway IP/MAC regardless of their physical location. The VRF also provides an optimized traffic path since the routing can take place anywhere in the Fabric, avoiding traffic hair-pinning to a central location, even within the same data center. Finally, the VRF provides an extra highly scalable level of segmentation on top of what was previously offered by the existence of distinct vRouters; in fact, two subnets belonging to distinct VRFs communicate to each other only through an external router or, as in most of the cases, via an external firewall where we can define all the required inter-VRF policies.

At this point, the virtual machines connected in the different datacenters will be able to reach each other even if they are part of different subnets, as long as these subnets are assigned to the same VRF object; the system will be in a state represented in Figure 8 below.
So far, we have addressed the traffic paths within the same or between different extended Layer 2 domains across the multiple datacenters or east/west; the last component of the overlay configuration targets the north/south traffic paths, allowing hosts connected to subnets behind the VRFs in the Fabric, to access services which are outside of the data centers, including the public internet. This is achieved by pointing each VRF to a next-hop gateway, or two for equal cost redundancy. This last configuration component has only switch scope relevance, allowing this way to have multiple northbound connections for a certain VRF, eventually one or two per datacenter for an optimum traffic path.

At the end of this phase, the virtual machines connected in different data centers will be able to reach services located outside of the Fabric and the controlled data centers; the system will be in a state represented in Figure 9 below.

**Figure 9:**
High-level view after configuring the VRFs gateways for northbound traffic.

**Detailed Underlay Implementation Steps**

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Choose the “root” node and assign it an in-band IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2</td>
<td>Create a vRouter, configure all the required interfaces and ensure IP reachability across the network (BGP)</td>
</tr>
<tr>
<td>Step 3</td>
<td>Create the Fabric and indicate all the in-band subnets of the nodes that will be part of the Fabric</td>
</tr>
<tr>
<td>Step 4</td>
<td>Configure the next switch with in-band IP, its own vRouter and join the existing Fabric</td>
</tr>
<tr>
<td>Step 5</td>
<td>Configure the rest of the nodes across data centers and join them to the same Fabric</td>
</tr>
<tr>
<td>Step 6</td>
<td>Configure the clusters and the VRRP instances to be used for VTEP HA</td>
</tr>
<tr>
<td>Step 7</td>
<td>Configure the VTEPs and implicitly the tunnels between them</td>
</tr>
<tr>
<td>Step 8</td>
<td>Reserve a port for VXLAN Loopback / packet recirculation (only for BUM traffic)</td>
</tr>
<tr>
<td>Step 9</td>
<td>Configure the vLAGs to connect to Edge devices</td>
</tr>
<tr>
<td>Step 10</td>
<td>Validate the underlay configuration</td>
</tr>
</tbody>
</table>
Step 1 – Choose the “root” node and assign it an in-band IP

Select a switch (in our case it is sw-test-s4048-1) as the root node where the Fabric is initially created and assign its in-band interface IP address. In subsequent steps all the other Netvisor powered nodes will join the already existing Fabric by pointing to this In-band IP interface. This is why it is important to ensure the IP reachability between these in-band interfaces of all the nodes; this will be covered in the next steps.

NOTE: Despite using the word “root node”, once the Fabric is joined by multiple nodes there is no hierarchical relevance for the switch where the Fabric is initially created, nor for any other Fabric member switch, being the Fabric is a full symmetric construct.

Figure 10:
Zoom view on DC1; select a “root” switch and start creating the Fabric.

Step 2 – Create a vRouter, configure all the required interfaces and ensure IP reachability across the network (BGP)

The vRouter that is going to be provisioned provides connectivity between the in-band interfaces of the nodes that are part of the Fabric over the BGP core. The first BGP connection is configured towards the generic core switch sw-test-6000-1. This step can be achieved with a single command or it can be broken in different commands. For better understanding of each atomic operation, we discuss first the option with multiple commands.

Figure 11:
Zoom view on sw-test-s4048-1; create the vRouter and attach all the required interfaces.
a) Create the vRouter with a specific attribute, Fabric-comm, used by Netvisor to identify a vRouter instance dedicated to Fabric control plane communication.

```
CLI (network-admin@sw-test-s4048-1) > vrouter-create name S4048-1 Fabric-comm
bgp-as 65012 router-id 10.10.1.1
```

**NOTE:** Even for switch models that are traditionally limited to one single vRouter instance, it is still allowed to create this additional one used for Fabric in-band communication.

b) Configure the Fabric-comm vRouter with an internal interface to provide connectivity to the in-band interface

```
CLI (network-admin@sw-test-s4048-1) > vrouter-interface-add vrouter-name S4048-1 vlan 1 ip 10.20.1.101/24
```

c) Configure the first BGP neighbor relationship with the upstream core switch s6000-1 and enable BFD for fast convergence

```
CLI (network-admin@sw-test-s4048-1) > vrouter-bgp-add vrouter-name S4048-1 neighbor 10.1.11.11 remote-as 65112 bfd
```

**QUICK CHECK:** Since the core switches are already configured, we should already establish a BGP connection with sw-test-6000-1 that can be confirmed by running the vrouter-bgp-neighbor-show command. We can also validate the routes we receive from this neighbor by running a vrouter-routes-show and vrouter-fib-routes-show commands.

If the dynamic protocol of our choice is BGP, the vRouter used for the Fabric control plane can be created in one single command that simultaneously forms an adjacency between the In-band interface and an interface on the vRouter itself, and additionally enables BGP and configures the first BGP neighbor adjacency. This command would summarize in one transaction the actions described from point a thru c above.

```
CLI (network-admin@sw-test-s4048-1) > Fabric-comm-vrouter-bgp-create name S4048-1
bgp-as 65012 router-id 10.10.1.1 bgp-nic-ip 10.1.11.1/24 bgp-nic-l3-port 57
neighbor 10.1.11.11 remote-as 65112 bfd in-band-nic-ip 10.20.1.101 in-band-nic-netmask 24
```

For simplicity, when creating Fabric-comm vRouters on the other Fabric nodes in steps 4 and 5 below, we will then issue a single-command `Fabric-comm-vrouter-bgp-create`.

a) Enable ECMP in order to load balance and make use of all the available connections in between nodes:

```
CLI (network-admin@sw-test-s4048-1) > vrouter-modify name S4048-1 bgp-max-paths 4 bgp-bestpath-as-path multipath-relax
```

**NOTE:** In this specific topology example, we could have limited the number of BGP paths to 2 as we only have two core routers connected.
b) For high-availability, configure the BGP connection towards the second switch of the Core (s6000-2). Enable BFD to enhance the re-convergence time in case of a failover event:

```
CLI (network-admin@sw-test-s4048-1) > vrouter-interface-add vrouter-name S4048-1 l3-port 65 ip 10.1.12.1/24
CLI (network-admin@sw-test-s4048-1) > vrouter-bgp-add vrouter-name S4048-1 neighbor 10.1.12.12 remote-as 65112 bfd
```

a) As good design practice, configure a Loopback interface for the switch and connect it also to the vRouter; advertise its prefix into the underlay routing protocol, as this will help validate and debug possible connectivity issues:

```
CLI (network-admin@sw-test-s4048-1) > vrouter-loopback-interface-add vrouter-name S4048-1 ip 10.10.1.1 index 1
CLI (network-admin@sw-test-s4048-1) > vrouter-bgp-network-add vrouter-name S4048-1 network 10.10.1.1 netmask 255.255.255.255
```

b) Configure the VLAN interface used to connect to the other cluster member, which is part of the same BGP AS. First, create a VLAN and add it to a LAG or trunk port (port 128) which has been formed automatically with Netvisor Auto-trunk feature by aggregating physical ports 49 and 53:

```
CLI (network-admin@sw-test-s4048-1) > vlan-create id 12 scope local description L3-BGP-INTERCO-DC1 ports none
CLI (network-admin@sw-test-s4048-1) > vlan-port-add vlan-id 12 ports 128 tagged
CLI (network-admin@sw-test-s4048-1) > vrouter-interface-add vrouter-name S4048-1 vlan 12 ip 10.1.2.1/24
CLI (network-admin@sw-test-s4048-1) > vrouter-bgp-add vrouter-name S4048-1 neighbor 10.1.2.2 remote-as 65012 next-hop-self bfd
```

NOTE: Please remark the “next-hop-self” keyword used here to guarantee reachability to external networks for cluster peer in the same AS. At this point, the iBGP session with the cluster peer, which is not yet configured, so it is not up.

