The Fast-Track Guide to VXLAN BGP EVPN Fabrics

Implement Today's Multi-Tenant Software-Defined Networks

Rene Cardona

Apress®

The Fast-Track Guide to VXLAN BGP EVPN Fabrics

Implement Today's Multi-Tenant Software-Defined Networks

Rene Cardona

Apress®

The Fast-Track Guide to VXLAN BGP EVPN Fabrics

Rene Cardona
California, CA, USA

ISBN-13 (pbk): 978-1-4842-6929-9 ISBN-13 (electronic): 978-1-4842-6930-5
https://doi.org/10.1007/978-1-4842-6930-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aditee Mirashi
Development Editor: Matthew Moodie
Coordinating Editor: Aditee Mirashi

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6929-9. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

I dedicate this book to my mom, Myrna, and my dad, Rene. Here I am. Your care and love got me here. Thank you for giving me the tools and support to make it happen.

Mr. Jose Fernandez Laya, you believed in me and told me I could reach greatness, and so I did and will continue to do so. I miss enjoying a little glass of wine and talking to you.
Rest in peace my friend. This is for you.

Table of Contents

About the Author

Rene Cardona is a dual CCIE #62050 in data center and service provider disciplines. He is a network solutions architect with broad experience in core data center and network security infrastructure architecture design, consulting, and implementation. He has performed many security and data center architecture refreshes for major US corporations in logistics, retail, health care, and education. He has provided expert insight during data center migrations (campus to VXLAN, ACI, and UCS environments) and VXLAN multipod and multisite architectures.

Rene is more than proficient in hyperconverged data center environments, VXLAN BGP EVPN, SDN, SD-WAN, Multi-Datacenter High Availability (MDHA), and network security (firewalls, network admission control, and network security architectures). He has performed numerous VXLAN, BGP, EVPN, and ACI deployments for major corporations in various corporate sectors—all with great success! He is also an online IT trainer and has authored video training courses such as “Mastering Palo Alto Networks” and “Learning the Cisco Application Centric Infrastructure (ACI)”—all with highly rated student feedback.

About the Technical Reviewer

Hameed Alshamaa is a senior network engineer. He holds a master's degree in computer engineering and bachelor's degree in mechanical engineering. He is dual-CCIE in enterprise infrastructure and security. He is certified in various industry-leading areas, including PCSNE, NSE 4 and NSE 6, VMware VPC-DVC6.5, and VMware-NXT in data center and network virtualization. Hameed has more than 20 years of experience in the network industry, working mostly with data center networks, security design, solution enhancements, advanced threat solution design, endpoint security, and futures.

Acknowledgments

This book would not have happened if Aditee Mirashi did not give me the chance to shine. Thank you, Aditee, for believing in me and for your assistance and support during this journey.

Matthew Moodie, thanks for your help on this project. Your assistance was extremely important.

Hameed Alshamaa, I cannot thank you enough for your support during the project. Your input was very important for this work to be a total success. Thanks, buddy. I know you are a doer, do not quit.

To everyone that has always encouraged me to reach out for more, this is proof that I'm going forward, and I will not quit until I reach all my life goals. A key ingredient for success is passion. If you do not have the passion for what you do, you will not be successful. Do not quit your dreams. Life is only once.

Introduction

You want to learn, deploy, and support VXLAN BGP EVPN, right? This book provides the information to help you accomplish this. I have written this book to save you time and effort and avoid researching vendors' technical documentation to get all the details. You go from zero to hero by learning the foundational concepts of VXLAN and why you should adopt it. We then move forward into the underlay and what role it covers in a VXLAN BGP EVPN fabric. Then we move to the overlay aspects.

The cool thing about this book is that you learn the technology in layman's terms. The information has been redacted in a way that is easy to comprehend and fast to apply. I include all relevant configuration commands to make your engineering life easier. I wrote this book with you in mind because I know what it is like to have limited time to meet a deadline and deliver a solution. You do not have time to scroll through pages and pages of knowledge articles to learn VXLAN.

This book is all you need to become proficient in supporting VXLAN BGP EVPN. In every chapter that I configured and tested my configurations, I strongly encourage you to do the same. A great way to simulate a VXLAN environment is through a simulator platform called EVE-NG. With EVE-NG, you can lab most recognized vendor-centric VXLAN solutions. Cisco Nexus, Arista, and Juniper can be simulated with this application.

I strongly recommend that you read and study the book *and* practice the labs as we go through the chapters. That way, you solidify the concepts, and it is easier to understand each chapter as you go along. There is nothing else out there better than practice to make you proficient or an expert. Practice makes perfect!

I hope you enjoy the book.

CHAPTER 1

Introduction to Spine-and-Leaf Topologies

The traditional campus design topology has reached the limits of today's network architecture scalability and performance requirements.

A **spine-and-leaf VXLAN BGP EVPN fabric** provides a robust backbone network that handles the demand for high-density, multigigabit traffic requirements. It allows real-time access to high-performance databases, streaming media content in 4K resolution, and terabyte file transfers without latency or lack of speed when accessed by thousands of concurrent users.

The spine-and-leaf architecture (a very unusual name for a technology concept in the IT industry) solves the scalability and performance demand limitations on campus designs by applying a simple architectural approach. VXLAN BGP EVPN encapsulates layer 2 into layer 3 frames, which are transported using the L2VPN EVPN address family identifier (AFI) in BGP. Let's discuss where spine-and-leaf architecture fits in today's network design concepts and become familiar with each component.

R. Cardona, *The Fast-Track Guide to VXLAN BGP EVPN Fabrics*,
https://doi.org/10.1007/978-1-4842-6930-5_1

Note A spine-and-leaf architecture can run different applications, not just VXLAN. Another spine-and-leaf architecture use case is Cisco's application-centric infrastructure, which uses COOP (Council of Oracle Protocol) instead of VXLAN to perform its endpoint (IP) mappings and announce its location.

Spine-and-Leaf Architecture

Symmetric architecture is predictable. You can visualize a traffic pattern in a spine-and-leaf architecture. Connectivity is as simple as **leaf–spine–leaf**.

Traffic flow begins on the **source leaf**, which forwards it to the **spine**. Then, the spine forwards it to the **destination leaf**. Every source endpoint (any device, server, workstation, etc.) is only two hops away from its destination (see Figure 1-1).

- **First hop**: source leaf to spine
- **Second hop**: spine to destination leaf

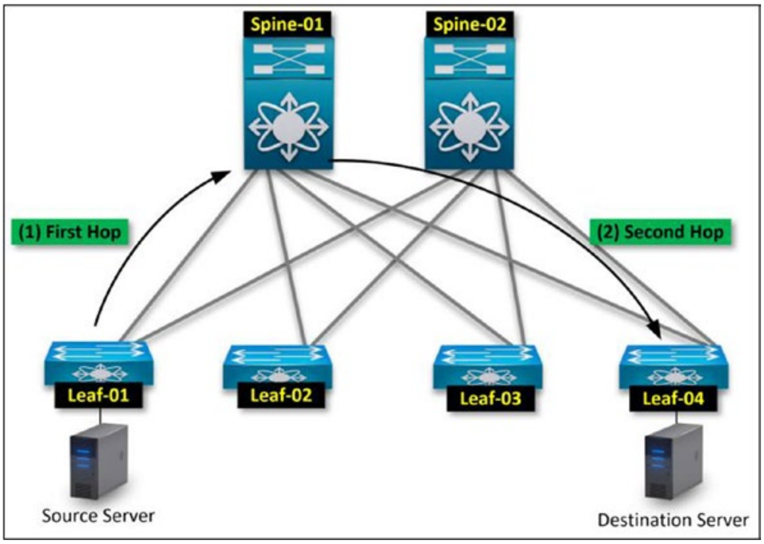


***Figure 1-1.** This spine-and-leaf topology shows the hop by hop traffic path between the source server on Leaf-01 and the destination server on Leaf-04. There's only a two-hop path to reach its destination*

Spine-and-Leaf Layers

There are two layers in a spine-and-leaf topology.

The **spine layer** is where the leafs connect. The spines reflect all routing information to their clients (in this case, to the leafs). The spine layer reflects BGP EVPN by designating the spines as route reflectors. (Later on, I discuss route reflectors and the fabric underlay.) In the spine layer, you also designate them as the rendezvous points for underlay multicast traffic (covered later in this book). Consider the spine layer as a distribution or aggregation layer in a three-tier design, but doing much more than just layer 2 aggregation.

The **leaf layer** provides all the endpoints access to the fabric and makes network routing decisions. All leafs are layer 3 cores. In a three-tier design, the core layer performs all the routing decisions. In three-tier designs, the core is usually a single active hardware, and redundant cores are set as standby nodes with FHRP (first-hop redundancy protocol). This is not the case with leafs in VXLAN BGP EVPN.

A very powerful feature in VXLAN BGP EVPN is the *anycast gateway* feature, which allows a leaf layer to act as a giant active core switch. Each leaf can route traffic to its destination. You aren't limited to a single active layer 3 core. In VXLAN fabrics, each leaf is an active core that provides notable performance and scalability functionalities in today's data center network requirements.

Redundancy in Spine-and-Leaf Topologies

As with all production environments, it is mandatory to have redundancy in place. In a spine-and-leaf architecture, this is no different. All leafs are connected to all spines. At least one link from a leaf goes to a spine. A fabric should have a minimum of two spines to comply with redundancy requirements. Figure 1-2 illustrates an example failover scenario in a four-leaf/two-spine topology.

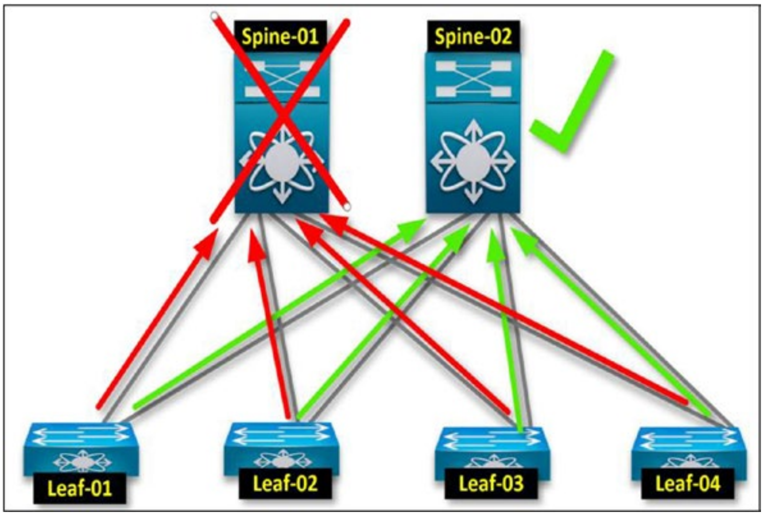


Figure 1-2. *If Spine-01 goes offline, there is no impact from a production standpoint since all leafs are also reachable via Spine-02*

Leaf Redundancy

Let's discuss redundancy on the leaf layer. Since a leaf connects all the network endpoints, access switches, servers, and so forth, the redundancy aspect is slightly different from the spine layer.

Let's briefly talk about vPC on the Cisco Nexus platform. vPC provides the required leaf redundancy by combining two independent leafs into a vPC domain. Let's assume you have a server with dual NIC connectivity. Since the leaf layer is where you connect all your end devices, access switches, and servers to the fabric, redundancy is provided to the end device, in this case, the server. It is achieved with an end-host vPC configuration. Redundancy is provided to the server by the leaf layer. If you lose Leaf-01, Leaf-02 should continue providing connectivity to the server (see Figure 1-3).

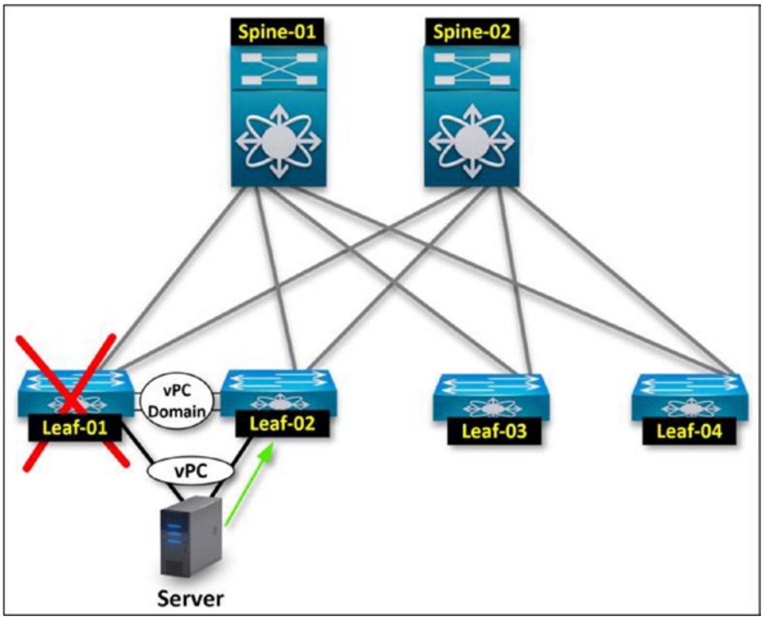


Figure 1-3. *The server is dual-homed to both Leaf-01 and Leaf-02, in case Leaf-01 goes offline. The server fails over and still communicates over Leaf-02. This is the physical redundancy aspect on the leaf layer*

Underlay Networking

When I first started learning VXLAN, it took me a while to get my head around *underlay* and *overlay*. Explaining to my colleagues and customers was also a challenge. Thankfully, I've learned the perfect analogy to explain it.

Let's look at the VXLAN underlay and overlay and liken them to a rollercoaster. A rollercoaster has rails, motors, and brakes, which are its *underlay*. A rollercoaster's underlay carries the cars and its riders, which are the *overlay*.

Now let's compare it to VXLAN. In the VXLAN underlay, the physical links between the leafs and the spines (the rails) are connected to allow client traffic (the rollercoaster cars and riders) to move on the fabric and reach its destination. A very important aspect of the underlay is leveraging equal-cost multipath (ECMP) routing on the links between leafs and spines. ECMP leverages active leaf-to-spine links for traffic flow. It's somewhat like link aggregation in L2, but you are doing it from a L3 standpoint (see Figure 1-4). Chapter 2 discusses ECMP in depth.

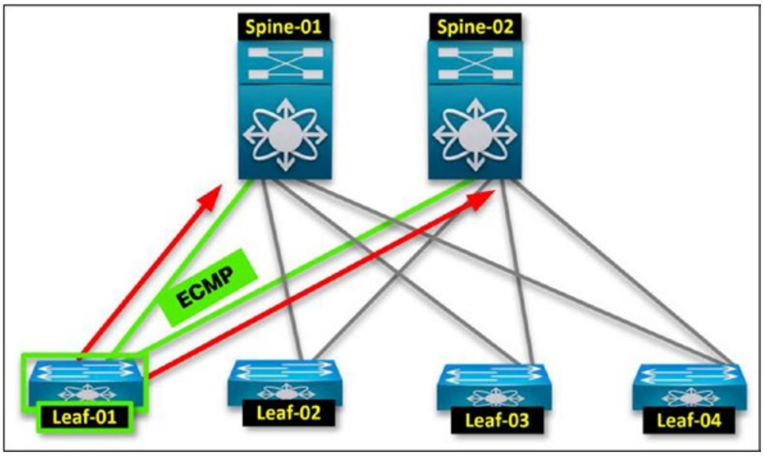


Figure 1-4. *Leaf-01 has two active paths: one path to Spine-01 and another to Spine-02. Since both paths have equal cost, the leaf maximizes traffic speed and performance by using both paths equally*

Overlay Networking

An overlay (the rollercoaster cars and riders) is where the VXLAN advantage over traditional networking occurs. VXLAN brings a deal-breaker characteristic called *multitenancy*. With multitenancy, you can run different client networks using the same fabric. A *tenant* refers to a virtual network inside the same VXLAN fabric, bringing one of the main advantages in software-defined networks (SDN).

In the rollercoaster analogy, the tenants are the rollercoaster cars. Each car (tenant) carries a group of riders (let's relate the riders to VLANs), and only the riders (VLANs) inside the same car (tenant) can talk among each other. A rider (VLAN) cannot talk to any rider in a different car (tenant), even if they are riding on the same rollercoaster (VXLAN fabric). Yes, there are ways to make the communication happen by configuring route leaking, but let's focus on multitenancy's main purpose.

To make everything more interesting, let's add a bit of icing to the cake. I mentioned ECMP and how to leverage the links on each leaf going to different spines. ECMP provides the rollercoaster car (tenant) the advantage of using two rails (the links between leafs and spines) simultaneously. The car (tenant) can run (flow) on top of two rails (links) at the same time for more speed. If one rail (link) breaks, the car still has another rail available to continue its ride (path) (see Figure 1-5).

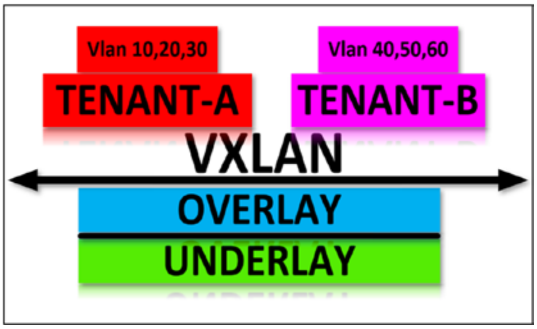


Figure 1-5. *The underlay carries the overlay. The overlay allows VXLAN communication on the fabric for the virtual network tenant-a or tenant-b*

Spine-and-leaf Fabric Traffic Flow

Now that you have a clearer picture of the critical VXLAN fabric components, I'll explain how the fabric operates and what VXLAN needs to communicate within the infrastructure.

Broadcast Unknown Unicast and Multicast (BUM Traffic)

Since L2 frames are encapsulated into L3 in VXLAN, you effectively suppress broadcast at the fabric level. Broadcast is how a network learns about its connected devices, but how does VXLAN learn since broadcast is effectively suppressed? With multicast! **BUM traffic** is the three types of messages to establish communication on a network: **broadcast**, **unicast**, and **multicast**. Multicast is an alternative to broadcast that can use L3 to propagate the information.

Underlay Multicast

Now that you know that multicast replaces broadcast, the multicast architecture must run in the underlay. How is it configured? You designate a multicast prefix to map multicast groups to the VXLAN identifier (VNI). There is one multicast group per VNI. The multicast messages are sent to a rendezvous point, which you usually designate to the spines. (Don't worry. I discuss this later.)

Underlay Routing

The underlay routing in VXLAN is crucial to building a fabric's foundation. A dynamic routing protocol such as OSPF or IS-IS is designated as the Interior Gateway Protocol (IGP). It establishes neighbor peering for all the leaf-to-spine physical uplinks. Once it is active, you bring control-plane BGP EVPN to the mix by running the BGP protocol on top of OSPF or IS-IS. You create a loopback address on each switch (spine-and-leaf) and advertise in OSPF to use that address as the BGP peering address. How do you do this? Pay close attention.

The first steps are to bring up OSPF or IS-IS between the leaf and spine links, configure a loopback interface per device, and advertise it to the IGP (OSPF) (see Figure 1-6).

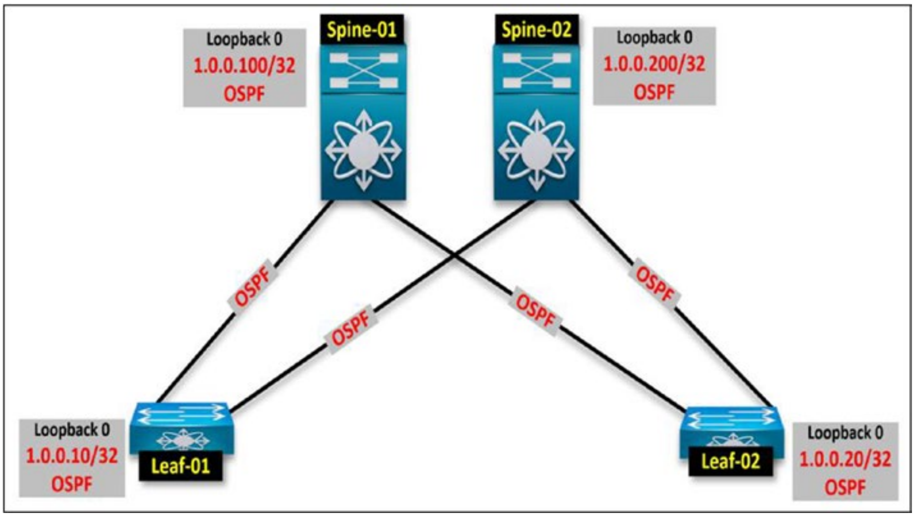


Figure 1-6. *After connecting all leaf-to-spine uplinks, you configure and establish the IGP. In this demonstration, I used OSPF. I then configured a loopback interface and advertised it in OSPF so all my fabric neighbors would know about it*

With the underlay routing in place, the next step is to use the loopback interface to peer BGP on top of OSPF. Leaf-01 has two valid paths to the spines. BGP is peered between them using the loopbacks (see Table 1-1).

Once you have reachability to the loopback interfaces, proceed to peer BGP between devices.

Table 1-1. *Loopbacks per Device*

BGP Neighbors	Leaf-01	Spine-01
	1.0.0.10	1.0.0.100
BGP Neighbors	**Leaf-01**	**Spine-02**
	1.0.0.10	1.0.0.200
BGP Neighbors	**Leaf-02**	**Spine-01**
	1.0.0.20	1.0.0.100
BGP Neighbors	**Leaf-02**	**Spine-02**
	1.0.0.20	1.0.0.200

Once you have performed this BGP peering, you have built the BGP underlay to carry VXLAN EVPN (see Figure 1-7).

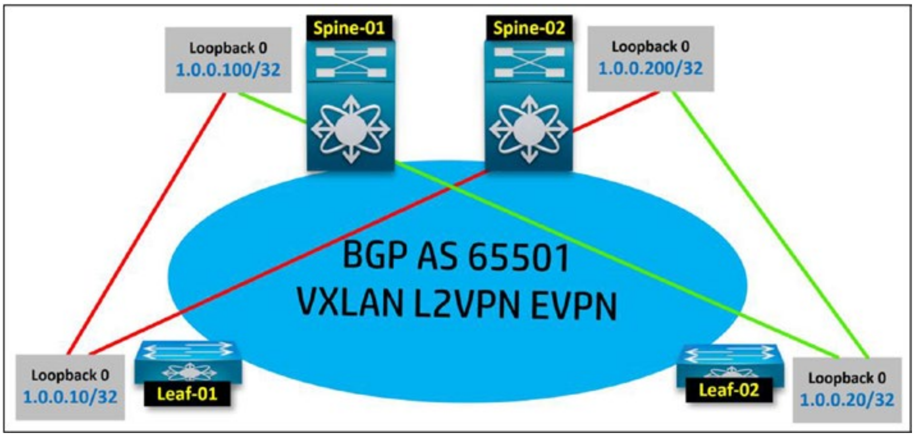


Figure 1-7. *A fully configured BGP backbone to carry VXLAN EVPN. Leaf-01 peered to both Spine-01 and Spine-02 and the same for Leaf-02, peered to both Spine-01 and Spine-02 in BGP*

Overlay Routing

You have the basic configuration needed to allow a “rollercoaster” to run the “cars.” In VXLAN terms, the underlay is ready to run VXLAN and the fabric tenants on top of it, or the overlay.

VXLAN runs the show in the overlay. You establish EVPN adjacency between the leafs. Why not the spines? **The spines do not encapsulate/ decapsulate L2 frames from/into VXLAN**. The spines are there to aggregate and serve as BGP route reflectors. (Don’t worry. I cover this later.)

You create a network virtual interface (NVE) and use it to encapsulate and decapsulate L2 frames. VXLAN uses the source loopback interface assigned to the NVE to establish a VXLAN tunnel between leafs.

The loopback assigned to the NVE is not the same as what you use to peer BGP. You could use it, but it is better to have a designated loopback for VXLAN NVE purposes. All your client traffic is tunneled when it arrives on the fabric via any leaf; then, this traffic is tunneled using the EVPN fabric between leafs.

It transparently passes through the spines; but you don’t see the spines from a traceroute perspective, only the source and destination leaf gateways. Figure 1-7 shows that loopback 0 is the peering loopback interface for BGP EVPN. You can then create another loopback to bring VXLAN tunnels between leafs.

Let’s choose loopback 1 as the NVE loopback. Once in place, you can effectively peer the NVEs. The loopback 1 interface address is called the Virtual Tunnel End Point (VTEP) (see Figure 1-8).

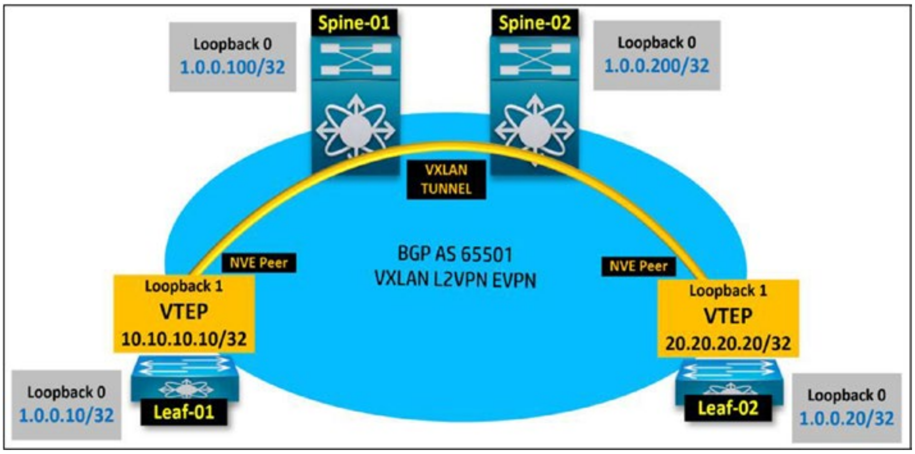


Figure 1-8. *VTEPs establish VXLAN tunnels between NVE peers (leafs)*

Now that you know how VXLAN is transported across the fabric, you need to understand a layer 2 communication flow in VXLAN from end to end. The following is a high-level walkthrough.

1. PC1 sends a communication request to PC2. PC1's local leaf is Leaf-01. PC2's local leaf is Leaf-02. PC1 sends the frame to Leaf-01.

2. Leaf-01 receives the L2 frame. It looks for the VLAN associated with the interface and encapsulates it in VXLAN by assigning the VNI associated with the client VLAN and adds the VXLAN overhead to the frame. All of this is performed at the NVE interface. The frame now becomes an L3 VXLAN frame.

3. Leaf-01 looks for the BGP EVPN route of the endpoint that PC1 is trying to reach. The BGP EVPN route is the MAC address and/or the end-host IP of the endpoint that you are trying to communicate with (PC2 in this case). Rather than having this

information stored in TCAM like traditional networks, it is a control plane that is learning and storing in BGP. It's called a VXLAN flavor— a VXLAN BGP EVPN control plane.

4. After Leaf-01 identifies that PC2's destination is located on Leaf-02 (the EVPN route table shows that the PC2 IP address is associated with the VTEP 20.20.20.20 (Leaf-02) address), Leaf-01 forwards the frame via the VXLAN tunnel from Leaf-01 to Leaf-02. It egresses out via the VXLAN tunnel. The tunnel source is VTEP 10.10.10.10 (Leaf-01). The tunnel destination is VTEP 20.20.20.20 (Leaf-02).

5. The VXLAN L3 packet arrives in Leaf-02 via the VXLAN tunnel associated with the PC's VNI on the NVE VTEP. Once the packet ingresses via the tunnel on the leaf, the leaf removes the L3 VXLAN header and forwards the packet in native L2 to the destination PC2.

Layer 3 Virtual Network Identifier (L3VNI)

You now understand how the overlay communicates with VLANs across the fabric, but how do different VLANs communicate? Remember, in EVPN, the MAC address is learned and stored in the BGP EVPN table as a route, specifically a type-2 route. (EVPN route types are covered in another chapter.) The source VLAN interface or default gateway gets this request from the endpoint to communicate to another VLAN. Also, the device you are trying to reach is at a different leaf, which means you need to traverse the EVPN fabric to get to the destination leaf. How do you accomplish this? With an L3VNI.

L3VNI is a VLAN with a VNI assigned to it exclusively to perform inter-VLAN traffic between leafs. If you want to communicate, let's say VLAN 10 on Leaf-01 to VLAN-20 on Leaf-02, the leaf sends this traffic over the L3VNI VXLAN tunnel assigned to the VRF (virtual routing and forwarding) that both VLANs belong to.

Per VRF, you need to designate an L3VNI so communication between VLANs can occur. To summarize, the architecture very simple: one VRF and one L3VNI and your VLANs. If you come from a service provider track, the L3VNI is like an MPLS label2 for VRF-VNI.

Figure 1-9 illustrates an example scenario. The VRF in the fabric is called tenant-a. It's assigned an L3VNI of 999999. L3VNI is the VXLAN fabric tunnel that VLANs use to communicate among different leafs.

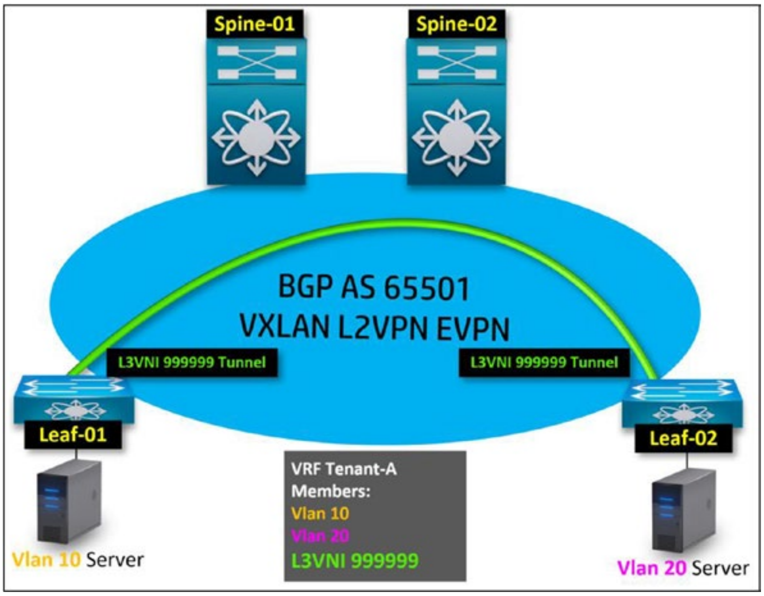


Figure 1-9. *L3VNI communicates with VLANs on the same VRF among different leafs*

Multipod Spine-and-Leaf VXLAN Topology

Let's discuss multipod VXLAN BGP EVPN architecture. One of the coolest features VXLAN offers is the ability to "stretch" your VLANs beyond your local network boundaries.

Local VLANs are available in different geographical locations—the same VLAN and the same subnet on two different sites. From a VLAN perspective, it thinks it's in the same location, but it's not!

Imagine this scenario. There are two data centers: one in Frankfurt, Germany, and one in Munich, Germany. VLAN10 is configured on both DC1 and DC2. VXLAN VNI for VLAN 10 is stretched using an EVPN route-target that imports and exports between data centers. (Later, I cover the configuration aspects.) From a server's perspective, it thinks its destination is on the same local VLAN (see Figure 1-10). How cool is that?

With multipod VXLAN, I can cross-connect my data center fabrics or pods together to act as one logical VXLAN fabric while at the same time having configuration parameters exclusive for each Pod. For example, in DC1, I have a tenant VRF named Frankfurt-Prod. In DC2, I have another tenant VRF named Munich-Prod. They are not named the same in either DC; however, I can import and export route-targets from the EVPN L3VNI and have BGP EVPN routes propagated between data centers. This means that I have the MAC address information from all my devices in the VLAN 10 on both data centers equally advertised on each VRF—DC1 to DC2 and DC2 to DC1. All my VLAN 10 endpoints communicate thinking they are in the same location! The service provider circuit is an L2 EVPN or VPLS circuit, like dark fiber.

I configure the circuit interfaces like all my other underlay links and advertise to OSPF so both DCs can get all their underlay information. Later, I provide low-level configurations, including the multipod configurations in BGP, to make all this magic happen.

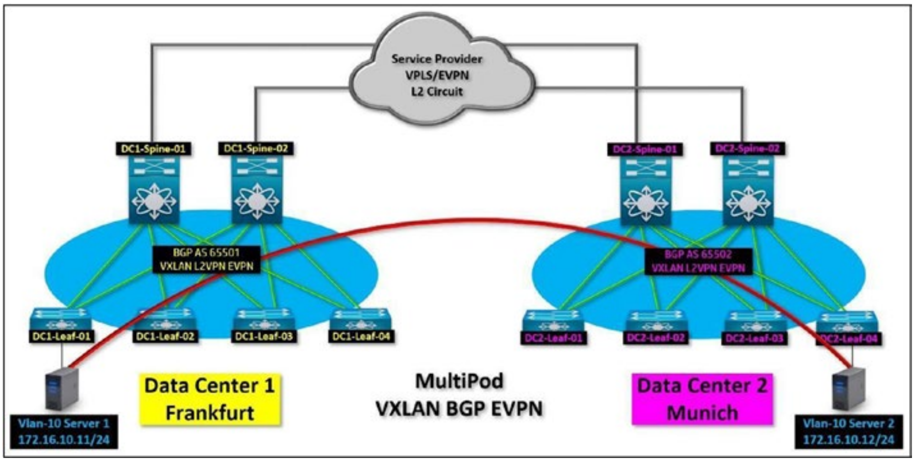


Figure 1-10. *In multipod VXLAN, you combine two geographically dispersed VXLAN fabrics and make them operate as one logical VXLAN fabric. VLAN 10 in the illustration has been stretched between data centers or pods*

Multisite Spine-and-Leaf VXLAN Topology

Let's discuss multisite architecture. In a multipod architecture, you combine at least two fabrics to make them look and behave like a single fabric. You use route-targets in EVPN to import and export your layer 2 information between pods (or fabrics). Technically, you could configure it to behave as one logical fabric. The logic behind a multisite architecture is quite different.

Multisite architecture has completely independent fabrics sharing specific resources and communicating via a middle service provider network. Let's say you have DC1 and DC2 in a multisite configuration. DC1 performs an eBGP neighbor relationship to your service provider, in which the service provider carries the EVPN traffic to your destination DC2.

The peering aspect occurs at the border gateway leaf (BGW). It is at this leaf where you communicate with your internal BGP (iBGP) autonomous system (AS) number to the external BGP (eBGP) AS number, which then relays to your service provider. The architecture is the same as DC2. BGW leaf peering your iBGP AS, which peers to your eBGP AS number, is illustrated in Figure 1-11.

The BGW leaf's role is to peer between iBGP and eBGP and perform what I call AS NAT, or translating the BGP AS number from internal to external for EVPN-specific traffic. In BGP, this is the rewrite AS or AS override function.

DC1 iBGP –> BGW eBGP –> Service Provider –> BGW eBGP –> iBGP DC2

Figure 1-11. *The full BGP path between two data centers in a multisite configuration*

Assume the following values: DC1 fabric AS is 65501, DC1 eBGP AS is 34563, DC2 fabric AS is 65502, and DC2 eBGP AS is 46673. The moment I send out my communication request to data center 2, the source of the communication changes from DC1 65501 to DC1 34563 eBGP AS. So, the BGW translates the 65501 to 34563 to move out to the service provider cloud.

Once the package gets to DC2, it is rewritten to 65502, which is the local DC2 fabric AS. The incoming traffic is spoofed to look like it's on the same fabric (see Figure 1-12).

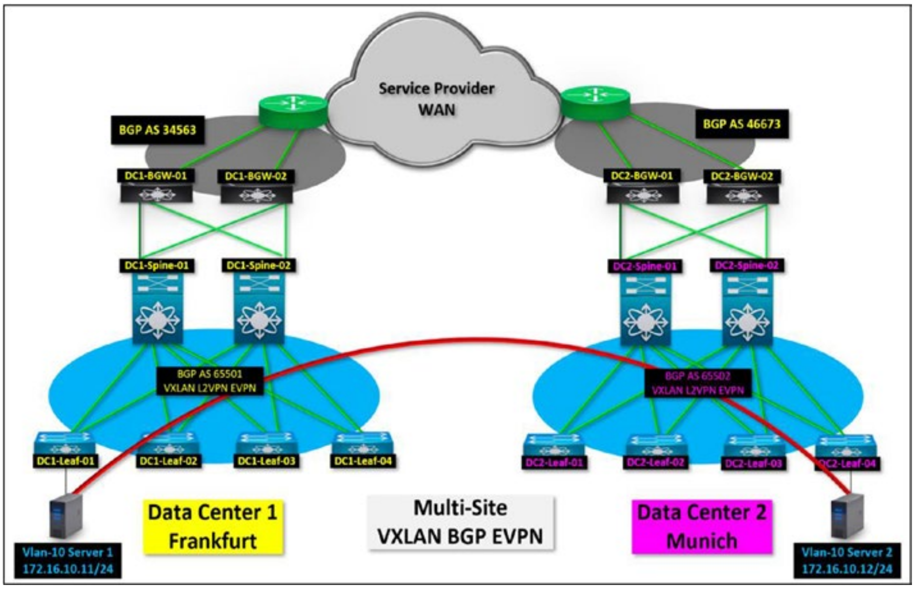


Figure 1-12. *The full BGP path between two data centers in a multisite configuration*

The VXLAN Border Leaf

By now, you should have a good understanding of spine-and-leaf architecture and how it operates from a VXLAN perspective. Next, I would like to discuss border leafs. You know the role of a leaf by now, but in the spine-and-leaf topology, you always have a **border leaf**. This leaf connects your fabric to an external network, such as the WAN, external firewall, L3 extensions, or external networks. Once it's participating in the client tenant VRF route advertisement or peering, the leaf that connects them is a border leaf.

What's the difference between this leaf and any other leaf if any leaf can be a border leaf? The border leaf has additional routing configurations performed on its CLI to make it happen.

Let's assume that you have a VXLAN fabric. Leaf-01 is connected to a router that participates in OSPF. You want to establish OSPF adjacency between the router and the fabric so that the router knows about the tenant networks, and the fabric knows about the router networks.

You configure OSPF and bring up adjacency between the router and Leaf-01. The tenant in the VXLAN fabric knows about the router's networks, but the other leafs have no idea about the router's networks because they are not connected and peered with OSPF to the router.

Connecting all the leafs to the router is not an efficient way to allow this communication. It is leaf-0's task to announce to the networks it knows about the rest of the VXLAN fabric. **Designate Leaf-01 as the border leaf by redistributing the external routes to the VXLAN fabric**.

The neighbor leafs then see the routes as EVPN type-5. The next hop shows the Leaf-01 VTEP loopback address. In an Internet-facing firewall, the scenario is the same; however, when you want to route traffic to the Internet, you use a 0.0.0.0/0 route that points to the inside-facing firewall interface IP, and you need to advertise this path in the fabric to the other leafs.

A 0.0.0.0/0 route cannot be advertised in BGP like any other route because BGP treats this route differently. In BGP, the network is advertised as a 0.0.0.0/0 route and adds a default-originate statement under the VRF so the route can originate from the border leaf inbound to the fabric. If you perform a show route for the tenant VRF you applied the route to, you find a 0.0.0.0/0 pointing again to the VTEP loopback interface address of the border leaf you originated the route from. It is shown as an EVPN type-5 route if you look at the BGP EVPN table.

Figure 1-13 doesn't show a redundant path to the ISP router, but redundancy is always required.

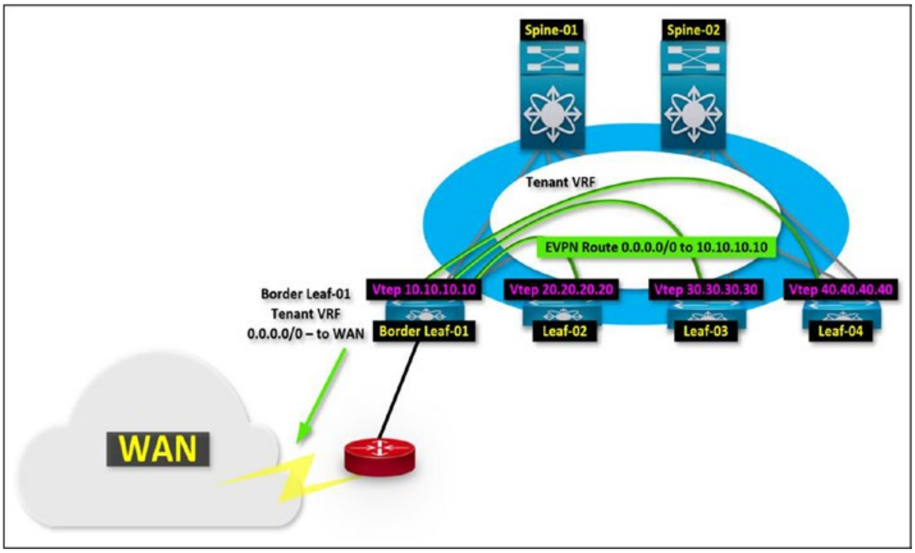


Figure 1-13. *Border Leaf-01 is connected to the ISP router and getting direct connectivity to the outside world. By redistributing the 0.0.0.0/0 in BGP EVPN, all other leafs in the VXLAN fabric see a 0.0.0.0/0 route on the assigned VRF pointing to border Leaf-01 VTEP address as next hop. This route is learned over the L3VNI tunnel from border Leaf-01*

External Path Redundancy in Spine-and-Leaf VXLAN BGP EVPN

How many times do you have to deal with redundancy? Always! I briefly discussed spine-and-leaf redundancy on a physical aspect. What about redundancy from a traffic flow perspective? You could have link redundancy by configuring some form of link aggregation. But let's say you don't have a physical link failure but a service provider issue that is not in your control to remediate. There is a secondary circuit in the data center that can be placed as a failover path if the main circuit goes down. But when it comes to implementing this kind of failover in VXLAN, well… it's kinda tricky. Not really. It requires some logical thinking when building redundant configuration.

Let's assume the following: Leaf-01 has connectivity via two service providers. Service provider 1's router hostname is SP1-RTR. Service provider 2's router hostname is SP2-RTR. SP1-RTR is the primary router. SP2-RTR is the secondary router. Assume that both service providers have an eBGP peer relationship to border Leaf-01. When you verify your tenant VRF route table, you see that all required destination routes point to SP1-RTR. To allow the redundancy to happen, you can manipulate BGP distance in the service provider neighbor.

BGP configurations to prefer a lower distance value set for service provider 1. When looking at the route table, you should see the external routes with a lower BGP distance, which is preferred for SP1-RTR. If SP1-RTR stopped advertising those routes for some reason, the leaf drops all SP1-RTR routes from the VRF, and BGP enables the routes from SP2-RTR (see Figure 1-14).

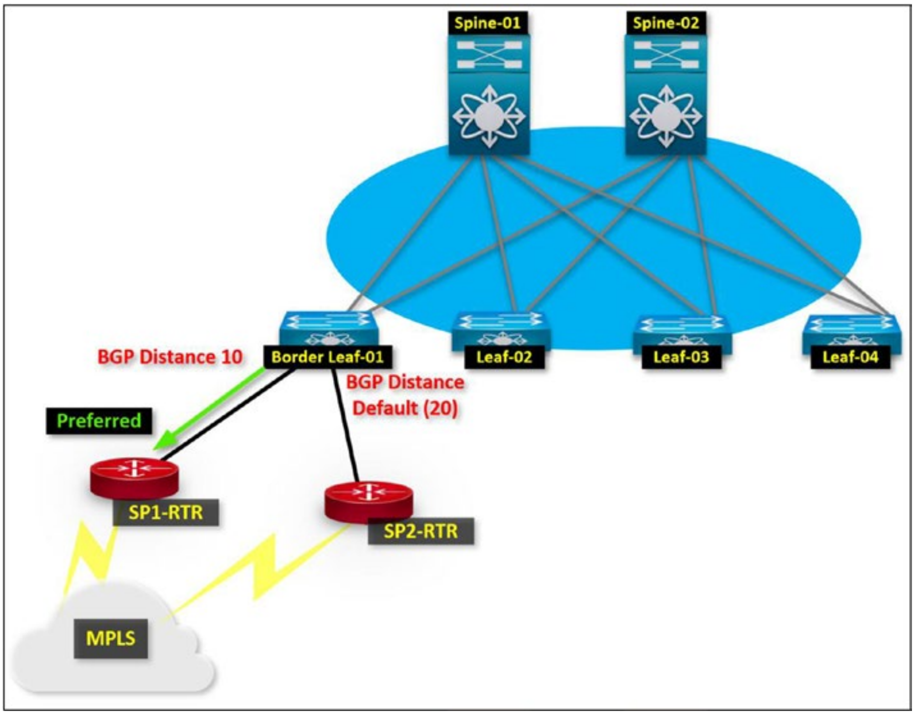


Figure 1-14. *Border Leaf-01 has two service provider connections to an MPLS cloud by manually configuring a specific BGP distance value lower than default for SP1-RTR. Border Leaf-01 used all routes advertised from SP1-RTR as preferred. If SP1-RTR goes down, Leaf-01 replaces the routes with the ones from SP2-RTR*

This sounds normal—like in traditional L3 core and three-tier designs, right? Well, yeah.... until you need to provide leaf and router redundancy. In the previous scenario, if you lose Leaf-01, you lose complete MPLS reachability. Both ISPs are out of reach since they are single-homed to only Leaf-01. A simple leaf firmware upgrade brings down your MPLS reachability. In today's data centers, this is not acceptable. It is the reason why you leverage multiple leafs to provide failover capabilities.

The redundancy layout in this scenario is simple. SP1-RTR and SP2-RTR are both dual-homed to Leaf-01 and Leaf-02. From a layer 3 perspective, you choose SP1-RTR to be the primary, but you must set a failover path from SP1-RTR to Leaf-02 because you don't want to failover the service provider side if you lose Leaf-01. You still want to use service provider 1 as the primary in either Leaf-01 or Leaf-02. If you lose SP1-RTR, SP2-RTR should take over on Leaf-01 since, from a leaf standpoint, nothing has happened to trigger traffic failover to Leaf-02. The topology shown in Figure 1-15 gives a clearer picture using a redundancy configuration.

