Juniper Networks | Engineering Simplicity

Data Center EVPN-VXLAN Fabric Architecture Guide

Published
2020-07-14

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement (“EULA”) posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

CHAPTER 1

Data Center Fabric Blueprint Architecture—Overview

About This Architecture Guide

The purpose of this guide is to provide networking professionals with the concepts and tools needed to build multiservice data center fabric networks.

The intended audience for this guide includes system integrators, infrastructure professionals, partners, and customers that are currently using or considering upgrading to a high-end IP fabric data center fabric architecture.

Terminology

This section provides a summary of commonly used terms, protocols, and building block technologies used in creating and maintaining data center networks.

Glossary Terms

- **ARP**—Address Resolution Protocol. A protocol defined in RFC 826 for mapping a logical IP address to a physical MAC address.
- **Backbone Device**—A device in the WAN cloud that is directly connected to a spine device or devices in a data center. Backbone devices are required in this reference topology to provide physical connectivity between data centers that are interconnected using a data center interconnect (DCI).
- **Bridged Overlay**—An Ethernet-based overlay service designed for data center environments that do not require routing within an EVPN/VXLAN fabric. IP routing can provided externally to the fabric as needed.
- **BUM**—Broadcast, Unknown Unicast, and Multicast. The BUM acronym collectively identifies the three traffic types.
- **Centrally-Routed Bridging Overlay**—A form of IRB overlay that provides routing at a central gateway and bridging at the edge of the overlay network. In an IRB overlay, a routed overlay and one or more bridged overlays connect at one or more locations through the use of IRB interfaces.
- **Contrail Command**—Contrail Enterprise Multicloud user interface. Provides a consolidated, easy-to-use software designed to automate the creation and management of data center networks.
- **Contrail Enterprise Multicloud**—A suite of products and software that combines Contrail Command as a single point of management for private and public clouds, QFX Series switches running Junos OS as an infrastructure for data center networking, and Contrail Insights (formerly known as AppFormix) for telemetry and network visualization.

- **Clos Network**—A multistage network topology first developed by Charles Clos for telephone networks that provides multiple paths to a destination at each stage of the topology. Non-blocking networks are possible in a Clos-based topology.
- **DCI**—Data Center Interconnect. The technology used to interconnect separate data centers.
- **Default instance**—A global instance in a Juniper Networks device that hosts the primary routing table such as **inet.0** (default routing instance) and the primary MAC address table (default switching instance).
- **DHCP relay**—A function that allows a DHCP server and client to exchange DHCP messages over the network when they are not in the same Ethernet broadcast domain. DHCP relay is typically implemented at a default gateway.
- **EBGP**—External BGP. A routing protocol used to exchange routing information between autonomous networks. It has also been used more recently in place of a traditional Interior Gateway Protocols, such as IS-IS and OSPF, for routing within an IP fabric.
- **Edge-Routed Bridging Overlay**—A form of IRB overlay that provides routing and bridging at the edge of the overlay network.
- **End System**—An endpoint device that connects into the data center. An end system can be a wide range of equipment but is often a server, a router, or another networking device in the data center.
- **ESI**—Ethernet segment identifier. An ESI is a 10-octet integer that identifies a unique Ethernet segment in EVPN. In this blueprint architecture, LAGs with member links on different access devices are assigned a unique ESI to enable Ethernet multihoming.
- **Ethernet-connected Multihoming**—An Ethernet-connected end system that connects to the network using Ethernet access interfaces on two or more devices.
- **EVPN**—Ethernet Virtual Private Network. A VPN technology that supports bridged, routed, and hybrid network overlay services. EVPN is defined in RFC 7432 with extensions defined in a number of IETF draft standards.
- **EVPN Type 2 Route**—Advertises MAC addresses and the associated IP addresses from end systems to devices participating in EVPN.
- **IBGP**—Internal BGP. In this blueprint architecture, IBGP with Multiprotocol BGP (MP-IBGP) is used for EVPN signalling between the devices in the overlay.
- **IP Fabric**—An all-IP fabric network infrastructure that provides multiple symmetric paths between all devices in the fabric.
- **IP-connected Multihoming**—An IP-connected end system that connects to the network using IP access interfaces on two or more devices.
- **IRB**—Integrated Routing and Bridging. A technique that enables routing between VLANs and allows traffic to be routed or bridged based on whether the destination is outside or inside of a bridging domain. To activate IRB, you associate a logical interface (IRB interface) with a VLAN and configure the IRB interface with an IP address for the VLAN subnet.

- **NDP**—Neighbor Discovery Protocol. An IPv6 protocol defined in RFC 4861 that combines the functionality of ARP and ICMP, and adds other enhanced capabilities.
- **NVE**—Network Virtualization Edge devices.
- **NVO**—Network virtualization overlay.
- **Routed Overlay**—An IP-based overlay service where no Ethernet bridging is required. Also referred to as an IP VPN. In this blueprint architecture, the routed overlay is based on EVPN Type 5 routes and their associated procedures, and supported by VXLAN tunneling.
- **Leaf Device**—An access level network device in an IP fabric topology. End systems connect to the leaf devices in this blueprint architecture.
- **Multiservice Cloud Data Center Network**—A data center network that optimizes the use of available compute, storage, and network access interfaces by allowing them to be shared flexibly across diverse applications, tenants, and use cases.
- **Spine Device**—A centrally-located device in an IP fabric topology that has a connection to each leaf device.
- **Storm Control**—Feature that prevents BUM traffic storms by monitoring BUM traffic levels and taking a specified action to limit BUM traffic forwarding when a specified traffic level is exceeded.
- **Underlay Network**—A network that provides basic network connectivity between devices.In this blueprint architecture, the underlay network is an IP Fabric that provides basic IP connectivity.
- **VLAN trunking**—The ability for one interface to support multiple VLANs.
- **VNI**—VXLAN Network Identifier. Uniquely identifies a VXLAN virtual network. A VNI encoded in a VXLAN header can support 16 million virtual networks.
- **VTEP**—VXLAN Tunnel Endpoint. A loopback or virtual interface where traffic enters and exits a VXLAN tunnel. Tenant traffic is encapsulated into VXLAN packets at a source VTEP, and de-encapsulated when the traffic leaves the VXLAN tunnel at a remote VTEP.
- **VXLAN**—Virtual Extensible LAN. Network virtualization tunneling protocol defined in RFC 7348 used to build virtual networks over an IP-routed infrastructure. VXLAN is used to tunnel tenant traffic over the IP fabric underlay from a source endpoint at an ingress device to a destination endpoint at the egress device. These tunnels are established dynamically by EVPN. Each VTEP device advertises its loopback address in the underlay network for VXLAN tunnel reachability between VTEP devices.

RELATED DOCUMENTATION

Contrail Enterprise Multicloud Components

Introducing the Data Center Fabric Blueprint Architecture

IN THIS SECTION

Data Center Fabric Blueprint Architecture Introduction

IN THIS SECTION

This section provides an introduction to the Data Center Fabric Blueprint Architecture.

It includes the following sections.

The Next Act for Data Center Networks

For truly cloud-native workloads that have no dependency on Ethernet broadcast, multicast, segmentation, multitenancy, or workload mobility, the best solution is typically a simple IP fabric network. When a unique workload instance requires mobility, the current host system can advertise the unique IP address of the workload. This can be performed with EBGP route exchange between the host system and the ToR. However, support for BUM and multitenancy require more advanced network functions. This is where overlays are added.

Over its evolution the data center network was a function of the demands and expectations placed on it. As the nature of workloads changed, the network had to adapt. Each solution simplified a set of problems by trading one form of complexity and cost for another. The cloud network is no different. In the end, bits must be moved from point A to point B reliably, securely, and at the desired throughput. Where operators need a single network to serve more than one purpose (the multiservice cloud network), they can add network-layer segmentation and other functions to share the infrastructure across diverse groups of

endpoints and tenants. Operational simplicity is achieved with a centralized controller that implements an intent model that is consistent with the cloud scale functions of the network layer. Technical simplicity is achieved using a reduced set of building blocks that are based on open standards and homogeneous across the end-to-end network.

This guide introduces a building block approach to creating multiservice cloud networks on the foundation of a modern IP fabric. The Juniper Networks solutions team will systematically review the set of building blocks required for an agile network, focus on specific, state-of-the-art, open standards-based technology that enables each function, and add new functionality to the guide as it becomes available in future software releases.

All the building blocks are fully synergistic and you can combine any of the building blocks with any other to satisfy a diverse set of use cases simultaneously — this is the hallmark of the cloud. You should consider how you can leverage the building blocks in this guide to achieve the use cases that are relevant to your network.

Building Blocks

The guide organizes the technologies used to build multiservice cloud network architectures into modular building blocks. Each building block includes features that either must be implemented together to build the network, are often implemented together because they provide complementary functionality, or are presented together to provide choices for particular technologies.

This blueprint architecture includes required building blocks and optional building blocks. The optional building blocks can be added or removed to support the needs of a specific multiservice data center fabric network.

This guide walks you through the design and technology choices associated with each building block, and provides information designed to help you choose the building blocks that best meet the needs for your multiservice data center fabric network. The guide also provides the implementation procedures for each building block.

The currently-supported building blocks include:

- IP Fabric Underlay Network
- Network Virtualization Overlays
  - Centrally-Routed Bridging Overlay
  - Edge-Routed Bridging Overlay
  - Routed Overlay
- Multihoming
  - Multihoming of Ethernet-connected End Systems

  - Multihoming of IP-connected End Systems
- Data Center Interconnect (DCI)
- Service Chaining
- Multicast
- Ingress Virtual Machine Traffic Optimization
- DHCP Relay
- Proxy ARP
- Layer 2 Port Security

Additional building blocks will be added to this guide as support for the technology becomes available and is validated by the Juniper Networks testing team.

Each building block is discussed in more detail in "Data Center Fabric Blueprint Architecture Components" on page 15.

For information about the hardware and software that serve as a foundation to your building blocks, see Table 1 on page 46.

RELATED DOCUMENTATION

Data Center Fabric Reference Design Overview and Validated Topology | 44

Data Center Fabric Blueprint Architecture Components | 15

Data Center Fabric Blueprint Architecture Components

IN THIS SECTION

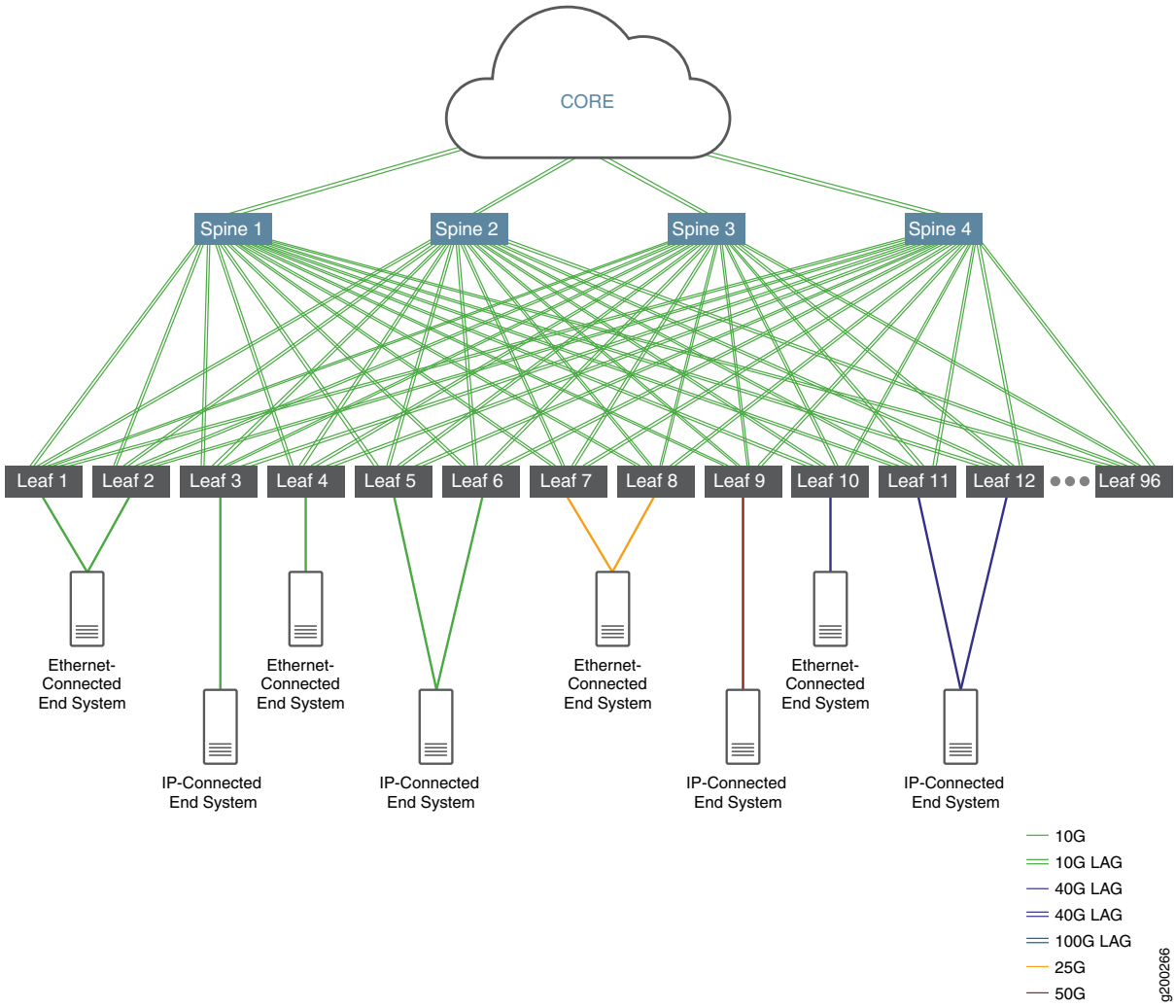
This section gives an overview of the building blocks used in this blueprint architecture. The implementation of each building block technology is explored in more detail later sections.

For information about the hardware and software that serve as a foundation to your building blocks, see the Table 1 on page 46.

The building blocks include:

IP Fabric Underlay Network

Figure 1: IP Fabric Underlay



The modern IP fabric underlay network building block provides IP connectivity across a Clos-based topology.

As shown in Figure 1 on page 17, the leaf and spine devices are interconnected using high-speed interfaces that are either single links or aggregated Ethernet interfaces. The aggregated Ethernet interfaces are optional—a single link between spine and leaf devices is typically used— but can be deployed to increase bandwidth and provide link level redundancy. Both options are covered.

We chose EBGP as the routing protocol in the underlay network for its dependability and scalability. Each spine and leaf device is assigned its own autonomous system with a unique autonomous system number to support EBGP. You can use other routing protocols in the underlay network; the usage of those protocols is beyond the scope of this document.

Micro Bidirectional Forwarding Detection (BFD)—the ability to run BFD on individual links in an aggregated Ethernet interface—can also be enabled in this building block to quickly detect link failures on any member links in aggregated Ethernet bundles that connect spine and leaf devices.

For information about implementing the IP fabric underlay network building block, see "IP Fabric Underlay Network Design and Implementation" on page 51.

IPv4 and IPv6 Support

Because many networks implement a dual stack environment that includes *IPv4 and IPv6*, this blueprint provides support for both IP protocols. IPv4 and IPv6 are interwoven throughout this guide to allow you to pick one or both of these protocols.

Network Virtualization Overlays

IN THIS SECTION

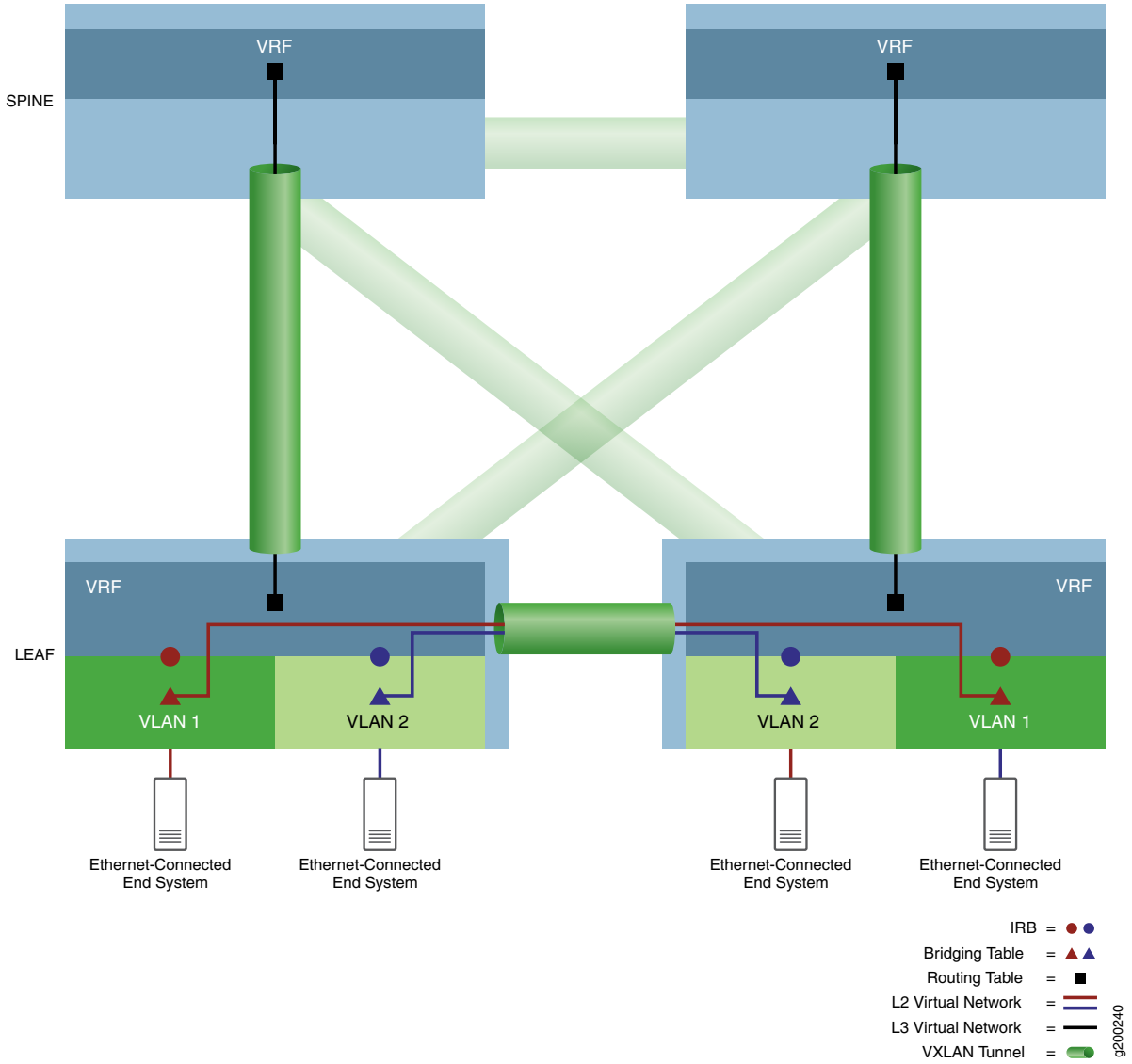
A *network virtualization overlay* is a virtual network that is transported over an IP underlay network. This building block enables multitenancy in a network, allowing you to share a single physical network across multiple tenants, while keeping each tenant's network traffic isolated from the other tenants.

A tenant is a user community (such as a business unit, department, workgroup, or application) that contains groups of endpoints. Groups may communicate with other groups in the same tenancy, and tenants may communicate with other tenants if permitted by network policies. A group is typically expressed as a subnet (VLAN) that can communicate with other devices in the same subnet, and reach external groups and endpoints by way of a virtual routing and forwarding (VRF) instance.

As seen in the overlay example shown in Figure 2 on page 19, Ethernet bridging tables (represented by triangles) handle tenant bridged frames and IP routing tables (represented by squares) process routed

packets. Inter-VLAN routing happens at the integrated routing and bridging (IRB) interfaces (represented by circles). Ethernet and IP tables are directed into virtual networks (represented by colored lines). To reach end systems attached to other VXLAN Tunnel Endpoint (VTEP) devices, tenant packets are encapsulated and sent over an EVPN-signalled VXLAN tunnel (represented by green tunnel icons) to the associated remote VTEP devices. Tunneled packets are de-encapsulated at the remote VTEP devices and forwarded to the remote end systems by way of the respective bridging or routing tables of the egress VTEP device.

Figure 2: VXLAN Tunnels—Ethernet Bridging, IP Routing, and IRB



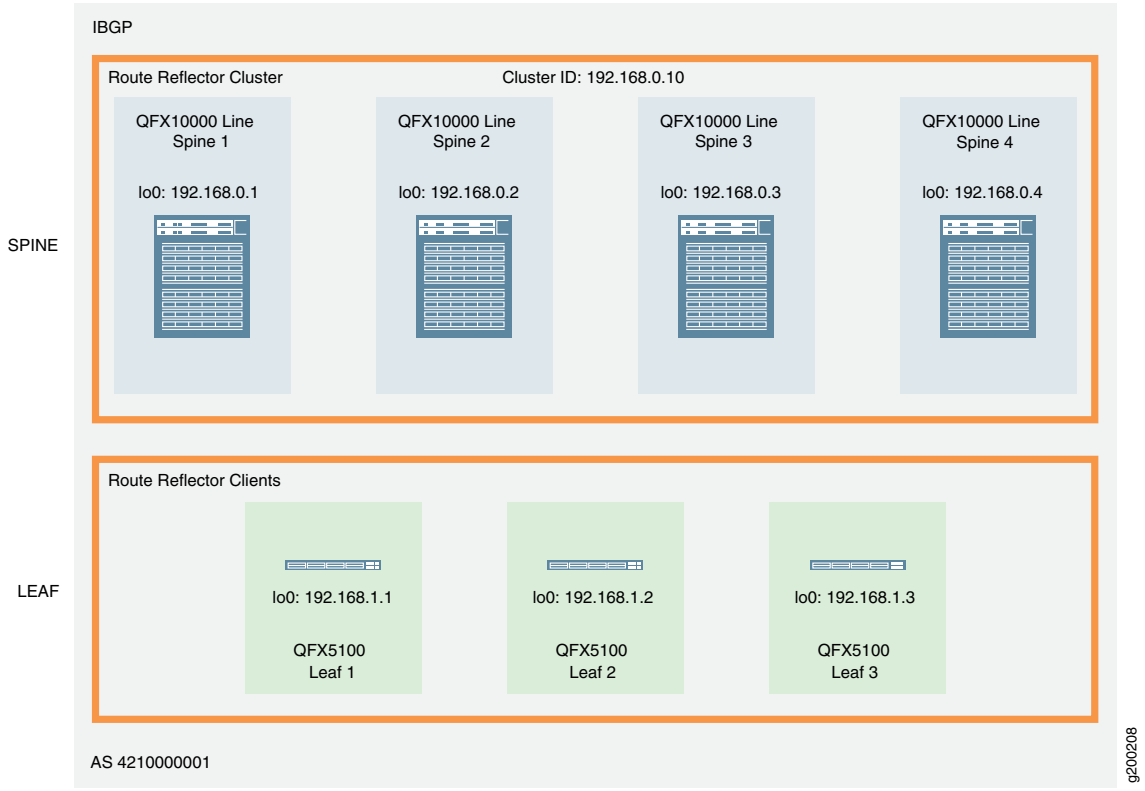
The following sections provide more details about overlay networks:

IBGP for Overlays

Internal BGP (IBGP) is a routing protocol that exchanges reachability information across an IP network. When IBGP is combined with Multiprotocol BGP (MP-IBGP), it provides the foundation for EVPN to exchange reachability information between VTEP devices. This capability is required to establish inter-VTEP VXLAN tunnels and use them for overlay connectivity services.

Figure 3 on page 20 shows that the spine and leaf devices use their loopback addresses for peering in a single autonomous system. In this design, the spine devices act as a route reflector cluster and the leaf devices are route reflector clients. Use of a route reflector satisfies the IBGP requirement for a full mesh without the need to peer all the VTEP devices directly with one another. As a result, the leaf devices peer only with the spine devices and the spine devices peer with both spine devices and leaf devices. Because the spine devices are connected to all the leaf devices, the spine devices can relay IBGP information between the indirectly peered leaf device neighbors.

Figure 3: IBGP for Overlays



You can place route reflectors almost anywhere in the network. However, you must consider the following:

- Does the selected device have enough memory and processing power to handle the additional workload required by a route reflector?

- Is the selected device equidistant and reachable from all EVPN speakers?
- Does the selected device have the proper software capabilities?

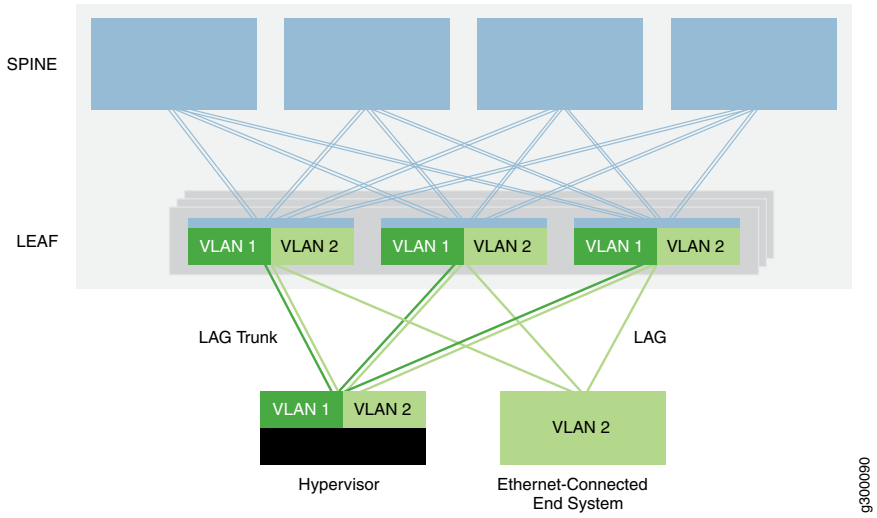
In this design, the route reflector cluster is placed at the spine layer. The QFX switches that you can use as a spine in this reference design have ample processing speed to handle route reflector client traffic in the network virtualization overlay.

For details about implementing IBGP in an overlay, see "Configuring IBGP for the Overlay" on page 67.

Bridged Overlay

The first overlay service type described in this guide is a *bridged overlay*, as shown in Figure 4 on page 21.

Figure 4: Bridged Overlay



In this overlay model, Ethernet VLANs are extended between leaf devices across VXLAN tunnels. These leaf-to-leaf VXLAN tunnels support data center networks that require Ethernet connectivity between leaf devices but do not need routing between the VLANs. As a result, the spine devices provide only basic underlay and overlay connectivity for the leaf devices, and do not perform routing or gateway services seen with other overlay methods.

Leaf devices originate VTEPs to connect to the other leaf devices. The tunnels enable the leaf devices to send VLAN traffic to other leaf devices and Ethernet-connected end systems in the data center. The simplicity of this overlay service makes it attractive for operators who need an easy way to introduce EVPN/VXLAN into their existing Ethernet-based data center.

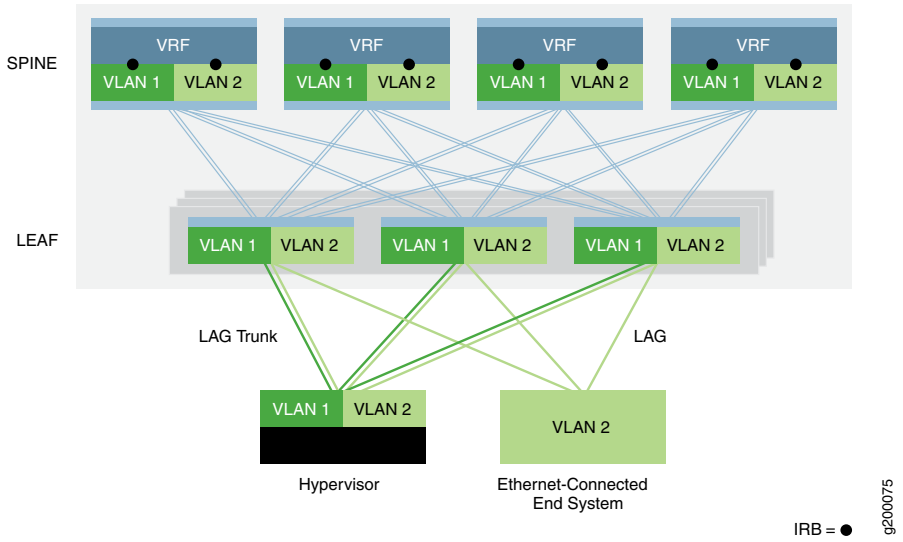
> **NOTE:** You can add routing to a bridged overlay by implementing an MX Series router or SRX Series security device external to the EVPN/VXLAN fabric. Otherwise, you can select one of the other overlay types that incorporate routing (such as an "edge-routed bridging overlay" on page 158, a "centrally-routed bridging overlay" on page 95, or a "routed overlay" on page 175).

For information on implementing a bridged overlay, see "Bridged Overlay Design and Implementation" on page 73.

Centrally-Routed Bridging Overlay

The second overlay service type is the *centrally-routed bridging overlay*, as shown in Figure 5 on page 22.

Figure 5: Centrally-Routed Bridging Overlay



In a centrally-routed bridging overlay routing occurs at a central gateway of the data center network (the spine layer in this example) rather than at the VTEP device where the end systems are connected (the leaf layer in this example).

You can use this overlay model when you need routed traffic to go through a centralized gateway or when your edge VTEP devices lack the required routing capabilities.

As shown above, traffic that originates at the Ethernet-connected end systems is forwarded to the leaf VTEP devices over a trunk (multiple VLANs) or an access port (single VLAN). The VTEP device forwards the traffic to local end systems or to an end system at a remote VTEP device. An integrated routing and bridging (IRB) interface at each spine device helps route traffic between the Ethernet virtual networks.

EVPN supports two VLAN-aware Ethernet service models in the data center. Juniper Networks supports both models. They are as follows:

- **VLAN-Aware**--This bridging overlay service model allows a collection of VLANs to be easily aggregated into the same overlay virtual network. It provides two options:

  1. **Default Instance VLAN-Aware**—In this option, you implement a single, default switching instance that supports a total of 4094 VLANs. All leaf platforms included in this design (see Table 1 on page 46) support the default instance style of VLAN-aware overlay.

     To configure this service model, see Configuring a VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance.

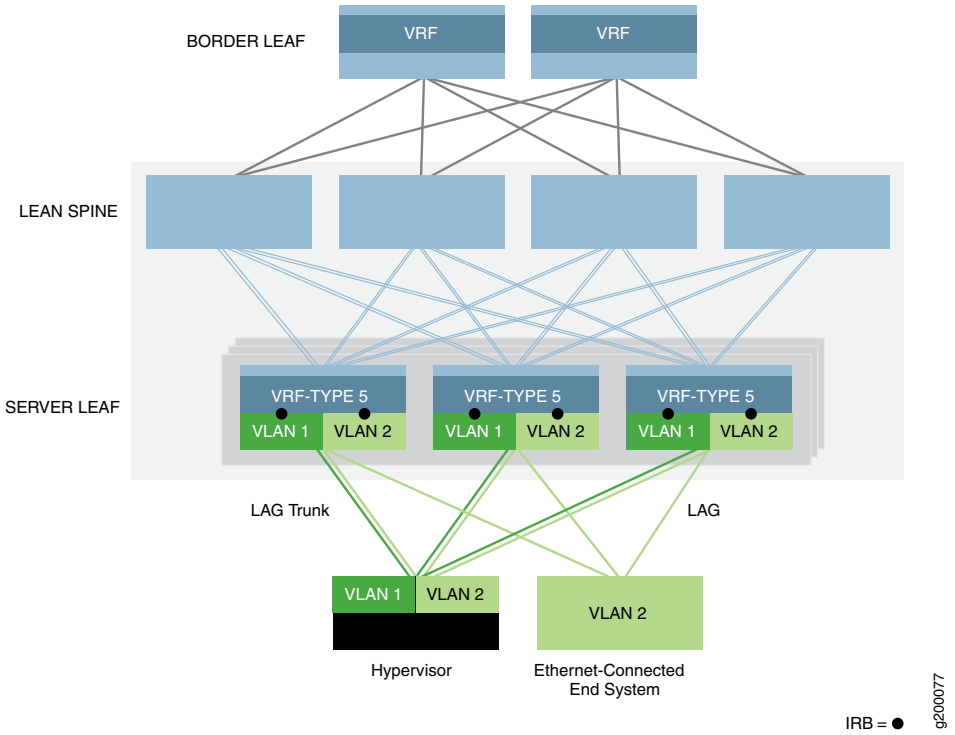
  2. **Virtual Switch VLAN-Aware**—In this option, multiple virtual switch instances support 4094 VLANs per instance. This Ethernet service model is ideal for overlay networks that require scalability beyond a single default instance. Support for this option is available currently on the QFX10000 line of switches.

     To implement this scalable service model, see Configuring a VLAN-Aware Centrally-Routed Bridging Overlay with Virtual Switches.

Edge-Routed Bridging Overlay

The third overlay service option is the *edge-routed bridging overlay*, as shown in Figure 6 on page 24.

Figure 6: Edge-Routed Bridging Overlay



In this Ethernet service model, the IRB interfaces are moved to leaf device VTEPs at the edge of the overlay network to bring IP routing closer to the end systems. Because of the special ASIC capabilities required to support bridging, routing, and EVPN/VXLAN in one device, edge-routed bridging overlays are only possible on certain switches. For a list of switches that we support as leaf devices in an edge-routed bridging overlay, see Table 1 on page 46.

This model allows for a simpler overall network. The spine devices are configured to handle only IP traffic, which removes the need to extend the bridging overlays to the spine devices.

This option also enables faster server-to-server, intra-data center traffic (also known as east-west traffic) where the end systems are connected to the same leaf device VTEP. As a result, routing happens much closer to the end systems than with centrally-routed bridging overlays.

> **NOTE:** When a QFX5110 or QFX5120 switch that functions as a leaf device is configured with IRB interfaces that are included in EVPN Type-5 routing instances, symmetric inter-IRB unicast routing is automatically enabled.

For information on implementing the edge-routed bridging overlay, see "Edge-Routed Bridging Overlay Design and Implementation" on page 158.

IRB Addressing Models in Bridging Overlays

The configuration of IRB interfaces in centrally-routed bridging and edge-routed bridging overlays requires an understanding of the models for the default gateway IP and MAC address configuration of IRB interfaces as follows:

- **Unique IRB IP Address**—In this model, a unique IP address is configured on each IRB interface in an overlay subnet.

  The benefit of having a unique IP address and MAC address on each IRB interface is the ability to monitor and reach each of the IRB interfaces from within the overlay using its unique IP address. This model also allows you to configure a routing protocol on the IRB interface.

  The downside of this model is that allocating a unique IP address to each IRB interface may consume many IP addresses of a subnet.

- **Unique IRB IP Address with Virtual Gateway IP Address**—This model adds a virtual gateway IP address to the previous model, and we recommend it for centrally-routed bridged overlays. It is similar to VRRP, but without the in-band data plane signaling between the gateway IRB interfaces. The virtual gateway should be the same for all default gateway IRB interfaces in the overlay subnet and is active on all gateway IRB interfaces where it is configured. You should also configure a common IPv4 MAC address for the virtual gateway, which becomes the source MAC address on data packets forwarded over the IRB interface.

  In addition to the benefits of the previous model, the virtual gateway simplifies default gateway configuration on end systems. The downside of this model is the same as the previous model.

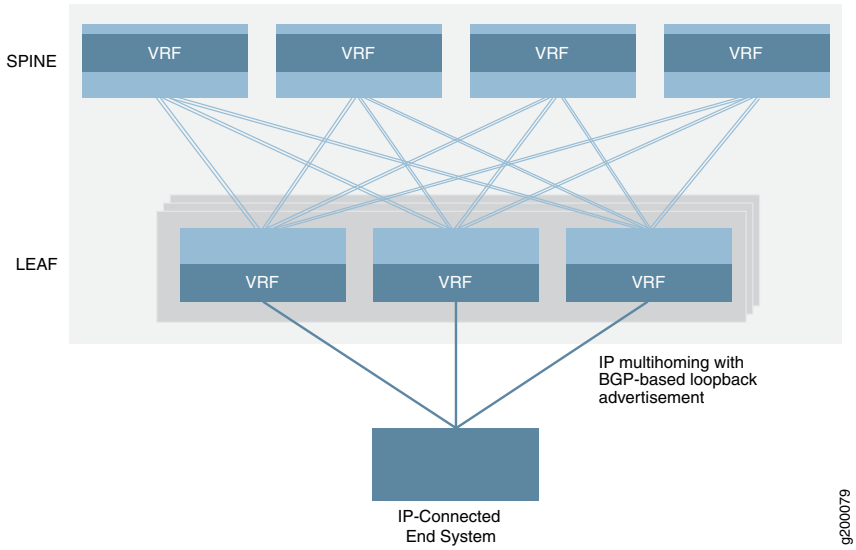
- **IRB with Anycast IP Address and MAC Address**—In this model, all default gateway IRB interfaces in an overlay subnet are configured with the same IP and MAC address. We recommend this model for edge-routed bridging overlays.

  A benefit of this model is that only a single IP address is required per subnet for default gateway IRB interface addressing, which simplifies default gateway configuration on end systems.

Routed Overlay using EVPN Type 5 Routes

The final overlay option is a *routed overlay*, as shown in Figure 7 on page 26.

Figure 7: Routed Overlay



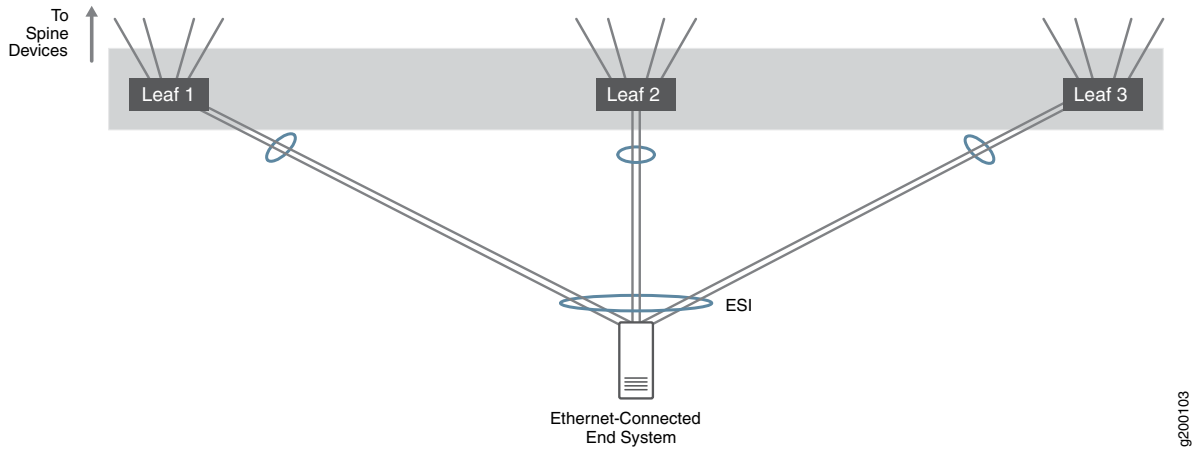
This option is an IP-routed virtual network service. Unlike an MPLS-based IP VPN, the virtual network in this model is based on EVPN/VXLAN.

Cloud providers prefer this virtual network option because most modern applications are optimized for IP. Because all communication between devices happens at the IP layer, there is no need to use any Ethernet bridging components, such as VLANs and ESIs, in this routed overlay model.

For information on implementing a routed overlay, see “Routed Overlay Design and Implementation” on page 175.

Multihoming Support for Ethernet-Connected End Systems

Figure 8: Ethernet-Connected End System Multihoming



Ethernet-connected multihoming allows Ethernet-connected end systems to connect into the Ethernet overlay network over a single-homed link to one VTEP device or over multiple links multihomed to different VTEP devices. Ethernet traffic is load-balanced across the fabric between VTEPs on leaf devices that connect to the same end system.

We tested setups where an Ethernet-connected end system was connected to a single leaf device or multihomed to 2 or 3 leaf devices to prove traffic is properly handled in multihomed setups with more than two leaf VTEP devices; in practice, an Ethernet-connected end system can be multihomed to a large number of leaf VTEP devices. All links are active and network traffic can be load balanced over all of the multihomed links.

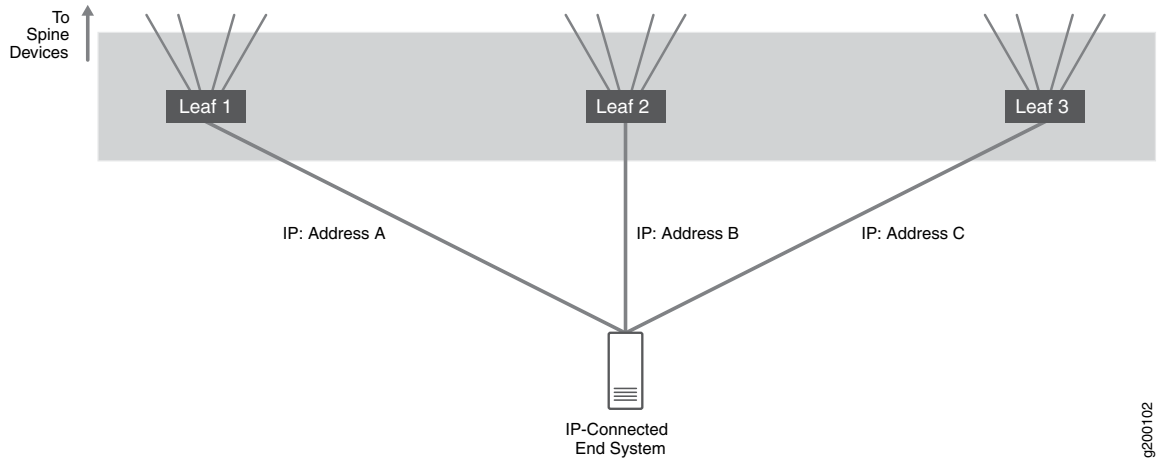
In this architecture, EVPN is used for Ethernet-connected multihoming. EVPN multihomed LAGs are identified by an Ethernet segment identifier (ESI) in the EVPN bridging overlay while LACP is used to improve LAG availability.

VLAN trunking allows one interface to support multiple VLANs. VLAN trunking ensures that virtual machines (VMs) on non-overlay hypervisors can operate in any overlay networking context.

For more information about Ethernet-connected multihoming support, see "Multihoming an Ethernet-Connected End System Design and Implementation" on page 151.

Multihoming Support for IP-Connected End Systems

Figure 9: IP-Connected End System Multihoming



IP-connected multihoming endpoint systems to connect to the IP network over multiple IP-based access interfaces on different leaf devices.

We tested setups where an IP–connected end system was connected to a single leaf or multihomed to 2 or 3 leaf devices. The setup validated that traffic is properly handled when multihomed to multiple leaf devices; in practice, an IP-connected end system can be multihomed to a large number of leaf devices.

In multihomed setups, all links are active and network traffic is forwarded and received over all multihomed links. IP traffic is load balanced across the multihomed links using a simple hashing algorithm.

EBGP is used to exchange routing information between the IP-connected endpoint system and the connected leaf devices to ensure the route or routes to the endpoint systems are shared with all spine and leaf devices.

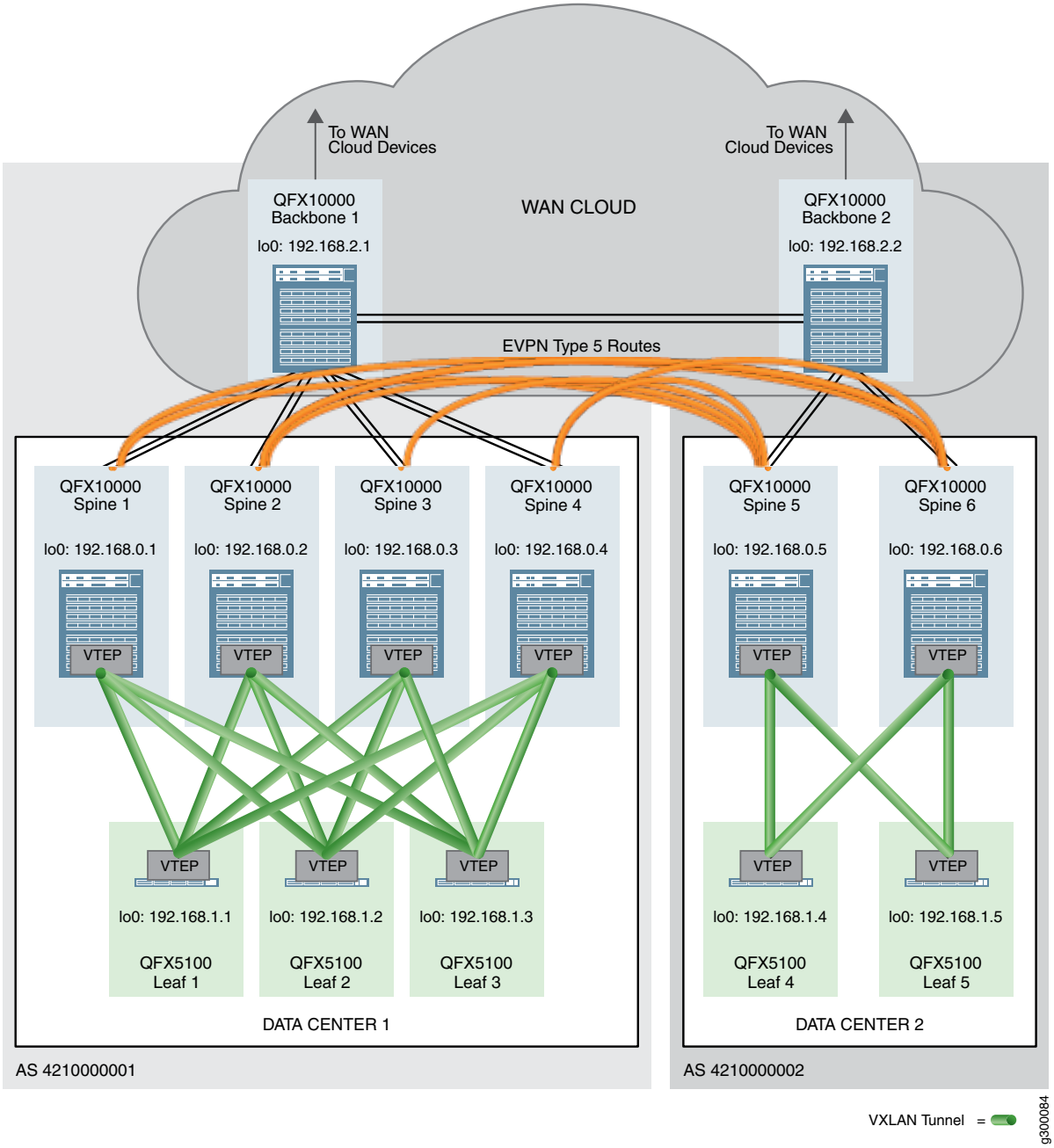
For more information about the IP-connected multihoming building block, see "Multihoming an IP-Connected End System Design and Implementation" on page 189.

Data Center Interconnect (DCI)

The data center interconnect (DCI) building block provides the technology needed to send traffic between data centers. The validated design supports DCI using EVPN Type 5 routes or IPVPN routes.

EVPN Type 5 or IPVPN routes are used in a DCI context to ensure inter-data center traffic between data centers using different IP address subnetting schemes can be exchanged. Routes are exchanged between spine devices in different data centers to allow for the passing of traffic between data centers.

Figure 10: DCI Using EVPN Type 5 Routes Topology Overview



Physical connectivity between the data centers is required before EVPN Type 5 messages or IPVPN routes can be sent between data centers. The physical connectivity is provided by backbone devices in a WAN cloud. A backbone device is connected to all spine devices in a single data center, as well as to the other backbone devices that are connected to the other data centers.

For information about configuring DCI, see:

- Data Center Interconnect Design and Implementation Using Type 5 Routes on page 193
- Data Center Interconnect Design and Implementation Using IPVPN on page 220

Service Chaining

In many networks, it is common for traffic to flow through separate hardware devices that each provide a service, such as firewalls, NAT, IDP, multicast, and so on. Each device requires separate operation and management. This method of linking multiple network functions can be thought of as physical service chaining.

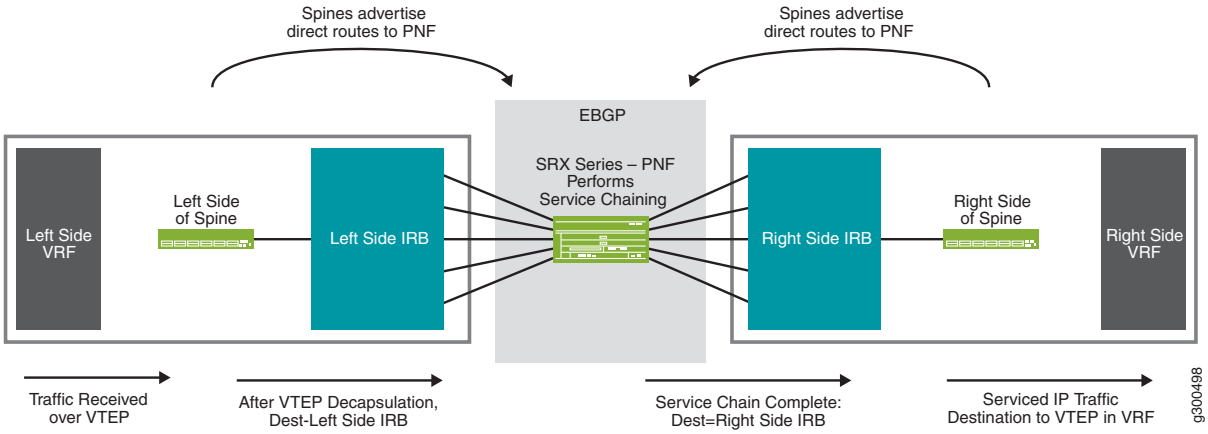
A more efficient model for service chaining is to virtualize and consolidate network functions onto a single device. In our blueprint architecture, we are using the SRX Series routers as the device that consolidates network functions and processes and applies services. That device is called a physical network function (PNF).

In this solution, service chaining is supported on both centrally-routed bridging overlay and edge-routed bridging overlay. It works only for inter-tenant traffic.

Logical View of Service Chaining

Figure 11 on page 30 shows a logical view of service chaining. It shows one spine with a right side configuration and a left side configuration. On each side is a VRF routing instance and an IRB interface. The SRX Series router in the center is the PNF, and it performs the service chaining.

Figure 11: Service Chaining Logical View



The flow of traffic in this logical view is:

1. The spine receives a packet on the VTEP that is in the left side VRF.
2. The packet is decapsulated and sent to the left side IRB interface.
3. The IRB interface routes the packet to the SRX Series router, which is acting as the PNF.
4. The SRX Series router performs service chaining on the packet and forwards the packet back to the spine, where it is received on the IRB interface shown on the right side of the spine.
5. The IRB interface routes the packet to the VTEP in the right side VRF.

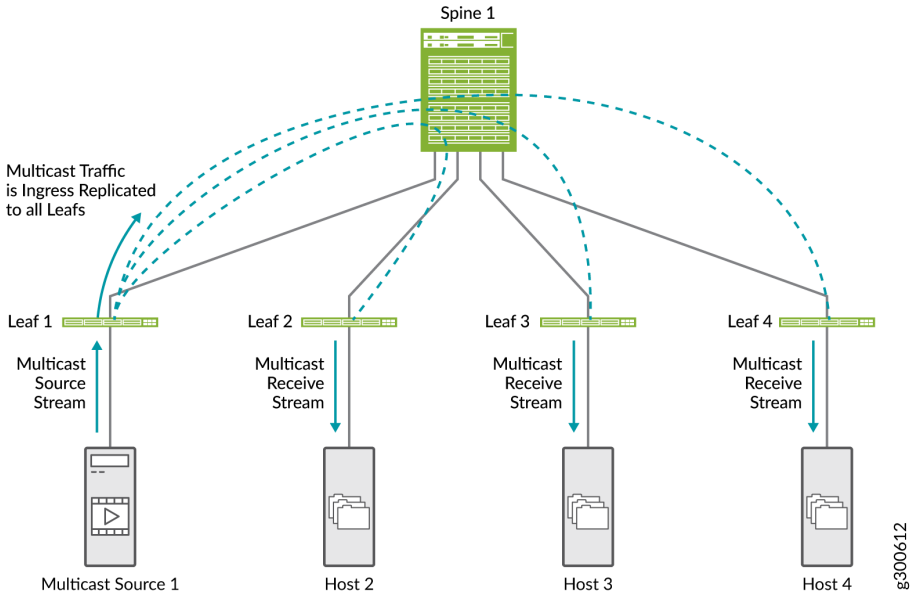
For information about configuring service chaining, see "Service Chaining Design and Implementation" on page 224.

Multicast Optimizations

IN THIS SECTION

Multicast optimizations help to preserve bandwidth and more efficiently route traffic in a multicast scenario in EVPN VXLAN environments. Without any multicast optimizations configured, all multicast replication is done at the ingress of the leaf connected to the multicast source as shown in Figure 12 on page 32. Multicast traffic is sent to all leaf devices that are connected to the spine. Each leaf device sends traffic to connected hosts.

Figure 12: Multicast without Optimizations



There are three types of multicast optimizations supported in EVPN VXLAN environments:

For information about Multicast support, see *Multicast Support in EVPN-VXLAN Overlay Networks*.

For information about configuring Multicast, see "Multicast Optimization Design and Implementation" on page 248.

IGMP Snooping

IGMP snooping in an EVPN-VXLAN fabric is useful to optimize the distribution of multicast traffic. IGMP snooping preserves bandwidth because multicast traffic is forwarded only on interfaces where there are IGMP listeners. Not all interfaces on a leaf device need to receive multicast traffic.

Without IGMP snooping, end systems receive IP multicast traffic that they have no interest in, which needlessly floods their links with unwanted traffic. In some cases when IP multicast flows are large, flooding unwanted traffic causes denial-of-service issues.

Figure 13 on page 33 shows how IGMP snooping works in an EVPN-VXLAN fabric. In this sample EVPN-VXLAN fabric, IGMP snooping is configured on all leaf devices, and multicast receiver 2 has previously sent an IGMPv2 join request.