**Step 3 – Create the Fabric and define all the In-band subnets of the nodes that will be part of the Fabric**

a) Create the Fabric by using the in-band interface option:

```
CLI (network-admin@sw-test-s4048-1) > Fabric-create name TME-DCI Fabric-network in-band control-network in-band
```

**QUICK CHECK:** Validate the Fabric state and the node membership state by running the Fabric-node-show command. At this point there is one single member.

b) Define all the subnets of the In-band interfaces of the Netvisor powered nodes that will be part of the Fabric. These commands are run only once with Fabric scope; all the other nodes joining the Fabric later will inherit the information dynamically. This way, each node of the Fabric will self-instruct on how to reach the other nodes, by automatically creating local routes from its own in-band interface towards these subnets, using the local vRouter interface connected to the in-band, as a next hop:
a) Connecting to sw-test-s4048-2 console, assign an In-band IP address:

```bash
CLI (network-admin@sw-test-s4048-2) > switch-setup-modify in-band-ip 10.20.2.2/24
```

b) Create a vRouter, enable BGP and establish the relationship with the first neighbor (in this case the same sw-test-s6000-1):

```bash
CLI (network-admin@sw-test-s4048-2) > Fabric-comm-vrouter-bgp-create name S4048-2 bgp-as 65012 router-id 10.10.2.2 bgp-nic-ip 10.2.11.2/24 bgp-nic-l3-port 57 neighbor 10.2.11.11 remote-as 65112 bfd Fabric-network 10.20.1.0/24 in-band-nic-ip 10.20.2.102 in-band-nic-netmask 24
```

**QUICK CHECK:** Verify that routes corresponding to these In-band prefixes are automatically created by running the `switch-route-show` command (part of the output has been omitted to keep it concise)

```
CLI (network-admin@sw-test-s4048-1*) > Fabric-in-band-network-show
```

```
CLI (network-admin@sw-test-s4048-1*) > switch-route-show
```

**Step 4 – Configure the next switch with the In-band IP, its own vRouter and join the existing Fabric**

Replicate an equivalent configuration on the next switch that will be part of the Fabric (in our example it is sw-test-s4048-2).

a) Connecting to sw-test-s4048-2 console, assign an In-band IP address:

```bash
CLI (network-admin@sw-test-s4048-2) > switch-setup-modify in-band-ip 10.20.2.2/24
```

b) Create a vRouter, enable BGP and establish the relationship with the first neighbor (in this case the same sw-test-s6000-1):
NOTE: Please notice how the keyword “fabric-network 10.20.1.0/24” refers to the in-band interface of the switch where the Fabric was initially created, as an add-on to the equivalent command ran on the first node. This is needed to instruct the current node on how to reach the existing Fabric; once it will join the Fabric, it will retrieve automatically the in-band subnets to all the other nodes. The Fabric-network keyword is basically equivalent to the switch-route-add command that can be used to provide connectivity to sw-test-s4048-1’s in-band interface from the local in-band interface.

For example, when using vrouter-create [...] Fabric-comm instead of Fabric-comm-vrouter-bgp-create, the following additional command would be required in order to instruct Netvisor to use the vRouter adjacency on the In-band interface to reach the other Netvisor nodes, whose IP address is part of the aggregated prefix 10.20.0.0/16

```
CLI (network-admin@sw-test-s4048-2) > switch-route-create network 10.20.0.0/16
gateway-ip 10.20.2.101
```

Or, using more specific prefixes:

```
CLI (network-admin@sw-test-s4048-2) > switch-route-create network 10.20.1.0/24
gateway-ip 10.20.2.101
```

c) Enable ECMP for load balancing purposes:

```
CLI (network-admin@sw-test-s4048-2) > vrouter-modify name S4048-2 bgp-max-paths 4 bgp-bestpath-as-path multipath-relax
```

d) Add the second core BGP neighbor for redundancy:

```
CLI (network-admin@sw-test-s4048-2) > vrouter-interface-add vrouter-name S4048-2 l3-port 65 ip 10.2.12.2/24
CLI (network-admin@sw-test-s4048-2) > vrouter-bgp-add vrouter-name S4048-2 neighbor 10.2.12.12 remote-as 65112 bfd
```
PLURIBUS NETWORKS   |   DATA CENTER INTERCONNECT VALIDATED DESIGN GUIDE    |

**e)** Add the loopback interface for connectivity validation and troubleshooting purposes:

```plaintext
CLI (network-admin@sw-test-s4048-2) > vrouter-loopback-interface-add vrouter-name S4048-2 ip 10.10.2.2 index 1
CLI (network-admin@sw-test-s4048-2) > vrouter-bgp-network-add vrouter-name S4048-2 network 10.10.2.2 netmask 255.255.255.255
```

**f)** Add another VLAN interface and configure the iBGP adjacency with the cluster peer:

```plaintext
CLI (network-admin@sw-test-s4048-2) > vlan-create id 12 scope local description L3-BGP-INTERCO-DC1 ports none
CLI (network-admin@sw-test-s4048-2) > vlan-port-add vlan-id 12 ports 128 tagged
CLI (network-admin@sw-test-s4048-2) > vrouter-interface-add vrouter-name S4048-2 vlan 12 ip 10.1.2.2/24
CLI (network-admin@sw-test-s4048-2) > vrouter-bgp-add vrouter-name S4048-2 neighbor 10.1.2.1 remote-as 65012 next-hop-self bfd
```

**QUICK CHECK:** Validate the reachability of the Fabric root by pinging its In-band interface from sw-test-s4048-2’s In-band interface, using the `ping` command. Additionally, if experiencing connectivity issue between In-band interfaces, you can use a `vrouter-ping` command to check reachability between vRouter containers on both nodes. If that works as expected, the second node is ready to join the Fabric.

```plaintext
CLI (network-admin@sw-test-s4048-2) > ping 10.20.1.1
PING 10.20.1.1 (10.20.1.1) 56(84) bytes of data.
64 bytes from 10.20.1.1: icmp_seq=1 ttl=60 time=1.12 ms
64 bytes from 10.20.1.1: icmp_seq=2 ttl=60 time=1.16 ms
```

**g)** Join the Fabric that was previously created by indicating the in-band IP of the first node:

```plaintext
CLI (network-admin@sw-test-s4048-2) > Fabric-join switch-ip 10.20.1.1
```

**QUICK CHECK:** Validate that the Fabric has two nodes at this moment by running the `Fabric-node-show` command.

The second node, joining the Fabric, will first appear as “off-line” and then will transition to “in-band-only” or “on-line”, depending on connectivity we have between out-of-band management ports on each switch.
From now on, any CLI command or configuration change for the nodes that are already part of the Fabric, can be applied from any switch console with consistent atomic transactions; the Fabric provides one single-point-of-management for all of its nodes.

**NOTE:** On nodes joining the Fabric, there is no need to declare the prefixes of the in-band interfaces of the other Fabric nodes: that information is automatically imported from the Fabric itself.

**Step 5 – Configure the rest of the nodes across data centers to join the existing Fabric**

Repeat a similar configuration on all the other nodes that we want to be part of the same Fabric. Proceeding with `sw-test-s4048-3`, `sw-test-s4048-4`, `sw-test-s4048-5` and `sw-test-s4048-6` as in Figure 13 below.