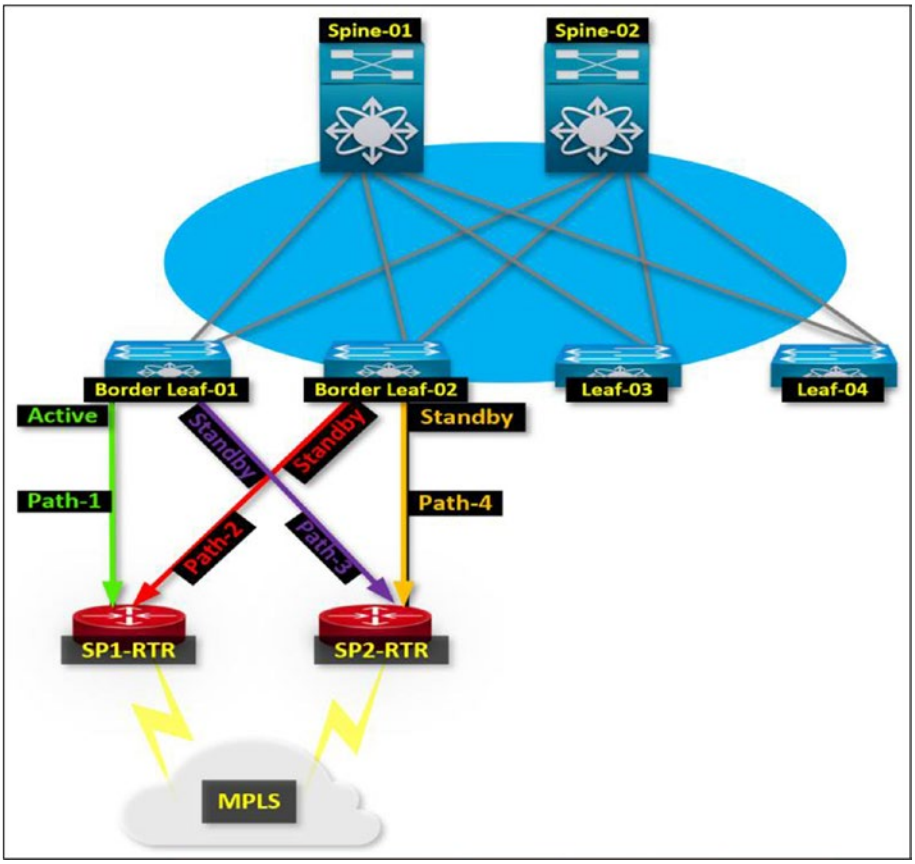


Figure 1-15. *In this type of redundancy, four valid paths are in the MPLS network from our VXLAN fabric. Two paths from Leaf-01, one to SP1-RTR, and another to SP2-RTR. The same goes for Leaf-02. The key is to properly architect a redundant connection by setting network priorities and distance to obtain the desired failover without inducing routing issues*

When implementing this type of architecture, it is very common to run into asymmetric route issues due to the complex failover configuration in the route process you need to perform. Maybe the path from Leaf-02 to SP1-RTR is different from the rest of the leafs. It could create quite a headache trying to trace and figure out the route path issue. Don't worry. I wrote this book to help you avoid the pitfalls of asymmetric routing issues.

After reading this chapter, I hope that you now have a basic idea of why and how VXLAN BGP EVPN operates to provide scalability and performance for today's data center requirements. It covered the high-level overview of the underlay, the overlay, and the logic behind VXLAN BGP EVPN routing from a layer 2 or layer 3 perspective.

Chapter 2 covers the VXLAN underlay elements on a low level. And, guess what? You are also configuring it. Let the fun begin! Let's get our hands dirty and start configuring your first VXLAN BGP EVPN spine-and-leaf architecture!

CHAPTER 2

Logical Components of a VXLAN BGP EVPN Spine-and-Leaf Architecture

Let's begin your VXLAN nitty-gritty learning phase. After reviewing VXLAN BGP EVPN spine-and-leaf topologies from a high-level perspective, let's discuss what you need to bring the VXLAN BGP EVPN fabric to life. Recall how the underlay provides the fabric's communication foundation. The VXLAN protocol runs on top of the underlay.

As a refresher, the underlay simply allows IP reachability from a leaf-to-spine standpoint. You need to provide a "backbone" architecture to run the BGP services. Imagine you're building a data center. You don't begin the construction by installing the rack and servers, right? You start by adding air conditioner units, wiring, power, lighting, and so forth. The same goes with the logic behind the fabric underlay. To have a fully functioning VXLAN fabric, you need to first configure the IGP you intend to run between the leafs and spines. You need to set your multicast configurations so the fabric leaf can announce what the VXLAN endpoints learned locally (BUM) to every neighbor leaf.

© Rene Cardona 2021
R. Cardona, *The Fast-Track Guide to VXLAN BGP EVPN Fabrics*,
https://doi.org/10.1007/978-1-4842-6930-5_2

You need to configure the loopbacks that carry BGP and VXLAN advertisements. The BGP peering happens on the underlay as well. There's quite a to-do list of items you need to complete before bringing the overlay into the fabric.

Configuring the IGP

Let's begin Chapter 2 by configuring our IGP to allow leaf-to-spine IP reachability for all the fabric devices. I cover the OSPF configuration from a physical interface standpoint, which is the first option you could use to build your underlay. The second option is to configure the IGP to use the IS-IS route protocol. With IS-IS, you can give the VXLAN fabric a service provider architecture approach. IS-IS provides robust service provider class options to peer your leafs to the spines.

Note When configuring your underlay IGP, select either OSPF or IS-IS. I can't suggest either one or the other since it all depends on your post-support network engineer resource to administer OSPF or IS-IS.

First, let's get familiar with the VXLAN topology that you configure in this chapter. It's a very straightforward fabric: two Cisco Nexus 9300 spines and four Cisco Nexus 9500 leafs. Arista, Juniper, and Cumulus Networks, to name a few, are other vendors that have a VXLAN platform comparable to Cisco. The CLI syntax could be different; however, learning the concepts from a configuration standpoint allows you to support all platforms.

Okay, enough with the talking—let's get to the action. Figure 2-1 shows the VXLAN topology configured in this chapter. Please review the connection paths from all leafs to all spines.

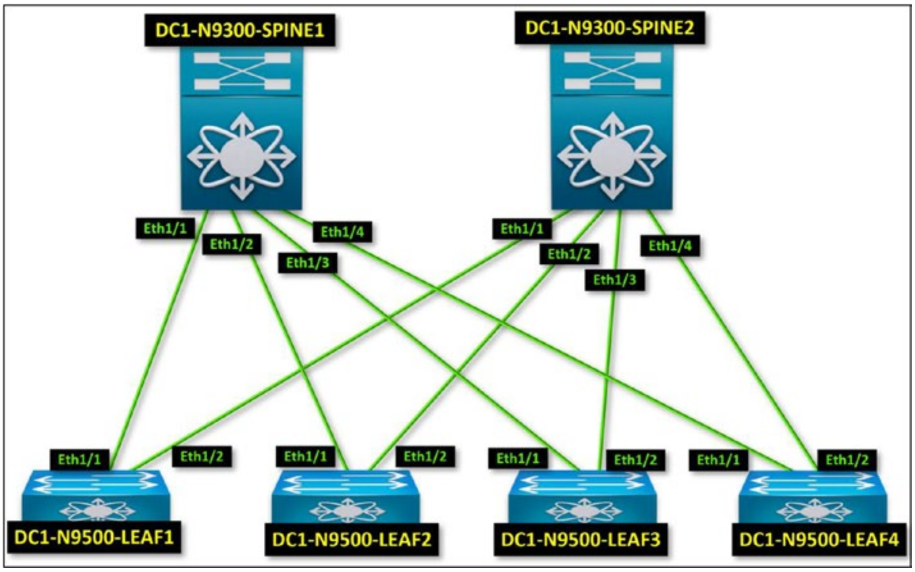


Figure 2-1. *A two-spine/four-leaf topology physical wiring matrix. Let's configure it using OSPF as our first option and IS-IS as our second option*

OSPF Point-to-Point for the Underlay (Option 1)

Okay, let's build the VXLAN underlay using OSPF. If you're working with Cisco's Nexus platform, the first thing you need to do is enable the OSPF feature, then configure your OSPF process. When configuring your OSPF process, you should assign the OSPF router ID. It's my preference to use the loopback0 IP as the OSPF router ID. From an IP numbering perspective, you should provide an IP address scheme relevant to the data center location and device number.

I used 10.0.0.*x* for DC1, where *x* is the leaf number, and 10.1.0.*y*, where *y* is the spine number. A similar approach can be used for DC2; so instead of 1, you use 2 to begin the address assignment. For example, DC1-N9500-Leaf-01's designated loopback0 interface address is 10.0.0.1/32. All leafs in DC1 and DC2 have the designated uplink to Spine-01 on port eth1/1 and Spine-02 on port eth1/2. All L3 interfaces participating in OSPF should be added to the OSPF process.

One very important item to note: the VXLAN overhead added to the layer 3 packet adds a significant increase to the frame size. You need to allow more than 1500 MTU to all the layer 3 underlay links and enable jumbo frames.

It's time to configure the physical interfaces with 9000 MTU or higher, making sure both leaf and spine sides match. With this information, let's configure OSPF (see Listing 2-1).

Listing 2-1. Configure the OSPF Underlay in DC1-N9500-Leaf-01

```
feature ospf
router ospf DC1-Underlay !(Designate an ospf process id/name)
router-id 10.0.0.1

int loopback0 ! (create the loopback0 interface)
ip address 10.0.0.1/32 ! (loopback 0 IP address)
ip router ospf DC1-Underlay area 0 !(add ospf process)
ip ospf network point-to-point ! (Set OSPF Point-to-Point)
!(Lets now configure the link on Leaf1 going to Spine1)
Int eth1/1
description TO_DC1_N9300_SPINE1_ETH1/1
No switchport
ip address 10.1.1.1/30 ! (IP format dc.spine.leaf.x/30)
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point
mtu 9000
```

```
!(Lets now configure the link on Leaf1 going to Spine2)
int eth1/2
description TO_DC1_N9300_SPINE2_ETH1/1
No switchport
ip address 10.2.1.1/30
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point
mtu 9000

!(The Full OSPF Underlay Configuration for DC1-N9500-LEAF1)
feature ospf
router ospf DC1-Underlay
router-id 10.0.0.1

int loopback0
ip address 10.0.0.1/32
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point

int eth1/1
description TO_DC1_N9300_SPINE1_ETH1/1
no switchport
ip address 10.1.1.1/30
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point
mtu 9000
no shut

int eth1/2
description TO_DC1_N9300_SPINE2_ETH1/1
no switchport
ip address 10.2.1.1/30
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point
mtu 9000
no shut
```

Note A /31 subnet mask is another option for configuring the physical interface. You can set the interface as point-to-point and use the network address as a host address; for example, 10.0.0.**0**/31 – 10.0.0.**1**/31. Since OSPF is running point to point, you don't need broadcast to bring up adjacency (see Listing 2-2).

Listing 2-2. Optional: Set Interface Mode in Point-to-Point and Use /31

```
int eth1/1
medium p2p
ip address 10.1.1.0/31
```

Template-Based Deployment

VXLAN fabrics are meant to be deployed and managed using automation techniques with Python or Ansible-based scripts. To show you how easy this is once your initial template is built is to deploy on all leafs, I copied and pasted from the Leaf-01 configuration template and replaced a single value from 1 to 2 (see Listing 2-3).

Listing 2-3. DC1-N9500-Leaf-02 Config Script

```
feature ospf
router ospf DC1-Underlay
router-id 10.0.0.2

int loopback0
```

```
ip address 10.0.0.2/32
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point

int eth1/1
description TO_DC1_N9300_SPINE1_ETH1/2
no switchport
ip address 10.1.2.1/30
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point
mtu 9000
no shut

int eth1/2
description TO_DC1_N9300_SPINE2_ETH1/2
no switchport
ip address 10.2.2.1/30
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point
mtu 9000
no shut
```

With the information provided, create the configurations for DC1-N9500-Leaf-03 and DC1-N9500-Leaf-04.

Next, let's build the OSPF underlay configuration for the spines. It is almost the same as the leafs but assigns a unique IP for each required interface.

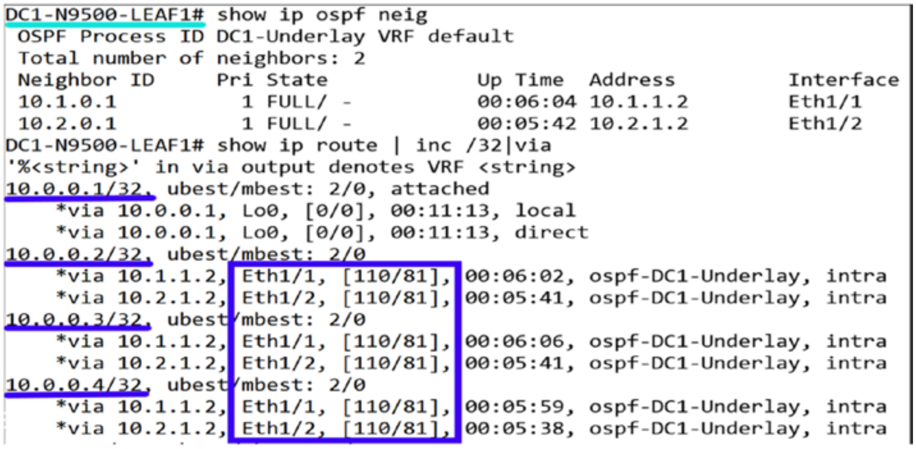
To make this simpler, I copied and pasted the configuration template from Leaf-02 and replaced the values. Once you build the first configuration, the other ones are a copy-and-paste-then-replace-the-values procedure (see Listing 2-4).

Listing 2-4. DC1-N9300-Spine-01 Config Script

```
feature ospf
 router ospf DC1-Underlay
 router-id 10.1.0.1

int loopback0
 ip address 10.1.0.1/32
 ip router ospf DC1-Underlay area 0
 ip ospf network point-to-point

int eth1/1
 description TO_DC1_N9500_LEAF1_ETH1/1
 no switchport
 ip address 10.1.1.2/30
 ip router ospf DC1-Underlay area 0
 ip ospf network point-to-point
 mtu 9000
 no shut

int eth1/2
 description TO_DC1_N9500_LEAF2_ETH1/1
 no switchport
 ip address 10.1.2.2/30
 ip router ospf DC1-Underlay area 0
 ip ospf network point-to-point
 mtu 9000
 no shut

int eth1/3
 description TO_DC1_N9500_LEAF3_ETH1/1
 no switchport
 ip address 10.1.3.2/30
 ip router ospf DC1-Underlay area 0
 ip ospf network point-to-point
 mtu 9000
 no shut

int eth1/4
 description TO_DC1_N9500_LEAF4_ETH1/1
 no switchport
 ip address 10.1.4.2/30
 ip router ospf DC1-Underlay area 0
 ip ospf network point-to-point
 mtu 9000
 no shut
```

Once you copy and paste your template into the command-line interface (CLI) across all fabric leafs and spines, confirm the OSPF status and whether you are advertising the loopback0 interface with an ECMP route.

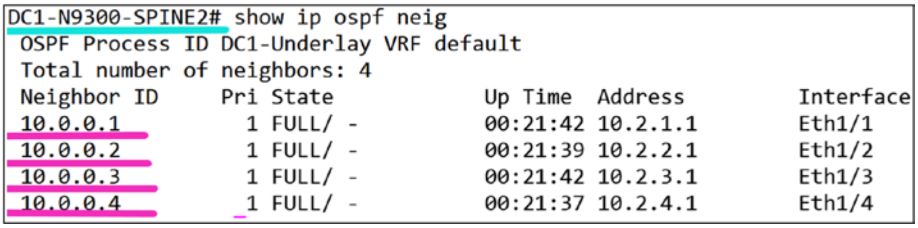
What do you need to look for in DC1-N9500-Leaf-01? You need to confirm: OSPF neighbor state in *full* reachability to Spine-01 and Spine-02, since one link goes to each spine. You are also doing point-to-point OSPF, and both links should have **equal-cost multipath** (ECMP) (see Figure 2-2).



```
DC1-N9500-LEAF1# show ip ospf neig
 OSPF Process ID DC1-Underlay VRF default
 Total number of neighbors: 2
 Neighbor ID     Pri State            Up Time  Address         Interface
 10.1.0.1          1 FULL/ -          00:06:04 10.1.1.2        Eth1/1
 10.2.0.1          1 FULL/ -          00:05:42 10.2.1.2        Eth1/2
DC1-N9500-LEAF1# show ip route | inc /32|via
'%<string>' in via output denotes VRF <string>
10.0.0.1/32, ubest/mbest: 2/0, attached
    *via 10.0.0.1, Lo0, [0/0], 00:11:13, local
    *via 10.0.0.1, Lo0, [0/0], 00:11:13, direct
10.0.0.2/32, ubest/mbest: 2/0
    *via 10.1.1.2, Eth1/1, [110/81], 00:06:02, ospf-DC1-Underlay, intra
    *via 10.2.1.2, Eth1/2, [110/81], 00:05:41, ospf-DC1-Underlay, intra
10.0.0.3/32, ubest/mbest: 2/0
    *via 10.1.1.2, Eth1/1, [110/81], 00:06:06, ospf-DC1-Underlay, intra
    *via 10.2.1.2, Eth1/2, [110/81], 00:05:41, ospf-DC1-Underlay, intra
10.0.0.4/32, ubest/mbest: 2/0
    *via 10.1.1.2, Eth1/1, [110/81], 00:05:59, ospf-DC1-Underlay, intra
    *via 10.2.1.2, Eth1/2, [110/81], 00:05:38, ospf-DC1-Underlay, intra
```

Figure 2-2. *Leaf-01 confirmed OSPF neighbor adjacency with Spine-01 and Spine-02 by executing the show `ip OSPF neighbor` command. You see ECMP for the Leafs-02,03,04 loopback0 address via the Leaf-01 links eth1/1 and eth1/2. Run the `show ip route` command on all leafs to confirm this information*

What do you need to look for on Spine-01 and Spine-02? You need to confirm the same information: OSPF neighbor state in full reachability to Leaf-01, Leaf-02, Leaf-03, and Leaf-04 since one link goes from each spine to each leaf. Let's look at Spine-02 (see Figure 2-3).



```
DC1-N9300-SPINE2# show ip ospf neig
 OSPF Process ID DC1-Underlay VRF default
 Total number of neighbors: 4
 Neighbor ID     Pri State            Up Time  Address         Interface
 10.0.0.1          1 FULL/ -          00:21:42 10.2.1.1        Eth1/1
 10.0.0.2          1 FULL/ -          00:21:39 10.2.2.1        Eth1/2
 10.0.0.3          1 FULL/ -          00:21:42 10.2.3.1        Eth1/3
 10.0.0.4          1 FULL/ -          00:21:37 10.2.4.1        Eth1/4
```

Figure 2-3. *DC1-N9300-Spine-02 confirmed OSPF neighbor adjacency to all leafs in the fabric. The same information must be confirmed in Spine-01*

That's it for an OSPF underlay. If you want to assign a link to have higher OSPF priority than others in the interface, you want OSPF to be preferred. Add the `ip ospf cost 1` command to lower the cost, so it's preferred. I chose the value 1 since it's the lowest available. By default, this is the value on 1 GbE and 10 GbE links. If you have two links, but one is slower than the other, then yes, it's a good practice to lower the cost value on the faster one so traffic always goes to it instead of both links.

IS-IS for the Underlay (Option 2)

Leaf-01	Loopback0	Ethernet1/1
(Figure 2-4)	1.0.0.10	1.0.0.10

A complete underlay fabric reachability is built with OSPF. Let's perform the same configuration, but let's select IS-IS as our IGP. In this scenario, I cover an even simpler approach to configuring your underlay. This time, let's configure it without assigning any IPs to the physical interfaces. What??? I can imagine what you're thinking. How do leafs and spines reach each other if there's no layer 3 address configured on their links?

The answer is the `ip unnumbered` command that is applied to interfaces. How does it work? Stay with me on this one. You borrow an available existing layer 3 interface IP address and assign it to the unnumbered interface. For example, assume you have a leaf with a configured loopback0 interface, and you need to configure Ethernet1/1, which connects to Spine-01. In this case, you set `ip unnumbered loopback0` in the interface. I assigned the same Lo0 IP to the Ethernet interface 1.0.0.10 being the loopback0 IP. Don't worry. I'll show the output as the underlay is configured with IS-IS.

The IS-IS routing approach is like OSPF since both use a shortest-path-first (SPF) algorithm. It can logically segment networks into areas like OSPF. It also uses a hello message to bring up adjacency, like OSPF.

In IS-IS, you set levels (level-1 or level-2) to inform the protocol that your routes are all under the same NET (network entity title), with IS-IS type level-1 or between different areas, with IS-IS type level-2. The NET value must be configured on all leafs and spines with a unique digit to differentiate each ID. Think of the NET value as your router ID.

Let's configure the underlay with IS-IS by using the same architecture as the previous OSPF underlay section. You are only configuring the layer 3 address under the loopback0 interface. All physical links are configured with the unnumbered command mapped to loopback0. Listing 2-5 is the IS-IS initial configuration.

Listing 2-5. Configure the IS-IS Underlay in DC1-N9500-Leaf-01

```
feature isis
router isis Underlay (we designate an IS-IS process id or name)
  net 47.0000.004d.0001.0001.0001.00 (NET ISIS SystemID).
  is-type level-1 (Level-1, No other IS-IS network participate)
int loopback0
  ip address 10.0.0.1/32
  ip router isis Underlay

int eth1/1
description TO_DC1_N9300_SPINE1_ETH1/1
medium p2p (Interface must be set point-to-point)
ip unnumbered lo0 (Set unnumbered and reference to Loopback0)
isis network point-to-point
ip router isis Underlay
mtu 9000
no shut
int eth1/2
description TO_DC1_N9300_SPINE2_ETH1/1
no switchport
ip unnumbered lo0
```

```
isis network point-to-point
ip router isis Underlay
mtu 9000
no shut
```

Let's modify the previous leaf IS-IS underlay template configuration to make it available for DC1-N9300-Spine-01. Since the `ip unnumbered` command is used in physical interfaces, the template is even easier. The only thing that needs to be changed is the IS-IS NET value and the loopback0 address. Please review the DC1-N9300-Spine-01 configuration in Listing 2-6.

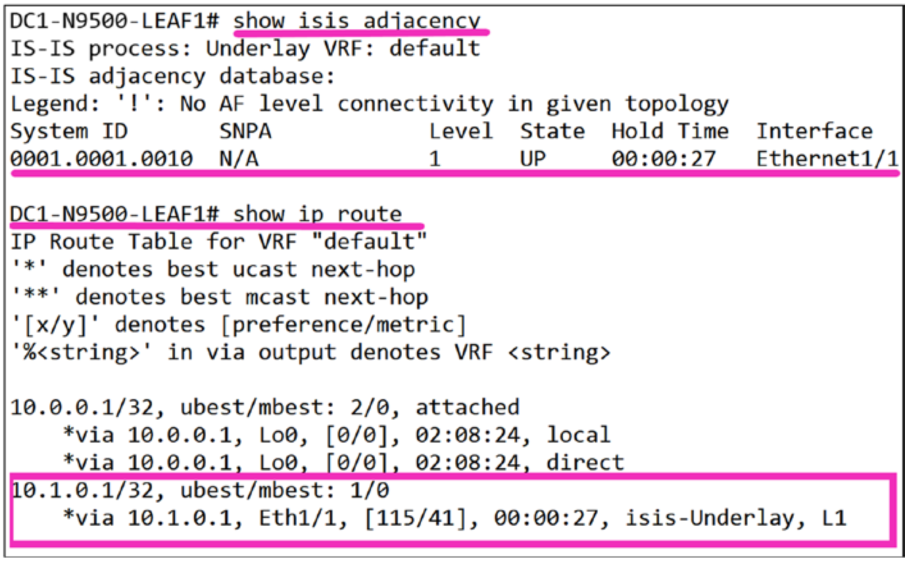
Listing 2-6. IS-IS Underlay Configuration for DC1-N9300-Spine-01

```
feature isis
router isis Underlay
  net 47.0000.004d.0001.0001.0011.00
  is-type level-1
int loopback0
  ip address 10.1.0.1/32
  ip router isis Underlay

int eth1/1
description TO_DC1_N9300_SPINE1_ETH1/1
medium p2p
ip unnumbered lo0
isis network point-to-point
ip router isis Underlay
mtu 9000
no shut
int eth1/2
description TO_DC1_N9500_LEAF2_ETH1/1
no switchport
```

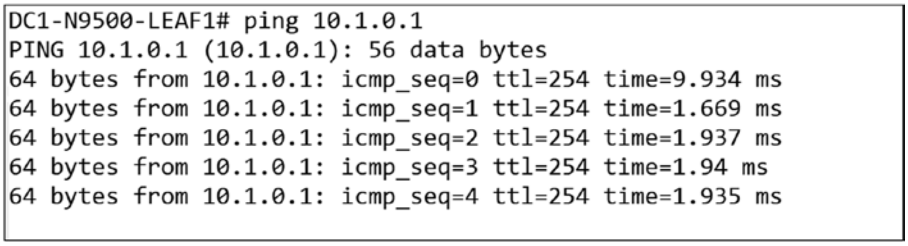
```
ip unnumbered lo0
isis network point-to-point
ip router isis Underlay
mtu 9000
no shut
int eth1/3
description TO_DC1_N9500_LEAF3_ETH1/1
no switchport
ip unnumbered lo0
isis network point-to-point
ip router isis Underlay
mtu 9000
no shut
int eth1/4
description TO_DC1_N9500_LEAF4_ETH1/1
no switchport
ip unnumbered lo0
isis network point-to-point
ip router isis Underlay
mtu 9000
no shut
```

All the configurations required from an IS-IS standpoint have been completed.

Let's confirm a successful IS-IS adjacency by running the `show isis adjacency` command to verify that IS-IS routes are distributed between DC1-N9500-Leaf-01 and DC1-N9300-Spine-01 (see Figure 2-4).



```
DC1-N9500-LEAF1# show isis adjacency
IS-IS process: Underlay VRF: default
IS-IS adjacency database:
Legend: '!': No AF level connectivity in given topology
System ID       SNPA            Level  State  Hold Time  Interface
0001.0001.0010  N/A             1      UP     00:00:27   Ethernet1/1

DC1-N9500-LEAF1# show ip route
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.0.0.1/32, ubest/mbest: 2/0, attached
    *via 10.0.0.1, Lo0, [0/0], 02:08:24, local
    *via 10.0.0.1, Lo0, [0/0], 02:08:24, direct
10.1.0.1/32, ubest/mbest: 1/0
    *via 10.1.0.1, Eth1/1, [115/41], 00:00:27, isis-Underlay, L1
```



```
DC1-N9500-LEAF1# ping 10.1.0.1
PING 10.1.0.1 (10.1.0.1): 56 data bytes
64 bytes from 10.1.0.1: icmp_seq=0 ttl=254 time=9.934 ms
64 bytes from 10.1.0.1: icmp_seq=1 ttl=254 time=1.669 ms
64 bytes from 10.1.0.1: icmp_seq=2 ttl=254 time=1.937 ms
64 bytes from 10.1.0.1: icmp_seq=3 ttl=254 time=1.94 ms
64 bytes from 10.1.0.1: icmp_seq=4 ttl=254 time=1.935 ms
```

Figure 2-4. *DC1-N9500-Leaf-01 confirms that via Eth1/1 established IS-IS adjacency to DC1-N9300-Spine-01. There is reachability to the loopback0 address on Spine-01 via its unnumbered interface. A simple ping shows full reachability to the address from Leaf-01*

Multicast and Rendezvous Points (RP)

Now that the IGP routing is configured for the underlay, let's configure the multicast requirements so VXLAN can announce the BUM traffic across all fabric leafs. If you haven't configured or supported multicast, let me give you a crash course on how it works from a VXLAN functionality perspective.

In multicast, senders and receivers “talk” and “listen” on a particular multicast group address. A sender talks to a group address. All the receivers listening in the same group address get the message from the sender. The sender’s “talk” is the BUM traffic announcement. Every VNI has a designated multicast address, so every BUM announcement that a particular VLAN does in VXLAN is sent to a group. All receivers (leafs) participating in the group get that announcement.

There’s one last item that plays an important role in this multicast traffic: the **rendezvous point** (RP). What is the rendezvous point? It is a “meeting point” for all multicast traffic. The sender and the receiver “meet” at the RP and start talking and listening. In the spine-and-leaf topology, you usually designate the spines as your multicast RPs. Two multicast RP configuration options are available.

Option 1: Anycast RP

An anycast RP allow you to effectively have active-active RPs. This means that instead of having a single RP, you could have the same RP active on all spines.

Using the same VXLAN topology, let’s build an anycast RP configuration. One additional loopback interface must be created for this purpose. This loopback interface is the designated anycast RP interface. You already have Spine-01’s loopback0 configured with 10.1.0.1/32. Let’s create another loopback, loopback1, and designate it as the anycast RP interface. Let’s assign an IP to it. I assigned some value to make this relevant to Spine-01 and Spine-02. I used the 10.**1**.10.**2**/32 address. I used the following scheme: *datacenter1**.spine1.**datacenter1**.spine2**/32*. You don’t have to do this, but it’s my way of making sense that this is the anycast RP. Remember, the loopback1 interface needs to be the same on both Spine-01 and Spine-02, meaning they have the same 10.1.10.2/32 address. Once you create loopback1, make sure to advertise it in OSPF or IS-IS, like `loopback0` (see Listing 2-7).

Listing 2-7. Create the Same loopback1 Interface in Spine-01 and Spine-02

```
int loopback1
  description ANYCAST-RP
  ip address 10.1.10.2/32
  ip router isis Underlay
```

After you finalize the anycast RP interface configuration and advertise it in the IGP routing process, **you need to enable the multicast PIM sparse mode functionality on a L3 links and interfaces participating in the underlay**. Loopback and Ethernet interfaces must have the `pim-sparse` command enabled. If you are working on the Nexus platform, make sure that you first enable the pim feature to enable multicast in the interfaces.

In Listing 2-8, I enabled the pim feature, then configured `ip pim sparse-mode` in the loopback0 and loopback1 interfaces. The physical interface range is from eth1/1 to eth1/4, since those are my underlay L3 links to the leafs. **This must be done on all the spines and leafs**.

Listing 2-8. Example on DC1-N9300-Spine-01

```
DC1-N9300-SPINE1
feature pim

int lo0
ip pim sparse-mode
int lo1
ip pim sparse-mode

int eth1/1-4
ip pim sparse-mode
```

Make sure PIM is enabled and active on all your fabric underlay links. You need to configure the anycast RP–specific configurations on both Spine-01 and Spine-02. The configuration tells the spine switch that the anycast RP spine member interface loopback0 is replying to any multicast-specific messages. **Both config lines must be applied to both spines**.

```
ip pim anycast-rp Anycast-RP-lo1 SPINE1-Lo0
ip pim anycast-rp Anycast-RP-lo1 SPINE2-Lo0
ip pim anycast-rp 10.1.10.2 10.1.0.1
ip pim anycast-rp 10.1.10.2 10.1.0.2
```

The anycast RP is configured on both spines, so you need to add the statements to finalize the multicast configurations. First, designate a multicast address space to grab the multicast group values from. You can choose two flavors from in multicast: any-source-multicast (ASM) or source-specific-multicast (SSM).

If you remember, the sender and receiver function in multicast. The **source is the sender who sends the information out to the designated multicast group. It is distributed to any receiver participating in it. In multicast, the value is classified as (S, G) or (source, group)**.

Let's assume you have VLAN 10. The VLAN 10 VNI is 100010. The 224.0.0.10 multicast group was assigned to it. Leaf-01 decides to make a BUM traffic announcement for VLAN 10 so that all the other leafs in the fabric know about it.

In the multicast route, you see the (S, G) with the Leaf-01 VXLAN VTEP and the VLAN 10 multicast group. It looks like this: (10.10.10.10, 224.0.0.10). Leaf-01 is the sender announcing to 224.0.0.10. What happens next? The group message is sent to the RP because all the multicast messages point to the RP address when you configure the multicast parameters on the leaf. Listing 2-9 is a multicast configuration for Leaf-01. The configuration is very straightforward. You set our multicast message for any group under the address space 224.0.0.0/16 to the RP address of 10.1.10.2, which is the Anycast RP address previously configured. All are underlay L3 interfaces; make sure they are enabled for multicast with `ip pim sparse-mode`.

Listing 2-9. Enabling multicast using PIM sparse-mode on L3 interfaces

```
ip pim rp-address 10.1.10.2 group-list 224.0.0.0/16

int lo0
ip pim sparse-mode

int eth1/1-2
ip pim sparse-mode
```

Now that all the leafs are configured with the same information, let’s do a high-level walk-through of a multicast message. The multicast flavor you need to choose is ASM. Why? Leaf-01 is a multicast sender, and so are all other leafs. By having all the leafs send and receive their requests via the RP, the multicast message is transmitted and received successfully. Every leaf “meets” at the RP location to send and receive messages.

What happens if you lose Spine-01? Nothing. Because the same RP is configured on Spine-02. You can provide multicast RP redundancy and load balance between two active devices. In ASM, the (S, G) becomes the (X, G) where X can be any source—Leaf-01, Leaf-02, Leaf-03, or Leaf-04.

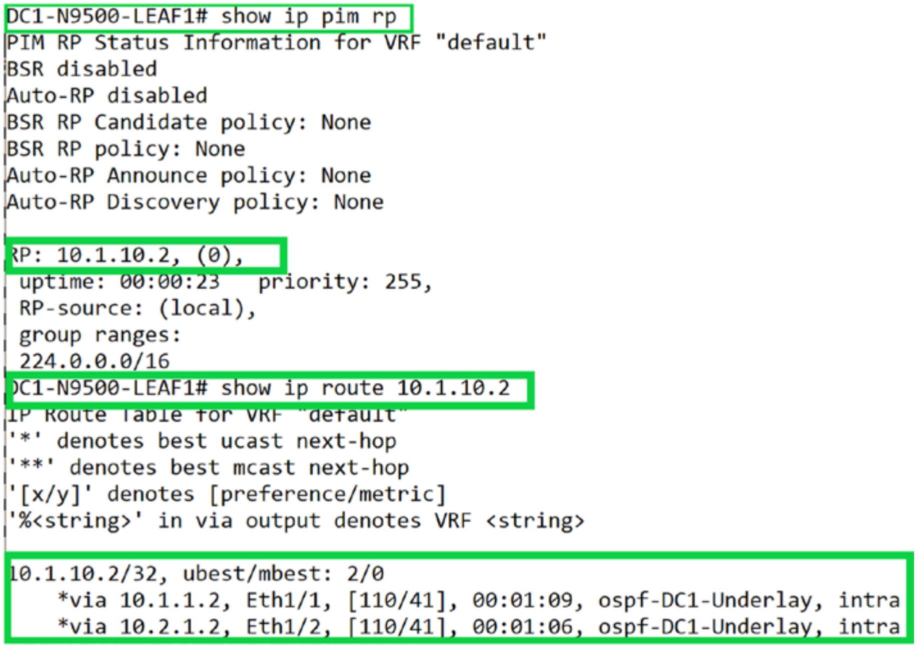
Tip Most underlay troubleshooting issues are multicast-related. A common scenario: leaf02 can’t see MACs from any fabric leaf.

Listing 2-10 is a fully working template that you can follow in your configuration.

Listing 2-10. The complete multicast configuration on a VXLAN spine switch

```
DC1-N9300-SPINE1 and DC1-N9300-SPINE2
feature pim

ip pim rp-address 10.1.10.2 group-list 224.0.0.0/16
ip pim anycast-rp 10.1.10.2 10.1.0.1
ip pim anycast-rp 10.1.10.2 10.1.0.2

int lo0
ip pim sparse-mode

int lo1
Description ANYCAST-RP
ip address 10.1.10.2/32
ip router ospf DC1-Underlay area 0
ip pim sparse-mode

int eth1/1-4
ip pim sparse-mode

ALL LEAF
feature pim
ip pim rp-address 10.1.10.2 group-list 224.0.0.0/16

int lo0
ip pim sparse-mode

int eth1/1-2
ip pim sparse-mode
```

It's time to trust but verify. Let's see how the fabric sees the anycast RP path from a leaf standpoint (see Figure 2-5).



```
DC1-N9500-LEAF1# show ip pim rp
PIM RP Status Information for VRF "default"
BSR disabled
Auto-RP disabled
BSR RP Candidate policy: None
BSR RP policy: None
Auto-RP Announce policy: None
Auto-RP Discovery policy: None

RP: 10.1.10.2, (0),
 uptime: 00:00:23   priority: 255,
 RP-source: (local),
 group ranges:
 224.0.0.0/16
DC1-N9500-LEAF1# show ip route 10.1.10.2
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.1.10.2/32, ubest/mbest: 2/0
    *via 10.1.1.2, Eth1/1, [110/41], 00:01:09, ospf-DC1-Underlay, intra
    *via 10.2.1.2, Eth1/2, [110/41], 00:01:06, ospf-DC1-Underlay, intra
```

Figure 2-5. *DC1-N9500-Leaf-01 shows the designated RP address is 10.1.10.2. Checking its route table shows two paths to 10.1.10.2, via DC1-N9300-Spine-01 and DC1-N9300-Spine-02, making it an anycast RP since it is active-active on both spines*

With the configuration in place, DC1-N9300-Spine-01 and DC1-N9300-Spine-02 are both active RPs (anycast RPs) (see Figure 2-6).

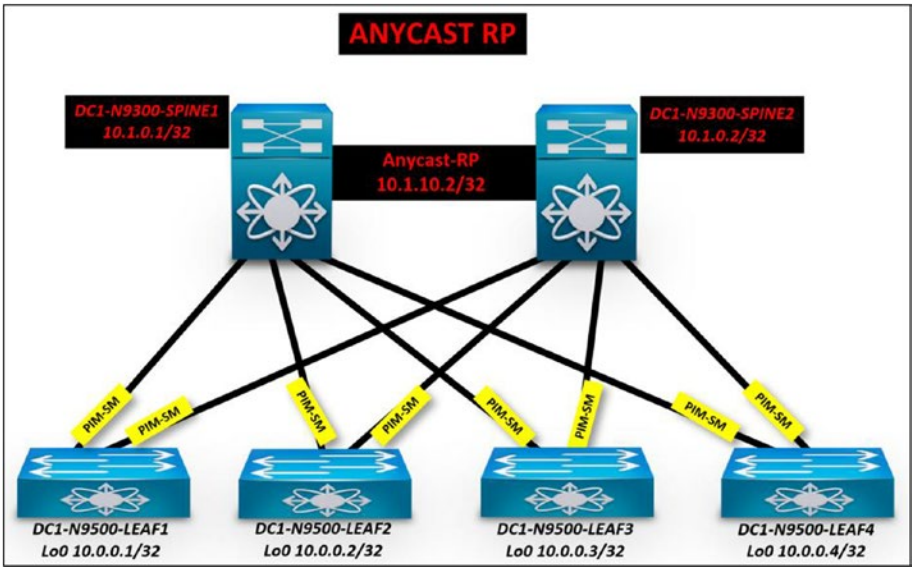


Figure 2-6. *Both DC1-N9300-Spine-01 and DC1-N9300-Spine-02 are designated as the active RPs. Spines are set up as anycast RPs*

Option 2: Phantom RP

Another approach for setting up your RP uses a **phantom RP**. It's called **phantom** since the RP address is non-existent from an interface standpoint. It's a method to make all multicast traffic reach a certain meeting point on the network. In L3 routing, the most specific network wins. Between 10.1.0.0/28 and 10.1.0.0/29, which network is more specific? 10.1.0.0/29. With this information, you can now configure the phantom RP.

The phantom RP address is non-existent on any interface, meaning that if I choose 10.1.10.2 as my RP address, I don't assign it to an interface and advertise it. Instead, I configure a loopback on each spine to have a network where the 10.1.10.2 address is part of it.

Each spine has a network with a specific subnet mask. However, Spine-01 is configured to have a smaller subnet mask, so it is preferred as the active RP. The leafs see a path to 10.1.10.2 being advertised via the most specific RP, which is Spine-01. If Spine-01 goes down, the RP becomes active on Spine-02 since the route to the RP network from Spine-01 is dropped, and the route to the RP network via Spine-02 is advertised. Let's configure a loopback for our phantom RP purpose.

For DC1-N9300-Spine-01, configure Lo1 to be 10.1.10.1/29, and advertise it in OSPF. For DC1-N9300-Spine-02, configure Lo1 to be 10.1.10.3/28 and advertise it in OSPF. With this in place, the route to the RP 10.1.10.2 address is preferred via DC1-N9300-Spine-01 since it's a more specific network (see Listing 2-11).

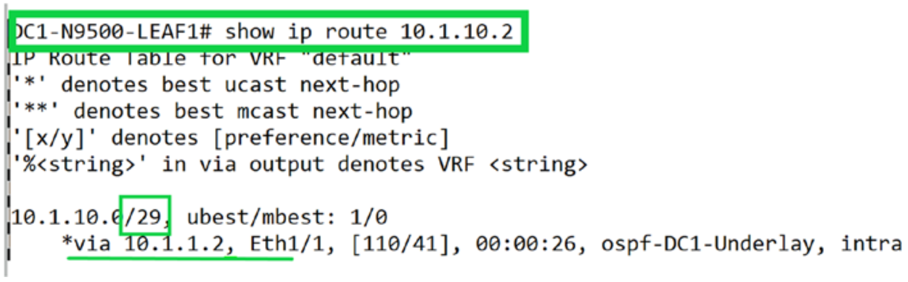
Listing 2-11. The complete multicast RP configuration with Phantom RP on a spine switch

DC1-N9300-SPINE1

```
feature pim

ip pim rp-address 10.1.10.2 group-list 224.0.0.0/16

int lo0
ip pim sparse-mode

int lo1
Description PHANTOM-RP
ip address 10.1.10.1/29
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point
ip pim sparse-mode

int eth1/1-4
ip pim sparse-mode

DC1-N9300-SPINE2
feature pim

ip pim rp-address 10.1.10.2 group-list 224.0.0.0/16
```

```
int lo0
ip pim sparse-mode

int lo1
Description PHANTOM-RP
ip address 10.1.10.3/28
ip router ospf DC1-Underlay area 0
ip ospf network point-to-point
ip pim sparse-mode

int eth1/1-4
ip pim sparse-mode

All LEAFS
feature pim
ip pim rp-address 10.1.10.2 group-list 224.0.0.0/16

int lo0
ip pim sparse-mode

int eth1/1-2
ip pim sparse-mode
```

It's time to trust but verify. Let's see how a leaf sees the route path to the phantom RP (see Figure 2-7).



```
DC1-N9500-LEAF1# show ip route 10.1.10.2
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.1.10.0/29, ubest/mbest: 1/0
    *via 10.1.1.2, Eth1/1, [110/41], 00:00:26, ospf-DC1-Underlay, intra
```

Figure 2-7. *The DC1-N9500-Leaf-01 route table shows the path to the phantom RP 10.1.10.2 is preferred over DC1-N9300-Spine-01 since the network address it belongs to is advertised in OSPF with a /29 subnet mask, smaller than /28 on DC1-N9300-Spine-02*

With the configuration in place, DC1-N9300-Spine-01 is elected as the preferred RP with a more specific subnet mask (see Figure 2-8).

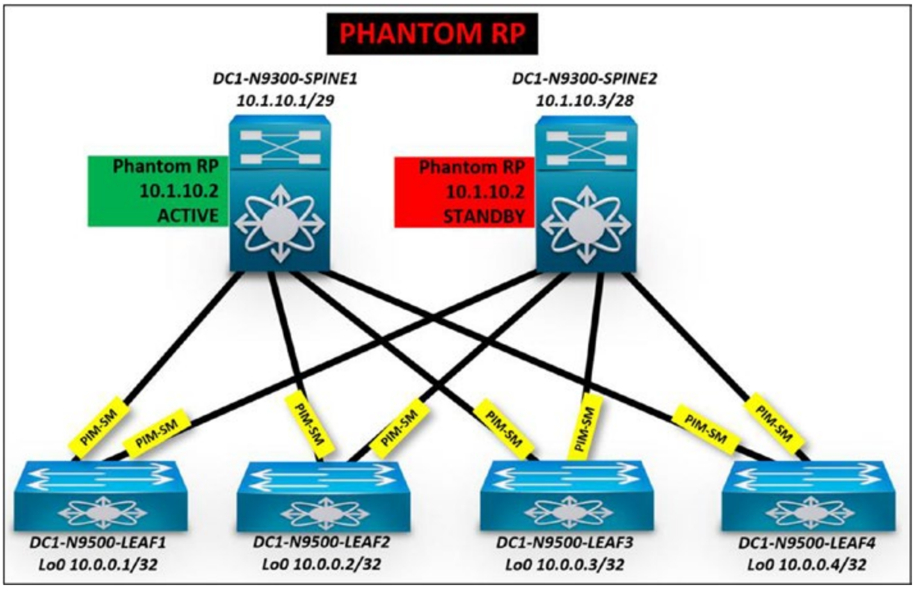


Figure 2-8. *DC1-N9300-Spine-01 is the preferred path to the RP 10.1.10.2 since the loopback is configured with a /29 mask. DC1-N9300-Spine-02, which is configured with a /28 mask, is on standby*

BGP L2VPN EVPN Address Family

Ladies and gentlemen, the time has come to discuss the BGP L2VPN EVPN address family (AFI). The magic of VXLAN happens with this protocol. Let's bring in a debate here. BGP is not just a routing protocol; it is an application that does many networking functions, including routing.

Each BGP AFI is an application under the BGP "operating system." Again, this is my opinion. I consider it the king of all network protocols, BGP. Thanks to BGP, we have the Internet.

Let's discuss the EVPN function in BGP. EVPN (Ethernet Virtual Private Network) is an encapsulation AFI in BGP that allows an L2 frame to be transported over an IP network. This means that a layer 2 frame is encapsulated into an L3 frame, which is then routed over an IP network.

I talked about one of the advantages of a VXLAN fabric. One of them is that since every uplink is configured at layer 3, you can effectively eliminate spanning-tree from your spine-and-leaf topology. With EVPN, service providers can offer L2 metro-Ethernet service circuits between distant locations. Think of it as a long Ethernet patch cord between locations that are miles apart.

EVPN functionality was brought into data centers to resolve the scalability limitations of campus networks. Spanning trees and broadcasting is no longer needed in data centers. VXLAN has changed networks forever.

Running the BGP L2VPN EVPN AFI has benefits. One benefit is that L2 learning is done over a control plane. Rather than relying on the data plane for information, you rely on the control plane, which provides a more flexible information distribution approach.

When I say *information*, I'm referring to MAC address learning and route learning. Rather than flooding my switch to learn where a MAC address is, I get this information from the BGP EVPN control plane as a route. Yes, a layer 2 route! Only the leaf that is locally adjacent to the end device needs to do the initial learning.

Once that’s done, it advertises its layer 2 address using EVPN. All the other leafs participating in EVPN get the route advertisement rather than flooding to find my end device over multiple layer 2 trunks, which prevents broadcast messages and ARPs. In summary, only one leaf needs to learn initially. The other leafs get the information from the source leaf. Just perfect.

Let’s get some insight into the BGP EVPN process for VXLAN. I’m going to specifically focus on EVPN. BGP has many functions. To completely understand BGP takes about a year.

In BGP, there are ways to classify your routes. You can set attributes to tag them to apply route policies and distinguish routes with community strings. There are many ways to do it. BGP VXLAN EVPN is no different. Each VNI has a designated route-target (RT). What’s a route-target? It’s a way to uniquely identify each route and manipulate its behavior in BGP.

Let’s say you have VLAN 10 with a VNI of 10010. The VNI resides in the VXLAN fabric with BGP AS #65500. You could “mark” all VXLAN traffic related to VLAN 10 by adding a 65500:10010 EVPN RT. What if you have another VXLAN fabric with AS #65501 that you intend to stretch VLAN 10 toward? Since the VLAN 10 VNI with a route-target is assigned on the first fabric, you can export the route-target from VLAN 10 in the first fabric, and then import the route-target to the second fabric. It’s very important to clearly understand this concept.

Remember, EVPN is all about route-target import and export. It doesn’t apply to fabric-to-fabric communication. It also applies to leaf-to-leaf communication. Yes, even if they are on the same fabric, you still need to import and export anything from a particular VNI. This is how you propagate the MAC information in EVPN.

To summarize, VXLAN EVPN allows L2 encapsulation and transport over an L3 network. You import and export route-targets to get this information to the local leaf, and advertise it from the local leaf out to the other leafs. In a single fabric, this is very straightforward. In the Nexus platform, you need to enable three features to bring VXLAN functionality on the device: `bgp`, `nv overlay`, and `vn-segment-vlan-based`.