1. Multicast Receiver 2 sends an IGMPv2 leave request.

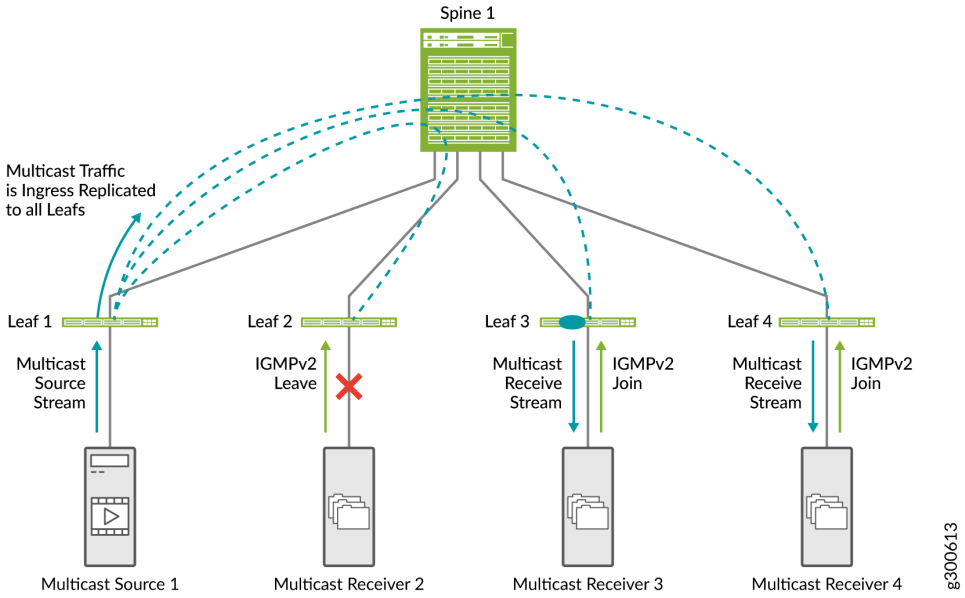
2. Multicast Receivers 3 and 4 send an IGMPv2 join request.

3. When leaf 1 receives ingress multicast traffic, it replicates it for all leaf devices, and forwards it to the spine.

4. The spine forwards the traffic to all leaf devices.

5. Leaf 2 receives the multicast traffic, but does not forward it to the receiver because the receiver sent an IGMP leave message.

Figure 13: Multicast with IGMP Snooping



In EVPN-VXLAN networks only IGMP version 2 is supported.

For more information about IGMP snooping, see *Overview of Multicast Forwarding with IGMP Snooping in an EVPN-VXLAN Environment*.

Selective Multicast Forwarding

Selective multicast Ethernet (SMET) forwarding provides greater end-to-end network efficiency and reduces traffic in the EVPN network. It conserves bandwidth usage in the core of the fabric and reduces the load on egress devices that do not have listeners.

Devices with IGMP snooping enabled use selective multicast forwarding to forward multicast traffic in an efficient way. With IGMP snooping enabled a leaf device sends multicast traffic only to the access interface with an interested receiver. With SMET added, the leaf device selectively sends multicast traffic to only the leaf devices in the core that have expressed an interest in that multicast group.

Figure 14 on page 34 shows the SMET traffic flow along with IGMP snooping.

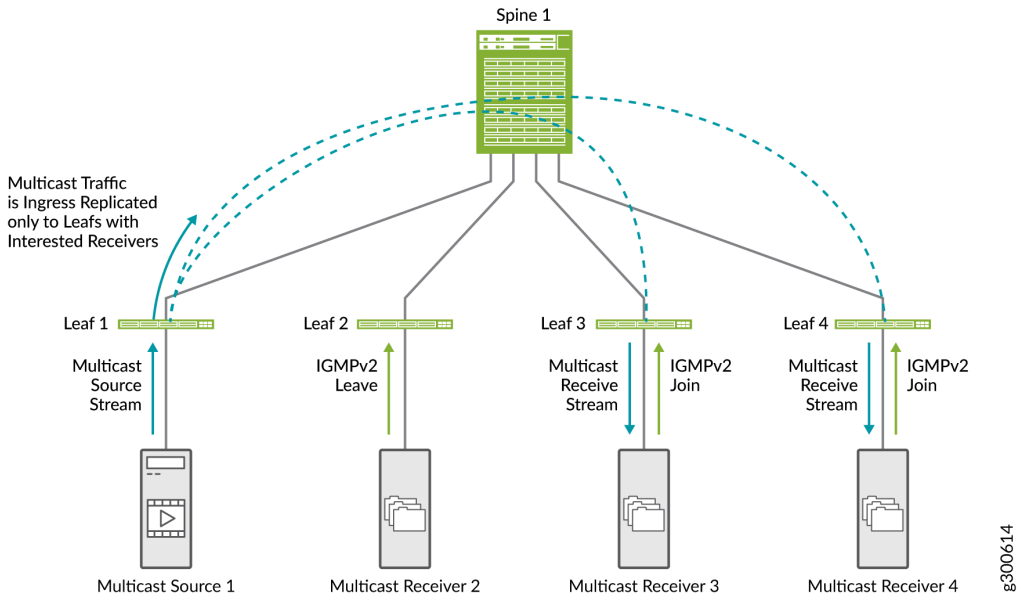
1. Multicast Receiver 2 sends an IGMPv2 leave request.

2. Multicast Receivers 3 and 4 send an IGMPv2 join request.

3. When leaf 1 receives ingress multicast traffic, it replicates the traffic only to leaf devices with interested receivers (leaf devices 3 and 4), and forwards it to the spine.

4. The spine forwards the traffic to leaf devices 3 and 4.

Figure 14: Selective Multicast Forwarding with IGMP Snooping



You do not need to enable SMET; it is enabled by default when IGMP snooping is configured on the device.

For more information about SMET, see *Overview of Selective Multicast Forwarding*.

Assisted Replication of Multicast Traffic

The assisted replication (AR) feature offloads EVPN-VXLAN fabric leaf devices from ingress replication tasks. The ingress leaf does not replicate multicast traffic. It sends one copy of the multicast traffic to a spine that is configured as an AR replicator device. The AR replicator device distributes and controls multicast traffic. In addition to reducing the replication load on the ingress leaf devices, this method conserves bandwidth in the fabric between the leaf and the spine.

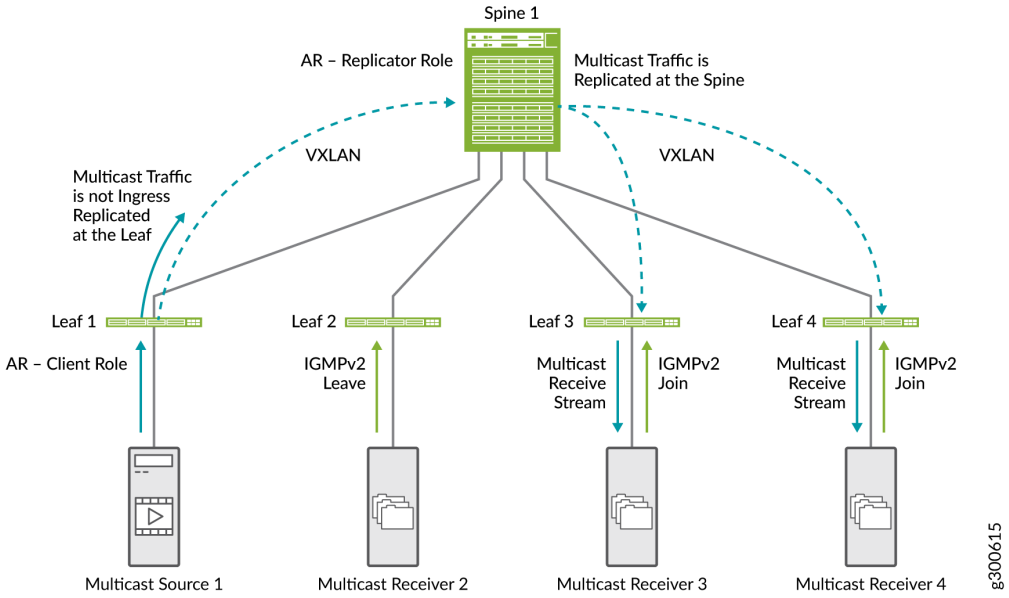
Figure 15 on page 35 shows how AR works along with IGMP snooping and SMET.

1. Leaf 1, which is set up as the AR leaf device, receives multicast traffic and sends one copy to the spine that is set up as the AR replicator device.

2. The spine replicates the multicast traffic. It replicates traffic for leaf devices that are provisioned with the VLAN VNI in which the multicast traffic originated from Leaf 1.

   Because we have IGMP snooping and SMET configured in the network, the spine sends the multicast traffic only to leaf devices with interested receivers.

Figure 15: Multicast with AR, IGMP Snooping, and SMET



In this document, we are showing multicast optimizations on a small scale. In a full-scale network with many spines and leafs, the benefits of the optimizations are much more apparent.

Ingress Virtual Machine Traffic Optimization for EVPN

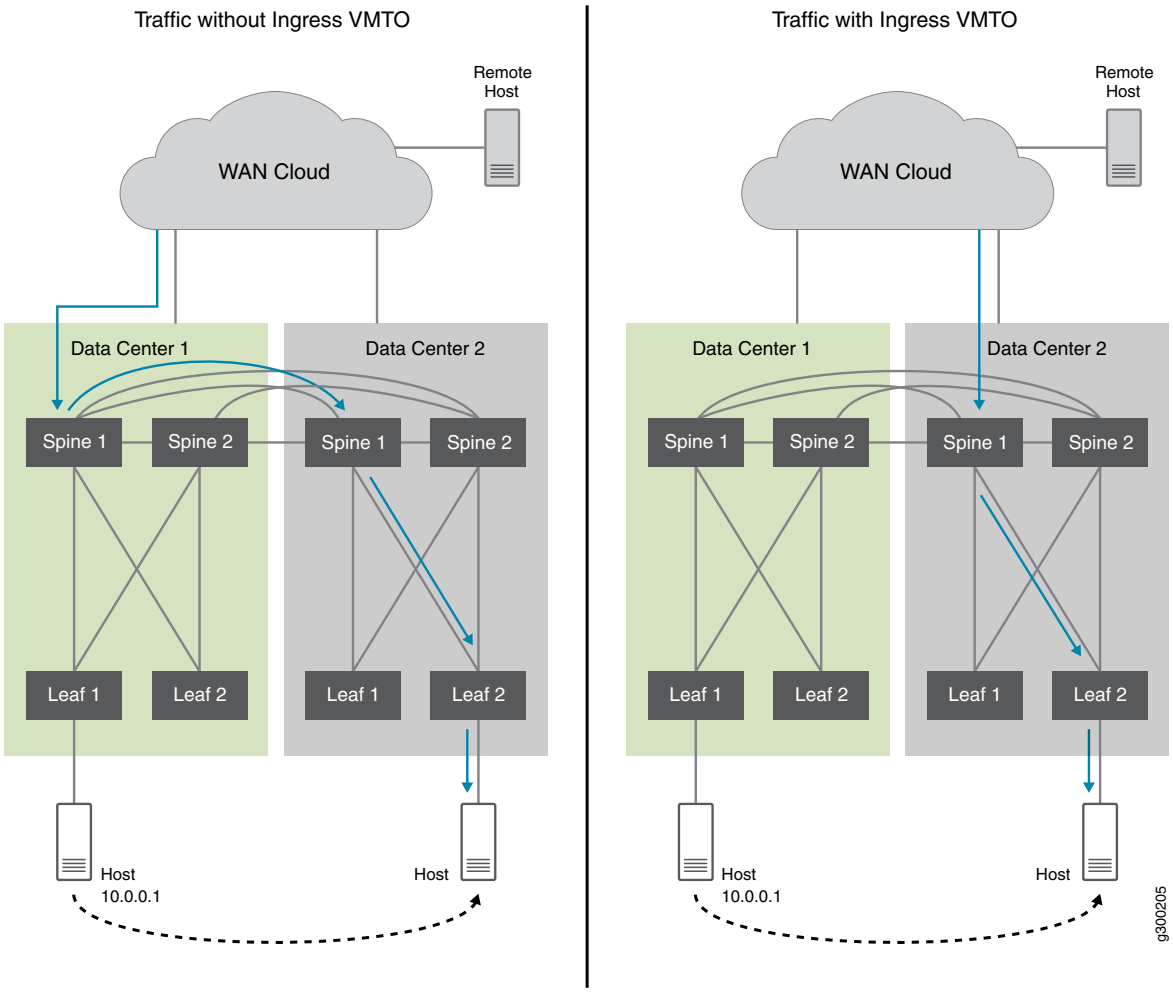
When virtual machines and hosts are moved within a data center or from one data center to another, network traffic can become inefficient if the traffic is not routed to the optimal gateway. This can happen when a host is relocated. The ARP table does not always get flushed and data flow to the host is sent to the configured gateway even when there is a more optimal gateway. The traffic is “tromboned” and routed unnecessarily to the configured gateway.

Ingress Virtual Machine Traffic Optimization (VMTO) provides greater network efficiency and optimizes ingress traffic and can eliminate the trombone effect between VLANs. When you enable ingress VMTO, routes are stored in a Layer 3 virtual routing and forwarding (VRF) table and the device routes inbound traffic directly back to host that was relocated.

Figure 16 on page 37 shows tromboned traffic without ingress VMTO and optimized traffic with ingress VMTO enabled.

- Without ingress VMTO, Spine 1 and 2 from DC1 and DC2 all advertise the remote IP host route 10.0.0.1 when the origin route is from DC2. The ingress traffic can be directed to either Spine 1 and 2 in DC1. It is then routed to Spine 1 and 2 in DC2 where route 10.0.0.1 was moved. This causes the tromboning effect.
- With ingress VMTO, we can achieve optimal forwarding path by configuring a policy for IP host route (10.0.01) to only be advertised by Spine 1 and 2 from DC2, and not from DC1 when the IP host is moved to DC2.

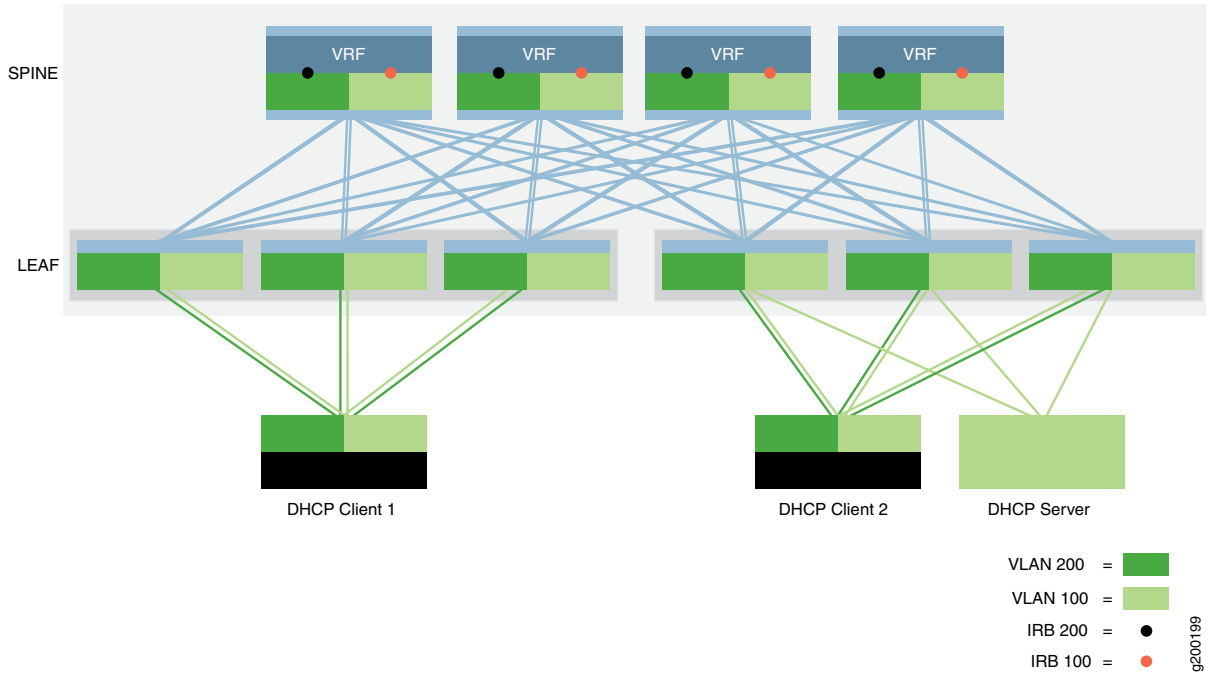
Figure 16: Traffic with and without Ingress VMTO



For information about configuring VMTO, see “Configuring VMTO” on page 263.

DHCP Relay

Figure 17: DHCP Relay in Centrally-Routed Bridging Overlay



The Dynamic Host Configuration Protocol (DHCP) relay building block allows the network to pass DHCP messages between a DHCP client and a DHCP server. The DHCP relay implementation in this building block moves DHCP packets through a centrally-routed bridging overlay where the gateway is located at the spine layer.

The DHCP server and the DHCP clients connect into the network using access interfaces on leaf devices. The DHCP server and clients can communicate with each other over the existing network without further configuration when the DHCP client and server are in the same VLAN. When a DHCP client and server are in different VLANs, DHCP traffic between the client and server is forwarded between the VLANs via the IRB interfaces on spine devices. You must configure the IRB interfaces on the spine devices to support DHCP relay between VLANs.

For information about implementing the DHCP relay, see “DHCP Relay Design and Implementation” on page 264.

Reducing ARP Traffic with ARP Synchronization and Suppression (Proxy ARP)

The goal of ARP synchronization is to synchronize ARP tables across all the VRFs that serve an overlay subnet to reduce the amount of traffic and optimize processing for both network devices and end systems. When an IP gateway for a subnet learns about an ARP binding, it shares it with other gateways so they do not need to discover the same ARP binding independently.

With ARP suppression, when a leaf device receives an ARP request, it checks its own ARP table that is synchronized with the other VTEP devices and responds to the request locally rather than flooding the ARP request.

Proxy ARP and ARP suppression are enabled by default on all QFX Series switches that can act as leaf devices in an edge-routed bridging overlay. For a list of these switches, see Table 1 on page 46.

IRB interfaces on the leaf device deliver ARP requests and NDP requests from both local and remote leaf devices. When a leaf device receives an ARP request or NDP request from another leaf device, the receiving device searches its MAC+IP address bindings database for the requested IP address.

- If the device finds the MAC+IP address binding in its database, it responds to the request.
- If the device does not find the MAC+IP address binding, it floods the ARP request to all Ethernet links in the VLAN and the associated VTEPs.

Because all participating leaf devices add the ARP entries and synchronize their routing and bridging tables, local leaf devices respond directly to requests from locally connected hosts and remove the need for remote devices to respond to these ARP requests.

For information about implementing the ARP synchronization, Proxy ARP, and ARP suppression, see Enabling Proxy ARP and ARP Suppression for the Edge-Routed Bridging Overlay.

Layer 2 Port Security Features on Ethernet-Connected End Systems

IN THIS SECTION

Centrally-routed bridging overlay and edge-routed bridging overlay supports the following security features on Layer 2 Ethernet-connected end systems:

For more information about these features, see *MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment*.

For information about configuring these features, see "Configuring Layer 2 Port Security Features on Ethernet-Connected End Systems" on page 275.

Preventing BUM Traffic Storms With Storm Control

Storm control can prevent excessive traffic from degrading the network. It lessens the impact of BUM traffic storms by monitoring traffic levels on EVPN-VXLAN interfaces, and dropping BUM traffic when a specified traffic level is exceeded.

In an EVPN-VXLAN environment, storm control monitors:

- Layer 2 BUM traffic that originates in a VXLAN and is forwarded to interfaces within the same VXLAN.
- Layer 3 multicast traffic that is received by an IRB interface in a VXLAN and is forwarded to interfaces in another VXLAN.

Using MAC Filtering to Enhance Port Security

MAC filtering enhances port security by limiting the number of MAC addresses that can be learned within a VLAN and therefore limit the traffic in a VXLAN. Limiting the number of MAC addresses protects the switch from flooding the Ethernet switching table. Flooding of the Ethernet switching table occurs when the number of new MAC addresses that are learned causes the table to overflow, and previously learned MAC addresses are flushed from the table. The switch relearns the MAC addresses, which can impact performance and introduce security vulnerabilities.

In this blueprint, MAC filtering limits the number of accepted packets that are sent to ingress-facing access interfaces based on MAC addresses. For more information about how MAC filtering works, see the MAC limiting information in Understanding MAC Limiting and MAC Move Limiting

Analyzing Traffic Using Port Mirroring

With analyzer-based port mirroring, you can analyze traffic down to the packet level in an EVPN-VXLAN environment. You can use this feature to enforce policies related to network usage and file sharing and to identify problem sources by locating abnormal or heavy bandwidth usage by particular stations or applications.

Port mirroring copies packets entering or exiting a port or entering a VLAN and sends the copies to a local interface for local monitoring or to a VLAN for remote monitoring. Use port mirroring to send traffic to

applications that analyze traffic for purposes such as monitoring compliance, enforcing policies, detecting intrusions, monitoring and predicting traffic patterns, correlating events, and so on.

RELATED DOCUMENTATION

- Infrastructure as a Service: EVPN and VXLAN Solution Guide
- Juniper Networks EVPN Implementation for Next-Generation Data Center Architectures

2

CHAPTER

Data Center Fabric Reference Design—Tested Implementation

Data Center Fabric Reference Design Overview and Validated Topology

IN THIS SECTION

This section provides a high-level overview of the Data Center Fabric reference design topology and summarizes the topologies that were tested and validated by the Juniper Networks Test Team.

It includes the following sections:

Reference Design Overview

The Data Center Fabric reference design tested by Juniper Networks is based on an IP Fabric underlay in a Clos topology that uses the following devices:

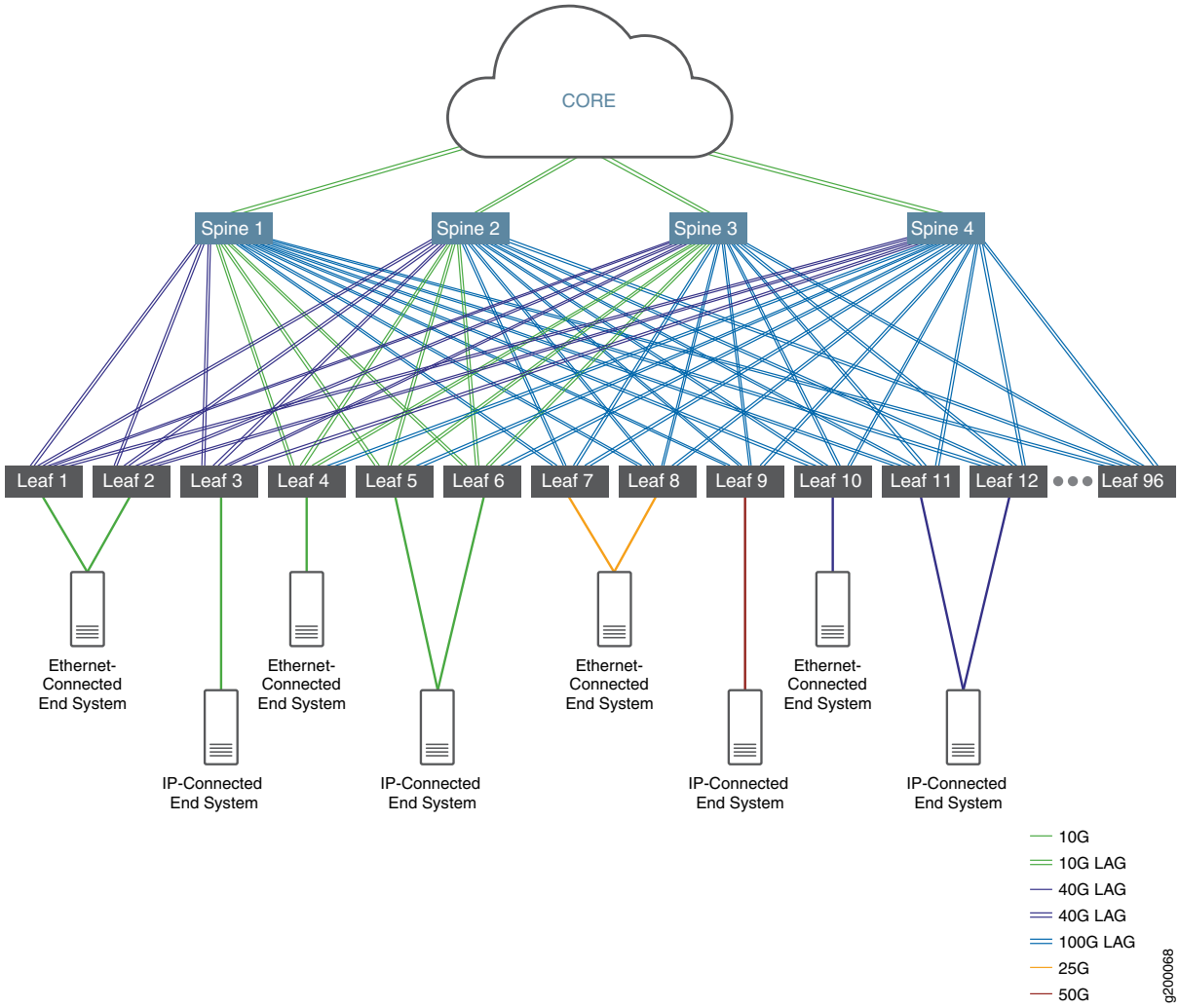
- Spine devices: up to four.
- Leaf devices: the number of leaf devices supported by Juniper Networks varies depending on the Junos OS software release and overlay type (centrally-routed bridging overlay or edge-routed bridging overlay). For more information, see "Data Center Fabric Reference Design Scaling Summary for Junos OS Release 17.3R3-S3" on page 291, "Data Center Fabric Reference Design Scaling Summary for Junos OS Release 18.4R2" on page 288, and "Data Center Fabric Reference Design Scaling Summary for Junos OS Release 19.1R2" on page 283.

  The number of tested leaf nodes reflected throughout this guide is 96, which was number tested in the initial reference design.

Each leaf device is interconnected to each spine device using either an aggregated Ethernet interface that includes two high-speed Ethernet interfaces (10-Gbps, 40-Gbps, or 100-Gbps) as LAG members or a single high-speed Ethernet interface.

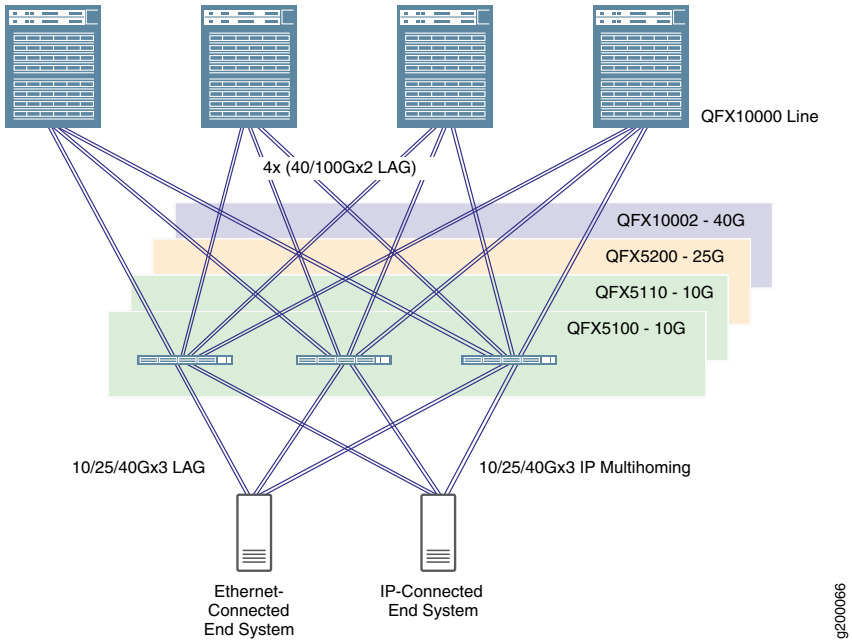
Figure 18 on page 45 provides an illustration of the topology used in this reference design:

Figure 18: Data Center Fabric Reference Design - Topology



End systems such as servers connect to the data center network through leaf device interfaces. Each end system was multihomed to three leaf devices using a 3-member aggregated Ethernet interface as shown in Figure 19 on page 46.

Figure 19: Data Center Fabric Reference Design - Multihoming



The objective for multihoming end systems to 3 different leaf devices is to verify that multihoming of an end system to more than 2 leaf devices is fully supported.

Hardware Summary

Table 1 on page 46 summarizes the hardware and associated software that you can use to create this reference design.

NOTE:

- For this reference design, we support the hardware listed in Table 1 on page 46 only with the associated Junos OS software releases.
- To learn about any existing issues and limitations for a hardware device in this reference design, see the release notes for the Junos OS software release with which the device was tested.

Table 1: Data Center Fabric Reference Design Hardware and Software Summary

Device Roles	Hardware	Junos OS Software Releases[1]
Centrally-Routed Bridging Overlay		

Table 1: Data Center Fabric Reference Design Hardware and Software Summary *(continued)*

Device Roles	Hardware	Junos OS Software Releases[1]
Spine	QFX10002-36Q/72Q QFX10008 QFX10016	17.3R3-S1
	QFX10002-60C QFX5120-32C	19.1R2
Leaf	QFX5100 QFX5110 QFX5200 QFX10002-36Q/72Q	17.3R3-S1
	QFX5120-48Y	18.4R2
	QFX5120-32C	19.1R2
Edge-Routed Bridging Overlay		
Lean spine	QFX10002-36Q/72Q QFX10008 QFX10016 QFX5200	17.3R3-S1
	QFX5110	18.1R3-S3
	QFX10002-60C QFX5120-32C	19.1R2

Table 1: Data Center Fabric Reference Design Hardware and Software Summary *(continued)*

Device Roles	Hardware	Junos OS Software Releases[1]
Leaf	QFX10002-36Q/72Q QFX10008 QFX10016	17.3R3-S1
	QFX10002-60C	19.1R2
	QFX5110	18.1R3-S3
	QFX5120-48Y	18.4R2
	QFX5120-32C	19.1R2
Data Center Interconnect (DCI) (using EVPN Type 5 routes and IPVPN), Service Chaining		
Border spine	QFX10002-36Q/72Q QFX10008 QFX10016	17.3R3-S1
	QFX10002-60C[2]	19.1R2
	QFX5120-48Y[3]	18.4R2-S4
	QFX5120-32C[3]	19.1R3
Border leaf	QFX10002-36Q/72Q QFX10008 QFX10016	17.3R3-S1
	MX204; MX240, MX480, and MX960 with MPC7E; MX10003[2]	18.4R2-S2
	QFX10002-60C[2]	19.1R2
	QFX5120-48Y[3]	18.4R2-S4
	QFX5120-32C[3]	19.1R3

[1]To keep things simple, this column includes the initial Junos OS release train with which we introduce support for the hardware in the reference design. For each initial Junos OS release train, we also support the hardware with later releases in the same release train.

[2]While functioning in this role, this hardware does not support multicast traffic.

[3]While functioning in this role, this hardware does not support DCI or multicast traffic.

This table does not include backbone devices that connect the data center to a WAN cloud. Backbone devices provide physical connectivity between data centers and are required for DCI. See *Data Center Interconnect Design and Implementation*.

Interfaces Summary

IN THIS SECTION

This section summarizes the interface connections between spine and leaf devices that were validated in this reference design.

It contains the following sections:

Interfaces Overview

In the validated reference design, spine and leaf devices are interconnected using either an aggregated Ethernet interface that includes two high-speed Ethernet interfaces or a single high-speed Ethernet interface.

The reference design was validated with the following combinations of spine and leaf device interconnections:

- QFX10002, QFX10008, or QFX10016 switches as spine devices and QFX5100, QFX5110, QFX5200, and QFX10002 switches as leaf devices.

  All 10-Gbps, 40-Gbps, or 100-Gbps interfaces on the supported platforms were used to interconnect a spine and leaf device.

- Combinations of aggregated Ethernet interfaces containing two 10-Gbps, 40-Gbps, or 100-Gbps member interfaces between the supported platforms were validated.
- Channelized 10-Gbps, 40-Gbps, or 100-Gbps interfaces used to interconnect spine and leaf devices as single links or as member links in a 2-member aggregated Ethernet bundle were also validated.

Spine Device Interface Summary

As previously stated, the validated design includes up to 4 spine devices and leaf devices that are interconnected by one or two high-speed Ethernet interfaces.

QFX10008 and QFX10016 switches were used as they can achieve the port density necessary for this reference design. See QFX10008 Hardware Overview or QFX10016 Hardware Overview for information on supported line cards and the number of high-speed Ethernet interfaces supported on these switches.

QFX10002-36Q/72Q, QFX10002-60C, and QFX5120-32C switches, however, do not have the port density to support this reference design at the larger scales but can be deployed as spine devices in smaller scale environments. See QFX10002 Hardware Overview and QFX5120 System Overview for information about the number of high-speed Ethernet interfaces supported on QFX10002-36Q/72Q, QFX10002-60C, and QFX5120-32C switches, respectively.

All channelized spine device interface options are tested and supported in the validated reference design.

Leaf Device Interface Summary

Each leaf device in the reference design connects to the four spine devices and has the port density to support this reference design.

The number and types of high-speed Ethernet interfaces used as uplink interfaces to spine devices vary by leaf device switch model.

To see which high-speed interfaces are available with each leaf device switch model, see the following documents:

- *QFX10002 Switch Description*
- *QFX5200 Switch Description*
- *QFX5100 Device Hardware Overview*
- *QFX5110 Hardware Overview*
- *QFX5120 System Overview*

RELATED DOCUMENTATION

IP Fabric Underlay Network Design and Implementation

IN THIS SECTION

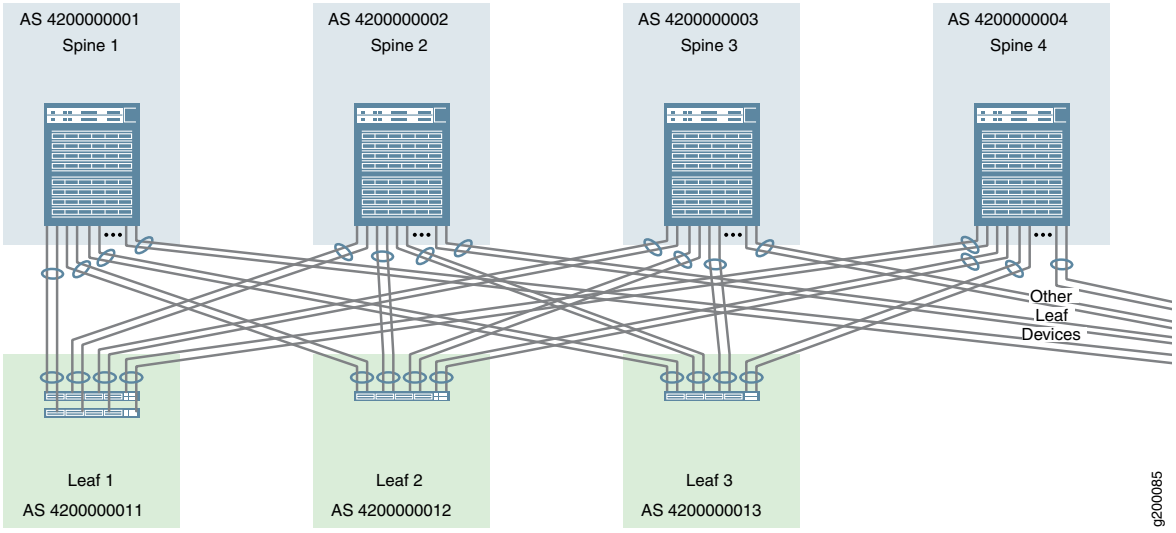
For an overview of the IP Fabric underlay components used in this design, see the IP Fabric Underlay Network section in "Data Center Fabric Blueprint Architecture Components" on page 15.

The IP underlay network building block is arranged in a Clos-based fabric topology. The underlay network uses EBGP as the routing protocol in place of a traditional IGP like OSPF. You can use other routing protocols in the underlay protocol in your data center; the usage of those routing protocols is beyond the scope of this document.

Aggregated Ethernet interfaces with MicroBFD are also used in this building block. MicroBFD improves fault detection in an aggregated Ethernet interface by running BFD on individual links of the aggregated Ethernet interface.

Figure 20 on page 52 provides a high-level illustration of the Clos-based IP fabric underlay network.

Figure 20: IP Fabric Underlay Network Overview



Configuring the Aggregated Ethernet Interfaces Connecting Spine Devices to Leaf Devices

In this design each spine device is interconnected to each leaf device using a single link or a two-member aggregated Ethernet interface. The decision to use a single link or an aggregated Ethernet interface largely depends on the needs of your network; see "Data Center Fabric Reference Design Overview and Validated Topology" on page 44 for more information on interface requirements.

The majority of IP Fabric topologies do not use aggregated Ethernet interfaces to interconnect spine and leaf devices. You can skip this section if you are connecting your spine and leaf devices using single links.

Use the following instructions to configure the interfaces that interconnect spine and leaf devices as aggregated Ethernet interfaces with two member links. An IPv4 address is assigned to each aggregated Ethernet interface. LACP with a fast periodic interval is also enabled.

Figure 21 on page 54 shows the spine device interfaces that are configured in this procedure:

Figure 21: Spine 1 Interfaces

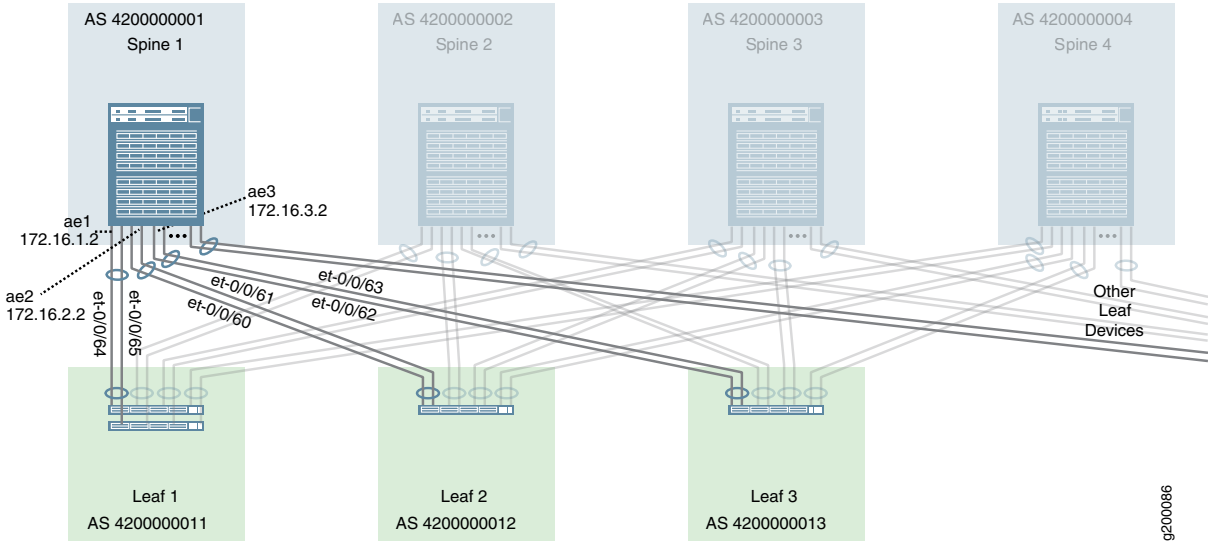
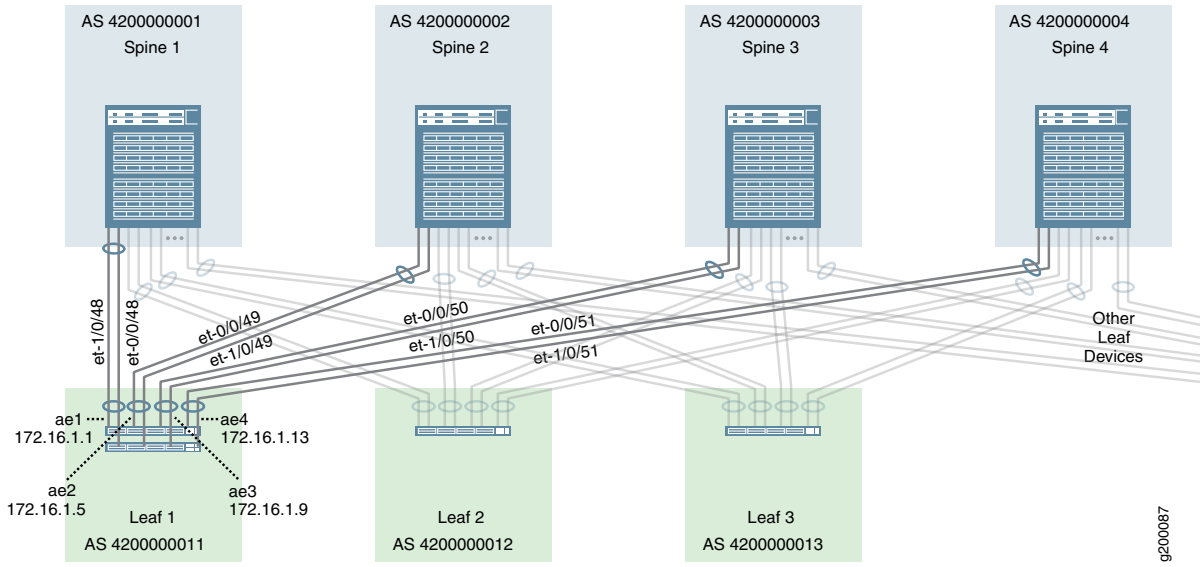


Figure 22 on page 54 shows the leaf device interfaces that are configured in this procedure:

Figure 22: Leaf 1 Interfaces



To configure aggregated Ethernet interfaces with fast LACP:

1. Set the maximum number of aggregated Ethernet interfaces permitted on the device.

   We recommend setting this number to the exact number of aggregated Ethernet interfaces on your device, including aggregated Ethernet interfaces that are not used for spine to leaf device connections.

   In this example, the aggregated Ethernet device count value is set at 10 for a leaf device and 100 for a spine device.

   Leaf Device:

   ```
   set chassis aggregated-devices ethernet device-count 10
   ```

   Spine Device:

   ```
   set chassis aggregated-devices ethernet device-count 100
   ```

2. Create and name the aggregated Ethernet interfaces, and optionally assign a description to each interface.

   This step shows how to create three aggregated Ethernet interfaces on spine 1 and four aggregated ethernet interfaces on leaf 1.

   Repeat his procedure for every aggregated Ethernet interface connecting a spine device to a leaf device.

   Spine 1:

```
set interfaces ae1 description "LEAF1"
set interfaces ae2 description "LEAF2"
set interfaces ae3 description "LEAF3"
```

Leaf 1:

```
set interfaces ae1 description "SPINE1"
set interfaces ae2 description "SPINE2"
set interfaces ae3 description "SPINE3"
set interfaces ae4 description "SPINE4"
```

3. Assign interfaces to each aggregated Ethernet interface on your device.

   Spine 1:

   ```
   set interfaces et-0/0/64 ether-options 802.3ad ae1
   set interfaces et-0/0/65 ether-options 802.3ad ae1
   set interfaces et-0/0/60 ether-options 802.3ad ae2
   set interfaces et-0/0/61 ether-options 802.3ad ae2
   set interfaces et-0/0/62 ether-options 802.3ad ae3
   set interfaces et-0/0/63 ether-options 802.3ad ae3
   ```

   Leaf 1:

   ```
   set interfaces et-0/0/48 ether-options 802.3ad ae1
   set interfaces et-1/0/48 ether-options 802.3ad ae1
   set interfaces et-0/0/49 ether-options 802.3ad ae2
   set interfaces et-1/0/49 ether-options 802.3ad ae2
   set interfaces et-0/0/50 ether-options 802.3ad ae3
   set interfaces et-1/0/50 ether-options 802.3ad ae3
   set interfaces et-0/0/51 ether-options 802.3ad ae4
   set interfaces et-1/0/51 ether-options 802.3ad ae4
   ```

4. Assign an IP address to each aggregated Ethernet interface on the device.

   Spine 1:

```
set interfaces ae1 unit 0 family inet address 172.16.1.2/30
set interfaces ae2 unit 0 family inet address 172.16.2.2/30
set interfaces ae3 unit 0 family inet address 172.16.3.2/30
```

Leaf 1:

```
set interfaces ae1 unit 0 family inet address 172.16.1.1/30
set interfaces ae2 unit 0 family inet address 172.16.1.5/30
set interfaces ae3 unit 0 family inet address 172.16.1.9/30
set interfaces ae4 unit 0 family inet address 172.16.1.13/30
```

5. Enable fast LACP on every aggregated Ethernet interface on the device.

   LACP is enabled using the fast periodic interval, which configures LACP to send a packet every second.

   Spine 1:

```
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast
```

   Leaf 1:

```
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
. . .
. . .
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
```

6. After the configuration is committed, confirm that the aggregated Ethernet interfaces are enabled, that the physical links are up, and that packets are being transmitted if traffic has been sent.

   The output below provides this confirmation information for **ae1** on Spine 1.

```
user@spine-1> show interfaces ae1
Physical interface: ae1, Enabled, Physical link is Up
 (some output removed for brevity)

  Logical interface ae1.0 (Index 549) (SNMP ifIndex 541)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :        609303          0      57831130            0
        Output:       7063505          0    4664278858            0

(some output removed for brevity)
```

7. Confirm the LACP receive state is **Current** and that the transmit state is **Fast** for each link in each aggregated Ethernet interface bundle.

   The output below provides LACP status for interface **ae1**.

```
user@spine-1> show lacp interfaces ae1
Aggregated interface: ae1
...(some output removed for brevity)
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/0                  Current   Fast periodic Collecting distributing

      et-0/0/1                  Current   Fast periodic Collecting distributing
(additional output removed for brevity)
```

8. Repeat this procedure for every device in your topology.

   The guide presumes that spine and leaf devices are interconnected by two-member aggregated Ethernet interfaces in other sections. When you are configuring or monitoring a single link instead of an aggregated Ethernet link, substitute the physical interface name of the single link interface in place of the aggregated Ethernet interface name.

Configuring an IP Address for an Individual Link

This section covers the procedure to add an IP address to a single link interface connecting a spine or leaf device. The process for adding an IP address to an aggregated Ethernet interface is covered in “Configuring the Aggregated Ethernet Interfaces Connecting Spine Devices to Leaf Devices” on page 53.

To add an IP address to a single link interface:

1. Assign an IP address to each interface.

   Spine 1 Example:

   ```
   set interfaces et-0/0/64 unit 0 family inet address 172.16.1.2/30
   set interfaces et-0/0/65 unit 0 family inet address 172.16.2.2/30
   set interfaces et-0/0/66 unit 0 family inet address 172.16.3.2/30
   ```

   Leaf 1 Example:

   ```
   set interfaces et-0/0/48 unit 0 family inet address 172.16.1.1/30
   set interfaces et-0/0/49 unit 0 family inet address 172.16.1.5/30
   set interfaces et-0/0/50 unit 0 family inet address 172.16.1.9/30
   set interfaces et-0/0/51 unit 0 family inet address 172.16.1.13/30
   ```

2. After the interface configuration is committed, confirm that the interfaces are enabled, that the physical links are up, and that packets are being transmitted if traffic has been sent.

   The output below provides this confirmation information for **et-0/0/64** on Spine 1.

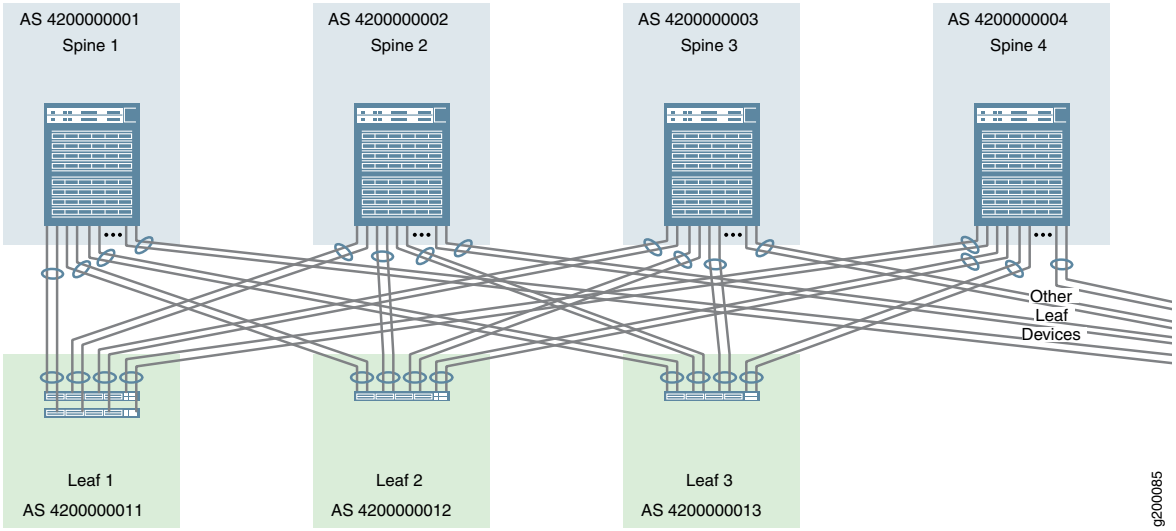
   ```
   user@spine-1> show interfaces et-0/0/64
   Physical interface: et-0/0/64, Enabled, Physical link is Up
    (some output removed for brevity)

     Logical interface ae1.0 (Index 549) (SNMP ifIndex 541)
       Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
       Statistics        Packets        pps         Bytes          bps
       Bundle:
           Input :        609303          0      57831130            0
           Output:       7063505          0    4664278858            0

   (some output removed for brevity)
   ```

Enabling EBGP as the Routing Protocol in the Underlay Network

In this design, EBGP is the routing protocol of the underlay network and each device in the IP fabric is assigned a unique 32-bit autonomous system number (ASN). The underlay routing configuration ensures that all devices in the underlay IP fabric are reliably reachable from one another. Reachability between VTEP across the underlay IP Fabric is also required to support overlay networking with VXLAN.

Figure 23 on page 60 shows the EBGP configuration of the underlay network.

Figure 23: EBGP Underlay Network Overview



To enable EBGP as the routing protocol for the underlay network on a device:

1. Create and name the BGP peer group. EBGP is enabled as part of this step.

   The underlay EBGP group is named **UNDERLAY** in this design.

   Spine or Leaf Device:

   ```
   set protocols bgp group UNDERLAY type external
   ```

2. Configure the ASN for each device in the underlay.

   Recall that in this design, every device is assigned a unique ASN in the underlay network. The ASN for EBGP in the underlay network is configured at the BGP peer group level using the **local-as** statement because the system ASN setting is used for MP-IBGP signaling in the overlay network.

   The examples below show how to configure the ASN for EBGP for Spine 1 and Leaf 1.

   Spine 1:

```
set protocols bgp group UNDERLAY local-as 4200000001
```

Leaf 1:

```
set protocols bgp group UNDERLAY local-as 4200000011
```

3. Configure BGP peers by specifying the ASN of each BGP peer in the underlay network on each spine and leaf device.

   In this design, for a spine device, every leaf device is a BGP peer, and for a leaf device, every spine device is a BGP peer.

   The example below demonstrates how to configure the peer ASN in this design.

   Spine 1:

   ```
   set protocols bgp group UNDERLAY neighbor 172.16.1.1 peer-as 4200000011
   set protocols bgp group UNDERLAY neighbor 172.16.2.1 peer-as 4200000012
   set protocols bgp group UNDERLAY neighbor 172.16.3.1 peer-as 4200000013
   ```

   Leaf 1:

   ```
   set protocols bgp group UNDERLAY neighbor 172.16.1.2 peer-as 4200000001
   set protocols bgp group UNDERLAY neighbor 172.16.1.6 peer-as 4200000002
   set protocols bgp group UNDERLAY neighbor 172.16.1.10 peer-as 4200000003
   set protocols bgp group UNDERLAY neighbor 172.16.1.14 peer-as 4200000004
   ```

4. Set the BGP hold time. The BGP hold time is the length of time, in seconds, that a peer waits for a BGP message—typically a keepalive, update, or notification message—before closing a BGP connection.

   Shorter BGP hold time values guard against BGP sessions staying open for unnecessarily long times in scenarios where problems occur, such as keepalives not being sent. A longer BGP hold time ensures BGP sessions remain active even during problem periods.

   The BGP hold time is configured at 10 seconds for every device in this design.

   Spine or Leaf Device:

   ```
   set protocols bgp group UNDERLAY hold-time 10
   ```

5. Configure EBGP to signal the unicast address family for the underlay BGP peer group.

Spine or Leaf Device:

```
set protocols bgp group UNDERLAY family inet unicast
```

6. Configure an export routing policy that advertises the IP address of the loopback interface to EBGP peering devices.

   This export routing policy is used to make the IP address of the loopback interface reachable from all devices in the IP Fabric. Loopback IP address reachability is required to allow leaf and spine device peering using MP-IBGP in the overlay network. IBGP peering in the overlay network must be established to allow devices in the fabric to share EVPN routes. See "Configuring IBGP for the Overlay" on page 67.

   The route filter IP address in this step—192.168.1.10—is the loopback address of the leaf device.

   Leaf 1:

```
set policy-options policy-statement BGP_LOOPBACK0 term TERM1 from protocol direct
set policy-options policy-statement BGP_LOOPBACK0 term TERM1 from route-filter 192.168.1.10/32
exact
set policy-options policy-statement BGP_LOOPBACK0 term TERM1 then accept
set protocols bgp group UNDERLAY export BGP_LOOPBACK0
```

7. After committing the configuration, enter the **show bgp summary** command on each device to confirm that the BGP state is established and that traffic paths are active.

   Issue the **show bgp summary** command on Spine 1 to verify EBGP status.

```
user@spine-1> show bgp summary
Groups: 1 Peers: 96 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0           834        232          0          0          0          0

Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn
 State|#Active/Received/Accepted/Damped...
172.16.1.1        4200000011      13530      13750       0       0    10:21:37
 Establ inet.0: 204/209/209/0
172.16.2.1        4200000012      13532      13821       0       0    10:21:33
 Establ inet.0: 27/209/209/0
(additional output removed for brevity)
```

8. Repeat this procedure for every spine and leaf device in your topology.

Enabling Load Balancing

ECMP load balancing allows traffic to be sent to the same destination over multiple equal cost paths. Load balancing must be enabled on all spine and leaf devices to ensure that traffic is sent over all available paths provided by the IP Fabric.

Traffic is load balanced per Layer 4 flow on Junos devices. The ECMP algorithm load balances each traffic flow over one of the multiple paths, and all traffic for that flow is transmitted using the selected link.

To enable ECMP-based load balancing on a device:

1. Enable multipath with the multiple AS option in BGP on all devices in the IP Fabric.

   EBGP, by default, selects one best path for each prefix and installs that route in the forwarding table. When BGP multipath is enabled, all equal-cost paths to a given destination are installed into the forwarding table. The **multiple-as** option enables load balancing between EBGP neighbors in different autonomous systems.

   All Spine and Leaf Devices:

   ```
   set protocols bgp group UNDERLAY multipath multiple-as
   ```

2. Create a policy statement that enables per-packet load balancing.

   All Spine and Leaf Devices:

   ```
   set policy-options policy-statement PFE-ECMP then load-balance per-packet
   ```

3. Export the policy statement to the forwarding table.

   All Spine and Leaf Devices:

   ```
   set routing-options forwarding-table export PFE-ECMP
   ```

Configuring Micro Bidirectional Forwarding Detection on Member Links in Aggregated Ethernet Interfaces

BFD is a simple bidirectional fault detection protocol that verifies bidirectional connectivity between directly-connected devices by periodically and rapidly sending a simple hello packet over the link or links that interconnect the devices. BFD can detect and communicate link faults in sub-second time frames to allow the control-plane software to quickly switch to an alternate path.