For `sw-test-s4048-3` apply the following configuration and validation commands:

```plaintext
CLI (network-admin@sw-test-s4048-3 > switch-setup-modify in-band-ip 10.20.3.3/24
CLI (network-admin@sw-test-s4048-3 > Fabric-comm-vrouter-bgp-create name S4048-3 bgp-as 65034 router-id 10.10.3.3 bgp-nic-ip 10.3.13.3/24 bgp-nic-l3-port 57 neighbor 10.3.13.13 remote-as 65134 bfd Fabric-network 10.20.1.0/24 in-band-nic-ip 10.20.3.103 in-band-nic-netmask 24
CLI (network-admin@sw-test-s4048-3 > vrouter-modify name S4048-3 bgp-max-paths 4 bgp-bestpath-as-path multipath-relax
CLI (network-admin@sw-test-s4048-3 > vrouter-interface-add vrouter-name S4048-3 l3-port 65 ip 10.3.14.3/24
CLI (network-admin@sw-test-s4048-3 > vrouter-bgp-add vrouter-name S4048-3 neighbor 10.3.14.14 remote-as 65134 bfd
CLI (network-admin@sw-test-s4048-3 > vrouter-loopback-interface-add vrouter-name S4048-3 ip 10.10.3.3 index 1
CLI (network-admin@sw-test-s4048-3 > vrouter-bgp-network-add vrouter-name S4048-3 network 10.10.3.3 netmask 255.255.255.255
CLI (network-admin@sw-test-s4048-3 > vlan-create id 34 scope local description L3-BGP-INTERCO-DC2 ports none
CLI (network-admin@sw-test-s4048-3 > vlan-port-add vlan-id 34 ports 128 tagged
CLI (network-admin@sw-test-s4048-3 > vrouter-interface-add vrouter-name S4048-3 vlan 34 ip 10.3.4.3/24
CLI (network-admin@sw-test-s4048-3 > vrouter-bgp-add vrouter-name S4048-3 neighbor 10.3.4.4 remote-as 65034 next-hop-self bfd
CLI (network-admin@sw-test-s4048-3 > Fabric-join switch-ip 10.20.1.1
```
Apply the following configuration to sw-test-s4048-4:

```
CLI (network-admin@sw-test-s4048-4 > switch-setup-modify in-band-ip 10.20.4.4/24
CLI (network-admin@sw-test-s4048-4 > Fabric-comm-vrouter-bgp-create name S4048-4
    bgp-as 65034 router-id 10.10.4.4 bgp-nic-ip 10.4.13.4/24 bgp-nic-l3-port 57
    neighbor 10.4.13.13 remote-as 65134 bfd Fabric-network 10.20.1.0/24
    in-band-nic-ip 10.20.4.104 in-band-nic-netmask 24
CLI (network-admin@sw-test-s4048-4 > vrouter-modify name S4048-4 bgp-max-paths 4
    bgp-bestpath-as-path multipath-relax
CLI (network-admin@sw-test-s4048-4 > vrouter-interface-add vrouter-name S4048-4
    l3-port 65 ip 10.4.14.4/24
CLI (network-admin@sw-test-s4048-4 > vrouter-bgp-add vrouter-name S4048-4
    neighbor 10.4.14.14 remote-as 65134 bfd
CLI (network-admin@sw-test-s4048-4 > vrouter-loopback-interface-add vrouter-name
    S4048-4 ip 10.10.4.4 index 1
CLI (network-admin@sw-test-s4048-4 > vrouter-bgp-network-add vrouter-name S4048-4
    network 10.10.4.4 netmask 255.255.255.255
CLI (network-admin@sw-test-s4048-4 > vlan-create id 34 scope local description
    L3-BGP-INTERCO-DC2 ports none
CLI (network-admin@sw-test-s4048-4 > vlan-port-add vlan-id 34 ports 128 tagged
CLI (network-admin@sw-test-s4048-4 > vrouter-interface-add vrouter-name S4048-4
    vlan 34 ip 10.3.4.4/24
CLI (network-admin@sw-test-s4048-4 > vrouter-bgp-add vrouter-name S4048-4
    neighbor 10.3.4.3 remote-as 65034 next-hop-self bfd
CLI (network-admin@sw-test-s4048-4 > Fabric-join switch-ip 10.20.1.1
```

Apply the following configuration to sw-test-s4048-5:

```
CLI (network-admin@sw-test-s4048-5 > switch-setup-modify in-band-ip 10.20.5.5/24
CLI (network-admin@sw-test-s4048-5 > port-config-modify port 57,65 speed 10g
CLI (network-admin@sw-test-s4048-5 > Fabric-comm-vrouter-bgp-create name S4048-5
    bgp-as 65056 router-id 10.10.5.5 bgp-nic-ip 10.5.15.5/24 bgp-nic-l3-port 57
    neighbor 10.5.15.15 remote-as 65156 bfd Fabric-network 10.20.1.0/24
    in-band-nic-ip 10.20.5.105 in-band-nic-netmask 24
CLI (network-admin@sw-test-s4048-5 > vrouter-modify name S4048-5 bgp-max-paths 4
    bgp-bestpath-as-path multipath-relax
CLI (network-admin@sw-test-s4048-5 > vrouter-interface-add vrouter-name S4048-5
    l3-port 65 ip 10.5.16.5/24
CLI (network-admin@sw-test-s4048-5 > vrouter-bgp-add vrouter-name S4048-5
    neighbor 10.5.16.16 remote-as 65156 bfd
CLI (network-admin@sw-test-s4048-5 > vrouter-loopback-interface-add vrouter-name
    S4048-5 ip 10.10.5.5 index 1
CLI (network-admin@sw-test-s4048-5 > vrouter-bgp-network-add vrouter-name S4048-5
    network 10.10.5.5 netmask 255.255.255.255
CLI (network-admin@sw-test-s4048-5 > vlan-create id 56 scope local description
    L3-BGP-INTERCO-DC3 ports none
CLI (network-admin@sw-test-s4048-5 > vlan-port-add vlan-id 56 ports 128 tagged
CLI (network-admin@sw-test-s4048-5 > vrouter-interface-add vrouter-name S4048-5
    vlan 56 ip 10.5.6.5/24
CLI (network-admin@sw-test-s4048-5 > vrouter-bgp-add vrouter-name S4048-5
    neighbor 10.5.6.6 remote-as 65056 next-hop-self bfd
CLI (network-admin@sw-test-s4048-5 > Fabric-join switch-ip 10.20.1.1
```
Apply the following configuration to sw-test-s4048-6:

```
CLI (network-admin@sw-test-s4048-6 > switch-setup-modify in-band-ip 10.20.6.6/24
CLI (network-admin@sw-test-s4048-6 > port-config-modify port 57,65 speed 10g
CLI (network-admin@sw-test-s4048-6 > Fabric-comm-vrouter-bgp-create name S4048-6
bgp-as 65056 router-id 10.10.6.6 bgp-nic-ip 10.6.15.6/24 bgp-nic-l3-port 57
neighbor 10.6.15.15 remote-as 65156 bfd Fabric-network 10.20.1.0/24
in-band-nic-ip 10.20.6.106 in-band-nic-netmask 24
CLI (network-admin@sw-test-s4048-6 > vrouter-modify name S4048-6 bgp-max-paths 4
bgp-bestpath-as-path multipath-relax
CLI (network-admin@sw-test-s4048-6 > vrouter-interface-add vrouter-name S4048-6
l3-port 65 ip 10.6.16.6/24
CLI (network-admin@sw-test-s4048-6 > vrouter-bgp-add vrouter-name S4048-6
neighbor 10.6.16.16 remote-as 65156 bfd
CLI (network-admin@sw-test-s4048-6 > vrouter-loopback-interface-add vrouter-name
S4048-6 ip 10.10.6.6 index 1
CLI (network-admin@sw-test-s4048-6 > vrouter-bgp-network-add vrouter-name
S4048-6 network 10.10.6.6 netmask 255.255.255.255
CLI (network-admin@sw-test-s4048-6 > vlan-create id 56 scope local description
L3-BGP-INTERCO-DC3 ports none
CLI (network-admin@sw-test-s4048-6 > vlan-port-add vlan-id 56 ports 128 tagged
CLI (network-admin@sw-test-s4048-6 > vrouter-loopback-interface-add vrouter-name S4048-6
vlan 56 ip 10.5.6.6/24
CLI (network-admin@sw-test-s4048-6 > vrouter-bgp-add vrouter-name S4048-6
neighbor 10.5.6.5 remote-as 65056 next-hop-self bfd
CLI (network-admin@sw-test-s4048-6 > Fabric-join switch-ip 10.20.1.1
```

**QUICK CHECK:** Verify that the Fabric contains all the 6 nodes as expected by running the *Fabric-node-show* command.

```
  sw-test-s4048-1  sw-test-s4048-2  sw-test-s4048-3
  sw-test-s4048-4  sw-test-s4048-5  sw-test-s4048-6

  10.10.10.2/32  10.10.20.2/32  10.10.30.2/32
  10.11.10.2/32  10.11.20.2/32  10.11.30.2/32

  public  public  public
  34   122   0
```

**Step 6 – Configure the clusters and the VRRP instances to be used for the HA VTEP**

We start with the cluster located in DC1 as in Figure 14 below:

**Figure 14:**
Zoom view on DC1; highlights the cluster and the VRRP instance between the cluster members.
a) Create the cluster that federates `sw-test-s4048-1` and `sw-test-s4048-2`. Please note that since all the nodes are part of the Fabric already, it is possible to run the commands from any of the switch consoles:

```
CLI (network-admin@sw-test-s4048-1 > cluster-create name S4048-12 cluster-node-1 sw-test-s4048-1 cluster-node-2 sw-test-s4048-2)
```

**NOTE:** There is no need to specify the cluster link due to the auto-trunk feature, which dynamically detects physical port adjacencies between two Fabric nodes: the cluster will form over this trunk port implicitly.

b) Create VLAN 4012 on both the nodes of the cluster; this will be used for the VTEP HA at later stage:

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 vlan-create scope cluster id 4012 description VTEP-HA-VLAN-S4048-12 ports none)
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-2 vlan-port-add vlan-id 4012 ports 128 tagged)
```

c) Assign IP addresses to the VLANs on the two cluster nodes, define the VRRP instance and assign a Virtual IP (VIP) to it:

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 vrouter-modify name S4048-1 hw-vrrp-id 12)
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-2 vrouter-modify name S4048-2 hw-vrrp-id 12)
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 vrouter-interface-add vrouter-name S4048-1 vlan 4012 ip 10.40.12.1/24)
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-2 vrouter-interface-add vrouter-name S4048-2 vlan 4012 ip 10.40.12.2/24)
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 vrouter-interface-add vrouter-name S4048-1 vlan 4012 ip 10.40.12.12/24 vrrp-id 12 vrrp-primary eth0.4012 vrrp-priority 250)
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-2 vrouter-interface-add vrouter-name S4048-2 vlan 4012 ip 10.40.12.12/24 vrrp-id 12 vrrp-primary eth0.4012 vrrp-priority 240)
```

d) Enable jumbo frames on ports facing the core of the network and the cluster link. This is mandatory for inter-data center traffic to be transported through the VXLAN tunnels, which introduce 50 extra bytes for VXLAN header. Without jumbo frame configuration on these ports, packets whose size is close to Ethernet Maximum Transmission Unit (MTU) of 1518 bytes would get dropped.

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-2 trunk-modify name auto-128 jumbo)
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-2 port-config-modify port 57,65 jumbo)
```

**NOTE:** In case jumbo frames are already configured at the edge, a common scenario in environments involving VM migration, it may be necessary to lower the edge MTU to accommodate for the 50-byte VXLAN header overhead.
Replicate similar configurations for the clusters in DC2 and DC3. In DC2, the cluster includes `sw-test-s4048-3` and `sw-test-s4048-4`:

```
CLI (network-admin@sw-test-s4048-1 > cluster-create name S4048-34 cluster-node-1 sw-test-s4048-3 cluster-node-2 sw-test-s4048-4
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3 vlan-create scope cluster id 4034 description VTEP-HA-VLAN-S4048-34 ports none
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3,sw-test-s4048-4 vlan-port-add vlan-id 4034 ports 128 tagged
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3 vrouter-modify name S4048-3 hw-vrrp-id 34
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3 vrouter-interface-add vrouter-name S4048-3 vlan 4034 ip 10.40.34.3/24
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-4 vrouter-modify name S4048-4 hw-vrrp-id 34
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-4 vrouter-interface-add vrouter-name S4048-4 vlan 4034 ip 10.40.34.4/24
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3 vrouter-interface-add vrouter-name S4048-3 vlan 4034 ip 10.40.34.34/24 vrrp-id 34 vrrp-primary eth0.4034 vrrp-priority 250
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-4 vrouter-interface-add vrouter-name S4048-4 vlan 4034 ip 10.40.34.34/24 vrrp-id 34 vrrp-primary eth0.4034 vrrp-priority 240
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3,sw-test-s4048-4 trunk-modify name auto-128 jumbo
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3,sw-test-s4048-4 port-config-modify port 57,65 jumbo
```

In DC3, configure the cluster with `sw-test-s4048-5` and `sw-test-s4048-6`:

```
CLI (network-admin@sw-test-s4048-1 > cluster-create name S4048-56 cluster-node-1 sw-test-s4048-5 cluster-node-2 sw-test-s4048-6
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5 vlan-create scope cluster id 4056 description VTEP-HA-VLAN-S4048-56 ports none
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5,sw-test-s4048-6 vlan-port-add vlan-id 4056 ports 128 tagged
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5 vrouter-modify name S4048-5 hw-vrrp-id 56
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5 vrouter-interface-add vrouter-name S4048-5 vlan 4056 ip 10.40.56.5/24
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-6 vrouter-modify name S4048-6 hw-vrrp-id 56
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-6 vrouter-interface-add vrouter-name S4048-6 vlan 4056 ip 10.40.56.6/24
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5 vrouter-interface-add vrouter-name S4048-5 vlan 4056 ip 10.40.56.56/24 vrrp-id 56 vrrp-primary eth0.4056 vrrp-priority 250
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-6 vrouter-interface-add vrouter-name S4048-6 vlan 4056 ip 10.40.56.56/24 vrrp-id 56 vrrp-primary eth0.4056 vrrp-priority 240
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5,sw-test-s4048-6 trunk-modify name auto-128 jumbo
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5,sw-test-s4048-6 port-config-modify port 57,65 jumbo
```
QUICK CHECK: At this stage, we can validate that the cluster nodes in DC1 can reach for instance the VRRP VIP of the clusters in DC2 and DC3 from their respective vRouters by running `switch sw-test-s4048-1 vrouter-ping vrouter-name S4048-1 host-ip 10.40.34.34` and `switch sw-test-s4048-1 vrouter-ping vrouter-name S4048-1 host-ip 10.40.56.56`. These target IP addresses function as tunnel end points for VXLAN traffic in DC2 and DC3.

**Step 7 – Configure the VXLAN Tunnel End-Points (VTEPs) and implicitly the tunnels between them**

a) Create the first HA VTEP object in DC1 by referencing as virtual-ip on both cluster members the same VIP previously created as a VRRP VIP. This will ensure that even if one of the cluster members fails, the second cluster member will continue to perform the local encapsulation / de-capsulation without affecting the traffic.

```plaintext
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 vtep-create name VTEP-S4048-1 vrouter-name S4048-1 ip 10.40.12.1 virtual-ip 10.40.12.12 location sw-test-s4048-1
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-2 vtep-create name VTEP-S4048-2 vrouter-name S4048-2 ip 10.40.12.2 virtual-ip 10.40.12.12 location sw-test-s4048-2
```

NOTE: The VTEP object can also reference a Physical IP (PIP) for both ip and virtual-ip properties, but in this case the tunnel would cease to operate in case the switch the PIP belongs to suffers from a failure. We strongly recommend to implement HA VTEP objects wherever it is possible to provide maximum availability in case of a node or link failure.

b) Create a similar VTEP object in DC2:

```plaintext
CLI (network-admin@sw-test-s4048-3 vtep-create name VTEP-S4048-3 vrouter-name S4048-3 ip 10.40.34.1 virtual-ip 10.40.34.34 location sw-test-s4048-3
CLI (network-admin@sw-test-s4048-4 vtep-create name VTEP-S4048-4 vrouter-name S4048-4 ip 10.40.34.2 virtual-ip 10.40.34.34 location sw-test-s4048-4
```

c) Create a similar VTEP object in DC3:

```plaintext
CLI (network-admin@sw-test-s4048-5 vtep-create name VTEP-S4048-5 vrouter-name S4048-5 ip 10.40.56.1 virtual-ip 10.40.56.56 location sw-test-s4048-5
CLI (network-admin@sw-test-s4048-6 vtep-create name VTEP-S4048-6 vrouter-name S4048-6 ip 10.40.56.2 virtual-ip 10.40.56.56 location sw-test-s4048-6
```

By creating the VTEP objects above, the Fabric will dynamically create a mesh of interconnecting tunnels among them without any additional operational effort.

QUICK CHECK: Verify the tunnels are configured and in active state by running the `tunnel-show` command.

- Quick Check: At this stage, we can validate that the cluster nodes in DC1 can reach for instance the VRRP VIP of the clusters in DC2 and DC3 from their respective vRouters by running `switch sw-test-s4048-1 vrouter-ping vrouter-name S4048-1 host-ip 10.40.34.34` and `switch sw-test-s4048-1 vrouter-ping vrouter-name S4048-1 host-ip 10.40.56.56`. These target IP addresses function as tunnel end points for VXLAN traffic in DC2 and DC3.

**Step 7 – Configure the VXLAN Tunnel End-Points (VTEPs) and implicitly the tunnels between them**

a) Create the first HA VTEP object in DC1 by referencing as virtual-ip on both cluster members the same VIP previously created as a VRRP VIP. This will ensure that even if one of the cluster members fails, the second cluster member will continue to perform the local encapsulation / de-capsulation without affecting the traffic.