Also, you need to enable it on the Nexus switch control plane for VXLAN by executing the `nv overlay evpn` command. Let's look at a configuration sample applied to the previous VLAN 10 scenario. Figure 2-9 shows a single fabric. Figure 2-10 shows different fabrics.

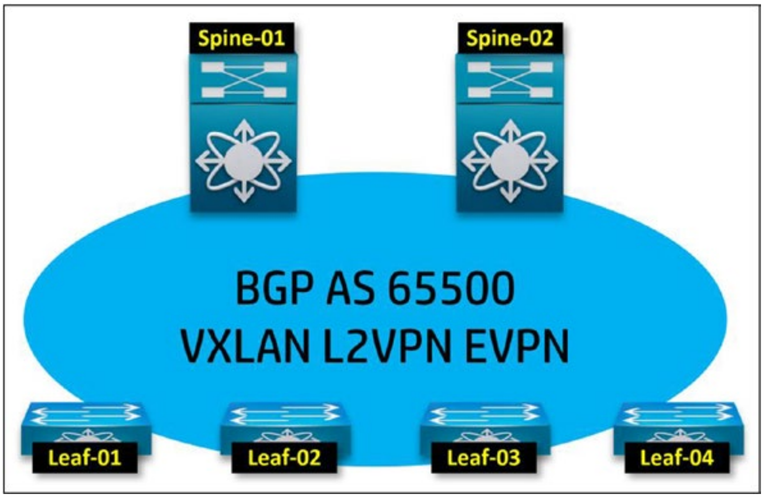


Figure 2-9. *A sample EVPN VXLAN configuration for VLAN 10*

Listing 2-12. Please refer to the L2VPN EVPN Address Family section. A sample configuration template to apply on leaf switches to enable BGP EVPN control-plane VXLAN

```
Apply to all leafs in the same fabric
feature nv overlay
feature bgp
feature vn-segment-vlan-based
nv overlay evpn

vlan 10
 vn-segment 10010
```

```
evpn
 vni 10010 l2
  rd auto
  route-target both 65500:10010
```

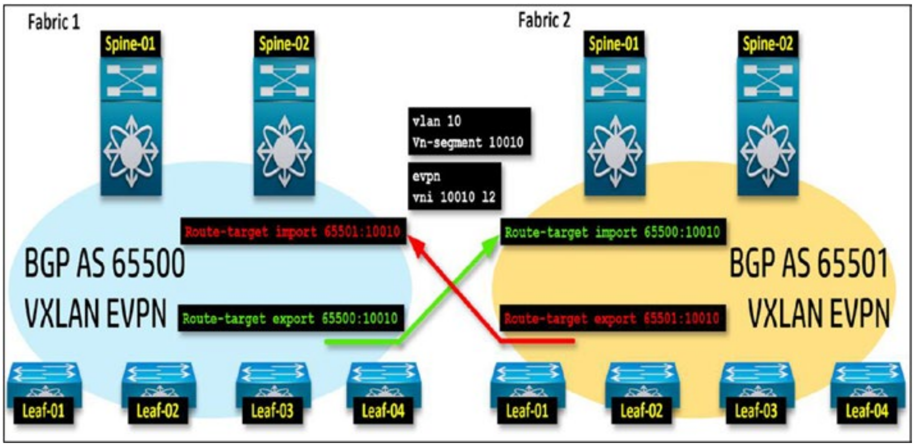


Figure 2-10. *Two separate VXLAN fabrics importing and exporting VLAN 10 RTs*

Listing 2-13. A sample EVPN route target import and export for L2VNIs. Please refer to the BGP L2VPN EVPN Address Family section for details

```
On all Fabric #1 Leaf
feature nv overlay
feature bgp
feature vn-segment-vlan-based
nv overlay evpn

vlan 10
 vn-segment 10010
evpn
 vni 10010 l2
  rd auto
  route-target both 65500:10010
  route-target import 65501:10010
```

```
On all Fabric #2 Leaf
feature nv overlay
feature bgp
feature vn-segment-vlan-based
nv overlay evpn

vlan 10
 vn-segment 10010
evpn
 vni 10010 l2
  rd auto
  route-target both 65501:10010
  route-target import 65500:10010
```

I used the `route-target both` command, which means that I'm both importing and exporting the particular RT. Once applied, you should see it in the following format.

```
Route-target import 65500:10010
Route-target export 65500:10010
```

BGP EVPN Route Reflectors

Let's talk about a very important item in your VXLAN BGP EVPN spine-and-leaf topology: route reflectors. Networks are meant to be designed using a simplistic approach. While BGP is a complex protocol, you can still apply techniques to make the architecture less difficult to administer and troubleshoot. Route reflectors are a great example of this.

The goal is to peer BGP across all leafs. With a route reflector, you can peer all your clients to a single BGP neighbor acting as the route reflector, and exchange BGP routes to other clients without having to peer to them. Let's assume you don't have a designated route reflector. What would BGP peering among four leafs look like?

Table 2-1. *A fabric without BGP route-reflectors would need an exponential amount of BGP neighbors. Please refer to the table below to see the peering between all leafs in the fabric*

Leaf-01	**Leaf-02**	**Leaf-03**	**Leaf-04**
PEERS	***PEERS***	***PEERS***	***PEERS***
Leaf-02	Leaf-01	Leaf-01	Leaf-01
Leaf-03	Leaf-03	Leaf-02	Leaf-02
Leaf-04	Leaf-04	Leaf-04	Leaf-03

There are three BGP peers for each leaf. You need to peer leafs between each other. While it may not seem too much of a problem, I'll tell you why it is a problem. In a production environment, you need redundancy, right? I hope you have redundancy! With this in mind, you need to provide dual connectivity between the leaf and the neighbor leaf. That means that you need double the physical connections (fiber, copper) per neighbor.

Table 2-2. *The previous table shows a single peering session between leafs. Since we need redundancy, adding a second path will increase significantly the amount of peering links*

Leaf-01	**Leaf-02**	**Leaf-03**	**Leaf-04**
Connections	***Connections***	***Connections***	***Connections***
Leaf-02 **x2**	Leaf-01	Leaf-01	Leaf-01
Leaf-03 **x2**	Leaf-03 **x2**	Leaf-02	Leaf-02
Leaf-04 **x2**	Leaf-04 **x2**	Leaf-04 **x2**	Leaf-03

In a four-leaf VXLAN fabric, without route reflectors, you end up with a total of 12 connections to peer four leafs and provide the required redundancy and aggregation for BGP. Your network has grown, and you need to add an extra leaf. Now it's 20 connections! Yes, if you add Leaf-05, you need double the connections to the other four leafs in the fabric.

Let's discuss how route reflectors remove scalability limitations and how it simplifies the architecture. In a VXLAN spine-and-leaf architecture, on the spine-and-leaf architecture, you designate the spine switches as route reflectors. Let's look at the same configuration if you add the spines as route reflectors.

Table 2-3. *With the benefit of using route reflectors we can drop the peering to only one or two route reflector neighbors for redundancy. A single peer to the route reflector will provide you BGP reachability to all your neighbor leafs without establishing peering between each other*

Leaf-01	Leaf-02	Leaf-03	Leaf-04	Leaf-05
PEERS	***PEERS***	***PEERS***	***PEERS***	***PEERS***
Spine-01 x1	Spine-01 x1	Spine-01 x1	Spine-01 x1	Spine-01 x1
Spine-02 x1	Spine-02 x1	Spine-02 x1	Spine-02 x1	Spine-02 x1

The difference is two connections: one to each spine. Our leafs can talk to each other in BGP thanks to the route reflectors. They reflect BGP routes to their clients. If Leaf-01 decides to advertise a route, it announces to the spines. The spines reflect the route information from Leaf-01 down to the other leafs. You can add more leafs, and the configuration remains the same. The leaf configuration for BGP is simpler because you only peer to two neighbors: Spine-01 and Spine-02.

Let's configure BGP EVPN on the spines to set them as route reflectors. Let's use the topology from Chapter 1 to perform the configurations (see Figure 2-11).

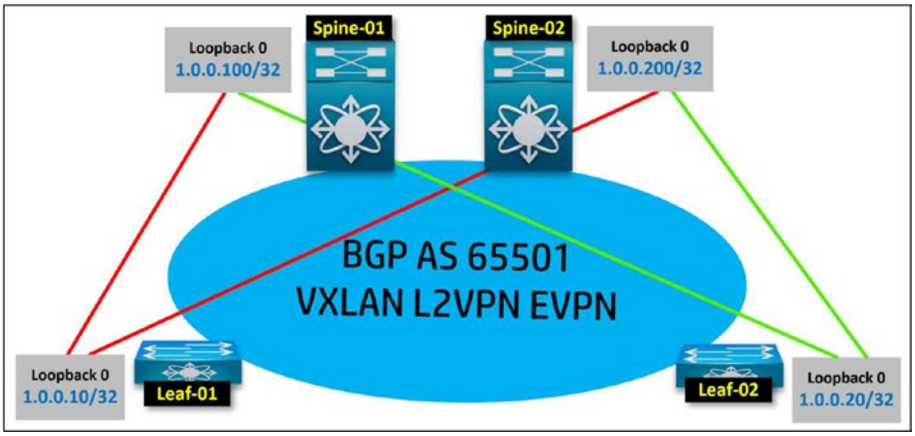


Figure 2-11. *The loopback interfaces are used as peering interfaces between leafs and spines*

Start the BGP configurations by enabling the BGP feature and running the BGP process. In the BGP process, specify the autonomous system number (AS).

```
feature bgp

router bgp 65501
```

Now let's configure the EVPN process in BGP for the spines as route reflectors. You need to enable two AFIs. The first is optional, but I recommend enabling IPv4 unicast AFI. The second is L2VPN EVPN. You then add your neighbor configs to the process. Your neighbors are the leafs' loopback0 interface addresses. Remember to configure the spines first. The BGP configuration is the same on both spines (see Listing 2-14).

Listing 2-14. A BGP EVPN spine switch configuration template

SPINE-01

```
feature bgp

router bgp 65501
log-neighbor-changes
```

```
address-family ipv4 unicast
address-family l2vpn evpn

neighbor 1.0.0.10 (LEAF-01)
 remote-as 65501
 update-source lo0
 address-family l2vpn evpn
 send-community
 send-community extended
 route-reflector-client

neighbor 1.0.0.20 (LEAF-02)
 remote-as 65501
 update-source lo0
 address-family l2vpn evpn
 send-community
 send-community extended
 route-reflector-client
```

Listing 2-14. A VXLAN BGP EVPN spine configuration example. The bold shows the enabled the route reflector function and identifies neighbor leafs as route reflector clients.

In the previous configuration, you need to add the same configuration per neighbor (leaf). Can you imagine having eight or nine leafs? That means that you end up with a very long BGP configuration, adding the same information over and over. There is a way to create a template and assign it to as many neighbors as you need. You end up adding the same configuration over and over.

Let me show you how to create a peer template and accomplish it. Let's create the template to point the neighbors to inherit it (see Listing 2-15).

Listing 2-15. In this configuration, a template with the configuration parameters was filled then applied to all four peers or leafs, making the configuration cleaner since you don't need to repeat the same configuration for each leafs

ALL SPINES

```
feature bgp
feature nv overlay
nv overlay evpn

router bgp 65501
log-neighbor-changes

address-family ipv4 unicast
address-family l2vpn evpn

template peer TO_LEAFS
 remote-as 65501
 update-source lo0
  address-family l2vpn evpn
  send-community
  send-community extended
  route-reflector-client

neighbor 1.0.0.10
 inherit peer TO_LEAFS
neighbor 1.0.0.20
 inherit peer TO_LEAFS
neighbor 1.0.0.30
 inherit peer TO_LEAFS
neighbor 1.0.0.40
 inherit peer TO_LEAFS
```

LEAF EVPN Configuration

Now it's time to configure the EVPN-specific configuration on the leaf. There's more than the BGP configuration. The leaf is running the show. They are in charge of layer 2 and layer 3 encapsulation/decapsulation using EVPN and properly routing all traffic in the fabric. The configuration parameters are as follows. You need to assign the VXLAN network identifier (VNI) per VLAN. Let's assume you have VLAN 10, VLAN 20, and VLAN 30 and assign each VLAN its respective VNI. Let's configure the EVPN route-target import and export and BGP EVPN to point the leaf to the peer spines (see Listing 2-16).

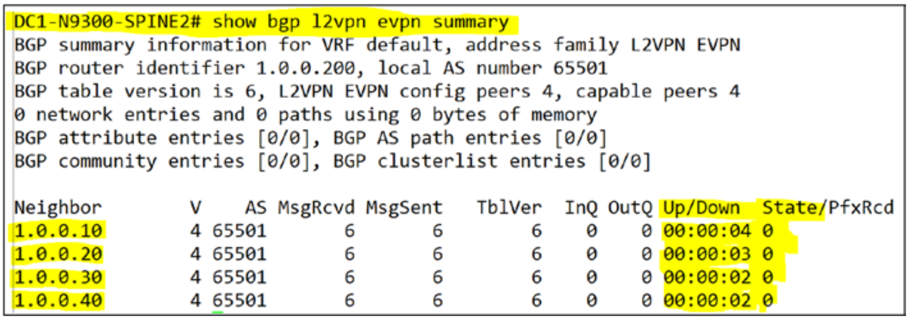
Listing 2-16. A sample BGP EVPN leaf configuration template

```
ALL LEAFS
feature bgp
feature nv overlay
feature vn-segment-vlan-based
feature nv overlay
nv overlay evpn

vlan 10
vn-segment 10010
vlan 20
vn-segment 10020

evpn
 vni 10010 l2
 rd auto
 route-target both 65501:10
 vni 10020 l2
 rd auto
 route-target both 65501:20
```

```
router bgp 65501
 log-neighbor-changes

 address-family ipv4 unicast
 address-family l2vpn evpn

template peer TO_SPINES
 remote-as 65501
 update-source lo0
  address-family l2vpn evpn
  send-community
  send-community extended

neighbor 1.0.0.100
 inherit peer TO_SPINES
neighbor 1.0.0.200
 inherit peer TO_SPINES
```

Listing 2-16. Each VLAN has the VNI assigned (vn-segment #). Add the RT to each VNI under EVPN and enable the BGP EVPN control plane.

If your underlay is correctly configured and the configurations on your fabric spines and leafs have been applied, you should see active BGP L2VPN EVPN adjacency. Execute the following command on either the spines or the leafs (see Figure 2-12).



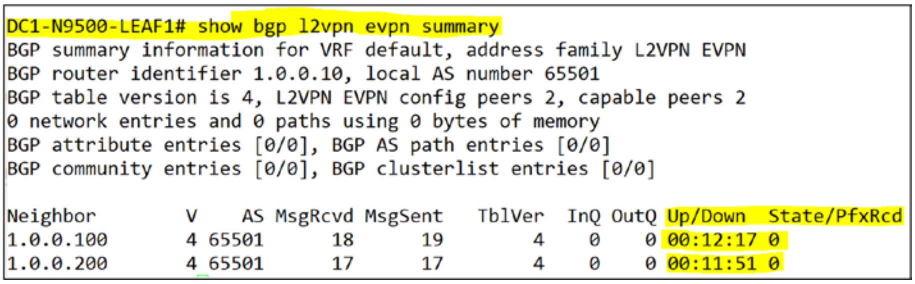
```
DC1-N9300-SPINE2# show bgp l2vpn evpn summary
BGP summary information for VRF default, address family L2VPN EVPN
BGP router identifier 1.0.0.200, local AS number 65501
BGP table version is 6, L2VPN EVPN config peers 4, capable peers 4
0 network entries and 0 paths using 0 bytes of memory
BGP attribute entries [0/0], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [0/0]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
1.0.0.10        4 65501       6       6        6    0    0 00:00:04 0
1.0.0.20        4 65501       6       6        6    0    0 00:00:03 0
1.0.0.30        4 65501       6       6        6    0    0 00:00:02 0
1.0.0.40        4 65501       6       6        6    0    0 00:00:02 0
```

Figure 2-12. *DC1-N9300-Spine-02 confirms successful BGP EVPN peering between the spine and all the leafs in the fabric. Showing a timer on the up/down status. State or prefix received 0 means that while I don't have EVPN advertisements yet, I'm fully adjacent to my BGP peer*

`show bgp l2vpn evpn summary`

Let's confirm the same information on the leaf. You should see a BGP peering session to each spine in the VXLAN fabric (see Figure 2-13).

`show bgp l2vpn evpn summary`



```
DC1-N9500-LEAF1# show bgp l2vpn evpn summary
BGP summary information for VRF default, address family L2VPN EVPN
BGP router identifier 1.0.0.10, local AS number 65501
BGP table version is 4, L2VPN EVPN config peers 2, capable peers 2
0 network entries and 0 paths using 0 bytes of memory
BGP attribute entries [0/0], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [0/0]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
1.0.0.100       4 65501      18      19        4    0    0 00:12:17 0
1.0.0.200       4 65501      17      17        4    0    0 00:11:51 0
```

Figure 2-13. *DC1-N9500-Leaf-01 confirms successful BGP EVPN peering from the leaf to the spine. You can confirm successful peering to DC1-N9300-Spine-01 and DC1-N9300-Spine-02*

Once you start advertising VXLAN information on the fabric, you can confirm the EVPN route information on any leaf using the `show bgp l2vpn evpn` command. It confirms any EVPN type routes received by the leaf.

Since I haven't gone into the VXLAN virtualization aspect, you haven't seen any routes yet. Don't worry. It's time to move on to Chapter 3 to do VXLAN encapsulation!

CHAPTER 3

Encapsulating Layer 2 over Layer 3 with VXLAN

You have reached the point of working with VXLAN and the VTEPs. Our initial fabric setup is complete. Underlay, multicast, and BGP EVPN are configured and ready for VXLAN implementation. What should you expect out of this chapter? You should expect to see layer 2 traffic encapsulated in BGP and advertised in the EVPN table. Now that you have everything set up to support VXLAN, let's discuss and configure the network virtual interface (NVE).

VTEPs and NVE

You have reached the point of working with VXLAN and the VTEPs. The initial fabric setup is complete. You now need to deploy the NVE. The NVE interface encapsulates and decapsulates layer 2 traffic into a VXLAN L3 frame. The moment your switch receives a packet, it is encapsulated via the NVE interface. Not only does the interface encap/decap, it also does part of the BUM traffic advertisements. If a leaf receives traffic destined

R. Cardona, *The Fast-Track Guide to VXLAN BGP EVPN Fabrics*,
https://doi.org/10.1007/978-1-4842-6930-5_3

for the VNI 10010, the NVE sees the traffic coming in, encapsulates it, then looks under its config for the assigned multicast group and advertises the MAC information via a multicast message to the assigned group. This multicast message has the VTEP interface as a source address, which is a dedicated loopback interface to perform the VXLAN traffic transport over the fabric.

The VTEP interface is announced in BGP EVPN. When all leafs have their VTEP interface address announced, they peer their NVE to exchange VXLAN traffic. It's great to know the basics behind this communication, but you'll understand it more once you configure the NVE and test.

Let's use the same fabric topology: two spines and two leafs. You are going to create the VTEP interface on each leaf and configure the NVE interface (see Figure 3-1).

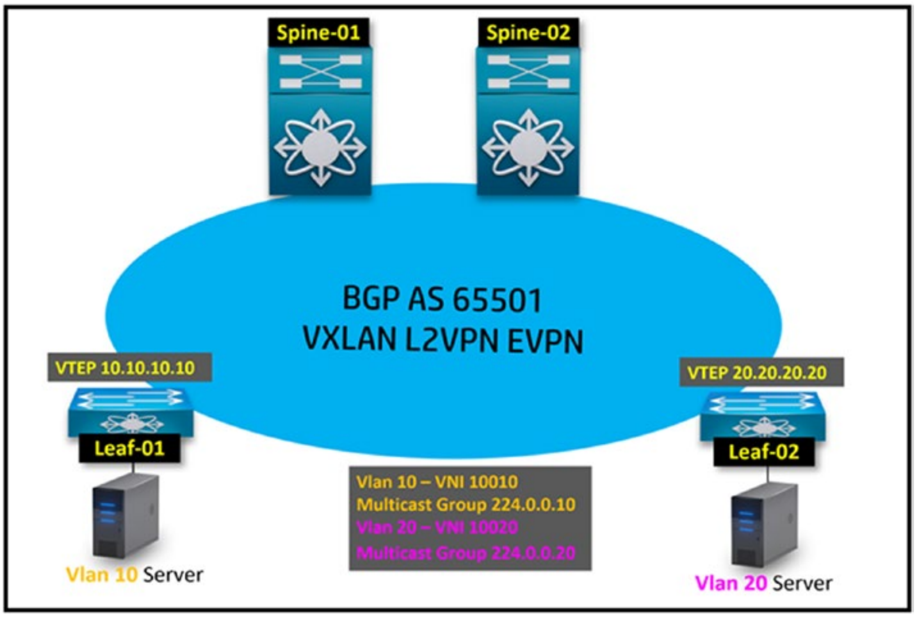


Figure 3-1. *Values for both Leaf-01 and Leaf-02 VTEP interface IP addresses. The required VXLAN information for VLAN 10 and VLAN 20 to enable EVPN encapsulation and BUM traffic*

Table 3-1. *A simple vlan to vni reference with the allocated multicast group address for BUM traffic*

LEAF Loopback1	VLAN ID	VNI	Multicast Group
Leaf-01 – 10.10.10.10	10	10010	224.1.1.10
Leaf-02 – 20.20.20.20	20	10020	224.1.1.20

You have the information needed to configure your VXLAN requirements for VLAN 10 and VLAN 20. Once you create a template configuration, it applies to all leafs except the loopback1 interface IP, which needs to be unique on each VTEP (leaf). Once you've validated your configurations, paste the config across all leafs. Yes, the configuration is the same for all leafs, which is an advantage in VXLAN.

Let's configure everything needed to let the fabric encapsulate and decapsulate VXLAN with BGP EVPN. You are configuring all the BGP EVPN aspects and advertising the new loopback1 interface in the BGP IPv4 unicast address family. Also make sure that the spines can perform route reflector functions for the IPv4 unicast AFI (see Listing 3-1).

Listing 3-1. Global configuration on spines

```
router bgp 65501
log-neighbor-changes

address-family ipv4 unicast
address-family l2vpn evpn

template peer TO_LEAFS
 remote-as 65501
 update-source lo0
  address-family ipv4 unicast
  send-community
  send-community extended
  route-reflector-client
  address-family l2vpn evpn
  send-community
  send-community extended
  route-reflector-client
```

Leaf-01 SPECIFIC CONFIG

```
feature bgp
feature vn-segment-vlan-based
feature nv overlay
nv overlay evpn

int lo1
ip address 10.10.10.10/32
ip pim sparse-mode

router bgp 65501
 log-neighbor-changes
 address-family ipv4 unicast
 network 10.10.10.10/32
 address-family l2vpn evpn

template peer TO_SPINES
 remote-as 65501
 update-source lo0
  address-family ipv4 unicast
  send-community
  send-community extended
  address-family l2vpn evpn
  send-community
  send-community extended

neighbor 1.0.0.100
```

```
 inherit peer TO_SPINES
neighbor 1.0.0.200
 inherit peer TO_SPINES
```

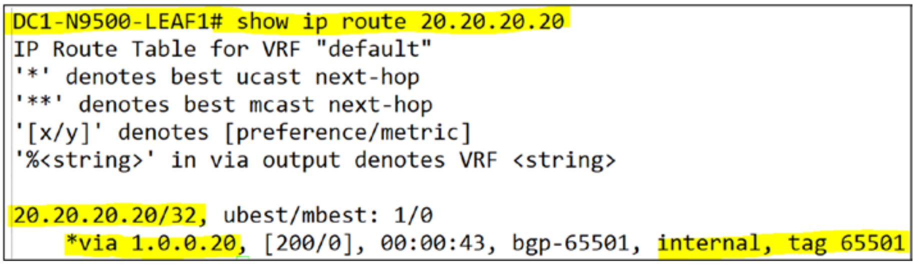
Leaf-02 SPECIFIC CONFIG

```
feature bgp
feature vn-segment-vlan-based
feature nv overlay
nv overlay evpn

int lo1
ip address 20.20.20.20/32
ip pim sparse-mode

router bgp 65501
 log-neighbor-changes
 address-family ipv4 unicast
 network 20.20.20.20/32
 address-family l2vpn evpn

template peer TO_SPINES
 remote-as 65501
 update-source lo0
  address-family ipv4 unicast
  send-community
  send-community extended
  address-family l2vpn evpn
  send-community
  send-community extended

neighbor 1.0.0.100
 inherit peer TO_SPINES
neighbor 1.0.0.200
 inherit peer TO_SPINES
```

GLOBAL CONFIGURATION ON ALL LEAFS

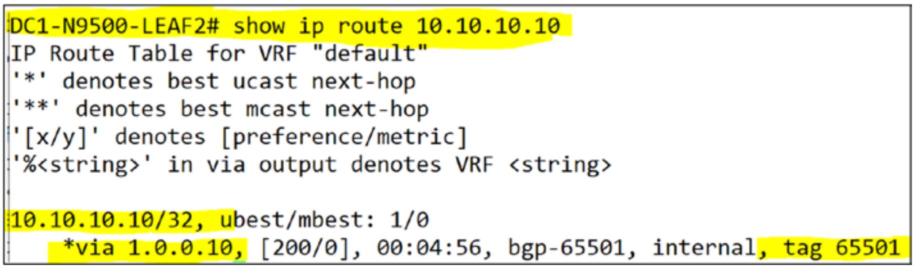
```
vlan 10
vn-segment 10010
vlan 20
vn-segment 10020

interface nve1
 no shutdown
 source-interface lo1
 host-reachability protocol bgp
 member vni 10010
 mcast-group 224.1.1.10
 member vni 10020
 mcast-group 224.1.1.20
```

Once you finish applying the configurations to the leafs, verify the NVE's status and whether NVE peer adjacency has formed between Leaf-01 and Leaf-02. First, let's verify if I can see the VTEP interface IP from Leaf-02 advertised in Leaf-01 via BGP.



```
DC1-N9500-LEAF1# show ip route 20.20.20.20
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

20.20.20.20/32, ubest/mbest: 1/0
    *via 1.0.0.20, [200/0], 00:00:43, bgp-65501, internal, tag 65501
```

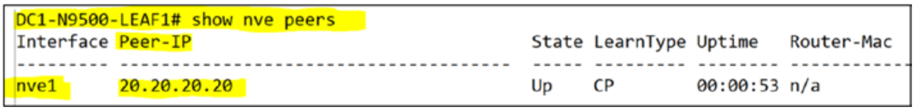
Leaf-01 learned the VTEP address of Leaf-02 via BGP. I assume that Leaf-02 should see the VTEP interface IP of Leaf-01 in the route table as well. Let's verify!



```
DC1-N9500-LEAF2# show ip route 10.10.10.10
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.10.10.10/32, ubest/mbest: 1/0
    *via 1.0.0.10, [200/0], 00:04:56, bgp-65501, internal, tag 65501
```

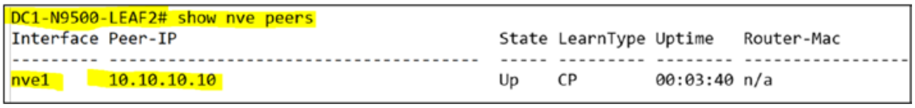
By confirming this information, it is safe to assume that you should have NVE peer reachability between Leaf-01 and Leaf-02. Let's confirm if this is the case in both leafs.

Leaf-01



```
DC1-N9500-LEAF1# show nve peers
Interface Peer-IP                                 State LearnType Uptime   Router-Mac
--------- --------------------------------------- ----- --------- -------- -----------
nve1      20.20.20.20                             Up    CP        00:00:53 n/a
```

Leaf-02



```
DC1-N9500-LEAF2# show nve peers
Interface Peer-IP                                 State LearnType Uptime   Router-Mac
--------- --------------------------------------- ----- --------- -------- -----------------
nve1      10.10.10.10                             Up    CP        00:03:40 n/a
```

Looking very good! One note on the output command: the router MAC shows n/a, which should be populated once you finish performing all the required anycast gateway and tenant VRF configurations. Don't worry!

Note If no peers are active when performing the NVE peer verification, you might need to generate end device traffic in the fabric so the NVE can become active and perform its peering to all other leafs. Continue with your fabric configuration. They should be up once your clients try to communicate.

EVPN Route-Targets

Let's discuss the EVPN routing aspect. In VXLAN, you transport layer 2 and layer 3. In VXLAN, you move layer 2 frames over a routed network and move L3 networks distributed with EVPN. When you work with BGP and VRF (virtual routing and forwarding), you assign a tag or a value to a specific routing instance by assigning a route-target.

Let me explain route-targets in plain English. There is Bob, Maria, and Tom. Each one decided to fly to another city, but they want to remain in the same country. You must make sure the airplanes don't get lost. You also want to make sure each person returns to where they came from. To accomplish this task, you need to assign a number to Bob's plane, Maria's plane, and Tom's plane. Also, you need to assign a number to the country they all live in. Knowing the requirements, let's assign the values (see Table 3-2).

Table 3-2. *Associating EVPN Route Targets. AS:ID Example: Country:Plane or 65501:10*

NAME	Country	Country#	Plane #
Bob	USA	65501	10
Maria	USA	65501	20
Tom	USA	65501	30

Okay, you have the information. Let's make sure Bob gets on plane 10. While he is traveling, you can track his plane on the radar to make sure he stays on route in the United States. Let's assign his target as **country#:plane#** or **65501:10**. Bob takes off from the same airport, **export 65501:10** so Bob can take off from the origin airport and let's **import 65501:10** on the destination airport so Bob can land in. The same approach goes for Maria and Tom. Let's export Maria, **65501:20** and once it gets to the destination let's import her **65501:20**. Finally, with Tom, let's export **65501:30** and then import **65501:30**.

Let's convert this analogy to VXLAN EVPN. The fabric (country) has AS number (country number) 65501. There is VLAN 10 (Bob), VLAN 20 (Maria), and VLAN 30 (Tom). Let's send EVPN paths from Leaf-01 (the origin airport) on the fabric for VLAN 10 devices with `the route-target export 65501:10` command. A device incoming from Leaf-01 in VLAN 10 needs to communicate to a device on Leaf-02 in VLAN 10. Let's import that information on Leaf-02 with the `route-target import 65501:10` command. Let's configure EVPN with this information (see Listing 3-2).

Listing 3-2. EVPN L2VNI route target import and export configurations

```
evpn
vni 10010 l2
rd auto
route-target import 65501:10
route-target export 65501:10

vni 10020 l2
rd auto
route-target import 65501:20
route-target export 65501:20

vni 10030 l2
rd auto
route-target import 65501:30
route-target export 65501:30
```

You need to apply this information across all leafs in the fabric. Why? Since **you receive and send VLAN traffic in the fabric, you need to export and import the same EVPN VNI information on all leafs**.

L2VNI and L3VNI

Let's discuss L2VNI and L3VNI. The L2VNI or layer 2 VXLAN identifier is the tag that you add to a VXLAN frame to classify this traffic on a VLAN. You want to transport L2 over L3 with EVPN. Inside the L2 frame, there is the VLAN 802.1q tag. **VXLAN doesn't forward L2 natively, so you encapsulate it in layer3 and tag it with an identification that relates the VXLAN L3 frame to the VLAN ID. This is the L2VNI.**

How do you perform a VXLAN VNI to a VLAN mapping? You turn on the `vn-segment-vlan-based` feature on Nexus switches. Then you can add the VN segment or L2VNI to each respective VLAN. In VXLAN, the VNI range is from 1 to 16777215. This is where VXLAN shines in multitenancy. By having such a large VXLAN ID range, you can run multiple virtual networks and have many IDs to assign rather than the VLAN limitation of 4094. Let's assign the L2VNI to each VLAN (see Listing 3-3).

Listing 3-3. Sample configuration to allocate a VNI (VXLAN ID) to a particular vlan

```
PASTE ON ALL LEAFS
feature vn-segment-vlan-based

Vlan 10
Vn-segment 100010
Vlan 20
Vn-segment 100020
Vlan 30
Vn-segment 100030
```

16,777,215 VNIs = VXLAN Networks?

I am frequently asked this question. There is a limit of 4094 usable VLAN IDs in the L2 world. If I need to map a VLAN to a VNI, I have one-to-one mapping. If you have 16 million possible VNIs and need to map to a VLAN, are you limited to 4094 VLANs? Yes and no. VXLAN is meant to be a software-defined network-based solution for data centers with multitenancy requirements. Every tenant is a virtual network run on top of the fabric as it moves on a layer 3 underlay. **The limitation of 4094 VLAN IDs to VXLAN VNIs is still present for each leaf!** If I need to assign VLAN 10 to three different networks on the same leaf, this is a limitation. However, chances are that if you are running a multitenant environment, you have more than two leafs. Let me show you how to leverage multitenancy and still use the same VLAN IDs for different customers.

Let's assume three customers or three tenants need to use VLAN 50 and 60. You don't want customers to be able to see other customers' traffic. Table 3-3 shows how VXLAN does the isolation in EVPN while still giving the customers the same VLAN IDs.

Table 3-3. *Tenants assigned an unique L2VNI to separate traffic destined to the same vlan id*

Tenant Name	VLAN ID	VXLAN L2VNI	Assigned Rack
Paramount	50	**152387**	Rack 1 and 3
Universal	50	**176322**	Rack 2 and 4
Pixar	50	**124435**	Rack 5 and 6
Paramount	60	**115243**	Rack 1 and 4
Universal	60	**104244**	Rack 2 and 5
Pixar	60	**161124**	Rack 3 and 6

As a cloud data center host provider, I'm not confining any tenants to a specific leaf. I can have workloads stretched across my compute infrastructure.

Let's take a closer look at the spine-and-leaf fabric diagram in Figure 3-2, which shows how to allow multiple tenants share the same VLAN while isolating them from each other.

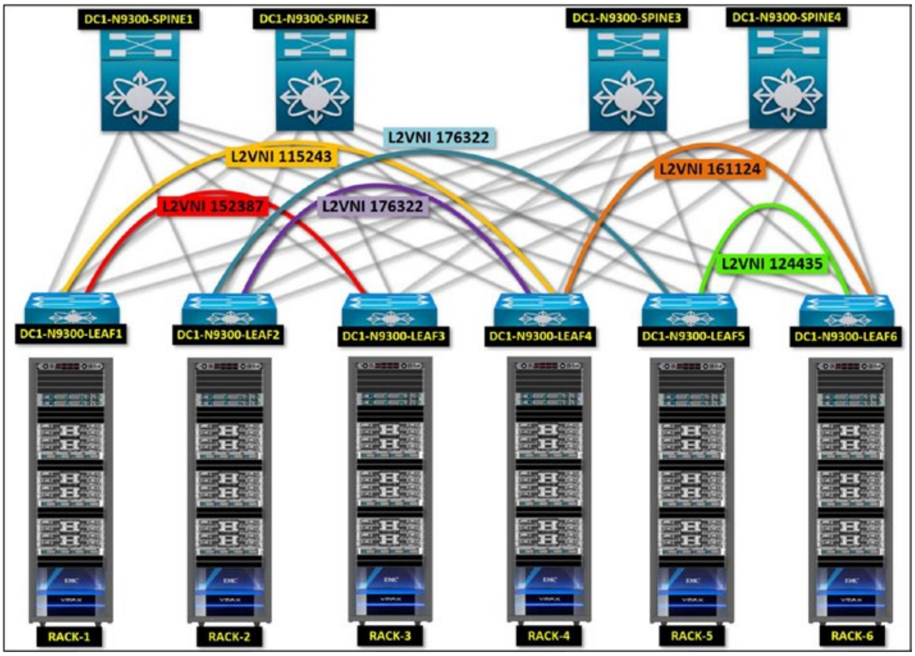


Figure 3-2. *A multitenant VXLAN fabric using same VLAN IDs for different customers*

As you can see in the diagram, three tenants are running and sharing the same VLAN IDs. The same VLAN IDs are allocated to different leafs, so you can assign them to each tenant separately. Since there are 16 million possible VNIs, I've assigned one unique VNI per customer for each VLAN. If customer A has assigned POD1 and POD3 hosted in racks 1 and 3, that means that I am allocating the VLAN 50 VNI for the first customer on Leafs-01 and 03. The same process applies to the other customers based on their assigned POD. I can still have other customers active on the same Leaf-01 and 03; however, the only one entitled to VLAN 50 on those leafs is the first customer.

If the requirement is to have two customers on the same leaf, both using the same VLAN ID, I could perform some VLAN translations or mappings for the second customer. The switch can translate the traffic from being tagged with VLAN 50 to VLAN 55, for example.

Future capability to assign the same VLAN ID to multiple VNIs on the same switch is on the roadmap. If it is released, it brings even more flexibility to the VXLAN multitenancy advantages.

Tenant VRF Role in VXLAN

Multitenancy is one of the biggest benefits of a VXLAN BGP EVPN fabric. How do you create a tenant (virtual) network to run inside a spine-and-leaf architecture? In the BGP EVPN fabric, the spine switches and the leaf switches are hypervisors, or the hardware that takes care of transporting all your virtual network traffic inside the fabric. With a VRF, you can logically create virtual networks inside the same fabric. Could you create multiple VRFs inside the same campus core and separate different networks? Yes, you could, but the layer 2 aspect is still present. Traffic is forwarded at L2, limiting the potential for multitenancy since it is tied to 4094 VLANs. In VXLAN, there's virtually no limitation—16 million possible VNIs to be exact. Also, the traffic is encapsulated at layer 3 and is forwarded independently with exclusive tunnels inside the VRF (L3VNIs).

VXLAN Fabric Tenant Routing with L3VNIs

You now understand how layer 2 encapsulation and forwarding in a VXLAN fabric works. Let's now discuss how a VXLAN fabric allows communication between different VLANs and leafs. In a campus topology, inter-VLAN routing happens at the active core layer. All VLAN-to-VLAN traffic "meets" at the core layer. This is not the case in VXLAN. With VXLAN in BGP EVPN, you can leverage a feature called **anycast gateways**.

Anycast gateways are covered in Chapter 4, but to summarize, an anycast gateway is the switch virtual interface (SVI) for a VLAN, but different from traditional campus design. This SVI can be duplicated across all leafs. Rather than having one active core instance, all leafs are technically core switches. If a request from a device on a source VLAN needs to communicate to a device on a destination VLAN, instead of sending the request traffic to the core switch, the local leaf to that source device performs the routing function.

Let's assume you have VLAN10-PC1 and VLAN20-PC2. PC1 on VLAN 10 needs to communicate with PC2 on VLAN 20. VLAN10-PC1 is attached to Leaf-01. VLAN20-PC2 is attached to Leaf-05.

Once Leaf-01 sees the request from PC1, it verifies that its route table is assigned to the tenant VRF for VLAN 10 and identifies that PC2 is located on Leaf-05. Here's where L3VNI operates. To send traffic from Leaf-01 to Leaf-05, the tenant VRF where both VLANs are located needs to have a designated L3VNI or "tunnel." Once the traffic is forwarded from the source leaf to the destination leaf, it "hops" onto the L3VNI interface (tunnel) and reaches its destination leaf. Imagine having a site-to-site VPN tunnel between leafs to send and receive inter-VLAN traffic. Every tenant in the fabric would need a unique L3VNI interface assigned. Let's review the L3VNI requirements. Figure 3-3 is a sample logical diagram.

1. Create a VLAN for L3VNI and assign a VNI to it.
2. Assign L3VNI to the tenant VRF.
3. Create an L3VNI SVI for VXLAN traffic forwarding.
4. Associate L3VNI to the VRF in the NVE interface.

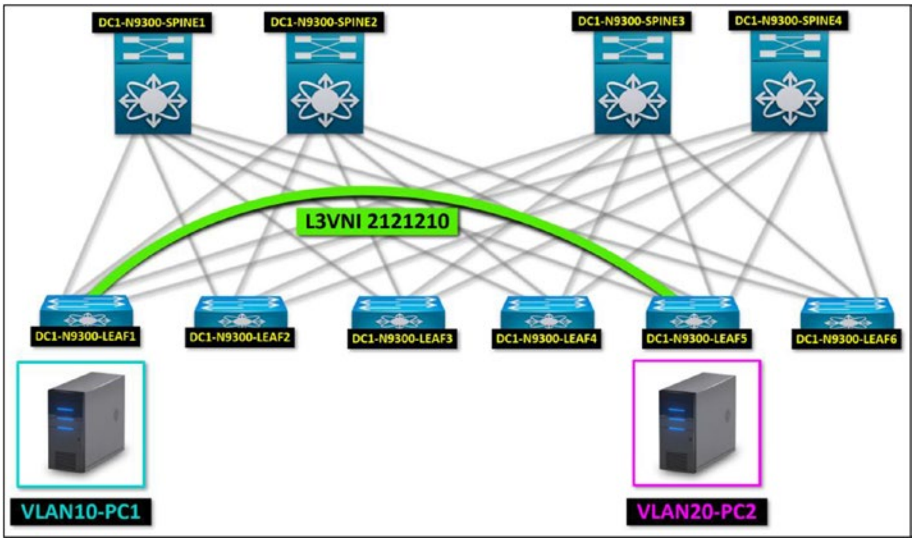


Figure 3-3. *An L3VNI in a VXLAN fabric carries all inter-VLAN and route redistribution in EVPN between leafs*

L3VNI has a designated virtual interface in which traffic is sent out to a VTEP (leaf). In Figure 3-3, PC1 sends its request to its default gateway on VLAN 10 to reach PC2. The anycast gateway configured for VLAN 10 forwards this traffic to L3VNI since the EVPN type-3 route is for PC2 on VLAN 20 in Leaf-05. Both Leaf-01 and Leaf-05 already have L3VNI active on the tenant. Leaf-01 then forward its request to Leaf-05 via L3VNI. Finally, it reaches Leaf-05 from L3VNI. The following is a sample L3VNI configuration breakdown. This configuration is applied across all leafs.

1. Create and assign a VLAN ID and VNI to L3VNI.

```
vlan 2210
name L3VNI-Tenant-1
vn-segment 2121210
```

2. Assign L3VNI to the tenant VRF.

```
vrf context Tenant-1
 vni 2121210
 rd auto
 address-family ipv4 unicast
  route-target both auto
  route-target both auto evpn
```

3. Create an L3VNI SVI for VXLAN traffic forwarding.

```
interface vlan 2210
 vrf member Tenant-1
 ip forward
 mtu 9000
 no shut
```

4. Associate L3VNI to the VRF in the NVE interface.

```
interface nve 1
 member vni 2121210 associate-vrf
```

Now that the configuration is in place, let's look at our fabric status, particularly the tenant VRF route table and the EVPN table, so you can see where L3VNI informs the destination leaf about a source device on another leaf.

VLAN 10 PC has a 10.10.0.10 IP, and VLAN20-PC has a 10.20.0.10 IP. Let's verify the EVPN MAC advertisements and see what's learned via L3VNI.

VXLAN EVPN Advertisements on Leaf-01

```
DC1-N9500-LEAF1# show bgp l2vpn evpn
BGP routing table information for VRF default, address family
L2VPN EVPN
```

```
BGP table version is 21, Local Router ID is 1.0.0.10

Network        Next Hop        Metric     LocPrf     Weight Path

Route Distinguisher: 1.0.0.10:32777    (L2VNI 10010)

*>l[2]:[0]:[0]:[48]:[0050.7966.6807]:[0]:[0.0.0.0]/216
               10.10.10.10                100       32768 i
*>l[2]:[0]:[0]:[48]:[0050.7966.6807]:[32]:[10.10.0.10]/272
               10.10.10.10                100       32768 i

Route Distinguisher: 1.0.0.10:32787    (L2VNI 10020)

*>i[2]:[0]:[0]:[48]:[0050.7966.6808]:[0]:[0.0.0.0]/216
               50.50.50.50                100           0 i
*>i[2]:[0]:[0]:[48]:[0050.7966.6808]:[32]:[10.20.0.10]/272
               50.50.50.50                100           0 i

Route Distinguisher: 1.0.0.40:32787

*>i[2]:[0]:[0]:[48]:[0050.7966.6808]:[0]:[0.0.0.0]/216
               50.50.50.50                100           0 i
* i            50.50.50.50                100           0 i
*>i[2]:[0]:[0]:[48]:[0050.7966.6808]:[32]:[10.20.0.10]/272
               50.50.50.50                100           0 i
* i            50.50.50.50                100           0 i

Route Distinguisher: 1.0.0.10:3    (L3VNI 2121210)

*>i[2]:[0]:[0]:[48]:[0050.7966.6808]:[32]:[10.20.0.10]/272
               50.50.50.50                100           0 i
```

Based on the `show bgp l2vpn evpn` output on Leaf-01, L3VNI 2121210 has advertised an EVPN L2 route for VLAN20-PC1 IP 10.20.0.10 from the VTEP 50.50.50.50 address, which belongs to Leaf-05. From a local EVPN route, VNI 10010 is assigned to VLAN 10 announcing the 10.10.0.10 IP, which belongs to VLAN10-PC1 from the local Leaf-01 VTEP of 10.10.10.10.

Now that you validated on Leaf-01 this information. You run the same command from Leaf-05. You should see the information inverted. Leaf-05 locally announcing to EVPN the L2VNI 10020 route for VLAN20-PC1 with 10.20.0.10 IP and from L3VNI the incoming L2 route for the VLAN10-PC1 10.10.0.10 IP via Leaf-01 VTEP. Let's confirm the information.

VXLAN EVPN Advertisements on Leaf-05

```
DC1-N9500-LEAF5# show bgp l2vpn evpn
BGP routing table information for VRF default, address family
L2VPN EVPN
BGP table version is 35, Local Router ID is 1.0.0.40

   Network      Next Hop     Metric     LocPrf     Weight Path

Route Distinguisher: 1.0.0.10:32777

* i[2]:[0]:[0]:[48]:[0050.7966.6807]:[0]:[0.0.0.0]/216
               10.10.10.10                100          0 i
*>i            10.10.10.10                100          0 i
* i[2]:[0]:[0]:[48]:[0050.7966.6807]:[32]:[10.10.0.10]/272
               10.10.10.10                100          0 i
*>i     10.10.10.10                       100          0 i

Route Distinguisher: 1.0.0.40:32777    (L2VNI 10010)

*>i[2]:[0]:[0]:[48]:[0050.7966.6807]:[0]:[0.0.0.0]/216
               10.10.10.10                100          0 i
*>i[2]:[0]:[0]:[48]:[0050.7966.6807]:[32]:[10.10.0.10]/272
               10.10.10.10                100          0 i
```

```
Route Distinguisher: 1.0.0.40:32787    (L2VNI 10020)

*>l[2]:[0]:[0]:[48]:[0050.7966.6808]:[0]:[0.0.0.0]/216
               50.50.50.50                100      32768 i
*>l[2]:[0]:[0]:[48]:[0050.7966.6808]:[32]:[10.20.0.10]/272
               50.50.50.50                100      32768 i

Route Distinguisher: 1.0.0.40:3    (L3VNI 2121210)

*>i[2]:[0]:[0]:[48]:[0050.7966.6807]:[32]:[10.10.0.10]/272
               10.10.10.10                100          0 i
```

As expected, you can confirm that Leaf-05 has received the EVPN L2 route of VLAN10-PC1 via L3VNI. The source of this advertisement is Leaf-01 with the 10.10.10.10 VTEP address. Locally, the Leaf-05 VTEP has advertised in EVPN the L2 route for VLAN20-PC1.

This is EVPN control plane endpoint learning. Still, BGP advertises a /32 prefix or host route for both devices. All your endpoint IPs are injected in BGP as host mobility manager (HMM) routes, or host routes.