MicroBFD allows BFD to run on individual member links in an aggregated Ethernet interface.

In this design, microBFD is supported on connections between QFX10002-36Q/72Q, QFX10008, and QFX10016 switches.

To enable microBFD:

1. Set the minimum interval for BFD.

   The minimum interval is the amount of time, in milliseconds, that increments before BFD sends a hello packet to a directly-connected neighbor. The minimum interval is also the amount of time that BFD waits to receive a response to the hello packet before directing traffic destined to that link to other member links in the aggregated Ethernet interface.

   If the minimum interval is set at 100, for instance, BFD sends a hello packet every 100 milliseconds and starts directing traffic to other member links in an aggregated Ethernet interface if the hello packet does not receive a response in 100 milliseconds

   Set the minimum interval to a lower value if you want to ensure traffic is quickly redirected after a link failure. Set the minimum interval to a higher value if you want to minimize the chance that an active link is not accidentally misidentified as a down link by BFD.

   The minimum interval is set to 100 milliseconds on all member links in this design.

   The output below shows how to configure the minimum interval to 100 milliseconds on aggregated Ethernet interface 1.

   Spine and Leaf Device:

   ```
   set interfaces ae1 aggregated-ether-options bfd-liveness-detection minimum-interval 100
   ```

2. Configure the IP address used by MicroBFD over the aggregated Ethernet bundle. For bridged AE interfaces, this address should be the IP address used by MicroBFD over the aggregated Ethernet bundle.

   The output below show how to assign the loopback IP address to ae1 on the leaf device.

   Spine and Leaf Device:

```
set interfaces ae1 aggregated-ether-options bfd-liveness-detection local-address 192.168.1.10
```

3. Specify the loopback IP address of the neighbor device on the other end of the aggregated Ethernet interface.

   The output below is from aggregated Ethernet interface 1 on leaf device 1, which is used to connect to spine device 1. Spine device 1 uses 192.168.0.1 as it's loopback IP address.

   Spine and Leaf Device:

   ```
   set interfaces ae1 aggregated-ether-options bfd-liveness-detection neighbor 192.168.0.1
   ```

4. Specify the minimum number of links that have to remain up before all links in the aggregated Ethernet interface stop sending and receiving traffic.

   Setting the minimum links to 1 ensures the aggregated Ethernet bundle always transmits traffic unless all links in the aggregated Ethernet interface are unable to forward or receive traffic. Consider setting the minimum links value to a higher number if you feel performance could be significantly degraded in your network if a number of member links in your aggregated Ethernet interface stopped forwarding traffic.

   All devices in the design set the minimum links value to 1.

   Spine and Leaf Device:

   ```
   set interfaces ae1 aggregated-ether-options minimum-links 1
   ```

   Repeat the above steps for each aggregated Ethernet interface in your topology to enable microBFD.

5. To verify BFD sessions after enabling microBFD, enter the **show bfd session** command:

   Leaf 10 Example:

   ```
   user@leaf10> show bfd session
                                                  Detect   Transmit
   Address                  State     Interface      Time     Interval  Multiplier
   192.168.0.1               Up        et-0/0/11      0.300     0.100        3
   192.168.0.1               Up        et-0/0/29      0.300     0.100        3
   ```

IP Fabric Underlay Network — Release History

Table 2 on page 66 provides a history of all of the features in this section and their support within this reference design.

Table 2: IP Fabric Underlay Network Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train also support all features documented in this section except the following: • MicroBFD, which is supported on QFX10002-36Q/72Q, QFX10008, and QFX10016 switches only.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section except MicroBFD.
18.1R3-S3	QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section except MicroBFD.
17.3R3-S1	All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section. The following is an exception: • MicroBFD, which is supported on QFX10002-36Q/72Q, QFX10008, and QFX10016 switches only.

RELATED DOCUMENTATION

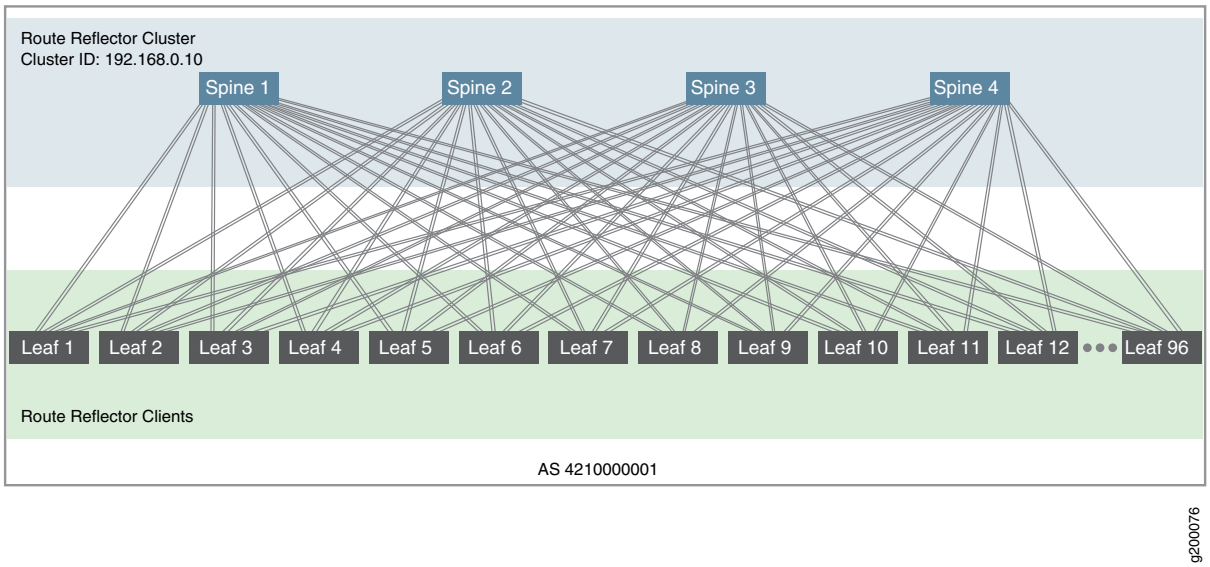
Software as a Service

Understanding Independent Micro BFD Sessions for LAG

Configuring IBGP for the Overlay

For a control-plane driven overlay, there must be a signalling path between all VTEP devices within an autonomous system using IBGP with Multiprotocol BGP (MP-IBGP). In this reference design, all overlay types use IBGP, the spine devices act as a route reflector cluster, and the leaf devices are route reflector clients, as shown in Figure 24 on page 67.

Figure 24: IBGP Route Reflector Cluster



To configure IBGP for the overlay, perform the following:

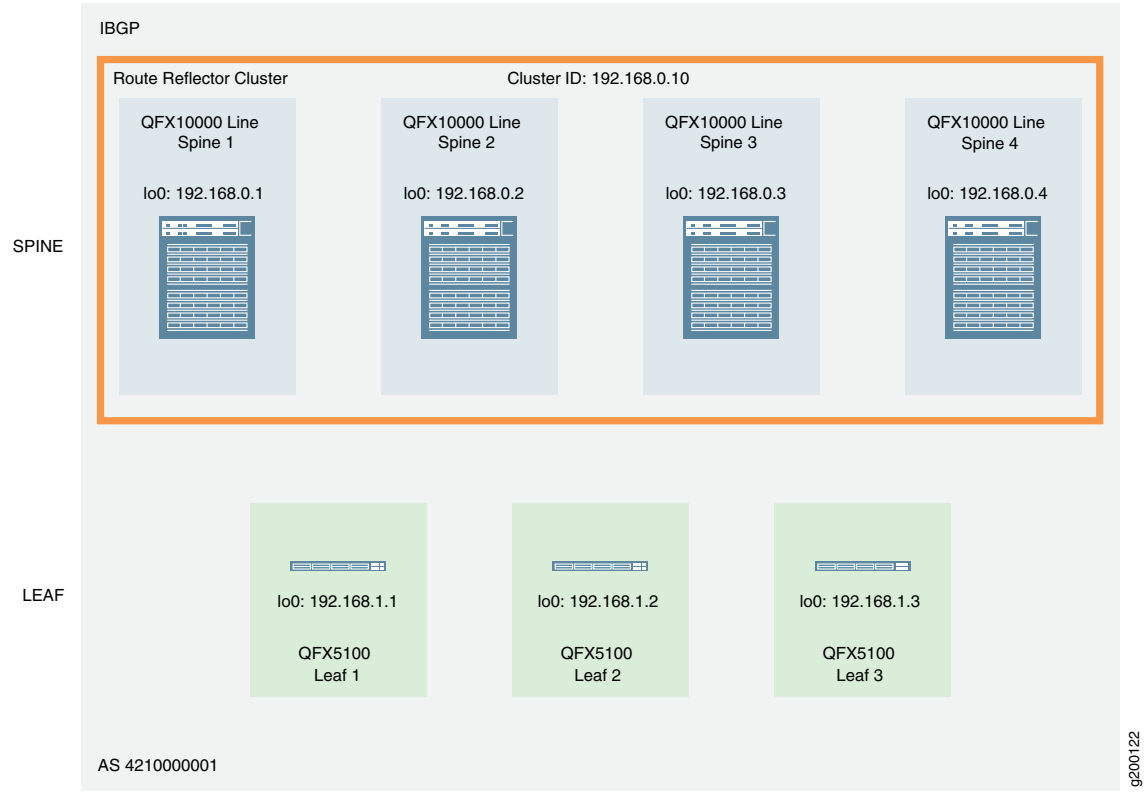
1. Configure an AS number for overlay IBGP. All leaf and spine devices participating in the overlay use the same AS number. In this example, the AS number is private AS 4210000001.

   Spine and Leaf Devices:

   ```
   set routing-options autonomous-system 4210000001
   ```

2. Configure IBGP using EVPN signaling on each spine device to peer with every leaf device (Leaf 1 through Leaf 96). Also, form the route reflector cluster (cluster ID 192.168.0.10) and configure equal cost multipath (ECMP) for BGP. The configuration included here belongs to Spine 1, as shown in Figure 25 on page 68.

Figure 25: IBGP – Spine Device



TIP: By default, BGP selects only one best path when there are multiple, equal-cost BGP paths to a destination. When you enable BGP multipath by including the **multipath** statement at the **[edit protocols bgp group *group-name*]** hierarchy level, the device installs all of the equal-cost BGP paths into the forwarding table. This feature helps load balance the traffic across multiple paths.

Spine 1:

```
set protocols bgp group OVERLAY type internal
set protocols bgp group OVERLAY local-address 192.168.0.1
set protocols bgp group OVERLAY family evpn signaling
set protocols bgp group OVERLAY cluster 192.168.0.10
set protocols bgp group OVERLAY multipath
set protocols bgp group OVERLAY neighbor 192.168.1.1
...
set protocols bgp group OVERLAY neighbor 192.168.1.96
```

3. Configure IBGP on the spine devices to peer with all the other spine devices acting as route reflectors. This step completes the full mesh peering topology required to form a route reflector cluster.

   Spine 1:

   ```
   set protocols bgp group OVERLAY_RR_MESH type internal
   set protocols bgp group OVERLAY_RR_MESH local-address 192.168.0.1
   set protocols bgp group OVERLAY_RR_MESH family evpn signaling
   set protocols bgp group OVERLAY_RR_MESH neighbor 192.168.0.2
   set protocols bgp group OVERLAY_RR_MESH neighbor 192.168.0.3
   set protocols bgp group OVERLAY_RR_MESH neighbor 192.168.0.4
   ```

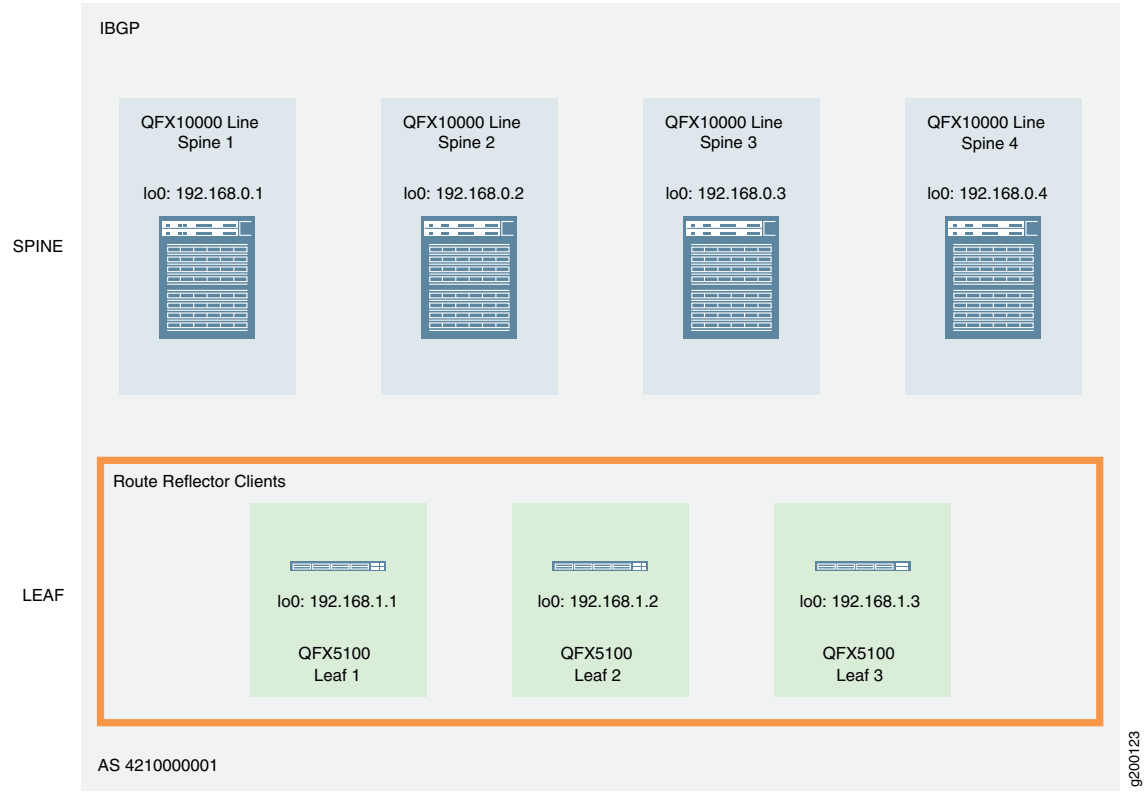
4. Configure BFD on all BGP groups on the spine devices to enable rapid detection of failures and reconvergence.

   Spine 1:

   ```
   set protocols bgp group OVERLAY bfd-liveness-detection minimum-interval 350
   set protocols bgp group OVERLAY bfd-liveness-detection multiplier 3
   set protocols bgp group OVERLAY bfd-liveness-detection session-mode automatic
   set protocols bgp group OVERLAY_RR_MESH bfd-liveness-detection minimum-interval 350
   set protocols bgp group OVERLAY_RR_MESH bfd-liveness-detection multiplier 3
   set protocols bgp group OVERLAY_RR_MESH bfd-liveness-detection session-mode automatic
   ```

5. Configure IBGP with EVPN signaling from each leaf device (route reflector client) to each spine device (route reflector cluster). The configuration included here belongs to Leaf 1, as shown in Figure 26 on page 70.

Figure 26: IBGP – Leaf Device



Leaf 1:

```
set protocols bgp group OVERLAY type internal
set protocols bgp group OVERLAY local-address 192.168.1.1
set protocols bgp group OVERLAY family evpn signaling
set protocols bgp group OVERLAY neighbor 192.168.0.1
set protocols bgp group OVERLAY neighbor 192.168.0.2
set protocols bgp group OVERLAY neighbor 192.168.0.3
set protocols bgp group OVERLAY neighbor 192.168.0.4
```

6. Configure BFD on the leaf devices to enable rapid detection of failures and reconvergence.

   Leaf 1:

```
set protocols bgp group OVERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group OVERLAY bfd-liveness-detection multiplier 3
set protocols bgp group OVERLAY bfd-liveness-detection session-mode automatic
```

7. Verify that IBGP is functional on the spine devices.

```
user@spine-1> show bgp summary
```

```
Groups: 5 Peers: 221 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                    9711        182          0          0          0
0
inet6.0
                       0          0          0          0          0
0
bgp.evpn.0
                   31520      31520          0          0          0
0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn
State|#Active/Received/Accepted/Damped...
192.168.0.2       421000001      28724      31106       0       0    22:40:41
Establ
  bgp.evpn.0: 8227/8227/8227/0
 default-switch.evpn.0: 54/54/54/0...

192.168.1.96      421000001       4831      73047       0       0    22:43:41
Establ
  bgp.evpn.0: 1549/1549/1549/0
  default-switch.evpn.0: 11/11/11/0
  __default_evpn__.evpn.0: 1471/1471/1471/0
---(more)---
```

8. Verify that BFD is operational on the spine devices.

```
user@spine-1> show bfd session
```

```
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
192.168.0.2              Up                       1.050     0.350        3
192.168.0.3              Up                       1.050     0.350        3
192.168.0.4              Up                       1.050     0.350        3
192.168.1.1              Up                       1.050     0.350        3
...
192.168.1.96             Up                       1.050     0.350        3
```

9. Verify that IBGP is operational on the leaf devices.

```
user@leaf-1> show bgp summary
```

```
Groups: 2 Peers: 8 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                     834        233          0          0          0
0
bgp.evpn.0
                    3193        833          0          0          0
0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn
State|#Active/Received/Accepted/Damped...
## IBGP Overlay
 192.168.0.1      4210000001       9371        596       0       2     4:17:03
 Establ
  bgp.evpn.0: 706/829/829/0
  default-switch.evpn.0: 701/824/824/0
  __default_evpn__.evpn.0: 5/5/5/0
192.168.0.2      4210000001      10175        579       0       2     4:16:35
Establ
  bgp.evpn.0: 43/834/834/0
  default-switch.evpn.0: 43/829/829/0
  __default_evpn__.evpn.0: 0/5/5/0
192.168.0.3      4210000001      10463        621       0       2     4:34:55
Establ
  bgp.evpn.0: 43/834/834/0
  default-switch.evpn.0: 43/829/829/0
  __default_evpn__.evpn.0: 0/5/5/0
192.168.0.4      4210000001       8250        463       0       1     3:12:47
Establ
  bgp.evpn.0: 41/696/696/0
  default-switch.evpn.0: 41/691/691/0
  __default_evpn__.evpn.0: 0/5/5/0
```

10. Verify that BFD is operational on the leaf devices.

```
user@leaf-10> show bfd session
```

```
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
192.168.0.1              Up                       1.050     0.350        3
192.168.0.2              Up                       1.050     0.350        3
192.168.0.3              Up                       1.050     0.350        3
192.168.0.4              Up                       1.050     0.350        3
```

RELATED DOCUMENTATION

BGP User Guide

Understanding BGP Route Reflectors

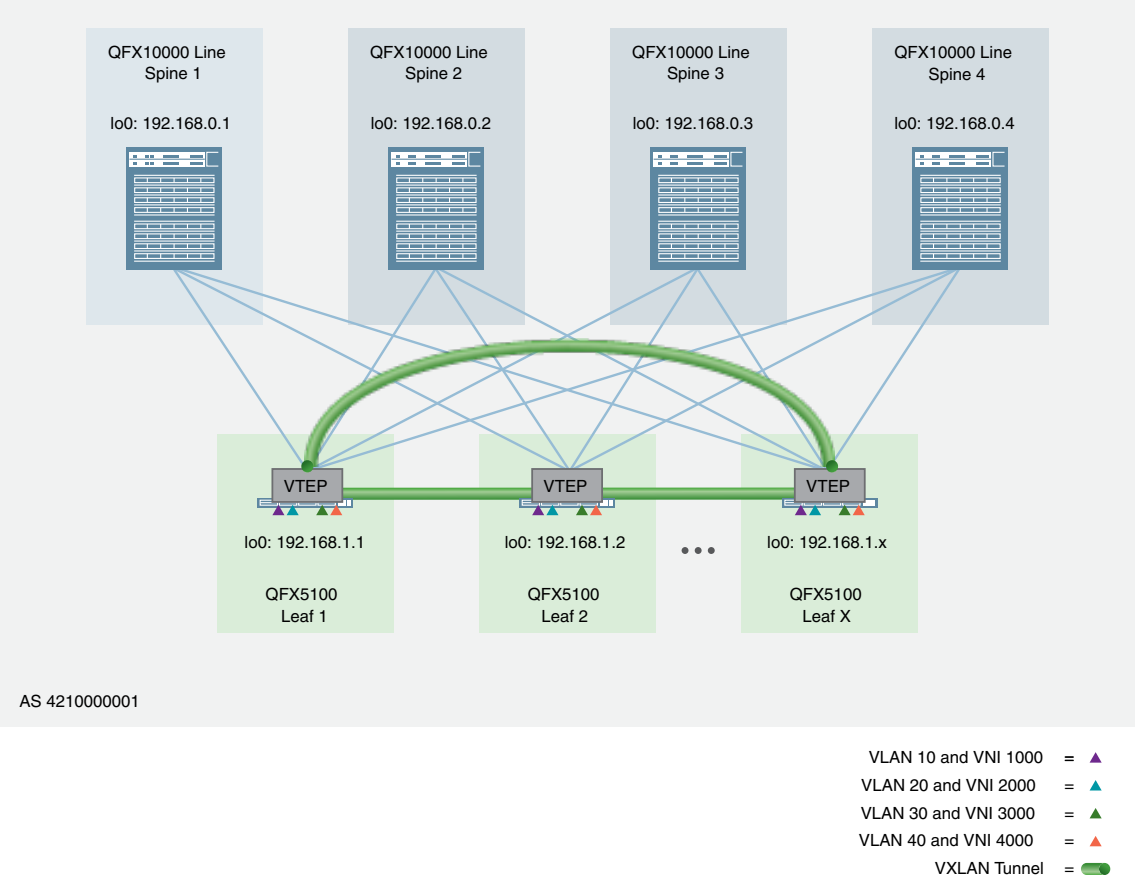
Bridged Overlay Design and Implementation

IN THIS SECTION

A *bridged overlay* provides Ethernet bridging between leaf devices in an EVPN network, as shown in Figure 27 on page 74. This overlay type simply extends VLANs between the leaf devices across VXLAN tunnels. Bridged overlays provide an entry level overlay style for data center networks that require Ethernet connectivity but do not need routing services between the VLANs.

In this example, loopback interfaces on the leaf devices act as VXLAN tunnel endpoints (VTEPs). The tunnels enable the leaf devices to send VLAN traffic to other leaf devices and Ethernet-connected end systems in the data center. The spine devices only provide basic EBGP underlay and IBGP overlay connectivity for these leaf-to-leaf VXLAN tunnels.

Figure 27: Bridged Overlay



NOTE: If inter-VLAN routing is required for a bridged overlay, you can use an MX Series router or SRX Series security device that is external to the EVPN/VXLAN fabric. Otherwise, you can select one of the other overlay types that incorporate routing (such as an "edge-routed bridging overlay" on page 158, a "centrally-routed bridging overlay" on page 95, or a "routed overlay" on page 175) discussed in this Cloud Data Center Architecture Guide.

The following sections provide the detailed steps of how to configure a bridged overlay:

Configuring a Bridged Overlay

IN THIS SECTION

Bridged overlays are supported on all platforms included in this reference design. To configure a bridged overlay, you configure VNIs, VLANs, and VTEPs on the leaf devices, and BGP on the spine devices.

When you implement this style of overlay on a spine device, the focus is on providing overlay transport services between the leaf devices. Consequently, you configure an IP fabric underlay and an IBGP overlay. There are no VTEPs or IRB interfaces needed, because the spine device does not provide any routing functionality or EVPN/VXLAN capabilities in a bridged overlay.

When you implement this style of overlay on a leaf device, you enable EVPN with VXLAN encapsulation to connect to other leaf devices, configure VTEPs, establish route targets and route distinguishers, configure Ethernet Segment Identifier (ESI) settings, and map VLANs to VNIs. Again, you do not include IRB interfaces or routing on the leaf devices for this overlay method.

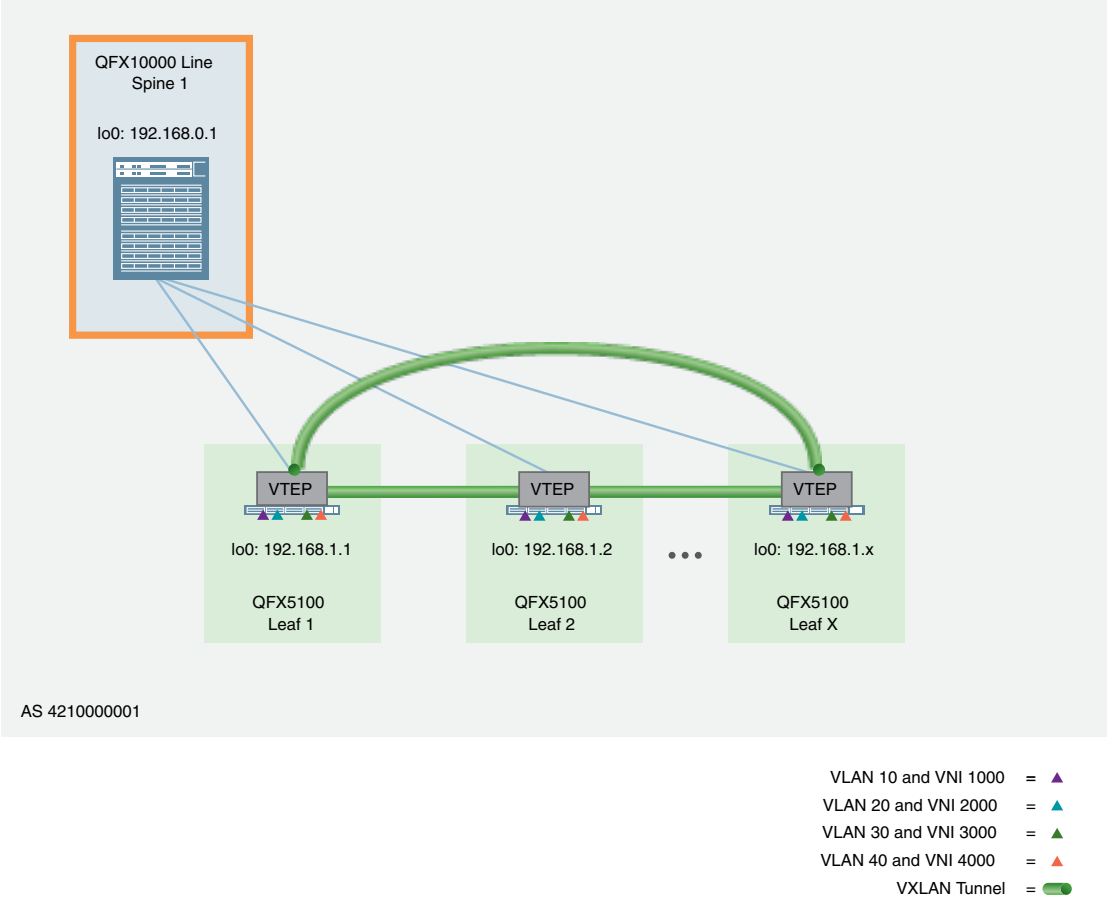
The following sections provide the detailed steps of how to configure and verify the bridged overlay:

Configuring a Bridged Overlay on the Spine Device

To configure a bridged overlay on a spine device, perform the following:

> **NOTE:** The following example shows the configuration for Spine 1, as shown in Figure 28 on page 76.

Figure 28: Bridged Overlay – Spine Device



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on a spine device, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine devices, see "Configuring IBGP for the Overlay" on page 67.

Verifying a Bridged Overlay on the Spine Device

Issue the following commands to verify that the overlay is working properly on your spine devices:

1. Verify that the spine device has reachability to the leaf devices. This output shows the possible routes to Leaf 1.

   `user@spine-1>` **show route 192.168.1.1**

```
inet.0: 446 destinations, 19761 routes (446 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.1.1/32     *[BGP/170] 00:06:29, localpref 100
                      AS path: 4200000010 I, validation-state: unverified
                    > to 172.16.1.1 via ae1.0

                     ...

                     [BGP/170] 00:06:18, localpref 100
                      AS path: 4200000106 4200000004 4200000010 I,
validation-state: unverified
                    > to 172.16.96.1 via ae96.0
```

2. Verify that IBGP is functional on the spine devices acting as a route reflector cluster. You should see peer relationships with all spine device loopback interfaces (192.168.0.1 through 192.168.0.4) and all leaf device loopback interfaces (192.168.1.1 through 192.168.1.96).

   `user@spine-1>` **show bgp summary**

```
Groups: 5 Peers: 221 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                    9711        182          0          0          0
0
...
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn
State|#Active/Received/Accepted/Damped...
192.168.0.2       421000001      28724      31106       0       0    22:40:41
Establ
192.168.0.3       421000001      27424      32106       0       0    22:43:41
```

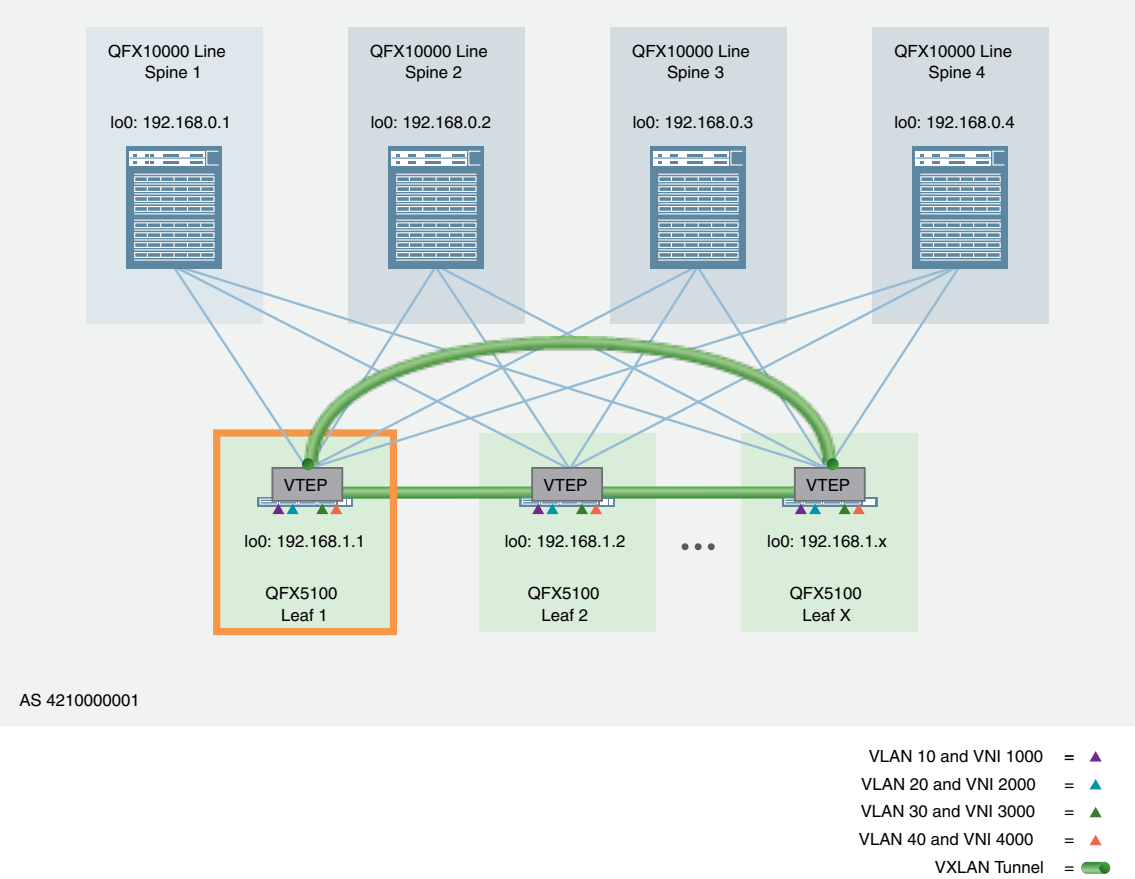
```
Establ
192.168.0.4       421000001      29457      30494       0       0    22:39:04
Establ
192.168.1.1       421000001       3814      75108       0       0    22:43:54
Establ
...
192.168.1.96      421000001       4831      73047       0       0    22:43:41
Establ
```

Configuring a Bridged Overlay on the Leaf Device

To configure a bridged overlay on a leaf device, perform the following:

> **NOTE:**
> - The following example shows the configuration for Leaf 1, as shown in Figure 29 on page 79.

Figure 29: Bridged Overlay – Leaf Device



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on a leaf device, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your leaf device, see "Configuring IBGP for the Overlay" on page 67.

3. Configure the EVPN protocol with VXLAN encapsulation, and specify the VTEP source interface (in this case, the loopback interface of the leaf device).

   Leaf 1:

   ```
   set protocols evpn encapsulation vxlan
   set protocols evpn extended-vni-list all
   set switch-options vtep-source-interface lo0.0
   ```

4. Define an EVPN route target and route distinguisher, and use the **auto** option to derive route targets automatically. Setting these parameters specifies how the routes are imported and exported. The import and export of routes from a bridging table is the basis for dynamic overlays. In this case, members of the global BGP community with a route target of target:64512:1111 participate in the exchange of EVPN/VXLAN information.

   Leaf 1:

   ```
   set switch-options route-distinguisher 192.168.1.1:1
   set switch-options vrf-target target:64512:1111
   set switch-options vrf-target auto
   ```

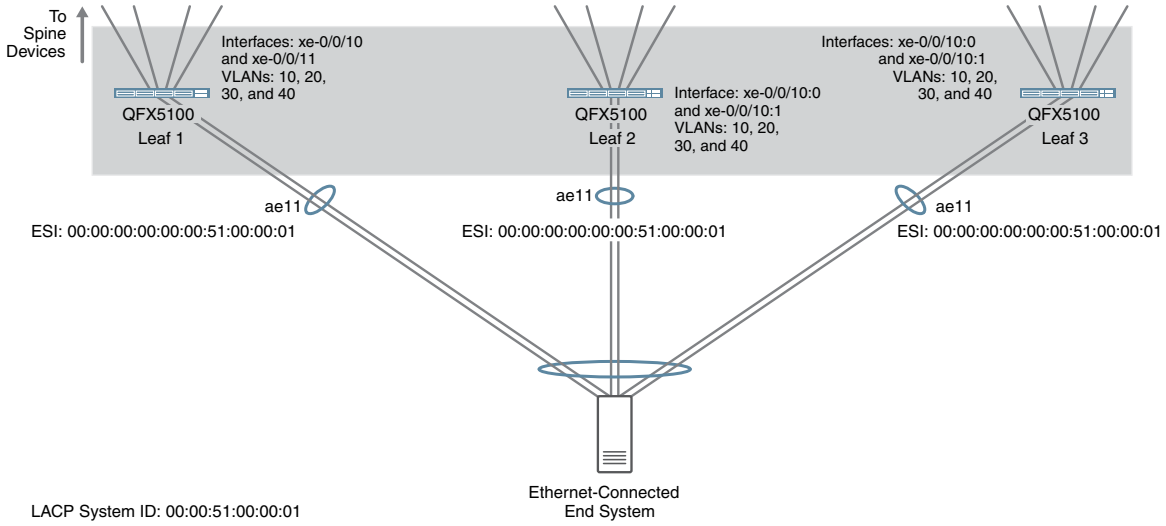
   NOTE: A specific route target processes EVPN Type 1 routes, while an automatic route target processes Type 2 routes. This reference design requires both route targets.

5. Configure ESI settings. Because the end systems in this reference design are multihomed to three leaf devices per device type cluster (such as QFX5100), you must configure the same ESI identifier and LACP system identifier on all three leaf devices for each unique end system. Unlike other topologies where you would configure a different LACP system identifier per leaf device and have VXLAN select a single designated forwarder, use the same LACP system identifier to allow the 3 leaf devices to appear as a single LAG to a multihomed end system. In addition, use the same aggregated Ethernet interface number for all ports included in the ESI.

   The configuration for Leaf 1 is shown below, but you must replicate this configuration on both Leaf 2 and Leaf 3 per the topology shown in Figure 30 on page 81.

   TIP: When you create an ESI number, always set the high order octet to 00 to indicate the ESI is manually created. The other 9 octets can be any hexadecimal value from 00 to FF.

Figure 30: ESI Topology for Leaf 1, Leaf 2, and Leaf 3



Leaf 1:

```
set interfaces ae11 esi 00:00:00:00:00:00:51:00:00:01
set interfaces ae11 esi all-active
set interfaces ae11 aggregated-ether-options lacp active
set interfaces ae11 aggregated-ether-options lacp system-id 00:00:51:00:00:01
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members [ 10 20 30 40 ]
set interfaces xe-0/0/10 ether-options 802.3ad ae11
set interfaces xe-0/0/11 ether-options 802.3ad ae11
```

6. Configure VLANs and map them to VNIs. This step enables the VLANs to participate in VNIs across the EVPN/VXLAN domain.

   Leaf 1:

```
set vlans VNI_1000 vlan-id 10
set vlans VNI_1000 vxlan vni 1000
set vlans VNI_2000 vlan-id 20
set vlans VNI_2000 vxlan vni 2000
set vlans VNI_3000 vlan-id 30
set vlans VNI_3000 vxlan vni 3000
set vlans VNI_4000 vlan-id 40
set vlans VNI_4000 vxlan vni 4000
```

Verifying the Bridged Overlay on the Leaf Device

Issue the following commands to verify that the overlay is working properly on your leaf devices:

1. Verify the interfaces are operational. Interfaces xe-0/0/10 and xe-0/0/11 are dual homed to the Ethernet-connected end system through interface ae11, while interfaces et-0/0/48 through et-0/0/51 are uplinks to the four spine devices.

user@leaf-1> **show interfaces terse | match ae.***

```
xe-0/0/10.0             up    up   aenet    --> ae11.0
et-0/0/48.0             up    up   aenet    --> ae1.0
et-0/0/49.0             up    up   aenet    --> ae2.0
et-0/0/50.0             up    up   aenet    --> ae3.0
et-0/0/51.0             up    up   aenet    --> ae4.0
xe-0/0/11.0             up    up   aenet    --> ae11.0
ae1                     up    up  ## To Spine 1
ae1.0                   up    up   inet     172.16.1.1/30
ae2                     up    up  ## To Spine 2
ae2.0                   up    up   inet     172.16.1.5/30
ae3                     up    up  ## To Spine 3
ae3.0                   up    up   inet     172.16.1.9/30
ae4                     up    up  ## To Spine 4
ae4.0                   up    up   inet     172.16.1.13/30
ae11                    up    up  ## To End System
ae11.0                  up    up   eth-switch
```

user@leaf-1> **show lacp interfaces**

```
Aggregated interface: ae1
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity

      et-0/0/48      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active

      et-0/0/48    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active

    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/48                Current   Fast periodic Collecting distributing


...

Aggregated interface: ae11
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity

      xe-0/0/10      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active

      xe-0/0/10    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
```

```
      xe-0/0/11      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active

      xe-0/0/11    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active

    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/10               Current   Fast periodic Collecting distributing

      xe-0/0/11               Current   Fast periodic Collecting distributing
```

user@leaf-1> **show ethernet-switching interface ae11**

```
Routing Instance Name : default-switch
Logical Interface flags (DL - disable learning, AD - packet action drop,
                         LH - MAC limit hit, DN - interface down,
                       MMAS - Mac-move action shutdown,  AS - Autostate-exclude
 enabled,
                         SCTL - shutdown by Storm-control, MI - MAC+IP limit
hit)

Logical         Vlan                  TAG   MAC    MAC+IP STP         Logical
          Tagging
interface       members                     limit  limit  state       interface
 flags
ae11.0                                       65535  0
          tagged
                VNI_1000              10    65535  0      Forwarding
          tagged
                VNI_2000              20    65535  0      Forwarding
          tagged
                VNI_3000              30    65535  0      Forwarding
          tagged
                VNI_4000              40    65535  0      Forwarding
          tagged
```

2. Verify on Leaf 1 and Leaf 3 that the Ethernet switching table has installed both the local MAC addresses and the remote MAC addresses learned through the overlay.

> **NOTE:** To identify end systems learned remotely from the EVPN overlay, look for the MAC address, ESI logical interface, and ESI number. For example, Leaf 1 learns about an end system with the MAC address of **02:0c:10:03:02:02** through **esi.1885**. This end system has an ESI number of **00:00:00:00:00:00:51:10:00:01**. Consequently, this matches the ESI number configured for Leaf 4, 5, and 6 (QFX5110 switches), so we know that this end system is multihomed to these three leaf devices.

```
user@leaf-1> show ethernet-switching table vlan-id 30

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent
 static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC,
 O - ovsdb MAC)


Ethernet switching table : 10 entries, 10 learned
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical              Active

   name                address             flags    interface            source
## Learned locally
   VNI_3000            02:0c:10:03:02:01   DL       ae11.0
## Learned from the QFX5110 switches - Leaf 4 to 6
   VNI_3000            02:0c:10:03:02:02   DR       esi.1885
00:00:00:00:00:00:51:10:00:01
## Learned from the QFX5200 switches - Leaf 7 to 9
   VNI_3000            02:0c:10:03:02:03   DR       esi.1887
00:00:00:00:00:00:52:00:00:01
## Learned from the QFX10002 switches - Leaf 10 to 12
   VNI_3000            02:0c:10:03:02:04   DR       esi.1892
00:00:00:00:00:01:00:00:00:01
## End System MAC address, connected locally to the leaf
device
02:0c:10:03:02:01
## MAC address learned over the overlay, these end systems
are also multihomed
02:0c:10:03:02:02,03,04
```

3. Verify the remote EVPN routes coming from VNI 1000 and MAC address 02:0c:10:01:02:02. In this case, they are coming from Leaf 4 (192.168.1.4) by way of Spine 1 (192.168.0.1).

NOTE: The format of the EVPN routes is *EVPN-route-type:route-distinguisher:vni:mac-address*.

`user@leaf-1>` **show route table bgp.evpn.0 evpn-ethernet-tag-id 1000 evpn-mac-address 02:0c:10:01:02:02**

```
bgp.evpn.0: 910 destinations, 3497 routes (904 active, 0 holddown, 24 hidden)
+ = Active Route, - = Last Active, * = Both

2:192.168.1.4:1::1000::02:0c:10:01:02:02/304 MAC/IP
                   *[BGP/170] 00:11:37, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                     > to 172.16.1.10 via ae3.0
                     [BGP/170] 00:11:37, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                     > to 172.16.1.10 via ae3.0
                     [BGP/170] 00:11:37, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                     > to 172.16.1.10 via ae3.0
                     [BGP/170] 00:11:37, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                     > to 172.16.1.10 via ae3.0
```

`user@leaf-1>` **show route table bgp.evpn.0 evpn-ethernet-tag-id 1000 evpn-mac-address 02:0c:10:01:02:02 detail**

```
bgp.evpn.0: 925 destinations, 3557 routes (919 active, 0 holddown, 24 hidden)
2:192.168.1.4:1::1000::02:0c:10:01:02:02/304 MAC/IP (4 entries, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.1.4:1
                Next hop type: Indirect, Next hop index: 0
                Address: 0xb3a2170
                Next-hop reference count: 160
                Source: 192.168.0.1
                Protocol next hop: 192.168.1.4
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <Active Int Ext>
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 13:42      Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.0.1
```

```
                AS path: I (Originator)
                Cluster list:  192.168.0.10
                Originator ID: 192.168.1.4
                Communities: target:62273:268445456 encapsulation:vxlan(0x8)
                Import Accepted
                Route Label: 1000
                ESI: 00:00:00:00:00:00:51:10:00:01
                Localpref: 100
                Router ID: 192.168.0.1
                Secondary Tables: default-switch.evpn.0
...
## This output has been abbreviated.
```

4. Verify the source and destination address of each VTEP interface and view their status.

> **NOTE:** There are 96 leaf devices, so there are 96 VTEP interfaces in this reference design - one VTEP interface per leaf device.

`user@leaf-1>` **show ethernet-switching vxlan-tunnel-end-point source**

```
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx
<default>                 0   192.168.1.1      lo0.0    0
    L2-RTT                   Bridge Domain              VNID     MC-Group-IP
    default-switch           VNI_1000+10                1000     0.0.0.0

    default-switch           VNI_2000+20                2000     0.0.0.0

    default-switch           VNI_3000+30                3000     0.0.0.0

    default-switch           VNI_4000+40                4000     0.0.0.0
```

`user@leaf-1>` **show interfaces terse vtep**

```
Interface               Admin Link Proto    Local                 Remote
vtep                    up    up
vtep.32768              up    up
vtep.32769              up    up   eth-switch
vtep.32770              up    up   eth-switch
vtep.32771              up    up   eth-switch
vtep.32772              up    up   eth-switch
```

```
...
vtep.32869              up    up   eth-switch
```

user@leaf-1> **show interfaces vtep**

```
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 646, SNMP ifIndex: 503
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited,
 Speed: Unlimited
  Device flags   : Present Running
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 554) (SNMP ifIndex 648)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
   VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 192.168.1.1, L2 Routing
 Instance: default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 0


...  Logical interface vtep.32814 (Index 613) (SNMP ifIndex 903)
    Flags: Up SNMP-Traps Encapsulation: ENET2
   VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.96, L2 Routing
 Instance: default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 6364
    Protocol eth-switch, MTU: Unlimited
      Flags: Trunk-Mode
```

5. Verify that each VNI maps to the associated VXLAN tunnel.

user@leaf-1> **show ethernet-switching vxlan-tunnel-end-point remote**

```
                0   192.168.1.1      lo0.0    0
 RVTEP-IP         IFL-Idx   NH-Id
 192.168.1.2      586       1782
    VNID          MC-Group-IP
    2000          0.0.0.0
    4000          0.0.0.0
    1000          0.0.0.0
```

```
    3000          0.0.0.0

...

RVTEP-IP         IFL-Idx   NH-Id
192.168.1.96     613       1820
    VNID          MC-Group-IP
    1000          0.0.0.0
    2000          0.0.0.0
    3000          0.0.0.0
    4000          0.0.0.0
```

6. Verify that MAC addresses are learned through the VXLAN tunnels.

user@leaf-1> **show ethernet-switching vxlan-tunnel-end-point remote mac-table**

```
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P
 -Pinned MAC)

Logical system   : <default>
Routing instance : default-switch
 Bridging domain : VNI_1000+10, VLAN : 10, VNID : 1000
   MAC                 MAC      Logical          Remote VTEP
   address             flags    interface        IP address
   02:0c:10:01:02:04   DR       esi.1764         192.168.1.11 192.168.1.12
192.168.1.10
   02:0c:10:01:02:02   DR       esi.1771         192.168.1.6 192.168.1.4
   02:0c:10:01:02:03   DR       esi.1774         192.168.1.7

...

   0e:ad:10:01:00:60   D        vtep.32784       192.168.1.84
   0e:ad:10:01:00:30   D        vtep.32785       192.168.1.36
   0e:ad:10:01:00:48   D        vtep.32786       192.168.1.60
   0e:ad:10:01:00:4e   D        vtep.32788       192.168.1.66
   0e:ad:10:01:00:4c   D        vtep.32789       192.168.1.64
   0e:ad:10:01:00:36   D        vtep.32790       192.168.1.42

...

---(more)---
```

7. Verify multihoming information of the gateway and the aggregated Ethernet interfaces.

   `user@leaf-1>` **show ethernet-switching vxlan-tunnel-end-point esi**

```
## Local AE link - QFX5100 leaf devices
ESI                           RTT                     VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
00:00:00:00:00:00:51:00:00:01 default-switch          1768  131078  esi.1768
 ae11.0,   2
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.2          vtep.32780     1782     1         2
    192.168.1.3          vtep.32772     1767     0         2
## Remote AE Link for QFX5110 leaf devices
ESI                           RTT                     VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
00:00:00:00:00:00:51:10:00:01 default-switch          1771  131081  esi.1771
           3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.6          vtep.32771     1766     2         2
    192.168.1.4          vtep.32770     1765     1         2
    192.168.1.5          vtep.32774     1770     0         2
## Remote AE Link for QFX5200 leaf devices
ESI                           RTT                     VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
00:00:00:00:00:00:52:00:00:01 default-switch          1774  131084  esi.1774
           3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.9          vtep.32778     1776     2         2
    192.168.1.8          vtep.32777     1775     1         2
    192.168.1.7          vtep.32776     1773     0         2
## Remote AE Link for QFX10002 leaf devices
ESI                           RTT                     VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
00:00:00:00:00:01:00:00:00:01 default-switch          1764  131074  esi.1764
           3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.11         vtep.32775     1772     2         2
    192.168.1.12         vtep.32773     1769     1         2
    192.168.1.10         vtep.32769     1759     0         2

...
```

8. Verify that the VXLAN tunnel from one leaf to another leaf is load balanced with equal cost multipathing (ECMP) over the underlay.

user@leaf-1> **show route forwarding-table table default-switch extensive | find vtep.32770**

```
Destination:  vtep.32770
  Route type: interface
  Route reference: 0                   Route interface-index: 576
  Multicast RPF nh index: 0
  P2mpidx: 0
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite             Index: 1765     Reference: 12
  Next-hop type: indirect              Index: 131076   Reference: 3
  Next-hop type: unilist               Index: 131193   Reference: 238
  Nexthop: 172.16.1.2
  Next-hop type: unicast               Index: 1791     Reference: 10
  Next-hop interface: ae1.0         Weight: 0x0
  Nexthop: 172.16.1.6
  Next-hop type: unicast               Index: 1794     Reference: 10
  Next-hop interface: ae2.0         Weight: 0x0
  Nexthop: 172.16.1.10
  Next-hop type: unicast               Index: 1758     Reference: 10
  Next-hop interface: ae3.0         Weight: 0x0
  Nexthop: 172.16.1.14
  Next-hop type: unicast               Index: 1795     Reference: 10
  Next-hop interface: ae4.0         Weight: 0x0
```

9. Verify that remote MAC addresses are reachable through ECMP.

user@leaf-1> **show route forwarding-table table default-switch extensive destination 02:0c:10:01:02:03/48**

```
Routing table: default-switch.evpn-vxlan [Index 4]
Bridging domain: VNI_1000.evpn-vxlan [Index 3]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:01:02:03/48
  Learn VLAN: 0                        Route type: user
  Route reference: 0                   Route interface-index: 582
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 169                  Epoch: 0
  Sequence Number: 0                   Learn Mask:
0x40000000000000000300000000000000000000
  L2 Flags: control_dyn
```

```
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite              Index: 1773     Reference: 12
  Next-hop type: indirect               Index: 131085   Reference: 3
  Next-hop type: unilist                Index: 131193   Reference: 238
  Nexthop: 172.16.1.2
  Next-hop type: unicast                Index: 1791     Reference: 10
  Next-hop interface: ae1.0             Weight: 0x0
  Nexthop: 172.16.1.6
  Next-hop type: unicast                Index: 1794     Reference: 10
  Next-hop interface: ae2.0             Weight: 0x0
  Nexthop: 172.16.1.10
  Next-hop type: unicast                Index: 1758     Reference: 10
  Next-hop interface: ae3.0             Weight: 0x0
  Nexthop: 172.16.1.14
  Next-hop type: unicast                Index: 1795     Reference: 10
  Next-hop interface: ae4.0             Weight: 0x0
```

NOTE: Though the MAC address is reachable over multiple VTEP interfaces, QFX5100, QFX5110, QFX5120-32C, and QFX5200 switches do not support ECMP across the overlay because of a merchant ASIC limitation. Only the QFX10000 line of switches contain a custom Juniper Networks ASIC that supports ECMP across both the overlay and the underlay.

`user@leaf-1>` **show ethernet-switching table vlan-id 10 | match 02:0c:10:01:02:03**

```
   VNI_1000            02:0c:10:01:02:03   DR       esi.1774
00:00:00:00:00:00:52:00:00:01
```

`user@leaf-1>` **show route forwarding-table table default-switch extensive destination 02:0c:10:01:02:03/48**

```
Routing table: default-switch.evpn-vxlan [Index 9]
Bridging domain: VNI_1000.evpn-vxlan [Index 3]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:01:02:03/48
  Learn VLAN: 0                         Route type: user
  Route reference: 0                    Route interface-index: 550
  Multicast RPF nh index: 0
```

```
  P2mpidx: 0
  IFL generation: 0                    Epoch: 0
  Sequence Number: 0                   Learn Mask:
0x40000000000000000100000000000000000000
  L2 Flags: control_dyn, esi
  Flags: sent to PFE
  Next-hop type: indirect              Index: 2097173  Reference: 5
  Nexthop:
  Next-hop type: composite             Index: 1947     Reference: 2
  Nexthop:
  Next-hop type: composite             Index: 1948     Reference: 8
  Next-hop type: indirect              Index: 2097174  Reference: 3
  Next-hop type: unilist               Index: 2097280  Reference: 241
  Nexthop: 172.16.10.2
  Next-hop type: unicast               Index: 1950     Reference: 11
  Next-hop interface: ae1.0            Weight: 0x0
  Nexthop: 172.16.10.6
  Next-hop type: unicast               Index: 1956     Reference: 10
  Next-hop interface: ae2.0            Weight: 0x0
  Nexthop: 172.16.10.10
  Next-hop type: unicast               Index: 1861     Reference: 10
  Next-hop interface: ae3.0            Weight: 0x0
  Nexthop: 172.16.10.14
  Next-hop type: unicast               Index: 1960     Reference: 10
  Next-hop interface: ae4.0            Weight: 0x0
```

10. Verify which device is the Designated Forwarder (DF) for broadcast, unknown, and multicast (BUM) traffic coming from the VTEP tunnel.

> **NOTE:** Because the DF IP address is listed as 192.168.1.2, Leaf 2 is the DF.

`user@leaf-1>` **show evpn instance esi 00:00:00:00:00:00:51:00:00:01 designated-forwarder**

```
Instance: default-switch
  Number of ethernet segments: 12
    ESI: 00:00:00:00:00:00:51:00:00:01
      Designated forwarder: 192.168.1.2
```

SEE ALSO

EVPN Overview

Understanding VXLANs

Understanding EVPN with VXLAN Data Plane Encapsulation

Configuring Aggregated Ethernet LACP (CLI Procedure)

Bridged Overlay — Release History

Table 3 on page 94 provides a history of all of the features in this section and their support within this reference design.

Table 3: Bridged Overlay in the Cloud Data Center Reference Design– Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section.
18.1R3-S3	QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section.
17.3R3-S2	All devices in the reference design that support Junos OS Release 17.3R3-S2 and later releases in the same release train also support all features documented in this section.