```plaintext
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 vtep-create name VTEP-S4048-1 vrouter-name S4048-1 ip 10.40.12.1 virtual-ip 10.40.12.12 location sw-test-s4048-1
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-2 vtep-create name VTEP-S4048-2 vrouter-name S4048-2 ip 10.40.12.2 virtual-ip 10.40.12.12 location sw-test-s4048-2
```

NOTE: The VTEP object can also reference a Physical IP (PIP) for both ip and virtual-ip properties, but in this case the tunnel would cease to operate in case the switch the PIP belongs to suffers from a failure. We strongly recommend to implement HA VTEP objects wherever it is possible to provide maximum availability in case of a node or link failure.

b) Create a similar VTEP object in DC2:

```plaintext
CLI (network-admin@sw-test-s4048-3 vtep-create name VTEP-S4048-3 vrouter-name S4048-3 ip 10.40.34.1 virtual-ip 10.40.34.34 location sw-test-s4048-3
CLI (network-admin@sw-test-s4048-4 vtep-create name VTEP-S4048-4 vrouter-name S4048-4 ip 10.40.34.2 virtual-ip 10.40.34.34 location sw-test-s4048-4
```

c) Create a similar VTEP object in DC3:

```plaintext
CLI (network-admin@sw-test-s4048-5 vtep-create name VTEP-S4048-5 vrouter-name S4048-5 ip 10.40.56.1 virtual-ip 10.40.56.56 location sw-test-s4048-5
CLI (network-admin@sw-test-s4048-6 vtep-create name VTEP-S4048-6 vrouter-name S4048-6 ip 10.40.56.2 virtual-ip 10.40.56.56 location sw-test-s4048-6
```

By creating the VTEP objects above, the Fabric will dynamically create a mesh of interconnecting tunnels among them without any additional operational effort.

QUICK CHECK: Verify the tunnels are configured and in active state by running the `tunnel-show` command.
**Step 8 – Reserve a port for VXLAN loopback / packet recirculation**

On all the switches that perform the VTEP function, due to the internal ASIC architecture of the Open Networking platforms, it is necessary to dedicate a front-panel port for handling broadcast and unknown multicast (BUM) traffic that needs to be sent across the mesh of VXLAN tunnels. In fact, an internal physical loopback facility is required to recirculate this traffic in order to perform Head End Replication (HER) and send a unicast packet to each remote VTEP associated to the same Layer 2 extended segment or VNI.

The same loopback infrastructure can be leveraged for VXLAN routing, as explained further down in this document when discussing the anycast gateway functionality.

```
CLI (network-admin@sw-test-s4048-1 > switch * trunk-modify name VXLAN-loopback-trunk port 61 jumbo
```

**NOTE:** Certain hardware platforms have internal ports available for this task so this step is not necessary with those devices.

**Step 9 – Configure the vLAGs connecting to edge devices**

The vLAGs will ensure a resilient fast-converging redundant connection to the edge devices.

a) We start by configuring the vLAG in DC 1; as in Figure 15 below, we indicate port 3 on each of the cluster members:

![Figure 15: Zoom view of DC1; highlights the vLAG used as a redundant connection to the edge devices.](image)

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 vlag-create name S4048-12-TO-ESXI-1 port 3 peer-port 3 mode active-active lacp-mode active
```

**NOTE:** For this specific example where we configure a LACP-based vLAG, it is required that the host network interface cards (NICs) connecting to Netvisor powered switches are provisioned with LACP bonding; from the host perspective, the NICs are connected to a single link partner device, which Netvisor cluster and vLAG technology realizes out of two different physical switches. Implementing LACP is not a mandatory requirement: for example, to connect an VMware ESXi server, VMware provides some built-in redundancy mechanisms for the VM networks with teaming and failover policies.
For validating the underlay provisioning code and state, first verify the state of the auto-provisioned tunnels between the VTEPs on the leaf switches; the tunnels should be “active” and the state should be “OK” as below.

It is also useful to perform ping checks between the different VTEPs to validate the full reachability across the IP transport network between the different data centers. This is done by executing `vrouter-ping` commands from the vRouters defined in one DC targeting the VTEP VIPs located in different DCs, for example:

```
vrouter-ping vrouter-name s4048-1 host-ip 10.20.34.34
```

The third validation step confirms Equal Cost Multi-Path (ECMP) is implemented for every path between the VTEPs. This can be verified by checking the VTEP subnets have dual entries in both the routing table and the forwarding table of all BGP speakers. With the Fabric intelligence, this can be easily reviewed with one single-command per prefix:

```
vrouter-routes-show network 10.20.12.12
vrouter-routes-show network 10.20.34.34
vrouter-routes-show network 10.20.56.56
vrouter-fib-routes-show ip 10.20.12.12
vrouter-fib-routes-show ip 10.20.34.34
vrouter-fib-routes-show ip 10.20.56.56
```

In the output of the above command, we need to validate that all the vRouters in the Fabric have at least two ECMP paths available to reach the VTEP subnets. That will ensure redundancy against any link or node failure in the network.

---

**Step 10 – Validate the underlay configuration**

For validating the underlay provisioning code and state, first verify the state of the auto-provisioned tunnels between the VTEPs on the leaf switches; the tunnels should be “active” and the state should be “OK” as below.

It is also useful to perform ping checks between the different VTEPs to validate the full reachability across the IP transport network between the different data centers. This is done by executing `vrouter-ping` commands from the vRouters defined in one DC targeting the VTEP VIPs located in different DCs, for example:

```
vrouter-ping vrouter-name s4048-1 host-ip 10.20.34.34
```

The third validation step confirms Equal Cost Multi-Path (ECMP) is implemented for every path between the VTEPs. This can be verified by checking the VTEP subnets have dual entries in both the routing table and the forwarding table of all BGP speakers. With the Fabric intelligence, this can be easily reviewed with one single-command per prefix:

```
vrouter-routes-show network 10.20.12.12
vrouter-routes-show network 10.20.34.34
vrouter-routes-show network 10.20.56.56
vrouter-fib-routes-show ip 10.20.12.12
vrouter-fib-routes-show ip 10.20.34.34
vrouter-fib-routes-show ip 10.20.56.56
```

In the output of the above command, we need to validate that all the vRouters in the Fabric have at least two ECMP paths available to reach the VTEP subnets. That will ensure redundancy against any link or node failure in the network.
## Detailed “Overlay” Implementation Steps

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Create the Overlay VLANs and the associated VXLAN IDs (the unique identifiers of the bridge domains across the Fabric)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2</td>
<td>Set the ports facing the Edge devices as tagged ports of these new VLANs</td>
</tr>
<tr>
<td>Step 3</td>
<td>Add the VXLAN IDs to the required VTEPs</td>
</tr>
<tr>
<td>Step 4</td>
<td>Create the VRFs to provide the distributed routing and segmentation to previously created VLANs</td>
</tr>
<tr>
<td>Step 5</td>
<td>Create the Subnets assigned to each of the VLANs as the distributed anycast gateway and attach them to a VRF</td>
</tr>
<tr>
<td>Step 6</td>
<td>Configure the VRF gateways that define the northbound next-hops for traffic originating from previously created subnets</td>
</tr>
<tr>
<td>Step 7</td>
<td>Configure the vLAGs connecting to edge devices</td>
</tr>
<tr>
<td>Step 8</td>
<td>Validate the overlay layer</td>
</tr>
</tbody>
</table>

### Step 1 – Configure VLANs stretching across data centers and the associated VXLAN IDs

We create one VLAN that will stretch across all the three data centers and one VLAN that will be present only in two of the data centers.

a) Create VLAN 10 with cluster scope and associate VXLAN VNI 5030010 to it (the red lines in Figure 15):