Let's look at our tenant-1 VRF route table on both Leaf-01 and Leaf-05 and confirm this information. You should see two routes: one route for 10.10.0.10/32 advertised locally as HMM on Leaf-01 and an incoming EVPN VXLAN type-2 route for 10.20.0.10/32 advertised via the L3VNI tunnel 2121210 incoming from Leaf-05 VTEP 50.50.50.50. Let's confirm!

Tenant VRF VXLAN BGP Routes on Leaf-01

```
DC1-N9500-LEAF1# show ip route vrf tenant-1
IP Route Table for VRF "Tenant-1"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
```

```
1.1.1.0/30, ubest/mbest: 1/0, attached
    *via 1.1.1.1, Eth1/3, [0/0], 00:37:34, direct
1.1.1.1/32, ubest/mbest: 1/0, attached
    *via 1.1.1.1, Eth1/3, [0/0], 00:37:34, local
10.10.0.0/24, ubest/mbest: 1/0, attached
    *via 10.10.0.1, Vlan10, [0/0], 00:37:36, direct
10.10.0.1/32, ubest/mbest: 1/0, attached
    *via 10.10.0.1, Vlan10, [0/0], 00:37:36, local

10.10.0.10/32, ubest/mbest: 1/0, attached
    *via 10.10.0.10, Vlan10, [190/0], 00:34:37, hmm

10.20.0.0/24, ubest/mbest: 1/0, attached
    *via 10.20.0.1, Vlan20, [0/0], 00:37:35, direct
10.20.0.1/32, ubest/mbest: 1/0, attached
    *via 10.20.0.1, Vlan20, [0/0], 00:37:35, local

10.20.0.10/32, ubest/mbest: 1/0
    *via 50.50.50.50%default, [200/0], 00:30:56, bgp-65501,
    internal,  tag 65501, segid: 2121210 tunnelid: 0x32323232
    encap: VXLAN
```

The `show ip route VRF tenant-1` command was executed on Leaf-01 to display all the endpoints learned in VXLAN for tenant-1, and it has been confirmed. Leaf-01 has locally learned the HMM route of 10.10.0.10/32, which belongs to VLAN10-PC1. Additionally, Leaf-01 has learned the route of 10.20.0.10/32, which belongs to VLAN20-PC via L3VNI from Leaf-05 VTEP IP 50.50.50.50. If you perform the same show command on Leaf-05, you should see the same information backward. VLAN10-PC learned via L3VNI and VLAN20-PC locally learned on Leaf-05.

Tenant VRF VXLAN BGP Routes on Leaf-05

```
DC1-N9500-LEAF5# show ip route vrf Tenant-1
IP Route Table for VRF "Tenant-1"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

1.1.1.0/30, ubest/mbest: 1/0, attached
    *via 1.1.1.2, Eth1/4, [0/0], 00:55:28, direct
1.1.1.2/32, ubest/mbest: 1/0, attached
    *via 1.1.1.2, Eth1/4, [0/0], 00:55:28, local
4.4.4.4/32, ubest/mbest: 2/0, attached
    *via 4.4.4.4, Lo100, [0/0], 00:55:26, local
    *via 4.4.4.4, Lo100, [0/0], 00:55:26, direct
10.10.0.0/24, ubest/mbest: 1/0, attached
    *via 10.10.0.1, Vlan10, [0/0], 00:55:38, direct
10.10.0.1/32, ubest/mbest: 1/0, attached
    *via 10.10.0.1, Vlan10, [0/0], 00:55:38, local

10.10.0.10/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/0], 00:43:39, bgp-65501,
internal,     tag 65501, segid: 2121210 tunnelid: 0xa0a0a0a
encap: VXLAN

10.20.0.0/24, ubest/mbest: 1/0, attached
    *via 10.20.0.1, Vlan20, [0/0], 00:55:38, direct
10.20.0.1/32, ubest/mbest: 1/0, attached
    *via 10.20.0.1, Vlan20, [0/0], 00:55:38, local

10.20.0.10/32, ubest/mbest: 1/0, attached
    *via 10.20.0.10, Vlan20, [190/0], 00:41:36, hmm
```

Ladies and gentlemen, there you have it. Leaf-05 locally learned the 10.20.0.10/32 host route HMM. VLAN20-PC learned locally at layer 2.

Also, you can confirm that via the 2121210 L3VNI from Leaf-01 that it has learned the10.10.0.10/32 host route, which is the VLAN10-PC.

It's important to conclude this EVPN overview by discussing the types of EVPN routes.

Next, I discuss EVPN type-2, EVPN type-3, and EVPN type-5, which are important in troubleshooting more efficiently.

EVPN Route Types

EVPN route types are important. Every route information announced and exchanged in EVPN is classified based on the following type attributes: EVPN type-2, EVPN type-3, and EVPN type-5. You usually work with them during post-implementation support and troubleshooting scenarios.

EVPN Type-2

An EVPN type-2 route is used in a fabric to advertise any locally learned MAC address, host routes, or ARP entries. Any end-host information learned in a leaf is advertised as an EVPN type-2 route. Let's execute the `show bgp l2vpn evpn` command to identify our type-2 routes.

```
DC1-N9500-LEAF5# show bgp l2vpn evpn
BGP routing table information for VRF default, address family
L2VPN EVPN
BGP table version is 35, Local Router ID is 1.0.0.40

   Network       Next Hop      Metric     LocPrf     Weight Path

Route Distinguisher: 1.0.0.10:32777

* i[2]:[0]:[0]:[48]:[0050.7966.6807]:[0]:[0.0.0.0]/216
               10.10.10.10                100          0 i
*>i            10.10.10.10                100          0 i
```

```
* i[2]:[0]:[0]:[48]:[0050.7966.6807]:[32]:[10.10.0.10]/272
               10.10.10.10                100           0 i
*>i            10.10.10.10                100           0 i
```

Pay close attention to the value stored in brackets (e.g., [2]) at the beginning of the EVPN route. The [2] value indicates that this is an EVPN type-2 route. There are two type-2 routes in the output. The first EVPN type-2 route is a MAC advisement. The second route is a host route because it begins with [32]. The value is in brackets, and a host route is a route with a /32 prefix.

EVPN Type-3

An EVPN type-3 route advertises the multicast routes required to build a VXLAN VTEP communication. It's also used for VXLAN VTEP to VTEP tunnel establishments. Remember when I discussed L3VNIs? EVPN type-3 routes are exclusive for VXLAN fabric announcements.

EVPN Type-5

While all EVPN type routes are important, EVPN type-5 is even more important to understand. **EVPN type-5 routes are all externally learned routes advertised in EVPN**. In a nutshell, a VXLAN fabric is connected to an external router via the border leaf. The border leaf gets the external router advertised networks and re-advertises them in EVPN as type-5 routes. You usually accomplish this in VXLAN by redistributing either dynamic or static routes in the BGP EVPN L2VPN AFI for the tenant VRF. The next chapter takes a deeper look at external networks and redistribution in VXLAN BGP EVPN.

Let's now look at the `show bgp l2vpn evpn` output and identify an EVPN type-5 route.

```
DC1-9300-BorderLeaf-01# show bgp l2vpn evpn
BGP routing table information for VRF default, address family
L2VPN EVPN
BGP table version is 46, Local Router ID is 1.0.0.11

   Network     Next Hop          Metric   LocPrf    Weight Path

Route Distinguisher: 1.0.0.12:3
* i[5]:[0]:[0]:[24]:[172.22.14.0]:[0.0.0.0]/224
               12.12.12.12       0        100       0 ?
*>i            12.12.12.12       0        100       0 ?

Route Distinguisher: 1.0.0.12:4
* i[5]:[0]:[0]:[23]:[172.26.20.0]:[0.0.0.0]/224
               12.12.12.12       0        100       0 ?
*>i            12.12.12.12       0        100       0 ?
* i[5]:[0]:[0]:[23]:[172.26.22.0]:[0.0.0.0]/224
               12.12.12.12       0        100       0 ?
*>i            12.12.12.12       0        100       0 ?

Route Distinguisher: 1.0.0.11:3    (L3VNI 10101010)

*>l[5]:[0]:[0]:[24]:[172.22.14.0]:[0.0.0.0]/224
               11.11.11.11       20        100      32768 ?
*>l[5]:[0]:[0]:[24]:[172.22.15.0]:[0.0.0.0]/224
               11.11.11.11       20        100      32768 ?
```

In the output, you can identify the type-5 routes beginning with [5]. Type-5 has the subnet CIDR value at the beginning of the network route. For example, in L3VNI 10101010, the first route is [24] (/24). The network address is 172.22.14.0. It has learned from VTEP 11.11.11.11, which means that it's the border leaf facing the external network.

VXLAN Packet Flow

Let's conclude Chapter 3 by reviewing a high-level VXLAN packet flow in a spine-and-leaf fabric. Let's review the diagram in Figure 3-4 and each step the source packet takes to reach its destination in a spine-and-leaf architecture.

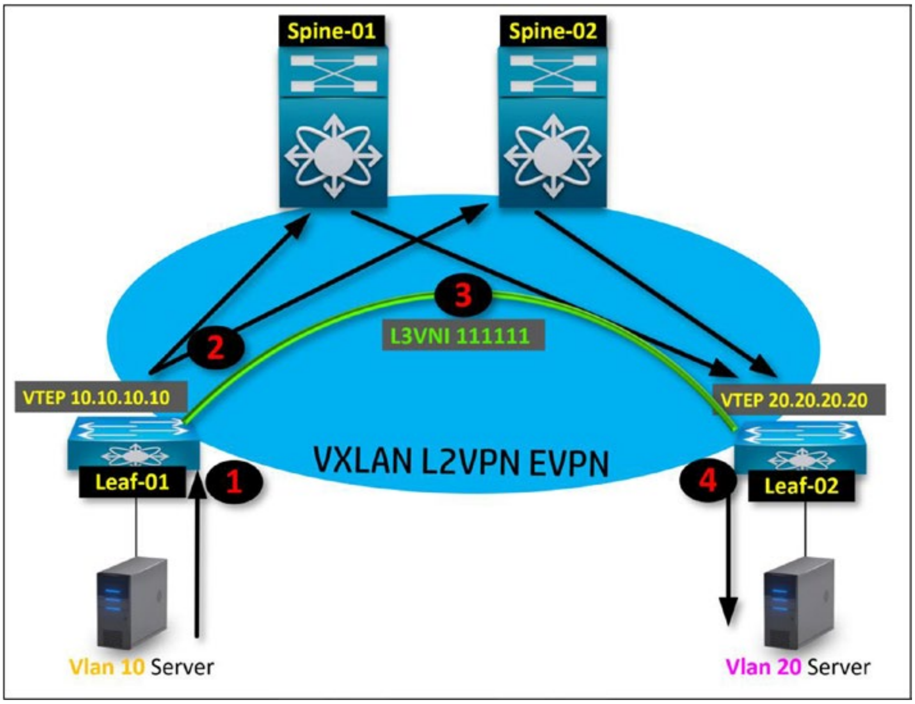


Figure 3-4. *A step-by-step packet transit in a spine and leaf VXLAN BGP EVPN fabric*

Step 1: Initial request from the source device to the network

From Leaf-01, a server on VLAN 10 has sent a packet destined to a server on VLAN 20. The destination server on VLAN 20 is located in Leaf-02. The packet ingresses on Leaf-01 via an access port configured for VLAN 10. Leaf-01 has a VNI of 10010 for VLAN 10 and encapsulates this layer 2 packet into an L3 VXLAN UDP frame with a VXLAN header tagged with VNI 10010. Leaf-01 verifies the destination of this packet, and it sees the destination to be on VLAN 20. It verifies the EVPN control-plane route table to identify the type-2 route that matches the packet's destination. It has located the destination leaf to be Leaf-02 which connects to the destination server.

Step 2: Source leaf forwards the request via L3VNI to its destination

Leaf-01 sends this traffic over L3VNI to Leaf-02. The L3VNI tunnel between Leaf-01 and Leaf-02 is established via the spines. Leaf-01 forwards it to L3VNI. The packet transparently goes northbound via Spine-01 or Spine-02.

Step 3: Packet flows via L3VNI using the spines as the transit path

During the transit via L3VNI, the traffic encapsulated in VXLAN EVPN is sent to the spines. The spines reflect the traffic to the destination leaf.

Step 4: Packet flows via L3VNI using the spines as the transit path

Once the VXLAN traffic traverses the spine layer, it egresses to the Leaf-02 destination via the L3VNI tunnel. Leaf-02 gets the packet, decapsulates it from VXLAN, and forwards it to its destination using the local MAC address table for VLAN 20. The packet arrives at the destination server via the end-host port facing the server.

CHAPTER 4

VXLAN Fabric Features and Functionalities

If you don't know what features a VXLAN fabric provides, then what is the point of even deploying it in the first place. Let's review the most important fabric features that make you transition from your slow, boring, spanning-tree-driven network to VXLAN BGP EVPN. The feature gain going to VXLAN BGP EVPN for your data center is worth the effort.

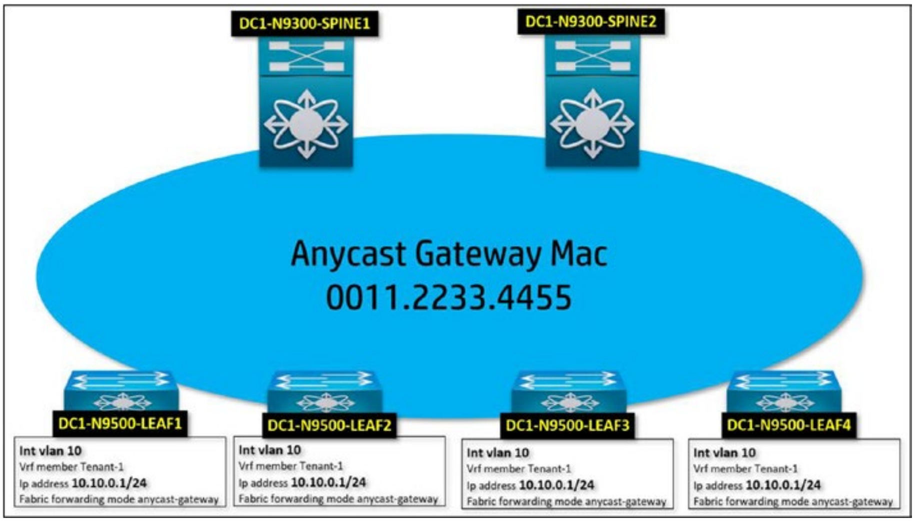
Fabric Anycast Gateways

To discuss anycast gateways, I must refresh your memory on FHRP (first-hop redundancy protocols). I'm sure you have had to support environments running HSRP, VRRP, or GLBP. You deploy HSRP or VRRP to provide a redundant L3 routing architecture. It works in an active/standby mode. With HSRP or VRRP, you configure a group ID, then add member routers (or switches) as members of the group. Once configured, you then allocate a virtual IP that is shared between all group members. This virtual IP is usually a VLAN primary SVI or a router-to-router failover address.

R. Cardona, *The Fast-Track Guide to VXLAN BGP EVPN Fabrics*,
https://doi.org/10.1007/978-1-4842-6930-5_4

Only one router or switch acts as the active arp and traffic holder for anything that hops via this virtual IP. If the active router fails, the FHRP moves the active state to the standby router in the group. It served the failover purpose, but a complete switch is standing by not doing anything unless a failover event makes it active. Wouldn't it be great to have all switches active performing routing functions rather than one?

The anycast gateway is what it is all about. In your VXLAN fabric, with an anycast gateway, all your leafs act as active L3 core switches. IP addresses are duplicated across the leaf layer (see Figure 4-1).



```
Int vlan 10
Vrf member Tenant-1
Ip address 10.10.0.1/24
Fabric forwarding mode anycast-gateway
```

```
Int vlan 10
Vrf member Tenant-1
Ip address 10.10.0.1/24
Fabric forwarding mode anycast-gateway
```

```
Int vlan 10
Vrf member Tenant-1
Ip address 10.10.0.1/24
Fabric forwarding mode anycast-gateway
```

```
Int vlan 10
Vrf member Tenant-1
Ip address 10.10.0.1/24
Fabric forwarding mode anycast-gateway
```

Figure 4-1. *All leafs have the same defined SVI IPs. Every leaf acts as the core switch. In essence try to look at the environment as one giant core switch performing all routing decisions*

Anycast gateways work in a VXLAN BGP EVPN fabric by allocating a virtual MAC address that is shared across all leaf. By this approach, you spoof all clients into thinking that there's a single core switch. This fabric anycast gateway MAC is assigned to all leafs. Figure 4-1 shows the anycast-gateway MAC is 0011.2233.4455, which you assign during the anycast gateway configurations. From a routing standpoint, your adjacent leaf performs routing functions.

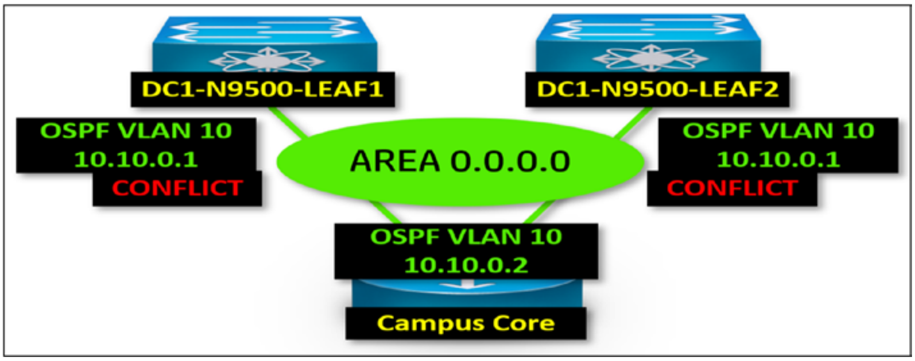


Figure 4-2. *An OSPF adjacency conflict between Leaf-01 and Leaf-02 to the campus core is present since you are using an anycast gateway SVI, which has the same IP configured on both leafs. The neighbor does not properly identify them since they share the same IP*

Let's say that PC1 is connected to Leaf-01, which then acts as the gateway for all PC1 communication requests. There are some issues that you could come across with anycast gateways. One of them is dynamic routing protocols like OSPF since the OSPF process sees a duplicate neighbor since multiple leafs share the same VLAN interface IP (see Figure 4-2).

Let's configure the anycast gateways. Our environment has three VLANs under the tenant-1 VRF Table 4-1. You need to create the SVIs and apply them to all leafs.

Table 4-1. *TENANT-1 Networks*

NAME	VLAN ID	IP
Db	10	10.10.0.1/24
Files	20	10.20.0.1/24
Web	30	10.30.0.1/24

Listing 4-1 is the fabric's anycast gateway configuration.

Listing 4-1. General anycast gateway configuration

```
! APPLY TO ALL LEAFS
feature fabric forwarding
feature interface-vlan
fabric forwarding anycast-gateway-mac 0011.1A2B.3C4D

int vlan 10
member vrf Tenant-1
ip address 10.10.0.1/24
fabric forwarding mode anycast-gateway
no shut

int vlan 20
member vrf Tenant-1
ip address 10.20.0.1/24
fabric forwarding mode anycast-gateway
no shut

int vlan 30
no shutdown
member vrf Tenant-1
ip address 10.30.0.1/24
fabric forwarding mode anycast-gateway
no shut
```

By applying this configuration script, you have successfully implemented fabric anycast gateways. Now all the fabric leafs are performing inter-VLAN routing functions. Rather than having a single core performing all routing, your data center has a robust and scalable routing domain.

Multitenancy with VRFs

Let's discuss and configure multitenancy in VXLAN BGP EVPN fabrics. A single spine-and-leaf VXLAN fabric can accommodate numerous tenants. Every tenant is an independent network running on the same physical architecture. Not all requirements are for different customers running on the same fabric. You can leverage VRFs to perform network segmentation for security reasons. Let's isolate the web server farm into its own tenant and all wired and wireless client networks on a completely different tenant.

Finally, let's get the DMZ network on a different tenant. Do you see the picture here? How can you then allow communication, for example, between the client and server tenants? Easy, either insert a stateful firewall between the client and server VRFs and inspect inbound and outbound traffic from those two tenants. You could implement VRF route leaking, which uses the control plane to "leak" traffic between different VRFs.

Let's look at the diagram in Figure 4-3 to get a clearer understanding of the concept.

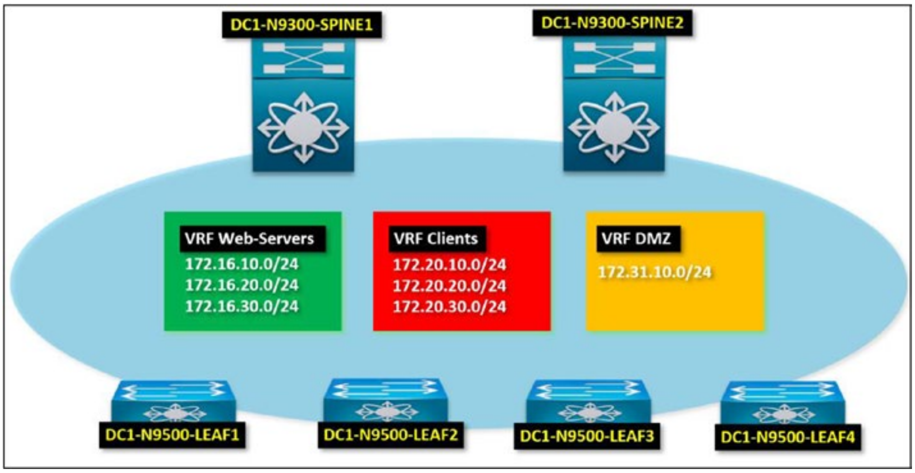


Figure 4-3. *The diagram shows a three-tenant vxlan fabric. You create isolated route tables with VRFs. This provides the multitenant feature in VXLAN BGP EVPN*

Now that you have a better idea of how multitenancy works in VXLAN. Let's configure our tenants. In Chapter 3, I mentioned that for every tenant, you need an L3VNI so that traffic for the assigned tenant can traverse across the fabric. Taking this into consideration, let's build our multitenant configs (see Tables 4-2, 4-3, and 4-4).

Table 4-2. *Tenant: Web Servers L3VNI 1111111*

VLAN	VLAN ID	VNI	IP
WebApp1	10	100010	172.16.10.0/24
WebApp2	20	100020	172.16.20.0/24
DB	30	100030	172.16.30.0/24

Table 4-3. *Tenant: Clients L3VNI 2222222*

VLAN	VLAN ID	VNI	IP
Wired	11	100011	172.20.11.0/24
Phones	12	100012	172.20.12.0/24
Wireless	13	100013	172.20.13.0/24

Table 4-4. *Tenant: DMZ L3VNI 3333333*

VLAN	VLAN ID	VNI	IP
DMZ	310	100310	172.31.10.0/24

With this information, you can build the configurations needed to enable those three tenants in the fabric. As I discussed, the tenants cannot reach each other unless you perform route leaking or insert a firewall/router between VRFs. Let's build the configuration template and include the anycast gateway for those VLANs on the configuration.

First, let's look at the tenant web servers, as shown in Listing 4-2.

Listing 4-2. A complete sample tenant configuration in VXLAN

```
Vlan 10
vn-segment 100010
name WebApp1

Vlan 20
vn-segment 100020
name WebApp2

Vlan 30
vn-segment 100030
name DB

vlan 111
name WebServers-L3VNI
vn-segment 1111111

vrf context Web-Servers
vni 1111111
rd auto
address-family ipv4 unicast
route-target both 65501:111111
route-target both 65501:111111 evpn

int vlan 10
no shut
vrf member Web-Servers
```

```
ip address 172.16.10.1/24
fabric forwarding mode anycast-gateway

int vlan 20
no shut
vrf member Web-Servers
ip address 172.16.20.1/24
fabric forwarding mode anycast-gateway

int vlan 30
no shut
vrf member Web-Servers
ip address 172.16.30.1/24
fabric forwarding mode anycast-gateway

int vlan 111
vrf member Web-Servers
ip forward

int nve1
member vni 1111111 associate-vrf
```

Next, let's look at the tenant clients, as shown in Listing 4-3.

Listing 4-3. Tenant "Clients" configuration template

```
Vlan 11
vn-segment 100011
name Wired

Vlan 12
vn-segment 100012
name Phones

Vlan 13
vn-segment 100013
name Wireless
```

```
vlan 222
name Clients-L3VNI
vn-segment 2222222

vrf context Clients
vni 2222222
rd auto
address-family ipv4 unicast
route-target both 65501:222222
route-target both 65501:222222 evpn

int vlan 11
no shut
vrf member Clients
ip address 172.20.11.1/24
fabric forwarding mode anycast-gateway

int vlan 12
no shut
vrf member Clients
ip address 172.20.12.1/24
fabric forwarding mode anycast-gateway

int vlan 13
no shut
vrf member Clients
ip address 172.20.13.1/24
fabric forwarding mode anycast-gateway

int vlan 222
vrf member Clients
ip forward

int nve1
member vni 2222222 associate-vrf
```

Now, let's look at the tenant DMZ, as shown in Listing 4-4.

Listing 4-4. Tenant "DMZ" configuration template

```
Vlan 310
vn-segment 100310
name DMZ

vlan 333
name DMZ-L3VNI
vn-segment 3333333

vrf context DMZ
vni 3333333
rd auto
address-family ipv4 unicast
route-target both 65501:333333
route-target both 65501:333333 evpn

int vlan 310
no shut
vrf member DMZ
ip address 172.31.10.1/24
fabric forwarding mode anycast-gateway

int vlan 333
no shut
vrf member DMZ
ip forward

int nve1
member vni 3333333 associate-vrf
```

Once you finish applying the tenant configurations to all leafs let's verify the active interfaces for each tenant and the VRF route table and confirm that the networks have been provisioned. The following is the interface assigned to a respective VRF type.

```
show ip interface brief vrf NAME
```

Replace **NAME** with your tenant name. The following is the route table for a particular tenant type.

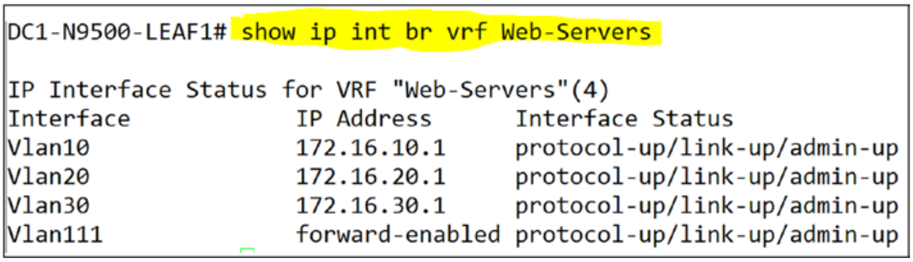
```
show ip route vrf NAME
```

With the information in place, let's verify our configuration on Leaf-01. You should see the route table and member interfaces on each tenant.

Tenant: Web Servers

The following command is demonstrated in Figure 4-4.

```
show ip interface brief vrf Web-Servers
```

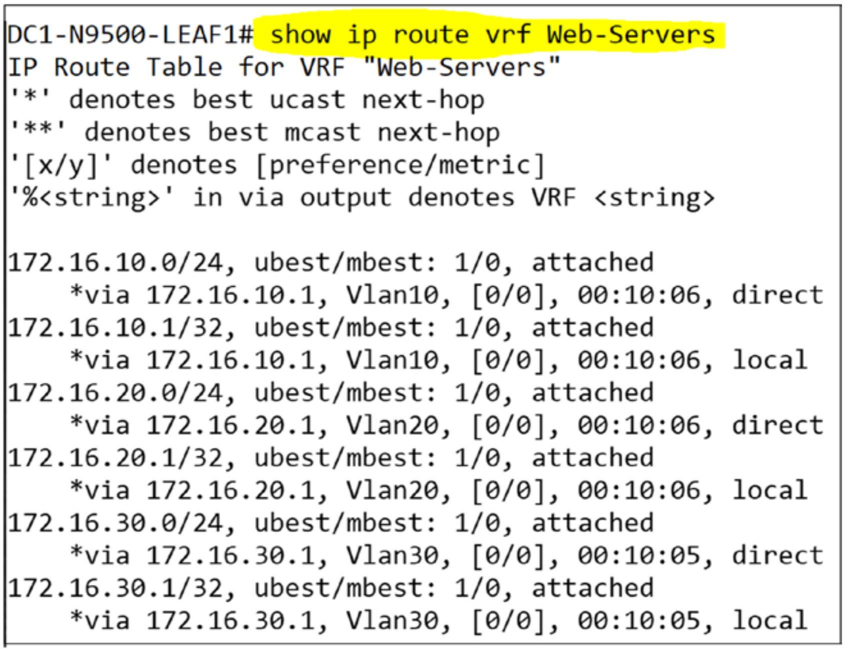


```
DC1-N9500-LEAF1# show ip int br vrf Web-Servers

IP Interface Status for VRF "Web-Servers"(4)
Interface            IP Address      Interface Status
Vlan10               172.16.10.1     protocol-up/link-up/admin-up
Vlan20               172.16.20.1     protocol-up/link-up/admin-up
Vlan30               172.16.30.1     protocol-up/link-up/admin-up
Vlan111              forward-enabled protocol-up/link-up/admin-up
```

Figure 4-4. *Lets display the active L3 interfaces on the Web-Servers vrf*

The following command is demonstrated in Figure 4-5.

```
show ip route vrf Web-Servers
```



```
DC1-N9500-LEAF1# show ip route vrf Web-Servers
IP Route Table for VRF "Web-Servers"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0, attached
    *via 172.16.10.1, Vlan10, [0/0], 00:10:06, direct
172.16.10.1/32, ubest/mbest: 1/0, attached
    *via 172.16.10.1, Vlan10, [0/0], 00:10:06, local
172.16.20.0/24, ubest/mbest: 1/0, attached
    *via 172.16.20.1, Vlan20, [0/0], 00:10:06, direct
172.16.20.1/32, ubest/mbest: 1/0, attached
    *via 172.16.20.1, Vlan20, [0/0], 00:10:06, local
172.16.30.0/24, ubest/mbest: 1/0, attached
    *via 172.16.30.1, Vlan30, [0/0], 00:10:05, direct
172.16.30.1/32, ubest/mbest: 1/0, attached
    *via 172.16.30.1, Vlan30, [0/0], 00:10:05, local
```

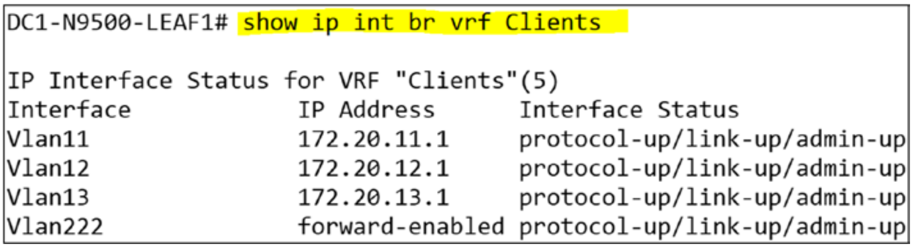
Figure 4-5. *Lets display the active routes on the Web-Servers vrf*

You can confirm that the tenant web servers have its own route table and interfaces. The same verification applies to the other two tenants, clients, and a DMZ. Remember, you are running three different tenants on the same physical switching hardware.

Tenant: Clients

The following command is demonstrated in Figure 4-6.

```
show ip interface brief vrf Clients
```

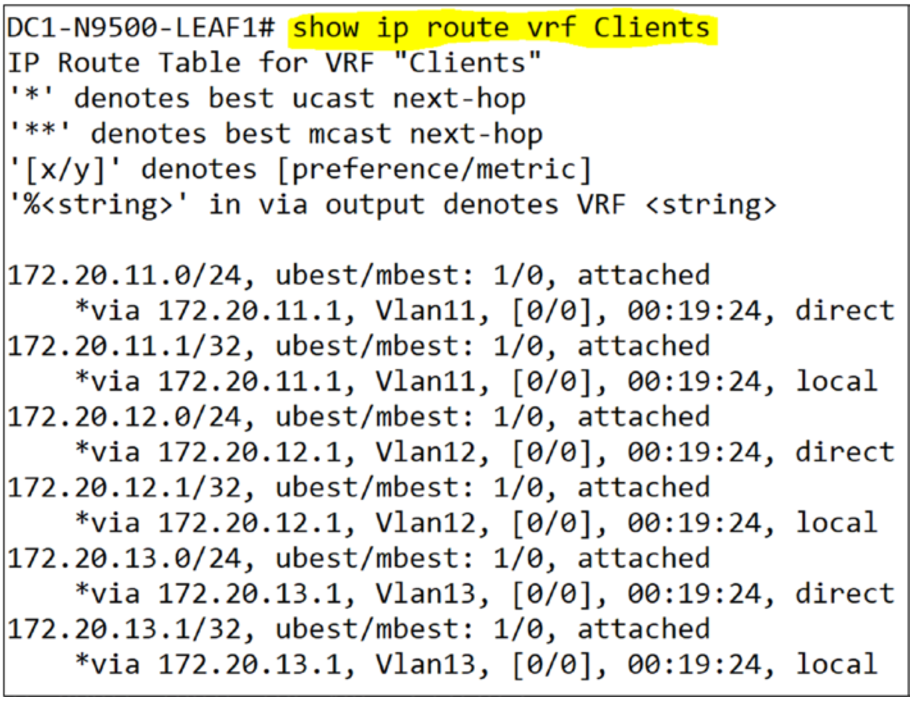


```
DC1-N9500-LEAF1# show ip int br vrf Clients

IP Interface Status for VRF "Clients"(5)
Interface            IP Address      Interface Status
Vlan11               172.20.11.1     protocol-up/link-up/admin-up
Vlan12               172.20.12.1     protocol-up/link-up/admin-up
Vlan13               172.20.13.1     protocol-up/link-up/admin-up
Vlan222              forward-enabled protocol-up/link-up/admin-up
```

***Figure 4-6.** Lets display the active L3 interfaces on the Clients vrf*

The following command is demonstrated in Figure 4-7.

```
show ip route vrf Clients
```



```
DC1-N9500-LEAF1# show ip route vrf Clients
IP Route Table for VRF "Clients"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.20.11.0/24, ubest/mbest: 1/0, attached
    *via 172.20.11.1, Vlan11, [0/0], 00:19:24, direct
172.20.11.1/32, ubest/mbest: 1/0, attached
    *via 172.20.11.1, Vlan11, [0/0], 00:19:24, local
172.20.12.0/24, ubest/mbest: 1/0, attached
    *via 172.20.12.1, Vlan12, [0/0], 00:19:24, direct
172.20.12.1/32, ubest/mbest: 1/0, attached
    *via 172.20.12.1, Vlan12, [0/0], 00:19:24, local
172.20.13.0/24, ubest/mbest: 1/0, attached
    *via 172.20.13.1, Vlan13, [0/0], 00:19:24, direct
172.20.13.1/32, ubest/mbest: 1/0, attached
    *via 172.20.13.1, Vlan13, [0/0], 00:19:24, local
```

***Figure 4-7.** Lets display the active routes on the Clients vrf*

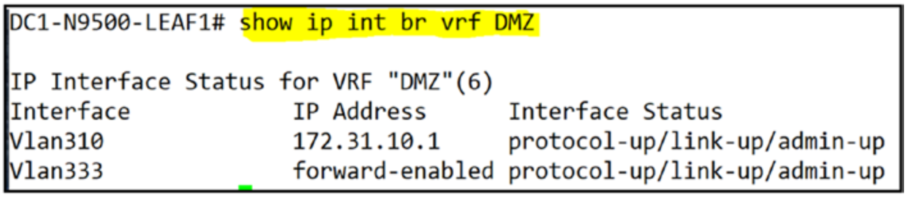
Note that all the commands are being executed from the same Leaf-01. Three different environments are running on the same fabric switch. Very cool indeed.

Finally, let's perform the verification on the DMZ tenant.

Tenant: DMZ

The following command is demonstrated in Figure 4-8.

```
show ip interface brief vrf DMZ
```

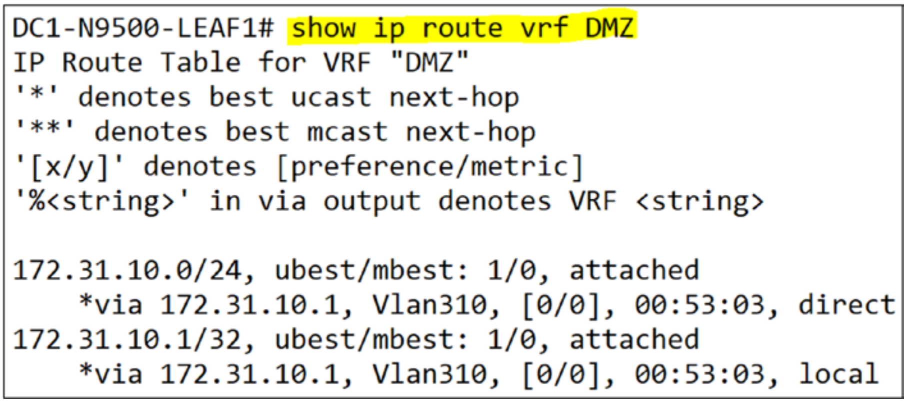


```
DC1-N9500-LEAF1# show ip int br vrf DMZ

IP Interface Status for VRF "DMZ"(6)
Interface            IP Address      Interface Status
Vlan310              172.31.10.1     protocol-up/link-up/admin-up
Vlan333              forward-enabled protocol-up/link-up/admin-up
```

Figure 4-8. *Lets display the active L3 interfaces on the DMZ vrf*

The following command is demonstrated in Figure 4-9.

```
show ip route vrf DMZ
```



```
DC1-N9500-LEAF1# show ip route vrf DMZ
IP Route Table for VRF "DMZ"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.31.10.0/24, ubest/mbest: 1/0, attached
    *via 172.31.10.1, Vlan310, [0/0], 00:53:03, direct
172.31.10.1/32, ubest/mbest: 1/0, attached
    *via 172.31.10.1, Vlan310, [0/0], 00:53:03, local
```

Figure 4-9. *Lets display the active routes on the DMZ vrf*

As you can see, the tenants, web servers, clients, and DMZ are configured on the same spine-and-leaf VXLAN fabric. Now isolated networks are inside the same physical hardware.

Let's now talk about how to allow communication between those three tenants. You can be selective and allow unidirectional communication or bidirectional communication between the VRFs. To enable this communication, you must either add a routing device in the path of your VRFs or perform route leaking in the VRF.

First, add an external routing device to your VRFs. One great example is a stateful firewall. With a stateful firewall between your VRFs, you can restrict and/or grant who can talk to what. It adds the required network security to control what ingress and egress out of your tenants. It is very important to discuss our L3 connectivity options to allow VRF to VRF routing functionality. Let's start by working with plain vanilla static routes.

Static Routing in Tenant VRFs

During the tenant discussion, you built a three-tenant VXLAN fabric. The tenants are web servers, clients, and DMZ. Let's assume that you've been tasked to allow communication between the client VRF and a specific subnet in the web servers VRF. The CISO (chief information security officer) requires inspecting all inbound traffic to the web servers VRF from the client's VRF. Let's configure the VXLAN fabric to allow the required network connectivity. You are going to insert a stateful firewall to route and inspect all traffic from the client VRF to the web server VRF. Table 4-5 lists the specific details to perform the task.

Table 4-5. *Allow Reachability Between the Following Subnets*

Client VRF Subnets	web servers VRF Subnet
172.20.11.0/24	172.16.10.0/24
172.20.12.0/24	
172.20.13.0/24	

Table 4-6. *L3 addressing between the Client VRF and the external firewall*

Client VRF IP	Leaf-02 Interface	Client Zone Firewall IP
172.21.0.1/30	**Eth1/5**	172.21.0.2/30
web servers VRF IP	**Leaf-02 Interface**	**web server Zone Firewall IP**
172.22.0.1/30	**Eth1/6**	172.22.0.2/30

This information allows you to build the logical design knowing how to execute the actual configurations. There is a layer 3 link between the client VRF and the external firewall. There is another link between the web servers VRF and the firewall. There are IP assignments for those links. You know which networks from the client VRF should reach the specific web servers VRF.

There are two choices available to limit the client to web server reachability to a single destination subnet. You can either provide a static route to the client VRF for the single subnet in the web servers VRF or implement a firewall policy to block inbound reachability from the client VRF to the other subnets in the web servers VRF. Since you want to focus on performing VRF configurations, let's configure the required static route on the client VRF.

A logical diagram for this scenario is shown in Figure 4-10.

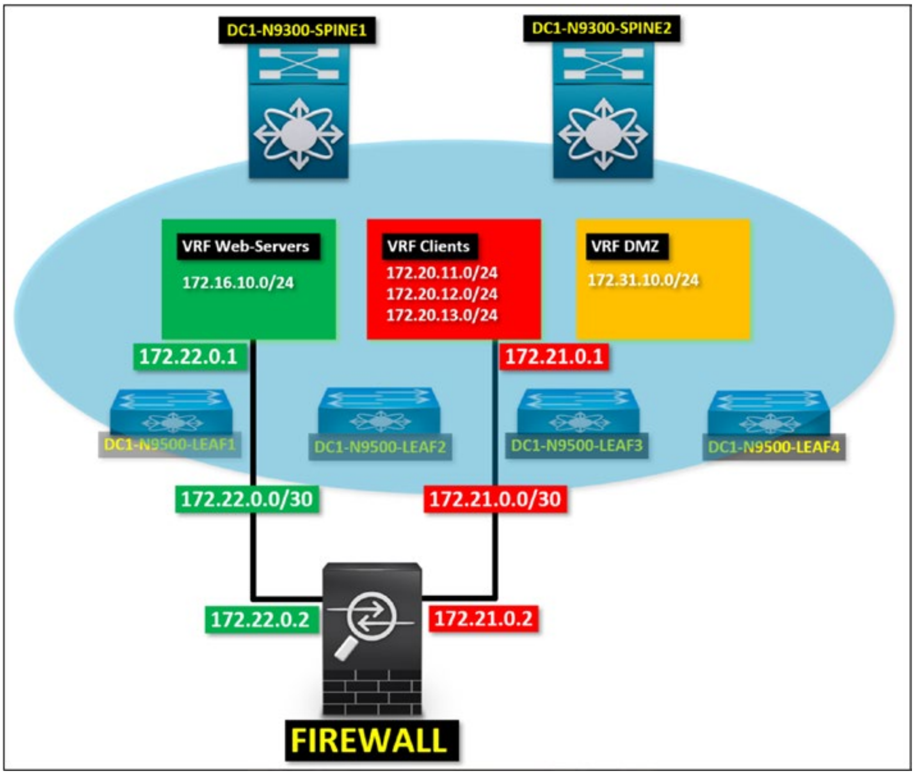


Figure 4-10. *VRF-to-VRF routing using a firewall as the routed hop and integrating an external firewall to a VXLAN fabric*

Now that you have a better understanding of what we are trying to accomplish, let's perform the configuration. As shown in Figure 4-10, the connection path is between the VRF to the firewall, but physically, you need to designate a leaf to be our border leaf for this VRF to firewall to VRF reachability. Let's choose Leaf-02 to be our border leaf and perform the required configurations. You start by configuring an interface on Leaf-02 for the client VRF to the client zone on the firewall and another interface on Leaf-02 for the web servers VRF to the web servers zone on the firewall.

Let's begin by configuring the client VRF and interface (see Listing 4-5), and follow with the web servers VRF and interface (see Listing 4-6).

Listing 4-5. Client VRF L3 Configuration

```
!LEAF2 Config
int eth1/5
no switchport
vrf member Clients
ip address 172.21.0.1/30
no shut

vrf context Clients
ip route 172.16.10.0/24 172.21.0.2
```

Listing 4-5 is the sample configuration to allow routing from the client VRF to the web servers VRF. There are two items. The first is the L3 interface facing the firewall. You add the VRF membership and then configure the IP address. In the VRF context for clients, add a route for the web servers' subnet you need to reach via the firewall (see Listing 4-6).

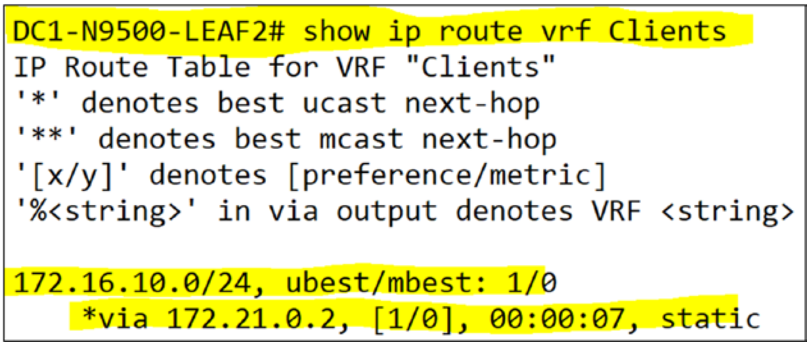
Listing 4-6. Web Servers VRF L3 Configuration

```
!LEAF2 Config
int eth1/6
no switchport
vrf member Web-Servers
ip address 172.22.0.1/30
no shut

vrf context Web-Servers
ip route 172.20.11.0/24 172.22.0.2
ip route 172.20.12.0/24 172.22.0.2
ip route 172.20.13.0/24 172.22.0.2
```

Listing 4-6 is a similar configuration, but this time allows routing from the web servers VRF to the client VRF.

Now that you've successfully configured the static routes, let's do some checks on the Leaf-02 to confirm that you see the routes on the VRF. You reviewed Leaf-02 but if you're curious like me, you've already asked about the other leafs. Yes, you are getting there!