RELATED DOCUMENTATION

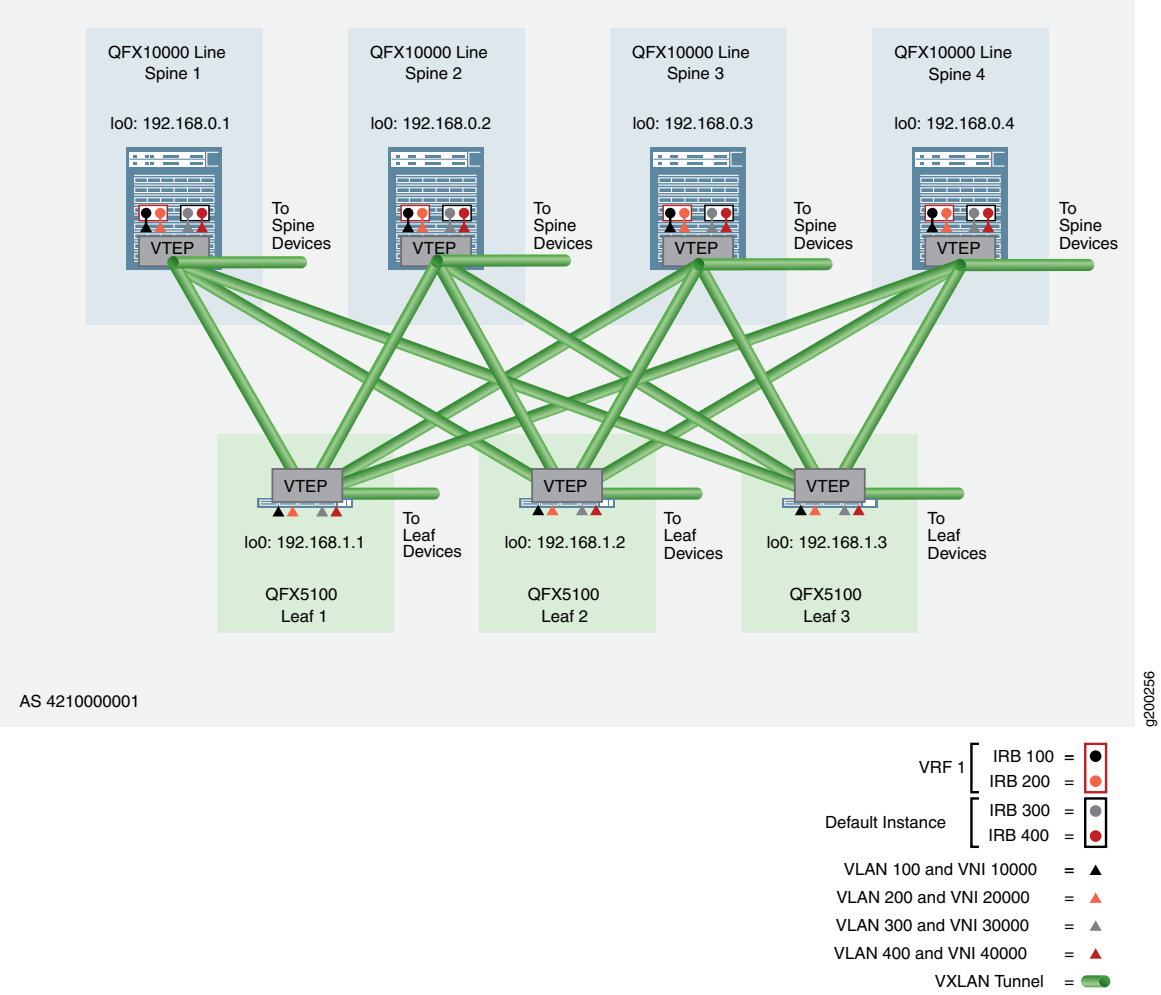
EVPN User Guide

Centrally-Routed Bridging Overlay Design and Implementation

IN THIS SECTION

A *centrally-routed bridging overlay* performs routing at a central location in the EVPN network as shown in Figure 31 on page 96, In this example, IRB interfaces are configured in the overlay at each spine device to route traffic between the VLANs that originate at the leaf devices and end systems. For an overview of centrally-routed bridging overlays, see the Centrally-Routed Bridging Overlay section in "Data Center Fabric Blueprint Architecture Components" on page 15.

Figure 31: Centrally-Routed Bridging Overlay



The following sections provide the detailed steps of how to implement a centrally-routed bridging overlay:

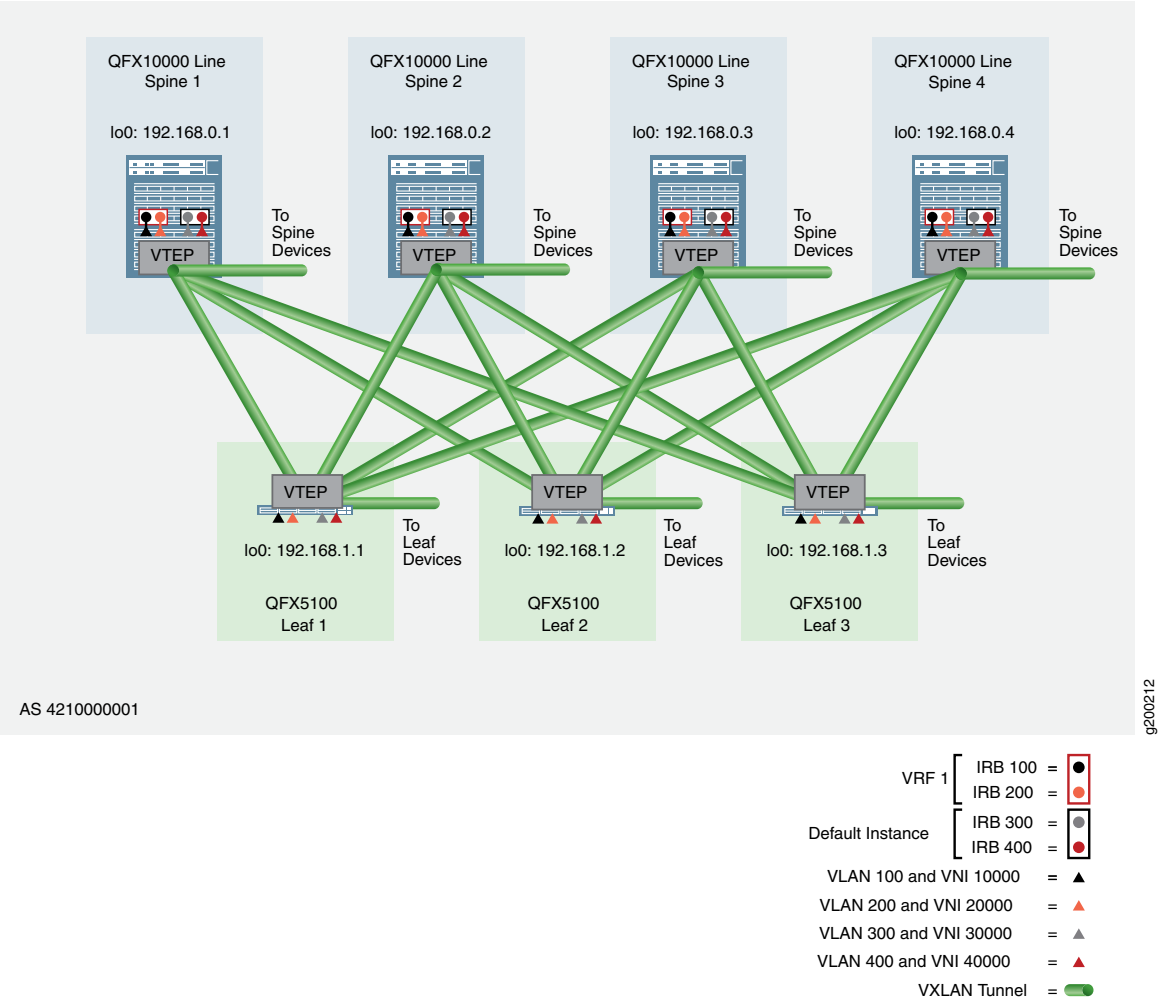
Configuring a VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance

IN THIS SECTION

This basic form of overlay is supported on all platforms included in this reference design. It uses the simplest VLAN-aware method to enable a single, default switching instance that supports up to 4094 VLANs.

As shown in Figure 32 on page 98, you configure VLANs at the leaf devices, and IRB interfaces for routing at the spine devices. Such configuration is placed in the default switching instance at the **[edit vlans]**, **[edit interfaces]**, **[edit protocols evpn]**, and **[edit switch-options]** hierarchy levels. Routing instances are not required for this overlay style, but can be implemented as an option depending on the needs of your network.

Figure 32: VLAN-Aware Centrally-Routed Bridging Overlay



When you implement this style of overlay on a spine device, you configure IRB interfaces to route traffic between Ethernet virtual network instances, set virtual gateway addresses, add VXLAN features to optimize traffic paths, configure EVPN with VXLAN encapsulation in the default switching instance or in a routing instance, set the loopback interface as the VTEP, configure route distinguishers and route targets to direct traffic to peers, and map VLANs to VNIs.

When you implement this style of overlay on a leaf device, you configure Ethernet Segment Identifier (ESI) settings, enable EVPN with VXLAN encapsulation in the default switching instance, establish route targets and route distinguishers, and map VLANs to VNIs.

For an overview of VLAN-aware centrally-routed bridging overlays, see the Centrally-Routed Bridging Overlay section in “Data Center Fabric Blueprint Architecture Components” on page 15. If you need to implement more than 4094 VLANs, see “Configuring a VLAN-Aware Centrally-Routed Bridging Overlay with Virtual Switches” on page 131.

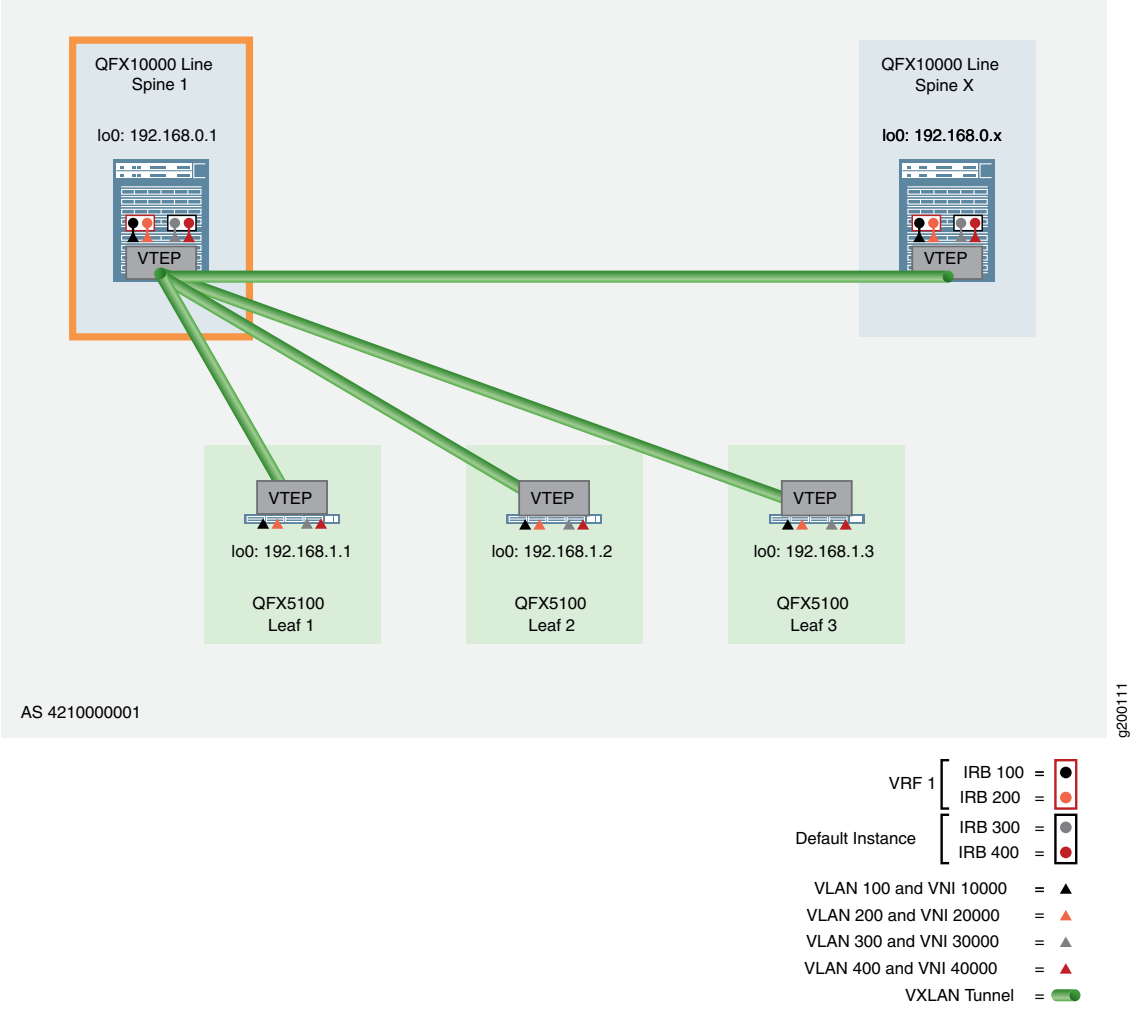
The following sections provide the detailed steps of how to configure and verify the VLAN-aware centrally-routed bridging overlay in the default switching instance:

Configuring a VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance on the Spine Device

To configure a VLAN-aware centrally-routed bridging overlay in the default switching instance on a spine device, perform the following:

> **NOTE:** The following example shows the configuration for Spine 1, as shown in Figure 33 on page 100.

Figure 33: VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance – Spine Device



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on a spine device, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine devices, see "Configuring IBGP for the Overlay" on page 67.

3. Configure the VTEP tunnel endpoint as the loopback address, and add a route distinguisher and a route target (target:64512:1111). Also, keep your configuration simple by using the auto route target option, which uses one target for both import and export.

   Spine 1:

```
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 192.168.0.1:1
set switch-options vrf-target target:64512:1111
set switch-options vrf-target auto
```

4. Configure IRB interfaces for each VNI and the corresponding virtual gateway address (which uses .254 in the 4th octet for each prefix). Include VXLAN features, such as **proxy-macip-advertisement** and **virtual-gateway-accept-data**, to improve performance and manageability.

   NOTE:
   - The **proxy-macip-advertisement** statement allows MAC address plus IP address information (ARP entries) learned locally for a subnet to be sent by one central gateway (spine device) to the other central gateways. This is referred to as ARP synchronization. This feature improves convergence times and traffic handling in the EVPN/VXLAN network.
   - You must configure both the **virtual-gateway-accept-data** statement and the preferred IPv4 and IPv6 addresses to use the ping operation and verify connectivity to the virtual gateway IP address from the end system.

   Spine 1:

```
set interfaces irb unit 100 family inet address 10.1.0.1/24 virtual-gateway-address 10.1.0.254
set interfaces irb unit 100 family inet address 10.1.0.1/24 preferred
set interfaces irb unit 100 proxy-macip-advertisement
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 family inet6 address 2001:db8::10:1:0:1/112 virtual-gateway-address
  2001:db8::10:1:0:254
set interfaces irb unit 100 family inet6 address fe80::10:1:0:254/112
set interfaces irb unit 200 family inet address 10.1.1.1/24 virtual-gateway-address 10.1.1.254
set interfaces irb unit 200 family inet address 10.1.1.1/24 preferred
set interfaces irb unit 200 proxy-macip-advertisement
set interfaces irb unit 200 virtual-gateway-accept-data
set interfaces irb unit 200 family inet6 address 2001:db8::10:1:1:1/112 virtual-gateway-address
  2001:db8::10:1:1:254
set interfaces irb unit 200 family inet6 address fe80::10:1:1:254/112
set interfaces irb unit 300 family inet address 10.1.2.1/24 virtual-gateway-address 10.1.2.254
```

```
set interfaces irb unit 300 family inet address 10.1.2.1/24 preferred
set interfaces irb unit 300 proxy-macip-advertisement
set interfaces irb unit 300 virtual-gateway-accept-data
set interfaces irb unit 300 family inet6 address 2001:db8::10:1:2:1/112 virtual-gateway-address
  2001:db8::10:1:2:254
set interfaces irb unit 300 family inet6 address fe80::10:1:2:254/112
set interfaces irb unit 400 family inet address 10.1.3.1/24 virtual-gateway-address 10.1.3.254
set interfaces irb unit 400 family inet address 10.1.3.1/24 preferred
set interfaces irb unit 400 proxy-macip-advertisement
set interfaces irb unit 400 virtual-gateway-accept-data
set interfaces irb unit 400 family inet6 address 2001:db8::10:1:3:1/112 virtual-gateway-address
  2001:db8::10:1:3:254
set interfaces irb unit 400 family inet6 address fe80::10:1:3:254/112
```

5. Configure a secondary logical unit on the loopback interface for the default switching instance.

   Spine 1:

   ```
   set interfaces lo0 unit 1 family inet address 192.168.0.101/32
   ```

6. Configure EVPN with VXLAN encapsulation. Include the **no-gateway-community** option to advertise the virtual gateway and IRB MAC addresses to the EVPN peer devices so that Ethernet-only PE devices can learn these MAC addresses.

   Spine 1:

   ```
   set protocols evpn encapsulation vxlan
   set protocols evpn default-gateway no-gateway-community
   set protocols evpn extended-vni-list all
   ```

7. Configure mapping between VLANs and VXLAN VNIs.

   Spine 1:

   ```
   set vlans VNI_10000 vlan-id 100
   set vlans VNI_10000 l3-interface irb.100
   set vlans VNI_10000 vxlan vni 10000
   set vlans VNI_20000 vlan-id 200
   set vlans VNI_20000 l3-interface irb.200
   set vlans VNI_20000 vxlan vni 20000
   set vlans VNI_30000 vlan-id 300
   set vlans VNI_30000 l3-interface irb.300
   ```

```
set vlans VNI_30000 vxlan vni 30000
set vlans VNI_40000 vlan-id 400
set vlans VNI_40000 l3-interface irb.400
set vlans VNI_40000 vxlan vni 40000
```

8. Configure a routing instance named VRF 1, and map IRB interfaces irb.100 (VNI 10000) and irb.200 (VNI 20000) to this instance.

> **NOTE:** Because the irb.300 (VNI 30000) and irb.400 (VNI 40000) interfaces are not configured inside a routing instance, they are part of the default switching instance for the spine devices. The end result of your configuration should match the diagram shown in Figure 33 on page 100.

Spine 1:

```
set routing-instances VRF_1 instance-type vrf
set routing-instances VRF_1 interface irb.100
set routing-instances VRF_1 interface irb.200
set routing-instances VRF_1 interface lo0.1
set routing-instances VRF_1 route-distinguisher 192.168.0.1:100
set routing-instances VRF_1 vrf-target target:62273:10000
```

Verifying the VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance for the Spine Device

Issue the following commands to verify that the overlay is working properly on your spine devices:

1. Verify the IRB interfaces are operational for both IPv4 and IPv6.

   `user@spine-1>` **show interfaces terse irb**

```
Interface               Admin Link Proto    Local                 Remote
irb                     up    up
irb.100                 up    up   inet     10.1.0.1/24
                                    inet6    2001:db8::10:1:0:1/112
                                             fe80::10:1:0:254/112
irb.200                 up    up   inet     10.1.1.1/24
                                    inet6    2001:db8::10:1:1:1/112
                                             fe80::10:1:1:254/112
irb.300                 up    up   inet     10.1.2.1/24
                                    inet6    2001:db8::10:1:2:1/112
                                             fe80::10:1:2:254/112
irb.400                 up    up   inet     10.1.3.1/24
                                    inet6    2001:db8::10:1:3:1/112
                                             fe80::10:1:3:254/112
```

2. Verify that the VTEP interfaces are up.

   `user@spine-1>` **show interfaces terse vtep**

```
Interface               Admin Link Proto    Local                 Remote
vtep                    up    up
vtep.32768              up    up
vtep.32769              up    up   eth-switch
vtep.32770              up    up   eth-switch
vtep.32771              up    up   eth-switch
vtep.32772              up    up   eth-switch
...
vtep.32804              up    up   eth-switch
---(more)---
```

   `user@spine-1>` **show interfaces terse vtep | match eth-switch | count**

```
Count: 109 lines
```

3. Verify the endpoint destination IP address for the VTEP interfaces. The spine devices display their VTEPs as loopback addresses in the range of 192.168.0.*x* (1 - 4) and the leaf devices display their VTEPs as loopback addresses in the range of 192.168.1.*x* (1 - 96).

   user@spine-1> **show interfaces vtep**

```
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 240, SNMP ifIndex: 504
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited,
 Speed: Unlimited
  Device flags   : Present Running
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 670) (SNMP ifIndex 505)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 192.168.0.1, L2 Routing
 Instance: default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 0

...

  Logical interface vtep.32771 (Index 802) (SNMP ifIndex 536)
    Flags: Up SNMP-Traps Encapsulation: ENET2
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.4, L2 Routing
 Instance: default-switch, L3 Routing Instance: default
    Input packets : 1979
    Output packets: 9867
    Protocol eth-switch, MTU: Unlimited
---(more)---
```

4. Verify that the spine device has all the routes to the leaf devices.

   user@spine-2> **show route 192.168.1.1**

```
inet.0: 446 destinations, 19761 routes (446 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.1.1/32     *[BGP/170] 00:06:29, localpref 100
                      AS path: 4200000011 I, validation-state: unverified
```

```
                    > to 172.16.1.5 via ae1.0
                    [BGP/170] 00:06:22, localpref 100
                      AS path: 4200000023 4200000004 4200000011 I,
validation-state: unverified
                    > to 172.16.13.5 via ae13.0

                     ...

                     [BGP/170] 00:06:18, localpref 100
                      AS path: 4200000032 4200000004 4200000011 I,
validation-state: unverified
                    > to 172.16.22.5 via ae22.0
---(more)---
```

5. Verify that each end system resolves the virtual gateway MAC address for a subnet using the gateway IRB address on the central gateways (spine devices).

   user@spine-1> **show arp no-resolve vpn VRF_1**

```
MAC Address       Address         Interface         Flags
06:4b:8c:cd:13:f8 10.1.0.2        irb.100 [vtep.32796]     none  ## Spine 2 IRB
 interface
06:4b:8c:cd:c4:38 10.1.0.3        irb.100 [vtep.32878]     none  ## Spine 3 IRB
 interface
06:38:e1:6f:30:29 10.1.0.4        irb.100 [vtep.32821]     none  ## Spine 4 IRB
 interface02:0c:10:01:02:01 10.1.0.201      irb.100 [.local..11]     none  ##
End system behind the QFX5100s
02:0c:10:01:02:02 10.1.0.202      irb.100 [.local..11]     none  ## End system
 behind the QFX5110s
02:0c:10:01:02:03 10.1.0.203      irb.100 [.local..11]     none  ## End system
 behind the QFX5200s
02:0c:10:01:02:04 10.1.0.204      irb.100 [.local..11]     none  ## End system
 behind the QFX10002s
00:00:5e:00:01:01 10.1.0.254      irb.100                  permanent published
 gateway
                                                           ## Virtual gateway
IP and MAC address
06:4b:8c:cd:13:f8 10.1.1.2        irb.200 [vtep.32796]     none
06:4b:8c:cd:c4:38 10.1.1.3        irb.200 [vtep.32878]     none
06:38:e1:6f:30:29 10.1.1.4        irb.200 [vtep.32821]     none
0e:ad:10:02:00:01 10.1.1.101      irb.200 [vtep.32776]     none
```

   user@spine-1> **show ipv6 neighbors**

```
IPv6 Address                  Linklayer Address  State       Exp Rtr Secure
Interface
2001:db8::10:1:0:2            06:4b:8c:cd:13:f8  stale       325 no  no
irb.100 [vtep.32796]
2001:db8::10:1:0:3            06:4b:8c:cd:c4:38  stale       514 yes no
irb.100 [vtep.32878]
2001:db8::10:1:0:4            06:38:e1:6f:30:29  stale       326 no  no
irb.100 [vtep.32821]
2001:db8::10:1:0:201          02:0c:10:01:02:01  stale       1114 no no
irb.100 [.local..11]
2001:db8::10:1:0:202          02:0c:10:01:02:02  stale       443 no  no
irb.100 [.local..11]
2001:db8::10:1:0:203          02:0c:10:01:02:03  stale       853 no  no
irb.100 [.local..11]
2001:db8::10:1:0:204          02:0c:10:01:02:04  stale       1181 no no
irb.100 [.local..11]
2001:db8::10:1:0:254          00:00:5e:00:02:01  reachable   0   no  no
irb.100
2001:db8::10:1:1:2            06:4b:8c:cd:13:f8  stale       325 no  no
irb.200 [vtep.32796]
2001:db8::10:1:1:3            06:4b:8c:cd:c4:38  stale       514 yes no
irb.200 [vtep.32878]
2001:db8::10:1:1:4            06:38:e1:6f:30:29  stale       326 no  no
irb.200 [vtep.32821]
2001:db8::10:1:1:201          02:0c:10:02:02:01  stale       1121 no no
irb.200 [.local..11]
2001:db8::10:1:1:202          02:0c:10:02:02:02  stale       423 no  no
irb.200 [.local..11]
2001:db8::10:1:1:203          02:0c:10:02:02:03  stale       1081 no no
irb.200 [.local..11]
2001:db8::10:1:1:204          02:0c:10:02:02:04  stale       1167 no no
irb.200 [.local..11]
```

6. Verify the switching table for VNI 10000 to see entries for end systems and the other spine devices.

   `user@spine-1>` **show ethernet-switching table vlan-id 100**

```
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent
 static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC,
 O - ovsdb MAC)
```

```
Ethernet switching table : 105 entries, 105 learned
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical              Active

   name                address             flags    interface            source

   VNI_10000           00:00:5e:00:01:01   DR       esi.2453
05:19:17:f3:41:00:00:27:10:00
```

```
## Entries for the spine devices
   VNI_10000           06:4b:8c:cd:13:f8   D        vtep.32796
192.168.0.2
   VNI_10000           06:4b:8c:cd:c4:38   D        vtep.32878
192.168.0.3
   VNI_10000           06:38:e1:6f:30:29   D        vtep.32821
192.168.0.4
## The next four MAC addresses belong to the end systems
connected to Leaf 1 - 3 (QFX5100), Leaf 4-6 (QFX5110), Leaf 7-9 (QFX5200),
and Leaf 10-12 (QFX10002).
   VNI_10000           02:0c:10:01:02:01   DR       esi.2443
00:00:00:00:00:00:51:00:00:01
   VNI_10000           02:0c:10:01:02:02   DR       esi.2497
00:00:00:00:00:00:51:10:00:01
   VNI_10000           02:0c:10:01:02:03   DR       esi.2427
00:00:00:00:00:00:52:00:00:01
   VNI_10000           02:0c:10:01:02:04   DR       esi.2610
00:00:00:00:00:01:00:00:00:01
   ...
   VNI_10000           0e:ad:10:01:00:02   D        vtep.32814
192.168.1.96
```

7. Verify MAC address and ARP information learned from the leaf devices over the control plane.

   `user@spine-1>` **show evpn database mac-address 02:0c:10:01:02:01 extensive**

```
Instance: default-switch

VN Identifier: 10000, MAC address:: 02:0c:10:01:02:01
  Source: 00:00:00:00:00:00:51:00:00:01, Rank: 1, Status: Active
    Remote origin: 192.168.1.2  ## Leaf 2 and Leaf 3 advertised this route
    Remote origin: 192.168.1.3
    Timestamp: Jul 13 23:35:37 (0x59686639)
    State: <Remote-To-Local-Adv-Done>
```

```
    IP address: 10.1.0.201  ## MAC Address + IP
      Flags: <Proxy>
      Remote origin: 192.168.1.2
      Remote origin: 192.168.1.3
    IP address: 2001:db8::10:1:0:201   ## MAC Address + IPv6
      Remote origin: 192.168.1.2
      Remote origin: 192.168.1.3    History db:
      Time                  Event
      Jul 13 23:35:38 2017  Applying remote state to peer 192.168.1.2
      Jul 13 23:35:38 2017  Remote peer 192.168.1.2 updated
     Jul 13 23:35:38 2017  MAC+IP not updated, source l2ald is not owner (type2)

      Jul 13 23:35:38 2017  Updated
      Jul 13 23:35:38 2017  No change to MAC state
      Jul 13 23:35:38 2017  Applying remote state to peer 192.168.1.3
      Jul 13 23:35:38 2017  Remote peer 192.168.1.3 updated
     Jul 13 23:35:38 2017  MAC+IP not updated, source l2ald is not owner (type2)

      Jul 13 23:35:38 2017  Updated
      Jul 13 23:35:38 2017  No change to MAC state
```

8. Verify the remote VXLAN tunnel end points.

   `user@spine-1>` **show ethernet-switching vxlan-tunnel-end-point remote**

```
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx
<default>                 0   192.168.0.1      lo0.0    0
 RVTEP-IP         IFL-Idx   NH-Id
 192.168.1.1      827       2444
    VNID          MC-Group-IP
    10000         0.0.0.0
    20000         0.0.0.0
    30000         0.0.0.0
    40000         0.0.0.0
 RVTEP-IP         IFL-Idx   NH-Id
...
RVTEP-IP         IFL-Idx   NH-Id
 192.168.1.96      812       2428
    VNID          MC-Group-IP
    10000         0.0.0.0
    20000         0.0.0.0
```

```
    30000         0.0.0.0
    40000         0.0.0.0
```

9. Verify that MAC addresses are learned through the VXLAN tunnel.

   `user@spine-1>` **show ethernet-switching vxlan-tunnel-end-point remote mac-table**

```
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P
 -Pinned MAC)

Logical system   : <default>
Routing instance : default-switch
 Bridging domain : VNI_10000+100, VLAN : 100, VNID : 10000
   MAC                 MAC      Logical          Remote VTEP
   address             flags    interface        IP address
   02:0c:10:01:02:03   DR       esi.2427         192.168.1.8 192.168.1.7
192.168.1.9
   02:0c:10:01:02:01   DR       esi.2443         192.168.1.2 192.168.1.3
## This next entry shows that the virtual gateway MAC address
of 00:00:5e:00:01:01 has been learned by
the other spine devices.
   00:00:5e:00:01:01   DR       esi.2453         192.168.0.3 192.168.0.4
192.168.0.2
   02:0c:10:01:02:02   DR       esi.2497         192.168.1.6 192.168.1.4
   02:0c:10:01:02:04   DR       esi.2610         192.168.1.12 192.168.1.10
192.168.1.11
   06:4b:8c:cd:13:f8   D        vtep.32796       192.168.0.2
---(more)---
```

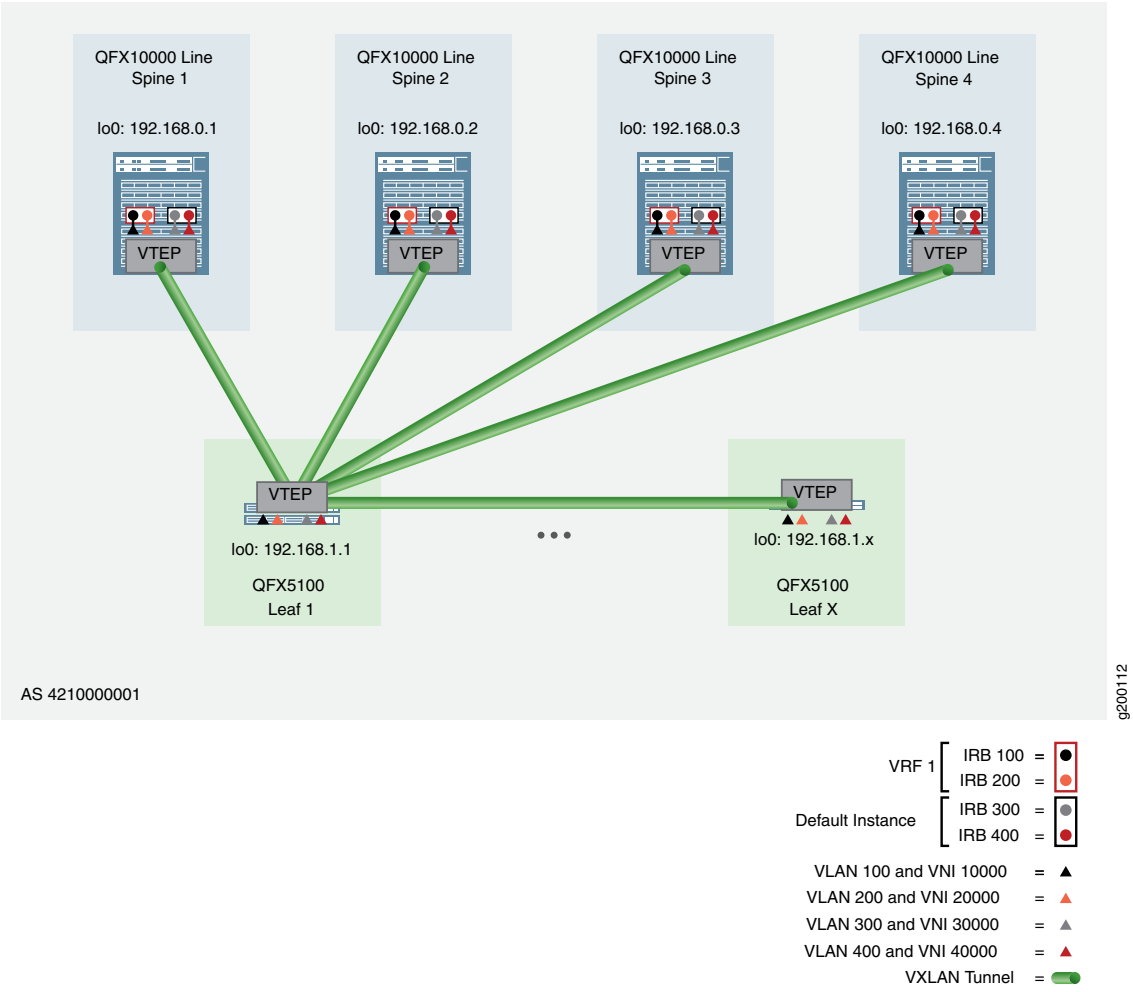
Configuring a VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance on the Leaf Device

To configure a VLAN-aware centrally-routed bridging overlay in the default switching instance on a leaf device, perform the following:

NOTE:

- The following example shows the configuration for Leaf 1, as shown in Figure 34 on page 111.

Figure 34: VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance – Leaf Device



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on a leaf device, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your leaf device, see "Configuring IBGP for the Overlay" on page 67.

3. Configure the EVPN protocol with VXLAN encapsulation, and specify the VTEP source interface (in this case, the loopback interface of the leaf device).

   Leaf 1:

   ```
   set protocols evpn encapsulation vxlan
   set protocols evpn extended-vni-list all
   set switch-options vtep-source-interface lo0.0
   ```

4. Define an EVPN route target and route distinguisher, and use the **auto** option to derive route targets automatically.Setting these parameters specifies how the routes are imported and exported. The import and export of routes from a routing or bridging table is the basis for dynamic overlays. In this case, members of the global BGP community with a route target of target:64512:1111 participate in the exchange of EVPN/VXLAN information.

   Leaf 1:

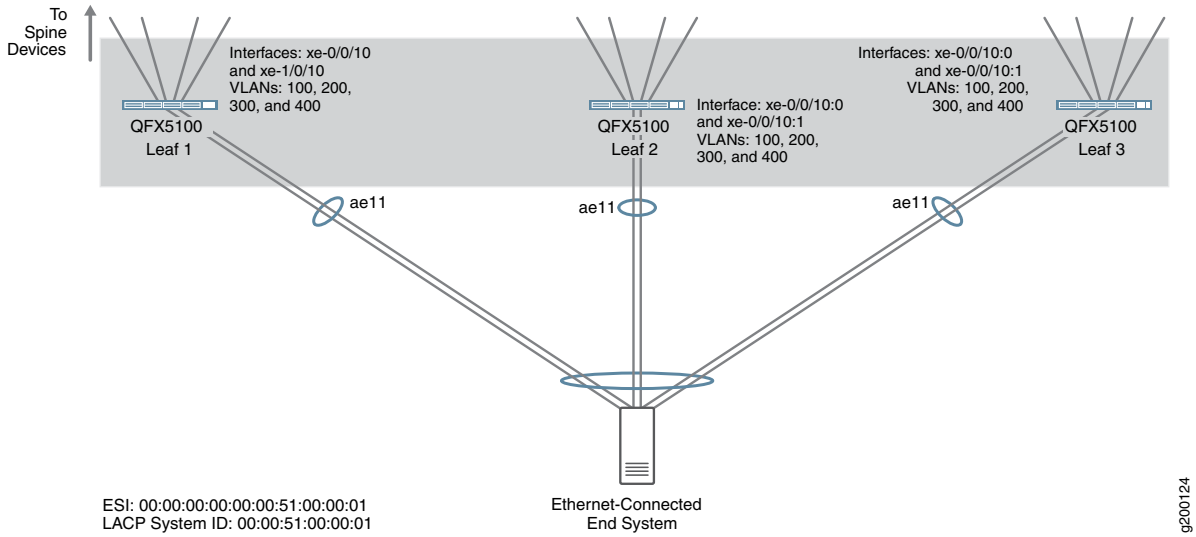
   ```
   set switch-options route-distinguisher 192.168.1.1:1
   set switch-options vrf-target target:64512:1111
   set switch-options vrf-target auto
   ```

5. Configure ESI settings on all similar leaf devices. Because the end systems in this reference design are multihomed to three leaf devices per device type cluster (such as QFX5100), you must configure the same ESI identifier and LACP system identifier on all three leaf devices for each unique end system. Unlike other topologies where you would configure a different LACP system identifier per leaf device and have VXLAN select a single designated forwarder, use the same LACP system identifier to allow the 3 leaf devices to appear as a single LAG to a multihomed end system. In addition, use the same aggregated Ethernet interface number for all ports included in the ESI.

   The configuration for Leaf 1 is shown below, but you must replicate this configuration on both Leaf 2 and Leaf 3 per the topology shown in Figure 35 on page 113.

   TIP: When you create an ESI number, always set the high order octet to 00 to indicate the ESI is manually created. The other 9 octets can be any hexadecimal value from 00 to FF.

Figure 35: ESI Topology for Leaf 1, Leaf 2, and Leaf 3



Leaf 1:

```
set interfaces ae11 esi 00:00:00:00:00:00:51:00:00:01
set interfaces ae11 esi all-active
set interfaces ae11 aggregated-ether-options lacp system-id 00:00:51:00:00:01
set interfaces xe-0/0/10 ether-options 802.3ad ae11
set interfaces xe-1/0/10 ether-options 802.3ad ae11
```

6. Configure VLANs and map them to VNIs. This step enables the VLANs to participate in VNIs across the EVPN/VXLAN domain.

   Leaf 1:

```
set vlans VNI_10000 vlan-id 100
set vlans VNI_10000 vxlan vni 10000
set vlans VNI_20000 vlan-id 200
set vlans VNI_20000 vxlan vni 20000
set vlans VNI_30000 vlan-id 300
set vlans VNI_30000 vxlan vni 30000
set vlans VNI_40000 vlan-id 400
set vlans VNI_40000 vxlan vni 40000
```

Verifying the VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance for the Leaf Device

Issue the following commands to verify that the overlay is working properly on your leaf devices:

1. Verify the interfaces are operational.

```
user@leaf-1> show interfaces terse | match ae.*
xe-0/0/10.0             up    up   aenet    --> ae11.0
et-0/0/48.0             up    up   aenet    --> ae1.0
et-0/0/49.0             up    up   aenet    --> ae2.0
et-0/0/50.0             up    up   aenet    --> ae3.0
et-0/0/51.0             up    up   aenet    --> ae4.0
xe-1/0/10.0             up    up   aenet    --> ae11.0
et-1/0/48.0             up    up   aenet    --> ae1.0
et-1/0/49.0             up    up   aenet    --> ae2.0
et-1/0/50.0             up    up   aenet    --> ae3.0
et-1/0/51.0             up    up   aenet    --> ae4.0
ae1                     up    up  ## To Spine 1
ae1.0                   up    up   inet     172.16.1.1/30
ae2                     up    up  ## To Spine 2
ae2.0                   up    up   inet     172.16.1.5/30
ae3                     up    up  ## To Spine 3
ae3.0                   up    up   inet     172.16.1.9/30
ae4                     up    up  ## To Spine 4
ae4.0                   up    up   inet     172.16.1.13/30
ae11                    up    up  ## To End System
ae11.0                  up    up   eth-switch
```

```
user@leaf-1> show lacp interfaces
Aggregated interface: ae1
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/48      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/48    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-1/0/48      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-1/0/48    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/48               Current   Fast periodic Collecting distributing
      et-1/0/48               Current   Fast periodic Collecting distributing

...
```

```
Aggregated interface: ae11
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/10      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/10    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-1/0/10      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-1/0/10    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/10               Current   Fast periodic Collecting distributing
      xe-1/0/10               Current   Fast periodic Collecting distributing
```

2. Verify that the EVPN routes are being learned through the overlay.

NOTE:

- Only selected excerpts of this output are displayed.
- The format of the EVPN routes is *EVPN-route-type*:*route-distinguisher*:*vni*:*mac-address*.

`user@leaf-1>` **show route table bgp.evpn.0 evpn-ethernet-tag-id 10000**

```
bgp.evpn.0: 828 destinations, 3169 routes (827 active, 0 holddown, 4 hidden)
+ = Active Route, - = Last Active, * = Both

## Spine 1: Virtual Gateway MAC Address for IPv4
2:192.168.0.1:1::10000::00:00:5e:00:01:01/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
```

```
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 1: Virtual Gateway MAC Address for IPv6
2:192.168.0.1:1::10000::00:00:5e:00:02:01/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 1: IRB MAC Address
2:192.168.0.1:1::10000::06:4b:8c:67:0f:f0/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 1: ARP for the virtual gateway
2:192.168.0.1:1::10000::00:00:5e:00:01:01::10.1.0.254/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
```

```
## Spine 1: IRB IPv4 ARP
2:192.168.0.1:1::10000::06:4b:8c:67:0f:f0::10.1.0.1/304 MAC/IP
                   *[BGP/170] 00:04:50, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 00:04:50, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 00:04:50, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 00:04:50, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 2: ARP for the virtual gateway
2:192.168.0.2:1::10000::00:00:5e:00:01:01::10.1.0.254/304 MAC/IP
                   *[BGP/170] 07:55:22, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:31:11, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:29:37, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
## Spine 2: IRB IPv4 ARP
2:192.168.0.2:1::10000::06:4b:8c:cd:13:f8::10.1.0.2/304 MAC/IP
                   *[BGP/170] 07:55:22, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:31:11, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:29:37, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
## Spine 1: IPv6 ARP for the virtual gateway
2:192.168.0.1:1::10000::00:00:5e:00:02:01::2001:db8::10:1:0:254/304 MAC/IP
```

```
                    *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                       AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                       AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                       AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                       AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
## Spine 1: IRB IPv6 ARP
2:192.168.0.1:1::10000::06:4b:8c:67:0f:f0::2001:db8::10:1:0:1/304 MAC/IP
                    *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                       AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                       AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                       AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0
                     [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                       AS path: I, validation-state: unverified
                     > to 172.16.1.2 via ae1.0

...
```

3. Verify on Leaf 1 and Leaf 3 that the Ethernet switching table has installed both the local MAC addresses and the remote MAC addresses learned through the overlay.

> **NOTE:** To identify end systems learned remotely from the EVPN overlay, look for the MAC address, ESI logical interface, and ESI number. For example, Leaf 1 learns about an end system with the MAC address of **02:0c:10:03:02:02** through **esi.1885**. This end system has an ESI number of **00:00:00:00:00:00:51:10:00:01**. Consequently, this matches the ESI number configured for Leaf 4, 5, and 6 (QFX5110 switches), so we know that this end system is multihomed to these three leaf devices.

```
user@leaf-1> show ethernet-switching table vlan-id 300
```

```
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent
 static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC,
 O - ovsdb MAC)


Ethernet switching table : 10 entries, 10 learned
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical              Active

   name                address             flags    interface            source

   VNI_30000           00:00:5e:00:01:01   DR       esi.1679
05:19:17:f3:41:00:00:75:30:00
   VNI_30000           00:00:5e:00:02:01   DR       esi.1679
05:19:17:f3:41:00:00:75:30:00
   VNI_30000           06:4b:8c:67:0f:f0   D        vtep.32770
192.168.0.1
   VNI_30000           06:4b:8c:cd:13:f8   D        vtep.32783
192.168.0.2
   VNI_30000           06:4b:8c:cd:c4:38   D        vtep.32769
192.168.0.3
   VNI_30000           06:38:e1:6f:30:29   D        vtep.32879
192.168.0.4
## Learned locally
   VNI_30000           02:0c:10:03:02:01   DL       ae11.0
## Learned from the QFX5110 switches - Leaf 4 to 6
   VNI_30000           02:0c:10:03:02:02   DR       esi.1885
00:00:00:00:00:00:51:10:00:01
## Learned from the QFX5200 switches - Leaf 7 to 9
   VNI_30000           02:0c:10:03:02:03   DR       esi.1887
00:00:00:00:00:00:52:00:00:01
## Learned from the QFX10002 switches - Leaf 10 to 12
   VNI_30000           02:0c:10:03:02:04   DR       esi.1892
00:00:00:00:00:01:00:00:00:01
## IPv4 virtual gateway MAC address learned over the overlay
and distributed to the leaf devices by Spine 1, 2, 3 and 4
00:00:5e:00:01:01
# IPv6 virtual gateway MAC address learned over Overlay
00:00:5e:00:02:01
## IRB MAC address prefix for Spine 1, 2, and 3 (Physical
MAC address)
06:4b:*
```

```
## End System MAC address, connected locally to the leaf
device
02:0c:10:03:02:01
## MAC address learned over the overlay, these end systems
are also multihomed
02:0c:10:03:02:02,03,04
```

user@leaf-3> **show ethernet-switching table vlan-id 100**

```
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent
 static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC,
 O - ovsdb MAC)


Ethernet switching table : 106 entries, 106 learned
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical              Active

   name                address             flags    interface            source
## 00:00:5e:00:01:01 is the virtual gateway MAC address
for the spine devices and is reachable over the dynamically created
logical link esi.1679. As a result, you can use this ESI number to
filter future command output by using esi.1679 to find the virtual
gateway.
   VNI_10000           00:00:5e:00:01:01   DR       esi.1769
05:19:17:f3:41:00:00:27:10:00
   VNI_10000           00:00:5e:00:02:01   DR       esi.1769
05:19:17:f3:41:00:00:27:10:00
   VNI_10000           06:4b:8c:67:0f:f0   D        vtep.32781
192.168.0.1
   VNI_10000           06:4b:8c:cd:13:f8   D        vtep.32782
192.168.0.2
   VNI_10000           06:4b:8c:cd:c4:38   D        vtep.32775
192.168.0.3
## Learned locally
   VNI_10000           02:0c:10:01:02:01   DL       ae11.0
## Learned through the overlay
   VNI_10000           02:0c:10:01:02:02   DR       esi.1760
00:00:00:00:00:00:51:10:00:01
   VNI_10000           02:0c:10:01:02:03   DR       esi.1782
00:00:00:00:00:00:52:00:00:01
   VNI_10000           02:0c:10:01:02:04   DR       esi.1758
```

```
00:00:00:00:00:01:00:00:00:01
   VNI_10000           06:38:e1:6f:30:29   D        vtep.32783
192.168.0.4
   VNI_10000           0e:ad:10:01:00:01   D        vtep.32821
192.168.1.85
```

4. Verify on Leaf 1 that the virtual gateway ESI (esi.1679) is reachable by all the spine devices.

   `user@leaf-1>` **show ethernet-switching vxlan-tunnel-end-point esi | find esi.1679**

```
ESI                             RTT                      VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
05:19:17:f3:41:00:00:75:30:00 default-switch           1679  131072  esi.1679
           4
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.0.4          vtep.32879     1890     3         2
    192.168.0.2          vtep.32783     1795     2         2
    192.168.0.1          vtep.32770     1682     1         2
    192.168.0.3          vtep.32769     1764     0         2
```

5. Verify the remote EVPN routes coming from VNI 10000 and MAC address 02:0c:10:01:02:02. In this case, they are coming from Leaf 4 (192.168.1.4) by way of Spine 1 (192.168.0.1).

   NOTE: The format of the EVPN routes is *EVPN-route-type:route-distinguisher:vni:mac-address*.

   `user@leaf-1>` **show route table bgp.evpn.0 evpn-ethernet-tag-id 10000 evpn-mac-address 02:0c:10:01:02:02**

```
bgp.evpn.0: 910 destinations, 3497 routes (904 active, 0 holddown, 24 hidden)
+ = Active Route, - = Last Active, * = Both

2:192.168.1.4:1::10000::02:0c:10:01:02:02/304 MAC/IP
                   *[BGP/170] 00:11:37, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                     > to 172.16.1.10 via ae3.0
                     [BGP/170] 00:11:37, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                     > to 172.16.1.10 via ae3.0
                     [BGP/170] 00:11:37, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
```

```
                    > to 172.16.1.10 via ae3.0
                    [BGP/170] 00:11:37, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
```

user@leaf-1> **show route table bgp.evpn.0 evpn-ethernet-tag-id 10000 evpn-mac-address 02:0c:10:01:02:02 detail**

```
bgp.evpn.0: 925 destinations, 3557 routes (919 active, 0 holddown, 24 hidden)
2:192.168.1.4:1::10000::02:0c:10:01:02:02/304 MAC/IP (4 entries, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.1.4:1
                Next hop type: Indirect, Next hop index: 0
                Address: 0xb3a2170
                Next-hop reference count: 160
                Source: 192.168.0.1
                Protocol next hop: 192.168.1.4
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <Active Int Ext>
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 13:42      Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.0.1
                AS path: I (Originator)
                Cluster list:  192.168.0.10
                Originator ID: 192.168.1.4
                Communities: target:62273:268445456 encapsulation:vxlan(0x8)
                Import Accepted
                Route Label: 10000
                ESI: 00:00:00:00:00:00:51:10:00:01
                Localpref: 100
                Router ID: 192.168.0.1
                Secondary Tables: default-switch.evpn.0
...
## This output has been abbreviated. In a full set of output,
there should also be routes sourced by Spine 2 (192.168.0.2), Spine
3 (192.168.0.3), and Spine 4 (192.168.0.4).
```

6. Verify the source and destination address of each VTEP interface and view their status.

NOTE: There are 96 leaf devices and four spine devices, so there are 100 VTEP interfaces in this reference design - one VTEP interface per device.

user@leaf-1> **show ethernet-switching vxlan-tunnel-end-point source**

```
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx
<default>                 0   192.168.1.1      lo0.0    0
    L2-RTT                   Bridge Domain              VNID     MC-Group-IP
    default-switch           VNI_10000+100              10000    0.0.0.0

    default-switch           VNI_20000+200              20000    0.0.0.0

    default-switch           VNI_30000+300              30000    0.0.0.0

    default-switch           VNI_40000+400              40000    0.0.0.0
```

user@leaf-1> **show interfaces terse vtep**

```
Interface               Admin Link Proto    Local                 Remote
vtep                    up    up
vtep.32768              up    up
vtep.32769              up    up   eth-switch
vtep.32770              up    up   eth-switch
vtep.32771              up    up   eth-switch
vtep.32772              up    up   eth-switch
...
vtep.32869              up    up   eth-switch
```

user@leaf-1> **show interfaces vtep**

```
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 646, SNMP ifIndex: 503
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited,
 Speed: Unlimited
  Device flags   : Present Running
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 554) (SNMP ifIndex 648)
```

```
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
   VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 192.168.1.1, L2 Routing
 Instance: default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 0


...  Logical interface vtep.32814 (Index 613) (SNMP ifIndex 903)
    Flags: Up SNMP-Traps Encapsulation: ENET2
   VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.96, L2 Routing
 Instance: default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 6364
    Protocol eth-switch, MTU: Unlimited
      Flags: Trunk-Mode
```

7. Verify that each VNI maps to the associated VXLAN tunnel.

   `user@leaf-1>` **show ethernet-switching vxlan-tunnel-end-point remote**

```
                0   192.168.1.1      lo0.0    0
 RVTEP-IP         IFL-Idx   NH-Id
 192.168.0.1      587       1792
    VNID          MC-Group-IP
    10000         0.0.0.0
    20000         0.0.0.0
    30000         0.0.0.0
    40000         0.0.0.0

...

RVTEP-IP         IFL-Idx   NH-Id
192.168.1.96     613       1820
    VNID          MC-Group-IP
    10000         0.0.0.0
    20000         0.0.0.0
    30000         0.0.0.0
    40000         0.0.0.0
```

8. Verify that MAC addresses are learned through the VXLAN tunnels.

   `user@leaf-1>` **show ethernet-switching vxlan-tunnel-end-point remote mac-table**

```
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P
 -Pinned MAC)

Logical system   : <default>
Routing instance : default-switch
 Bridging domain : VNI_10000+100, VLAN : 100, VNID : 10000
   MAC                 MAC      Logical          Remote VTEP
   address             flags    interface        IP address
   02:0c:10:01:02:04   DR       esi.1764         192.168.1.11 192.168.1.12
192.168.1.10
   02:0c:10:01:02:02   DR       esi.1771         192.168.1.6 192.168.1.4
   02:0c:10:01:02:03   DR       esi.1774         192.168.1.7
   00:00:5e:00:01:01   DR       esi.1781         192.168.0.4 192.168.0.2
192.168.0.1 192.168.0.3
   06:4b:8c:cd:c4:38   D        vtep.32779       192.168.0.3
   06:4b:8c:67:0f:f0   D        vtep.32781       192.168.0.1
   06:4b:8c:cd:13:f8   D        vtep.32782       192.168.0.2
   06:38:e1:6f:30:29   D        vtep.32783       192.168.0.4
---(more)---
```

9. Verify multihoming information of the gateway and the aggregated Ethernet interfaces.

   `user@leaf-1>` **show ethernet-switching vxlan-tunnel-end-point esi**

```
## Local AE link - QFX5100 leaf devices
ESI                           RTT                      VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
00:00:00:00:00:00:51:00:00:01 default-switch           1768  131078  esi.1768
 ae11.0,   2
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.2          vtep.32780     1782     1         2
    192.168.1.3          vtep.32772     1767     0         2
## Remote AE Link for QFX5110 leaf devices
ESI                           RTT                      VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
00:00:00:00:00:00:51:10:00:01 default-switch           1771  131081  esi.1771
           3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.6          vtep.32771     1766     2         2
    192.168.1.4          vtep.32770     1765     1         2
    192.168.1.5          vtep.32774     1770     0         2
## Remote AE Link for QFX5200 leaf devices
```

```
ESI                             RTT                     VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
00:00:00:00:00:00:52:00:00:01 default-switch           1774  131084  esi.1774
           3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.9          vtep.32778     1776     2         2
    192.168.1.8          vtep.32777     1775     1         2
    192.168.1.7          vtep.32776     1773     0         2
## Remote AE Link for QFX10002 leaf devices
ESI                             RTT                     VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
00:00:00:00:00:01:00:00:00:01 default-switch           1764  131074  esi.1764
           3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.11         vtep.32775     1772     2         2
    192.168.1.12         vtep.32773     1769     1         2
    192.168.1.10         vtep.32769     1759     0         2
## ESI multihoming to the VTEP for each segment
ESI                             RTT                     VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
05:19:17:f3:41:00:00:27:10:00 default-switch           1781  131091  esi.1781
           4
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.0.4          vtep.32783     1796     3         2
    192.168.0.2          vtep.32782     1793     2         2
    192.168.0.1          vtep.32781     1792     1         2
    192.168.0.3          vtep.32779     1777     0         2

...
```

10. Verify that the VXLAN tunnel from one leaf to another leaf is load balanced with equal cost multipathing (ECMP) over the underlay.