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-3,
sw-test-s4048-5 vlan-create scope cluster id 10 VXLAN 5030010 ports none
```

b) Create VLAN 12 with cluster scope and associate VXLAN VNI 5030012 to it:

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-3,
sw-test-s4048-5 vlan-create scope cluster id 12 VXLAN 5030012 ports none
```

### Step 2 – Set the ports facing the edge devices as tagged ports of these new VLANs

As seen in the physical connectivity diagram (Figure 14), the edge device in our case is a VMware ESXi host connected to each Netvisor cluster in each data center, on port 3 of each switch:

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-2,
sw-test-s4048-3,sw-test-s4048-4,sw-test-s4048-5,sw-test-s4048-6 vlan-port-add
vlan-id 10 ports 3 tagged
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-2,
sw-test-s4048-3,sw-test-s4048-4,sw-test-s4048-5,sw-test-s4048-6 vlan-port-add
vlan-id 12 ports 3 tagged
```
Step 3 – Add the VXLAN IDs to the required VTEPs

- c) Since we have VLAN 10 present in all our three data centers and we want the VMs belonging to this VLAN to be able to reach each other as being part of the same bridge domain, we add its VXLAN VNI to all the VTEPs:

  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-1 VXLAN 5030010
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-2 VXLAN 5030010
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-3 VXLAN 5030010
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-4 VXLAN 5030010
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-5 VXLAN 5030010
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-6 VXLAN 5030010

- d) VLAN 12 is present only in DC2 and DC3 so we only need to add its associated VXLAN to the VTEPs connecting these data centers:

  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-3 VXLAN 5030012
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-4 VXLAN 5030012
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-5 VXLAN 5030012
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-6 VXLAN 5030012

- NOTE: Currently we need to add the VXLAN to each of the VTEPs even if we have VTEP HA implemented at the cluster level; in the next releases this will be automated having to run the command only once per cluster.

- d) VLAN 12 is present only in DC2 and DC3 so we only need to add its associated VXLAN to the VTEPs connecting these data centers:

  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-3 VXLAN 5030012
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-4 VXLAN 5030012
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-5 VXLAN 5030012
  CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-6 VXLAN 5030012

- QUICK CHECK: Verify the VTEP – VXLAN correspondence by running the vtep-VXLAN-show command.

It is important to notice that we only need to create the VTEPs once and then we just associate new VXLAN IDs to them as required.
Step 4 – Create the VRFs that are providing the distributed routing and segmentation to previously created VLANs

So far, the virtual machines attached to VLAN 10 are able to communicate to each other and similarly the ones attached to VLAN 12, regardless of which data center they are physically located in. Inter-VLAN communication between VLAN 10 and 12 stations is still not possible. Current step implements a distributed VRF function that will allow these virtual machines placed in different VLANs and subnets to mutually communicate, by providing the routing capability between these Layer 2 domains.

The `vrf-create` command could have local, cluster or Fabric scope significance; we will use the Fabric scope in this case to allow full mobility of the VMs across data centers; by being a Fabric scope object, the same anycast gateway will be present on all the Fabric nodes regardless of the physical location.

```
CLI (network-admin@sw-test-s4048-1 > vrf-create name VRF_1 scope Fabric vnet TME-DCI-global
```

**NOTE:** As seen in the command above, VRFs must be assigned to a certain virtual network or vNET (in this case the default Fabric vNET), allowing for an additional level of network segmentation. Optionally the VRF allows a user-defined MAC address to be used for the anycast gateway function by specifying the optional keyword `anycast-mac`.

The distributed anycast gateway provided by the VRF created above, is represented by the red hexagons on all the leaf nodes in Figure 17 below. Netvisor distributed VRF construct is very lean and scales with the Layer 3 virtualization capacity of the hardware, which is typically in the order of thousands of VRFs. To represent this capability, we display a second VRF (yellow hexagons) providing a different routing domain for the overlay traffic.

![Figure 17: Distributed anycast gateway with Layer 3 segmentation using distributed VRF.](image)

Step 5 – Create the Subnets assigned to each of the VLANs as the distributed anycast gateway and attach them to a VRF