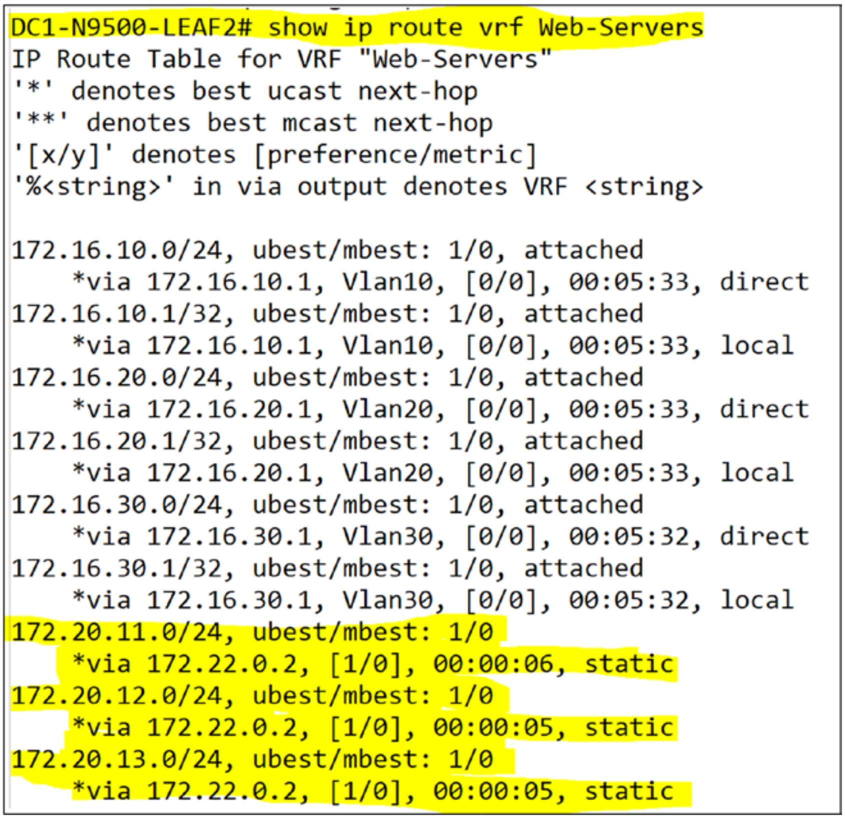
Figure 4-11 shows a client VRF route table.



```
DC1-N9500-LEAF2# show ip route vrf Clients
IP Route Table for VRF "Clients"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0
    *via 172.21.0.2, [1/0], 00:00:07, static
```

Figure 4-11. *Statically configured routes in the Clients tenant*

Figure 4-12 shows a web server VRF route table.



```
DC1-N9500-LEAF2# show ip route vrf Web-Servers
IP Route Table for VRF "Web-Servers"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0, attached
    *via 172.16.10.1, Vlan10, [0/0], 00:05:33, direct
172.16.10.1/32, ubest/mbest: 1/0, attached
    *via 172.16.10.1, Vlan10, [0/0], 00:05:33, local
172.16.20.0/24, ubest/mbest: 1/0, attached
    *via 172.16.20.1, Vlan20, [0/0], 00:05:33, direct
172.16.20.1/32, ubest/mbest: 1/0, attached
    *via 172.16.20.1, Vlan20, [0/0], 00:05:33, local
172.16.30.0/24, ubest/mbest: 1/0, attached
    *via 172.16.30.1, Vlan30, [0/0], 00:05:32, direct
172.16.30.1/32, ubest/mbest: 1/0, attached
    *via 172.16.30.1, Vlan30, [0/0], 00:05:32, local
172.20.11.0/24, ubest/mbest: 1/0
    *via 172.22.0.2, [1/0], 00:00:06, static
172.20.12.0/24, ubest/mbest: 1/0
    *via 172.22.0.2, [1/0], 00:00:05, static
172.20.13.0/24, ubest/mbest: 1/0
    *via 172.22.0.2, [1/0], 00:00:05, static
```

Figure 4-12. *Statically configured routes in the Web-Servers tenant*

You can confirm that the statically configured routes on both VRF route tables are in the VRF context for both clients and web servers. This communication between VRFs works on leaf02 only. This isn't going to work on the remaining leafs because the configurations were made only on leaf02, and there is no local connectivity to the firewall from the remaining leafs. Here's where the L3VNI and route redistribution in EVPN does the magic. Remember EVPN type-5? Now it's time to make good use of it.

Static Route Redistribution in EVPN Using Tags

You have enabled connectivity between the tenant's web servers and clients. The problem is that the connectivity only happens in the local border leaf in which you connected the external firewall to the VRFs. The rest of the VXLAN fabric doesn't know about any routes to allow both VRFs to communicate, so let's fix it.

Let's get the static route path from the border leaf (Leaf-02) redistributed onto the fabric in EVPN. You perform this configuration once the BGP EVPN process advertises a route to the destination network via the VTEP IP of the border leaf.

So, let's say from the client tenant-VRF you want to reach the web server tenant VRF. Once it is redistributed, EVPN tells all the other leafs to send the traffic to Leaf-02 (the border leaf) via its 12.12.12.12 VTEP address and forward the traffic via the L3VNI tunnel between the local leaf and Leaf-02. I can explain it to you verbally, but Figure 4-13 shows the full picture.

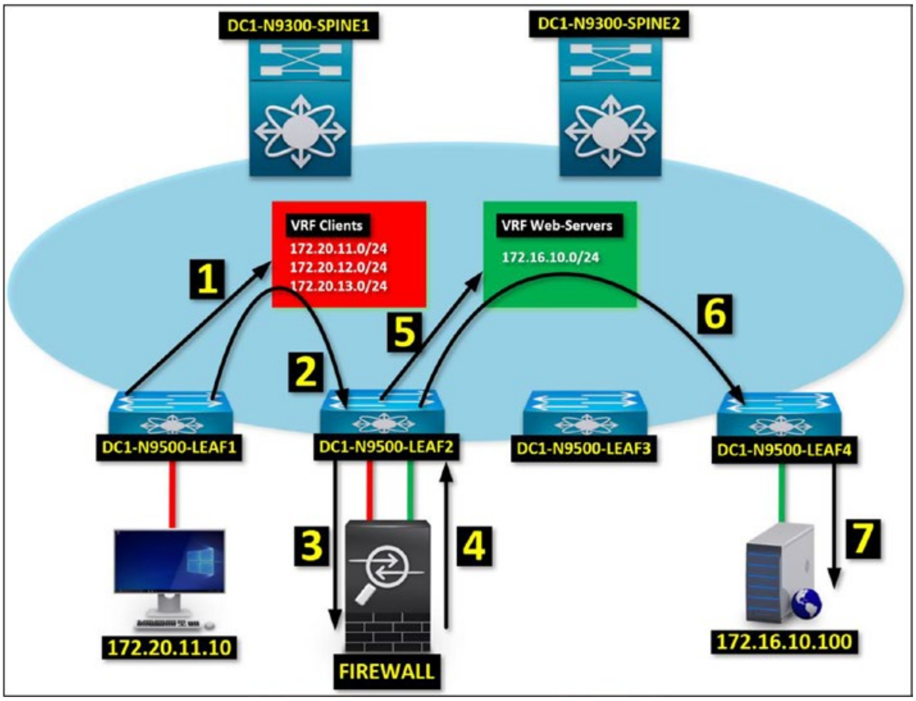


Figure 4-13. *VRF to VRF routing steps using an external firewall redistributed in EVPN as type-5 external route*

The following explains the steps for VRF-to-VRF routing with an EVPN type-5 path.

1. The client PC 172.20.11.10 sends a communication request to the web server. The request arrives at the local leaf client VRF.

2. The local leaf identifies the next hop to the web server's destination IP via the L3VNI on the client VRF between Leaf-01 and Leaf-02. It forwards the request via the L3VNI to Leaf-02. The request arrives at Leaf-02 in the client-VRF.

3. Leaf-02 has a configured static route that points to the web server subnet destination via the external firewall, Leaf-02 sends the request to the external firewall. At this point, the traffic leaves the client VRF.
4. The client request arrives at the external firewall. The external firewall has a route for the web server's subnet via the link to Leaf-02 in the web servers VRF. The firewall forwards the request to Leaf-02.
5. The client PC request arrives at the web servers VRF in Leaf-02.
6. At the route table in the Leaf-02 web server's VRF, there is an advertised EVPN host route of the web server's IP via the L3VNI tunnel between Leaf-02 and Leaf-04. Leaf-02 forwards the request via the tunnel to Leaf-04. Leaf-04 receives the request and forwards it to the web server.
7. The client PC request arrives at the web server.

A VXLAN BGP EVPN spine-and-leaf architecture offers endless routing design options. You leverage EVPN to redistribute routes in the fabric from external networks. VXLAN BGP EVPN not only encapsulates layer 2 over layer 3, it also performs scalable routing topologies. In the next section, I perform the configuration aspect of the static route redistribution to EVPN scenario.

Static Route Redistribution in EVPN with Route Tags

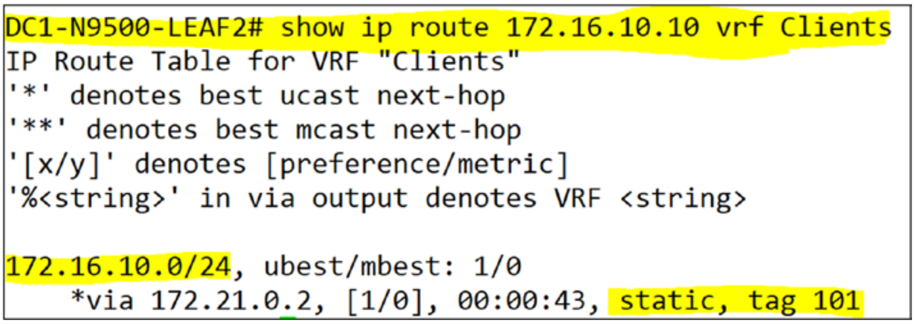
If you haven't performed route redistribution with route tags, you're in luck. I show you how easy and flexible it is to redistribute networks with tags. You assign a value to an interface or a route with a tag. Let's say you have the following route: 172.16.10.0/24 via 172.21.0.2. You want to reference this route on your redistribution command statement, and then assign a tag to it.

```
ip route 172.16.10.0/24 172.21.0.2 tag 101
```

You use the tag to build a route-map and reference the value. The route-map is then associated with the route redistribution statement. The route redistribution statement is associated with the routing process to which you intend to redistribute. The following is an example.

```
vrf context Clients
address-family ipv4 unicast
ip route 172.16.10.0/24 172.21.0.2 tag 101
route-map static-to-evpn permit 10
 match tag 101

router bgp 65501
vrf Clients
address-family ipv4 unicast
 redistribute static route-map static-to-evpn
```

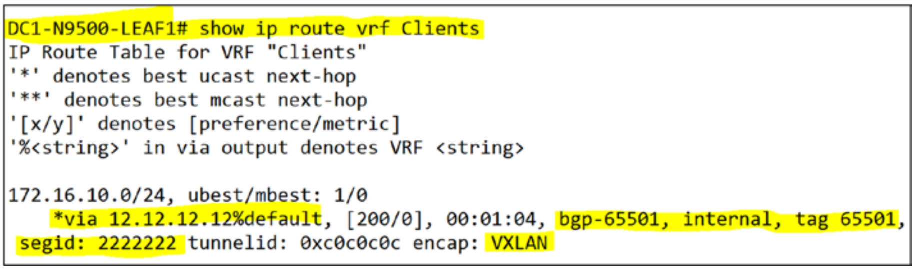
In a nutshell, you tag the static route. You then create the route-map and match the tag value. Finally, under the BGP process for the tenant VRF configure the redistribution command. What happens next is that BGP injects the route statement in the control plane EVPN to the other leafs and advertises the next hop of this destination network via the L3VNI to the leaf you are redistributing from. Let's test it out (see Figure 4-14).



```
DC1-N9500-LEAF2# show ip route 172.16.10.10 vrf Clients
IP Route Table for VRF "Clients"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0
    *via 172.21.0.2, [1/0], 00:00:43, static, tag 101
```

Figure 4-14. *On Leaf-02, the static route is present on the client VRF route table with the assigned tag value of 101*

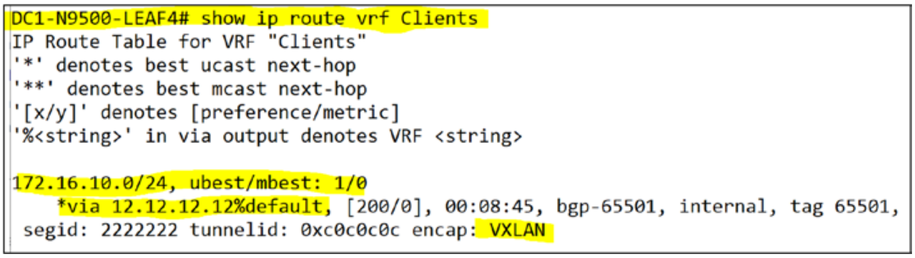
You applied the route redistribution commands on Leaf-02. You should expect an EVPN route to 172.16.10.0/24 via the L3VNI to Leaf-02 (VTEP 12.12.12.12) Let's verify on Leaf-01 and Leaf-04 (see Figure 4-15).



```
DC1-N9500-LEAF1# show ip route vrf Clients
IP Route Table for VRF "Clients"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/0], 00:01:04, bgp-65501, internal, tag 65501,
 segid: 2222222 tunnelid: 0xc0c0c0c encap: VXLAN
```

Figure 4-15. *A static route redistribution in EVPN output in the client VRF received by a member leaf in the fabric*

Figure 4-16 looks at Leaf-04 to confirm that you see the same redistributed route.



```
DC1-N9500-LEAF4# show ip route vrf Clients
IP Route Table for VRF "Clients"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/0], 00:08:45, bgp-65501, internal, tag 65501,
 segid: 2222222 tunnelid: 0xc0c0c0c encap: VXLAN
```

Figure 4-16. *The route is successfully redistributed via the Clients vrf L3VNI interface*

This is how you execute a static route redistribution with BGP EVPN in a VXLAN fabric.

BGP IPv4 Unicast Address Family

The BGP IPv4 unicast address family provides the control plane process to handle IPv4 routing advertisements in a BGP autonomous system. In a VXLAN fabric, it's no different. You can use the IPv4 AFI to advertise networks in the VRF under the BGP process. With BGP EVPN, you also use the IPv4 unicast with VRF lite to advertise the same network in different tenants. For example, you have two tenants, and each tenant must advertise its subnets between all leafs in the VXLAN fabric. You must add the VRF under the BGP process and enable the IPv4 unicast address family.

Listing 4-7 shows the BGP process configuration, including VRFs.

Listing 4-7. The complete BGP IPv4 Unicast configuration for both the underlay and the tenants

```
router bgp 65501
  log-neighbor-changes
  address-family ipv4 unicast (IPv4 Unicast AFI in the BGP
  Underlay.)
    network 10.10.10.10/32
  address-family l2vpn evpn
  template peer TO_SPINES
    remote-as 65501
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
    address-family l2vpn evpn
      send-community
      send-community extended
  neighbor 1.0.0.100
    inherit peer TO_SPINES
  neighbor 1.0.0.200
    inherit peer TO_SPINES
  vrf Clients (Below the BGP Neighbor configs its where we add
  the VRF's to the process)
    address-family ipv4 unicast
  vrf Tenant-1
    address-family ipv4 unicast
      advertise l2vpn evpn
```

The IPv4 unicast address family advertises networks by adding the network statement or adding a route redistribution statement. You can manipulate parameters such as distance and metrics. In the next chapter, you heavily work with dynamic routing in VRFs. I expect you to become very familiar with managing the IPv4 unicast AFI once you complete Chapter 5.

BGP Route Leaking Between VRFs

At this stage, you should have a very solid foundational knowledge of routing networks for a tenant VRF in a VXLAN BGP EVPN fabric. You know how to route between tenants using an external routing device such as a router or firewall. But what if you want to route between VRFs without the need for an external device? You can do it!

Route leaking in BGP with VRF is performed by exporting and importing route targets. Please go back to Chapter 2's discussion of the L2VPN EVPN address family if you want a refresher on route-target import and export. I explain the whole idea behind it. Route leaking for IPv4 unicast AFI is no different. I export the route-target from my source VRF and import it to my destination VRF (see Figure 4-17).

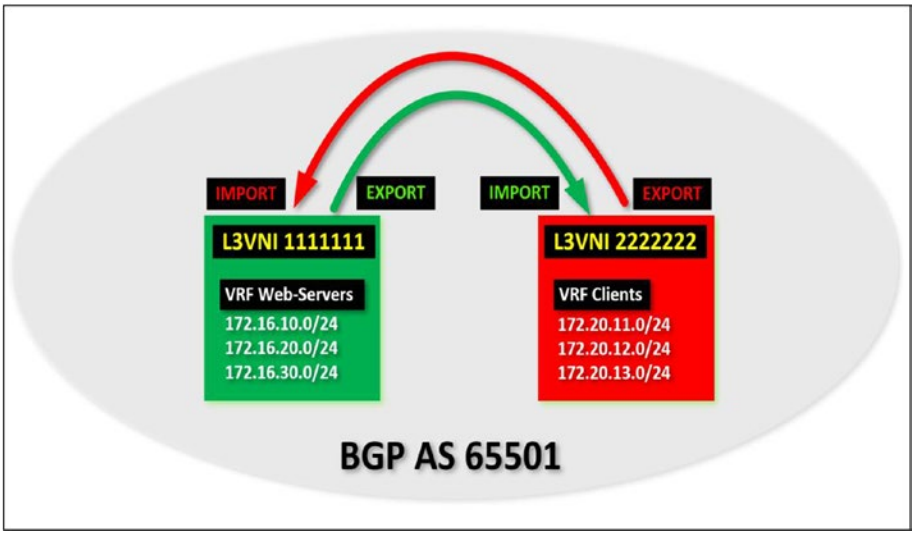


Figure 4-17. *Performing route leaking between two VRFs in the VXLAN fabric eliminates the need for an external routing device. Leverages the fabric underlay to perform the inter-VRF communication*

Let's configure a route leaking between both web servers and clients VRF. Once you execute the configurations, you should see a route to either VRF via the default VRF, which says it using the default route table in control plane BGP to advertise paths to both VRFs. Let's take a quick look at the configuration and then perform the verifications.

Route Leaking Configuration

Apply the following configuration to all leafs.

```
! VRF Clients

vrf context Clients
address-family ipv4 unicast
route-target import 65501:1111111
route-target import 65501:1111111 evpn
route-target export 65501:2222222
route-target export 65501:2222222 evpn

! VRF Web-Servers

vrf context Web-Servers
address-family ipv4 unicast
route-target import 65501:2222222
route-target import 65501:2222222 evpn
route-target export 65501:1111111
route-target export 65501:1111111 evpn
```

You need to advertise something in the BGP process so that it can be leaked to another VRF. If you don't have any routes, then no routes are leaked. Common sense, right? Let's redistribute our tenant VRF VLAN interfaces. You need to add a tag to all interfaces in the VRF. For the web servers VRF, I assign the tag 111, and for the client's VRF, I assign the tag of 222. Add the following configuration to all leafs!

```
interface Vlan10
  no shutdown
  vrf member Web-Servers
  ip address 172.16.10.1/24 tag 111
  fabric forwarding mode anycast-gateway

interface Vlan11
  no shutdown
  vrf member Clients
  ip address 172.20.11.1/24 tag 222
  fabric forwarding mode anycast-gateway

interface Vlan12
  no shutdown
  vrf member Clients
  ip address 172.20.12.1/24 tag 222
  fabric forwarding mode anycast-gateway

interface Vlan13
  no shutdown
  vrf member Clients
  ip address 172.20.13.1/24 tag 222
  fabric forwarding mode anycast-gateway

interface Vlan20
  no shutdown
  vrf member Web-Servers
  ip address 172.16.20.1/24 tag 111
  fabric forwarding mode anycast-gateway

interface Vlan30
  no shutdown
  vrf member Web-Servers
  ip address 172.16.30.1/24 tag 111
  fabric forwarding mode anycast-gateway
```

Let's create the route-map for each redistribution.

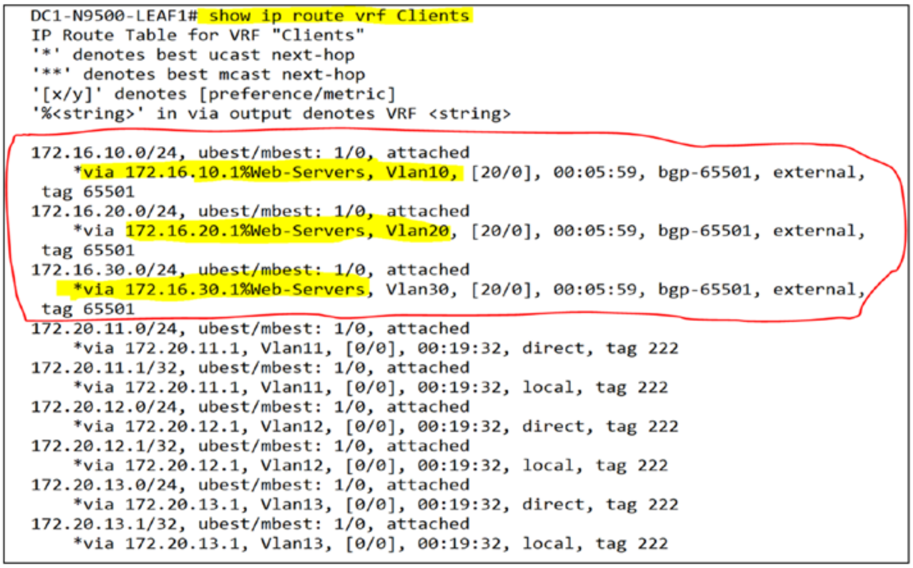
```
Route-map Clients-SVI permit 10
 match tag 222

Route-map WebServers-SVI permit 10
 match tag 111
```

Configure the redistribution on both VRFs. To redistribute interfaces, use the `direct` command after redistribute.

```
router bgp 65501

vrf Clients
address-family ipv4 unicast
redistribute direct route-map Clients-SVI

vrf Web-Servers
address-family ipv4 unicast
redistribute direct route-map WebServers-SVI
```

Verifying the Configuration

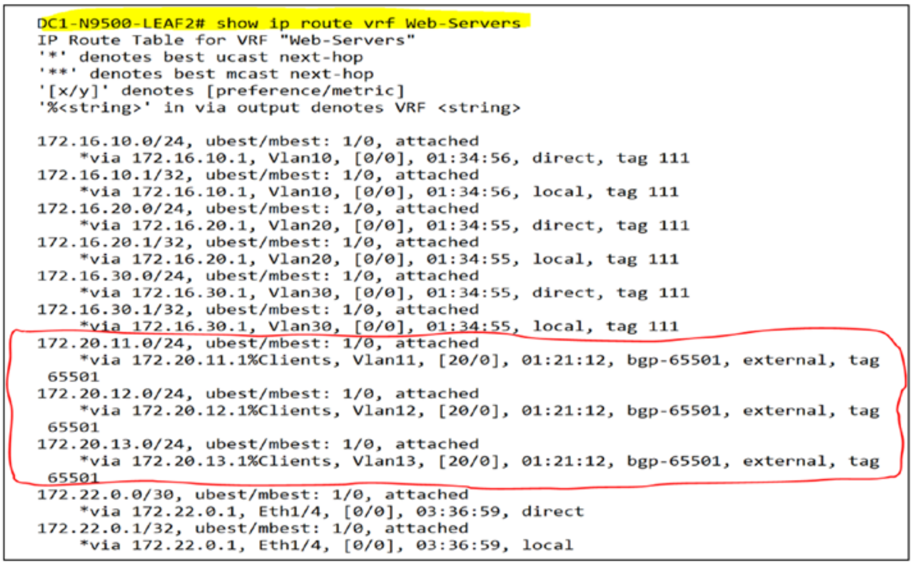
Now it's time to verify if you have successfully leaked all networks between VRFs. Use the `show ip route` command for each VRF to confirm if you can see the networks on the neighbor VRF advertised in the source VRF. You see show ip route vrf Clients in Figure 4-18.



```
DC1-N9500-LEAF1# show ip route vrf Clients
IP Route Table for VRF "Clients"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0, attached
    *via 172.16.10.1%Web-Servers, Vlan10, [20/0], 00:05:59, bgp-65501, external,
 tag 65501
172.16.20.0/24, ubest/mbest: 1/0, attached
    *via 172.16.20.1%Web-Servers, Vlan20, [20/0], 00:05:59, bgp-65501, external,
 tag 65501
172.16.30.0/24, ubest/mbest: 1/0, attached
    *via 172.16.30.1%Web-Servers, Vlan30, [20/0], 00:05:59, bgp-65501, external,
 tag 65501
172.20.11.0/24, ubest/mbest: 1/0, attached
    *via 172.20.11.1, Vlan11, [0/0], 00:19:32, direct, tag 222
172.20.11.1/32, ubest/mbest: 1/0, attached
    *via 172.20.11.1, Vlan11, [0/0], 00:19:32, local, tag 222
172.20.12.0/24, ubest/mbest: 1/0, attached
    *via 172.20.12.1, Vlan12, [0/0], 00:19:32, direct, tag 222
172.20.12.1/32, ubest/mbest: 1/0, attached
    *via 172.20.12.1, Vlan12, [0/0], 00:19:32, local, tag 222
172.20.13.0/24, ubest/mbest: 1/0, attached
    *via 172.20.13.1, Vlan13, [0/0], 00:19:32, direct, tag 222
172.20.13.1/32, ubest/mbest: 1/0, attached
    *via 172.20.13.1, Vlan13, [0/0], 00:19:32, local, tag 222
```

Figure 4-18. *Output of the client VRF after successfully performing a route leaking with the web servers VRF. The networks in the web servers VRF have been injected into the client VRF*

You see `show ip route VRF web servers` in Figure 4-19.



```
DC1-N9500-LEAF2# show ip route vrf Web-Servers
IP Route Table for VRF "Web-Servers"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0, attached
    *via 172.16.10.1, Vlan10, [0/0], 01:34:56, direct, tag 111
172.16.10.1/32, ubest/mbest: 1/0, attached
    *via 172.16.10.1, Vlan10, [0/0], 01:34:56, local, tag 111
172.16.20.0/24, ubest/mbest: 1/0, attached
    *via 172.16.20.1, Vlan20, [0/0], 01:34:55, direct, tag 111
172.16.20.1/32, ubest/mbest: 1/0, attached
    *via 172.16.20.1, Vlan20, [0/0], 01:34:55, local, tag 111
172.16.30.0/24, ubest/mbest: 1/0, attached
    *via 172.16.30.1, Vlan30, [0/0], 01:34:55, direct, tag 111
172.16.30.1/32, ubest/mbest: 1/0, attached
    *via 172.16.30.1, Vlan30, [0/0], 01:34:55, local, tag 111
172.20.11.0/24, ubest/mbest: 1/0, attached
    *via 172.20.11.1%Clients, Vlan11, [20/0], 01:21:12, bgp-65501, external, tag
 65501
172.20.12.0/24, ubest/mbest: 1/0, attached
    *via 172.20.12.1%Clients, Vlan12, [20/0], 01:21:12, bgp-65501, external, tag
 65501
172.20.13.0/24, ubest/mbest: 1/0, attached
    *via 172.20.13.1%Clients, Vlan13, [20/0], 01:21:12, bgp-65501, external, tag
 65501
172.22.0.0/30, ubest/mbest: 1/0, attached
    *via 172.22.0.1, Eth1/4, [0/0], 03:36:59, direct
172.22.0.1/32, ubest/mbest: 1/0, attached
    *via 172.22.0.1, Eth1/4, [0/0], 03:36:59, local
```

Figure 4-19. *The same result can be confirmed on the web servers VRF. Route leaking occurs from the client VRF to the web servers VRF. The client VRF networks are leaked into the web servers VRF*

VXLAN Fabric to External Network Connectivity

How can you allow a VXLAN fabric to talk to an external network? Which implementation options are available? This is what Chapter 5 is all about. When it comes to interfacing your fabric to an external network, designate your border leaf to perform this role. It doesn't matter if your external network is running dynamic route protocols or static routes. It still attaches to the border leaf since it's the entry point to your fabric. Let's explore and configure the available options.

BGP to OSPF Route Redistribution

An external campus network needs to communicate with the data center. The campus network is running OSPF, and you want to peer in OSPF, but your fabric propagates all tenant routes using BGP. You need to redistribute OSPF to BGP so you can learn all the campus side routes in your VXLAN fabric. There are three tenants, but only two should perform OSPF adjacency to the campus network. Please review the design specifics in Tables 5-1 and 5-2 and review the architecture diagram shown in Figure 5-1.

R. Cardona, *The Fast-Track Guide to VXLAN BGP EVPN Fabrics*,
https://doi.org/10.1007/978-1-4842-6930-5_5

Table 5-1. *Tenant-A*

Campus Network	VLAN ID	IP
User VLAN	10	10.0.0.1/23

Table 5-2. *Tenant-B*

Campus Network	VLAN ID	IP
User VLAN	15	10.10.0.1/23
Staff VLAN	20	10.20.0.1/23

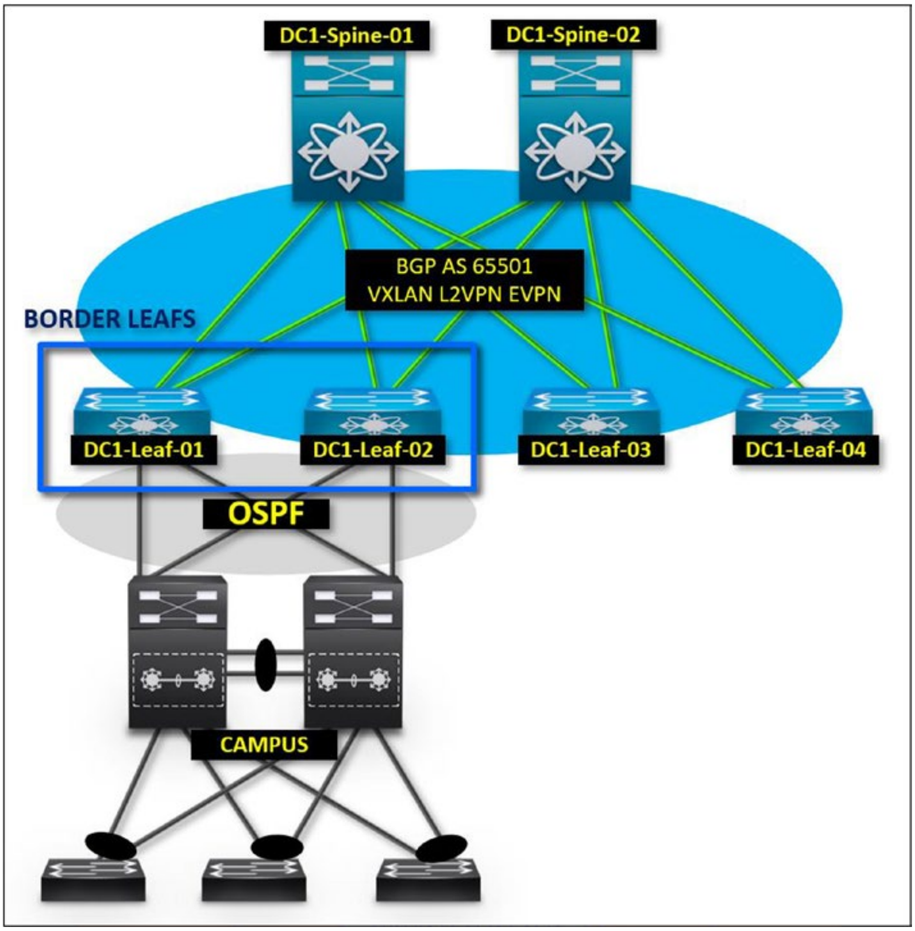


Figure 5-1. *A campus network is connected to border Leaf-01 and Leaf-02. The campus network is configured with two Catalyst 9000s in a virtual stack*

The goal is to get the campus core peered to the fabric border leafs. You want to provide access to both tenant-a and tenant-b, but you also want to avoid tenant-a from reaching tenant-b via the campus core. You want to filter the routes that are not shared between neighbors. Let's perform the configurations.

Campus Core

The campus core has a single MC-LAG or, in the Cisco world, a multi-chassis EtherChannel northbound to DC1-Leaf-01 and DC1-Leaf-02. Both tenants peer to the core switch using OSPF (see Tables 5-3 and 5-4). You use VLAN interfaces to peer OSPF and allow a designated "peering" VLAN ID for each tenant.

- Tenant-A – Leaf-01 VLAN 101
- Tenant-A – Leaf-02 VLAN 102
- Tenant-B – Leaf-01 VLAN 201
- Tenant-B – Leaf-02 VLAN 202

Table 5-3. *Tenant-A OSPF Links*

Tenant-A Leaf-01	**CORE**
172.16.101.1/30	172.16.101.2/30
Tenant-A Leaf-02	**CORE**
172.16.102.1/30	172.16.102.2/30

Table 5-4. *Tenant-B OSPF Links*

Tenant-B Leaf-01	**CORE**
172.16.201.1/30	172.16.201.2/30
Tenant-B Leaf-02	**CORE**
172.16.202.1/30	172.16.202.2/30

The first step is to create the OSPF process for each VRF tenant. You could also create a single OSPF process then nest the tenants under it. But I like to have a separate process per tenant. Let's configure Leaf-01 first (see Listing 5-1) and then Leaf-02 (see Listing 5-2).

Listing 5-1. DC1-Leaf-01

```
router ospf Tenant-A
 vrf Tenant-A
  router-id 172.16.101.1

router ospf Tenant-B
 vrf Tenant-B
  router-id 172.16.201.1

vlan 101
 name A-OSPF_CORE
 vn-segment 1000101

vlan 201
 name B-OSPF_CORE
 vn-segment 1000201

Int vlan 101
 vrf member Tenant-A
 ip router ospf Tenant-A area 0
 ip address 172.16.101.1/30
 fabric forwarding mode anycast-gateway
 no shut

Int vlan 201
 vrf member Tenant-B
 ip router ospf Tenant-B area 0
 ip address 172.16.201.1/30
 fabric forwarding mode anycast-gateway
 no shut
```

```
int nve 1
 member vni 1000101
  mcast-group 224.1.1.101
  suppress-arp
 member vni 1000201
  mcast-group 224.1.1.201
  suppress-arp

evpn
vni 1000101 l2
 rd auto
 route-target both auto
vni 1000201 l2
 rd auto
 route-target both auto
```

Listing 5-2. DC1-Leaf-02

```
router ospf Tenant-A
 vrf Tenant-A
  router-id 172.16.102.2

router ospf Tenant-B
 vrf Tenant-B
  router-id 172.16.202.2

vlan 102
 name A-OSPF_CORE
 vn-segment 1000102

vlan 202
 name B-OSPF_CORE
 vn-segment 1000202
```

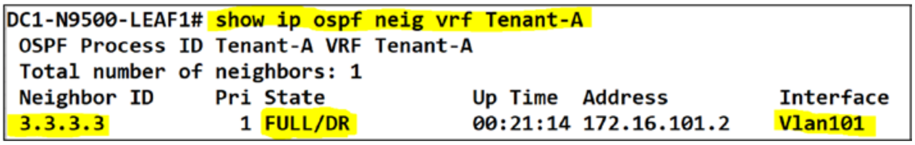
```
Int vlan 102
 vrf member Tenant-A
 ip router ospf Tenant-A area 0
 ip address 172.16.102.1/30
 fabric forwarding mode anycast-gateway
 no shut

Int vlan 202
 vrf member Tenant-B
 ip router ospf Tenant-B area 0
 ip address 172.16.202.1/30
 fabric forwarding mode anycast-gateway
 no shut

int nve 1
 member vni 1000102
  mcast-group 224.1.1.102
  suppress-arp
 member vni 1000202
  mcast-group 224.1.1.202
  suppress-arp

evpn
vni 1000102 l2
 rd auto
 route-target both auto
vni 1000202 l2
 rd auto
 route-target both auto
```

I've configured OSPF process IDs for both tenant-A and tenant-B. I've also allocated a set of VLANs in Leaf-01 and Leaf-02 to peer OSPF for both tenants. Now you need to confirm connectivity to the campus core Switch (SW). For your convenience, I'm providing the core configuration (see Listing 5-3).

Listing 5-3. Campus Core Configuration

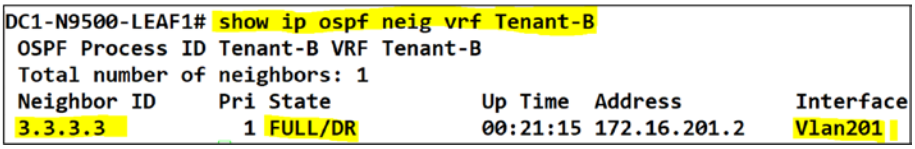
```
router ospf 1
router-id 3.3.3.3

vlan 101
name tenantA_Leaf01_OSPF
vlan 102
name TenantA_Leaf02_OSPF
vlan 201
name TenantB_Leaf01_OSPF
vlan 202
name TenantB_Leaf02_OSPF

int vlan 101
no shut
ip address 172.16.101.2 255.255.255.252
ip ospf 1 area 0

int vlan 102
no shut
ip address 172.16.102.2 255.255.255.252
ip ospf 1 area 0

int vlan 201
no shut
ip address 172.16.201.2 255.255.255.252
ip ospf 1 area 0

int vlan 202
no shut
ip address 172.16.202.2 255.255.255.252
ip ospf 1 area 0
```

After validating that all the required configurations are in place, let's confirm if you can see OSPF adjacency for both tenants on both leafs facing the core switch (see Figures 5-2, 5-3, 5-4, and 5-5).



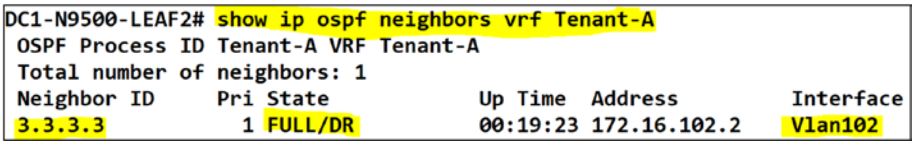
```
DC1-N9500-LEAF1# show ip ospf neig vrf Tenant-A
 OSPF Process ID Tenant-A VRF Tenant-A
 Total number of neighbors: 1
 Neighbor ID     Pri State            Up Time  Address         Interface
 3.3.3.3           1 FULL/DR          00:21:14 172.16.101.2    Vlan101
```

Figure 5-2. *DC1-Leaf-01 tenant-A*



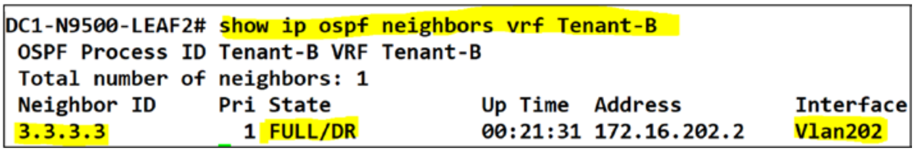
```
DC1-N9500-LEAF1# show ip ospf neig vrf Tenant-B
 OSPF Process ID Tenant-B VRF Tenant-B
 Total number of neighbors: 1
 Neighbor ID     Pri State            Up Time  Address         Interface
 3.3.3.3           1 FULL/DR          00:21:15 172.16.201.2    Vlan201
```

Figure 5-3. *DC1-Leaf-01 tenant-B*



```
DC1-N9500-LEAF2# show ip ospf neighbors vrf Tenant-A
 OSPF Process ID Tenant-A VRF Tenant-A
 Total number of neighbors: 1
 Neighbor ID     Pri State            Up Time  Address         Interface
 3.3.3.3           1 FULL/DR          00:19:23 172.16.102.2    Vlan102
```

Figure 5-4. *DC1-Leaf-02 tenant-A*



```
DC1-N9500-LEAF2# show ip ospf neighbors vrf Tenant-B
 OSPF Process ID Tenant-B VRF Tenant-B
 Total number of neighbors: 1
 Neighbor ID     Pri State            Up Time  Address         Interface
 3.3.3.3           1 FULL/DR          00:21:31 172.16.202.2    Vlan202
```

Figure 5-5. *DC1-Leaf-02 tenant-B*

You can confirm that the tenants on both Leaf-01 and Leaf-02 have OSPF adjacency to the campus core.

Now you can begin advertising the tenant networks in the campus network. The first thing that I did was redistribute the connected SVIs to OSPF. You can accomplish it by redistributing route tags in OSPF. Let's go ahead with the configuration. You configure the networks for each tenant, and then you configure the redistribution into OSPF. Apply to both leafs (see Listings 5-4 and 5-5).

Listing 5-4. Tenant-A

```
Vlan 10
 name USERS-Tenant-A
 vn-segment 1001010

int vlan 10
 description USERS
 vrf member Tenant-A
 ip address 10.0.0.1/23
 fabric forwarding mode anycast-gateway
 no shut

int nve 1
member vni 1001010
 mcast-group 224.1.10.10
 suppress-arp

evpn
 vni 1001010 l2
 rd auto
 route-target both auto

! Interface Redistribution to OSPF

int vlan 10
 ip address 10.0.0.1/23 tag 11
```

```
route-map Tenant-A-Vlans-OSPF permit 10
 match tag 11

router ospf Tenant-A
 vrf Tenant-A
 redistribute direct route-map Tenant-A-Vlans-OSPF
```

Listing 5-5. Tenant-B

```
Vlan 15
 name USERS-Tenant-B
 vn-segment 1001015
Vlan 20
Name Staff-Tenant-B
Vn-segment 1001120

int vlan 15
 description USERS
 vrf member Tenant-B
 ip address 10.10.0.1/23
 fabric forwarding mode anycast-gateway
 no shut

int vlan 20
 description STAFF
 vrf member Tenant-B
 ip address 10.20.0.1/23
 fabric forwarding mode anycast-gateway
 no shut

int nve 1
 member vni 1001015
  mcast-group 224.1.10.15
  suppress-arp
 member vni 1001120
  mcast-group 224.1.10.20
  suppress-arp

evpn
vni 1001015 l2
rd auto
route-target both auto
vni 1001120 l2
rd auto
route-target both auto
```

Interface Redistribution to OSPF

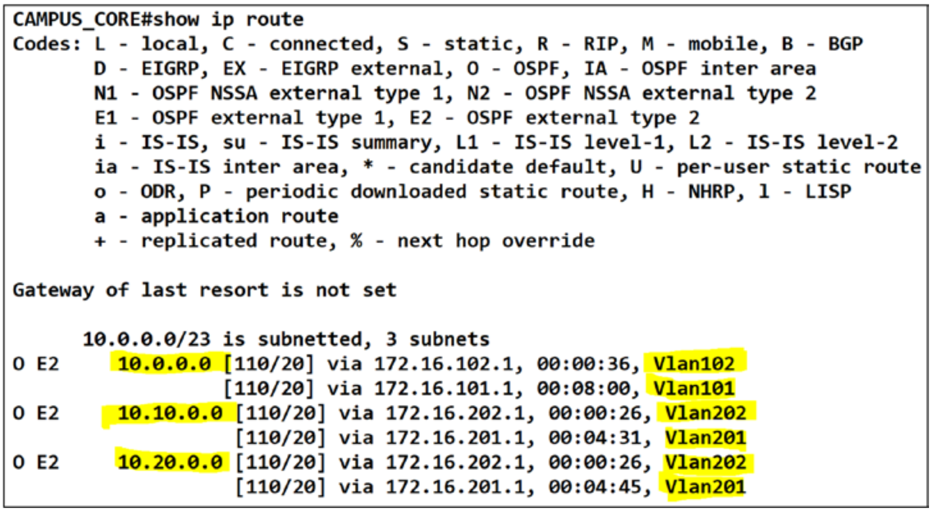
Let's redistribute in Listing 5-6.

Listing 5-6. After executing the command "*show ip route*" we can confirm route redistribution into OSPF from the VXLAN fabric

```
int vlan 15
 ip address 10.10.0.1/23 tag 22
int vlan 20
 ip address 10.20.0.1/23 tag 22

route-map Tenant-B-Vlans-OSPF permit 10
 match tag 22

router ospf Tenant-B
 vrf Tenant-B
 redistribute direct route-map Tenant-B-Vlans-OSPF
```

Let's confirm that the OSPF routes were built at the campus core switch from the VXLAN tenants.

Figure 5-6 shows the campus core route table.

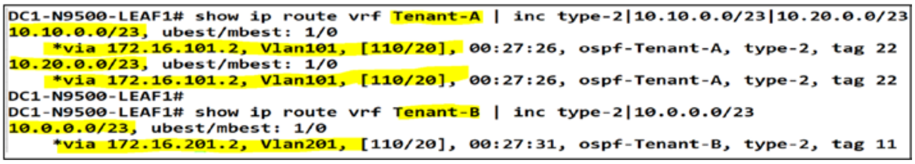


```
CAMPUS_CORE#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/23 is subnetted, 3 subnets
O E2     10.0.0.0 [110/20] via 172.16.102.1, 00:00:36, Vlan102
                  [110/20] via 172.16.101.1, 00:08:00, Vlan101
O E2     10.10.0.0 [110/20] via 172.16.202.1, 00:00:26, Vlan202
                   [110/20] via 172.16.201.1, 00:04:31, Vlan201
O E2     10.20.0.0 [110/20] via 172.16.202.1, 00:00:26, Vlan202
                   [110/20] via 172.16.201.1, 00:04:45, Vlan201
```

Figure 5-6. *Campus core route table*

The next step is to filter any tenant-A and tenant-B routes from being advertised between each other by using the campus core. Let's do some route filtering to accomplish that. Let's look at Leaf-01 and Leaf-02. The routes from tenant-B are shown in tenant-A and vice versa.

Figure 5-7 shows Leaf-01 tenant-A and tenant-B.



```
DC1-N9500-LEAF1# show ip route vrf Tenant-A | inc type-2|10.10.0.0/23|10.20.0.0/23
10.10.0.0/23, ubest/mbest: 1/0
    *via 172.16.101.2, Vlan101, [110/20], 00:27:26, ospf-Tenant-A, type-2, tag 22
10.20.0.0/23, ubest/mbest: 1/0
    *via 172.16.101.2, Vlan101, [110/20], 00:27:26, ospf-Tenant-A, type-2, tag 22
DC1-N9500-LEAF1#
DC1-N9500-LEAF1# show ip route vrf Tenant-B | inc type-2|10.0.0.0/23
10.0.0.0/23, ubest/mbest: 1/0
    *via 172.16.201.2, Vlan201, [110/20], 00:27:31, ospf-Tenant-B, type-2, tag 11
```

Figure 5-7. *Leaf-01 tenant-A and tenant-B*

The tenants can reach each other through the campus core switch. You need to avoid the tenant's from seeing each other. The campus core network is only allowed to reach both tenants.

Let's now add a route filter on Leaf-01 to filter incoming OSPF routes from tenants leaked in OSPF via the campus core. In summary, you create a route map with the "deny" action, and then match all the interfaces participating in the OSPF process.

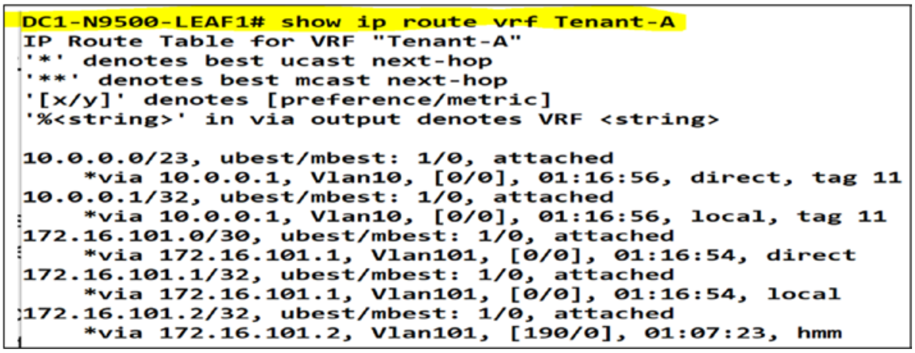
OSPF Inbound Route Filtering Configuration

Apply OSPF inbound route filtering configuration to both Leaf-01 and Leaf-02, as shown in Listing 5-7.