`user@leaf-1>` **show route forwarding-table table default-switch extensive | find vtep.32770**

```
Destination:  vtep.32770
  Route type: interface
  Route reference: 0                   Route interface-index: 576
  Multicast RPF nh index: 0
  P2mpidx: 0
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite             Index: 1765     Reference: 12
```

```
  Next-hop type: indirect              Index: 131076   Reference: 3
  Next-hop type: unilist               Index: 131193   Reference: 238
  Nexthop: 172.16.1.2
  Next-hop type: unicast               Index: 1791     Reference: 10
  Next-hop interface: ae1.0         Weight: 0x0
  Nexthop: 172.16.1.6
  Next-hop type: unicast               Index: 1794     Reference: 10
  Next-hop interface: ae2.0         Weight: 0x0
  Nexthop: 172.16.1.10
  Next-hop type: unicast               Index: 1758     Reference: 10
  Next-hop interface: ae3.0         Weight: 0x0
  Nexthop: 172.16.1.14
  Next-hop type: unicast               Index: 1795     Reference: 10
  Next-hop interface: ae4.0         Weight: 0x0
```

11. Verify that remote MAC addresses are reachable through ECMP.

`user@leaf-1>` **show route forwarding-table table default-switch extensive destination 02:0c:10:01:02:03/48**

```
Routing table: default-switch.evpn-vxlan [Index 4]
Bridging domain: VNI_10000.evpn-vxlan [Index 3]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:01:02:03/48
  Learn VLAN: 0                        Route type: user
  Route reference: 0                   Route interface-index: 582
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 169                  Epoch: 0
  Sequence Number: 0                   Learn Mask:
0x40000000000000000300000000000000000000
  L2 Flags: control_dyn
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite             Index: 1773     Reference: 12
  Next-hop type: indirect              Index: 131085   Reference: 3
  Next-hop type: unilist               Index: 131193   Reference: 238
  Nexthop: 172.16.1.2
  Next-hop type: unicast               Index: 1791     Reference: 10
  Next-hop interface: ae1.0            Weight: 0x0
  Nexthop: 172.16.1.6
  Next-hop type: unicast               Index: 1794     Reference: 10
```

```
  Next-hop interface: ae2.0            Weight: 0x0
  Nexthop: 172.16.1.10
  Next-hop type: unicast               Index: 1758     Reference: 10
  Next-hop interface: ae3.0            Weight: 0x0
  Nexthop: 172.16.1.14
  Next-hop type: unicast               Index: 1795     Reference: 10
  Next-hop interface: ae4.0            Weight: 0x0
```

> **NOTE:** Though the MAC address is reachable over multiple VTEP interfaces, QFX5100, QFX5110, QFX5120-32C, and QFX5200 switches do not support ECMP across the overlay because of a merchant ASIC limitation. Only the QFX10000 line of switches contain a custom Juniper Networks ASIC that supports ECMP across both the overlay and the underlay.

`user@leaf-1>` **show ethernet-switching table vlan-id 100 | match 02:0c:10:01:02:03**

```
   VNI_10000           02:0c:10:01:02:03   DR       esi.1774
00:00:00:00:00:00:52:00:00:01
```

`user@leaf-1>` **show route forwarding-table table default-switch extensive destination 02:0c:10:01:02:03/48**

```
Routing table: default-switch.evpn-vxlan [Index 9]
Bridging domain: VNI_10000.evpn-vxlan [Index 3]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:01:02:03/48
  Learn VLAN: 0                        Route type: user
  Route reference: 0                   Route interface-index: 550
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 0                    Epoch: 0
  Sequence Number: 0                   Learn Mask:
0x40000000000000000100000000000000000000
  L2 Flags: control_dyn, esi
  Flags: sent to PFE
  Next-hop type: indirect              Index: 2097173  Reference: 5
  Nexthop:
  Next-hop type: composite             Index: 1947     Reference: 2
  Nexthop:
```

```
  Next-hop type: composite              Index: 1948     Reference: 8
  Next-hop type: indirect               Index: 2097174  Reference: 3
  Next-hop type: unilist                Index: 2097280  Reference: 241
  Nexthop: 172.16.10.2
  Next-hop type: unicast                Index: 1950     Reference: 11
  Next-hop interface: ae1.0             Weight: 0x0
  Nexthop: 172.16.10.6
  Next-hop type: unicast                Index: 1956     Reference: 10
  Next-hop interface: ae2.0             Weight: 0x0
  Nexthop: 172.16.10.10
  Next-hop type: unicast                Index: 1861     Reference: 10
  Next-hop interface: ae3.0             Weight: 0x0
  Nexthop: 172.16.10.14
  Next-hop type: unicast                Index: 1960     Reference: 10
  Next-hop interface: ae4.0             Weight: 0x0
```

12. Verify which device is the Designated Forwarder (DF) for broadcast, unknown, and multicast (BUM) traffic coming from the VTEP tunnel.

> **NOTE:** Because the DF IP address is listed as 192.168.1.2, Leaf 2 is the DF.

`user@leaf-1>` **show evpn instance esi 00:00:00:00:00:00:51:00:00:01 designated-forwarder**

```
Instance: default-switch
  Number of ethernet segments: 12
    ESI: 00:00:00:00:00:00:51:00:00:01
      Designated forwarder: 192.168.1.2
```

SEE ALSO

- *Overview of VLAN Services for EVPN*
- *EVPN Overview*
- *Understanding VXLANs*
- *Understanding EVPN with VXLAN Data Plane Encapsulation*
- *Configuring EVPN Routing Instances*
- *Configuring Aggregated Ethernet LACP (CLI Procedure)*

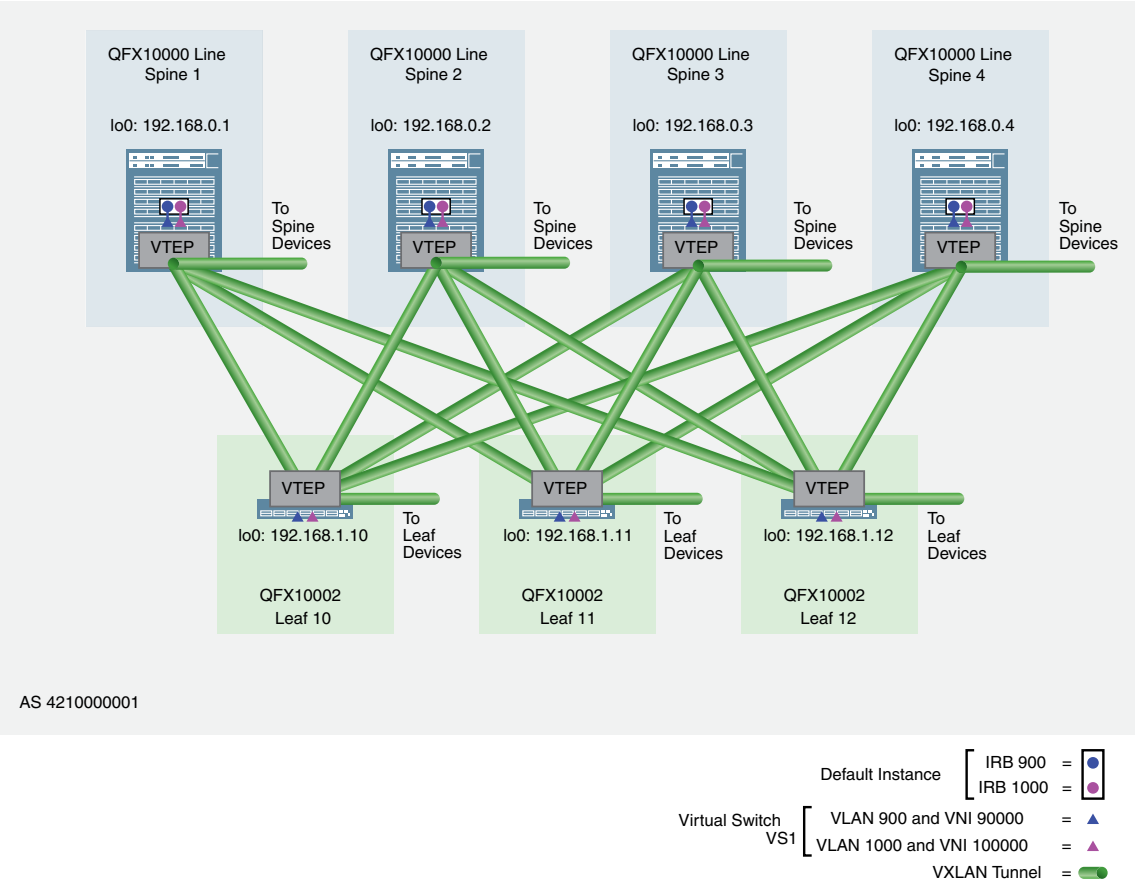
Configuring a VLAN-Aware Centrally-Routed Bridging Overlay with Virtual Switches

IN THIS SECTION

The second VLAN-aware centrally-routed bridging overlay model uses virtual switches, which enables you to configure multiple switching instances where each switching instance can support up to 4094 VLANs per instance.

The configuration method for VLANs (at the leaf devices) and IRB interfaces (at the spine devices) is similar to the default instance method for VLAN-aware centrally-routed bridging overlays. The main difference is that these elements are now configured inside virtual switching instances, as shown in Figure 36 on page 132.

Figure 36: VLAN-Aware Centrally-Routed Bridging Overlay — Virtual Switch Instance



When you implement this style of overlay on a spine device, you configure virtual gateways, virtual MAC addresses, and a virtual switch instance with the loopback interface as the VTEP, VXLAN encapsulation, VLAN to VNI mapping, and IRB interfaces (to provide routing between VLANs).

To implement this overlay style on a leaf device, you must use one of the QFX10000 line of switches. Preparing a leaf device to participate in this overlay type requires end system-facing elements (ESI, flexible VLAN tagging, extended VLAN bridge encapsulation, LACP settings, and VLAN IDs) and a virtual switch configuration (setting the loopback interface as the VTEP, configuring route distinguishers and targets, EVPN/VXLAN, and VLAN to VNI mapping).

For an overview of VLAN-aware centrally-routed bridging overlays, see the Centrally-Routed Bridging Overlay section in “Data Center Fabric Blueprint Architecture Components” on page 15.

NOTE:

- For a simpler method that works on all leaf platforms used in this reference design, see "Configuring a VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance" on page 96

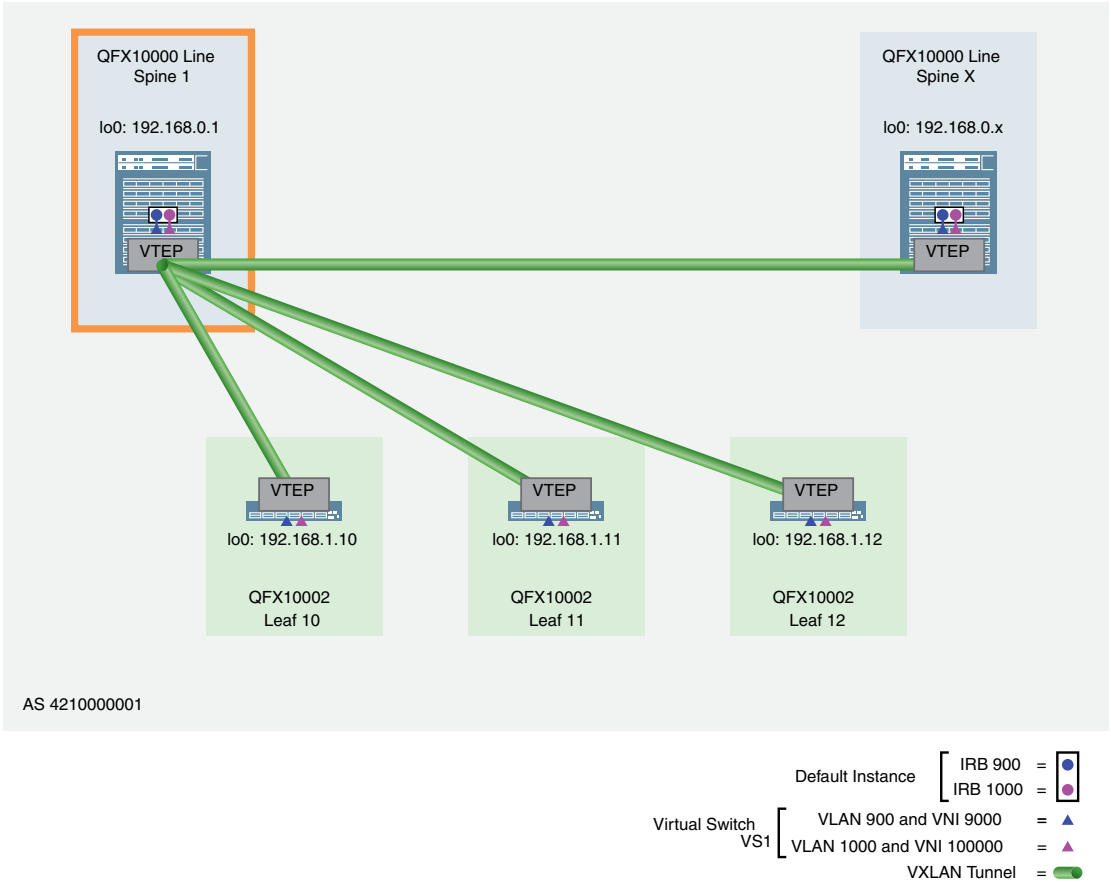
The following sections provide the detailed steps of how to configure and verify the VLAN-aware centrally-routed bridging overlay with virtual switches:

Configuring the VLAN-Aware Centrally-Routed Bridging Overlay with Virtual Switches on a Spine Device

To configure a VLAN-aware style of centrally-routed bridging overlay on a spine device, perform the following:

> **NOTE:** The following example shows the configuration for Spine 1, as shown in Figure 37 on page 134.

Figure 37: VLAN-Aware Centrally-Routed Bridging Overlay with Virtual Switches – Spine Device



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on spine devices, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine devices, see "Configuring IBGP for the Overlay" on page 67.

3. Configure a virtual switch instance for the VLAN-aware method. Include VTEP information, VXLAN encapsulation, VLAN to VNI mapping, IRB interfaces, and other instance details as part of the configuration.

   Spine 1:

```
set routing-instances VS1 vtep-source-interface lo0.0
set routing-instances VS1 instance-type virtual-switch
set routing-instances VS1 route-distinguisher 192.168.0.1:900
set routing-instances VS1 vrf-target target:62273:90000
set routing-instances VS1 vrf-target auto
set routing-instances VS1 protocols evpn encapsulation vxlan
set routing-instances VS1 protocols evpn extended-vni-list all
set routing-instances VS1 protocols evpn default-gateway no-gateway-community
set routing-instances VS1 vlans VNI_90000 vlan-id none
set routing-instances VS1 vlans VNI_90000 l3-interface irb.900
set routing-instances VS1 vlans VNI_90000 vxlan vni 90000
set routing-instances VS1 vlans VNI_100000 vlan-id none
set routing-instances VS1 vlans VNI_100000 l3-interface irb.1000
set routing-instances VS1 vlans VNI_100000 vxlan vni 100000
```

4. Configure spine devices for the VLAN-aware method. Include settings for the IPv4 and IPv6 virtual gateways and virtual MAC addresses. This example shows the configuration for Spine 1.

   Spine 1:

```
set interfaces irb unit 900 family inet address 10.1.8.1/24 virtual-gateway-address 10.1.8.254
set interfaces irb unit 900 family inet6 address 2001:db8::10:1:8:1/112 virtual-gateway-address
    2001:db8::10:1:8:254
set interfaces irb unit 900 family inet6 address fe80::10:1:8:1/112
set interfaces irb unit 900 virtual-gateway-v4-mac 00:00:5e:90:00:00
set interfaces irb unit 900 virtual-gateway-v6-mac 00:00:5e:90:00:00
set interfaces irb unit 1000 family inet address 10.1.9.1/24 virtual-gateway-address 10.1.9.254
set interfaces irb unit 1000 family inet6 address 2001:db8::10:1:9:1/112 virtual-gateway-address
    2001:db8::10:1:9:254
set interfaces irb unit 1000 family inet6 address fe80::10:1:9:1/112
set interfaces irb unit 1000 virtual-gateway-v4-mac 00:00:5e:a0:00:00
set interfaces irb unit 1000 virtual-gateway-v6-mac 00:00:5e:a0:00:00
```

Verifying the VLAN-Aware Model for a Centrally-Routed Bridging Overlay with Virtual Switches on a Spine Device

To verify this style of overlay on a spine device, perform the following:

1. Verify the IRB interfaces for VNIs 90000 and 100000 are operational for both IPv4 and IPv6.

   `user@spine-1>` **show interfaces terse irb | find irb\.900**

```
irb.900                 up    up   inet     10.1.8.1/24
                                            10.1.8.254/24
                                  inet6    2001:db8::10:1:8:1/112
                                           2001:db8::10:1:8:254/112
                                           fe80::10:1:8:1/112irb.1000
        up    up   inet     10.1.9.1/24
                                            10.1.9.254/24
                                  inet6    2001:db8::10:1:9:1/112
                                           2001:db8::10:1:9:254/112
                                           fe80::10:1:9:1/112
```

2. Verify switching details about the EVPN routing instance. This output includes information about the route distinguisher (192.168.1.10:900), VXLAN encapsulation, ESI (00:00:00:00:00:01:00:00:00:02), verification of the VXLAN tunnels for VLANs 900 and 1000, EVPN neighbors (Spine 2 - 4, and Leaf 10 - 12), and the source VTEP IP address (192.168.0.1).

   `user@spine-1>` **show evpn instance VS1 extensive**

```
Instance: VS1
  Route Distinguisher: 192.168.0.1:900
  Encapsulation type: VXLAN
  MAC database status                     Local  Remote
    MAC advertisements:                       2      14
    MAC+IP advertisements:                   14      26
    Default gateway MAC advertisements:       4       0
  Number of local interfaces: 0 (0 up)
  Number of IRB interfaces: 2 (2 up)
    Interface name  VLAN   VNI    Status  L3 context
    irb.1000               100000  Up     master
    irb.900                90000   Up     master
  Number of bridge domains: 2
    VLAN  Domain ID   Intfs / up    IRB intf   Mode            MAC sync  IM
route label  SG sync  IM core nexthop
    8191  90000          0    0     irb.900    Extended        Enabled   90000
           Disabled
    8191  100000         0    0     irb.1000   Extended        Enabled   100000
```

```
        Disabled
Number of neighbors: 6
  Address               MAC    MAC+IP        AD        IM        ES Leaf-label

  192.168.0.2             4        10         2         2         0
  192.168.0.3             4        10         2         2         0
  192.168.0.4             4        10         2         2         0
  192.168.1.10            1         2         2         2         0
  192.168.1.11            0         0         2         2         0
  192.168.1.12            1         2         2         2         0
Number of ethernet segments: 3
  ESI: 00:00:00:00:00:01:00:00:00:02
    Status: Resolved
    Number of remote PEs connected: 3
      Remote PE        MAC label  Aliasing label  Mode
      192.168.1.12     90000      0               all-active
      192.168.1.11     90000      0               all-active
      192.168.1.10     100000     0               all-active
  ESI: 05:19:17:f3:41:00:01:5f:90:00
    Local interface: irb.900, Status: Up/Forwarding
    Number of remote PEs connected: 3
      Remote PE        MAC label  Aliasing label  Mode
      192.168.0.3      90000      0               all-active
      192.168.0.2      90000      0               all-active
      192.168.0.4      90000      0               all-active
  ESI: 05:19:17:f3:41:00:01:86:a0:00
    Local interface: irb.1000, Status: Up/Forwarding
    Number of remote PEs connected: 3
      Remote PE        MAC label  Aliasing label  Mode
      192.168.0.3      100000     0               all-active
      192.168.0.2      100000     0               all-active
      192.168.0.4      100000     0               all-active
Router-ID: 192.168.0.1
Source VTEP interface IP: 192.168.0.1
```

3. Verify the MAC address table on the leaf device.

NOTE:

- 00:00:5e:90:00:00 and 00:00:5e:a0:00:00 are the IP subnet gateways on the spine device.
- 02:0c:10:09:02:01 and 02:0c:10:08:02:01 are end systems connected through the leaf device.

user@spine-1> **show ethernet-switching table instance VS1**

```
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent
 static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC,
 O - ovsdb MAC)


Ethernet switching table : 5 entries, 5 learned
Routing instance : VS1
   Vlan                MAC                 MAC      Logical              Active

   name                address             flags    interface            source

   VNI_100000          00:00:5e:a0:00:00   DR       esi.2454
05:19:17:f3:41:00:01:86:a0:00
   VNI_100000          06:4b:8c:cd:13:f8   D        vtep.32773
192.168.0.2
   VNI_100000          06:4b:8c:cd:c4:38   D        vtep.32787
192.168.0.3
   VNI_100000          02:0c:10:09:02:01   DR       esi.2467
00:00:00:00:00:01:00:00:00:02
   VNI_100000          06:38:e1:6f:30:29   D        vtep.32796
192.168.0.4

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent
 static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC,
 O - ovsdb MAC)


Ethernet switching table : 5 entries, 5 learned
Routing instance : VS1
   Vlan                MAC                 MAC      Logical              Active

   name                address             flags    interface            source

   VNI_90000           00:00:5e:90:00:00   DR       esi.2455
05:19:17:f3:41:00:01:5f:90:00
   VNI_90000           06:4b:8c:cd:13:f8   D        vtep.32773
192.168.0.2
   VNI_90000           06:4b:8c:cd:c4:38   D        vtep.32787
192.168.0.3
   VNI_90000           02:0c:10:08:02:01   DR       esi.2467
```

```
00:00:00:00:00:01:00:00:00:02
   VNI_90000           06:38:e1:6f:30:29   D        vtep.32796
192.168.0.4
```

4. Verify the end system MAC address is reachable from all three leaf devices.

   `user@spine-1>` **show ethernet-switching vxlan-tunnel-end-point esi | find esi.2467**

```
00:00:00:00:00:01:00:00:00:02 VS1                     2467  2097182 esi.2467
          3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.10         vtep.32789     2522     2         2
    192.168.1.11         vtep.32782     2475     1         2
    192.168.1.12         vtep.32779     2466     0         2
ESI                           RTT                     VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
```

5. Verify the end system is reachable through the forwarding table.

   `user@spine-1>` **show route forwarding-table table VS1 destination 02:0c:10:09:02:01/48 extensive**

```
Routing table: VS1.evpn-vxlan [Index 11]
Bridging domain: VNI_100000.evpn-vxlan [Index 9]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:09:02:01/48
  Learn VLAN: 0                       Route type: user
  Route reference: 0                  Route interface-index: 676
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 0                   Epoch: 0
  Sequence Number: 0                  Learn Mask:
0x40000000000000000600000000000000000000
  L2 Flags: control_dyn, esi
  Flags: sent to PFE
  Next-hop type: indirect             Index: 2097182  Reference: 3
  Nexthop:
  Next-hop type: composite            Index: 2467     Reference: 2
  Nexthop:
  Next-hop type: composite            Index: 2522     Reference: 6
  Next-hop type: indirect             Index: 2097154  Reference: 5
  Nexthop: 172.16.10.1
  Next-hop type: unicast              Index: 2172     Reference: 11
```

```
  Next-hop interface: ae10.0
  Nexthop:
  Next-hop type: composite             Index: 2475     Reference: 6
  Next-hop type: indirect              Index: 2097193  Reference: 5
  Nexthop: 172.16.11.1
  Next-hop type: unicast               Index: 2194     Reference: 11
  Next-hop interface: ae11.0
  Nexthop:
  Next-hop type: composite             Index: 2466     Reference: 6
  Next-hop type: indirect              Index: 2097195  Reference: 5
  Nexthop: 172.16.12.1
  Next-hop type: unicast               Index: 2916     Reference: 11
  Next-hop interface: ae12.0
```

6. Verify end system information (MAC address, IP address, etc.) has been added to the IPv4 ARP table and IPv6 neighbor table.

   `user@spine-1>` **show arp no-resolve expiration-time | match "irb.900|irb.1000"**

```
06:4b:8c:cd:13:f8 10.1.8.2        irb.900 [vtep.32773]     none 1035
06:4b:8c:cd:c4:38 10.1.8.3        irb.900 [vtep.32787]     none 1064
06:38:e1:6f:30:29 10.1.8.4        irb.900 [vtep.32796]     none 964
02:0c:10:08:02:01 10.1.8.201      irb.900 [.local..11]     none 781
06:4b:8c:cd:13:f8 10.1.9.2        irb.1000 [vtep.32773]    none 910
06:4b:8c:cd:c4:38 10.1.9.3        irb.1000 [vtep.32787]    none 1344
06:38:e1:6f:30:29 10.1.9.4        irb.1000 [vtep.32796]    none 1160
02:0c:10:09:02:01 10.1.9.201      irb.1000 [.local..11]    none 1099
```

   `user@spine-1>` **show ipv6 neighbors | match "irb.900|irb.1000"**

```
2001:db8::10:1:8:2           06:4b:8c:cd:13:f8  stale       969 yes no
irb.900 [vtep.32773]
2001:db8::10:1:8:3           06:4b:8c:cd:c4:38  stale       1001 yes no
irb.900 [vtep.32787]
2001:db8::10:1:8:4           06:38:e1:6f:30:29  stale       971 yes no
irb.900 [vtep.32796]
2001:db8::10:1:8:201         02:0c:10:08:02:01  stale       1178 no no
irb.900 [.local..11]
2001:db8::10:1:9:2           06:4b:8c:cd:13:f8  stale       955 yes no
irb.1000 [vtep.32773]
2001:db8::10:1:9:3           06:4b:8c:cd:c4:38  stale       1006 yes no
irb.1000 [vtep.32787]
2001:db8::10:1:9:4           06:38:e1:6f:30:29  stale       990 yes no
irb.1000 [vtep.32796]
```

```
2001:db8::10:1:9:201         02:0c:10:09:02:01  stale       1199 no no
irb.1000 [.local..11]
fe80::10:1:8:2               06:4b:8c:cd:13:f8  stale       991 yes no
irb.900 [vtep.32773]
fe80::10:1:8:3               06:4b:8c:cd:c4:38  stale       989 yes no
irb.900 [vtep.32787]
fe80::10:1:8:4               06:38:e1:6f:30:29  stale       966 yes no
irb.900 [vtep.32796]
fe80::10:1:9:2               06:4b:8c:cd:13:f8  stale       978 yes no
irb.1000 [vtep.32773]
fe80::10:1:9:3               06:4b:8c:cd:c4:38  stale       994 yes no
irb.1000 [vtep.32787]
fe80::10:1:9:4               06:38:e1:6f:30:29  stale       1006 yes no
irb.1000 [vtep.32796]
```

7. Verify that the EVPN database contains the MAC address (02:0c:10:08:02:01) and ARP information learned from an end system connected to the leaf device.

   `user@spine-1>` **show evpn database mac-address 02:0c:10:08:02:01 extensive**

```
Instance: VS1

VN Identifier: 90000, MAC address:: 02:0c:10:08:02:01
  Source: 00:00:00:00:00:01:00:00:00:02, Rank: 1, Status: Active
    Remote origin: 192.168.1.10
    Remote origin: 192.168.1.11
    Remote origin: 192.168.1.12
    Timestamp: Sep 10 23:47:37 (0x59b63189)
    State: <Remote-To-Local-Adv-Done>
    IP address: 10.1.8.201
      Flags: <Proxy>
      Remote origin: 192.168.1.10
      Remote origin: 192.168.1.11
      Remote origin: 192.168.1.12
    IP address: 2001:db8::10:1:8:201
      Remote origin: 192.168.1.10
      Remote origin: 192.168.1.11
      Remote origin: 192.168.1.12    History db:
      Time                 Event
      Sep 10 23:47:39 2017  Applying remote state to peer 192.168.1.11
      Sep 10 23:47:39 2017  Remote peer 192.168.1.11 updated
     Sep 10 23:47:39 2017  MAC+IP not updated, source l2ald is not owner (type2)

      Sep 10 23:47:39 2017  Updated
```

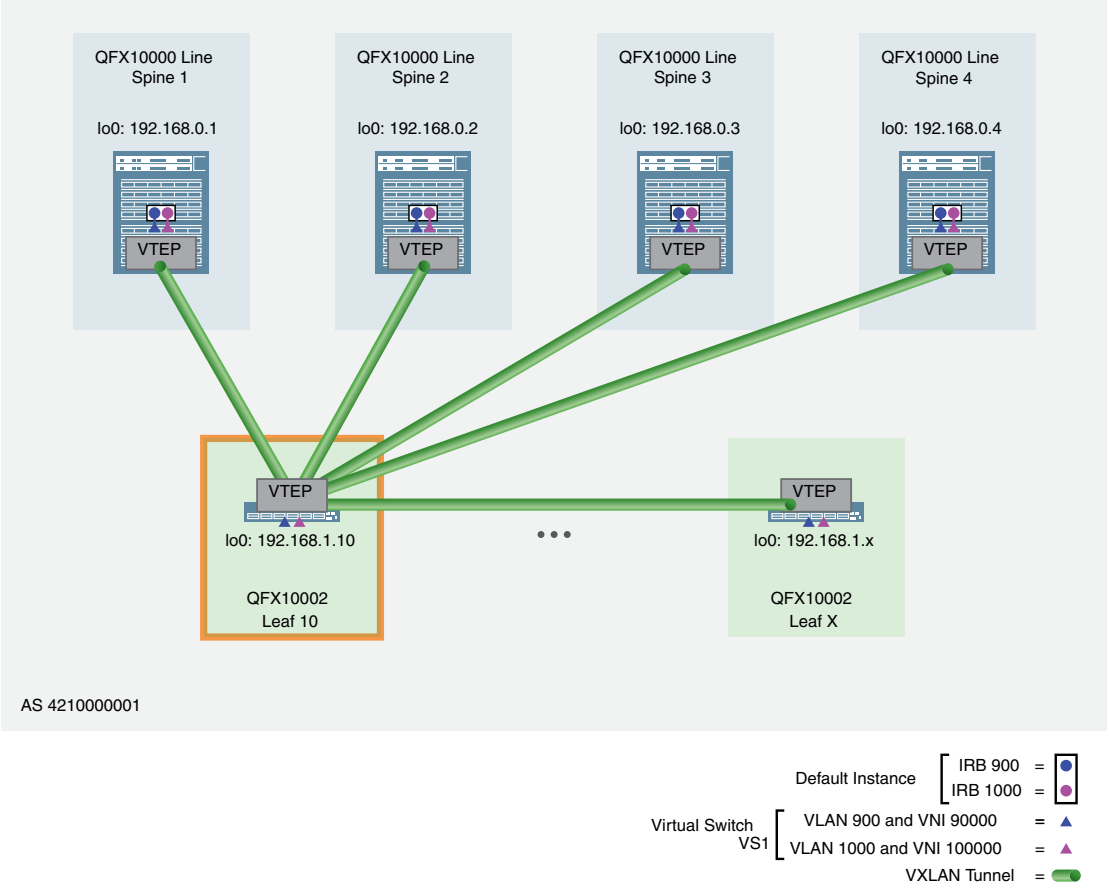
```
 Sep 10 23:47:39 2017  No change to MAC state
 Sep 10 23:47:39 2017  Applying remote state to peer 192.168.1.12
 Sep 10 23:47:39 2017  Remote peer 192.168.1.12 updated
Sep 10 23:47:39 2017  MAC+IP not updated, source l2ald is not owner (type2)

 Sep 10 23:47:39 2017  Updated
 Sep 10 23:47:39 2017  No change to MAC state
```

Configuring the VLAN-Aware Centrally-Routed Bridging Overlay with Virtual Switches on a Leaf Device

To configure a VLAN-aware centrally-routed bridging overlay in a virtual switch on a leaf device, perform the following:

> **NOTE:** The following example shows the configuration for Leaf 10, as shown in Figure 38 on page 143.

Figure 38: VLAN-Aware Centrally-Routed Bridging Overlay with Virtual Switches – Leaf Device



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on leaf devices, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your leaf devices,

see "Configuring IBGP for the Overlay" on page 67.

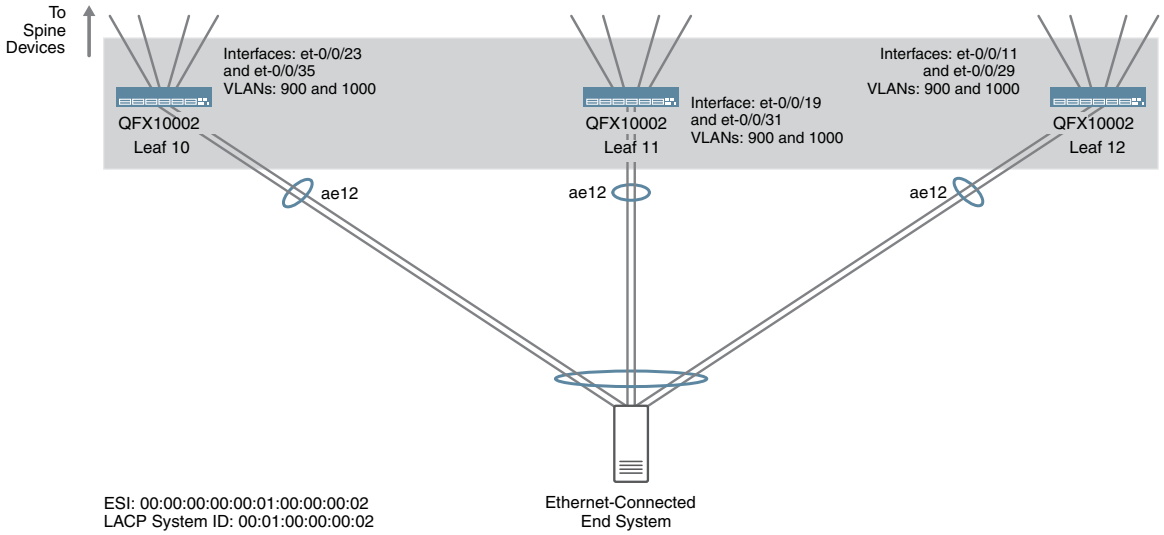
3. Configure a virtual switch instance VS1 to enable EVPN/VXLAN and map VLANs 900 and 1000 to VNIs 90000 and 100000.

   Leaf 10:

```
set routing-instances VS1 vtep-source-interface lo0.0
set routing-instances VS1 instance-type virtual-switch
set routing-instances VS1 route-distinguisher 192.168.1.10:900
set routing-instances VS1 vrf-target target:62273:90000
set routing-instances VS1 vrf-target auto
set routing-instances VS1 protocols evpn encapsulation vxlan
set routing-instances VS1 protocols evpn extended-vni-list all
set routing-instances VS1 protocols evpn default-gateway no-gateway-community
set routing-instances VS1 vlans VNI_90000 interface ae12.900
set routing-instances VS1 vlans VNI_90000 vxlan vni 90000
set routing-instances VS1 vlans VNI_100000 interface ae12.1000
set routing-instances VS1 vlans VNI_100000 vxlan vni 100000
```

4. Configure the leaf device to communicate with the end system. In this example, configure Leaf 10 with LACP options, an all active ESI, and VLANs 900 and 1000 (which are reserved in this example for the VLAN-aware virtual switch method). An illustration of the topology is shown in Figure 39 on page 144.

Figure 39: ESI Topology for Leaf 10, Leaf 11, and Leaf 12



Leaf 10:

```
set interfaces ae12 flexible-vlan-tagging
set interfaces ae12 encapsulation extended-vlan-bridge
set interfaces ae12 esi 00:00:00:00:00:01:00:00:00:02
set interfaces ae12 esi all-active
set interfaces ae12 aggregated-ether-options lacp active
set interfaces ae12 aggregated-ether-options lacp periodic fast
set interfaces ae12 aggregated-ether-options lacp system-id 00:01:00:00:00:02
set interfaces ae12 unit 900 vlan-id 900
set interfaces ae12 unit 1000 vlan-id 1000
set interfaces et-0/0/23 ether-options 802.3ad ae12
set interfaces et-0/0/35 ether-options 802.3ad ae12
```

Verifying the VLAN-Aware Centrally-Routed Bridging Overlay with Virtual Switches on a Leaf Device

To verify this style of overlay on a leaf device, perform the following:

1. Verify that the aggregated Ethernet interface is operational on the leaf device.

   `user@leaf-10>` **show interfaces terse ae12**

```
Interface               Admin Link Proto    Local                 Remote
ae12                    up    up
ae12.900                up    up   eth-switch
ae12.1000               up    up   eth-switch
ae12.32767              up    up
```

2. Verify switching details about the EVPN routing instance. This output includes information about the route distinguisher (192.168.1.10:900), VXLAN encapsulation, ESI (00:00:00:00:00:01:00:00:00:02), verification of the VXLAN tunnels for VLANs 900 and 1000, EVPN neighbors (Spine 1 - 4, and Leaf 11 and 12), and the source VTEP IP address (192.168.1.10).

   `user@leaf-10>` **show evpn instance VS1 extensive**

```
Instance: VS1
  Route Distinguisher: 192.168.1.10:900
  Encapsulation type: VXLAN
  MAC database status                    Local  Remote
    MAC advertisements:                      0      20
    MAC+IP advertisements:                   0      32
    Default gateway MAC advertisements:      0       0
  Number of local interfaces: 2 (2 up)
    Interface name  ESI                            Mode             Status
```

```
 AC-Role
    ae12.1000       00:00:00:00:00:01:00:00:00:02  all-active       Up
 Root
    ae12.900        00:00:00:00:00:01:00:00:00:02  all-active       Up
 Root
  Number of IRB interfaces: 0 (0 up)
  Number of bridge domains: 2
    VLAN  Domain ID   Intfs / up    IRB intf   Mode             MAC sync  IM
route label  SG sync  IM core nexthop
    None  90000          1    1               Extended         Enabled   90000
           Disabled
    None  100000         1    1               Extended         Enabled   100000
          Disabled
  Number of neighbors: 6
    Address              MAC    MAC+IP        AD        IM        ES Leaf-label

    192.168.0.1            4        10         2         2         0
    192.168.0.2            4        10         2         2         0
    192.168.0.3            4        10         2         2         0
    192.168.0.4            4        10         2         2         0
    192.168.1.11           2         4         2         2         0
    192.168.1.12           2         4         2         2         0
  Number of ethernet segments: 3
    ESI: 00:00:00:00:00:01:00:00:00:02
      Status: Resolved by IFL ae12.900
      Local interface: ae12.1000, Status: Up/Forwarding
      Number of remote PEs connected: 2
        Remote PE        MAC label  Aliasing label  Mode
        192.168.1.12     100000     0               all-active
        192.168.1.11     90000      0               all-active
      DF Election Algorithm: MOD based
      Designated forwarder: 192.168.1.10
      Backup forwarder: 192.168.1.11
      Backup forwarder: 192.168.1.12
      Last designated forwarder update: Sep 10 23:22:07
    ESI: 05:19:17:f3:41:00:01:5f:90:00
      Status: Resolved
      Number of remote PEs connected: 4
        Remote PE        MAC label  Aliasing label  Mode
        192.168.0.1      90000      0               all-active
        192.168.0.3      90000      0               all-active
        192.168.0.2      90000      0               all-active
        192.168.0.4      90000      0               all-active
    ESI: 05:19:17:f3:41:00:01:86:a0:00
```

```
      Status: Resolved
      Number of remote PEs connected: 4
        Remote PE        MAC label  Aliasing label  Mode
        192.168.0.1      100000     0               all-active
        192.168.0.3      100000     0               all-active
        192.168.0.2      100000     0               all-active
        192.168.0.4      100000     0               all-active
  Router-ID: 192.168.1.10
  Source VTEP interface IP: 192.168.1.10
```

3. View the MAC address table on the leaf device to confirm that spine device and end system MAC addresses appear in the table.

> **NOTE:**
> - 00:00:5e:90:00:00 and 00:00:5e:a0:00:00 are the IP subnet gateways on the spine device.
> - 02:0c:10:09:02:01 and 02:0c:10:08:02:01 are end systems connected through the leaf device.

`user@leaf-10>` **show ethernet-switching table instance VS1**

```
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent
 static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC,
 O - ovsdb MAC)


Ethernet switching table : 6 entries, 6 learned
Routing instance : VS1
   Vlan                MAC                 MAC      Logical              Active

   name                address             flags    interface            source

   VNI_100000          00:00:5e:a0:00:00   DR       esi.2139
05:19:17:f3:41:00:01:86:a0:00
   VNI_100000          06:4b:8c:67:0f:f0   D        vtep.32799
192.168.0.1
   VNI_100000          06:4b:8c:cd:13:f8   D        vtep.32798
192.168.0.2
   VNI_100000          06:4b:8c:cd:c4:38   D        vtep.32804
192.168.0.3
```

```
   VNI_100000          02:0c:10:09:02:01   DR       ae12.1000
   VNI_100000          06:38:e1:6f:30:29   D        vtep.32807
192.168.0.4

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent
 static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC,
 O - ovsdb MAC)


Ethernet switching table : 6 entries, 6 learned
Routing instance : VS1
   Vlan                MAC                 MAC      Logical              Active

   name                address             flags    interface            source

   VNI_90000           00:00:5e:90:00:00   DR       esi.2144
05:19:17:f3:41:00:01:5f:90:00
   VNI_90000           06:4b:8c:67:0f:f0   D        vtep.32799
192.168.0.1
   VNI_90000           06:4b:8c:cd:13:f8   D        vtep.32798
192.168.0.2
   VNI_90000           06:4b:8c:cd:c4:38   D        vtep.32804
192.168.0.3
   VNI_90000           02:0c:10:08:02:01   DR       ae12.900
   VNI_90000           06:38:e1:6f:30:29   D        vtep.32807
192.168.0.4
```

4. Verify that the IP subnet gateway ESIs discovered in Step 3 (esi.2144 for VNI 90000 and esi.2139 for VNI 100000) are reachable from all four spine devices.

   `user@leaf-10>` **show ethernet-switching vxlan-tunnel-end-point esi | find esi.2144**

```
05:19:17:f3:41:00:01:5f:90:00 VS1                    2144  2097224 esi.2144
           4
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.0.4          vtep.32807     2033     3         2
    192.168.0.2          vtep.32798     2092     2         2
    192.168.0.3          vtep.32804     2174     1         2
    192.168.0.1          vtep.32799     2093     0         2
ESI                           RTT                    VLNBH INH     ESI-IFL
 LOC-IFL   #RVTEPs
```

   `user@leaf-10>` **show ethernet-switching vxlan-tunnel-end-point esi | find esi.2139**

```
05:19:17:f3:41:00:01:86:a0:00 VS1                    2139  2097221 esi.2139
          4
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.0.4          vtep.32807     2033     3         2
    192.168.0.2          vtep.32798     2092     2         2
    192.168.0.3          vtep.32804     2174     1         2
    192.168.0.1          vtep.32799     2093     0         2
```

5. Verify the IP subnet gateway on the spine device (00:00:5e:a0:00:00) is reachable through the forwarding table.

   `user@leaf-10>` **show route forwarding-table table VS1 destination 00:00:5e:a0:00:00/48 extensive**

```
Routing table: VS1.evpn-vxlan [Index 10]
Bridging domain: VNI_100000.evpn-vxlan [Index 15]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  00:00:5e:a0:00:00/48
  Learn VLAN: 0                       Route type: user
  Route reference: 0                  Route interface-index: 571
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 0                   Epoch: 0
  Sequence Number: 0                  Learn Mask:
0x400000000000000000f00000000000000000000
  L2 Flags: control_dyn, esi
  Flags: sent to PFE
  Next-hop type: indirect             Index: 2097221  Reference: 2
  Nexthop:
  Next-hop type: composite            Index: 2139     Reference: 2
  Nexthop:
  Next-hop type: composite            Index: 2033     Reference: 9
  Next-hop type: indirect             Index: 2097223  Reference: 5
  Nexthop: 172.16.10.14
  Next-hop type: unicast              Index: 2106     Reference: 10
  Next-hop interface: ae4.0
  Nexthop:
  Next-hop type: composite            Index: 2092     Reference: 9
  Next-hop type: indirect             Index: 2097172  Reference: 5
  Nexthop: 172.16.10.6
  Next-hop type: unicast              Index: 1951     Reference: 11
  Next-hop interface: ae2.0
  Nexthop:
```

```
Next-hop type: composite            Index: 2174     Reference: 9
Next-hop type: indirect             Index: 2097174  Reference: 5
Nexthop: 172.16.10.10
Next-hop type: unicast              Index: 2143     Reference: 11
Next-hop interface: ae3.0
Nexthop:
Next-hop type: composite            Index: 2093     Reference: 9
Next-hop type: indirect             Index: 2097165  Reference: 5
Nexthop: 172.16.10.2
Next-hop type: unicast              Index: 2153     Reference: 11
Next-hop interface: ae1.0
```

SEE ALSO

- *Overview of VLAN Services for EVPN*
- *EVPN Overview*
- *Understanding VXLANs*
- *Understanding EVPN with VXLAN Data Plane Encapsulation*
- *Configuring EVPN Routing Instances*
- *Configuring Aggregated Ethernet LACP (CLI Procedure)*

Centrally-Routed Bridging Overlay — Release History

Table 4 on page 150 provides a history of all of the features in this section and their support within this reference design.

Table 4: Centrally-Routed Bridging Overlay in the Data Center Fabric Reference Design– Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section.
17.3R3-S2	Adds support for Contrail Enterprise Multicloud, where you can configure centrally-routed bridging overlays from the Contrail Command GUI.
17.3R3-S1	All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section

RELATED DOCUMENTATION

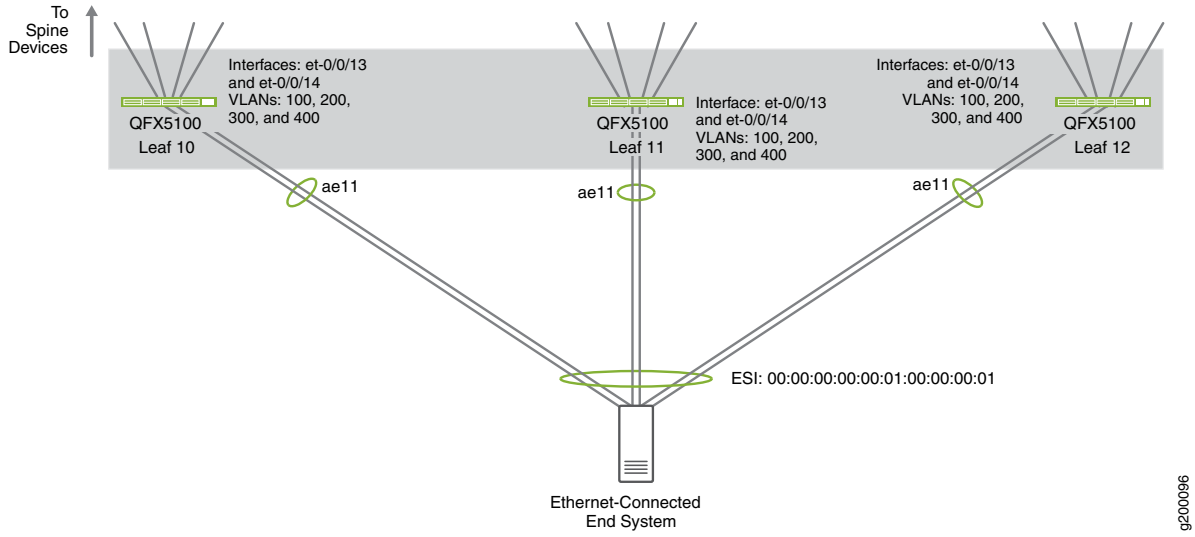
EVPN User Guide

Multihoming an Ethernet-Connected End System Design and Implementation

For an overview of multihoming an Ethernet-connected end system in this reference design, see the Multihoming Support for Ethernet-Connected End Systems section in "Data Center Fabric Blueprint Architecture Components" on page 15.

Figure 40 on page 151 illustrates the multihomed Ethernet-connected end system in this procedure:

Figure 40: Ethernet-Connected Multihoming Example Overview



Configuring a Multihomed Ethernet-Connected End System using EVPN Multihoming with VLAN Trunking

EVPN multihoming is used in this building block to connect an Ethernet-connected end system into the overlay network. EVPN multihoming works by treating two or more physical multihomed links as a single Ethernet segment identified by an EVPN Ethernet Segment ID (ESI). The set of physical links belonging to the same Ethernet segment are treated as one aggregated Ethernet interface. The member links—much like member links in a traditional aggregated Ethernet interface—provide redundant paths to and from the end system while also ensuring overlay network traffic is load-balanced across the multiple paths.

LACP with the fast timer mode is used to improve fault detection and disablement of impaired members of an Ethernet segment. MicroBFD may also be used to further improve fault isolation but may not scale to support all end-system facing ports. Furthermore, support for microBFD must exist at the end system.

The reference design tested an Ethernet-connected server was connected to a single leaf or multihomed to 2 or 3 leaf devices to verify that traffic can be properly handled in multihomed setups with more than 2 leaf devices; in practice, an Ethernet-connected server can be multihomed to a large number of leaf devices.

To configure a multihomed Ethernet-connected server:

1. (Aggregated Ethernet interfaces only) Create the aggregated Ethernet interfaces to connect each leaf device to the server. Enable LACP with a fast period interval for each aggregated Ethernet interface.

   Leaf 10:

   ```
   set interfaces et-0/0/13 ether-options 802.3ad ae11
   set interfaces et-0/0/14 ether-options 802.3ad ae11
   set interfaces ae11 aggregated-ether-options lacp active
   set interfaces ae11 aggregated-ether-options lacp periodic fast
   ```

   Leaf 11:

   ```
   set interfaces et-0/0/13 ether-options 802.3ad ae11
   set interfaces et-0/0/14 ether-options 802.3ad ae11
   set interfaces ae11 aggregated-ether-options lacp active
   set interfaces ae11 aggregated-ether-options lacp periodic fast
   ```

   Leaf 12:

   ```
   set interfaces et-0/0/13 ether-options 802.3ad ae11
   set interfaces et-0/0/14 ether-options 802.3ad ae11
   set interfaces ae11 aggregated-ether-options lacp active
   set interfaces ae11 aggregated-ether-options lacp periodic fast
   ```

   NOTE: The three leaf devices in this step use the same aggregated Ethernet interface name—ae11—and member link interfaces—et-0/0/13 and et-0/0/14— to organize and simplify network administration.

   Avoid using different AE names at each VTEP for the same ESI as this will require configuring the LACP admin-key so that the end system can identify the multihomed links as part of the same LAG.

2. Configure each interface into a trunk interface. Assign VLANs to each trunk interface.

> **NOTE:** If you are connecting your end system to the leaf device with a single link, replace the interface name—for example, *ae11*—with a physical interface name—for example, *et-0/0/13*—for the remainder of this procedure.

Leaf 10:

```
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members 100
set interfaces ae11 unit 0 family ethernet-switching vlan members 200
set interfaces ae11 unit 0 family ethernet-switching vlan members 300
set interfaces ae11 unit 0 family ethernet-switching vlan members 400
```

Leaf 11:

```
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members 100
set interfaces ae11 unit 0 family ethernet-switching vlan members 200
set interfaces ae11 unit 0 family ethernet-switching vlan members 300
set interfaces ae11 unit 0 family ethernet-switching vlan members 400
```

Leaf 12:

```
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members 100
set interfaces ae11 unit 0 family ethernet-switching vlan members 200
set interfaces ae11 unit 0 family ethernet-switching vlan members 300
set interfaces ae11 unit 0 family ethernet-switching vlan members 400
```

3. Configure the multihomed links with an ESI and specify an LACP system identifier for each link.

   Assign each multihomed interface into the ethernet segment—which is identified using the Ethernet Segment Identifier (ESI)—that is hosting the Ethernet-connected server. Ensure traffic is passed over all multihomed links by configuring each link as *all-active*.

   The ESI values must match on all multihomed interfaces.

   Leaf 10 :

```
set interfaces ae11 esi 00:00:00:00:00:01:00:00:00:01
set interfaces ae11 esi all-active
```

Leaf 11:

```
set interfaces ae11 esi 00:00:00:00:00:01:00:00:00:01
set interfaces ae11 esi all-active
```

Leaf 12:

```
set interfaces ae11 esi 00:00:00:00:00:01:00:00:00:01
set interfaces ae11 esi all-active
```

4. Enable LACP and configure a system identifier.

   The LACP system identifier must match on all multihomed interfaces.

   Leaf 10:

   ```
   set interfaces ae11 aggregated-ether-options lacp system-id 00:01:00:00:00:01
   ```

   Leaf 11:

   ```
   set interfaces ae11 aggregated-ether-options lacp system-id 00:01:00:00:00:01
   ```

   Leaf 12:

   ```
   set interfaces ae11 aggregated-ether-options lacp system-id 00:01:00:00:00:01
   ```

5. After committing the configuration, verify that the links on each leaf switch are in the *Up* state

   Example:

   ```
   user@leaf10# run show interfaces terse ae11
   Interface               Admin Link Proto    Local                 Remote
   ```

```
ae11                    up    up
ae11.0                  up    up   eth-switch
```

6. Verify that LACP is operational on the multihomed links.

```
user@leaf10# run show lacp interfaces ae11
Aggregated interface: ae11
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity

      et-0/0/13      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active

      et-0/0/13    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active

      et-0/0/14      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active

      et-0/0/14    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active

    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/13                Current   Fast periodic Collecting distributing

      et-0/0/14                Current   Fast periodic Collecting distributing
```

Enabling Storm Control

Storm control can be enabled as part of this building block. Storm control is used to prevent BUM traffic storms by monitoring BUM traffic levels and taking a specified action to limit BUM traffic forwarding when a specified traffic level—called the storm control level—is exceeded. See Understanding Storm Control for additional information on the feature.

In this reference design, storm control is enabled on server-facing aggregated Ethernet interfaces to rate limit broadcast, unknown unicast, and multicast (BUM) traffic. If the amount of BUM traffic exceeds 1% of the available bandwidth on the aggregated Ethernet interface, storm control drops BUM traffic to prevent broadcast storms.

To enable storm control:

1. Create the storm control profile that will be used to enable the feature. The interfaces that are configured using the storm control profile are specified in this step.