```
CLI (network-admin@sw-test-s4048-1 > subnet-create VXLAN 5030010 network 10.0.10.0/24 vrf VRF_1 anycast-gw-ip 10.0.10.1
CLI (network-admin@sw-test-s4048-1 > subnet-create VXLAN 5030012 network 10.0.12.0/24 vrf VRF_1 anycast-gw-ip 10.0.12.1
```
A distributed subnet object confers Layer 3 attributes (subnet prefix and mask using the `network` parameter, anycast gateway IP address using the `anycast-gw-ip` parameter and VRF instance using the `vrf` parameter) to a Layer 2 domain, which is typically referenced using the `vlan` keyword. As VLANS are also associated to a VXLAN VNI, which is the unique identifier for that bridge domain across the Fabric, it is also possible to directly reference the VXLAN VNI value in the subnet creation command, by specifying the `VXLAN` keyword.

Associating a bridge domain (VLAN or VNI) with a distributed subnet and by inference with a distributed VRF, means that the hosts which are VXLAN part of the bridge domain are able to communicate to other hosts connected to subnets that belong to the same VRF. This communication is achieved:

a) Using the best forwarding path, i.e. without hair-pinning to a centralized router
b) And with very efficient forwarding control plane that supports ARP optimization and seamless mobility and provides higher hardware scale than EVPN based methods thanks to conversational forwarding approach.

It should be also clear to the reader that a distributed subnet can be provisioned with one single-command regardless the number of switches providing the anycast gateway function. This contrasts with EVPN and other box-by-box approaches where the anycast gateway configuration needs to be repeated on each switch. To make a simple example, the provisioning effort involving 20 switches and 200 subnets requires 200 atomic commands with the Fabric and 4000 commands with typical third-party management architectures, resulting in a higher operational and automation cost.

**Figure 18:** VMs assigned to VLANs/subnets belonging to different VRFs.

---

**Step 6 – Configure the VRF gateways, indicating the northbound next-hops for traffic originating from previously created subnets**

The VRF object created above provides east-to-west routing between the directly connected subnets. In order to allow these subnets to connect to services outside the VRF (including a different VRF within the same Fabric), we need to provision a gateway function that the VRF routing instances can use as a next-hop for the northbound traffic.

The VRF gateway can be carried out by any network element capable of routing and advertising the VRF subnets to the rest of the network; it can be a vRouter on a Netvisor powered switch or a third-party physical router or an external firewall depending on the specific design and requirements.
NOTE: Best practice design recommendations prescribe implementing local VRF gateways in each of the data centers. This allows the northbound traffic originating inside a certain data-center to be forwarded by local routers/firewalls without having to go thru another datacenter or cross a security boundary asymmetrically, hence being suboptimal or causing state synchronization issues with security devices.

Applying this best practice to our case, we will provision VRF_1 with four external gateways: two located in DC1 and another two located in DC3. In fact, since the VRF gateway setting can be customized “per-switch”, leaf switches in DC1 can point to two local gateways while the rest of the leaf switches will use the gateway pair located in DC3. The two gateway entries configured on the switches are both active, realizing ECMP load-balancing and offering a highly available next-hop in case of failure.

To simplify the provisioning effort, we will rely on switch group objects that can be defined in a Fabric

a) Create switch groups based on the physical location

    CLI (network-admin@sw-test-s4048-1 > switch-group-create name RACK-1
    CLI (network-admin@sw-test-s4048-1 > switch-group-member-add name RACK-1 member sw-test-s4048-1
    CLI (network-admin@sw-test-s4048-1 > switch-group-member-add name RACK-1 member sw-test-s4048-2
    CLI (network-admin@sw-test-s4048-1 > switch-group-create name RACK-2
    CLI (network-admin@sw-test-s4048-1 > switch-group-member-add name RACK-2 member sw-test-s4048-3
    CLI (network-admin@sw-test-s4048-1 > switch-group-member-add name RACK-2 member sw-test-s4048-4
    CLI (network-admin@sw-test-s4048-1 > switch-group-create name RACK-3
    CLI (network-admin@sw-test-s4048-1 > switch-group-member-add name RACK-3 member sw-test-s4048-4
    CLI (network-admin@sw-test-s4048-1 > switch-group-member-add name RACK-3 member sw-test-s4048-5

b) Create the VLANs used to connect the border leaves in each of the DCs with the local VRF gateways. Use VLAN ID 101 and 102 in DC1 and VLANs 103 and 104 in DC3. These VLANs are assigned to the physical ports facing the gateways themselves, the cluster ports and the VXLAN loopback ports on all the switches in the Fabric.
CLI (network-admin@sw-test-s4048-1 > switch * vlan-create id 101 scope local VXLAN 5000101 ports none
CLI (network-admin@sw-test-s4048-1 > switch * vlan-create id 102 scope local VXLAN 5000102 ports none
CLI (network-admin@sw-test-s4048-1 > switch * vlan-create id 103 scope local VXLAN 5000103 ports none
CLI (network-admin@sw-test-s4048-1 > switch * vlan-create id 104 scope local VXLAN 5000104 ports none

CLI (network-admin@sw-test-s4048-1 > switch RACK-1 vlan-port-add vlan-id 101 ports 11,128
CLI (network-admin@sw-test-s4048-1 > switch RACK-1 vlan-port-add vlan-id 102 ports 12,128
CLI (network-admin@sw-test-s4048-1 > switch RACK-3 vlan-port-add vlan-id 103 ports 13,128
CLI (network-admin@sw-test-s4048-1 > switch RACK-3 vlan-port-add vlan-id 104 ports 14,128
CLI (network-admin@sw-test-s4048-1 > switch * port-vlan-add port 61 vlans 101,102,103,104

CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-1 VXLAN 5000101
CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-1 VXLAN 5000102
CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-2 VXLAN 5000101
CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-2 VXLAN 5000102
CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-5 VXLAN 5000103
CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-5 VXLAN 5000104
CLI (network-admin@sw-test-s4048-1 > vtep-VXLAN-add name VTEP-S4048-6 VXLAN 5000104

NOTE: Switch group aliases can be used to target a Netvisor command to a set of switches by prepending the command with the `switch` keyword followed by the group name. The star sign `*` defines a built-in switch group comprising all switches that are part of the Fabric.

CLI (network-admin@sw-test-s4048-1 > switch RACK-1 vlan-port-add vlan-id 101 ports 11,128
CLI (network-admin@sw-test-s4048-1 > switch RACK-1 vlan-port-add vlan-id 102 ports 12,128
CLI (network-admin@sw-test-s4048-1 > switch RACK-3 vlan-port-add vlan-id 103 ports 13,128
CLI (network-admin@sw-test-s4048-1 > switch RACK-3 vlan-port-add vlan-id 104 ports 14,128
CLI (network-admin@sw-test-s4048-1 > switch * port-vlan-add port 61 vlans 101,102,103,104

CLI (network-admin@sw-test-s4048-1 > vlag-create name GW-1 port 11 peer-port 11 mode active-active
CLI (network-admin@sw-test-s4048-1 > vlag-create name GW-2 port 12 peer-port 12 mode active-active
CLI (network-admin@sw-test-s4048-1 > vlag-create name GW-3 port 13 peer-port 13 mode active-active
CLI (network-admin@sw-test-s4048-1 > vlag-create name GW-4 port 14 peer-port 14 mode active-active

Pluribus Networks | Data Center Interconnect Validated Design Guide | 30
NOTE: On the external gateways, south-bound VRF subnets that are required to be reachable from the external network must have a corresponding static route configured: this is done by defining the transit subnet anycast gateway as next-hop for the aforementioned static routes. For example, the external gateway router in DC1 can use 10.0.101.1 on VLAN 101 as next-hop. In a future Netvisor release the southbound reachability of the VRF will be automated using vRouter advertisements.

In order to maintain the Layer 3 isolation achieved thru the VRF construct, we recommend to provision a different pair of external gateways for each distributed VRF; this could also mean the same physical device can act as a shared gateway for different VRFs, but offering a distinct Layer 3 interface to each of them.

NOTE: On the external gateways, south-bound VRF subnets that are required to be reachable from the external network must have a corresponding static route configured: this is done by defining the transit subnet anycast gateway as next-hop for the aforementioned static routes. For example, the external gateway router in DC1 can use 10.0.101.1 on VLAN 101 as next-hop. In a future Netvisor release the southbound reachability of the VRF will be automated using vRouter advertisements.

Step 8 – Validate the overlay layer

The first validation step is to confirm the reachability of the anycast gateway from hosts located in any of the VRF subnets, in any rack, in any data center. For instance, in Figure 17, the VM assigned to VLAN 10 with IP address of 10.0.10.101, is supposed to reach the anycast gateway of its corresponding subnet, 10.0.10.1. The same validation can be done for different subnets and physical locations of the VMs.

The second check is validating the Layer 2 extensions across data centers. For instance, VMs placed in the same VLAN / VNI in different locations should be able to reach each other through the VXLAN tunnels configured in the underlay.

A third checkpoint concerns seamless VM mobility enabled by the distributed anycast gateway design. For instance, start a continuous ping from any of the VMs targeting a device located outside of the data center, in the external networks. The traffic is routed by the VRF on the first Netvisor powered node and then forwarded to the VRF gateway configured on that particular node. While the continuous ping is running, migrate the VM to another rack in a different data center. Since the anycast gateway is a distributed Fabric object, the live mobility event is seamlessly handled with minimal traffic loss and latency.
Advanced DCI features

In addition to the capabilities covered in the previous sections of this document, the Pluribus DCI solution offers native support for additional advanced features such as, multi-tenancy and transparent pseudowires.

Multi-tenancy is based on Netvisor Private Virtual Networks (vNET) network virtualization technology which achieves the separation of different network object types, such as vRouters and VLANs, in isolated virtual network instances, or containers, that operate on dedicated pools of physical resources, such as bandwidth, physical interfaces or hardware queues. Multi-tenancy is a broader topic that is covered in a different design guide.

The transparent pseudowire feature is based on Pluribus Virtual Link Extension (vLE) technology and consists in the ability of emulating virtual wires across DCI or multi-site connections. Devices physically connected on each side of the vLE connection, are able to form a transparent Layer 2 adjacency as if they were directly connected.

The vLE technology can also be used to build redundant point-to-point links that can be logically bundled in a LAG by configuring on the client devices a protocol like IEEE 802.3ad LACP, which is transparently transported across the vLE infrastructure.

Transparent Pseudowire with Virtual Link Extension

The goal for this section is to create multiple pseudowires between DC1, DC2 and DC3, so that client devices represented in red and purple in the image below, located in these geographically dispersed data centers, see each other as they were directly connected. There are two vLE use cases represented in Figure 20 below:

- A generic point-to-point service between two switches (purple boxes) to exchange layer-2 traffic (like MPLS or QinQ) over an IP network
- A direct link emulation between two firewall devices (red boxes), in order to stretch a firewall cluster over different geographical locations

A third use case explains how to deliver redundant point-to-point services and will be discussed in the last step of this procedure.

Figure 20: Two vLEs emulating a point-to-point direct connection between client devices.
Virtual Link Extension Implementation Steps

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Define tunnels between the switches where the access ports of the vLE are located</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2</td>
<td>Create a transparent bridge domain with associated VLAN/VNI for the client port pair that attaches to the pseudowire</td>
</tr>
<tr>
<td>Step 3</td>
<td>Create the vLE object with port link tracking</td>
</tr>
<tr>
<td>Step 4</td>
<td>Attach the transparent VXLAN VNI to tunnels</td>
</tr>
<tr>
<td>Step 5</td>
<td>Validate the vLE functionality</td>
</tr>
<tr>
<td>Step 6</td>
<td>Repeat previous steps to create redundant vLE groups</td>
</tr>
</tbody>
</table>

**Step 1 – Define tunnels between the switches where access ports of the vLE are located**

The vLE transport relies on Physical IP addresses for VXLAN tunnel termination and not on VTEP HA; this means that the vLE is not by itself a redundant object. However, end-to-end high availability can be provided by defining multiple vLEs using different physical switches at the end locations, as explained in the last step of this section.

Create two unidirectional tunnels between `sw-test-s4048-1` and `sw-test-s4048-3`:

```bash
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 tunnel-create name S4048-1-TO-S4048-3 scope local vrouter-name S4048-1 local-ip 10.40.12.1 remote-ip 10.40.34.3

CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3 tunnel-create name S4048-3-TO-S4048-1 scope local vrouter-name S4048-3 local-ip 10.40.34.3 remote-ip 10.40.12.1
```

**NOTE:** The Fabric does not participate in any spanning tree protocol or control plane originating from a vLE access port. There is no logical relationship between vLEs configured in the same cluster and no loop prevention mechanism operating on vLE objects, so the client devices connected on both ends must implement their own mechanism to secure the client devices against any loop.

Similarly, create tunnels between `sw-test-s4048-2` in DC1 and `sw-test-s4048-6` in DC3:

```bash
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 tunnel-create name S4048-1-TO-S4048-6 scope local vrouter-name S4048-1 local-ip 10.40.12.2 remote-ip 10.40.56.6

CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-6 tunnel-create name S4048-6-TO-S4048-1 scope local vrouter-name S4048-6 local-ip 10.40.56.6 remote-ip 10.40.12.2
```

Each tunnel between two physical switches, as the ones defined above, provides the IP transport to multiple vLEs and not just one vLE.

**Step 2 – Create a transparent bridge domain with associated VLAN/VNI for the client port pair that attaches to the pseudowire**

For the transparent pseudowire between port 47 on `sw-test-s4048-1` in DC1 and port 47 on `sw-test-s4048-3` in DC2, create VLAN 4001 that will transparently transport the packets ingressing the client access ports over the IP transport core infrastructure. Assign also a VXLAN VNI to it.
NOTE: The VLAN ID used to carry transparent bridged traffic over vLE cannot be re-used for regular Layer 2 bridging with other co-located hosts. This VLAN must be dedicated for this vLE service. The VLAN IDs associated to a vLE on both ends can be different. The VLAN ID used for vLE is only locally significant.

The vLE interconnecting access ports 47 on sw-test-s4048-1 in DC1 and sw-test-s4048-6 in DC3 use a different VLAN ID and VNI:

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-3
vlan-create id 4001 VXLAN 5004001 VXLAN-mode transparent scope local description VLE-S4048-1-S4048-3 ports 47
```

```
CLI (network-admin@sw-test-s4048-2,sw-test-s4048-6
vlan-create id 4002 VXLAN 5004002 VXLAN-mode transparent scope local description VLE-S4048-2-S4048-6 ports 47
```

**Step 3 – Create the vLE object with port link tracking**

Each vLE is created as a single Fabric object interconnecting the access port pair. When the tracking keyword is used, the Fabric control plane keeps track of the link state at each vLE access port and synchronizes the administrative port state with the operational link state of the remote port. This Fabric service works in conjunction with a high availability design implemented on the client network to protect client traffic from ingressing an access port and be black-holed because the remote port is down.

```
CLI (network-admin@sw-test-s4048-1 > vle-create name FIREWALL-CLUSTER node-1 sw-test-s4048-1 node-1-port 47 node-2 sw-test-s4048-3 node-2-port 47
CLI (network-admin@sw-test-s4048-1 > vle-create name SWITCH-P2P-1 node-1 sw-test-s4048-2 node-1-port 47 node-2 sw-test-s4048-6 node-2-port 47
```

In future Netvisor releases the vLE object will automatically self-provision its VLAN and VNI sub-objects and remove the need of configuring Step 2 and Step 4.

**Step 4 – Attach the transparent VXLAN VNI to tunnels**

For the vLE between sw-test-s4048-1 and sw-test-s4048-3, attach VXLAN VNI 5004001 to the corresponding tunnels:

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 tunnel-VXLAN-add name S4048-1-TO-S4048-3 VXLAN 5004001
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-3 tunnel-VXLAN-add name S4048-3-TO-S4048-1 VXLAN 5004001
```

Replicate a similar configuration for the other vLE in between sw-test-s4048-2 and sw-test-s4048-6:

```
CLI (network-admin@sw-test-s4048-2 > switch sw-test-s4048-2 tunnel-VXLAN-add name S4048-2-TO-S4048-3 VXLAN 5004002
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-6 tunnel-VXLAN-add name S4048-6-TO-S4048-2 VXLAN 5004002
```

**Step 5 – Validate the vLE functionality**

Using Figure 19 as reference, connect to the console of the third-party client devices and check the neighbor device is discovered by the LLDP protocol or any other similar protocol used for advertising the device identity and capabilities.
On each of the devices we should observe that the other third-party device is directly connected, even if in reality there are multiple Layer 3 hops in between.

**NOTE:** The vLE allows to easily create overlay topologies ad-hoc, connecting devices across multiple geographical locations, with practically no intervention or change required in the underlay transport infrastructure in the middle.

**Step 6 – Repeat previous steps to create redundant vLE groups**

This last step demonstrates how it is possible to use multiple vLE objects in order to provide resilient point-to-point services with switch and link protection that work in conjunction with any high availability mechanism, active-active or active-backup, implemented on the client network. A classic example is an LACP LAG created between two links that originate on the client devices and are extended using two vLE objects implemented on different cluster members, providing node and link diversity. This is represented in Figure 21 where a redundant link/vLE is added to the client switches (purple boxes), engaging switches sw-test-s4048-1 and sw-test-s4048-5, and an LACP group is created with these two links, by leveraging the unique characteristics of vLE, like the protocol transparency and link state tracking.

**Figure 21:**
Two vLEs emulating a point-to-point LAG connection between client devices.

The following commands are similar to steps 1—4 above.

Create tunnels between *sw-test-s4048-1* in DC1 and *sw-test-s4048-5* in DC3:

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 tunnel-create name S4048-1-TO-S4048-5 scope local vrouter-name S4048-1 local-ip 10.40.12.1 remote-ip 10.40.56.5
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5 tunnel-create name S4048-5-TO-S4048-1 scope local vrouter-name S4048-5 local-ip 10.40.56.5 remote-ip 10.40.12.1
```

Create the transparent bridge domain with associated VLAN ID and VNI.

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1,sw-test-s4048-5 vlan-create id 4003 VXLAN 5004003 VXLAN-mode transparent scope local description VLE-S4048-1-S4048-5 ports 48
```
Create the vLE object

```
CLI (network-admin@sw-test-s4048-1 > vle-create name SWITCH-P2P-2 node-1 sw-test-s4048-1 node-1-port 48 node-2 sw-test-s4048-5 node-2-port 48
```

Attach VXLAN VNI 5004003 to the corresponding tunnels:

```
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-1 tunnel-VXLAN-add name S4048-1-TO-S4048-5 VXLAN 5004003
CLI (network-admin@sw-test-s4048-1 > switch sw-test-s4048-5 tunnel-VXLAN-add name S4048-5-TO-S4048-1 VXLAN 5004003
```

Configure and validate LLDP and LACP adjacencies by connecting to the client switch.

**Conclusion**

The Pluribus Data Center Interconnect solution offers a scalable, cost-effective and robust architecture to address need for reliable and efficient data center interconnection capabilities. Built with the scale-out capabilities of the Adaptive Cloud Fabric, the Pluribus approach to DCI addresses important requirements that a multi-site data center solution needs including transparency, redundancy, growth capability, fast convergence in case of a failure event and support for virtualized multi-tenancy. Additionally, it provides a set of unique capabilities not found in traditional architectures:

- Single management Fabric scales geographically for multi-site networks
- Fully transparent point-to-point services provided as pseudowire emulation circuits
- Intelligent and scalable Layer 3 VPN and Layer 2 VPN services configurable as single Fabric-wide objects
- Built-in flow analytics for point-to-point and point-to-multipoint services with the Insight Analytics platform

The Pluribus Adaptive Cloud Fabric is powered by the deployment-proven Netvisor ONE OS, which is an open, secure and programable next-generation Network OS that is purpose-built to optimize the power and performance of bare metal Open Networking hardware. Deployment-proven in production in mission-critical enterprise and carrier networks, Netvisor ONE meets the most stringent performance requirements and delivers the maximum levels of reliability and flexibility at scale without compromise.

Netvisor ONE runs on many Open Compute Project (OCP), and Open Network Install Environment (ONIE) hardware compliant switches, including devices from D-Link Systems, Dell EMC, Edge-Core, and the Pluribus Freedom series network switches. This flexibility allows organizations the choice of open networking hardware to build scale-out networks with 10, 25, 40 or 100 Gigabit Ethernet interfaces. This allows an entire data center to be built with only a few physical switch models to improve operational consistency, lower costs, and simplifying sparing strategies.