Listing 5-7. Without any route filtering in-place, Tenant-A and Tenant-B can reach each other via the campus core

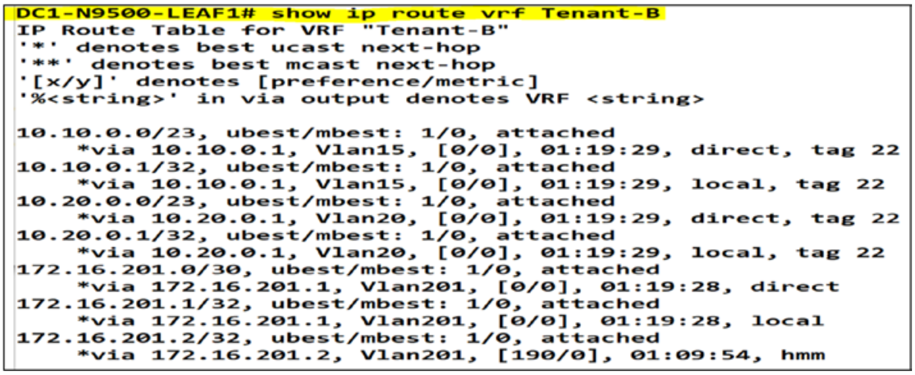
```
route-map Deny-OSPF-Routes deny 10it right to stop
  match interface Vlan101 Vlan102 Vlan201 Vlan202
router ospf Tenant-A
  vrf Tenant-A
    table-map Deny-OSPF-Routes filter
router ospf Tenant-B
  vrf Tenant-B
    table-map Deny-OSPF-Routes filter
```

Figure 5-8 shows the post-change result on Leaf-01 in tenant-A and tenant-B.



```
DC1-N9500-LEAF1# show ip route vrf Tenant-A
IP Route Table for VRF "Tenant-A"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.0.0.0/23, ubest/mbest: 1/0, attached
    *via 10.0.0.1, Vlan10, [0/0], 01:16:56, direct, tag 11
10.0.0.1/32, ubest/mbest: 1/0, attached
    *via 10.0.0.1, Vlan10, [0/0], 01:16:56, local, tag 11
172.16.101.0/30, ubest/mbest: 1/0, attached
    *via 172.16.101.1, Vlan101, [0/0], 01:16:54, direct
172.16.101.1/32, ubest/mbest: 1/0, attached
    *via 172.16.101.1, Vlan101, [0/0], 01:16:54, local
172.16.101.2/32, ubest/mbest: 1/0, attached
    *via 172.16.101.2, Vlan101, [190/0], 01:07:23, hmm
```

Figure 5-8. *Post-change result on Leaf-01 in tenant-A and tenant-B*



```
DC1-N9500-LEAF1# show ip route vrf Tenant-B
IP Route Table for VRF "Tenant-B"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.10.0.0/23, ubest/mbest: 1/0, attached
    *via 10.10.0.1, Vlan15, [0/0], 01:19:29, direct, tag 22
10.10.0.1/32, ubest/mbest: 1/0, attached
    *via 10.10.0.1, Vlan15, [0/0], 01:19:29, local, tag 22
10.20.0.0/23, ubest/mbest: 1/0, attached
    *via 10.20.0.1, Vlan20, [0/0], 01:19:29, direct, tag 22
10.20.0.1/32, ubest/mbest: 1/0, attached
    *via 10.20.0.1, Vlan20, [0/0], 01:19:29, local, tag 22
172.16.201.0/30, ubest/mbest: 1/0, attached
    *via 172.16.201.1, Vlan201, [0/0], 01:19:28, direct
172.16.201.1/32, ubest/mbest: 1/0, attached
    *via 172.16.201.1, Vlan201, [0/0], 01:19:28, local
172.16.201.2/32, ubest/mbest: 1/0, attached
    *via 172.16.201.2, Vlan201, [190/0], 01:09:54, hmm
```

Figure 5-9. *After Applying route filtering we can confirm that Tenant-A and Tenant-B can't reach each other*

You no longer see routes advertised from the campus core to the tenants. You need to be aware of an active OSPF peering to both Leaf-01 and Leaf-02. To avoid any asymmetric routing issues, you must make sure that only one OSPF path is active at a time. How do you accomplish this? First, you need to identify which leaf becomes the active path for the campus core. I chose Leaf-01 to become the active OSPF path to the campus network. On the campus core side, you need to configure OSPF costing to influence routes to be preferred to either OSPF neighbor, in this case, Leaf-01. Do a simple OSPF costing configuration that is lower on the VLAN interfaces peering to Leaf-01. VLANs 101 and 201 are the designated OSPF peering interfaces between Leaf-01 and the campus core. You'll see more of this in the next section.

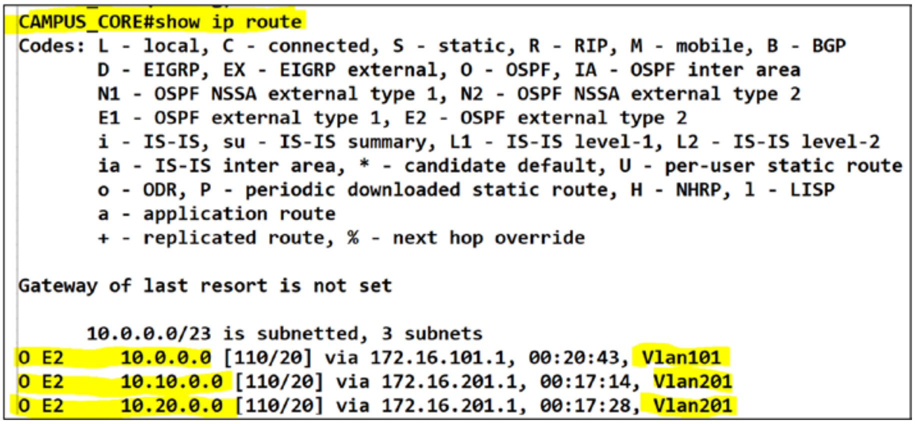
Listing 5-8 shows the campus core OSPF cost configuration.

Listing 5-8. Campus Core OSPF Cost Configuration

```
int vlan 101 (TO LEAF1)
ip ospf cost 1
int vlan 102 (TO LEAF2)
ip ospf cost 100
```

```
int vlan 201 (TO LEAF1)
ip ospf cost 1
int vlan 202 (TO LEAF2)
ip ospf cost 100
```

Adding the static OSPF cost values, you now confirm that only one active OSPF route is active for all advertised tenant networks. In my configuration, I set the lowest cost value possible to elect Leaf-01 as the primary path. If Leaf-01 goes down, Leaf-02 starts advertising its path. Let's verify (see Figure 5-10).



```
CAMPUS_CORE#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/23 is subnetted, 3 subnets
O E2     10.0.0.0 [110/20] via 172.16.101.1, 00:20:43, Vlan101
O E2     10.10.0.0 [110/20] via 172.16.201.1, 00:17:14, Vlan201
O E2     10.20.0.0 [110/20] via 172.16.201.1, 00:17:28, Vlan201
```

Figure 5-10. *By setting up OSPF costing on any layer 3 interface participating in OSPF we can pin traffic to be preferred via a specific path*

As you can see, there is one active route path via Leaf-01. Asymmetric routing issues are very common to face in VXLAN spines and leafs since the route paths are more elastic. I can route in many directions, but I still need to make sure that the traffic flow remains symmetric. In a nutshell, you send traffic one way, but the return traffic has a better path advertised via another leaf or so. This causes your asymmetric route flow issue.

A better way to explain it is by describing an asymmetric and symmetric flow sample. The diagrams presented in Figure 5-11 and Figure 5-12 are two examples of fabric efficiency. The diagram shown in Figure 5-11 is the asymmetric route issue using a firewall between the two tenants. Figure 5-12 is a symmetric route flow sample with the same scenario but traffic correctly pinned in one direction.

Asymmetric Route Issues in VXLAN BGP EVPN

Figure 5-11 describes each hop between a client tenant PC trying to communicate to the web servers tenant. By following each hop in the diagram, you see that the return traffic does not flow using the same path as the source request traffic from the client PC.

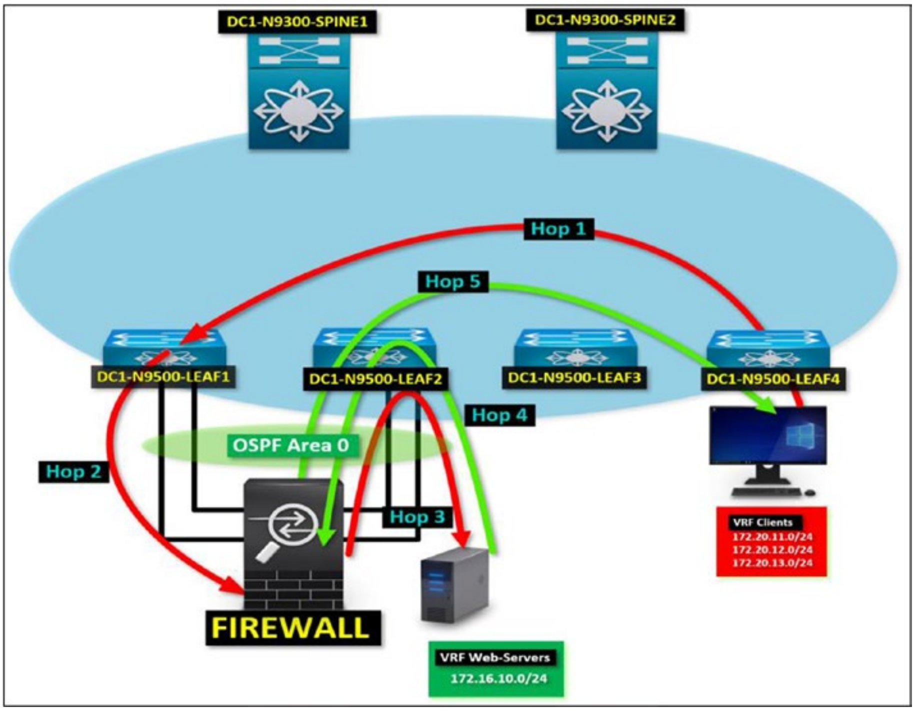


Figure 5-11. *In asymmetric routing issues, traffic does not return using the original path it came from. Due to many factors, mostly due to poor routing design and/or path selection*

- **HOP 1**: Client tenant PC sends a request to a web server. Leaf-04 sends the request to the VXLAN fabric. Traffic lands in Leaf-01 since it's the preferred path to the web servers tenant via EVPN.
- **HOP 2**: Leaf-01 sends the request to the external firewall since there's an OSPF route advertised to the web server tenant.
- **HOP 3**: Traffic leaves the firewall and lands in Leaf-02 since there's no preferred path to the web server's tenant. Leaf-02 sends the request to the web server's tenant.
- **HOP 4**: A server on the web server tenant replies to the request. Leaf-02 receives the request and forwards it to Leaf-04 instead of using the source path via the firewall peered in OSPF. The route to the client tenant is preferred over EVPN rather than OSPF.
- **HOP 5**: At this point, the request fails to return due to an asymmetric route. The return traffic uses a different path due to OSPF equal-cost path between Leaf-01 and Leaf-02.

Symmetric Routing in VXLAN BGP EVPN

By confirming a symmetric route path, you guarantee that all requests and replies follow the same route path. In the diagram shown in Figure 5-12, you see the same scenario covered in the asymmetric route discussion. The difference is that the server traffic returns to the client using the same route path. How do you accomplish this? By making sure that you only configure a lower cost or distance in the dynamic route protocol, you are implementing so the traffic returns in the same direction it came from.

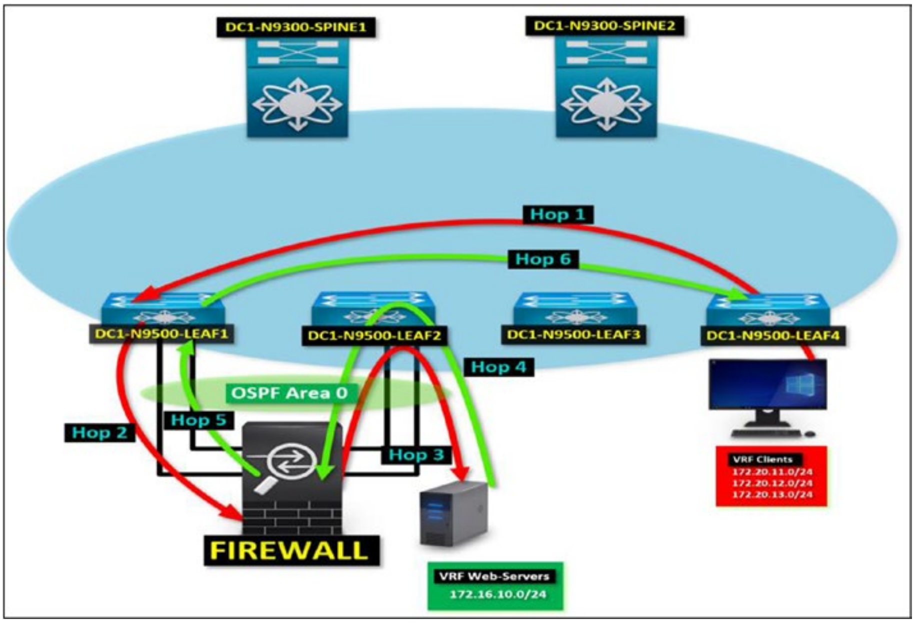


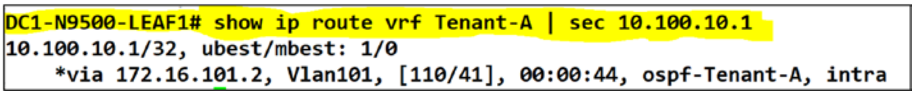
Figure 5-12. *A symmetric route path between two tenants behind an external firewall*

OSPF to BGP Route Redistribution

You can ensure that the traffic flows symmetrically by configuring OSPF costing or BGP distance (depending on which one you are using). Let's configure and advertise the VRFs in a symmetric path. Let's designate Leaf-01 to be the primary path to both VRFs. There is a network on the campus core that you want to reach. The address space is 10.100.10.0/24. All leafs should prefer Leaf-01 to reach this network. Leaf-02 should remain as a standby path to the destination of 10.100.10.0/24.

Routes Before Changes

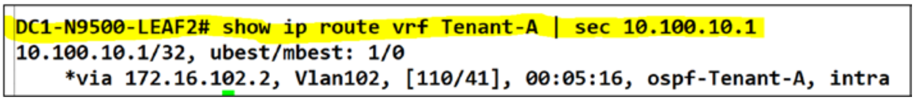
Figure 5-13 illustrates tenant-A VRF on Leaf-01.



```
DC1-N9500-LEAF1# show ip route vrf Tenant-A | sec 10.100.10.1
10.100.10.1/32, ubest/mbest: 1/0
    *via 172.16.101.2, Vlan101, [110/41], 00:00:44, ospf-Tenant-A, intra
```

Figure 5-13. *Tenant-A VRF on Leaf-01*

Figure 5-14 illustrates tenant-A VRF on Leaf-02.



```
DC1-N9500-LEAF2# show ip route vrf Tenant-A | sec 10.100.10.1
10.100.10.1/32, ubest/mbest: 1/0
    *via 172.16.102.2, Vlan102, [110/41], 00:05:16, ospf-Tenant-A, intra
```

Figure 5-14. *Tenant-A VRF on Leaf-02*

In Leaf-02, the preferred path is still local to the campus core. You want to re-route the traffic from all leafs to prefer Leaf-01 then Leaf-01 forward to the campus core. You need to fix this by redistributing OSPF to BGP and tweaking the BGP distance, so it is preferred via Leaf-01 using EVPN type-5 route advertisements.

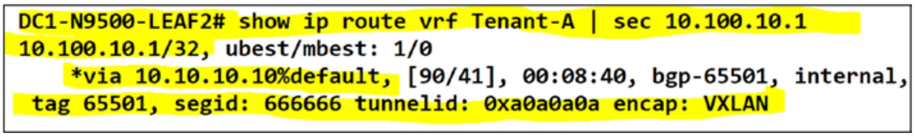
Redistribute OSPF to BGP EVPN on Leaf-01.

```
route-map OSPF-to-BGP permit 10
  match route-type internal external type-1 type-2

router bgp 65501
  vrf Tenant-A
    address-family ipv4 unicast
      redistribute ospf Tenant-A route-map OSPF-to-BGP
```

Modify BGP distance on Tenant-A in Leaf-02:

```
router bgp 65501

vrf Tenant-A
    address-family ipv4 unicast
      advertise l2vpn evpn
      distance 20 90 220
```

You have accomplished a few things. First, you redistributed all OSPF routes learned in Leaf-01 to the VXLAN fabric via BGP EVPN. Second, you modified the iBGP distance in Leaf-02, so routes advertised in BGP are preferred over OSPF. OSPF's AD is 110, and BGP is 200. I've lowered the BGP distance to 90, so it is preferred over OSPF. Let's validate route 10.100.10.0/24 in Leaf-02 (see Figure 5-15).



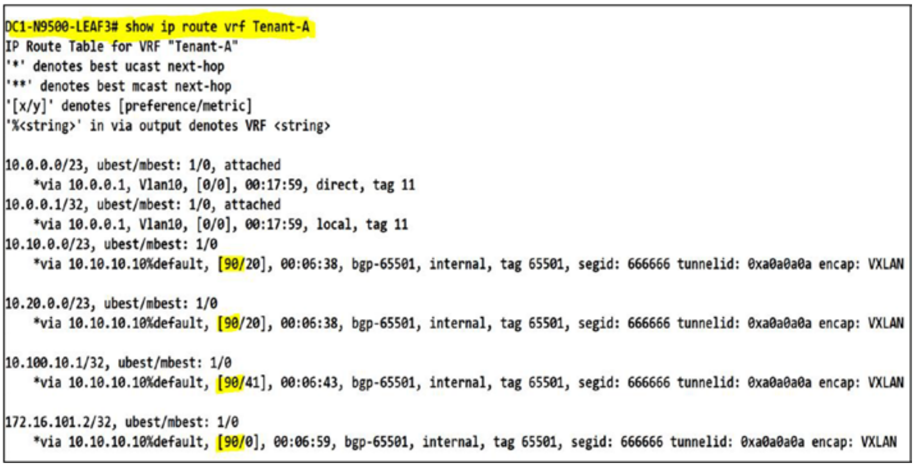
```
DC1-N9500-LEAF2# show ip route vrf Tenant-A | sec 10.100.10.1
10.100.10.1/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [90/41], 00:08:40, bgp-65501, internal,
 tag 65501, segid: 666666 tunnelid: 0xa0a0a0a encap: VXLAN
```

Figure 5-15. *Modifying the iBGP distance to a value less than the default OSPF distance will make iBGP a preferred route over OSPF*

Yes, you can confirm that tenant-A on Leaf-02 prefers to send any request destined for 10.100.10.0/24 via the L3VNI to Leaf-01 VTEP 10.10.10.10 Now you need to make sure that all the other leafs in the VXLAN fabric are following the same path. You need to modify each tenant VRF distance in BGP to be the same as Leaf-01 or a lower number than the default OSPF AD of 110. In my case, I changed iBGP from 200 to 90 as the administrative distance. Apply on all leafs to select iBGP as a preferred route over OSPF. Why do you still need to do this if you don't have OSPF adjacency on the other leafs? iBGP is always preferred for consistency, regardless of other route protocols. Apply the following configuration to the remaining leafs.

```
router bgp 65501

vrf Tenant-A
    address-family ipv4 unicast
      advertise l2vpn evpn
      distance 20 90 220
```

Let's apply it on Leaf-03 and confirm that the distance change is now reflected on the tenant-A VRF routes (see Figure 5-16).



```
DC1-N9500-LEAF3# show ip route vrf Tenant-A
IP Route Table for VRF "Tenant-A"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.0.0.0/23, ubest/mbest: 1/0, attached
    *via 10.0.0.1, Vlan10, [0/0], 00:17:59, direct, tag 11
10.0.0.1/32, ubest/mbest: 1/0, attached
    *via 10.0.0.1, Vlan10, [0/0], 00:17:59, local, tag 11
10.10.0.0/23, ubest/mbest: 1/0
    *via 10.10.10.10%default, [90/20], 00:06:38, bgp-65501, internal, tag 65501, segid: 666666 tunnelid: 0xa0a0a0a encap: VXLAN

10.20.0.0/23, ubest/mbest: 1/0
    *via 10.10.10.10%default, [90/20], 00:06:38, bgp-65501, internal, tag 65501, segid: 666666 tunnelid: 0xa0a0a0a encap: VXLAN

10.100.10.1/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [90/41], 00:06:43, bgp-65501, internal, tag 65501, segid: 666666 tunnelid: 0xa0a0a0a encap: VXLAN

172.16.101.2/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [90/0], 00:06:59, bgp-65501, internal, tag 65501, segid: 666666 tunnelid: 0xa0a0a0a encap: VXLAN
```

Figure 5-16. *You can confirm that after applying the config under Leaf-03, the preference for iBGP routes has changed to 90. It provides a consistent route path through the VXLAN fabric*

Dynamic Routing Protocols Under VRFs

Working with multitenant environments can bring a level of complexity that can make any network engineer frightened. It is why you must build every environment with consistency. Something as simple as your naming scheme can make a big difference when troubleshooting needs to happen. When you decide to peer VXLAN tenants to an external network using dynamic routing protocols like BGP, OSPF, EIGRP, or IS-IS, it can become quite a challenge to maintain consistency due to the number of extra configurations required to accomplish it.

Let's say you have a tenant called Italy, and another tenant called France. They both run services and host applications using your VXLAN fabric as a data transport. You want to have both tenants completely isolated. That's the sole reason for multitenancy. But also, all the relevant operational processes for each tenant should be isolated. Italy has requested to peer their tenant to an external MPLS network.

The service provider has provisioned a demarcation for this circuit, and it's ready to peer to the tenant on the fabric. The CE (customer edge) router holding the MPLS boundary is going to exchange routes using eBGP. The Italy network needs to reach the Italy tenant. You need to peer them to their MPLS using eBGP.

On the other side, France needs to peer its remote offices to their tenants on the fabric using OSPF. France has provided a router awaiting to establish an OSPF neighbor relationship.

You need to fulfill the request from both customers and make sure that other tenants aren't affected by this change. Remember its multitenant environment. You might have more than ten tenants using your fabric to run their hosted servers and applications. You don't want to disrupt any other tenant. You need to plan the configurations ahead of time and validate them to be 100 percent confident during the implementation.

The first thing you do is gather the information provided by the customer. From this information, you can then build the configuration script to be applied to the border leafs. Please find the information provided by the customer to fulfill the request.

Tenant: Italy

The following is the Italy tenant's peer information.

– MPLS L3VPN Router Interface: Gi0/1

– MPLS L3VPN Router Interface IP: 145.11.24.25

– MPLS L3VPN Router Interface Subnet 145.11.24.24/30

– Italy MPLS eBGP AS: 43321

– Italy VXLAN tenant BGP AS: 54423

– Networks announced from the MPLS cloud

- 172.21.10.0/24
- 172.21.20.0/24
- 172.21.30.0/24

You need to perform this configuration to receive the MPLS subnets on the Italy tenant VRF route table. **The ISP informed that if you peer to them using private AS numbers, you won't have an AS path or network reachability since those AS numbers are already in use in other locations.**

Now that you have all the information you need, let's enable BGP and OSPF under the tenants and allow connectivity to the external networks. Let's begin configuring the connection for Italy's tenant. Then you do the same for France. Please refer to the diagram in Figure 5-17 for the logical connectivity to the external sites.

When you enable tenants with dynamic routing protocols, you need to designate either an existing routing process or create a new one per tenant. I prefer to work with independent routing processes because if there is an issue with one process, the other tenants aren't impacted.

You already have BGP running for the VXLAN EVPN address family. You need to configure a BGP IPv4 address family for the Italy tenant. You need to add a unique BGP AS number for the Italy tenant. You need to use the same AS number to peer the tenants. That might be an issue since you don't control what tenants assign to their BGP community from AS number perspective. Instead, allow them as or override the AS number, so you don't have conflicts with any other possible BGP neighbor in the AS path using the same number.

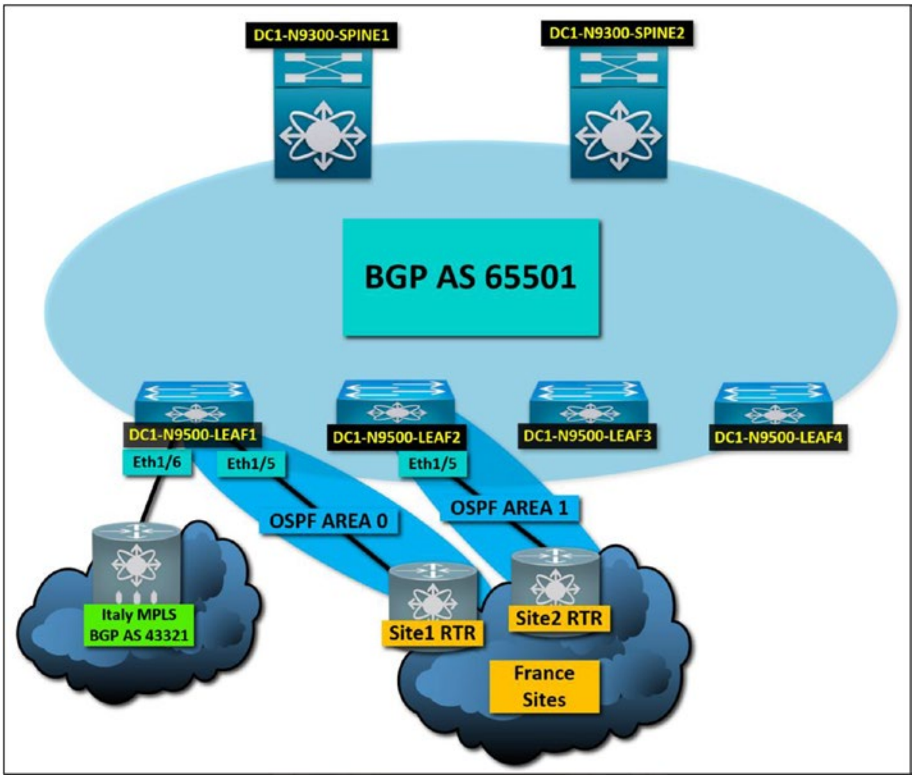


Figure 5-17. *A tenant to carrier connectivity diagram. The Italy MPLS peers to the border leaf*

Enabling BGP for a Tenant VRF

To enable BGP in a VRF, you need to add the VRF under the global BGP router process. Add the BGP neighbor and inform BGP to use an interface to source the hello message. You want to use the layer 3 interface that faces the MPLS router. Next, enable the AFI that you are working with. In our case, it is IPv4 unicast. With the information provided by the service provider, you need to establish adjacency to the MPLS router, which is connected to DC1-N9500-Leaf-01. Let's use the information provided and perform the configurations.

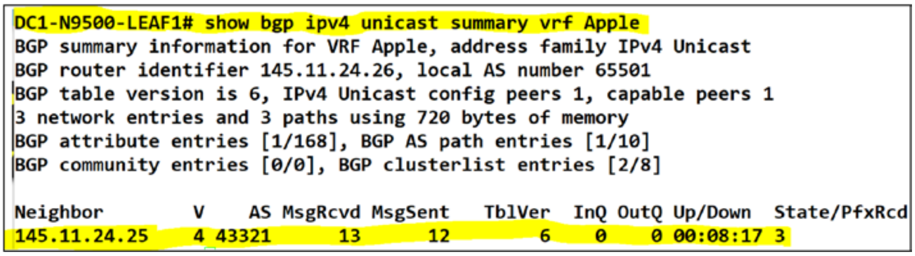
Listing 5-9 shows the Italy tenant BGP configuration on Leaf-01.

Listing 5-9. Italy Tenant BGP Configuration on Leaf-01

```
int eth1/6
no switchport
vrf member Italy
ip address 145.11.24.26/30
no shut

Router BGP 65501
  vrf Italy
    router-id 145.11.24.26
    address-family ipv4 unicast
    neighbor 145.11.24.25
      remote-as 43321
      local-as 54423
      update-source Ethernet1/6
      address-family ipv4 unicast
```

Now let's do the verifications. You want to first confirm that there is a BGP neighbor adjacency with the MPLS router. To perform this check, type in Leaf-01, as follows. It is illustrated it Figure 5-18.

```
show bgp ipv4 unicast summary vrf Italy
```

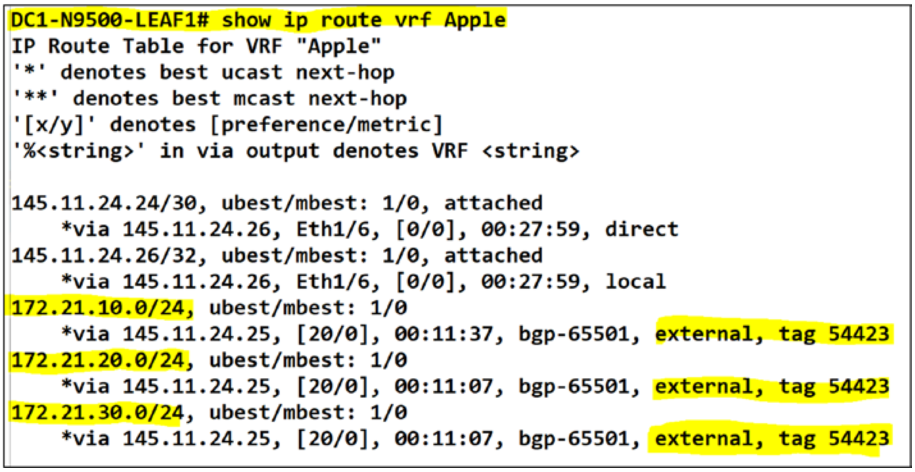


```
DC1-N9500-LEAF1# show bgp ipv4 unicast summary vrf Apple
BGP summary information for VRF Apple, address family IPv4 Unicast
BGP router identifier 145.11.24.26, local AS number 65501
BGP table version is 6, IPv4 Unicast config peers 1, capable peers 1
3 network entries and 3 paths using 720 bytes of memory
BGP attribute entries [1/168], BGP AS path entries [1/10]
BGP community entries [0/0], BGP clusterlist entries [2/8]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
145.11.24.25    4 43321      13      12        6    0    0 00:08:17 3
```

Figure 5-18. *A BGP adjacency to an external router in the tenant has been confirmed*

You can confirm adjacency to the MPLS network. The `show` command output indicates a neighbor of 145.11.24.25, which is the BGP router ID for the MPLS router. You also have received three prefixes if you look at the State/PfxRcd column. That should give us an indication that the remote networks are advertised on the Italy VRF route table. Let's confirm with the following command. It is illustrated it Figure 5-19.

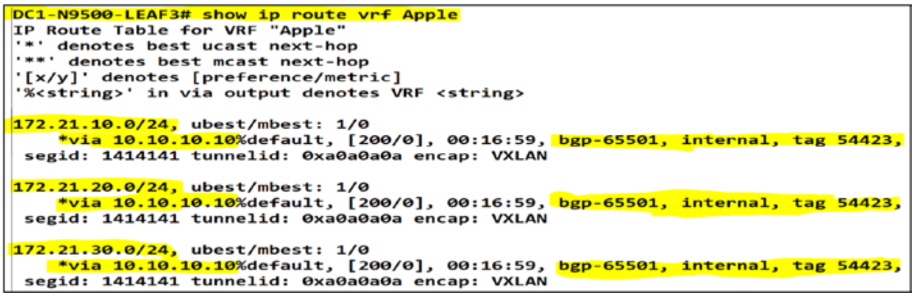
```
show ip route vrf Italy
```



```
DC1-N9500-LEAF1# show ip route vrf Apple
IP Route Table for VRF "Apple"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

145.11.24.24/30, ubest/mbest: 1/0, attached
    *via 145.11.24.26, Eth1/6, [0/0], 00:27:59, direct
145.11.24.26/32, ubest/mbest: 1/0, attached
    *via 145.11.24.26, Eth1/6, [0/0], 00:27:59, local
172.21.10.0/24, ubest/mbest: 1/0
    *via 145.11.24.25, [20/0], 00:11:37, bgp-65501, external, tag 54423
172.21.20.0/24, ubest/mbest: 1/0
    *via 145.11.24.25, [20/0], 00:11:07, bgp-65501, external, tag 54423
172.21.30.0/24, ubest/mbest: 1/0
    *via 145.11.24.25, [20/0], 00:11:07, bgp-65501, external, tag 54423
```

Figure 5-19. *Routes have been learned in the tenant from the external MPLS router once BGP adjacency is established*

The MPLS routes are advertised in the Italy tenant VRF, but you need to make sure that the routes are available across all leafs. If the L3VNI is properly configured and assigned to the Italy tenant, you should have the routes across the VXLAN fabric. Let's confirm if you see the routes in Leaf-03 (see Figure 5-20).



```
DC1-N9500-LEAF3# show ip route vrf Apple
IP Route Table for VRF "Apple"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.21.10.0/24, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/0], 00:16:59, bgp-65501, internal, tag 54423,
 segid: 1414141 tunnelid: 0xa0a0a0a encap: VXLAN

172.21.20.0/24, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/0], 00:16:59, bgp-65501, internal, tag 54423,
 segid: 1414141 tunnelid: 0xa0a0a0a encap: VXLAN

172.21.30.0/24, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/0], 00:16:59, bgp-65501, internal, tag 54423,
 segid: 1414141 tunnelid: 0xa0a0a0a encap: VXLAN
```

Figure 5-20. *The L3VNI interface in Leaf-03 is receiving the routes from Leaf-01 as EVPN route redistribution*

Amazing. EVPN is doing its job. The MPLS networks are distributed in L3VNI to the other leafs. The next hop is the VTEP IP for Leaf-01.

Enabling OSPF for a Tenant VRF

Enabling OSPF for a tenant VRF is quite similar to BGP. The only difference with OSPF is that you can create a new OSPF router process and assign it to the tenant. In BGP, you only run a single BGP process. Once you create the OSPF process, you add the tenant VRF under it and configure your OSPF router ID. Here's the catch. With BGP, you don't have to redistribute the external networks so EVPN can re-advertise to all my leafs. In OSPF, you need to perform redistribution to announce the networks to the tenant in all leafs. Okay, let's tackle the request to peer OSPF for the France tenant.

The diagram in Figure 5-21 should help you understand the connectivity layout for both sites. Site 1 router is homed to Leaf-01 via eth1/5. The site 2 router is homed to Leaf-02 via eth1/5. To be successful, all fabric leafs need to have advertised site 1 and site 2 networks in the France tenant VRF.

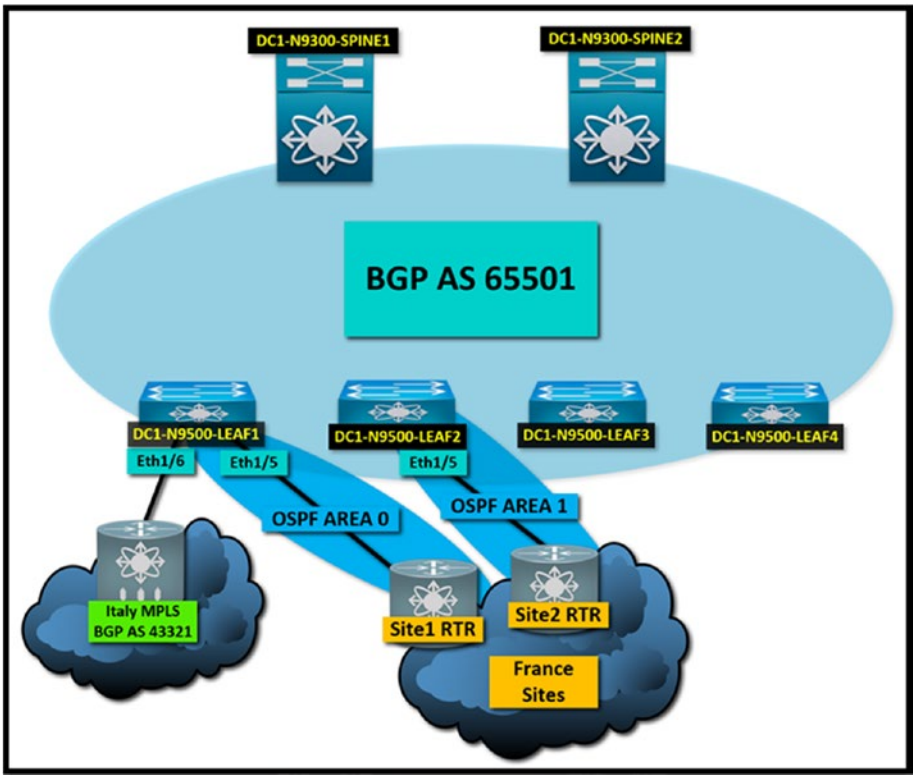


Figure 5-21. *A multitenant VXLAN fabric peered with OSPF and BGP to external neighbors. The tenants are completely unaware of each other's networks*

Tenant: France

The following is the France tenant's peer information.

- Site 1 – OSPF Area 0
- Site 1 Router Interface IP 172.16.1.1
- Site 1 Router Interface Subnet 172.16.1.0/30
- Site 2 – OSPF Area 1

– Site 2 Router Interface IP 172.16.2.1

– Site 2 Router Interface Subnet 172.16.2.0/30

– Networks received via OSPF from site 1

  - 172.21.10.0/24
  - 172.21.20.0/24
  - 172.21.30.0/24

– Networks received via OSPF from site 2

  - 172.22.10.0/24
  - 172.22.20.0/24
  - 172.22.30.0/24

Listing 5-10 shows the France tenant OSPF configuration on Leaf-01.

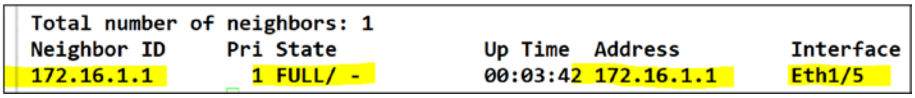
Listing 5-10. France Tenant OSPF Configuration on Leaf-01

```
Router ospf France
 vrf France
  router-id 172.16.1.2

Int eth1/5
 no shut
 vrf member France
  ip address 172.16.1.2/30
  ip router ospf France area 0
  ip ospf network point-to-point

! Redistribute OSPF to BGP for Tenant France

route-map France-OSPF permit 10
  match route-type internal external type-1 type-2
```

```
router bgp 65501
 vrf France
  address-family ipv4 unicast
   redistribute ospf France route-map France-OSPF
```

Let's apply the configurations to both Leaf-01 and Leaf-02 and verify if you are successfully receiving routes from site 1 and site 2. Let's confirm that you see the routes in the border leaf by using the following command. Then you should also see all the other leafs via EVPN (see Figure 5-22).

show ip ospf neighbors vrf France

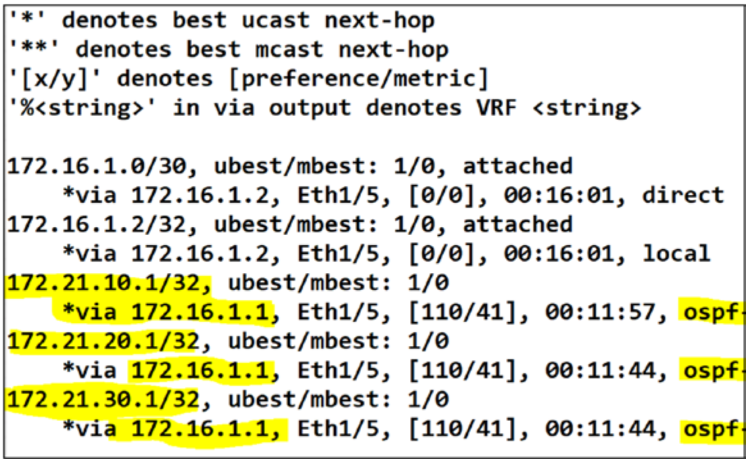


```
Total number of neighbors: 1
Neighbor ID     Pri State            Up Time  Address         Interface
172.16.1.1        1 FULL/ -          00:03:42 172.16.1.1      Eth1/5
```

Figure 5-22. *Site 1 to France tenant OSPF neighbor in Leaf-01*

You have established full OSPF adjacency to site 1's router 172.16.1.1. Now let's confirm if the routes were received from the site 1 router in the France tenant on Leaf-01 using the following command (see Figure 5-23).

show ip route vrf France

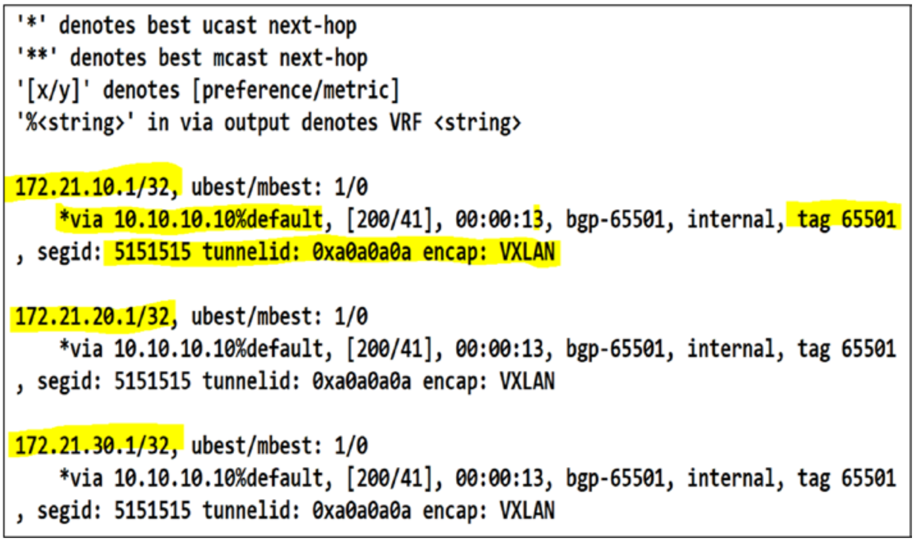


```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.1.0/30, ubest/mbest: 1/0, attached
    *via 172.16.1.2, Eth1/5, [0/0], 00:16:01, direct
172.16.1.2/32, ubest/mbest: 1/0, attached
    *via 172.16.1.2, Eth1/5, [0/0], 00:16:01, local
172.21.10.1/32, ubest/mbest: 1/0
    *via 172.16.1.1, Eth1/5, [110/41], 00:11:57, ospf
172.21.20.1/32, ubest/mbest: 1/0
    *via 172.16.1.1, Eth1/5, [110/41], 00:11:44, ospf
172.21.30.1/32, ubest/mbest: 1/0
    *via 172.16.1.1, Eth1/5, [110/41], 00:11:44, ospf
```

Figure 5-23. *The routes in the France tenant are available via OSPF*

You learned the routes from site 1 in the France tenant on the border leaf. Now you also performed a route redistribution from OSPF to BGP. You should have the routes learned fabric wide in the France tenant. Let's verify on Leaf-03 to confirm if this is the case using the following command (see Figure 5-24).

show ip route vrf France



```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.21.10.1/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/41], 00:00:13, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xa0a0a0a encap: VXLAN

172.21.20.1/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/41], 00:00:13, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xa0a0a0a encap: VXLAN

172.21.30.1/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/41], 00:00:13, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xa0a0a0a encap: VXLAN
```

Figure 5-24. *The routes from the tenant France on Leaf-01 are advertised via EVPN to the rest of the fabric. Leaf-03 has received the routes via the France VRF L3VNI*

Site 1 has been fully integrated into the VXLAN fabric. You have confirmed adjacency and route redistribution to all leafs. Let's do the same now for site 2, which connects to Leaf-02 (see Listing 5-11).

Listing 5-11. France Tenant OSPF Configuration on Leaf-02

```
Router ospf France
 vrf France
  router-id 172.16.2.2
```

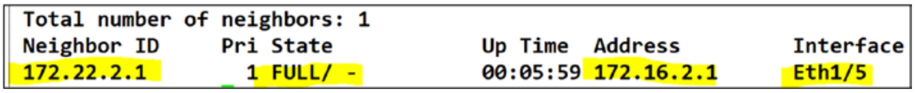
```
Int eth1/5
 No switchport
 no shut
 vrf member France
  ip address 172.16.2.2/30
  ip router ospf France area 1
  ip ospf network point-to-point
```

! Redistribute OSPF to BGP for Tenant France

```
route-map France-OSPF permit 10
  match route-type internal external type-1 type-2

router bgp 65501
 vrf France
  address-family ipv4 unicast
   redistribute ospf France route-map France-OSPF
```

Now that you applied the configurations on Leaf-02, let's confirm if you are receiving the routes from site 2. But first, using the following command, let's make sure that there is OSPF adjacency to site 2 (see Figure 5-25).

show ip ospf neighbors vrf France

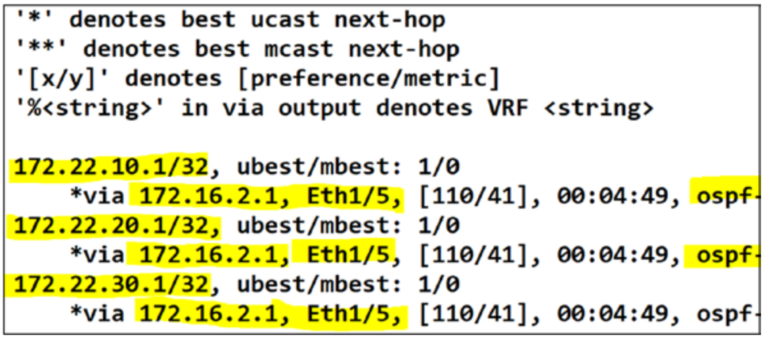


```
 Total number of neighbors: 1
 Neighbor ID     Pri State            Up Time  Address        Interface
 172.22.2.1        1 FULL/ -          00:05:59 172.16.2.1     Eth1/5
```

Figure 5-25. Site 2 to France tenant OSPF neighbor in Leaf-02

After confirming OSPF adjacency to site 2 via Leaf-02 on the France tenant, let's verify if the routes were received from site 2 by using the following command (see Figure 5-26).

show ip route ospf-France vrf France

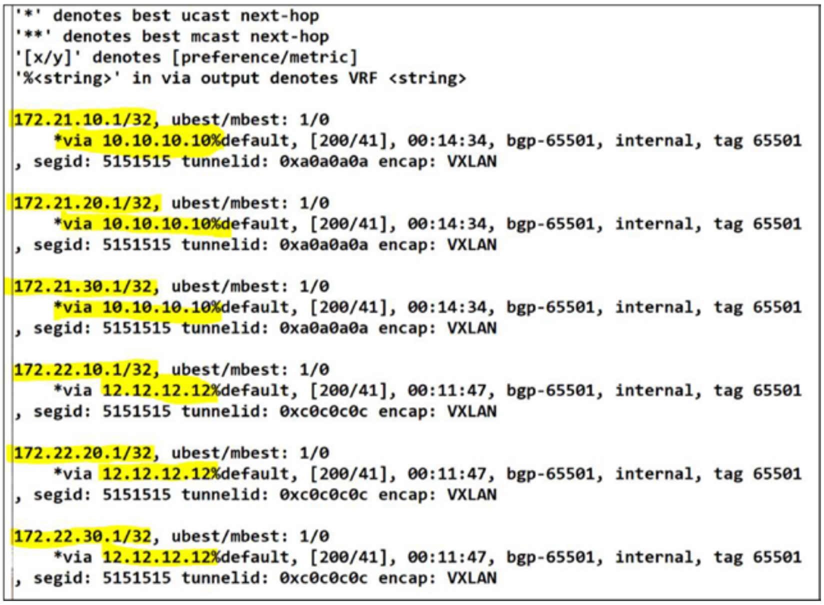


```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.22.10.1/32, ubest/mbest: 1/0
    *via 172.16.2.1, Eth1/5, [110/41], 00:04:49, ospf-
172.22.20.1/32, ubest/mbest: 1/0
    *via 172.16.2.1, Eth1/5, [110/41], 00:04:49, ospf-
172.22.30.1/32, ubest/mbest: 1/0
    *via 172.16.2.1, Eth1/5, [110/41], 00:04:49, ospf-
```

Figure 5-26. *We can confirm that the OSPF routes in Leaf-02 are received from the site 2 router*

Yes, routes are being received from site 2 to the France tenant. Finally, since you performed route redistribution from OSPF to BGP, let's confirm that other leafs have received the routes via Leaf-02 L3VNI in EVPN.