   Leaf Device:

```
set interfaces ae11 unit 0 family ethernet-switching storm-control STORM-CONTROL
```

2. Set the storm control configuration within the profile.

   In this reference design, storm control is configured to strategically drop BUM traffic when the amount of BUM traffic exceeds 1% of all available interface bandwidth.

```
set forwarding-options storm-control-profiles STORM-CONTROL all bandwidth-percentage 1
```

> **NOTE:** Dropping BUM traffic is the only supported storm control action in the Cloud Data Center architecture.

> **NOTE:** The storm control settings in this version of the reference design drop multicast traffic that exceeds the configured storm control threshold. If your network supports multicast-based applications, consider using a storm control configuration—such as the **no-multicast** option in the **storm-control-profiles** statement—that is not represented in this reference design.
>
> Storm control settings in support of multicast-based applications will be included in a future version of this reference design.

3. To verify storm control activity, filter system log messages related to storm control by entering the **show log messages | match storm** command.

```
user@leaf10> show log messages | match storm
Sep 27 11:35:34 leaf1-qfx5100 l2ald[1923]: L2ALD_ST_CTL_IN_EFFECT: ae11.0: storm
 control in effect on the port
```

Multihoming a Ethernet-Connected End System—Release History

Table 5 on page 158 provides a history of all of the features in this section and their support within this reference design.

Table 5: Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section.
18.1R3-S3	QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section.
17.3R3-S1	All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section.

RELATED DOCUMENTATION

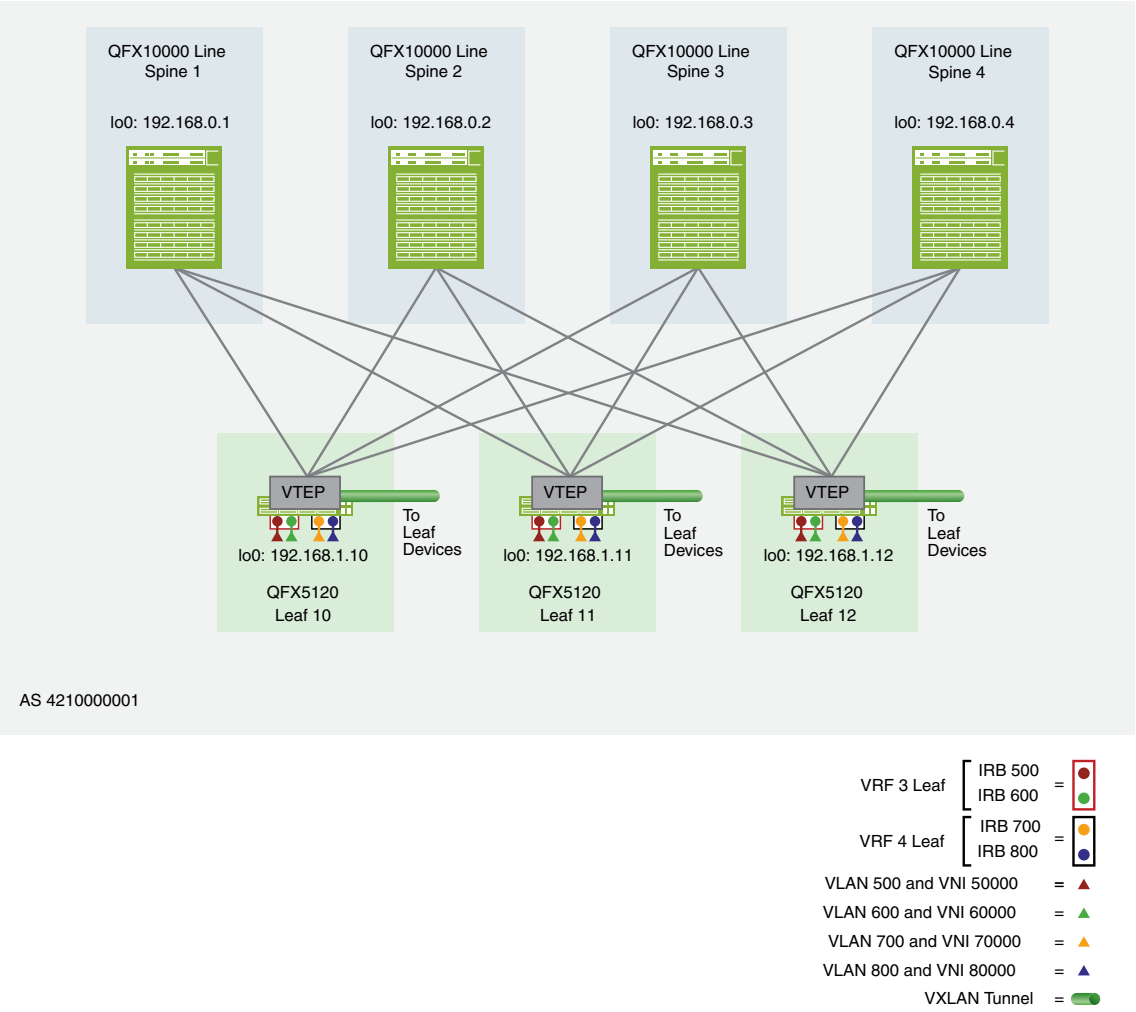
EVPN Multihoming Overview

Edge-Routed Bridging Overlay Design and Implementation

IN THIS SECTION

A second overlay option for this reference design is the edge-routed bridging overlay, as shown in Figure 41 on page 159.

Figure 41: Edge-Routed Bridging Overlay



The edge-routed bridging overlay performs routing at IRB interfaces located at the edge of the overlay (most often at the leaf devices). As a result, Ethernet bridging and IP routing happen as close to the end systems as possible, but still support Ethernet dependent applications at the end system level.

For a list of switches that we support as lean spine and leaf devices in an edge-routed bridging overlay, see the Table 1 on page 46.

Lean spine devices handle only IP traffic, which removes the need to extend the bridging overlay to the lean spine devices. With this limited role, you must configure only the IP fabric underlay and IBGP overlay on these devices.

On the leaf devices, configure a leaf-to-end system aggregated Ethernet interface as a trunk to carry multiple VLANs, establish LACP and ESI functionality, map VLANs to VNIs, configure proxy-macip-advertisement, virtual gateways, and static MAC addresses on the IRB interfaces, configure

EVPN/VXLAN in the default instance, enable VRF routing instances and IP prefix route properties for EVPN Type 5, and configure a default instance with the loopback interface as a VTEP source interface.

For an overview of edge-routed bridging overlays, see the Edge-Routed Bridging Overlay section in "Data Center Fabric Blueprint Architecture Components" on page 15.

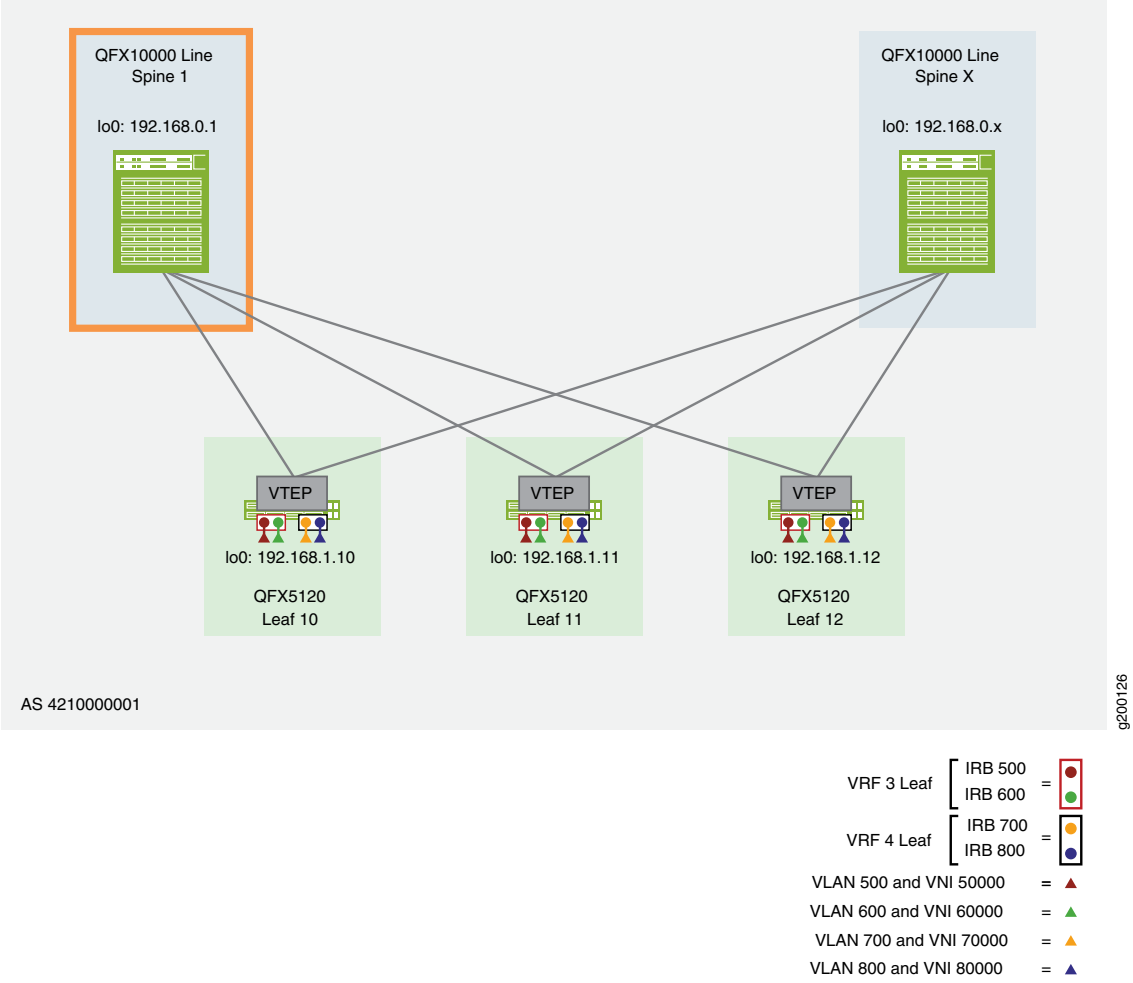
The following sections show the steps of how to configure and verify the edge-routed bridging overlay:

Configuring an Edge-Routed Bridging Overlay on a Lean Spine Device

To enable the edge-routed bridging overlay on a lean spine device, perform the following:

> **NOTE:** The following example shows the configuration for Spine 1, as shown in Figure 42 on page 161.

Figure 42: Edge-Routed Bridging Overlay – Lean Spine Devices



1. Ensure the IP fabric underlay is in place. To see the steps required to configure an IP fabric on a spine device, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine device, see “Configuring IBGP for the Overlay” on page 67.

Verifying the Edge-Routed Bridging Overlay on a Lean Spine Device

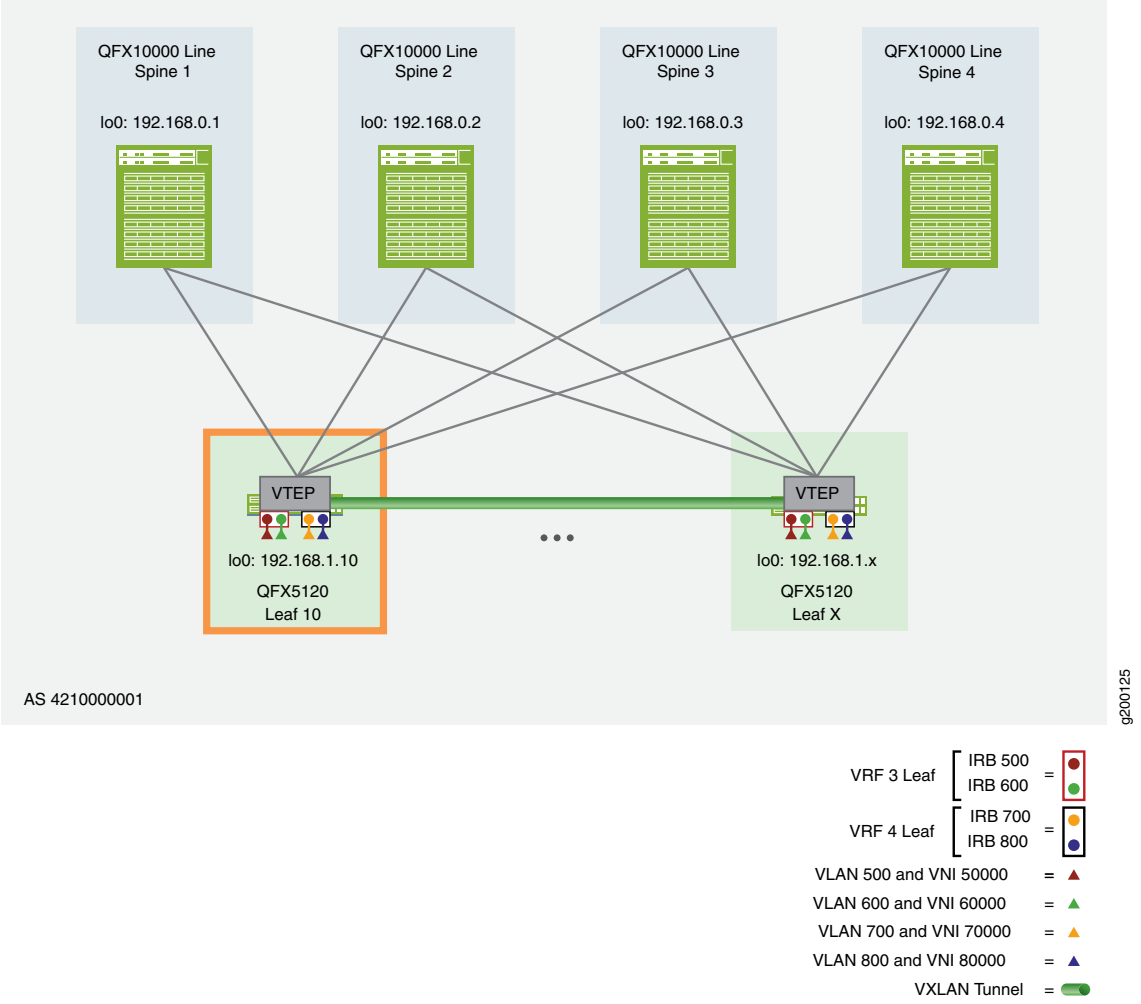
To verify that IBGP is functional on a lean spine device, use the **show bgp summary** command as described in “Configuring IBGP for the Overlay” on page 67. In the output that displays, ensure that the state of the lean spine device and its peers is **Establ** (established).

Configuring an Edge-Routed Bridging Overlay on a Leaf Device

To enable the edge-routed bridging overlay on a leaf device, perform the following:

> **NOTE:** The following example shows the configuration for Leaf 10, as shown in Figure 43 on page 163.

Figure 43: Edge-Routed Bridging Overlay – Leaf Devices



1. Ensure the IP fabric underlay is in place. To see the steps required to configure an IP fabric on a leaf device, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your leaf device, see "Configuring IBGP for the Overlay" on page 67.

3. Configure the loopback interface as a VTEP source interface within the default instance.

   Leaf 10:

   ```
   set switch-options vtep-source-interface lo0.0
   set switch-options route-distinguisher 192.168.1.10:1
   set switch-options vrf-target target:64512:1111
   set switch-options vrf-target auto
   ```

4. Configure the leaf-to-end system aggregated Ethernet interface as a trunk carrying four VLANs. Include the appropriate ESI and LACP values for your topology.

   Leaf 10:

   ```
   set interfaces ae11 esi 00:00:00:00:00:01:00:00:00:01
   set interfaces ae11 esi all-active
   set interfaces ae11 aggregated-ether-options lacp active
   set interfaces ae11 aggregated-ether-options lacp periodic fast
   set interfaces ae11 aggregated-ether-options lacp system-id 00:01:00:00:00:01
   set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
   set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_50000
   set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_60000
   set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_70000
   set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_80000
   ```

   NOTE: When configuring ESI-LAGs on QFX5*xxx* switches that serve as leaf devices in an edge-routed bridging overlay, keep in mind that we currently support only the Enterprise style of interface configuration, which is shown in this step.

5. Configure the mapping of VLANs to VNIs and associate one IRB interface per VLAN.

   Leaf 10:

   ```
   set vlans VNI_50000 vlan-id 500
   set vlans VNI_50000 l3-interface irb.500
   set vlans VNI_50000 vxlan vni 50000
   set vlans VNI_60000 vlan-id 600
   set vlans VNI_60000 l3-interface irb.600
   ```

```
set vlans VNI_60000 vxlan vni 60000
set vlans VNI_70000 vlan-id 700
set vlans VNI_70000 l3-interface irb.700
set vlans VNI_70000 vxlan vni 70000
set vlans VNI_80000 vlan-id 800
set vlans VNI_80000 l3-interface irb.800
set vlans VNI_80000 vxlan vni 80000
```

6. Configure the IRB interfaces for VNIs 50000 and 60000 with both IPv4 and IPv6 dual stack addresses for both the IRB IP address and virtual gateway IP address.

   There are two methods for configuring gateways for IRB interfaces:

   - Method 1: unique IRB IP Address with Virtual Gateway IP Address, which is shown in step 6.
   - Method 2: IRB with Anycast IP Address and MAC Address, which is shown in step 7.

   Leaf 10:

```
set interfaces irb unit 500 family inet address 10.1.4.1/24 virtual-gateway-address 10.1.4.254
set interfaces irb unit 500 family inet6 address 2001:db8::10:1:4:1/112 virtual-gateway-address
  2001:db8::10:1:4:254
set interfaces irb unit 500 family inet6 address fe80:10:1:4::1/64 virtual-gateway-address fe80:10:1:4::254
set interfaces irb unit 600 family inet address 10.1.5.1/24 virtual-gateway-address 10.1.5.254
set interfaces irb unit 600 family inet6 address 2001:db8::10:1:5:1/112 virtual-gateway-address
  2001:db8::10:1:5:254
set interfaces irb unit 600 family inet6 address fe80:10:1:5::1/64 virtual-gateway-address fe80:10:1:5::254
set interfaces irb unit 500 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 500 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 600 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 600 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

7. Configure the IRB interfaces for VNIs 70000 and 80000 with a dual stack Anycast IP address.

   Leaf 10:

```
set interfaces irb unit 700 family inet address 10.1.6.254/24
set interfaces irb unit 700 family inet6 address 2001:db8::10:1:6:254/112
set interfaces irb unit 700 family inet6 address fe80::10:1:6:254/112
set interfaces irb unit 700 mac 0a:fe:00:00:00:01
set interfaces irb unit 800 family inet address 10.1.7.254/24
set interfaces irb unit 800 family inet6 address 2001:db8::10:1:7:254/112
set interfaces irb unit 800 family inet6 address fe80::10:1:7:254/112
set interfaces irb unit 800 mac 0a:fe:00:00:00:02
```

For more information about IRB and virtual gateway IP address configuration, see the *IRB Addressing Models in Bridging Overlays* section in "Data Center Fabric Blueprint Architecture Components" on page 15.

8. Enable the ping operation for IRB interfaces 500 and 600, which are configured in step 6.

```
set interfaces irb unit 500 family inet address 10.1.4.1/24 preferred
set interfaces irb unit 500 family inet6 address 2001:db8::10:1:4:1/112 preferred
set interfaces irb unit 500 virtual-gateway-accept-data
set interfaces irb unit 600 family inet address 10.1.5.1/24 preferred
set interfaces irb unit 600 family inet6 address 2001:db8::10:1:5:1/112 preferred
set interfaces irb unit 600 virtual-gateway-accept-data
```

9. Configure EVPN VXLAN on the leaf device in the default instance.

   Leaf 10:

```
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn extended-vni-list all
```

10. Configure a policy called EXPORT_HOST_ROUTES to match on and accept /32 and /128 host routes, direct routes, and static routes. You will use this policy in step 12.

```
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 from protocol evpn
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 from route-filter 0.0.0.0/0
   prefix-length-range /32-/32
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 then accept
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_2 from protocol direct
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_2 then accept
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_3 from protocol static
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_3 then accept
set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from family inet6
set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from protocol evpn
set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from route-filter 0::0/0
   prefix-length-range /128-/128
set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 then accept
```

11. Configure the loopback interface with two logical interfaces. (You will assign one logical interface to each VRF routing instance in the next step).

```
set interfaces lo0 unit 3 family inet
set interfaces lo0 unit 4 family inet
```

12. Configure two tenant VRF routing instances, one for VNIs 50000 and 60000 (VRF 3), and one for VNIs 70000 and 80000 (VRF 4). Assign one logical interface from the loopback to each routing instance so that the VXLAN gateway can resolve ARP requests. Configure IP prefix route properties for EVPN type-5 to advertise ARP routes to the spine devices. Set up dummy IPv4 and IPv6 static routes, which you can use to discard traffic that results from denial of service (DoS) attacks. Enable load balancing for Layer 3 VPNs.

    Leaf 10:

```
set routing-instances VRF_3 instance-type vrf
set routing-instances VRF_3 interface irb.500
set routing-instances VRF_3 interface irb.600
set routing-instances VRF_3 interface lo0.3
set routing-instances VRF_3 route-distinguisher 192.168.1.10:500
set routing-instances VRF_3 vrf-target target:62273:50000
set routing-instances VRF_3 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_3 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_3 protocols evpn ip-prefix-routes vni 16777214
set routing-instances VRF_3 protocols evpn ip-prefix-routes export EXPORT_HOST_ROUTES
set routing-instances VRF_3 routing-options static route 10.10.20.30/32 discard
set routing-instances VRF-3 routing-options multipath
set routing-instances VRF-3 routing-options rib VRF-3.inet6.0 static route 2600:db8::101:1:1:1/128 discard
set routing-instances VRF-3 routing-options rib VRF-3.inet6.0 multipath
set routing-instances VRF_4 instance-type vrf
set routing-instances VRF_4 interface irb.700
set routing-instances VRF_4 interface irb.800
set routing-instances VRF_4 interface lo0.4
set routing-instances VRF_4 route-distinguisher 192.168.1.10:600
set routing-instances VRF_4 vrf-target target:62273:60000
set routing-instances VRF_4 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_4 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_4 protocols evpn ip-prefix-routes vni 16777105
set routing-instances VRF_4 protocols evpn ip-prefix-routes export EXPORT_HOST_ROUTES
set routing-instances VRF_4 routing-options static route 10.10.20.30/32 discard
set routing-instances VRF-4 routing-options multipath
set routing-instances VRF-4 routing-options rib VRF-4.inet6.0 static route 2600:db8::101:1:1:1/128 discard
set routing-instances VRF-4 routing-options rib VRF-4.inet6.0 multipath
```

13. If you are configuring a QFX5110, QFX5120-48Y, or QFX5120-32C switch, you must perform this step to support pure EVPN Type 5 routes on ingress EVPN traffic.

```
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set forwarding-options vxlan-routing overlay-ecmp
```

NOTE: Entering the **overlay-ecmp** statement causes the Packet Forwarding Engine to restart, which interrupts forwarding operations. We recommend using this configuration statement before the EVPN-VXLAN network becomes operational.

14. If you are configuring a QFX5110, QFX5120-48Y, or QFX5120-32C switch, and you expect that there will be more than 8000 ARP table entries and IPv6 neighbor entries, perform this step.

    Configure the maximum number of next hops reserved for use in the EVPN-VXLAN overlay network. By default, the switch allocates 8000 next hops for use in the overlay network. See *next-hop* for more details.

    NOTE: Changing the number of next hops causes the Packet Forwarding Engine to restart, which interrupts forwarding operations. We recommend using this configuration statement before the EVPN-VXLAN network becomes operational.

    ```
    set forwarding-options vxlan-routing next-hop 11000
    ```

Verifying the Edge-Routed Bridging Overlay on a Leaf Device

To verify that the edge-routed bridging overlay is working, perform the following:

1. Verify that the aggregated Ethernet interface is operational.

   `user@leaf-10>` **show interfaces terse ae11**

   ```
   Interface               Admin Link Proto    Local                 Remote
   ae11                    up    up
   ae11.0                  up    up   eth-switch
   ```

2. Verify the VLAN information (associated ESIs, VTEPs, etc.).

   `user@leaf-10>` **show vlans**

```
default-switch          VNI_50000            500
                                                        ae11.0*
                                                        esi.7585*
                                                        esi.7587*
                                                        esi.8133*
                                                        et-0/0/16.0
                                                        vtep.32782*
                                                        vtep.32785*
                                                        vtep.32802*
                                                        vtep.32806*
                                                        vtep.32826*

...

default-switch          VNI_80000            800
                                                        ae11.0*
                                                        esi.7585*
                                                        esi.8133*
                                                        et-0/0/16.0
                                                        vtep.32782*
                                                        vtep.32785*
                                                        vtep.32802*
                                                        vtep.32806*
                                                        vtep.32826*
```

Note: esi.7585 is the ESI of the remote aggregated Ethernet link for Leaf 4, Leaf 5, and Leaf 6.

`user@leaf-10>` **show ethernet-switching vxlan-tunnel-end-point esi | find esi.7585**

```
00:00:00:00:00:00:51:10:00:01 default-switch          7585  2097663 esi.7585
           3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.5          vtep.32826     8169     2         2
    192.168.1.6          vtep.32782     7570     1         2
    192.168.1.4          vtep.32785     7575     0         2
```

Note: esi.7587 is the ESI for all leaf devices that have the same VNI number (Leaf 4, Leaf 5, Leaf 6, Leaf 11, and Leaf 12).

`user@leaf-10>` **show ethernet-switching vxlan-tunnel-end-point esi | find esi.7587**

```
05:19:17:f3:41:00:00:c3:50:00 default-switch          7587  2097665 esi.7587
           5
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.5          vtep.32826     8169     4         2
```

```
    192.168.1.6          vtep.32782     7570     3         2
    192.168.1.12         vtep.32802     8127     2         2
    192.168.1.11         vtep.32806     8131     1         2
    192.168.1.4          vtep.32785     7575     0         2
```

Note: esi.8133 is the ESI for the local aggregated Ethernet interface shared with Leaf 11 and Leaf 12.

`user@leaf-10>` **show ethernet-switching vxlan-tunnel-end-point esi | find esi.8133**

```
00:00:00:00:00:01:00:00:00:01 default-switch           8133  2098194 esi.8133
 ae11.0,   2
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.12         vtep.32802     8127     1         2
    192.168.1.11         vtep.32806     8131     0         2
```

3. Verify the ARP table.

Note: 10.1.4.201 and 10.1.5.201 are remote end systems connected to the QFX5110 switches; and 10.1.4.202 and 10.1.5.202 are local end systems connected to Leaf 10 through interface ae11.

`user@leaf-10>` **show arp no-resolve vpn VRF_3**

```
MAC Address       Address         Interface         Flags
06:a7:39:9a:7b:c0 10.1.4.4        irb.500 [vtep.32785]     none
06:a7:39:f8:b1:00 10.1.4.5        irb.500 [vtep.32826]     none
06:e0:f3:1b:09:80 10.1.4.6        irb.500 [vtep.32782]     none
02:0c:10:04:02:01 10.1.4.201      irb.500 [.local..9]      none
02:0c:10:04:02:02 10.1.4.202      irb.500 [ae11.0]         none
00:00:5e:00:00:04 10.1.4.254      irb.500                  permanent published
 gateway
06:a7:39:9a:7b:c0 10.1.5.4        irb.600 [vtep.32785]     none
06:a7:39:f8:b1:00 10.1.5.5        irb.600 [vtep.32826]     none
06:e0:f3:1b:09:80 10.1.5.6        irb.600 [vtep.32782]     none
02:0c:10:05:02:01 10.1.5.201      irb.600 [.local..9]      none
02:0c:10:05:02:02 10.1.5.202      irb.600 [ae11.0]         none
00:00:5e:00:00:04 10.1.5.254      irb.600                  permanent published
 gateway
Total entries: 12
```

`user@leaf-10>` **show arp no-resolve vpn VRF_4**

```
MAC Address       Address         Interface         Flags
02:0c:10:06:02:01 10.1.6.201      irb.700 [.local..9]      permanent remote
02:0c:10:06:02:02 10.1.6.202      irb.700 [ae11.0]         none
```

```
02:0c:10:07:02:01 10.1.7.201      irb.800 [.local..9]      permanent remote
02:0c:10:07:02:02 10.1.7.202      irb.800 [ae11.0]         permanent remote
Total entries: 4
```

user@leaf-10> **show ipv6 neighbors**

```
IPv6 Address                 Linklayer Address  State       Exp Rtr Secure
Interface
2001:db8::10:1:4:2           06:ac:ac:23:0c:4e  stale       523 yes no
irb.500 [vtep.32806]
2001:db8::10:1:4:3           06:ac:ac:24:18:7e  stale       578 yes no
irb.500 [vtep.32802]
2001:db8::10:1:4:201         02:0c:10:04:02:01  stale       1122 no no
irb.500 [.local..9]
2001:db8::10:1:4:202         02:0c:10:04:02:02  stale       1028 no no
irb.500 [ae11.0]
2001:db8::10:1:4:254         00:00:5e:00:00:04   reachable   0   no  no
irb.500
2001:db8::10:1:5:2           06:ac:ac:23:0c:4e  stale       521 yes no
irb.600 [vtep.32806]
2001:db8::10:1:5:3           06:ac:ac:24:18:7e  stale       547 yes no
irb.600 [vtep.32802]
2001:db8::10:1:5:201         02:0c:10:05:02:01  stale       508 no  no
irb.600 [.local..9]
2001:db8::10:1:5:202         02:0c:10:05:02:02  stale       1065 no no
irb.600 [ae11.0]
2001:db8::10:1:5:254         00:00:5e:00:00:04   reachable   0   no  no
irb.600
2001:db8::10:1:6:202         02:0c:10:06:02:02  reachable   0   no  no
irb.700 [ae11.0]
2001:db8::10:1:7:201         02:0c:10:07:02:01  stale       647 no  no
irb.800 [.local..9]
```

4. Verify the MAC addresses and ARP information in the EVPN database.

user@leaf-10> **show evpn database mac-address 02:0c:10:04:02:01 extensive**

```
Instance: default-switch

VN Identifier: 50000, MAC address:: 02:0c:10:04:02:01
  Source: 00:00:00:00:00:00:51:10:00:01, Rank: 1, Status: Active
    Remote origin: 192.168.1.4
    Remote origin: 192.168.1.5
    Timestamp: Oct 10 16:09:54 (0x59dd5342)
```

```
    State: <Remote-To-Local-Adv-Done>
    IP address: 10.1.4.201
      Remote origin: 192.168.1.4
      Remote origin: 192.168.1.5
    IP address: 2001:db8::10:1:4:201
      Flags: <Proxy>
      Remote origin: 192.168.1.4
      Remote origin: 192.168.1.5    History db:
      Time                  Event
      Oct 10 16:09:55 2017  Advertisement route cannot be created (no local
state present)
      Oct 10 16:09:55 2017  Updating output state (change flags 0x0)
      Oct 10 16:09:55 2017  Advertisement route cannot be created (no local
source present)
      Oct 10 16:09:55 2017  IP host route cannot be created (No remote host
route for non-MPLS instance)
      Oct 10 16:09:55 2017  Updating output state (change flags 0x4000
<IP-Peer-Added>)
      Oct 10 16:09:55 2017  Creating MAC+IP advertisement route for proxy
      Oct 10 16:09:55 2017  Creating MAC+IP advertisement route for proxy
      Oct 10 16:09:55 2017  IP host route cannot be created (No remote host
route for non-MPLS instance)
      Oct 10 16:09:55 2017  Clearing change flags <IP-Added>
      Oct 10 16:09:55 2017  Clearing change flags <IP-Peer-Added>
```

`user@leaf-10>` **show evpn database mac-address 02:0c:10:04:02:02 extensive**

```
Instance: default-switch

VN Identifier: 50000, MAC address:: 02:0c:10:04:02:02
  Source: 00:00:00:00:00:01:00:00:00:01, Rank: 1, Status: Active
    Remote origin: 192.168.1.11
    Remote origin: 192.168.1.12
    Timestamp: Oct 10 16:14:32 (0x59dd5458)
    State: <Remote-To-Local-Adv-Done>
    IP address: 10.1.4.202
      Remote origin: 192.168.1.11
      Remote origin: 192.168.1.12
      L3 route: 10.1.4.202/32, L3 context: VRF_3 (irb.500)
    IP address: 2001:db8::10:1:4:202
      Remote origin: 192.168.1.11
     L3 route: 2001:db8::10:1:4:202/128, L3 context: VRF_3 (irb.500)    History
 db:
      Time                  Event
```

```
      Oct 10 16:14:32 2017  Advertisement route cannot be created (no local
state present)
      Oct 10 16:14:32 2017  Sent MAC add with NH 0, interface ae11.0 (index 0),
 RTT 9, remote addr 192.168.1.12, ESI 0100000001, VLAN 0, VNI 50000, flags 0x0,
 timestamp 0x59dd5458 to L2ALD
      Oct 10 16:14:32 2017  Sent peer 192.168.1.12 record created
      Oct 10 16:14:32 2017  Sent MAC+IP add, interface <none>, RTT 9, IP
10.1.4.202 remote peer 192.168.1.12, ESI 0100000001, VLAN 0, VNI 50000, flags
0x80, timestamp 0x59dd5458 to L2ALD
      Oct 10 16:14:32 2017  Updating output state (change flags 0x4000
<IP-Peer-Added>)
      Oct 10 16:14:32 2017  Advertisement route cannot be created (no local
source present)
      Oct 10 16:14:32 2017  Updating output state (change flags 0x0)
      Oct 10 16:14:32 2017  Advertisement route cannot be created (no local
source present)
      Oct 10 16:14:32 2017  Clearing change flags <ESI-Peer-Added>
      Oct 10 16:14:32 2017  Clearing change flags <IP-Peer-Added>
```

5. Verify the IPv4 and IPv6 end system routes appear in the forwarding table.

   `user@leaf-10>` **show route forwarding-table table VRF_1 destination 10.1.4.202 extensive**

```
Routing table: VRF_3.inet [Index 5]
Internet:
Enabled protocols: Bridging, All VLANs,

Destination:  10.1.4.202/32
  Route type: destination
  Route reference: 0                   Route interface-index: 563
  Multicast RPF nh index: 0
  P2mpidx: 0
  Flags: sent to PFE
  Nexthop: b0:c:10:4:2:2
  Next-hop type: unicast               Index: 10210    Reference: 1
  Next-hop interface: ae11.0
```

   `user@leaf-10>` **show route forwarding-table table VRF_1 destination 2001:db8::10:1:4:202 extensive**

```
Routing table: VRF_3.inet6 [Index 5]
Internet6:
Enabled protocols: Bridging, All VLANs,
```

```
Destination:  2001:db8::10:1:4:202/128
  Route type: destination
  Route reference: 0                   Route interface-index: 563
  Multicast RPF nh index: 0
  P2mpidx: 0
  Flags: sent to PFE
  Nexthop: b0:c:10:4:2:2
  Next-hop type: unicast               Index: 10244    Reference: 1
  Next-hop interface: ae11.0
```

SEE ALSO

- *Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Overlay Within a Data Center*
- *Example: Configuring a QFX5110 Switch as Layer 2 and 3 VXLAN Gateways in an EVPN-VXLAN Edge-Routed Bridging Overlay*

Edge-Routed Bridging Overlay — Release History

Table 6 on page 174 provides a history of all of the features in this section and their support within this reference design.

Table 6: Edge-Routed Bridging Overlay in the Cloud Data Center Reference Design– Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support edge-routed bridging overlays.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support edge-routed bridging overlays.
18.1R3-S3	QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support edge-routed bridging overlays.
17.3R3-S1	QFX10002-36Q/72Q switches running Junos 17.3R3-S1 and later releases in the same release train support edge-routed bridging overlays.

RELATED DOCUMENTATION

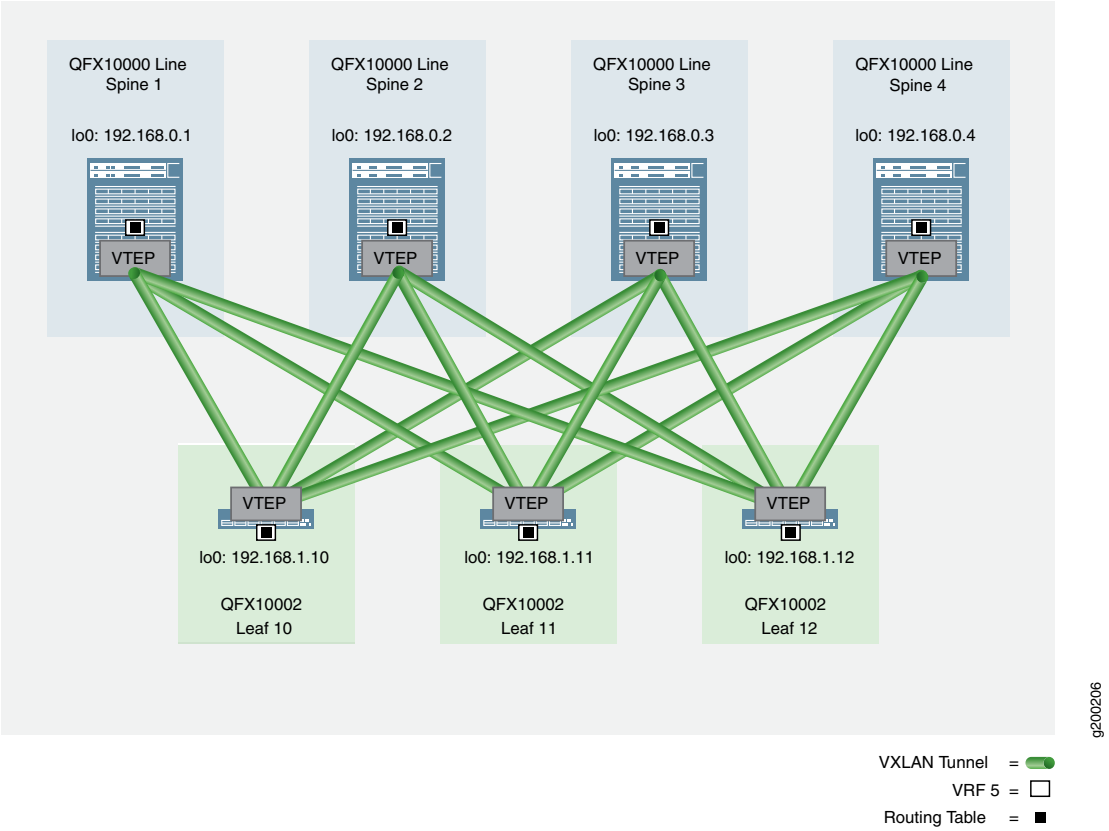
- *EVPN User Guide*
- *ip-prefix-routes*
- *proxy-macip-advertisement*
- *virtual-gateway-address*
- *EVPN Overview*
- *Understanding VXLANs*
- *Understanding EVPN with VXLAN Data Plane Encapsulation*
- *Configuring EVPN Routing Instances*
- *Example: Configuring an ESI on a Logical Interface With EVPN Multihoming*
- *Overview of VLAN Services for EVPN*
- *BGP User Guide*
- *Configuring Aggregated Ethernet LACP (CLI Procedure)*
- *Routing Instances Overview*
- *Example: Configuring VNI Route Targets Manually*

Routed Overlay Design and Implementation

IN THIS SECTION

A third overlay option for this Cloud Data Center reference design is a routed overlay, as shown in Figure 44 on page 176.

Figure 44: Routed Overlay



The routed overlay service supports IP-connected end systems that interconnect with leaf devices using IP (not Ethernet). The end systems can use dynamic routing protocols to exchange IP routing information with the leaf devices. The IP addresses and prefixes of the end systems are advertised as EVPN type-5 routes and imported by other EVPN/VXLAN-enabled spine and leaf devices into a corresponding VRF routing instance.

To implement a routed overlay, you must use one of the QFX10000 line of switches as the leaf devices. (The QFX5110 is planned to be verified in a future revision of this guide.)

On the spine devices, create a VRF routing instance to accept the EVPN type-5 routes. Include route distinguishers and route targets, use the **advertise direct-nexthop** option, and configure the same VNI used by the leaf devices.

To configure the leaf devices, create a policy that matches on the BGP routes received from the end systems, and send these routes into a VRF routing instance enabled for EVPN type-5 routes so they can be shared with other leaf and spine devices in the same IP VNI.

For an overview of routed overlays, see the Routed Overlay section in “Data Center Fabric Blueprint Architecture Components” on page 15.

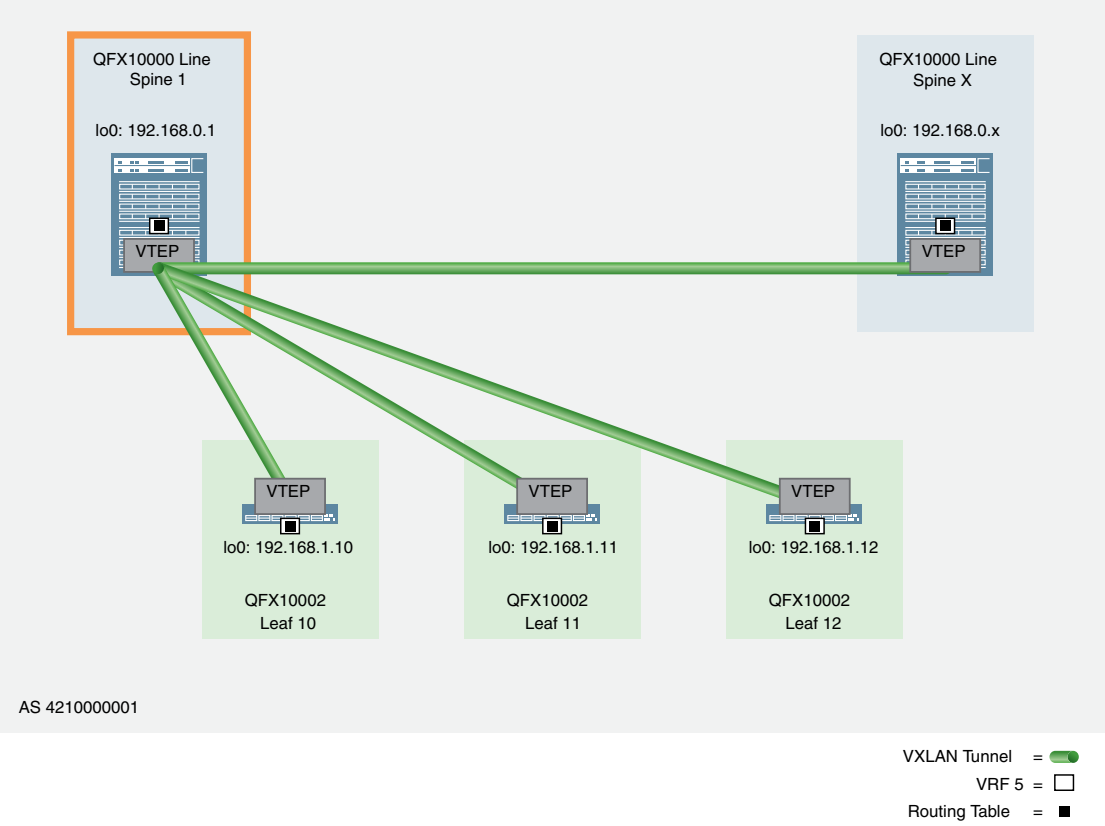
The following sections show the detailed steps of how to configure and verify the routed overlay:

Configuring the Routed Overlay on a Spine Device

To configure the routed overlay on a spine device, perform the following:

> **NOTE:** The following example shows the configuration for Spine 1, as shown in Figure 45 on page 177.

Figure 45: Routed Overlay – Spine Devices



1. Ensure the IP fabric underlay is in place. To see the steps required to configure an IP fabric on a spine device, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine device, see "Configuring IBGP for the Overlay" on page 67.

3. Create a VRF routing instance that enables the EVPN type-5 option (also known as IP prefix routes). Configure a route distinguisher and route target, provide load balancing for EVPN type-5 routes with the **multipath** ECMP option, enable EVPN to advertise direct next hops, specify VXLAN encapsulation, and assign the VNI.

   > **NOTE:** This VRF can be used for north-south traffic flow and will be explained in a future version of this guide.

   Spine 1:

```
set routing-instances VRF_5 instance-type vrf
set routing-instances VRF_5 route-distinguisher 192.168.0.1:1677
set routing-instances VRF_5 vrf-target target:62273:16776000
set routing-instances VRF_5 routing-options multipath
set routing-instances VRF_5 routing-options rib VRF_5.inet6.0 multipath
set routing-instances VRF_5 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_5 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_5 protocols evpn ip-prefix-routes vni 16776000
```

Verifying the Routed Overlay on a Spine Device

To verify that the routed overlay on a spine device is working, perform the following:

1. Verify that the EVPN type-5 routes are being advertised and received for IPv4 and IPv6.

   `user@spine-1>` **show evpn ip-prefix-database l3-context VRF_5**

```
L3 context: VRF_5
```

   EVPN to IPv4 Imported Prefixes

```
Prefix                                  Etag
172.16.104.0/30                           0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.10:10        16776000   06:31:46:e2:f0:2a  192.168.1.10
172.16.104.4/30                           0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
```

```
172.16.104.8/30                                0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
192.168.3.4/32                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.10:10        16776000   06:31:46:e2:f0:2a  192.168.1.10
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
```

```
EVPN-->IPv6 Imported Prefixes
Prefix                                         Etag
2001:db8::172:19:4:0/126                       0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.10:10        16776000   06:31:46:e2:f0:2a  192.168.1.10
2001:db8::172:19:4:4/126                       0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
2001:db8::172:19:4:8/126                       0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
2001:db8::192:168:3:4/128                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.10:10        16776000   06:31:46:e2:f0:2a  192.168.1.10
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
```

2. Verify that the end system is multihomed to all three QFX10000 leaf devices for both IPv4 and IPv6.

   `user@spine-1>` **show route 192.168.3.4 table VRF_5**

```
VRF_5.inet.0: 4 destinations, 7 routes (4 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

192.168.3.4/32     @[EVPN/170] 01:15:09
                     > to 172.16.10.1 via ae10.0
                    [EVPN/170] 01:19:53
                     > to 172.16.11.1 via ae11.0
                    [EVPN/170] 00:06:21
                     > to 172.16.12.1 via ae12.0
                    #[Multipath/255] 00:06:21, metric2 0
                        to 172.16.10.1 via ae10.0
```

```
                      to 172.16.11.1 via ae11.0
                    > to 172.16.12.1 via ae12.0
```

user@spine-1> **show route 2001:db8::192:168:3:4 table VRF_5**

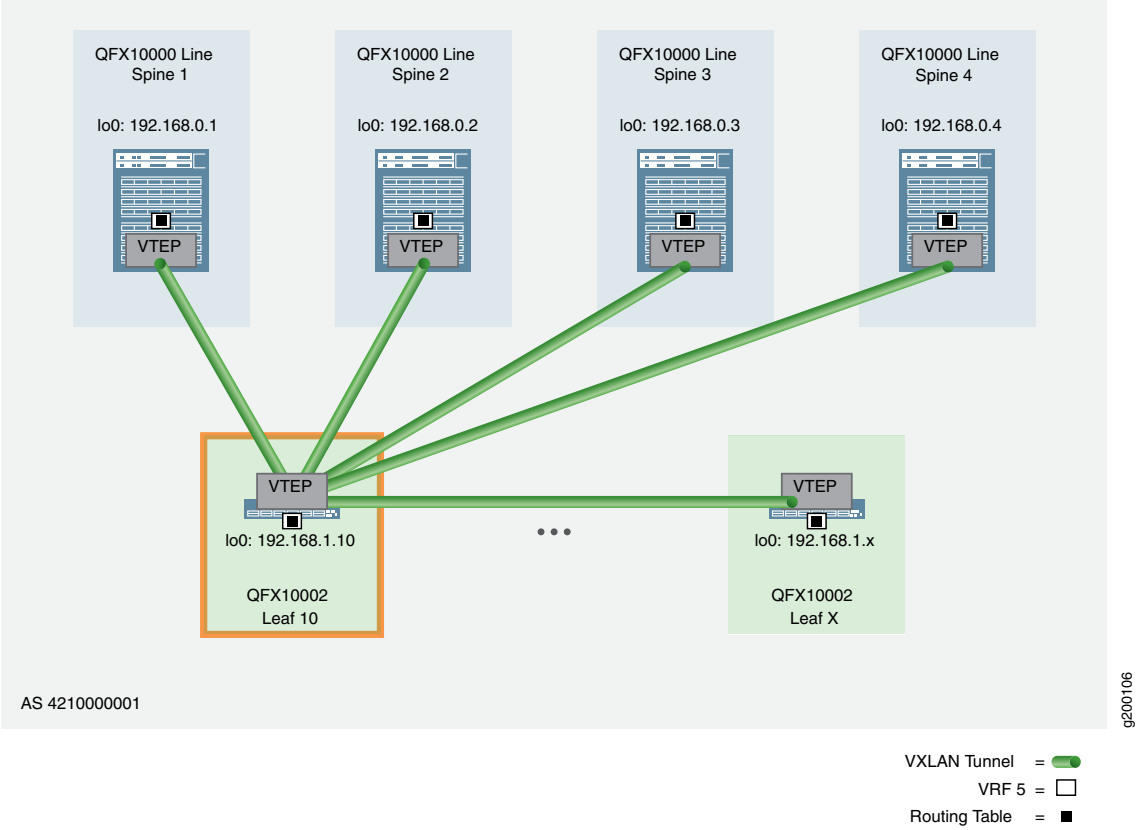
```
VRF_5.inet6.0: 4 destinations, 7 routes (4 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

2001:db8::192:168:3:4/128
                   @[EVPN/170] 01:17:04
                     > to 172.16.10.1 via ae10.0
                    [EVPN/170] 01:21:48
                     > to 172.16.11.1 via ae11.0
                    [EVPN/170] 00:08:16
                     > to 172.16.12.1 via ae12.0
                   #[Multipath/255] 00:00:12, metric2 0
                      to 172.16.10.1 via ae10.0
                     > to 172.16.11.1 via ae11.0
                      to 172.16.12.1 via ae12.0
```

Configuring the Routed Overlay on a Leaf Device

To configure the routed overlay on a leaf device, perform the following:

> **NOTE:** The following example shows the configuration for Leaf 10, as shown in Figure 46 on page 181.

Figure 46: Routed Overlay – Leaf Devices



1. Ensure the IP fabric underlay is in place. To see the steps required to configure an IP fabric on a leaf device, see "IP Fabric Underlay Network Design and Implementation" on page 51.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your leaf device, see "Configuring IBGP for the Overlay" on page 67.

3. Configure a VRF routing instance to extend EBGP peering to an end system. Specify a route target, route distinguisher, and the IP address and ASN (AS 4220000001) of the IP-connected end system to

allow the leaf device and end system to become BGP neighbors. To learn how to implement IP multihoming, see "Multihoming an IP-Connected End System Design and Implementation" on page 189.

> **NOTE:** Each VRF routing instance in the network should have a unique route distinguisher. In this reference design, the route distinguisher is a combination of the loopback interface IP address of the device combined with a unique identifier to signify the device. For VRF 5, the loopback interface IP address on Leaf 10 is 192.168.1.10 and the ID is 15, making the route distinguisher *192.168.1.10:15*.

Leaf 10:

```
set routing-instances VRF_5 instance-type vrf
set routing-instances VRF_5 interface et-0/0/15.0
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 type external
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 neighbor 172.16.104.2 peer-as
    4220000001
set routing-instances VRF_5 route-distinguisher 192.168.1.10:15
set routing-instances VRF_5 vrf-target target:62273:16776000
```

4. Create a policy to match direct routes and BGP routes.

   Leaf 10:

   ```
   set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_1 from protocol bgp
   set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_1 then accept
   set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_2 from protocol direct
   set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_2 then accept
   set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_3 then reject
   ```

5. Complete your configuration of the VRF routing instance by configuring the EVPN type-5 option. To implement this feature, configure EVPN to advertise direct next hops such as the IP-connected end system, specify VXLAN encapsulation, assign the VNI, and export the BGP policy that accepts the end system routes.

   Leaf 10:

   ```
   set routing-instances VRF_5 protocols evpn ip-prefix-routes advertise direct-nexthop
   set routing-instances VRF_5 protocols evpn ip-prefix-routes encapsulation vxlan
   set routing-instances VRF_5 protocols evpn ip-prefix-routes vni 16776000
   ```

```
set routing-instances VRF_5 protocols evpn ip-prefix-routes export ADVERTISE_EDGE_ROUTES
```

Verifying the Routed Overlay on a Leaf Device

NOTE: The operational mode command output included in the following sections assumes you have configured IP multihoming as explained in "Multihoming an IP-Connected End System Design and Implementation" on page 189.

To verify that the routed overlay on a leaf device is working, perform the following:

1. Verify that the IPv4 routes from the VRF are converted to EVPN type-5 routes are advertised to EVPN peers.

   `user@leaf-10>` **show evpn ip-prefix-database l3-context VRF_5**

```
L3 context: VRF_5
```

IPv4 to EVPN Exported Prefixes

```
Prefix                                  EVPN route status
172.16.104.0/30                         Created
192.168.3.4/32                          Created  ## Note: this is the Lo0
 interface of a end system
```

IPv6 to EVPN Exported Prefixes

```
Prefix                                  EVPN route status
2001:db8::172:19:4:0/126                Created
2001:db8::192:168:3:4/128               Created  ## Note: this is the Lo0
 interface of an end systemEVPN to IPv4 Imported Prefixes. These are received
as a
type-5 route, and converted back to IPv4.
Prefix                                  Etag
172.16.104.4/30                           0  ## From Leaf 11
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
```

```
172.16.104.8/30                                0  ## From Leaf 12
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
192.168.3.4/32                                 0  ## The loopback address of the
 end system
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
```

EVPN to IPv6 Imported Prefixes

```
Prefix                                         Etag
2001:db8::172:19:4:4/126                       0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
2001:db8::172:19:4:8/126                       0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
2001:db8::192:168:3:4/128                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
```

2. View the VRF route table for IPv4, IPv6, and EVPN to verify that the end system routes and spine device routes are being exchanged.

   `user@leaf-10>` **show route table VRF_5**

   This is the IPv4 section

```
VRF_5.inet.0: 5 destinations, 7 routes (5 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both
172.16.104.0/30      *[Direct/0] 01:33:42
                     > via et-0/0/15.0
172.16.104.1/32      *[Local/0] 01:33:42
                       Local via et-0/0/15.0
172.16.104.4/30      *[EVPN/170] 01:10:08
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
172.16.104.8/30      *[EVPN/170] 00:01:25
                     > to 172.16.10.2 via ae1.0
                       to 172.16.10.6 via ae2.0
192.168.3.4/32     *[BGP/170] 01:33:39, localpref 100
```

```
                      AS path: 4220000001 I
                     validation-state: unverified, > to 172.16.104.2 via
et-0/0/15.0
                     [EVPN/170] 01:10:08
                        to 172.16.10.2 via ae1.0
                      > to 172.16.10.6 via ae2.0
                     [EVPN/170] 00:01:25
                        to 172.16.10.2 via ae1.0
                      > to 172.16.10.6 via ae2.0
```

This is the IPv6 section

```
VRF_5.inet6.0: 6 destinations, 8 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8::172:19:4:0/126
                    *[Direct/0] 01:33:33
                     > via et-0/0/15.0
2001:db8::172:19:4:1/128
                    *[Local/0] 01:33:33
                       Local via et-0/0/15.0
2001:db8::172:19:4:4/126
                    *[EVPN/170] 01:10:08
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
2001:db8::172:19:4:8/126
                    *[EVPN/170] 00:01:25
                     > to 172.16.10.2 via ae1.0
                       to 172.16.10.6 via ae2.0
2001:db8::192:168:3:4/128
                    *[BGP/170] 01:33:28, localpref 100
                       AS path: 4220000001 I, validation-state: unverified
                     > to 2001:db8::172:19:4:2 via et-0/0/15.0
                     [EVPN/170] 01:10:08
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
                     [EVPN/170] 00:01:25
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
fe80::231:4600:ae2:f06c/128
                    *[Local/0] 01:33:33
                       Local via et-0/0/15.0
```

This is the EVPN section. EVPN routes use the following convention:

Type:Route Distinguisher::0::IP Address::Prefix/NLRI Prefix

```
VRF_5.evpn.0: 12 destinations, 36 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

5:192.168.1.10:10::0::172.16.104.0::30/248
                   *[EVPN/170] 01:33:42
                      Indirect
5:192.168.1.10:10::0::192.168.3.4::32/248
                   *[EVPN/170] 01:33:39
                      Indirect
5:192.168.1.11:10::0::172.16.104.4::30/248
                   *[BGP/170] 01:10:08, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 01:09:54, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 01:10:08, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 01:09:39, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0

...