Let's confirm in Leaf-03 (see Figure 5-27).



```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.21.10.1/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/41], 00:14:34, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xa0a0a0a encap: VXLAN

172.21.20.1/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/41], 00:14:34, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xa0a0a0a encap: VXLAN

172.21.30.1/32, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/41], 00:14:34, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xa0a0a0a encap: VXLAN

172.22.10.1/32, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/41], 00:11:47, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xc0c0c0c encap: VXLAN

172.22.20.1/32, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/41], 00:11:47, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xc0c0c0c encap: VXLAN

172.22.30.1/32, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/41], 00:11:47, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xc0c0c0c encap: VXLAN
```

Figure 5-27. *The routes learned via Leaf-02 are successfully advertised in EVPN to the other leafs in the fabric*

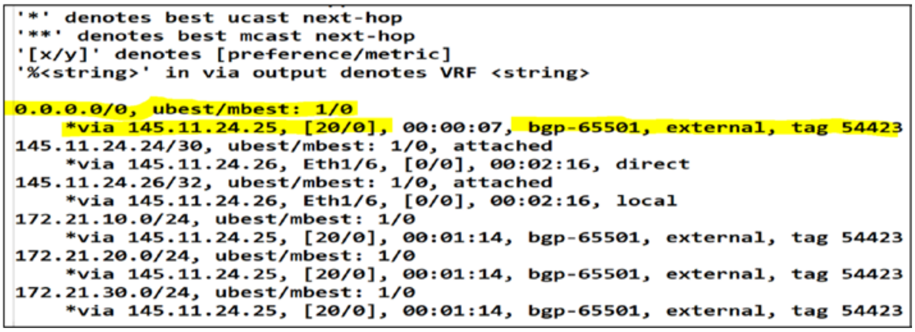
Excellent! You can confirm that you have the routes from site 1 via EVPN to the L3VNI for Leaf-01 (10.10.10.10) and the routes from site 2 via EVPN to for Leaf-02's L3VNI. The France tenant has end-to-end reachability to both sites across the entire VXLAN fabric. Success!

Default-Route Advertisement

How do you manage a default route in VXLAN BGP EVPN? In traditional campus networks, you usually have a border router or firewall connected to the edge of your network. When it comes to a spine-and-leaf topology, you usually connect this edge device between two leafs. You either statically assign a default route on those two leafs, or you get a default-route dynamically advertised from the edge device itself. But what about the other leafs. Do they also get it? No, unless you perform a default originate in the BGP IPv4 unicast address family on the tenant that receives it.

This section is about redistributing or re-advertising a default route in VXLAN BGP EVPN across the fabric for a tenant. Taking advantage of the previous exercise, let's work with the same topology. Both France and Italy have requested to send all Internet-bound traffic via their remote networks. Italy would like to send from the tenant, traffic to the Internet via the MPLS circuit. France would like to send all Internet traffic to site 2. Let's make sure that the default gateway is available across the Italy and France tenants' fabric.

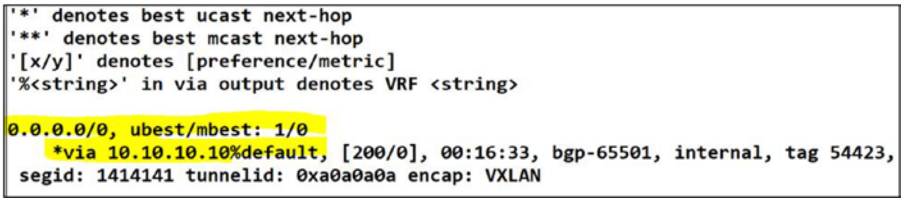
You should already have a default route on Leaf-01 for the Italy tenant, since Leaf-01 is connected to the MPLS router and should receive the route via BGP (see Figure 5-28).



```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

0.0.0.0/0, ubest/mbest: 1/0
    *via 145.11.24.25, [20/0], 00:00:07, bgp-65501, external, tag 54423
145.11.24.24/30, ubest/mbest: 1/0, attached
    *via 145.11.24.26, Eth1/6, [0/0], 00:02:16, direct
145.11.24.26/32, ubest/mbest: 1/0, attached
    *via 145.11.24.26, Eth1/6, [0/0], 00:02:16, local
172.21.10.0/24, ubest/mbest: 1/0
    *via 145.11.24.25, [20/0], 00:01:14, bgp-65501, external, tag 54423
172.21.20.0/24, ubest/mbest: 1/0
    *via 145.11.24.25, [20/0], 00:01:14, bgp-65501, external, tag 54423
172.21.30.0/24, ubest/mbest: 1/0
    *via 145.11.24.25, [20/0], 00:01:14, bgp-65501, external, tag 54423
```

Figure 5-28. *Italy tenant Leaf-01 route table*

You can confirm that you are receiving the default route from the MPLS router. Now let's confirm if the route was redistributed to all the other leafs (see Figure 5-29).



```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

0.0.0.0/0, ubest/mbest: 1/0
    *via 10.10.10.10%default, [200/0], 00:16:33, bgp-65501, internal, tag 54423,
 segid: 1414141 tunnelid: 0xa0a0a0a encap: VXLAN
```

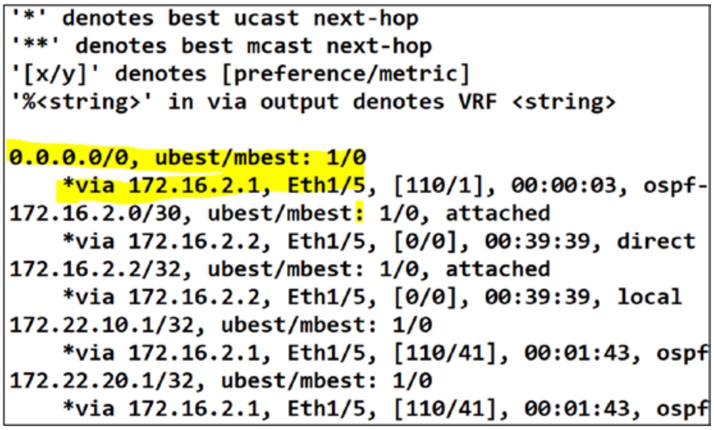
Figure 5-29. *The default route has been learned from Leaf-01 in EVPN. The rest of the fabric will use this route as last resort thru the L3VNI*

Yes, the route has been redistributed. If you want to redistribute this route to another external network from the fabric—simple from the leaf that connects to the external network you need to perform the following BGP configuration.

```
Router bgp 65501
 vrf Italy
 address-family ipv4 unicast
  default-information originate
  network 0.0.0.0/0
```

There are two important commands. The `default-information originate` command tells the routing protocol to advertise a default route in its process. By adding this command to a static default route, you tell the process to redistribute the default route via its neighbors. The second command is `network 0.0.0.0/0`. In BGP, redistributing a default route is much more than the `default-originate` command. BGP expects a network statement of the default route, which is 0.0.0.0/0; otherwise, it does not advertise it.

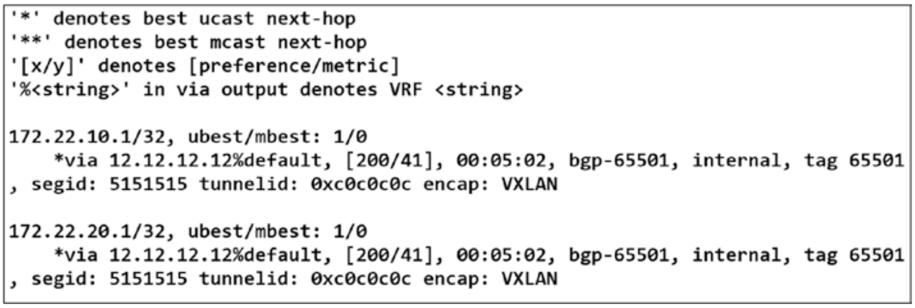
Let's now work with the France tenant. In Leaf-02 in the France tenant, you should already see a default route learned via OSPF. Let's confirm it (see Figure 5-30).



```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

0.0.0.0/0, ubest/mbest: 1/0
    *via 172.16.2.1, Eth1/5, [110/1], 00:00:03, ospf-
172.16.2.0/30, ubest/mbest: 1/0, attached
    *via 172.16.2.2, Eth1/5, [0/0], 00:39:39, direct
172.16.2.2/32, ubest/mbest: 1/0, attached
    *via 172.16.2.2, Eth1/5, [0/0], 00:39:39, local
172.22.10.1/32, ubest/mbest: 1/0
    *via 172.16.2.1, Eth1/5, [110/41], 00:01:43, ospf
172.22.20.1/32, ubest/mbest: 1/0
    *via 172.16.2.1, Eth1/5, [110/41], 00:01:43, ospf
```

Figure 5-30. *France tenant Leaf-02 route table*

You have the default route injected from site 2 via OSPF in Leaf-02. But let's check on Leaf-03 to see if the same default route was advertised (see Figure 5-31).



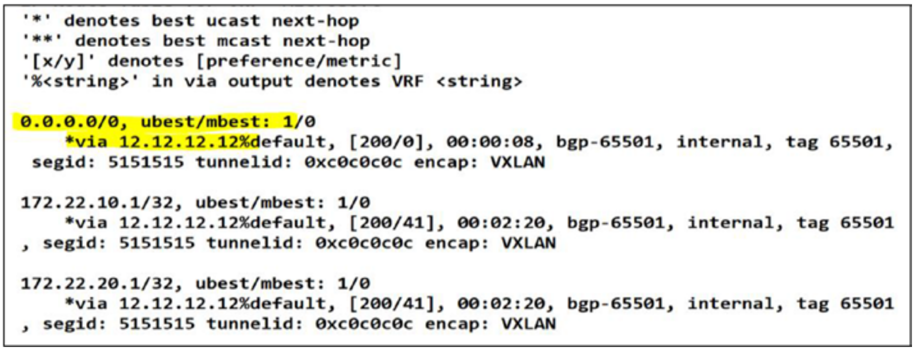
```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.22.10.1/32, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/41], 00:05:02, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xc0c0c0c encap: VXLAN

172.22.20.1/32, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/41], 00:05:02, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xc0c0c0c encap: VXLAN
```

Figure 5-31. *An additional configuration is required under BGP to readvertise a default route to all peer routers or in this case Leafs*

There is a problem. You don't have the default route redistributed to BGP. While you can see the routes from site 2 networks, you can't see the default route redistributed. Let's fix it! You need to add the same commands I showed you from the Italy tenant to the BGP process in the France VRF. It needs to be done on the border leaf that you got from the default route advertised on site 2.

```
Router bgp 65501
 vrf France
 address-family ipv4 unicast
  default-information originate
  network 0.0.0.0/0
```

After applying the configurations, let's verify Leaf-03 once again (see Figure 5-32).



```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

0.0.0.0/0, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/0], 00:00:08, bgp-65501, internal, tag 65501,
 segid: 5151515 tunnelid: 0xc0c0c0c encap: VXLAN

172.22.10.1/32, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/41], 00:02:20, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xc0c0c0c encap: VXLAN

172.22.20.1/32, ubest/mbest: 1/0
    *via 12.12.12.12%default, [200/41], 00:02:20, bgp-65501, internal, tag 65501
, segid: 5151515 tunnelid: 0xc0c0c0c encap: VXLAN
```

Figure 5-32. *After applying the configuration changes we can confirm a successful default route advertisement in BGP*

Success! You have redistributed the default route from site 2 to all leafs in the France tenant. Mission accomplished!

CHAPTER 6

VXLAN Fabric Topology Designs

You understand how VXLAN BGP EVPN works and how to configure all its major features. Let's now explore available deployment designs. Each design is deployed to serve a specific operational mode in your VXLAN BGP EVPN fabric. When you reviewed the traditional three-tier architecture, you had the core layer, the distribution or aggregation layer, and finally, the access layer. From an L3 routing standpoint, the core layer performs this function.

In a VXLAN spine-and-leaf architecture, the leafs are performing all the tenant networks routing decisions and operations. However, VXLAN in BGP EVPN brings a more security-centric deployment design. There are **single pod**, **multipod**, and **multisite** VXLANs.

Another deployment design not commonly used but also available is **head-end replication** without spines. Spines allow expansion in the topology with an east to west direction and aggregate bandwidth among the leafs using ECMP. I already discussed the single pod VXLAN, which you have been configuring throughout this book—a single spine-and-leaf architecture. Now let's discuss a multipod VXLAN.

Multipod VXLAN Architecture

A customer has two data centers. Data center 1 is located in Dallas, TX, while data center 2 is located in Los Angeles, CA. The customer is looking at the possibility of distributing compute workload between the two locations. If you need to perform virtual machine migrations from DC1 to DC2, you must present the VLANs from which the VMs communicate from. That means that you must somehow "stretch" from DC1 to DC2 every VLAN required to perform the virtual machine migrations. The customer approves the migration from the traditional three-tier architecture to VXLAN BGP EVPN.

You need to present the same VLANs from data center 1 to data center 2. By implementing a multisite VXLAN fabric, you create two identically configured VXLAN fabrics and share VXLAN advertisements to look like one big fabric.

In terms of data center configuration differences, a multipod makes the architecture look like a single large fabric. Having DC1 as pod 1 and DC2 as pod 2, you can tailor VLANs exclusively for each site and have stretched or shared VLANs between the two pods because they operate like one single VXLAN fabric from an EVPN route input and output. How does the architecture look from a logical view? Please review the diagram shown in Figure 6-1.

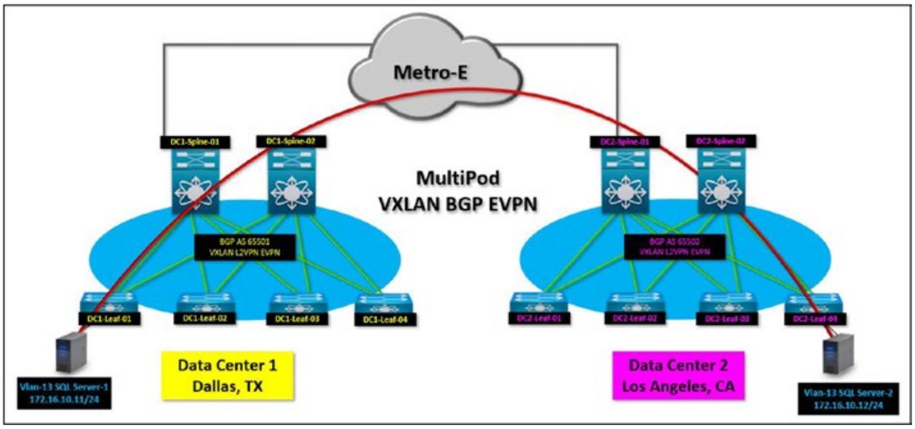


Figure 6-1. *A multipod VXLAN BGP EVPN fabric, two data centers with a stretched VLAN, and a server in DC1 on the same VLAN as a server in DC2. From a compute standpoint, it thinks it is on the same network*

Note The deployment options for a multipod vary. This diagram performs intra-data center connectivity or DCI (Data Center Interconnect) via the spines on both data centers. Another approach to perform the DCI is using transit leafs. The configurations to perform the DC-to-DC connectivity are reviewed next.

Multipod Integration

Multiple configuration options are available to enable a multifabric VXLAN multipod integration. The end goal is to build a logically stretched VXLAN fabric between geographically distant locations. When I say *logically stretched VXLAN fabric,* I mean that although there are two separate data centers with designated spines and leafs, you can configure them so that they operate like a single VXLAN fabric. This is known as a **single data plane VXLAN architecture**. There are multiple ways of performing the VXLAN fabric integration. You can peer both fabrics using the SPINE as the data center–to–data center hop. The spines advertise EVPN information in and out (see Figure 6-2).

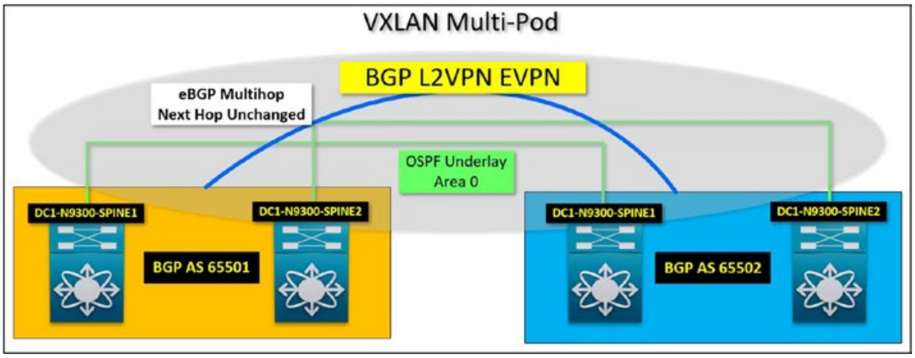


Figure 6-2. *A multipod VXLAN fabric configured using the spines as the transit between data centers*

Another option is to use transit leafs for this purpose. Transit leafs cross-connect from one data center to another and perform EVPN redistribution between data centers. Both spine-to-spine and leaf-to-leaf are valid EVPN transits, depending on the fabric configurations if you designate the transit leafs.

It is configured similarly to the spine-to-spine. You need to configure an IGP for the underlay (OSPF) and the BGP control-plane EVPN, which peers between the two data centers. A spine-to-spine or a leaf-to-leaf option still performs the same EVPN route-target import and export between VRFs (see Figure 6-3).

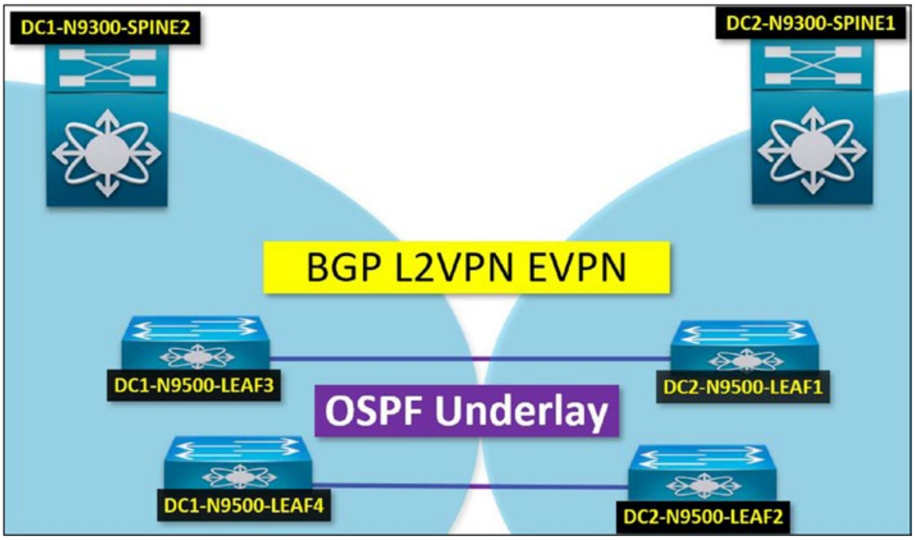


Figure 6-3. *A multipod VXLAN fabric design using transit leafs between data centers*

Configuring a Multipod VXLAN Fabric

Let's assume that you are deploying a multipod VXLAN fabric with two data centers. There is a single L2 circuit between DC1 and DC2. Let's configure a spine-to-spine multipod EVPN Transit path. Please find the high-level steps and required configurations to enable the multipod transit.

Step 1: Configure the Underlay Link Between DC1 and DC2 on Spine-01

The underlay link is the physical point-to-point L2 circuit link between DC1 and DC2. You configure it like any other leaf to spine underlay link and enable IGP; in this case, OSPF (see Listings 6-1 and 6-2).

Listing 6-1. DC1-N9300-Spine-01

```
interface Ethernet1/3
  description To_DC2-N9300-SPINE1-eth1/3
  no switchport
  mtu 9216
  medium p2p
  no ip redirects
  ip address 100.200.0.1/30
  no ipv6 redirects
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown
```

Listing 6-2. DC2-N9300-Spine-01

```
interface Ethernet1/3
  description TO_DC1-N9300-SPINE1-eth1/3
  no switchport
  mtu 9216
  medium p2p
  no ip redirects
  ip address 100.200.0.2/30
  no ipv6 redirects
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown
```

A logical configuration diagram is shown in Figure 6-4.

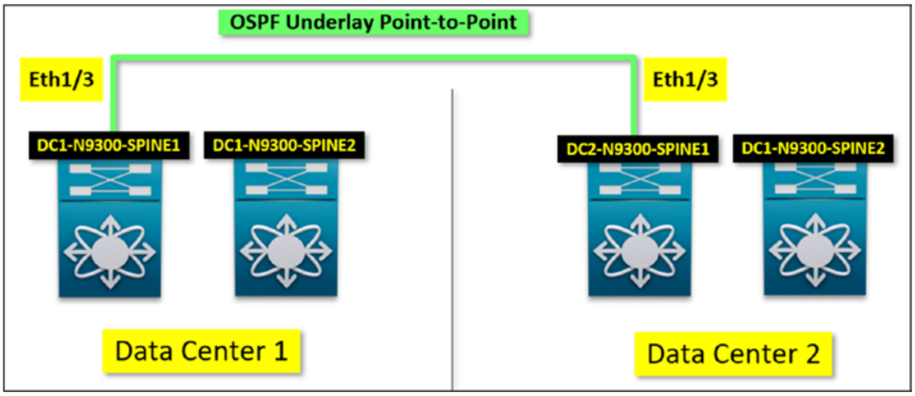


Figure 6-4. *Step 1 configuration enables the underlay between DC1 and DC2 to bring EVPN overlay between data centers*

Step 2: Configure the BGP EVPN Multihop Peering Between Spines

You need to add the required configuration in the BGP process so DC1 and DC2 can bring EVPN adjacency up between the spines. You also need to make sure that the next hop IP doesn't change as traffic passes the spine layer (see Listings 6-3 and 6-4).

Listing 6-3. DC1-N9300-Spine-01

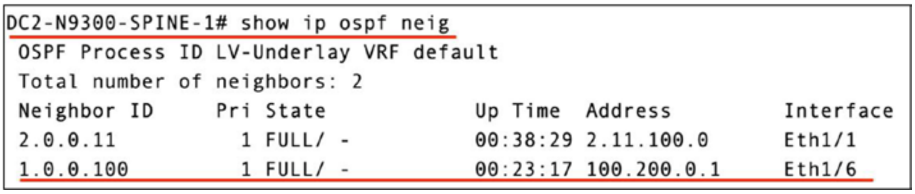
```
! Create and configure the route-map to set ip next hop to
unchanged. Apply this to both ! DC1 and DC2 Spine.
route-map DC1-to-DC2-MultiPod-VXLAN permit 10
 set ip next-hop unchanged
! On DC1 Spine-01 configure the BGP neighbor for DC2 configs.
Apply the route-map under the L2VPN EVPN AFI.
  neighbor 2.0.0.100
    remote-as 65502
    update-source loopback0
    ebgp-multihop 10
    address-family ipv4 unicast
    address-family l2vpn evpn
      send-community
      send-community extended
      route-map DC1-to-DC2-MultiPod-VXLAN out
```

Listing 6-4. DC2-N9300-Spine-01

```
route-map DC1-to-DC2-MultiPod-VXLAN permit 10
 set ip next-hop unchanged

  neighbor 1.0.0.100
    remote-as 65501
    update-source loopback0
    ebgp-multihop 10
    address-family ipv4 unicast
    address-family l2vpn evpn
      send-community
      send-community extended
      route-map DC1-to-DC2-MultiPod-VXLAN out
```

Once the configuration has been completed, you can verify that the OSPF and BGP neighbors are established for DC1 Spine-01 and DC2 Spine-01.

Let's confirm OSPF adjacency (see Figure 6-5).



```
DC2-N9300-SPINE-1# show ip ospf neig
 OSPF Process ID LV-Underlay VRF default
 Total number of neighbors: 2
 Neighbor ID     Pri State            Up Time  Address         Interface
 2.0.0.11          1 FULL/ -          00:38:29 2.11.100.0      Eth1/1
 1.0.0.100         1 FULL/ -          00:23:17 100.200.0.1     Eth1/6
```

Figure 6-5. *After executing "show ip ospf neighbor" we can confirm proper underlay adjacency to DC1-N9300-Spine-01*

From DC2, there is OSPF adjacency to DC1 Spine-01. Let's now verify if BGP EVPN peering is established with DC1 (see Figure 6-6).

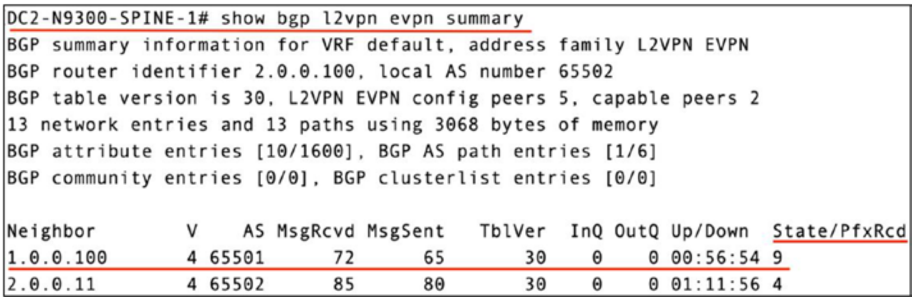

```
DC2-N9300-SPINE-1# show bgp l2vpn evpn summary
BGP summary information for VRF default, address family L2VPN EVPN
BGP router identifier 2.0.0.100, local AS number 65502
BGP table version is 30, L2VPN EVPN config peers 5, capable peers 2
13 network entries and 13 paths using 3068 bytes of memory
BGP attribute entries [10/1600], BGP AS path entries [1/6]
BGP community entries [0/0], BGP clusterlist entries [0/0]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
1.0.0.100       4 65501      72      65       30    0    0 00:56:54 9
2.0.0.11        4 65502      85      80       30    0    0 01:11:56 4
```

Figure 6-6. *We can confirm that evpn routes are being received from DC1-N9300-Spine-01*

Based on the output, you can confirm that BGP EVPN adjacency to DC1 via Spine-01 was established. You have received EVPN prefixes from DC1. Now let's confirm that there is NVE peering on Leaf-01 at both data centers (see Listing 6-5).

Listing 6-5. We can confirm that the VXLAN NVE interface has peered to VTEP 21.21.21.21

```
DC1-N9500-LEAF-1# show ip route
IP Route Table for VRF "default"

1.0.0.11/32, ubest/mbest: 2/0, attached
    *via 1.0.0.11, Lo0, [0/0], 01:18:01, local
    *via 1.0.0.11, Lo0, [0/0], 01:18:01, direct
1.0.0.100/32, ubest/mbest: 1/0
    *via 1.11.100.1, Eth1/1, [110/41], 01:17:03, ospf-Underlay,
     intra
2.0.0.11/32, ubest/mbest: 1/0
    *via 1.11.100.1, Eth1/1, [110/121], 01:03:15, ospf-
     Underlay, intra
2.0.0.100/32, ubest/mbest: 1/0
```

```
    *via 1.11.100.1, Eth1/1, [110/81], 01:03:15, ospf-Underlay,
     intra
11.11.11.11/32, ubest/mbest: 2/0, attached
    *via 11.11.11.11, Lo1, [0/0], 01:18:02, local
    *via 11.11.11.11, Lo1, [0/0], 01:18:02, direct
21.21.21.21/32, ubest/mbest: 1/0
    *via 2.0.0.100, [200/0], 01:03:12, bgp-65501, internal, tag
     65502

DC1-N9500-LEAF-1# show nve peers
Interface Peer-IP          State LearnType Uptime   Router-Mac
--------- ---------------  ----- --------- -------- -----------
------
nve1      21.21.21.21      Up    CP        01:06:18
                                           5000.0006.0007
```

There is NVE peering to DC2-N9500-Leaf-01 from DC1-N9500-Leaf-01. This confirms a successful BGP EVPN multipod communication.

EVPN L3VNI Routing in Multipod VXLAN

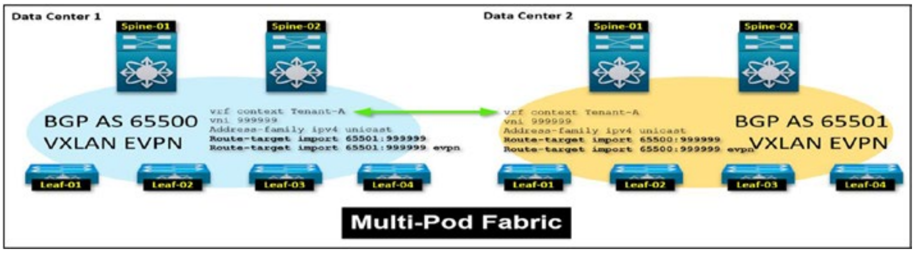
The multipod fabric is up. Data center 1 and data center 2 are fully integrated into a multipod architecture. Fifty percent of the multipod configuration is complete. Next, you need to stretch the tenants between data centers. If you have customer A in DC1, the same customer A should be available in DC2. For that, you must perform the route-target import and export.

If you want to refresh your mind with route-target import and export, please go to Chapter 3 and review the EVPN route-target section. In VXLAN with BGP EVPN, the L3VNI assigned to the VRF transport inter-VLAN traffic in the VXLAN fabric between leafs. The same L3VNI needs to be stretched between data centers. Let's say you have tenant-A in DC1, and the information in Table 6-1 is the current information regarding its tenant network.

***Table 6-1.** VXLAN BGP EVPN Tenant-A Information to Configure on Both Data Centers*

Name	Tenant-A
L3VNI	999999
L2VNI's	100010, 100020

The first thing you must do to allow inter-VLAN routing between data centers is to import the L3VNI information in the BGP control plane, advertised in the respective VRF. Once this is in place, you should start getting EVPN route distribution through the entire multipod domain. Please refer to the diagram shown in Figure 6-7 for a logical view of this setup.



```
vrf context Tenant-A
vni 999999
Address-family ipv4 unicast
Route-target import 65501:999999
Route-target import 65501:999999 evpn
```

```
vrf context Tenant-A
vni 999999
Address-family ipv4 unicast
Route-target import 65500:999999
Route-target import 65500:999999 evpn
```

***Figure 6-7.** With route-target import and export we can import routes from the adjacent VXLAN pod and export our routes outbound to the adjacent VXLAN pod*

EVPN L3VNI Multipod Configuration

Following the diagram shown in Figure 6-7, perform the required configurations to allow tenant-A to leverage both data centers in a multipod VXLAN (see Listing 6-6).

Listing 6-6. DC1 Configuration (All Leafs)

```
vrf context Tenant-A
vni 999999
Address-family ipv4 unicast
route-target import 65501:999999
route-target import 65501:999999 evpn
```

What is the configuration doing in DC1? If you look at the route-target value, there are two import actions. The first import action allows DC1 to import any IPv4 BGP AFI route from DC2 to DC1. The second route-target action is to import any EVPN type–based routes from DC2 to DC1. It means that any learned L2 address over EVPN, any route-redistribution in EVPN is advertised to the neighbor fabric or member pod in the multipod domain.

The value assigned to the RTs, 65501:999999, is arbitrary. In my case, I labeled it **ASN:L3VNI** to make it more consistent and easier to understand. ASN 65501 is the assigned iBGP autonomous system number to data center 2 and 999999 is the L3VNI value for tenant-A. Let's configure DC2. As you should already know, you need to change the ASN value from 65501 to 65500 (DC1 ASN); everything else stays the same (see Listing 6-7).

Listing 6-7. DC2 Configuration (All Leafs)

```
vrf context Tenant-A
vni 999999
address-family ipv4 unicast
route-target import 65500:999999
route-target import 65500:999999 evpn
```

The diagram in Figure 6-8 illustrates what the configuration is doing traffic-wise.

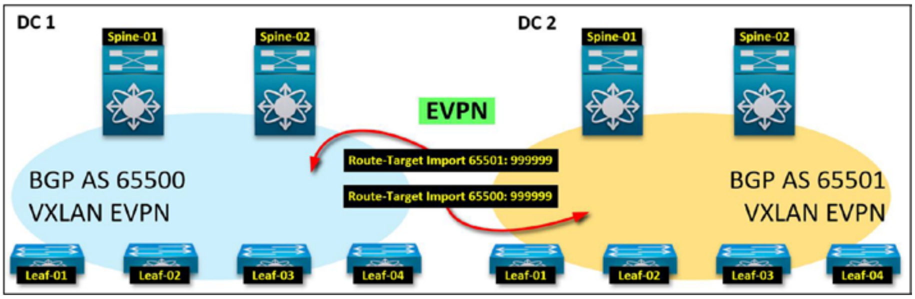


Figure 6-8. *A multipod L3VNI route-target import for both DC1 and DC2. By performing this configuration, you effectively import EVPN and BGP RTs between data centers*

With the configuration in place, you can expect L3 information learned from either data center to be distributed to the remote data center. VXLAN operation would look like a single data center meaning that a leaf-to-leaf VXLAN traffic flow use NVE peering like it happens on a standalone VXLAN fabric.

Multisite VXLAN Architecture

A multisite VXLAN architecture is a cluster of VXLAN fabrics communicating specific traffic over an external IP network. Each fabric operates independently; however, they are configured to send or receive specific EVPN traffic. You use an external network (service provider) to establish communication among all sites. In a multisite architecture, independent VXLAN fabrics are sharing only specific traffic in the EVPN control plane.

Let’s look at the diagram in Figure 6-9. There are two sites: site 1 is in Frankfurt, and site 2 is in Munich. They are communicating using a service provider cloud peered in eBGP.

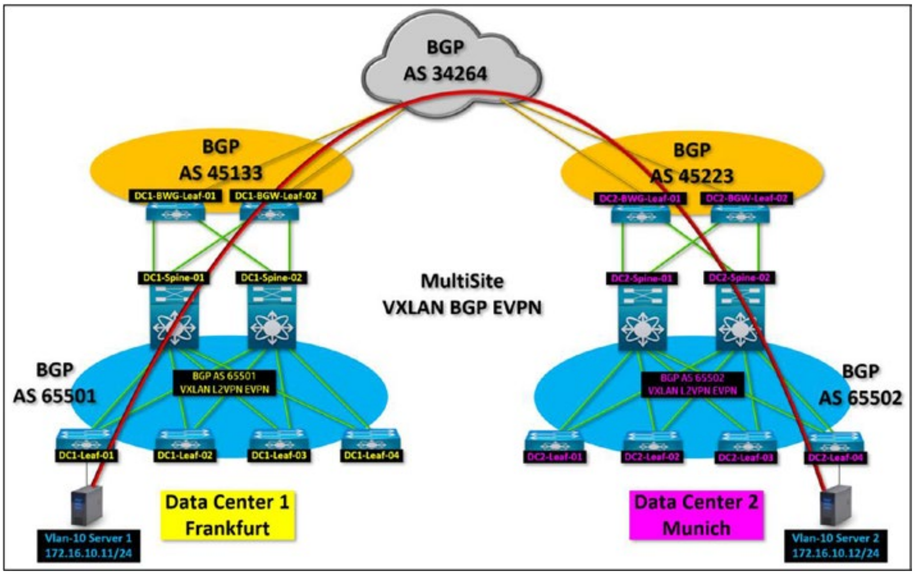


Figure 6-9. *In a multisite VXLAN BGP EVPN, you allow independent VXLAN fabrics to share specific traffic over a service provider network backbone. Sites are independent of each other*

Knowing this information, on each VXLAN fabric, you add a device that allows the communication and the exchange from the external BGP (service provider) to the internal BGP (VXLAN fabric) and vice versa. It is called a **border gateway leaf** or BGW.

Border Gateway Leaf or BGW

BGW allows communication and adjacency between the local VXLAN fabric and the external transit network (service provider). EVPN traffic is forwarded to the transit IP network to reach its destination fabric using the border gateway. It also announces adjacent VXLAN fabrics from a participating site in the "cluster". The local VXLAN fabric sees that it knows EVPN route types via an eBGP advertisement.

The cool thing about the BGW is that in BGP, it rewrites the ASN number as the traffic egresses to the backbone network. That means that if resources from the Frankfurt VXLAN fabric need to communicate to the VXLAN fabric in Munich as the traffic egresses, the BGW in Frankfurt advertises information changes from the internal 65501 ASN to an external 45133 ASN. Once the traffic reaches the remote site, the BGW at the remote site changes the eBGP ASN to an iBG P65502 ASN. It is fooling the remote VXLAN fabric into thinking that the source is locally present.

There are a couple of items to take into consideration. An additional configuration to the BGW links beside the underlay aspect is that you must enable a specific role to the uplinks participating in VXLAN. You must specify which link goes to the spines and which links go to the service provider. Another important item with a BGW is assigning an ID that is the designated site ID in the multisite cluster. Let’s say Frankfurt has site ID 100 and Munich has site ID 200.

The following configures the Frankfurt site ID on the Frankfurt BGW.

```
evpn multisite border-gateway 100
```

The following configures the Munich site ID on the Munich BGW.

```
evpn multisite border-gateway 200
```

EVPN Multisite Fabric Tracking

The `evpn multisite fabric-tracking` command informs the BGW that the interface ID facing the internal EVPN fabric. The border gateway then understands and applies the required action to the traffic that ingress or egresses from that interface. A multisite architecture provides the required information from the local fabric to the BGW to get it propagated to any other remote VXLAN fabric.

EVPN Multisite Data Center Interconnect Tracking

The `evpn multisite dci-tracking` command informs the BGW that the interface where it is applied is bound to the service provider network. DCI stands for data center interconnect, and that is it. It allows interconnection between data center fabrics located in remote sites. BGP uses that interface as a source to peer to the external service provider network. BGP source hello originate from this interface to the service provider or external network.

Ultimately, traffic incoming and outgoing from this interface is strictly eBGP. It has already been assigned the external BGP AS number from the local BGW. In Frankfurt, this would be from 65501 to 45133. It is great to talk about it, but it's better to see the command in action in Figure 6-10.

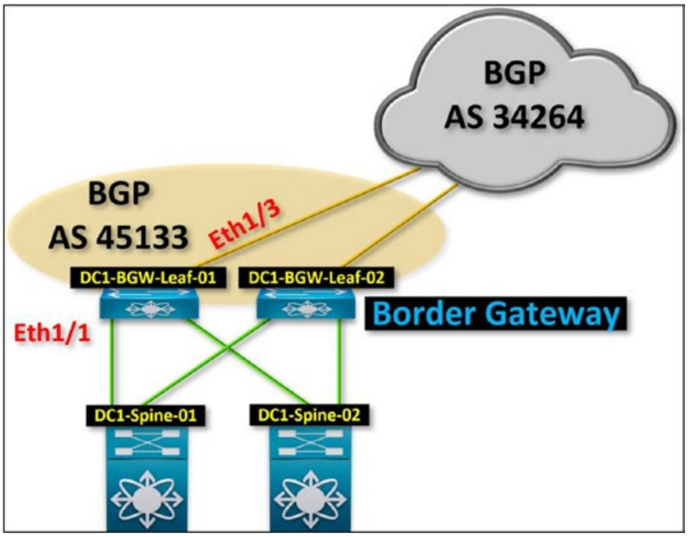


Figure 6-10. *Take a close look at this diagram. Interface Eth1/3 relates to the service provider router (external network). Interface Eth1/1 relates to DC1 Spine-01. You need to inform the BGP EVPN control plane which interface faces which network by applying the respective EVPN multisite configuration*

Listing 6-8. The required configuration to establish a BGP peering between the BGW and the local VXLAN Fabric. Required for EVPN Multi-Site

```
interface Ethernet1/1
  description TO_SPINE1
  mtu 9216
  medium p2p
  ip address 100.0.40.1/31
  ip ospf network point-to-point
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode
  evpn multisite fabric-tracking
  no shutdown
```

In Listing 6-8, an interface from the BGW facing the local VXLAN spine must be configured with the `evpn multisite fabric-tracking` command. This command tells the BGP process on the BGW that it's facing the local VXLAN fabric site and traffic to a remote site and from a remote site with traverse this link to the internal iBGP AS (the local AS number for the VXLAN fabric). Also, any outgoing traffic from the fabric is relayed to the BGW from the spine via this interface. The BGP peering source address to the spine from the BGW comes from this interface.

Listing 6-9. Sample configuration to establish a BGP peering from the BGW to the Service Provider

```
interface Ethernet1/3
  description to_ServiceProvider
  no switchport
  ip address 10.111.111.1/30 tag 12345
  evpn multisite dci-tracking
  no shutdown
```

In Listing 6-9, an interface facing the service provider needs to be configured with the `evpn multisite dci-tracking` command, which informs the local fabric that it would be the path to reach any other VXLAN fabric site in the cluster. A tag must be assigned to the IP address because this is how you advertise the BGW peering IP to the remote site fabric.

Once both locations are reachable from a routing standpoint, the VXLAN overlay transit is established and allows both VXLAN sites to communicate. At this point, routing for VXLAN becomes purely eBGP between sites.

Border Gateway BGP Configuration

Let's break down the BGP configurations on a border gateway. What can you expect from this configuration? There are a series of required configurations. I'm breaking down each one, describe it, and finally provide the full configuration.

Please review each one carefully.

A BGW needs a VTEP address which needs to be distributed to all other sites and allow the VXLAN overlay communication. Please review Listing 6-10.

Listing 6-10. BGW VXLAN VTEP Interface configurations

```
!Frankfurt Data Center BGW-01
interface loopback100
  description Frankfurt-VIP Multi-Site 1
  ip address 10.1.111.111/32 tag 12345
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode

!Munich Data Center BGW-01
interface loopback100
  description Munich-VIP Multi-Site 1
  ip address 20.1.111.111/32 tag 12345
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode
```

In Listing 6-10, a loopback interface is allocated for multisite EVPN peering configuration. Let's assume that DC1-BGW-01 is in Frankfurt, and DC2-BGW-01 is in Munich. From a BGP configuration standpoint, you need to build the multisite peering using the IP of 10.1.111.111/32 (Frankfurt) and 20.1.111.111/32 (Munich). A VXLAN transport is established between both BGWs using the loopbacks previously configured.

You also assign a tag to the interface IP because that is how you redistribute an interface in BGP to advertise it to the remote data center. A route-map is configured to match this tag then used in BGP to reference the redistribution.

```
route-map REDIST-LOCAL permit 10
  match tag 12345
```

Let's breakdown the BGW BGP configuration (see Listing 6-11).

Listing 6-11. Global BGP configuration on the BGW

```
! Begin Configuration

Router bgp 45133
router-id 10.10.10.111
  address-family ipv4 unicast
    redistribute direct route-map REDIST-LOCAL
    maximum-paths 4
```

In Listing 6-11, the first section of the configuration enables the IPv4 unicast AFI and advertises any local interface with the tag of 12345 assigned thanks to the route-map REDIST-LOCAL. You also specify the maximum BGP paths or hops you can take in the BGP path to send the information.

Listing 6-12. A BGP template to peer the adjacent VXLAN site thru the BGWs

```
  template peer TO_DC2_Multi-Site
    remote-as 65502
    update-source loopback0
    ebgp-multihop 2
    peer-type fabric-external
    address-family l2vpn evpn
      send-community
      send-community extended
      rewrite-evpn-rt-asn
```

In Listing 6-12's second section, I configured a series of peer templates. The beauty of using templates is that instead of writing the same attributes for each neighbor that you intend to peer over and over. You write the template and assign it to each neighbor. This template allows the multisite peering from the BGW to the remote site.

You specify a maximum of two hops between BGP neighbors to reach the destination. You also identify on the template that the neighbor you intend to peer to would be an external fabric peer in the multisite cluster. Under the L2VPN EVPN AFI, you enable the magic command that rewrites the RT for the ASN as it traverses the BGW.

Listing 6-13. BGP template to peer the BGW's to the Spines

```
template peer TO_SPINES
  remote-as 65501
  update-source loopback0
  ebgp-multihop 10
  address-family ipv4 unicast
    send-community
    send-community extended
  address-family l2vpn evpn
    send-community
    send-community extended
```

In Listing 6-13, I created a second template. This template has all the configuration required to peer the neighbor spines to the BGW. With templates, you significantly lower the amount of configuration syntax, making the BGP configuration cleaner.

Listing 6-14. A BGP configuration template to peer the remote spines to the BGWs

```
neighbor 10.111.111.2
  remote-as 34264
  address-family ipv4 unicast
neighbor 10.222.111.2
  remote-as 34264
  address-family ipv4 unicast
```

In Listing 6-14, the neighbors specified in this configuration are my service provider routers with AS 34264. I enabled the AFI for IPv4 unicast in BGP. This is needed to advertise to the remote fabric my BGW loopback (`loopback100`).

Listing 6-15. Sample configuration to apply a peering template to a BGP neighbor

```
  neighbor 20.20.20.111
    inherit peer TO_DC2_Multi-Site
  neighbor 20.20.20.222
    inherit peer TO_DC2_Multi-Site
```

In Listing 6-15, the configuration allows my local BGW to peer to the remote site BGWs. As you can see, I now associate my previously configured template applying all relevant configuration parameters.

Listing 6-16. Sample configuration to apply a peering template to a BGP neighbor

```
  neighbor 100.100.100.100
    inherit peer TO_SPINES
  neighbor 200.200.200.200
    inherit peer TO_SPINES
```

In Listing 6-16, the configuration allows the BGW to peer to the local spines. Also, I associate the template I configured with all required configurations to peer BGP with the local spines.

```
evpn multisite border-gateway 100
```

Listing 6-17. Full BGP BGW Configuration

```
interface nve1
  no shutdown
  host-reachability protocol bgp
  source-interface loopback1
  multisite border-gateway interface loopback100
  member vni 10010
     suppress-arp
     multisite ingress-replication
     mcast-group 225.1.1.10
  member vni 10011-10020
     suppress-arp
     mcast-group 225.1.1.111
  member vni 11111 associate-vrf

router bgp 45133
  router-id 10.10.10.111
  address-family ipv4 unicast
    redistribute direct route-map REDIST-LOCAL
    maximum-paths 4
  template peer TO_DC2_Multi-Site
    remote-as 65502
    update-source loopback0
    ebgp-multihop 2
    peer-type fabric-external
    address-family l2vpn evpn
      send-community
      send-community extended
      rewrite-evpn-rt-asn
  template peer TO_SPINES
    remote-as 65501
    update-source loopback0
    ebgp-multihop 2
    address-family ipv4 unicast
      send-community
      send-community extended
    address-family l2vpn evpn
      send-community
      send-community extended
  neighbor 10.111.111.2
    remote-as 34264
    address-family ipv4 unicast
  neighbor 10.222.111.2
    remote-as 34264
    address-family ipv4 unicast
  neighbor 20.20.20.111
    inherit peer TO_DC2_Multi-Site
  neighbor 20.20.20.222
    inherit peer TO_DC2_Multi-Site
  neighbor 100.100.100.100
    inherit peer TO_SPINES
  neighbor 200.200.200.200
    inherit peer TO_SPINES
```

Validate Multisite VXLAN Communication

Here's a new scenario. There are two servers on VLAN 10. The Frankfurt server's IP is 172.16.10.10 IP. The Munich server's IP is 172.16.10.20. How do you verify that both fabrics have learned the corresponding MAC and IP information from those two servers and the servers can communicate? Let's look at Figure 6-11.

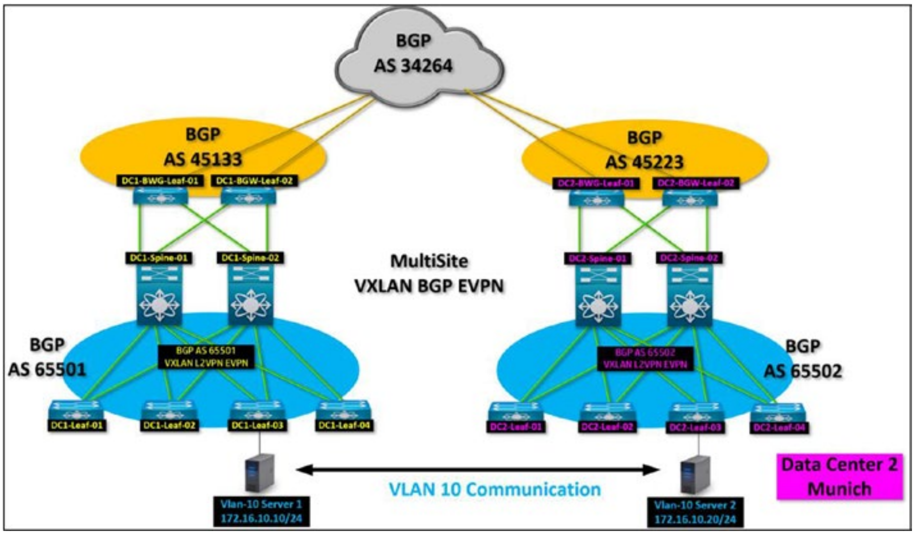


Figure 6-11. *A Multisite VXLAN Architecture*

Next, let's look at the Frankfurt border gateway DC1-BGW-01.

Listing 6-18. We can confirm if we are receiving any EVPN routes by invoking the "show bgp l2vpn evpn"

```
show bgp l2vpn evpn
```

Listing 6-19. Output of all BGP EVPN type routes learned on a particular leaf

```
DC1-BGW-011# show bgp l2vpn evpn

   Network      Next Hop      Metric      LocPrf      Weight Path

Route Distinguisher: 10.10.10.111:32777    (L2VNI 10010)
*>e[2]:[0]:[0]:[48]:[0050.7966.6812]:[0]:[0.0.0.0]/216
              20.2.222.222         2000              0 65502 i
*>e[2]:[0]:[0]:[48]:[500c.0000.1b08]:[0]:[0.0.0.0]/216
              20.222.222.202                      0 65502 i
```

```
*>e[2]:[0]:[0]:[48]:[0050.7966.6812]:[32]:[172.16.10.20]/272
              20.2.222.222          2000             0 65502 i
*>e[3]:[0]:[32]:[20.222.222.202]/88
           20.222.222.202                             0 65502 i

Route Distinguisher: 10.10.10.111:3    (L3VNI 11111)
*>e[2]:[0]:[0]:[48]:[0050.7966.6812]:[32]:[172.16.10.20]/272
               20.2.222.222          2000             0 65502 i
```

Listing 6-9 confirms the MAC address or type-2 EVPN advertisement from the remote BGW in Munich (20.2.222.22) in the L2VNI. It also confirms the 172.16.10.20 location from Frankfurt's BGW pointing to Munich.

Listing 6-20. Lets confirm how many active BGP EVPN neighbor border gateways we can see from DC1-BGW1

```
show bgp l2vpn evpn summary
```

Listing 6-21. As you can see on the diagram we are currently exchanging EVPN routes/prefixes with 20.20.20.222 which is the second border gateway at DC2 over EVPN

```
DC1-BGW1# show bgp l2vpn evpn summary
BGP summary information for VRF default, address family L2VPN EVPN
BGP router identifier 10.10.10.111, local AS number 65500
BGP table version is 41, L2VPN EVPN config peers 4, capable peers 2
21 network entries and 21 paths using 4080 bytes of memory
BGP attribute entries [21/3528], BGP AS path entries [1/6]
BGP community entries [0/0], BGP clusterlist entries [0/0]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/
Down  State/PfxRcd
20.20.20.111    4 65502       0       0        0    0    0
00:32:46 Idle
```

```
20.20.20.222    4 65502      54       39        41     0     0
00:31:08 7
100.100.100.100 4 65500      42       40        41     0     0
00:32:23 0
200.200.200.200 4 65500       0        0         0     0     0
00:32:46 Idle

Neighbor        T    AS PfxRcd      Type-2      Type-3      Type-4
Type-5
20.20.20.111    E 65502 Idle        0           0           0
0
20.20.20.222    E 65502 7           6           1           0
0
100.100.100.100 I 65500
0          0          0          0          0
200.200.200.200 I 65500 Idle        0           0           0
0
```

Listing 6-10 confirms with `show bgp l2vpn evpn summary` output that you are successfully peered in EVPN from Frankfurt's BGW to Munich BGW with IP 20.20.20.222

You are receiving type-2 and type-3 routes in EVPN, which is a very good sign that the configuration is working. You should see a similar output for the Munich BGW.

Listing 6-22. BGP EVPN border gateway adjacency from DC2 to DC1

```
DC2-BGW2# show bgp l2vpn evpn summary
BGP summary information for VRF default, address family L2VPN EVPN
BGP router identifier 20.20.20.222, local AS number 65502
BGP table version is 41, L2VPN EVPN config peers 4, capable peers 2
21 network entries and 21 paths using 4080 bytes of memory
BGP attribute entries [21/3528], BGP AS path entries [1/6]
BGP community entries [0/0], BGP clusterlist entries [1/4]
```

```
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/
Down  State/PfxRcd
10.10.10.111    4 65500      55      43       41    0    0
00:34:39 5
10.10.10.222    4 65500       0       0        0    0    0
00:36:19 Idle
101.101.101.101 4 65502       0       0        0    0    0
00:36:19 Idle
201.201.201.201 4 65502      50      44       41    0    0
00:35:37 2

Neighbor        T    AS PfxRcd     Type-2     Type-3     Type-4
Type-5
10.10.10.111    E 65500 5          4          1          0
0
10.10.10.222    E 65500 Idle       0          0          0
0
101.101.101.101 I 65502 Idle       0          0          0
0
201.201.201.201 I 65502
2          2          0          0          0
```

Listing 6-11 confirms the same information in Munich. You learn EVPN type-2 and type-3 routes from the BGW in Frankfurt (10.10.10.111).

Let's do the final verification in DC1-Leaf-03 in Frankfurt to confirm that you see the BGP EVPN prefixes from Munich. You should see the advertised 172.16.10.20 server in the Frankfurt tenant-VRF.

Show ip route vrf Tenant-1

Listing 6-23. We can now got confirmation that we have received EVPN type-5 routes on DC1-Leaf-03

```
DC1-Leaf3# show ip route vrf Tenant-1
IP Route Table for VRF "Tenant-1"
```

```
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.10.0/24, ubest/mbest: 1/0, attached
    *via 172.16.10.1, Vlan10, [0/0], 00:38:58, direct
172.16.10.1/32, ubest/mbest: 1/0, attached
    *via 172.16.10.1, Vlan10, [0/0], 00:38:58, local

172.16.10.10/32, ubest/mbest: 1/0, attached
    *via 172.16.10.10, Vlan10, [190/0], 00:24:30, hmm

172.16.10.20/32, ubest/mbest: 1/0
    *via 10.1.111.111%default, [200/2000], 00:24:13, bgp-65500,
internal, tag 65502, segid: 11111 tunnelid: 0xa016f6f encap: VXLAN
```

In Listing 6-23, the EVPN route from Munich advertises the 172.16.10.20 server IP in the Frankfurt leaf for tenant-1. You can see the locally learned Frankfurt server's 172.16.10.10 IP.

Ingress Replication VXLAN Architecture

In VXLAN BGP EVPN, you leverage multicast to announce the BUM traffic information. This is performed by a rendezvous point (RP) configured with a routing device that participates in the VXLAN fabric underlay. When designing a VXLAN architecture, you need to analyze your overall fabric configuration and leaf capability requirements.

Simpler fabric designs are also available for deployment. Maybe you don't need a large-scale spine-and-leaf architecture. Your environment could be handled with only four leafs, considering that port density requirements aren't critical and that there's no expectation for any future growth. With this in mind, you can deploy a VXLAN fabric on a much smaller scale.

One simpler/smaller architecture can be configured with head-end replication (HER), also known as ingress replication (IR). IR handles BUM traffic announcements using unicast instead of multicast, which means that the underlay multicast configuration is not performed. Multicast-specific architecture configurations such as VNI multicast groups and rendezvous points are removed, making the underlay configuration much simpler. It sounds good from an implementation standpoint, but there's a drawback and a reason why it may not be suited for your environment.

Announcing BUM using unicast requires more hardware resources. Every time you advertise to BUM traffic, it requires a unicast message to do so. Every leaf needs to communicate its BUM traffic information to all neighbor leafs. Leaf-01 sends the BUM to Leaf-02, Leaf-03, and Leaf-04 rather than having a multicast group and a single announcement performed since every leaf listens to the group, gets the information, and responds within vs a unicast message performed by all leafs every time it needs to send the BUM announcement.

A VXLAN BGP EVPN Fabric Without Spines?

VXLAN BGP EVPN without spines is possible with ingress replication (IR) since it removes the need for multicast. A rendezvous point (RP) configured at the spine layer is no longer needed. The need for east-west fabric expansion is no longer needed, so the spines that allowed the expansion benefit are not needed.

You still take advantage of ECMP in the underlay IGP and VXLAN overlay since you can use dual home leafs between each other. For example, Leaf-01 can be homed to Leaf-03 and Leaf-04. Leaf-03 can be homed to Leaf-01 and Leaf-02. Connection redundancy is still present (N+1). With an IR design, you connect the leafs underlay links together (see Figure 6-12).

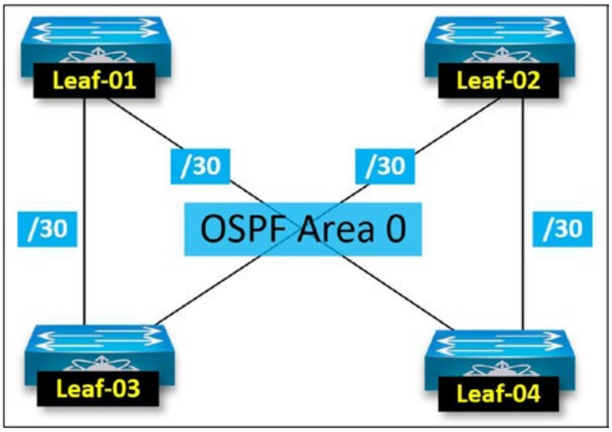


Figure 6-12. *IR/HER VXLAN BGP EVPN architecture without spines. Underlay links are configured the same way as spine-and-leaf architecture but without the need for multicast. Enabled multicast is not required on the interface configuration*

A BGP EVPN operation in IR/HER works similarly to multicast in spine-and-leaf. The only difference is that you no longer route-reflect EVPN on the spines, so the NVE peers between leafs. You enable NVE peering between the VTEPs (leafs). The BGP EVPN route path is leaf-to-leaf at this point, so no transit via any spines. VTEP NVE peering happens the same way as the spine-and-leaf model, Leaf-01 NVE interface peers to all neighbor leafs (see Figure 6-13).

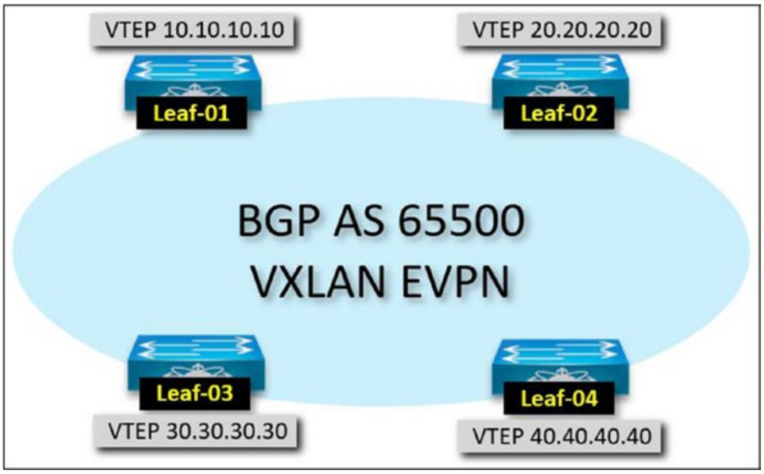


Figure 6-13. *BGP EVPN communicates similarly without spines in the fabric. The VTEPs peer each other using the adjacency established in the IGP. VXLAN traffic path remains the same, leaf to leaf*

Note IR/HER is also supported in a spine-and-leaf architecture. The spines still reflect the BGP EVPN AFI traffic to the leafs. The environment remains multicast-independent from an underlay standpoint.

Configuring IR/HER

With IR/HER, the leaf configurations are significantly simpler than configuring it with underlay multicast. The only configuration needed in IR/HER is performed under the NVE interface (see Listing 6-24).

Listing 6-24. Enabling ingress-replication (HER) for a particular VNI, in this case 10010

```
int nve 1
source-interface lo0
host-reachability protocol bgp
member vni 10010
  suppress-arp
  ingress-replication protocol bgp
```

In Listing 6-24, the only configuration required with IR/HER is to specify under the NVE interface that you use IR to perform the BUM messages for a member VNI with unicast rather than multicast groups.

Let me provide you a sample leaf configuration template using IR/HER. Please review the configuration script and compare it with a multicast-enabled underlay configuration so you can spot the differences. It is quite noticeable.

Sample IR/HER Leaf Configuration Template

Listing 6-25 is the template.

Listing 6-25. A sample ingress replication configuration leaf template

```
feature bgp
feature fabric-forwarding
feature ospf
feature vn-segment-vlan-based
feature nv overlay
nv overlay evpn

vlan 10
name Servers
vn-segment 10010
```

```
router ospf Underlay
 router-id 1.0.0.10

int lo0
ip address 1.0.0.10/32
ip router ospf Underlay area 0

int lo1
ip address 10.10.10.10/32
router bgp 65501
 log-neighbor-changes
 address-family ipv4 unicast
 network 10.10.10.10/32
 address-family l2vpn evpn
template peer TO_SPINES
 remote-as 65501
 update-source lo0
  address-family ipv4 unicast
  send-community
  send-community extended
  address-family l2vpn evpn
  send-community
  send-community extended

neighbor 1.0.0.100
 inherit peer TO_SPINES
neighbor 1.0.0.200
 inherit peer TO_SPINES

int nve 1
 no shut
 source-interface lo1
 host-reachability protocol bgp
 member vni 10010
 ingress-replication protocol bgp
 suppress-arp
```

IR/HER VXLAN Fabric Without Spines

Let's perform some show commands to see how an IR/HER fabric performs. You should see the same results as with a multicast-enabled underlay fabric. Confirm which OSPF neighbors have established adjacency. You should not see any other device but leafs. Also, confirm that you see EVPN routes advertised and the MACs learned. Finally, make sure multicast is not running on the environment.

Listing 6-26 shows the EVPN status.

Listing 6-26. By performing a "show bgp l2vpn evpn summary" from DC1-Leaf-01 we have confirmed that we have the other three leaves adjacent from a BGP EVPN peering standpoint

```
DC1-LEAF-01# show bgp l2vpn evpn summary
BGP summary information for VRF default, address family L2VPN EVPN
BGP router identifier 1.0.0.10, local AS number 65501
BGP table version is 32, L2VPN EVPN config peers 3, capable peers 3
14 network entries and 14 paths using 2640 bytes of memory
BGP attribute entries [11/1848], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [0/0]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/
Down  State/PfxRcd
1.0.0.20        4 65501      61      59       32    0    0
00:50:07 1
1.0.0.30        4 65501      61      60       32    0    0
00:50:20 1
1.0.0.40        4 65501      53      48       32    0    0
00:38:49 3
```

BGP L2VPN EVPN Neighbors are active and established.

```
DC1-LEAF-01# show nve peers
Interface Peer-IP                                 State
LearnType Uptime   Router-Mac
--------- --------------------------------------  ----- -------
-- -------- ----------------
nve1      20.20.20.20                             Up    CP
00:59:07 n/a
nve1      30.30.30.30                             Up    CP
00:59:19 n/a
nve1      40.40.40.40                             Up    CP
00:47:41    5004.0000.1b08
VXLAN NVE peering is established from DC1-LEAF-01 to all other
leafs.
```

OSPF status.

```
DC1-LEAF-01# show ip ospf neig
 OSPF Process ID Underlay VRF default
 Total number of neighbors: 2
 Neighbor ID     Pri State            Up
Time  Address         Interface
 1.0.0.30          1 FULL/ -          01:09:06
10.30.0.2       Eth1/1
 1.0.0.40          1 FULL/ -          00:57:31
10.40.0.2       Eth1/2
The OSPF status is active, DC1-LEAF-01 neighbors are LEAF-03
and LEAF-04. This confirms that the fabric does not have any
spines participating in OSPF. Leafs are peering to leafs.
```

Tenant VRF status

```
DC1-LEAF-01# show ip route vrf Tenant-1
```

Listing 6-27. Under the tenant route table we can confirm that we have received EVPN host routes from DC1-Leaf-04 under VTEP IP 40.40.40.40

```
10.10.0.0/24, ubest/mbest: 1/0, attached
    *via 10.10.0.1, Vlan10, [0/0], 01:19:38, direct
10.10.0.1/32, ubest/mbest: 1/0, attached
    *via 10.10.0.1, Vlan10, [0/0], 01:19:38, local
10.10.0.10/32, ubest/mbest: 1/0, attached
    *via 10.10.0.10, Vlan10, [190/0], 00:56:46, hmm
10.10.0.40/32, ubest/mbest: 1/0
    *via 40.40.40.40%default, [200/0], 00:53:59, bgp-65501,
internal, tag 65501, segid: 99999 tunnelid: 0x28282828 encap: VXLAN
```

DC1-Leaf-01 has learned an end point on VLAN 10 with address 10.10.0.40 via Leaf-04 VTEP 40.40.40.40 using the L3VNI 99999.

No multicast should be running in the underlay for BUM traffic.

```
DC1-LEAF-01# show ip mroute
IP Multicast Routing Table for VRF "default"
(initializing - suppressing MFDM updates, time-left: 00:00:02)

DC1-LEAF-01#
```

```
No multicast is running on the fabric for BUM traffic. You are
performing BUM advertisements using ingress replication.
```

Verify the MAC address table (see Listing 6-28).

Listing 6-28. We have learned a L2 address from VTEP 40.40.40.40

```
DC1-LEAF-01# show mac address-table vlan 10
Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC,
O - Overlay MAC
```

```
        age - seconds since last seen,+ - primary entry using
vPC Peer-Link,
        (T) - True, (F) - False, C - ControlPlane MAC, ~ - vsan
   VLAN     MAC Address      Type      age     Secure NTFY Ports
---------+-----------------+--------+---------+------+----+-----
-------------
*   10     0050.7966.6805   dynamic  0          F      F
Eth1/3
C   10     0050.7966.6807   dynamic  0          F      F
nve1(40.40.40.40)
G   10     5002.0000.1b08   static   -          F      F
sup-eth1(R)
```

DC1-Leaf-01 has learned a MAC from DC1-Leaf-04 (VTEP 40.40.40.40)

Listing 6-29. DC1-Leaf-01 has also learned a L2 address from DC1-Leaf-04

```
DC1-LEAF-04# show mac address-table vlan 10
Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC,
        O - Overlay MAC
        age - seconds since last seen,+ - primary entry using
        vPC Peer-Link,
        (T) - True, (F) - False, C - ControlPlane MAC, ~ - vsan
   VLAN     MAC Address      Type      age     Secure NTFY
Ports
---------+-----------------+--------+---------+------+----+
------------------
C   10     0050.7966.6805   dynamic  0          F      F
nve1(10.10.10.10)
```

```
*   10     0050.7966.6807   dynamic  0         F      F
Eth1/3
G   10     5004.0000.1b08   static   -         F      F
sup-eth1(R)
```

DC1-Leaf-04 has learned the MAC from DC1-Leaf-01 (VTEP 10.10.10.10).

This confirms that the four-leaf VXLAN BGP EVPN spineless fabric is running without multicast using IR/HER for BUM traffic advertisement.

CHAPTER 7

VXLAN BGP EVPN Fabric Configuration Templates

As you reach the end of your learning journey in this book, I've compiled all the VXLAN fabric design scenarios in functional VXLAN templates that you can copy and use to configure on your fabric equipment. The templates provided in this chapter are meant for the Cisco Nexus 9K data center switching platform. It doesn't mean that you can't use this layout with other vendors. You can use it as a model for any vendor that supports BGP EVPN because the principle and configuration requirements are the same. It's almost certain that the configuration syntax will change.

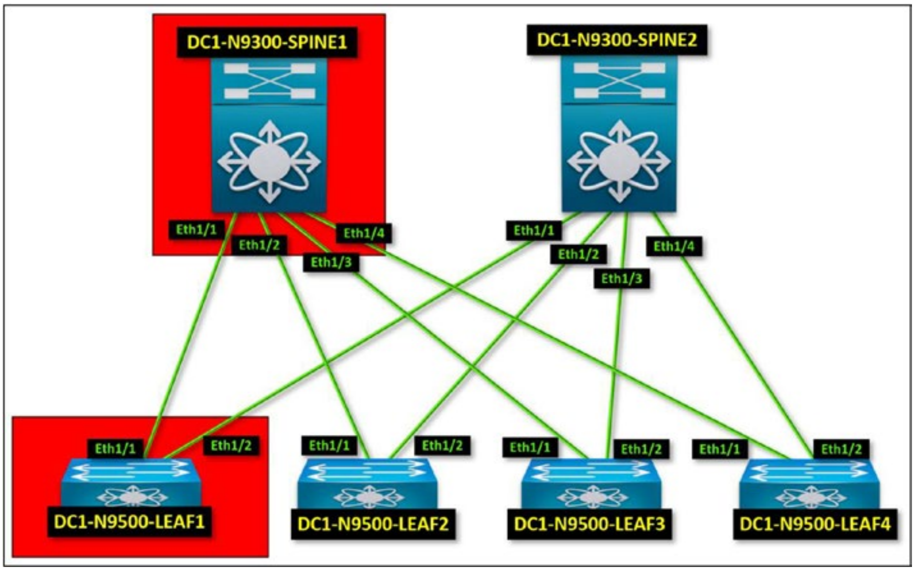


Figure 7-1. *A standalone VXLAN spine-and-leaf topology*

VXLAN Spine-and-Leaf Fabric (Stand-alone)

Listing 7-1. LEAF (DC1-Leaf-01)

```
nv overlay evpn
feature ospf
feature bgp
feature pim
feature fabric forwarding
feature interface-vlan
feature vn-segment-vlan-based
feature nv overlay

fabric forwarding anycast-gateway-mac cc1e.0006.2050
ip pim rp-address 10.1.10.2 group-list 224.0.0.0/16
```

```
vlan 10
  name Users
  vn-segment 100010
vlan 11
  name Wired
  vn-segment 100011
vlan 20
  name Phones
  vn-segment 100020

vlan 2210
  name L3VNI-Tenant-1
  vn-segment 2121210

route-map Clients-SVI permit 10
  match tag 222

vrf context Tenant-1
  vni 2121210
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target both auto evpn

interface Ethernet1/1
  description TO_DC1_N9300_SPINE1_ETH1/1
  mtu 9216
  ip address 10.1.1.1/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown
```

```
interface Ethernet1/2
  description TO_DC1_N9300_SPINE2_ETH1/1
  mtu 9216
  ip address 10.2.1.1/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown

interface Vlan10
  description USERS
  no shutdown
  vrf member Tenant-1
  ip address 10.0.0.1/23 tag 222
  fabric forwarding mode anycast-gateway

interface Vlan11
  no shutdown
  vrf member Tenant-1
  ip address 172.20.11.1/24 tag 222
  fabric forwarding mode anycast-gateway

interface Vlan20
  no shutdown
  vrf member Clients
  ip address 172.20.12.1/24 tag 222
  fabric forwarding mode anycast-gateway

interface Vlan2210
  no shutdown
  mtu 9000
  vrf member Tenant-1
  ip forward
```

```
interface nve1
  no shutdown
  host-reachability protocol bgp
  source-interface loopback1
  member vni 100010
    suppress-arp
    mcast-group 224.1.1.10
  member vni 100011
    suppress-arp
    mcast-group 224.1.1.11
  member vni 100020
    suppress-arp
    mcast-group 224.1.1.20
  member vni 2121210 associate-vrf

router ospf Underlay
  router-id 1.0.0.10

interface loopback0
description BGP_SOURCE_INTERFACE
  ip address 1.0.0.10/32
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode

interface loopback1
description VXLAN_VTEP_INTERFACE
  ip address 10.10.10.10/32
  ip pim sparse-mode

router bgp 65501
  log-neighbor-changes
  address-family ipv4 unicast
    network 10.10.10.10/32
  address-family l2vpn evpn
  template peer TO_SPINES
    remote-as 65501
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
    address-family l2vpn evpn
      send-community
      send-community extended
  neighbor 1.0.0.100
    inherit peer TO_SPINES
  neighbor 1.0.0.200
    inherit peer TO_SPINES
  vrf Tenant-1
    address-family ipv4 unicast
    redistribute direct route-map Clients-SVI
      advertise l2vpn evpn

evpn
  vni 100010 l2
    rd auto
    route-target import 65501:10
    route-target export 65501:10
  vni 100011 l2
    rd auto
    route-target import 65501:11
    route-target export 65501:11
  vni 100020 l2
    rd auto
    route-target import 65501:20
    route-target export 65501:20
```

Listing 7-2. SPINE (DC1-Spine-01)

```
nv overlay evpn
feature ospf
feature bgp
feature pim
feature nv overlay

ip pim rp-address 10.1.10.2 group-list 224.0.0.0/16
ip pim ssm range 232.0.0.0/8
ip pim anycast-rp 10.1.10.2 1.0.0.100
ip pim anycast-rp 10.1.10.2 1.0.0.200

interface Ethernet1/1
  description TO_DC1_N9500_LEAF1_ETH1/1
  mtu 9216
  ip address 10.1.1.2/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown

interface Ethernet1/2
  description TO_DC1_N9500_LEAF2_ETH1/1
  mtu 9216
  ip address 10.1.2.2/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown

interface Ethernet1/3
  description TO_DC1_N9500_LEAF3_ETH1/1
  mtu 9216
  ip address 10.1.3.2/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown

interface Ethernet1/4
  description TO_DC1_N9500_LEAF4_ETH1/1
  mtu 9216
  ip address 10.1.4.2/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown

interface loopback0
description BGP_SOURCE_INTERFACE
  ip address 1.0.0.100/32
  ip ospf network point-to-point
  ip router ospf DC1-Underlay area 0.0.0.0
  ip pim sparse-mode

interface loopback2
  description ANYCAST-RP
  ip address 10.1.10.2/32
  ip router ospf DC1-Underlay area 0.0.0.0
  ip pim sparse-mode

router ospf Underlay
  router-id 10.1.0.1
```

```
router bgp 65501
  log-neighbor-changes
  address-family ipv4 unicast
  address-family l2vpn evpn
  template peer TO_LEAFS
    remote-as 65501
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
      route-reflector-client
    address-family l2vpn evpn
      send-community
      send-community extended
      route-reflector-client
  neighbor 1.0.0.10
    inherit peer TO_LEAFS
  neighbor 1.0.0.20
    inherit peer TO_LEAFS
  neighbor 1.0.0.30
    inherit peer TO_LEAFS
  neighbor 1.0.0.40
    inherit peer TO_LEAFS
```

VXLAN Multipod Fabric (DC1-DC2)

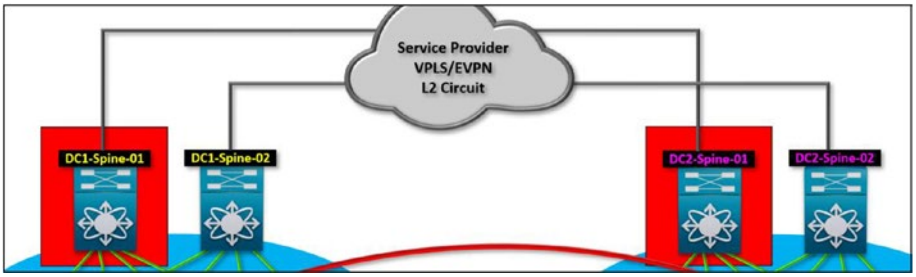


Figure 7-2. *The multipod data center interconnect for VXLAN BGP EVPN*

In a multipod fabric, all fabric leafs are configured similarly to a stand-alone fabric leaf. The parameters required in a multipod are configured on the spines.

Listing 7-3. DC1-Spine-01 (This Spine Connects to DC2-Spine-01)

```
nv overlay evpn
feature ospf
feature bgp
feature pim
feature fabric forwarding
feature interface-vlan
feature vn-segment-vlan-based
feature nv overlay
ip pim rp-address 1.1.2.2 group-list 224.0.0.0/16
ip pim ssm range 232.0.0.0/8
ip pim anycast-rp 1.1.2.2 1.0.0.100
ip pim anycast-rp 1.1.2.2 1.0.0.200
ip pim anycast-rp 1.1.2.2 2.0.0.100
ip pim anycast-rp 1.1.2.2 2.0.0.200
```

```
route-map MultiPod-VXLAN permit 10
  set ip next-hop unchanged

interface Ethernet1/4
  description To_DC2-SPINE-01 MultiPod
  no switchport
  mtu 9216
  medium p2p
  no ip redirects
  ip address 100.200.0.1/30
  no ipv6 redirects
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown

interface loopback0
  description BGP-Peer-Loopback
  ip address 1.0.0.100/32
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode

interface loopback1
  description VXLAN-VTEP
  ip address 101.101.101.101/32
  ip pim sparse-mode

interface loopback3
  description ANYCAST-RP
  ip address 1.1.2.2/32
  ip router ospf BUR-Underlay area 0.0.0.0
  ip pim sparse-mode
```

```
router ospf Underlay
  router-id 1.0.0.100

router bgp 65501
  log-neighbor-changes
  address-family ipv4 unicast
  address-family l2vpn evpn
  template peer EVPN-TO-LEAFS
    remote-as 65501
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
      route-reflector-client
    address-family l2vpn evpn
      send-community
      send-community extended
      route-reflector-client
  neighbor 1.0.0.11
    inherit peer EVPN-TO-LEAFS
  neighbor 1.0.0.12
    inherit peer EVPN-TO-LEAFS

! GOING TO DC2-SPINE-01 (MultiPod)
  neighbor 2.0.0.100
    remote-as 65502
    update-source loopback0
    ebgp-multihop 10
    address-family ipv4 unicast
    address-family l2vpn evpn
      send-community
      send-community extended
      route-map MultiPod-VXLAN out
```

Listing 7-4. DC2-Spine-01 (This Spine Connects to DC1-Spine-01)

```
nv overlay evpn
feature ospf
feature bgp
feature pim
feature interface-vlan
feature vn-segment-vlan-based
feature nv overlay

ip pim rp-address 1.1.2.2 group-list 224.0.0.0/16
ip pim ssm range 232.0.0.0/8
ip pim anycast-rp 1.1.2.2 1.0.0.100
ip pim anycast-rp 1.1.2.2 1.0.0.200
ip pim anycast-rp 1.1.2.2 2.0.0.100
ip pim anycast-rp 1.1.2.2 2.0.0.200

route-map MultiPod-VXLAN permit 10
  set ip next-hop unchanged

interface Ethernet1/6
  description To_DC1-SPINE-01 MultiPod
  no switchport
  mtu 9216
  medium p2p
  no ip redirects
  ip address 100.200.0.2/30
  no ipv6 redirects
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode
  no shutdown
```

```
interface loopback0
  description BGP-Peer-Loopback
  ip address 2.0.0.100/32
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode

interface loopback1
  description VXLAN-VTEP
  ip address 201.201.201.201/32
  ip pim sparse-mode

interface loopback3
  description ANYCAST-RP
  ip address 1.1.2.2/32
  ip router ospf Underlay area 0.0.0.0
  ip pim sparse-mode

router ospf Underlay
  router-id 2.0.0.100

router bgp 65502
  log-neighbor-changes
  address-family ipv4 unicast
  address-family l2vpn evpn
  template peer EVPN-TO-LEAFS
    remote-as 65502
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
      route-reflector-client
    address-family l2vpn evpn
      send-community
      send-community extended
      route-reflector-client
  neighbor 2.0.0.11
    inherit peer EVPN-TO-LEAFS
  neighbor 2.0.0.12
    inherit peer EVPN-TO-LEAFS
```

```
!MultiPod VXLAN Configuration
```

```
  neighbor 1.0.0.100
    remote-as 65501
    update-source loopback0
    ebgp-multihop 10
    address-family ipv4 unicast
    address-family l2vpn evpn
      send-community
      send-community extended
      route-map MultiPod-VXLAN out
```

VXLAN Multisite Fabric (Frankfurt-Munich)

In a multisite VXLAN, the border gateways, or BGW, control the egress and ingress from/to a VXLAN fabric site in BGP EVPN. Spines and leafs are configured the same way as in a standalone VXLAN spine-and-leaf fabric.

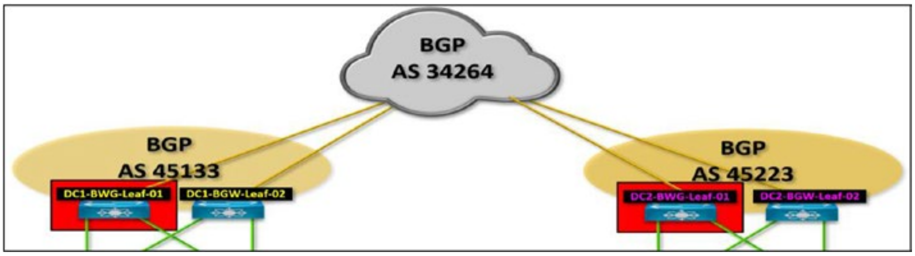


Figure 7-3. *The multisite border gateway leaf peering to the service provider*

Listing 7-5. DC1-BGW-01 Frankfurt Data Center/Multisite Border Gateway (BGW)

```
nv overlay evpn
feature ospf
feature bgp
feature pim
feature vn-segment-vlan-based
feature nv overlay

evpn multisite border-gateway 100

ip pim rp-address 10.0.10.1 group-list 225.1.1.0/24 bidir
ip pim ssm range 232.0.0.0/8

evpn storm-control broadcast level 10
evpn storm-control multicast level 10
evpn storm-control unicast level 10

vlan 1,10-12,20-22,1111
vlan 10
  vn-segment 10010
vlan 11
  vn-segment 10011
vlan 12
```

```
  vn-segment 10012
vlan 20
  vn-segment 10020
vlan 21
  vn-segment 10021
vlan 22
  vn-segment 10022
vlan 1111
  vn-segment 11111

route-map REDIST-LOCAL permit 10
  match tag 12345

vrf context Tenant-1
  vni 11111
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target both auto evpn

interface nve1
  no shutdown
  host-reachability protocol bgp
  source-interface loopback1
  multisite border-gateway interface loopback100
  member vni 10010
    suppress-arp
    multisite ingress-replication
    mcast-group 225.1.1.10
  member vni 10011-10020
    mcast-group 225.1.1.111
  member vni 11111 associate-vrf
```

```
! Interfaces to the Spines

interface Ethernet1/1
  description TO_SPINE1_ETH1/4
  no switchport
  mtu 9216
  medium p2p
  ip address 100.0.40.1/31
  ip ospf network point-to-point
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode
  evpn multisite fabric-tracking
  no shutdown

interface Ethernet1/2
  description TO_SPINE2_ETH1/4
  no switchport
  mtu 9216
  medium p2p
  ip address 200.0.40.1/31
  ip ospf network point-to-point
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode
  evpn multisite fabric-tracking
  no shutdown

! Interfaces to the Service Provider

interface Ethernet1/3
  description To_eBGP-RTR1_Gi1
  no switchport
  ip address 10.111.111.1/30 tag 12345
  evpn multisite dci-tracking
  no shutdown

interface Ethernet1/4
  description To_eBGP-RTR2_Gi1
  no switchport
  ip address 10.222.111.1/30 tag 12345
  evpn multisite dci-tracking
  no shutdown

interface loopback0
  description RID
  ip address 10.10.10.111/32 tag 12345
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode

interface loopback1
  description PIP VTEP
  ip address 10.111.111.101/32 tag 12345
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode

interface loopback100
  description VIP Multi-Site 1
  ip address 10.1.111.111/32 tag 12345
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode

router ospf 1
  router-id 10.10.10.111

router bgp 65501
  router-id 10.10.10.111
  address-family ipv4 unicast
    network 10.1.111.111/32
    network 10.10.10.111/32
    network 10.111.111.101/32
    redistribute direct route-map REDIST-LOCAL
    maximum-paths 4
```

! Template to peer to the remote BGW

```
  template peer TO_DC2_Multi-Site
    remote-as 65502
    update-source loopback0
    ebgp-multihop 2
    peer-type fabric-external
    address-family l2vpn evpn
      send-community
      send-community extended
      rewrite-evpn-rt-asn
```

! Template to peer the SPINES

```
  template peer TO_SPINES
    remote-as 65501
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
    address-family l2vpn evpn
      send-community
      send-community extended
```

! eBGP peering to the Service Provider

```
  neighbor 10.111.111.2
    remote-as 45133
    address-family ipv4 unicast
  neighbor 10.222.111.2
    remote-as 45133
    address-family ipv4 unicast
```

! eBGP peering to the remote BGWs

```
  neighbor 20.20.20.111
    inherit peer TO_DC2_Multi-Site
  neighbor 20.20.20.222
    inherit peer TO_DC2_Multi-Site
```

! Local iBGP peering to the Spines

```
  neighbor 100.100.100.100
    inherit peer TO_SPINES
  neighbor 200.200.200.200
    inherit peer TO_SPINES

evpn
  vni 10010 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10011 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10012 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10020 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10021 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10022 l2
    rd auto
    route-target import auto
    route-target export auto
```

Listing 7-6. DC2-BGW-01 Munich Data Center/Multisite Border Gateway (BGW)

```
nv overlay evpn
feature ospf
feature bgp
feature pim
feature vn-segment-vlan-based
feature nv overlay

evpn multisite border-gateway 200

ip pim rp-address 11.0.10.1 group-list 225.1.1.0/24 bidir
ip pim ssm range 232.0.0.0/8

evpn storm-control broadcast level 10
evpn storm-control multicast level 10
evpn storm-control unicast level 10

vlan 1,10-12,20-22,1111
vlan 10
  vn-segment 10010
vlan 11
  vn-segment 10011
```

```
vlan 12
  vn-segment 10012
vlan 20
  vn-segment 10020
vlan 21
  vn-segment 10021
vlan 22
  vn-segment 10022
vlan 1111
  vn-segment 11111

route-map REDIST-LOCAL permit 10
  match tag 12345

vrf context Tenant-1
  vni 11111
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target both auto evpn

interface nve1
  no shutdown
  host-reachability protocol bgp
  source-interface loopback1
  multisite border-gateway interface loopback100
  member vni 10010
    suppress-arp
    multisite ingress-replication
    mcast-group 225.1.1.10
  member vni 10011-10020
    mcast-group 225.1.1.111
  member vni 11111 associate-vrf
```

```
! Interfaces to the Spines

interface Ethernet1/1
  description TO_SPINE1_1/1
  no switchport
  mtu 9216
  medium p2p
  ip address 200.1.40.1/31
  ip ospf network point-to-point
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode
  evpn multisite fabric-tracking
  no shutdown

interface Ethernet1/2
  description TO_SPINE2_1/1
  no switchport
  mtu 9216
  medium p2p
  ip address 201.0.40.1/31
  ip ospf network point-to-point
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode
  evpn multisite fabric-tracking
  no shutdown

! Interfaces to the Service Provider

interface Ethernet1/3
  description To_eBGP-RTR1_Gi3
  no switchport
  ip address 20.111.111.1/30 tag 12345
  evpn multisite dci-tracking
  no shutdown
```

```
interface Ethernet1/4
  description TO_eBGP-RTR2_Gi3
  no switchport
  ip address 20.222.111.1/30 tag 12345
  evpn multisite dci-tracking
  no shutdown

interface loopback0
  ip address 20.20.20.111/32 tag 12345
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode

interface loopback1
  ip address 20.222.222.101/32 tag 12345
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode

interface loopback100
  ip address 20.2.222.222/32 tag 12345
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode

router ospf 1
  router-id 20.20.20.111

router bgp 65502
  router-id 20.20.20.111
  address-family ipv4 unicast
    redistribute direct route-map REDIST-LOCAL
    maximum-paths 4

! Template to peer to the remote BGW

  template peer TO_DC1_Multi-Site
    remote-as 65500
    update-source loopback0
    ebgp-multihop 2
    peer-type fabric-external
    address-family l2vpn evpn
      send-community
      send-community extended
      rewrite-evpn-rt-asn
```

! Template to peer the SPINES

```
  template peer TO_SPINES
    remote-as 65502
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
    address-family l2vpn evpn
      send-community
      send-community extended
```

! eBGP peering to the remote BGWs

```
  neighbor 10.10.10.111
    inherit peer TO_DC1_Multi-Site
  neighbor 10.10.10.222
    inherit peer TO_DC1_Multi-Site
```

! eBGP peering to the Service Provider

```
  neighbor 20.111.111.2
    remote-as 65498
    address-family ipv4 unicast
  neighbor 20.222.111.2
    remote-as 65498
    address-family ipv4 unicast
```

```
! Local iBGP peering to the Spines
  neighbor 101.101.101.101
    inherit peer TO_SPINES
  neighbor 201.201.201.201
    inherit peer TO_SPINES

evpn
  vni 10010 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10011 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10012 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10020 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10021 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 10022 l2
    rd auto
    route-target import auto
    route-target export auto
```

VXLAN Ingress Replication (Spineless)

In VXLAN ingress replication or head-end replication, there are no requirements to configure a multicast network in the underlay for BUM traffic. You send the BUM traffic in unicast. There are no spines needed in this design; only a small VXLAN fabric composed of four leafs.

In this section, I provide the first two leaf configurations. The remaining two are nearly the same configuration, except each needs a unique interface IP address.

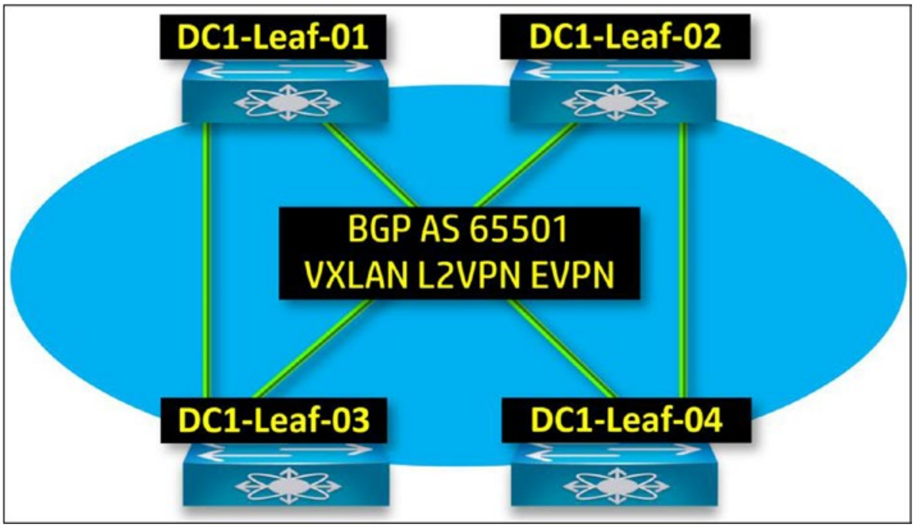


Figure 7-4. *A VXLAN BGP EVPN ingress replication topology without spines*

Listing 7-7. DC1-Leaf-01 (VXLAN: No Multicast Required IR/HER)

```
nv overlay evpn
feature ospf
feature bgp
feature fabric forwarding
feature interface-vlan
feature vn-segment-vlan-based
feature nv overlay

fabric forwarding anycast-gateway-mac cc1e.0006.2050
vlan 1,10,999
vlan 10
  vn-segment 10010
vlan 999
  vn-segment 99999

vrf context Tenant-1
  vni 99999
  rd auto
  address-family ipv4 unicast
    route-target import 65501:99999
    route-target import 65501:99999 evpn
    route-target export 65501:99999
    route-target export 65501:99999 evpn

interface Vlan10
  no shutdown
  vrf member Tenant-1
  ip address 10.10.0.1/24
  fabric forwarding mode anycast-gateway

interface Vlan999
  no shutdown
  vrf member Tenant-1
  ip forward

interface nve1
  no shutdown
  host-reachability protocol bgp
  source-interface loopback1
  member vni 10010
    suppress-arp
    ingress-replication protocol bgp
  member vni 99999 associate-vrf

interface Ethernet1/1
description_TO_LEAF_03
  mtu 9216
  medium p2p
  ip address 10.30.0.1/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  no shutdown

interface Ethernet1/2
  description_TO_LEAF_04
  mtu 9216
  medium p2p
  ip address 10.40.0.1/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  no shutdown

interface loopback0
  ip address 1.0.0.10/32
  ip router ospf Underlay area 0.0.0.0

interface loopback1
  ip address 10.10.10.10/32

router ospf Underlay
  router-id 1.0.0.10
```

```
router bgp 65501
  address-family ipv4 unicast
    network 10.10.10.10/32
  template peer LEAFS
    remote-as 65501
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
    address-family l2vpn evpn
      send-community
      send-community extended
  neighbor 1.0.0.20
    inherit peer LEAFS
  neighbor 1.0.0.30
    inherit peer LEAFS
  neighbor 1.0.0.40
    inherit peer LEAFS
  vrf Tenant-1
    address-family ipv4 unicast
      advertise l2vpn evpn
evpn
  vni 10010 l2
    rd auto
    route-target import auto
    route-target export auto
```

Listing 7-8. DC1-Leaf-02 (VXLAN: No Multicast or Spine Required)

```
nv overlay evpn
feature ospf
feature bgp
feature fabric forwarding
feature interface-vlan
feature vn-segment-vlan-based
feature nv overlay

fabric forwarding anycast-gateway-mac cc1e.0006.2050
vlan 1,10,999
vlan 10
  vn-segment 10010
vlan 999
  vn-segment 99999

vrf context Tenant-1
  vni 99999
  rd auto
  address-family ipv4 unicast
    route-target import 65501:99999
    route-target import 65501:99999 evpn
    route-target export 65501:99999
    route-target export 65501:99999 evpn

interface Vlan10
  no shutdown
  vrf member Tenant-1
  ip address 10.10.0.1/24
  fabric forwarding mode anycast-gateway

interface Vlan999
  no shutdown
  vrf member Tenant-1
  ip forward

interface nve1
  no shutdown
  host-reachability protocol bgp
  source-interface loopback1
  member vni 10010
    suppress-arp
    ingress-replication protocol bgp
  member vni 99999 associate-vrf

interface Ethernet1/1
  mtu 9216
  medium p2p
  ip address 10.30.1.1/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  no shutdown

interface Ethernet1/2
  mtu 9216
  medium p2p
  ip address 10.40.1.1/30
  ip ospf network point-to-point
  ip router ospf Underlay area 0.0.0.0
  no shutdown

interface loopback0
  ip address 1.0.0.20/32
  ip router ospf Underlay area 0.0.0.0

interface loopback1
  ip address 20.20.20.20/32

router ospf Underlay
  router-id 1.0.0.20

router bgp 65501
```

```
  address-family ipv4 unicast
    network 20.20.20.20/32
  template peer LEAFS
    remote-as 65501
    update-source loopback0
    address-family ipv4 unicast
      send-community
      send-community extended
    address-family l2vpn evpn
      send-community
      send-community extended
  neighbor 1.0.0.10
    inherit peer LEAFS
  neighbor 1.0.0.30
    inherit peer LEAFS
  neighbor 1.0.0.40
    inherit peer LEAFS

  vrf Tenant-1
    address-family ipv4 unicast
      advertise l2vpn evpn

evpn
  vni 10010 l2
    rd auto
    route-target import auto
    route-target export auto
```

Index