5:192.168.1.12:10::0::192.168.3.4::32/248
                   *[BGP/170] 00:01:20, localpref 100, from 192.168.0.1
                      AS path: 4220000001 I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:21, localpref 100, from 192.168.0.2
                      AS path: 4220000001 I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:25, localpref 100, from 192.168.0.3
                      AS path: 4220000001 I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:17, localpref 100, from 192.168.0.4
```

```
                      AS path: 4220000001 I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
```

```
5:192.168.1.10:10::0::2001:db8::172:19:4:0::126/248
                   *[EVPN/170] 01:33:33
                      Indirect
5:192.168.1.10:10::0::2001:db8::192:168:3:4::128/248
                   *[EVPN/170] 01:33:28
                      Indirect
5:192.168.1.11:10::0::2001:db8::172:19:4:4::126/248
                   *[BGP/170] 01:10:08, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 01:09:54, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 01:10:08, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 01:09:39, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0

...

5:192.168.1.12:10::0::2001:db8::192:168:3:4::128/248
                   *[BGP/170] 00:01:20, localpref 100, from 192.168.0.1
                      AS path: 4220000001 I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:21, localpref 100, from 192.168.0.2
                      AS path: 4220000001 I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:25, localpref 100, from 192.168.0.3
                      AS path: 4220000001 I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:17, localpref 100, from 192.168.0.4
```

```
                    AS path: 4220000001 I, validation-state: unverified
                    to 172.16.10.2 via ae1.0
                  > to 172.16.10.6 via ae2.0
```

SEE ALSO

- *Understanding EVPN Pure Type-5 Routes*
- *ip-prefix-routes*

Routed Overlay — Release History

Table 7 on page 188 provides a history of all of the features in this section and their support within this reference design.

Table 7: Routed Overlay in the Cloud Data Center Reference Design – Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section.
18.1R3-S3	QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section.
17.3R3-S1	All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section.

RELATED DOCUMENTATION

- *EVPN User Guide*
- *EVPN Overview*
- *Understanding VXLANs*
- *Understanding EVPN with VXLAN Data Plane Encapsulation*

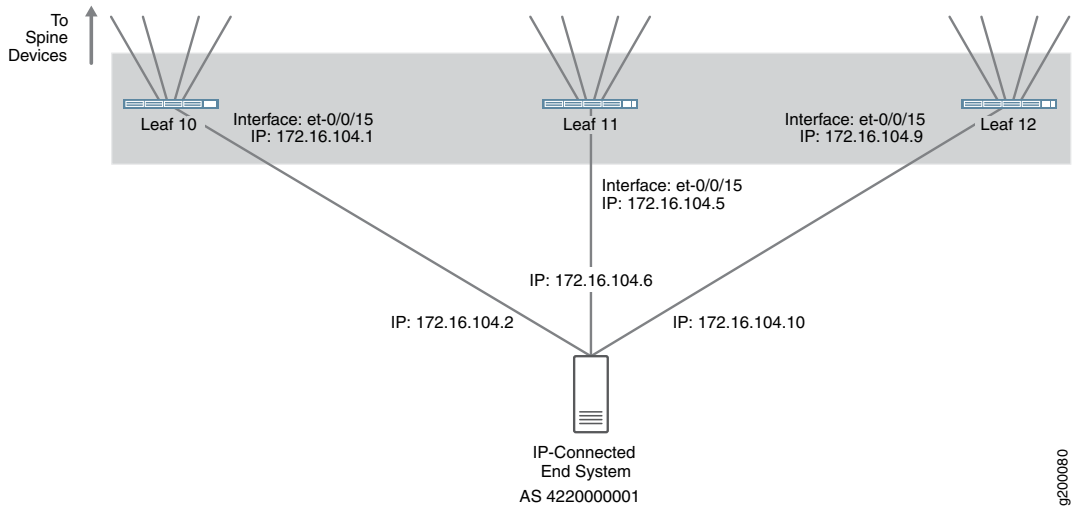
Multihoming an IP-Connected End System Design and Implementation

IN THIS SECTION

For an overview of multihoming an IP-connected end system, see the Multihoming Support for IP-Connected End Systems section in "Data Center Fabric Blueprint Architecture Components" on page 15.

Figure 47 on page 189 illustrates the multihomed IP-connected end system—in this procedure, the end system is an IP-connected server—that is enabled using this procedure:

Figure 47: Multihomed IP-Connected End System Example



NOTE: This configuration uses one VLAN per leaf device access interface. See *Configuring a Layer 3 Subinterface (CLI Procedure)* to enable VLAN tagging if your setup requires multiple VLANs per leaf device access interface.

Configuring the End System-Facing Interfaces on a Leaf Device

To configure the IP address of each end system-facing interface on the leaf devices:

1. Configure an IP address on each end system-facing leaf device interface, and put the interface in a VRF.

   Leaf 10:

   ```
   set interfaces et-0/0/15 unit 0 family inet address 172.16.104.1/30
   set routing-instances VRF_5 interface et-0/0/15.0
   ```

   NOTE: The **VRF_5** routing instance was configured earlier in this guide with this EBGP configuration. See "Routed Overlay Design and Implementation" on page 175.

   Leaf 11:

   ```
   set interfaces et-0/0/15 unit 0 family inet address 172.16.104.5/30
   set routing-instances VRF_5 interface et-0/0/15.0
   ```

   Leaf 12:

   ```
   set interfaces et-0/0/15 unit 0 family inet address 172.16.104.9/30
   set routing-instances VRF_5 interface et-0/0/15.0
   ```

> **NOTE:** The **VRF_5** routing instance was not explicitly configured for Leaf 11 and 12 earlier in this guide.
>
> To configure the **VRF_5** routing instance to run on Leaf 11 and 12, repeat the procedures in "Routed Overlay Design and Implementation" on page 175 for these leaf devices. Be sure to configure a unique route distinguisher for each route to ensure that each leaf device route is treated as a unique route by the route reflectors.

2. After committing the configuration, confirm that each link is up and that the IP address is properly assigned.

   Sample output from Leaf 10 only is provided in this step.

   Leaf 10:

```
user@leaf10> show interfaces terse et-0/0/15
Interface               Admin Link Proto    Local                 Remote
et-0/0/15               up    up
et-0/0/15.0             up    up   inet     172.16.104.1/30
```

Configuring EBGP Between the Leaf Device and the IP-Connected End System

EBGP—which is already used in this reference design to pass routes between spine and leaf devices in the underlay network—is also used in this reference design to pass routes between the IP-connected server and a leaf device.

> **NOTE:** Other routing protocols can be used to pass routes between an IP-connected end system and the leaf device. However this is not recommended.

To configure EBGP between a leaf device and an IP-connected end system:

1. Configure EBGP in a VRF routing instance.

   Leaf 10:

```
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 type external
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 neighbor 172.16.104.2 peer-as
4220000001
```

2. Repeat this procedure on all leaf device interfaces that are multihomed to the IP-connected end system.

   Leaf 11:

```
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 type external
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 neighbor 172.16.104.6 peer-as
4220000001
```

   Leaf 12:

```
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 type external
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 neighbor 172.16.104.10 peer-as
4220000001
```

3. After committing the configurations, confirm that BGP is operational.

   A sample of this verification procedure on Leaf 10 is provided below.

   Leaf 10:

```
user@leaf10> show bgp summary instance VRF_5
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
VRF_5.inet.0    1          1          0          0          0          0
VRF_5.mdt.0     0          0          0          0          0          0
VRF_5.evpn.0   32          8          0          0          0          0

Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn
State|#Active/Received/Accepted/Damped...
172.16.104.2        422000001     28         24       0       1     10:37
Establ VRF_5.inet.0: 1/1/1/0
```

Multihoming an IP-Connected End System—Release History

Table 8 on page 193 provides a history of all of the features in this section and their support within this reference design.

Table 8: Multihoming IP-Connected End Systems Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section.
17.3R3-S1	All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section.

RELATED DOCUMENTATION

EVPN Multihoming Overview

Routed Overlay Design and Implementation | **175**

Edge-Routed Bridging Overlay Design and Implementation | **158**

Data Center Interconnect Design and Implementation Using Type 5 Routes

IN THIS SECTION

- Data Center Interconnect Using EVPN Type 5 Routes | **194**

Data Center Interconnect Using EVPN Type 5 Routes

IN THIS SECTION

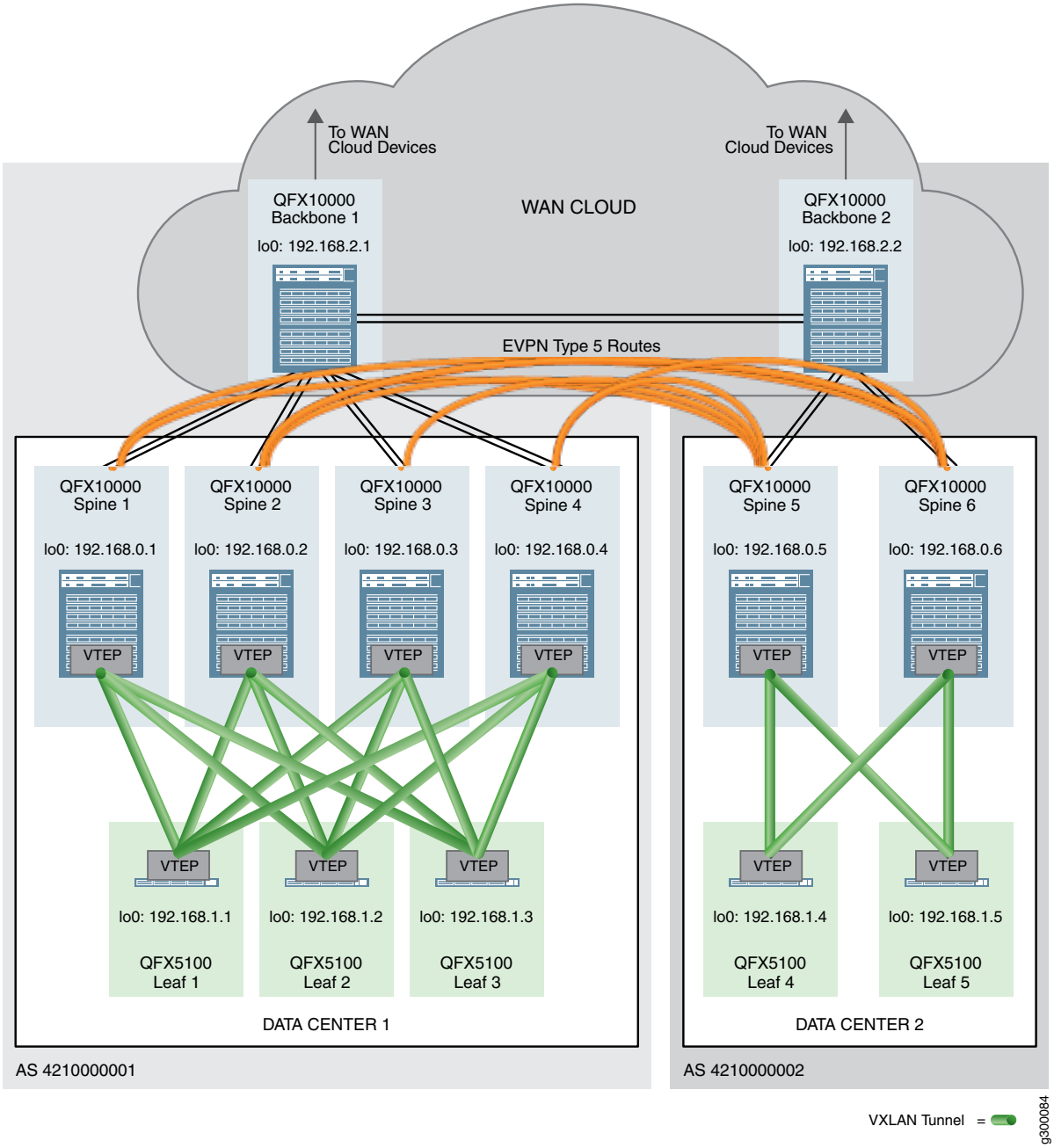
EVPN Type 5 routes, also known as IP prefix routes, are used in a DCI context to pass traffic between data centers that are using different IP address subnetting schemes.

In this reference architecture, EVPN Type 5 routes are exchanged between spine devices in different data centers to allow for the passing of traffic between data centers.

Physical connectivity between the data centers is required before EVPN Type 5 messages can be sent across data centers. This physical connectivity is provided by backbone devices in a WAN cloud. A backbone device is connected to each spine device in a single data center and participates in the overlay IBGP and underlay EBGP sessions. EBGP also runs in a separate BGP group to connect the backbone devices to each other; EVPN signaling is enabled in this BGP group.

Figure 48 on page 195 shows two data centers using EVPN Type 5 routes for DCI.

Figure 48: DCI Using EVPN Type 5 Routes Topology Overview



For additional information on EVPN Type 5 routes, see EVPN Type-5 Route with VXLAN encapsulation for EVPN-VXLAN.

All procedures in this section assume that EVPN Type 2 routes are successfully being passed in the data centers. See “Centrally-Routed Bridging Overlay Design and Implementation” on page 95 for setup instructions.

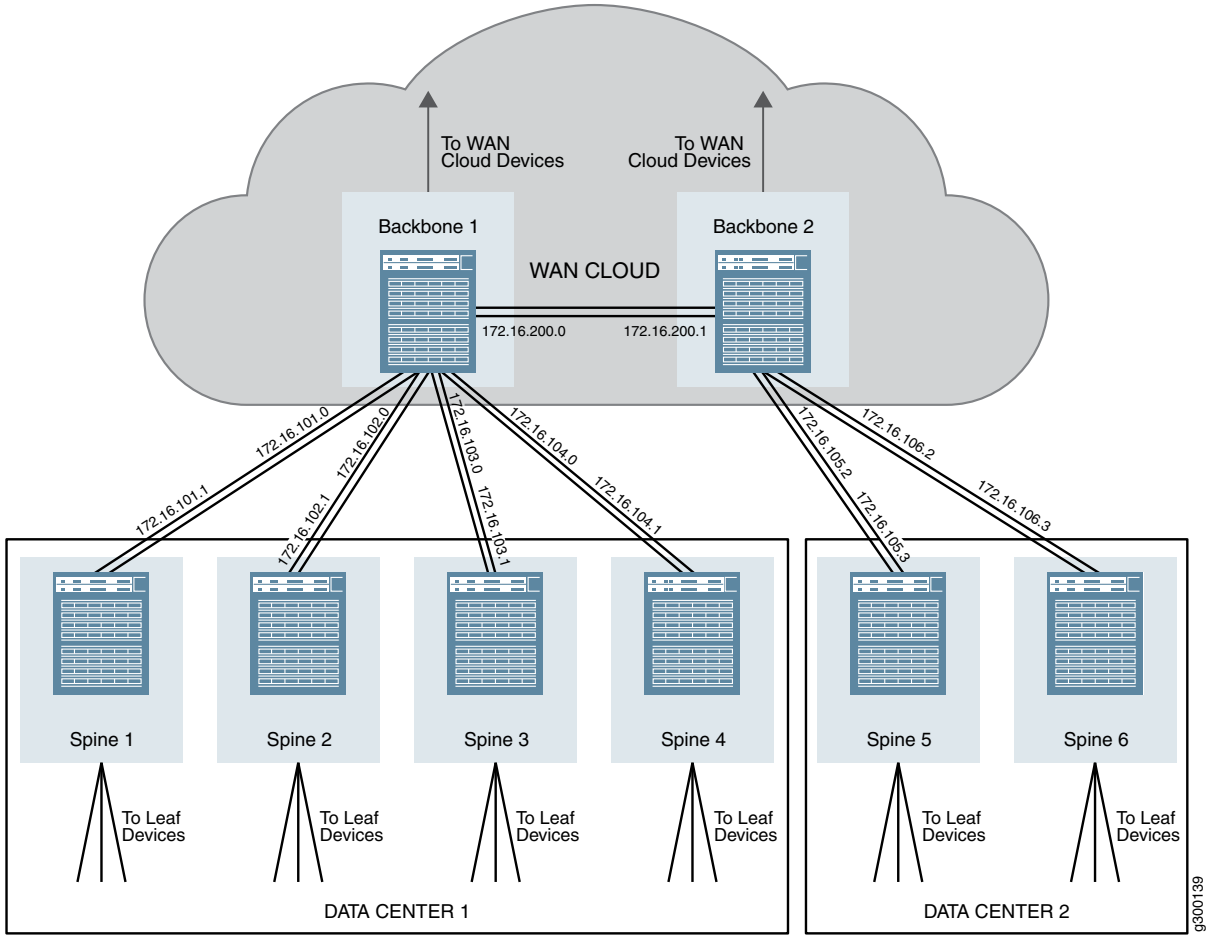
This section covers the processes for configuring a DCI using EVPN Type 5 routes, and includes the following procedures:

Configuring Backbone Device Interfaces

The backbone devices in this architecture are part of the WAN cloud and must provide connectivity both to the spine devices in each data center as well as to the other backbone device. This connectivity must be established before EVPN Type 5 routes can be exchanged between spine devices in different data centers.

Figure 49 on page 197 provides an overview of the IP addresses that are configured in these steps.

Figure 49: IP Address Summary for Backbone and Spine Devices



To configure the spine device and backbone device interfaces:

1. Set up the interfaces and assign IP addresses:

   - (Aggregated Ethernet interfaces) Configure the aggregated Ethernet interfaces on the spine device switches in Data Centers 1 and 2 and on the backbone devices.

     This step shows the assignment of the IP address to the aggregated Ethernet interfaces only. For complete step-by-step instructions on creating aggregated Ethernet interfaces, see Configuring Link Aggregation.

     Spine Device 1 in Data Center 1:

     ```
     set interfaces ae3 unit 0 family inet address 172.16.101.1/31
     ```

     Spine Device 2 in Data Center 1:

     ```
     set interfaces ae3 unit 0 family inet address 172.16.102.1/31
     ```

     Spine Device 3 in Data Center 1:

     ```
     set interfaces ae3 unit 0 family inet address 172.16.103.1/31
     ```

     Spine Device 4 in Data Center 1:

     ```
     set interfaces ae3 unit 0 family inet address 172.16.104.1/31
     ```

     Spine Device 5 in Data Center 2:

     ```
     set interfaces ae4 unit 0 family inet address 172.16.105.3/31
     ```

     Spine Device 6 in Data Center 2:

     ```
     set interfaces ae4 unit 0 family inet address 172.16.106.3/31
     ```

     Backbone Device 1:

     ```
     set interfaces ae1 unit 0 family inet address 172.16.101.0/31
     set interfaces ae2 unit 0 family inet address 172.16.102.0/31
     set interfaces ae3 unit 0 family inet address 172.16.103.0/31
     ```

```
set interfaces ae4 unit 0 family inet address 172.16.104.0/31
set interfaces ae200 unit 0 family inet address 172.16.200.0/31
```

Backbone Device 2:

```
set interfaces ae5 unit 0 family inet address 172.16.105.2/31
set interfaces ae6 unit 0 family inet address 172.16.106.2/31
set interfaces ae200 unit 0 family inet address 172.16.200.1/31
```

- (Standalone interfaces that are not included in aggregated Ethernet interfaces) See Configuring the Interface Address.

Enabling EBGP as the Underlay Network Routing Protocol Between the Spine Devices and the Backbone Devices

EBGP is used as the routing protocol of the underlay network in this reference design. The backbone devices must participate in EBGP with the spine devices to support underlay connectivity.

The process for enabling EBGP on the spine and leaf devices is covered in the "IP Fabric Underlay Network Design and Implementation" on page 51 section of this guide. This procedure assumes EBGP has already been enabled on the spine and leaf devices, although some EBGP configuration on the spine devices needs to be updated to support backbone devices and is therefore included in these steps.

EBGP works in this reference design by assigning each leaf, spine, and backbone device into it's own unique 32-bit autonomous system (AS) number.

Figure 50 on page 200 shows an overview of the EBGP topology for the spine and backbone devices when backbone devices are included in the reference design.

Figure 50: EBGP Topology with Backbone Devices

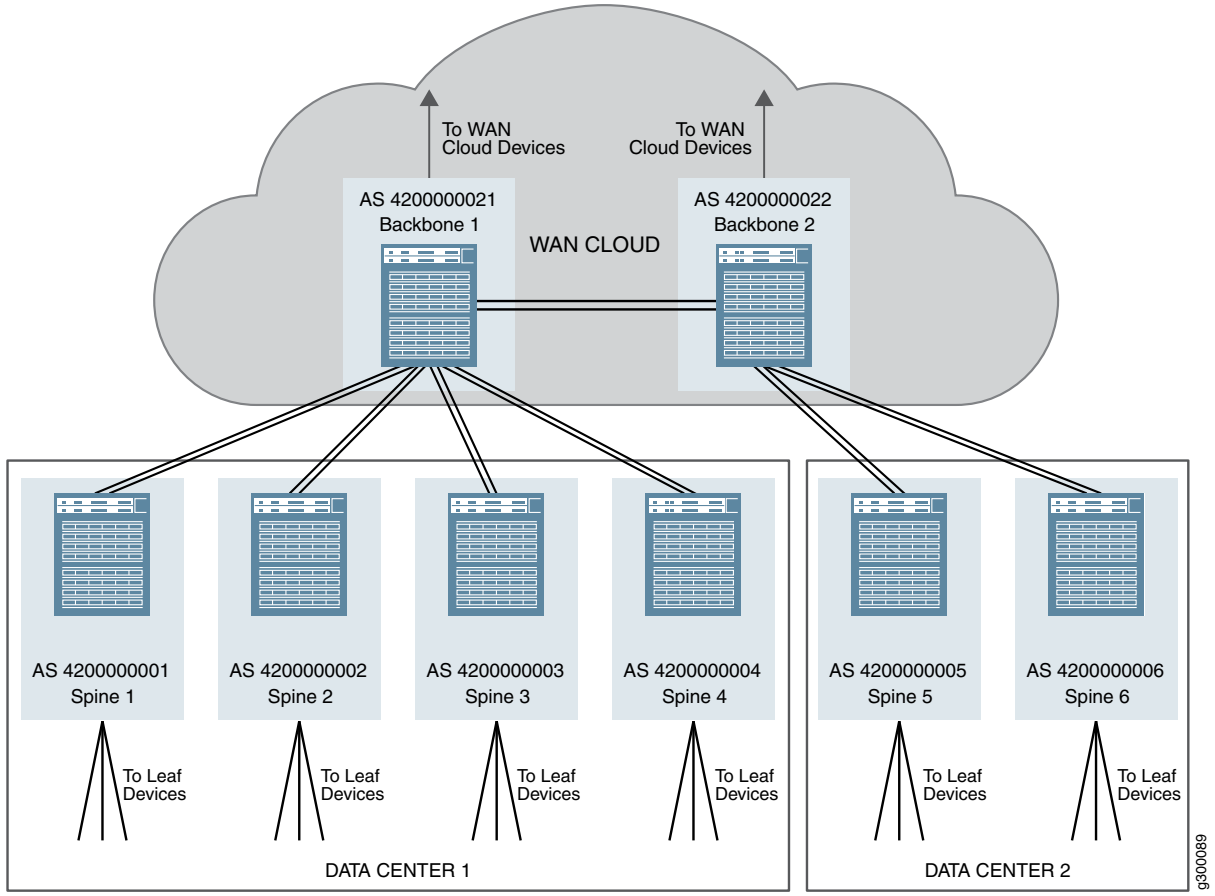
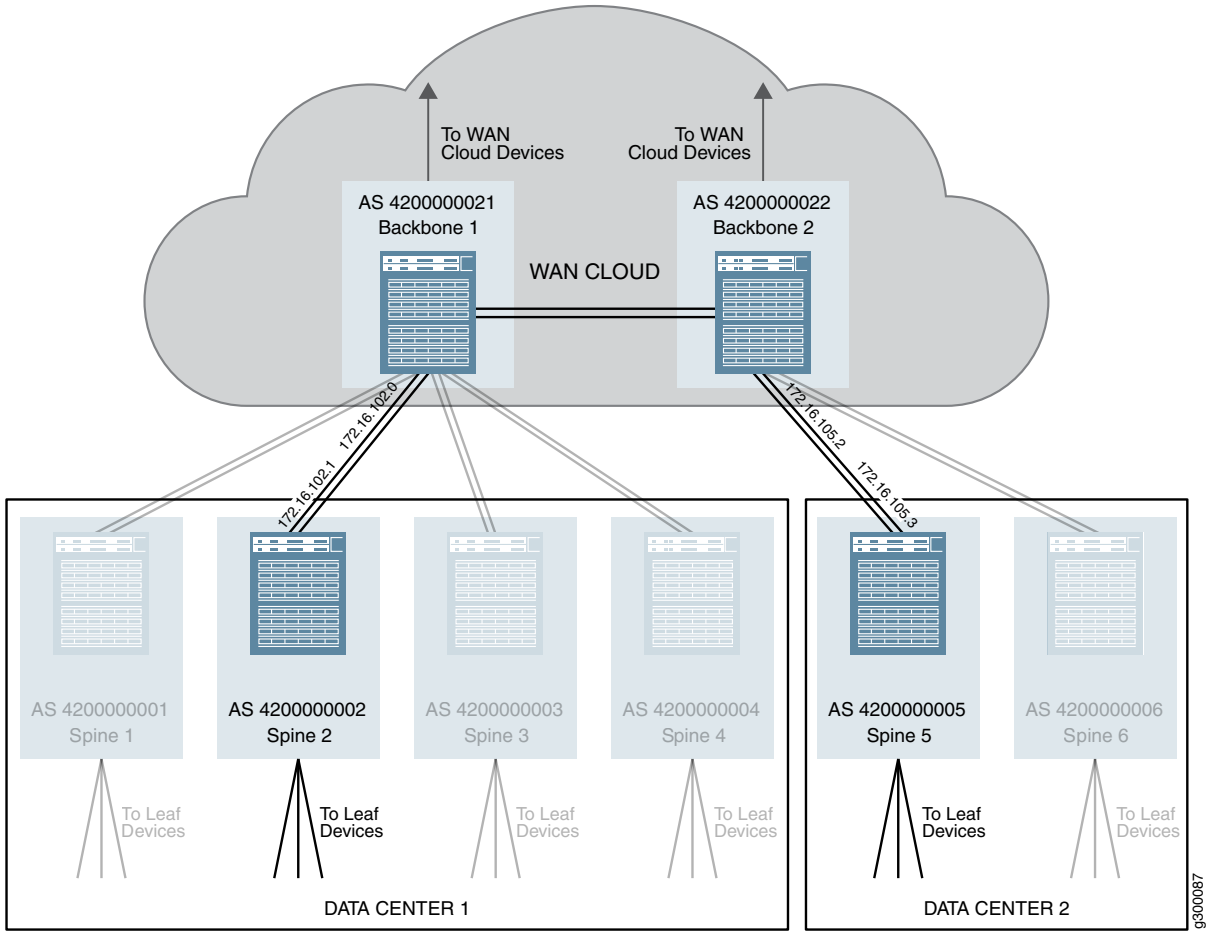


Figure 51 on page 201 illustrates the EBGP protocol parameters that are configured in this procedure. Repeat this process for the other devices in the topology to enable EBGP on the remaining devices.

Figure 51: EBGP Configuration in a Backbone Topology



To enable EBGP to support the underlay network in this reference design:

1. Create and name the BGP peer group. EBGP is enabled as part of this step.

   All Spine and Backbone Devices:

   ```
   set protocols bgp group UNDERLAY-BGP type external
   ```

2. Configure the ASN for each device in the underlay.

   In this reference design, every device is assigned a unique ASN in the underlay network. The ASN for EBGP in the underlay network is configured at the BGP peer group level using the **local-as** statement because the system ASN setting is used for MP-IBGP signaling in the overlay network.

   Spine Device 2 in Data Center 1 Example:

```
set protocols bgp group UNDERLAY-BGP local-as 4200000002
```

Spine Device 5 in Data Center 2 Example:

```
set protocols bgp group UNDERLAY-BGP local-as 4200000005
```

Backbone Device 1:

```
set protocols bgp group UNDERLAY-BGP local-as 4200000021
```

Backbone Device 2:

```
set protocols bgp group UNDERLAY-BGP local-as 4200000022
```

3. Configure BGP peers by specifying the ASN of each BGP peer in the underlay network on each spine and backbone device.

   In this reference design, the backbone devices peer with every spine device in the connected data center and the other backbone device.

   The spine devices peer with the backbone device that connects them into the WAN cloud.

   Spine Device 2 in Data Center 1 Example:

   ```
   set protocols bgp group UNDERLAY-BGP neighbor 172.16.102.0 peer-as 4200000021
   ```

   Spine Device 5 in Data Center 2 Example:

   ```
   set protocols bgp group UNDERLAY-BGP neighbor 172.16.105.2 peer-as 4200000022
   ```

   Backbone Device 1:

   ```
   set protocols bgp group UNDERLAY-BGP neighbor 172.16.101.1 peer-as 4200000001
   set protocols bgp group UNDERLAY-BGP neighbor 172.16.102.1 peer-as 4200000002
   set protocols bgp group UNDERLAY-BGP neighbor 172.16.103.1 peer-as 4200000003
   set protocols bgp group UNDERLAY-BGP neighbor 172.16.104.1 peer-as 4200000004
   ```

   Backbone Device 2:

```
set protocols bgp group UNDERLAY-BGP neighbor 172.16.105.3 peer-as 4200000005
set protocols bgp group UNDERLAY-BGP neighbor 172.16.106.3 peer-as 4200000006
```

4. Create a routing policy that identifies and includes the loopback interface in EBGP routing table updates and apply it.

   This export routing policy is applied and is used to advertise loopback interface reachability to all devices in the IP Fabric in the overlay network.

   Each Spine Device and Backbone Device:

   ```
   set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
   set policy-options policy-statement underlay-clos-export term loopback then accept
   set protocols bgp group UNDERLAY-BGP export underlay-clos-export
   ```

5. Enable multipath to ensure all routes are installed and shared in the forwarding table.

   Each Spine Device and Backbone Device:

   ```
   set protocols bgp group UNDERLAY-BGP multipath multiple-as
   ```

Enabling IBGP for the Overlay Network on the Backbone Device

The backbone devices must run IBGP to have overlay network connectivity and be able to support DCI using EVPN Type 5 routes.

Figure 52 on page 204 shows the IBGP configuration of the validated reference design when backbone devices are included in the topology. In the validated reference design, all spine and leaf devices in the same data center are assigned into the same autonomous system. The backbone devices are assigned into the same autonomous system as the spine and leaf devices of the data center that is using the backbone device as the entry point into the WAN cloud.

Figure 52: IBGP Overview with Backbone Devices

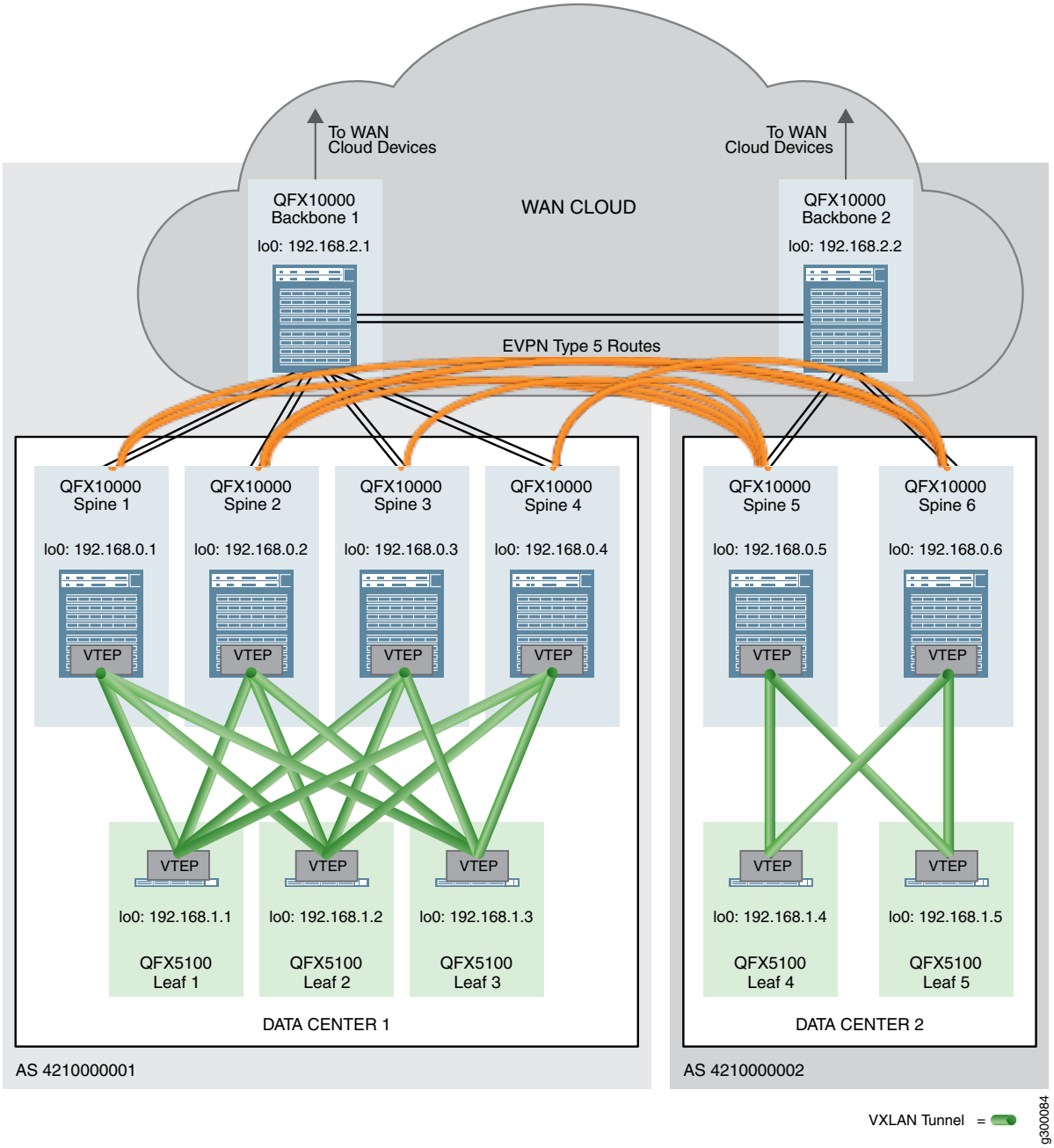
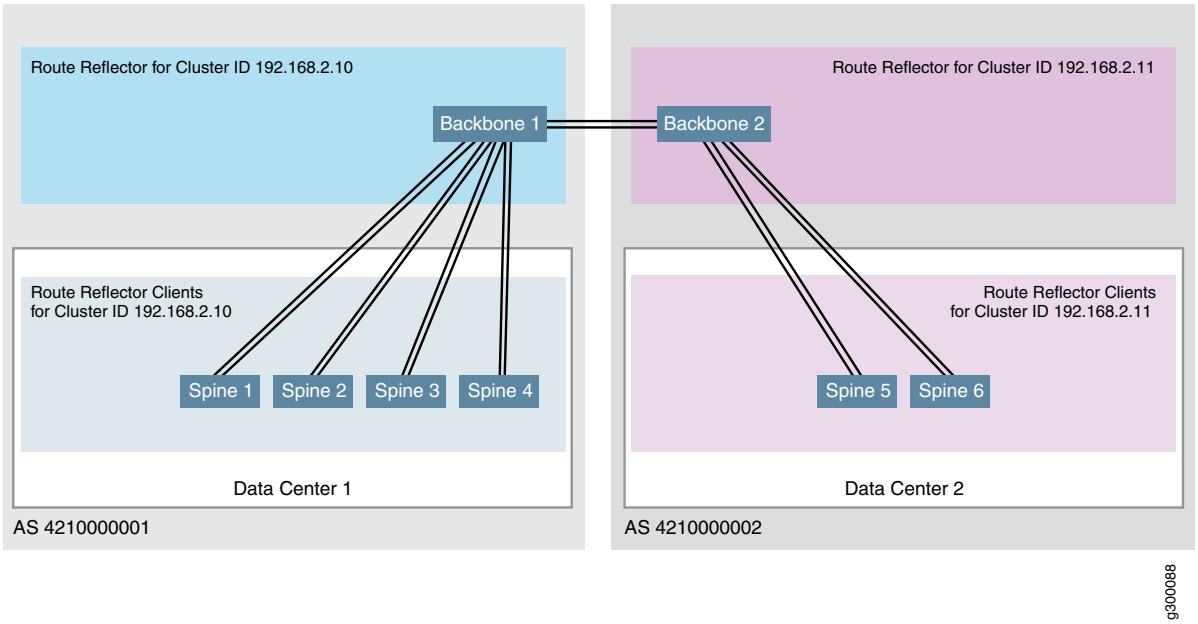


Figure 53 on page 205 illustrates the route reflector configuration in the validated reference design. One route reflector cluster—cluster ID *192.168.2.10*—includes backbone device 1 as the route reflector and all spine devices in data center 1 as route reflector clients. Another route reflector cluster—cluster ID *192.168.2.11*—includes backbone device 2 as the route reflector and all spine devices in data center 2 as route reflector clients.

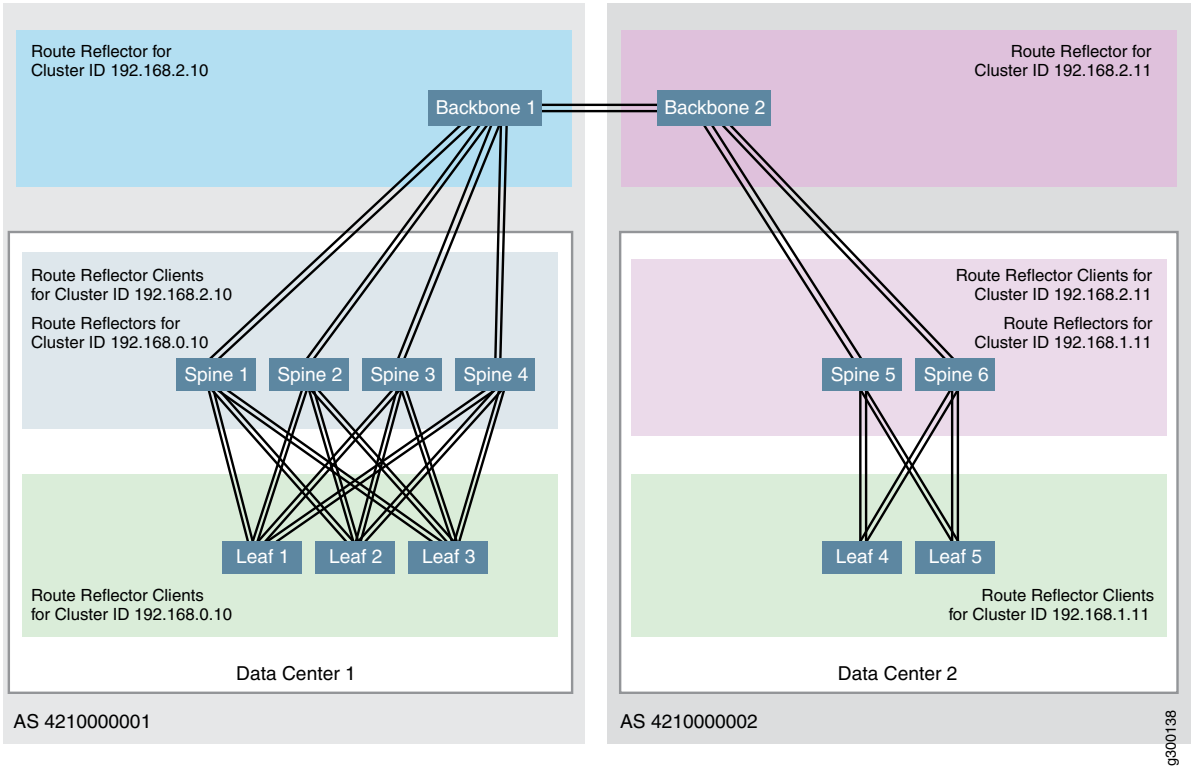
Figure 53: IBGP Route Reflector Topology



The validated reference design supports multiple hierarchical route reflectors, where one cluster includes backbone devices acting as route reflectors for the spine device clients and another cluster includes spine devices acting as route reflectors for leaf device clients. To see the configuration steps for configuring the other route reflector, see “Configuring IBGP for the Overlay” on page 67.

Figure 54 on page 206 shows the full hierarchical route reflector topology when two data centers are connected:

Figure 54: Hierarchical IBGP Route Reflector Topology



For more information on BGP route reflectors, see Understanding BGP Route Reflectors.

This procedure assumes IBGP has been enabled for the spine and leaf devices as detailed in “Configuring IBGP for the Overlay” on page 67. The spine device configurations are included in this procedure to illustrate their relationships to the backbone devices.

To setup IBGP connectivity for the backbone devices:

1. Configure an AS number for overlay IBGP. All leaf and spine devices in the same data center are configured into the same AS. The backbone devices are configured into the same AS as the spine and leaf devices in the data centers using the backbone device as the entry point into the WAN cloud.

   Backbone Device 1 and All Spine and Leaf Devices in Data Center 1:

   ```
   set routing-options autonomous-system 4210000001
   ```

   Backbone Device 2 and All Spine and Leaf Devices in Data Center 2:

   ```
   set routing-options autonomous-system 4210000002
   ```

2. Configure IBGP using EVPN signaling on the backbone devices. Form the route reflector clusters (cluster IDs 192.168.2.10 and 192.168.2.11) and configure BGP multipath and MTU Discovery.

   Backbone Device 1:

   ```
   set protocols bgp group OVERLAY-BGP type internal
   set protocols bgp group OVERLAY-BGP local-address 192.168.2.1
   set protocols bgp group OVERLAY-BGP family evpn signaling
   set protocols bgp group OVERLAY-BGP cluster 192.168.2.10
   set protocols bgp group OVERLAY-BGP mtu-discovery
   set protocols bgp group OVERLAY-BGP multipath
   set protocols bgp group OVERLAY-BGP neighbor 192.168.0.1
   set protocols bgp group OVERLAY-BGP neighbor 192.168.0.2
   set protocols bgp group OVERLAY-BGP neighbor 192.168.0.3
   set protocols bgp group OVERLAY-BGP neighbor 192.168.0.4
   ```

   Backbone Device 2:

   ```
   set protocols bgp group OVERLAY-BGP type internal
   set protocols bgp group OVERLAY-BGP local-address 192.168.2.2
   set protocols bgp group OVERLAY-BGP family evpn signaling
   set protocols bgp group OVERLAY-BGP cluster 192.168.2.11
   set protocols bgp group OVERLAY-BGP mtu-discovery
   set protocols bgp group OVERLAY-BGP multipath
   set protocols bgp group OVERLAY-BGP neighbor 192.168.0.5
   set protocols bgp group OVERLAY-BGP neighbor 192.168.0.6
   ```

3. Configure IBGP using EVPN signaling on the spine devices. Enable BGP multipath and MTU Discovery.

   Spine Device 2 in Data Center 1 Example:

   ```
   set protocols bgp group OVERLAY-BGP-TO-RR type internal
   set protocols bgp group OVERLAY-BGP-TO-RR local-address 192.168.0.2
   set protocols bgp group OVERLAY-BGP-TO-RR mtu-discovery
   set protocols bgp group OVERLAY-BGP-TO-RR family evpn signaling
   set protocols bgp group OVERLAY-BGP-TO-RR vpn-apply-export
   set protocols bgp group OVERLAY-BGP-TO-RR multipath
   set protocols bgp group OVERLAY-BGP-TO-RR neighbor 192.168.2.1
   ```

   Spine Device 5 in Data Center 2 Example:

```
set protocols bgp group OVERLAY-BGP-TO-RR type internal
set protocols bgp group OVERLAY-BGP-TO-RR local-address 192.168.0.5
set protocols bgp group OVERLAY-BGP-TO-RR mtu-discovery
set protocols bgp group OVERLAY-BGP-TO-RR family evpn signaling
set protocols bgp group OVERLAY-BGP-TO-RR vpn-apply-export
set protocols bgp group OVERLAY-BGP-TO-RR multipath
set protocols bgp group OVERLAY-BGP-TO-RR neighbor 192.168.2.2
```

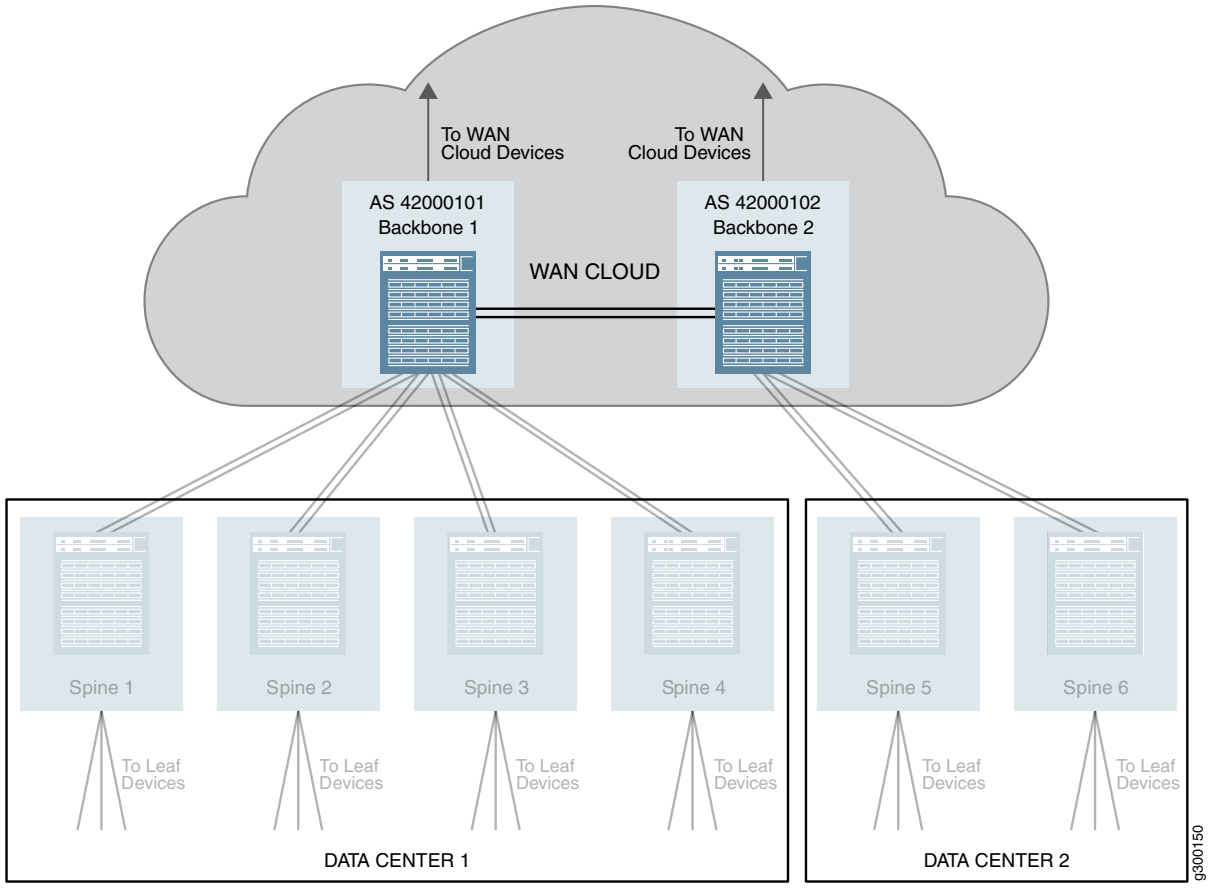
Enabling EBGP as the Routing Protocol Between the Backbone Devices

EBGP is also used as the routing protocol between the backbone devices in this reference design. The backbone devices are connected using IP and the backbone devices must be configured as EBGP peers.

A second EBGP group—*BACKBONE-BGP*—is created in these steps to enable EBGP between the backbone devices. Each backbone device is assigned into a unique 32-bit AS number within the new EBGP group in these steps. The backbone devices, therefore, are part of two EBGP groups—*UNDERLAY-BGP* and *BACKBONE-BGP*—and have a unique AS number within each group. EVPN signaling, which has to run to support EVPN between the backbone devices, is also configured within the EBGP group during this procedure.

Figure 55 on page 209 illustrates the attributes needed to enable EBGP between the backbone devices.

Figure 55: EBGP Topology for Backbone Device Connection



To enable EBGP as the routing protocol between the backbone devices:

1. Create and name the BGP peer group. EBGP is enabled as part of this step.

   Both Backbone Devices:

   ```
   set protocols bgp group BACKBONE-BGP type external
   ```

2. Configure the ASN for each backbone device.

   Backbone Device 1:

   ```
   set protocols bgp group BACKBONE-BGP local-as 4200000101
   ```

   Backbone Device 2:

```
set protocols bgp group BACKBONE-BGP local-as 4200000102
```

3. Configure the backbone devices as BGP peers.

   Backbone Device 1:

   ```
   set protocols bgp group BACKBONE-BGP neighbor 172.16.200.1 peer-as 4200000102
   ```

   Backbone Device 2:

   ```
   set protocols bgp group BACKBONE-BGP neighbor 172.16.200.0 peer-as 4200000101
   ```

4. Enable EVPN signaling between the backbone devices:

   Both Backbone Devices:

   ```
   set protocols bgp group BACKBONE-BGP family evpn signaling
   ```

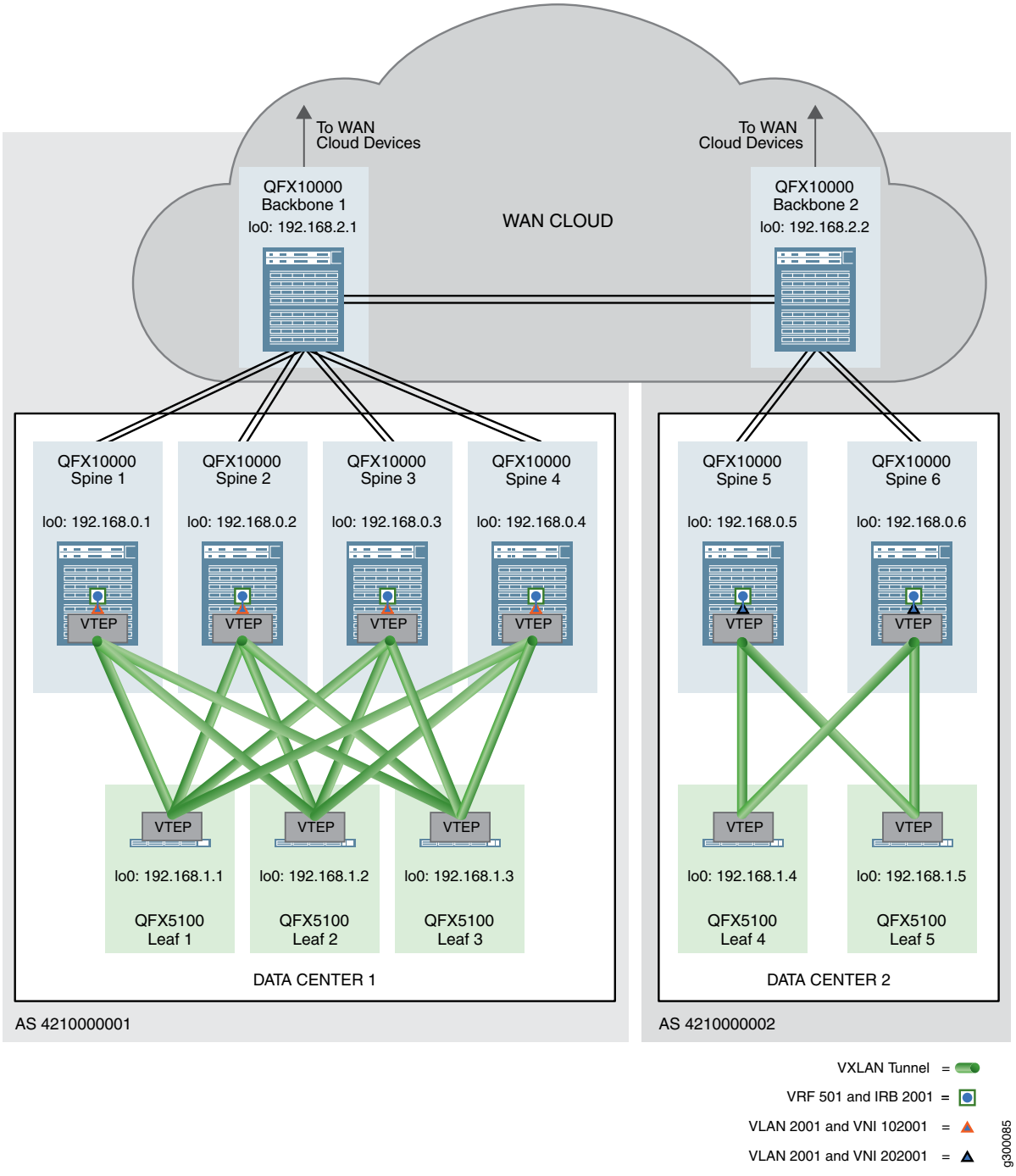
Configuring DCI Using EVPN Type 5 Routes

EVPN Type 5 messages are exchanged between IRB interfaces on spine devices in different data centers when EVPN Type 5 routes are used for DCI. These IRB interfaces are configured in a routing instance.

Each data center has a unique virtual network identifier (VNI 102001 and 202001) in this configuration, but both VNIs are mapped to the same VLAN (VLAN 2001) in the same routing instance (VRF 501).

See Figure 56 on page 211 for an illustration of the routing instance.

Figure 56: DCI Using EVPN Type 5 Routes



To enable DCI using EVPN Type 5 routes:

> **NOTE:** This procedure assumes that the routing instances, IRBs, & VLANs created earlier in this guide are operational. See "Centrally-Routed Bridging Overlay Design and Implementation" on page 95.
>
> When implementing border leaf functionality on an MX router, keep in mind that the router supports virtual switch instances only. MX routers do not support default instances.

1. Configure the preferred addresses of the IRB interfaces.

   Spine Device 2 in Data Center 1:

   ```
   set interfaces irb unit 2001 family inet address 10.20.1.242/24 preferred
   set interfaces irb unit 2001 family inet6 address 2001:db8::10:20:1:242/112 preferred
   ```

   Spine Device 5 in Data Center 2:

   ```
   set interfaces irb unit 2001 family inet address 10.30.1.245/24 preferred
   set interfaces irb unit 2001 family inet6 address 2001:db8::10:30:1:245/112 preferred
   ```

2. Configure mapping between VLANs and the IRB interfaces.

   Spine Device 2 in Data Center 1:

   ```
   set vlans 2001 vlan-id 2001
   set vlans 2001 l3-interface irb.2001
   ```

   Spine Device 5 in Data Center 2:

   ```
   set vlans 2001 vlan-id 2001
   set vlans 2001 l3-interface irb.2001
   ```

3. Configure a routing instance, and map the IRB interface to this instance.

   Spine Device 2 in Data Center 1:

```
set routing-instances VRF-501 instance-type vrf
set routing-instances VRF-501 interface irb.2001
set routing-instances VRF-501 interface lo0.501
set routing-instances VRF-501 route-distinguisher 192.168.0.2:501
set routing-instances VRF-501 vrf-target import target:200:501
set routing-instances VRF-501 vrf-target export target:100:501
set routing-instances VRF-501 routing-options rib VRF-501.inet6.0 multipath
set routing-instances VRF-501 routing-options multipath
```

Spine Device 5 in Data Center 2:

```
set routing-instances VRF-501 instance-type vrf
set routing-instances VRF-501 interface irb.2001
set routing-instances VRF-501 interface lo0.501
set routing-instances VRF-501 route-distinguisher 192.168.0.5:501
set routing-instances VRF-501 vrf-target import target:100:501
set routing-instances VRF-501 vrf-target export target:200:501
set routing-instances VRF-501 routing-options rib VRF-501.inet6.0 multipath
set routing-instances VRF-501 routing-options multipath
```

4. Configure the VRF instance to generate EVPN Type 5 Routes.

> **NOTE:** The VNI of the local or remote data center—VNI 100501 or 200501 in this reference architecture—must be entered as the VNI in the **set routing-instances VRF-501 protocols evpn ip-prefix-routes vni** command.

Spine Device 2 in Data Center 1:

```
set routing-instances VRF-501 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF-501 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF-501 protocols evpn ip-prefix-routes vni 200501
```

Spine Device 5 in Data Center 2:

```
set routing-instances VRF-501 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF-501 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF-501 protocols evpn ip-prefix-routes vni 100501
```

5. On QFX5*xxx* switches that function as spine devices, enable the chained composite next hop feature. With this feature enabled, the switches can more efficiently process large amounts of EVPN Type 5 routes by directing routes that share the same destination to a common forwarding next hop.

   NOTE: On QFX10000 switches, this feature is enabled by default.

   Spine Device 2 in Data Center 1 and Spine Device 5 in Data Center 2:

   ```
   set routing-options forwarding-table chained-composite-next-hop ingress evpn
   ```

Verifying That DCI Using EVPN Type 5 Routes is Operating

Enter the following commands to verify that traffic can be sent between data centers using EVPN Type 5 routes:

1. Verify that an EVPN Type 5 route has been received from the spine device in the other data center by entering the **show route table** command. Enter the VRF instance number and the route distinguisher in the command line to filter the results.

   Spine Device 2 in Data Center 1:

   ```
   user@spine-device-2> show route table VRF-501.evpn.0 match-prefix 5:192.168.0.5:501*

   5:192.168.0.5:501::0::10.30.1.0::24/248
                      *[BGP/170] 07:52:25, localpref 100, from 192.168.2.1
                         AS path: I, validation-state: unverified
                         to 172.16.102.0 via ae3.0
                       > to 172.16.102.2 via ae4.0
   ```

   Spine Device 5 in Data Center 2:

   ```
   user@spine-device-5> show route table VRF-501.evpn.0 match-prefix 5:192.168.0.2:501*

   VRF-501.evpn.0: 33 destinations, 49 routes (33 active, 0 holddown, 0 hidden)
   ```

```
+ = Active Route, - = Last Active, * = Both

5:192.168.0.2:501::0::10.20.1.0::24/248
                    [BGP/170] 07:46:03, localpref 100, from 192.168.2.2
                      AS path: I, validation-state: unverified
                    > to 172.16.105.0 via ae3.0
                      to 172.16.105.2 via ae4.0
```

2. Verify that EVPN Type 5 routes are exported and imported in the VRF instance by entering the **show evpn ip-prefix-database l3-context** command and specifying the VRF instance.

   Spine Device 2 in Data Center 1:

```
user@spine-device-2> show evpn ip-prefix-database l3-context VRF-501
L3 context: VRF-501

IPv4->EVPN Exported Prefixes
Prefix                                  EVPN route status
10.20.1.0/24                            Created


IPv6->EVPN Exported Prefixes
Prefix                                  EVPN route status
2001:db8::10:20:1:0/112                 Created


EVPN->IPv4 Imported Prefixes
Prefix                                  Etag
10.30.1.0/24                             0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.5:501        200501     0c:86:10:cd:bf:f2  192.168.0.5

EVPN->IPv6 Imported Prefixes
Prefix                                  Etag
2000::200:0:1:0/112                     0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.5:501        200501     0c:86:10:cd:bf:f2  192.168.0.5
```

   Spine Device 5 in Data Center 2:

```
user@spine-device-5> show evpn ip-prefix-database l3-context VRF-501
L3 context: VRF-501
```

```
IPv4->EVPN Exported Prefixes
Prefix                                  EVPN route status
10.30.1.0/24                            Created


IPv6->EVPN Exported Prefixes
Prefix                                  EVPN route status
2001:db8::10:30:1:0/112                 Created


EVPN->IPv4 Imported Prefixes
Prefix                                  Etag
10.20.1.0/24                             0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.2:501        200501     54:4b:8c:cd:c4:38  192.168.0.2


EVPN->IPv6 Imported Prefixes
Prefix                                  Etag
2001:db8::10:20:1:0/112                  0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.2:501        200501     54:4b:8c:cd:c4:38  192.168.0.2
```

3. Verify the EVPN Type 5 route encapsulation details by entering the **show route table** command with the **extensive** option.

   Spine Device 2 in Data Center 1:

```
user@spine-device-2> show route table VRF-501.inet.0 match-prefix 10.30.1.0/24 extensive
VRF-501.inet.0: 21 destinations, 37 routes (21 active, 0 holddown, 0 hidden)
10.30.1.0/24 (3 entries, 1 announced)
        State: <CalcForwarding>
TSI:
KRT in-kernel 10.30.1.0/24 -> {list:composite(120781), composite(120780)}
        @EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x2874fc70
                Next-hop reference count: 11
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.102.0 via ae3.0
                Session Id: 0x0
                Next hop: 172.16.102.2 via ae4.0, selected
                Session Id: 0x0
```

```
                Protocol next hop: 192.168.0.5
                Composite next hop: 0x13f31948 120781 INH Session ID: 0x0
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 506
                    Encap VNI: 200501, Decap VNI: 200501
                    Source VTEP: 192.168.0.2, Destination VTEP: 192.168.0.5
                    SMAC: 54:4b:8c:cd:c4:38, DMAC: 0c:86:10:cd:bf:f2
                Indirect next hop: 0xfc5ae80 2101590 INH Session ID: 0x0
                State: <Active Int Ext>
                Age: 7:54:59    Metric2: 0
                Validation State: unverified
                Task: VRF-501-EVPN-L3-context
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.5
                Composite next hops: 1
                        Protocol next hop: 192.168.0.5
                        Composite next hop: 0x13f31948 120781 INH Session ID:
0x0
                          VXLAN tunnel rewrite:
                            MTU: 0, Flags: 0x0
                            Encap table ID: 0, Decap table ID: 506
                            Encap VNI: 200501, Decap VNI: 200501
                            Source VTEP: 192.168.0.2, Destination VTEP:
192.168.0.5
                            SMAC: 54:4b:8c:cd:c4:38, DMAC: 0c:86:10:cd:bf:f2
                        Indirect next hop: 0xfc5ae80 2101590 INH Session ID:
0x0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.102.0 via ae3.0
                                Session Id: 0x0
                                Next hop: 172.16.102.2 via ae4.0
                                Session Id: 0x0
                                192.168.0.5/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Nexthop: 172.16.102.0 via ae3.0
                                        Session Id: 0
                                        Nexthop: 172.16.102.2 via ae4.0
                                        Session Id: 0
```

Spine Device 5 in Data Center 2:

```
user@spine-device-5> show route table VRF-501.inet.0 match-prefix 10.20.1.0/24 extensive

VRF-501.inet.0: 22 destinations, 39 routes (22 active, 0 holddown, 4 hidden)
10.20.1.0/24 (4 entries, 2 announced)
        State: <CalcForwarding>
TSI:
KRT in-kernel 10.20.1.0/24 -> {list:composite(108574), composite(103341)}
Page 0 idx 0, (group Internet type External) Type 1 val 0x283c765c (adv_entry)
   Advertised metrics:
     Nexthop: Self
     AS path: 4210000001 I
     Communities:
Path 10.20.1.0 Vector len 4.  Val: 0
        @EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x32a78cb0
                Next-hop reference count: 9
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.105.0 via ae3.0, selected
                Session Id: 0x0
                Next hop: 172.16.105.2 via ae4.0
                Session Id: 0x0
                Protocol next hop: 192.168.0.2
                Composite next hop: 0xfbc7e68 108574 INH Session ID: 0x0
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 506
                    Encap VNI: 200501, Decap VNI: 200501
                    Source VTEP: 192.168.0.5, Destination VTEP: 192.168.0.2
                    SMAC: 0c:86:10:cd:bf:f2, DMAC: 54:4b:8c:cd:c4:38
                Indirect next hop: 0xfc5d400 2103405 INH Session ID: 0x0
                State: <Active Int Ext>
                Age: 7:49:18    Metric2: 0
                Validation State: unverified
                Task: VRF-501-EVPN-L3-context
                Announcement bits (2): 3-rt-export 4-BGP_RT_Background
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.2
                Composite next hops: 1
                        Protocol next hop: 192.168.0.2
                        Composite next hop: 0xfbc7e68 108574 INH Session ID:
0x0
```

VXLAN tunnel rewrite:

```
        MTU: 0, Flags: 0x0
        Encap table ID: 0, Decap table ID: 506
        Encap VNI: 200501, Decap VNI: 200501
        Source VTEP: 192.168.0.5, Destination VTEP: 192.168.0.2
        SMAC: 0c:86:10:cd:bf:f2, DMAC: 54:4b:8c:cd:c4:38
Indirect next hop: 0xfc5d400 2103405 INH Session ID: 0x0
                   Indirect path forwarding next hops: 2
                           Next hop type: Router
                           Next hop: 172.16.105.0 via ae3.0
                           Session Id: 0x0
                           Next hop: 172.16.105.2 via ae4.0
                           Session Id: 0x0
                           192.168.0.2/32 Originating RIB: inet.0
                             Node path count: 1
                             Forwarding nexthops: 2
                                   Nexthop: 172.16.105.0 via ae3.0
                                   Session Id: 0
                                   Nexthop: 172.16.105.2 via ae4.0
                                   Session Id: 0
```

DCI Using Type 5 Routes — Release History

Table 9 on page 219 provides a history of all of the features in this section and their support within this reference design.

Table 9: DCI Using Type 5 Routes Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section.
18.4R2-S2	QFX5110 and QFX5120-48Y switches, and MX routers running Junos OS Release 18.4R2-S2 and later releases in the same release train support all features documented in this section.

SEE ALSO

Understanding EVPN Pure Type-5 Routes
Configuring IBGP for the Overlay | 67
Centrally-Routed Bridging Overlay Design and Implementation | 95

Data Center Interconnect Design and Implementation Using IPVPN

IN THIS SECTION

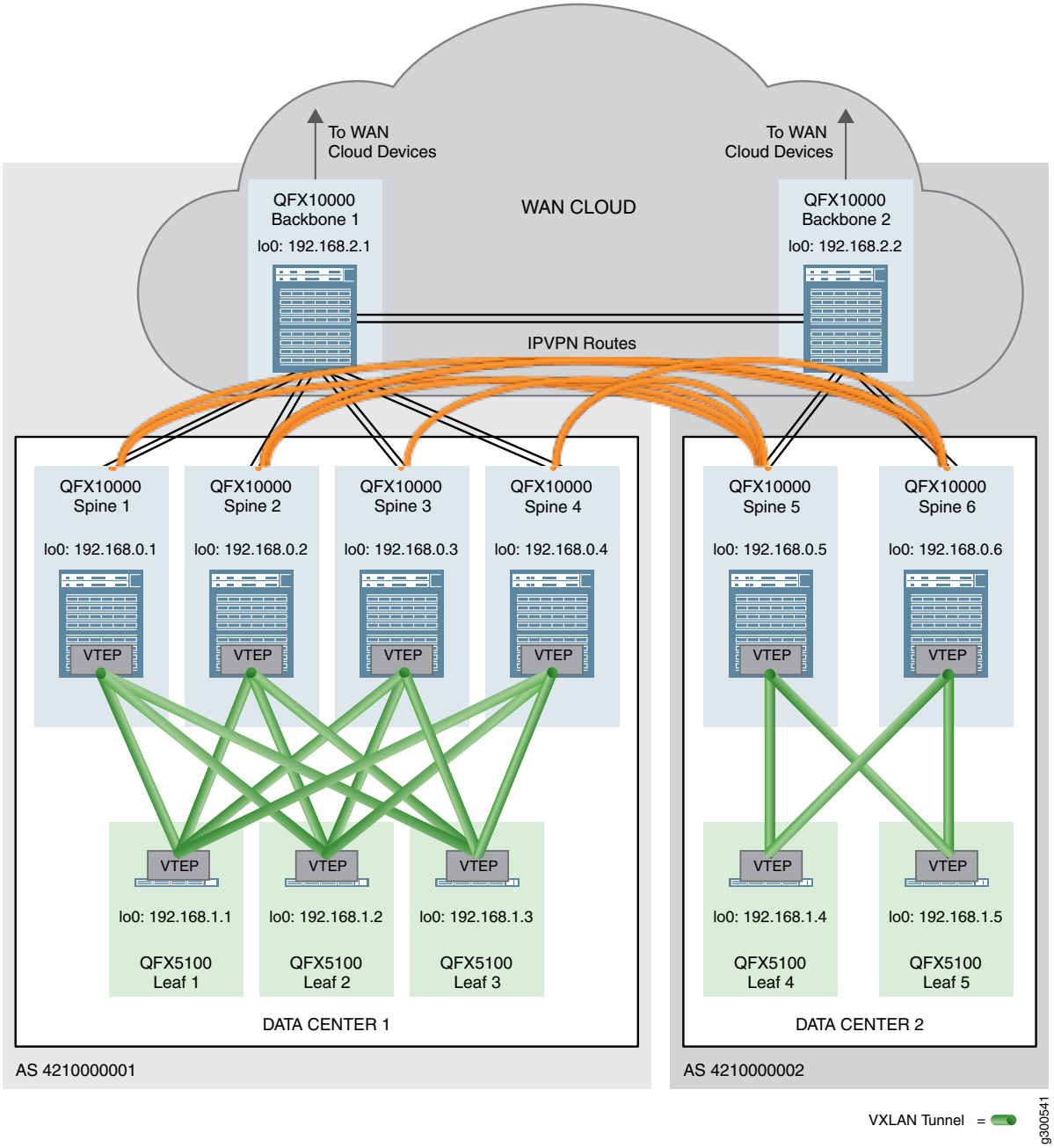
This section describes how to configure DCI using IPVPN. We are using IPVPN to pass traffic between data centers.

In this reference architecture, IPVPN routes are exchanged between spine devices in different data centers to allow for the passing of traffic between data centers.

Physical connectivity between the data centers is required before IPVPN routes can be sent across data centers. The backbone devices in a WAN cloud provide the physical connectivity. A backbone device is connected to each spine device in a single data center and participates in the overlay IBGP and underlay EBGP sessions. EBGP also runs in a separate BGP group to connect the backbone devices to each other; EVPN signaling and IPVPN (inet-vpn) is enabled in this BGP group.

Figure 57 on page 221 shows two data centers using IPVPN for DCI.

Figure 57: Data Center Interconnect using IPVPN



Configuring Data Center Interconnect Using IPVPN

Configuring DCI for IPVPN is similar to configuring DCI for EVPN Type 5 routes with the exceptions shown in this section.

In this example, we are showing the configuration of IPVPN on Spine 1.

1. Configure the underlay link from the spine to the backbone.

```
set interfaces ae4 aggregated-ether-options minimum-links 1
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
set interfaces ae4 unit 0 family inet address 172.16.104.3/31
set interfaces ae4 unit 0 family mpls
```

2. On the backbone device, configure IBGP and MPLS for the overlay network. IPVPN requires that you use MPLS.

```
set protocols bgp group IPVPN-BGP local-address 192.168.0.1
set protocols bgp group IPVPN-BGP neighbor 192.168.2.1 family inet unicast
set protocols bgp group IPVPN-BGP neighbor 192.168.2.1 family inet-vpn unicast
set protocols bgp group IPVPN-BGP neighbor 192.168.2.1 family inet6-vpn unicast
set protocols rsvp interface ae4.0
set protocols rsvp interface lo0.0
set protocols mpls label-switched-path SPINE01-TO-REMOTE-SPINE4 to 192.168.0.4
set protocols mpls label-switched-path SPINE01-TO-REMOTE-SPINE4 no-cspf
set protocols mpls label-switched-path SPINE01-TO-REMOTE-SPINE5 to 192.168.0.5
set protocols mpls label-switched-path SPINE01-TO-REMOTE-SPINE5 no-cspf
set protocols mpls interface ae4.0
set protocols mpls interface lo0.0
```

3. On the spine, configure a routing instance to support DCI using IPVPN routes. This routing instance accepts L3VPN routes and also advertises data center routes as L3VPN routes to other IPVPN provider edge routers.

```
set routing-instances VRF-601 instance-type vrf
set routing-instances VRF-601 interface irb.2401
set routing-instances VRF-601 interface irb.2402
set routing-instances VRF-601 interface irb.2403
set routing-instances VRF-601 interface irb.2404
set routing-instances VRF-601 interface lo0.601
set routing-instances VRF-601 route-distinguisher 192.168.0.1:601
```

```
set routing-instances VRF-601 vrf-target target:200:601
set routing-instances VRF-601 vrf-table-label
set routing-instances VRF-601 routing-options rib VRF-601.inet6.0 multipath
set routing-instances VRF-601 routing-options multipath
```

Verifying Data Center Interconnect Using IPVPN

1. Verify that data center routes are advertised as IPVPN routes to remote data centers.

   host@SPINE-1> **show interfaces terse irb.2401**

```
Interface               Admin Link Proto    Local                 Remote
irb.2401                up    up   inet     30.1.145.244/24
                                            30.1.145.254/24
                                  inet6    2001:db8::30:0:191:244/112
                                           2001:db8::30:0:191:254/112
                                           fe80::e86:1009:61cd:bff2/64
```

   host@SPINE-1> **show route advertising-protocol bgp 192.168.2.1 table VRF-601.inet.0 match-prefix 30.1.145.0 extensive**

```
VRF-601.inet.0: 6091 destinations, 6115 routes (6091 active, 0 holddown, 0
hidden)
* 30.1.145.0/24 (1 entry, 1 announced)
BGP group underlay-bgp type External
     Route Distinguisher: 192.168.0.4:601
     VPN Label: 18
     Nexthop: Self
     Flags: Nexthop Change
     AS path: [4200000004] I
     Communities: target:200:601
```

2. On Spine 4, verify that the remote data center accepts the routes as IPVPN routes.

   host@SPINE-4> **show route table VRF-601.inet.0 match-prefix 30.1.145.0**

```
VRF-601.inet.0: 6447 destinations, 6752 routes (6447 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both
```

```
30.1.145.0/24      *[BGP/170] 1d 01:40:26, localpref 100
                      AS path: 4200000004 I, validation-state: unverified
                    > to 192.16.0.1 via ae4.0, Push 18
                    [BGP/170] 23:48:18, localpref 100
                      AS path: 4200000005 I, validation-state: unverified
                    > to 192.16.0.2 via ae5.0, Push 18
```

Data Center Interconnect—Release History

Table 8 on page 193 provides a history of all of the features in this section and their support within this reference design.

Table 10: DCI Using IPVPN Release History

Release	Description
19.1R2	QFX10002-60C switches running Junos OS Release 19.1R2 and later releases in the same release train support DCI using IPVPN.
18.4R2-S2	MX routers running Junos OS Release 18.4R2-S2 and later releases in the same release train also support DCI using IPVPN.
18.1R3-S5	All devices in the reference design that support Junos OS Release 18.1R3-S5 and later releases in the same release train also support DCI using IPVPN.

Service Chaining Design and Implementation

IN THIS SECTION

For an overview of service chaining, see the Service Chaining section in "Data Center Fabric Blueprint Architecture Components" on page 15.

The following sections show how to implement service chaining and service chaining of multicast in a EVPN VXLAN network.

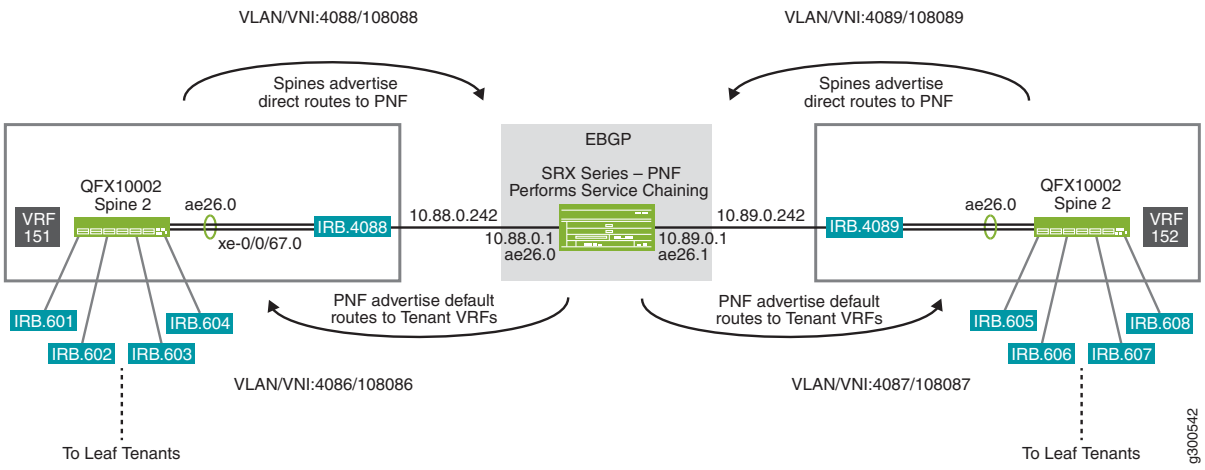
Service Chaining

IN THIS SECTION

Service Chaining Design

Figure 58 on page 225 shows a logical view of the service chaining configuration. It shows the configuration of one spine with a left and right side VRF configuration. The SRX Series router is the PNF, and is performing the Service chaining.

Figure 58: Service Chaining Logical View



The flow of traffic for the spine:

1. Traffic from the leaf tenants enters tenant VRF-151, and is destined to a network attached to tenant VRF-152.

2. Because there is no route from VRF-151 to VRF-152, a default route lookup is performed using routes received from the PNF.

3. After it receives the default route from the PNF, VRF-151 routes traffic toward the PNF via IRB.4088.

4. Traffic reaches the PNF, and the PNF performs the service chaining. Traffic is forwarded out ae26.1 using routes received from EBGP.

5. Traffic now reaches tenant VRF-152, and follows regular route lookup and forwarding action towards the leaf tenants.

Configuring Service Chaining

This section shows how to configure the spine for service chaining as shown in Figure 58 on page 225. This configuration is based on the "Centrally-Routed Bridging Overlay Design and Implementation" on page 95 configuration.

1. Configure the ESI connection towards the PNF.

```
set interfaces xe-0/0/67:0 gigether-options 802.3ad ae26
set interfaces ae26 esi 00:88:88:88:88:88:88:88:88:88
set interfaces ae26 esi all-active
set interfaces ae26 aggregated-ether-options lacp active
set interfaces ae26 aggregated-ether-options lacp periodic fast
set interfaces ae26 aggregated-ether-options lacp system-id 00:00:00:00:00:22
set interfaces ae26 unit 0 family ethernet-switching interface-mode trunk
```

2. Configure the Layer 3 connection for VRF-151 towards the PNF.

```
set interfaces irb unit 4088 virtual-gateway-accept-data
set interfaces irb unit 4088 family inet address 10.88.0.242/24 preferred
set interfaces irb unit 4088 family inet address 10.88.0.242/24 virtual-gateway-address 10.88.0.254
set interfaces irb unit 4088 family inet6 address 2001:db8::10:58:0:242/112 preferred
set interfaces irb unit 4088 family inet6 address 2001:db8::10:58:0:242/112 virtual-gateway-address
    2001:db8::10:58:0:254
set interfaces irb unit 4088 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae26 unit 0 family ethernet-switching vlan members 4088
```

```
set vlans 4088 vlan-id 4088
set vlans 4088 l3-interface irb.4088
set vlans 4088 vxlan vni 104088
```

3. Configure the Layer 3 connection for VRF-152 towards the PNF.

```
set interfaces irb unit 4089 virtual-gateway-accept-data
set interfaces irb unit 4089 family inet address 10.89.0.242/24 preferred
set interfaces irb unit 4089 family inet address 10.89.0.242/24 virtual-gateway-address 10.89.0.254
set interfaces irb unit 4089 family inet6 address 2001:db8::10:59:0:242/112 preferred
set interfaces irb unit 4089 family inet6 address 2001:db8::10:59:0:242/112 virtual-gateway-address
  2001:db8::10:59:0:254
set interfaces irb unit 4089 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae26 unit 0 family ethernet-switching vlan members 4089
set vlans 4089 vlan-id 4089
set vlans 4089 l3-interface irb.4089
set vlans 4089 vxlan vni 104089
```

4. Configure a policy to export local routes from VRF-151 and VRF-152 to the PNF.

```
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 10 from protocol direct
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 10 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 20 from protocol bgp
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 20 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 30 then reject
```

5. Configure the VRF-151 routing instance.

```
set routing-instances VRF-151 instance-type vrf
set routing-instances VRF-151 interface irb.601
set routing-instances VRF-151 interface irb.602
set routing-instances VRF-151 interface irb.603
set routing-instances VRF-151 interface irb.604
set routing-instances VRF-151 interface lo0.151
set routing-instances VRF-151 interface irb.4088
set routing-instances VRF-151 route-distinguisher 192.186.0.2:151
set routing-instances VRF-151 vrf-target target:100:151
set routing-instances VRF-151 protocols bgp group to-srx type external
set routing-instances VRF-151 protocols bgp group to-srx export DIRECT-EXPORT-TO-PNF
set routing-instances VRF-151 protocols bgp group to-srx local-as 4200000002
set routing-instances VRF-151 protocols bgp group to-srx neighbor 10.88.0.1 family inet unicast
```

```
set routing-instances VRF-151 protocols bgp group to-srx neighbor 10.88.0.1 peer-as 4000000001
set routing-instances VRF-151 protocols bgp group to-srx neighbor 2001:db8::10:58:0:1 family inet6 unicast
set routing-instances VRF-151 protocols bgp group to-srx neighbor 2001:db8::10:58:0:1 peer-as 4000000001
```

6. Configure the VRF-152 routing instance.

```
set routing-instances VRF-152 instance-type vrf
set routing-instances VRF-152 interface irb.605
set routing-instances VRF-152 interface irb.606
set routing-instances VRF-152 interface irb.607
set routing-instances VRF-152 interface irb.608
set routing-instances VRF-152 interface lo0.152
set routing-instances VRF-152 interface irb.4089
set routing-instances VRF-152 route-distinguisher 192.186.0.2:152
set routing-instances VRF-152 vrf-target target:100:152
set routing-instances VRF-152 protocols bgp group to-srx type external
set routing-instances VRF-152 protocols bgp group to-srx export DIRECT-EXPORT-TO-PNF
set routing-instances VRF-152 protocols bgp group to-srx local-as 4200000002
set routing-instances VRF-152 protocols bgp group to-srx neighbor 10.89.0.1 family inet unicast
set routing-instances VRF-152 protocols bgp group to-srx neighbor 10.89.0.1 peer-as 4000000001
set routing-instances VRF-152 protocols bgp group to-srx neighbor 2001:db8::10:59:0:1 family inet6 unicast
set routing-instances VRF-152 protocols bgp group to-srx neighbor 2001:db8::10:59:0:1 peer-as 4000000001
```

7. Repeat this procedure for every spine in your topology.

Verifying Service Chaining

To verify that service chaining is working:

1. On the spine, display the local routes in VRF-152.

   host@SPINE-2> **show route table VRF-152.inet.0 protocol direct**

```
VRF-152.inet.0: 61 destinations, 62 routes (61 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.2.93.0/24       *[Direct/0] 1d 15:30:29
                    >  via irb.605
10.2.94.0/24       *[Direct/0] 1d 15:30:29
                    >  via irb.606
10.2.95.0/24       *[Direct/0] 1d 15:30:29
                    >  via irb.607
10.2.96.0/24       *[Direct/0] 1d 15:30:29
                    >  via irb.608
10.87.0.0/24       *[Direct/0] 1d 15:30:37
                    >  via irb.4087
10.89.0.0/24       *[Direct/0] 1d 15:30:37
                    >  via irb.4089
```

2. On the spine, display the local routes in VRF-151.

   host@SPINE2> **show route table VRF-151.inet.0 protocol direct**

```
VRF-151.inet.0: 63 destinations, 64 routes (63 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.2.89.0/24       *[Direct/0] 1d 15:30:36
                    >  via irb.601
10.2.90.0/24       *[Direct/0] 1d 15:30:36
                    >  via irb.602
10.2.91.0/24       *[Direct/0] 1d 15:30:36
                    >  via irb.603
10.2.92.0/24       *[Direct/0] 1d 15:30:36
                    >  via irb.604
10.86.0.0/24       *[Direct/0] 1d 15:30:44
                    >  via irb.4086
10.88.0.0/24       *[Direct/0] 1d 15:30:44
                    >  via irb.4088
```

3. On the spine, check that a route lookup for VRF-151 points to the default route received from PNF.

   host@SPINE2> **show route table VRF-151.inet.0 10.2.93.0**

```
VRF-151.inet.0: 63 destinations, 64 routes (63 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 1d 15:33:08, localpref 100
                      AS path: 4000000001 I, validation-state: unverified
                    >  to 10.88.0.1 via irb.4088
```

4. On the PNF device, check the route lookup toward the tenant VRF-152 destination.

   host@PNF> **show route 10.2.93.0**

```
inet.0: 20 destinations, 52 routes (20 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.2.93.0/24       *[BGP/170] 2d 00:39:44, localpref 100
                      AS path: 4200000002 I, validation-state: unverified
                    > to 10.89.0.242 via ae26.1
```

5. On the PNF device, check that it is advertising only default routes to the EBGP peer on tenant VRF-152.

   host@PNF> **show route advertising-protocol bgp 10.89.0.242**

```
inet.0: 20 destinations, 52 routes (20 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 0.0.0.0/0               Self                                    I
```

6. On the PNF device, check that it is advertising only default routes to the EBGP peer on tenant VRF-151.

   host@PNF> **show route advertising-protocol bgp 10.88.0.242**

```
inet.0: 20 destinations, 52 routes (20 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 0.0.0.0/0               Self                                    I
```

7. On the spine, verify that the spine is advertising local VRF-152 routes to the PNF.

   host@SPINE2> **show route advertising-protocol bgp 10.89.0.1**

```
VRF-152.inet.0: 52 destinations, 53 routes (52 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.2.93.0/24            Self                                    I
* 10.2.94.0/24            Self                                    I
* 10.2.95.0/24            Self                                    I
* 10.2.96.0/24            Self                                    I
* 10.87.0.0/24            Self                                    I
* 10.89.0.0/24            Self                                    I
* 192.68.152.1/32         Self                                    4100000000 I
```

8. On the spine, verify that the spine is advertising local VRF-151 routes to the PNF.

   `host@SPINE2>` **show route advertising-protocol bgp 10.88.0.1**

```
VRF-151.inet.0: 53 destinations, 54 routes (53 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.2.89.0/24            Self                                    I
* 10.2.90.0/24            Self                                    I
* 10.2.91.0/24            Self                                    I
* 10.2.92.0/24            Self                                    I
* 10.86.0.0/24            Self                                    I
* 10.88.0.0/24            Self                                    I
* 192.68.151.1/32         Self                                    4100000000 I
```

Service Chaining with Multicast

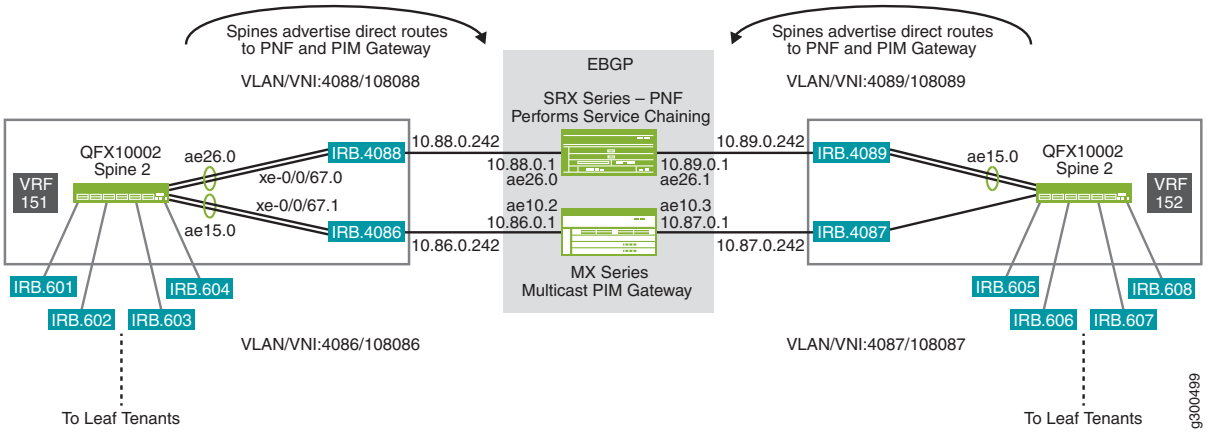
IN THIS SECTION

Service Chaining with Multicast Design

Figure 59 on page 232 shows the logical view of the service chaining configuration. The VRF routing instances and IRB interfaces are all configured on the same spine.

In this design, we are performing service chaining with an SRX Series router as the PNF and with a PIM gateway.

Figure 59: Service Chaining Logical View



The flow of traffic for Spine 2 when multicast receivers start sending IGMP reports:

1. The leaf devices snoop the IGMP report and advertise EVPN Type 6 routes to the spines to notify the spines of the interested Multicast receiver.

2. The spine receives the EVPN Type 6 routes based on the VNI mapping, and creates a PIM join (*.G) entry in VRF-152.

   The spine configuration includes the address of the RP on the PIM gateway. However, on the VRF-152 routing instance, there is only a default route toward the RP via the PNF.

3. The PIM designated router (DR) on the receiver side IRBs (irb.605, irb.606 irb.607, irb.608) sends a PIM join (*.G) towards the PNF device on irb.4089.

4. The PNF device, creates a PIM (*.G) entry with ae26.1 as the outgoing interface.

   The PNF is configured with the RP.

   On the PNF, the RP is also configured on the PNF and the lookup to the PIM RP points toward interface ae26.0 -> interface towards VRF-151 on spines

5. The PIM join arrives on VRF-151 on irb.4088 and creates a PIM (*,G) state with irb.4088 as the outgoing interface and lookup towards the RP points to irb.4086.

6. The Spine sends the PIM join entry on irb.4086 towards the PIM gateway.

7. The PIM gateway receives the PIM (*,G) join on ae10.2.

The flow of traffic for Spine 2 when the multicast source on VRF-151 starts sending packets:

1. VRF-151 creates a PIM (*,G) entry with irb.4088 as the outgoing interface and also the RP reachable via irb.4086.

2. Spine 2 sends two packets—one toward the PNF on irb.4088 and one toward the PIM gateway on irb.4086.

   When the PNF receives the packet:

   a. The PNF forwards it based on the PIM (*.G) entry it created with outgoing interface ae26.1

   b. Multicast traffic arrives on irb.4089 in VRF-512 and is forwarded to the receivers on the leafs.

   When the PIM gateway receives the packet, it forwards the packet based on its (*,G) PIM entry with the outgoing interface as ae10.2.

3. When traffic arrives at irb.4086 from the PIM gateway on Spine 2, Spine 2 prunes the RPT/shared tree.

Configuring Service Chaining With Multicast

This section shows how to configure the spine for service chaining as shown in Figure 59 on page 232.

1. Configure the ESI connection towards the PNF.

```
set interfaces xe-0/0/67:0 gigether-options 802.3ad ae26
set interfaces ae26 esi 00:88:88:88:88:88:88:88:88:88
set interfaces ae26 esi all-active
set interfaces ae26 aggregated-ether-options lacp active
set interfaces ae26 aggregated-ether-options lacp periodic fast
set interfaces ae26 aggregated-ether-options lacp system-id 00:00:00:00:00:22
set interfaces ae26 unit 0 family ethernet-switching interface-mode trunk
```

2. Configure the Layer 3 connection for VRF-151 towards the PNF.

```
set interfaces irb unit 4088 virtual-gateway-accept-data
set interfaces irb unit 4088 family inet address 10.88.0.242/24 preferred
```

```
set interfaces irb unit 4088 family inet address 10.88.0.242/24 virtual-gateway-address 10.88.0.254
set interfaces irb unit 4088 family inet6 address 2001:db8::40:58:0:242/112 preferred
set interfaces irb unit 4088 family inet6 address 2001:db8::40:58:0:242/112 virtual-gateway-address
  2001:db8::40:58:0:254
set interfaces irb unit 4088 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae26 unit 0 family ethernet-switching vlan members 4088
set vlans 4088 vlan-id 4088
set vlans 4088 l3-interface irb.4088
set vlans 4088 vxlan vni 104088
```

3. Configure the Layer 3 connection for VRF-152 towards the PNF.

```
set interfaces irb unit 4089 proxy-macip-advertisement
set interfaces irb unit 4089 virtual-gateway-accept-data
set interfaces irb unit 4089 family inet address 10.89.0.242/24 preferred
set interfaces irb unit 4089 family inet address 10.89.0.242/24 virtual-gateway-address 10.89.0.254
set interfaces irb unit 4089 family inet6 address 2001:db8::10:59:0:242/112 preferred
set interfaces irb unit 4089 family inet6 address 2001:db8::10:59:0:242/112 virtual-gateway-address
  2001:db8::10:59:0:254
set interfaces irb unit 4089 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae26 unit 0 family ethernet-switching vlan members 4089
set vlans 4089 vlan-id 4089
set vlans 4089 l3-interface irb.4089
set vlans 4089 vxlan vni 104089
```

4. Configure the ESI connection towards the PIM gateway.

```
set interfaces xe-0/0/67:1 gigether-options 802.3ad ae15
set interfaces ae15 esi 00:22:22:22:22:22:22:22:22:22
set interfaces ae15 esi all-active
set interfaces ae15 aggregated-ether-options lacp active
set interfaces ae15 aggregated-ether-options lacp periodic fast
set interfaces ae15 aggregated-ether-options lacp system-id 00:00:00:00:00:22
set interfaces ae15 unit 0 family ethernet-switching interface-mode trunk
```

5. Configure the Layer 3 link for VRF-151 towards PIM gateway.

```
set interfaces irb unit 4086 proxy-macip-advertisement
set interfaces irb unit 4086 virtual-gateway-accept-data
set interfaces irb unit 4086 family inet address 10.86.0.242/24 preferred
set interfaces irb unit 4086 family inet address 10.86.0.242/24 virtual-gateway-address 10.86.0.254
```

```
set interfaces irb unit 4086 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae15 unit 0 family ethernet-switching vlan members 4086
set vlans 4086 vlan-id 4086
set vlans 4086 l3-interface irb.4086
set vlans 4086 vxlan vni 104086
```

6. Configure the Layer 3 link for VRF-152 towards the PIM gateway.

```
set interfaces irb unit 4087 proxy-macip-advertisement
set interfaces irb unit 4087 virtual-gateway-accept-data
set interfaces irb unit 4087 family inet address 10.87.0.242/24 preferred
set interfaces irb unit 4087 family inet address 10.87.0.242/24 virtual-gateway-address 10.87.0.254
set interfaces irb unit 4087 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae15 unit 0 family ethernet-switching vlan members 4087
set vlans 4087 vlan-id 4087
set vlans 4087 l3-interface irb.4087
set vlans 4087 vxlan vni 104087
```

7. Configure a policy to export local routes from VRF-151 and VRF-152 to the PNF.

```
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 10 from protocol direct
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 10 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 20 from protocol bgp
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 20 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 30 then reject
```

8. Configure a policy to export local routes from VRF-151 and VRF-152 to the PIM gateway.

```
set policy-options policy-statement DIRECT-EXPORT-TO-PIM term 10 from protocol direct
set policy-options policy-statement DIRECT-EXPORT-TO-PIM term 10 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-PIM term 20 then reject
```

9. Configure a routing instance for VRF 151.

```
set routing-instances VRF-151 instance-type vrf
set routing-instances VRF-151 interface irb.601
set routing-instances VRF-151 interface irb.602
set routing-instances VRF-151 interface irb.603
set routing-instances VRF-151 interface irb.604
set routing-instances VRF-151 interface lo0.151
```

```
set routing-instances VRF-151 interface irb.4088
set routing-instances VRF-151 interface irb.4086
set routing-instances VRF-151 route-distinguisher 192.186.0.2:151
set routing-instances VRF-151 vrf-target target:100:151
set routing-instances VRF-151 protocols bgp group to-srx type external
set routing-instances VRF-151 protocols bgp group to-srx export DIRECT-EXPORT-TO-SRX
set routing-instances VRF-151 protocols bgp group to-srx local-as 4200000002
set routing-instances VRF-151 protocols bgp group to-srx neighbor 10.88.0.1 family inet unicast
set routing-instances VRF-151 protocols bgp group to-srx neighbor 10.88.0.1 peer-as 4000000001
set routing-instances VRF-151 protocols bgp group to-srx neighbor 2001:db8::40:58:0:1 family inet6 unicast
set routing-instances VRF-151 protocols bgp group to-srx neighbor 2001:db8::40:58:0:1 peer-as 4000000001
set routing-instances VRF-151 protocols bgp group to-pim type external
set routing-instances VRF-151 protocols bgp group to-pim export DIRECT-EXPORT-TO-PIM
set routing-instances VRF-151 protocols bgp group to-pim local-as 4200000002
set routing-instances VRF-151 protocols bgp group to-pim neighbor 10.86.0.1 family inet unicast
set routing-instances VRF-151 protocols bgp group to-pim neighbor 10.86.0.1 peer-as 4100000000
set routing-instances VRF-151 protocols pim rp static address 192.68.151.1 group-ranges 225.0.4.0/24
set routing-instances VRF-151 protocols pim rp static address 192.68.152.1 group-ranges 225.0.5.0/24
set routing-instances VRF-151 protocols pim interface irb.601 family inet
set routing-instances VRF-151 protocols pim interface irb.601 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.602 family inet
set routing-instances VRF-151 protocols pim interface irb.602 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.604 family inet
set routing-instances VRF-151 protocols pim interface irb.604 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.603 family inet
set routing-instances VRF-151 protocols pim interface irb.603 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.4086 family inet
set routing-instances VRF-151 protocols pim interface irb.4086 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.4088 family inet
set routing-instances VRF-151 protocols pim interface irb.4088 mode sparse-dense
```

10. Configure a routing instance for VRF 152.

```
set routing-instances VRF-152 instance-type vrf
set routing-instances VRF-152 interface irb.605
set routing-instances VRF-152 interface irb.606
set routing-instances VRF-152 interface irb.607
set routing-instances VRF-152 interface irb.608
set routing-instances VRF-152 interface lo0.152
set routing-instances VRF-152 interface irb.4089
set routing-instances VRF-152 interface irb.4087
set routing-instances VRF-152 route-distinguisher 192.186.0.2:152
set routing-instances VRF-152 vrf-target target:100:152
```

```
set routing-instances VRF-152 protocols bgp group to-srx type external
set routing-instances VRF-152 protocols bgp group to-srx export DIRECT-EXPORT-TO-SRX
set routing-instances VRF-152 protocols bgp group to-srx local-as 4200000002
set routing-instances VRF-152 protocols bgp group to-srx neighbor 10.89.0.1 family inet unicast
set routing-instances VRF-152 protocols bgp group to-srx neighbor 10.89.0.1 peer-as 4000000001
set routing-instances VRF-152 protocols bgp group to-srx neighbor 2001:db8::10:59:0:1 family inet6 unicast
set routing-instances VRF-152 protocols bgp group to-srx neighbor 2001:db8::10:59:0:1 peer-as 4000000001
set routing-instances VRF-152 protocols bgp group to-pim type external
set routing-instances VRF-152 protocols bgp group to-pim export DIRECT-EXPORT-TO-PIM
set routing-instances VRF-152 protocols bgp group to-pim local-as 4200000002
set routing-instances VRF-152 protocols bgp group to-pim neighbor 10.87.0.1 family inet unicast
set routing-instances VRF-152 protocols bgp group to-pim neighbor 10.87.0.1 peer-as 4100000000
set routing-instances VRF-152 protocols pim rp static address 192.68.151.1 group-ranges 225.0.4.0/24
set routing-instances VRF-152 protocols pim rp static address 192.68.152.1 group-ranges 225.0.5.0/24
set routing-instances VRF-152 protocols pim interface irb.605 family inet
set routing-instances VRF-152 protocols pim interface irb.605 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.606 family inet
set routing-instances VRF-152 protocols pim interface irb.606 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.607 family inet
set routing-instances VRF-152 protocols pim interface irb.607 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.608 family inet
set routing-instances VRF-152 protocols pim interface irb.608 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.4087 family inet
set routing-instances VRF-152 protocols pim interface irb.4087 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.4089 family inet
set routing-instances VRF-152 protocols pim interface irb.4089 mode sparse-dense
```

11. Repeat this procedure for every spine in your topology.

Verifying Service Chaining with Multicast

This section shows how to configure service chaining with Multicast. Note the following:

- The multicast source is on VRF-151 - VLAN 601
- The multicast receivers are on VRF-152 - VLAN 606

1. Display the interfaces on VRF-152 that are configured for PIM. This command shows the interfaces that are DRs.

   ```
   host@SPINE5> show pim interfaces instance VRF-152
   ```

   ```
   Stat = Status, V = Version, NbrCnt = Neighbor Count,
   S = Sparse, D = Dense, B = Bidirectional,
   ```

```
DR = Designated Router, DDR = Dual DR, DistDR = Distributed DR,
P2P = Point-to-point link, P2MP = Point-to-Multipoint,
Active = Bidirectional is active, NotCap = Not Bidirectional Capable

Name              Stat Mode IP V State        NbrCnt JoinCnt(sg/*g) DR address
irb.4087          Up   SD    4 2 DR,NotCap         4 0/0           10.87.0.245
irb.4089          Up   SD    4 2 DR,NotCap         4 50/50         10.89.0.245
irb.605           Up   SD    4 2 DR,NotCap         3 0/0           10.2.93.245
irb.606           Up   SD    4 2 DR,NotCap         3 0/0           10.2.94.245
irb.607           Up   SD    4 2 DR,NotCap         3 0/0           10.2.95.245
irb.608           Up   SD    4 2 DR,NotCap         3 0/0           10.2.96.245
pime.32780         Up   S     4 2 P2P,NotCap        0 0/0
pime.32781         Up   S     4 2 P2P,NotCap        0 0/0
```

2. Display PIM groups on the Upstream interface towards RP and Source are pointing to irb.4089 – towards the PNF

host@SPINE5> **show pim join extensive instance VRF-152 225.0.4.1**

```
Instance: PIM.VRF-152 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 225.0.4.1
    Source: *
    RP: 192.68.151.1
    Flags: sparse,rptree,wildcard
    Upstream interface: irb.4089
    Upstream neighbor: 10.89.0.1
    Upstream state: Join to RP
    Uptime: 00:14:30
    Downstream neighbors:
        Interface: irb.606
            10.2.94.245 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 00:14:30 Time since last Join: 00:14:30
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1

Group: 225.0.4.1
    Source: 10.2.89.5
    Flags: sparse,spt
    Upstream interface: irb.4089
    Upstream neighbor: 10.89.0.1
    Upstream state: Join to Source, No Prune to RP
```

```
    Keepalive timeout: 317
    Uptime: 00:13:51
    Downstream neighbors:
        Interface: irb.606
            10.2.94.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:13:51 Time since last Join: 00:13:51
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
```

Service Chaining— Release History

Table 11 on page 239 provides a history of all of the features in this section and their support within this reference design.

Table 11: Service Chaining Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section.
18.1R3-S3	All devices in the reference design that support Junos OS Release 18.1R3-S3 and later releases in the same release train also support all features documented in this section.

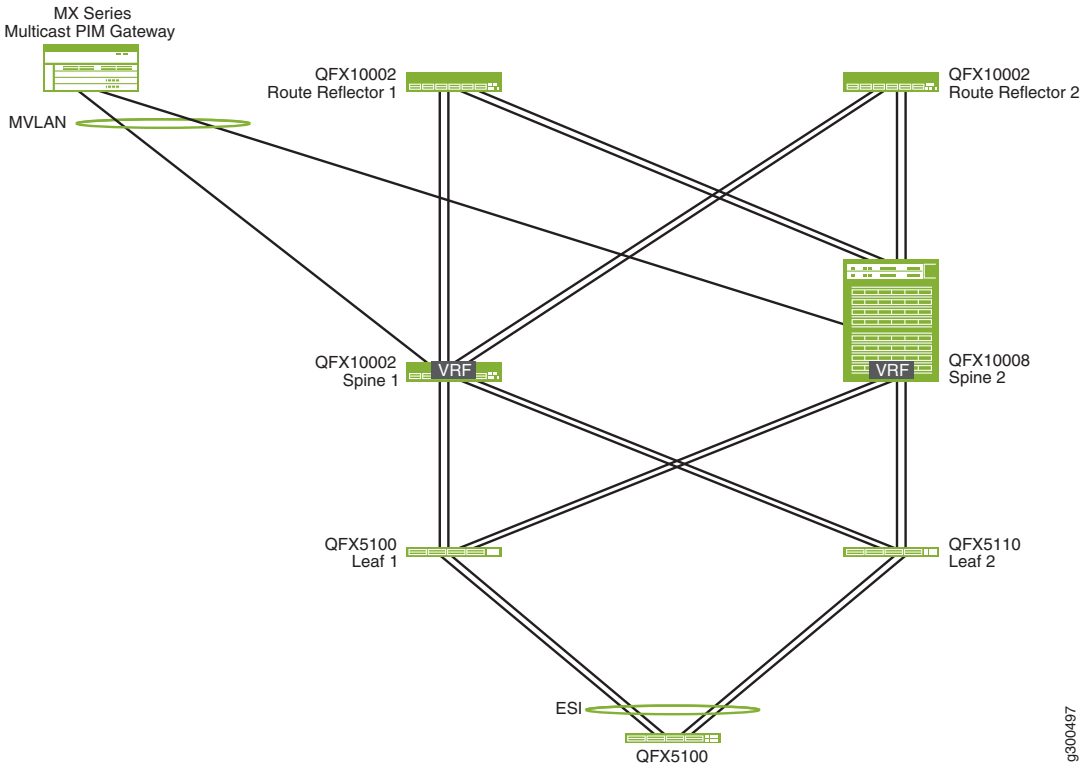
Multicast IGMP Snooping and PIM Design and Implementation

IN THIS SECTION

Use this design to configure Internet Group Management Protocol (IGMP) and Physical Interface Module (PIM) in your fabric to improve multicast replication. IGMP snooping preserves bandwidth because multicast traffic is forwarded only on interfaces where there are IGMP listeners. For instance, every leaf device does not need to receive every instance of multicast traffic.

For an overview of multicast, see “Multicast Optimizations” on page 31

In this design, we are using an external PIM gateway, which extends multicast beyond the data center, and is useful in DCI implementations.



The following sections show how to configure and verify multicast:

Configuring IGMP Snooping

In this design, we are using IGMP snooping to constrain multicast traffic in a broadcast domain to interested receivers and multicast devices.

To configure IGMP snooping:

1. Configure IGMP snooping on all VXLAN enabled VLANs on the leafs. The current implementation does not support IGMP snooping on selected VXLAN enabled VLANs.

```
set protocols igmp-snooping vlan BD-3 proxy
set vlans BD-3 vlan-id 3
set vlans BD-3 vxlan vni 100003
```

Verifying IGMP Snooping

1. Verify the local IGMP snooping state on the leaf.

```
user@leaf-2> show igmp snooping membership vlan BD-3
```

```
Instance: default-switch

Vlan: BD-3

Learning-Domain: default
Interface: ae11.0, Groups: 2
    Group: 225.0.1.1
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.0.3.7
        Group timeout:     215 Type: Dynamic
    Group: 225.0.1.2
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.0.3.7
        Group timeout:     207 Type: Dynamic
```

```
user@leaf-2> show evpn igmp-snooping database l2-domain-id 100003
```

```
Instance: default-switch
    VN Identifier: 100003
```

```
        Group IP: 225.0.1.1, Source IP: 0.0.0.0, Access OIF Count: 1
        Group IP: 225.0.1.2, Source IP: 0.0.0.0, Access OIF Count: 1
```

2. Verify that the leaf is advertising the EVPN Type 7 route where IGMP is snooped.

   `user@leaf-2>` **show route table __default_evpn__.evpn.0 match-prefix 7:*100003*225.0.1.[12]***

```
__default_evpn__.evpn.0: 215 destinations, 318 routes (215 active, 0 holddown,
 0 hidden)
+ = Active Route, - = Last Active, * = Both

7:192.168.1.4:10::99080706050403::100003::225.0.1.1::192.168.1.4/600

                   *[EVPN/170] 04:55:32
                       Indirect
7:192.168.1.4:10::99080706050403::100003::225.0.1.2::192.168.1.4/600

                   *[EVPN/170] 04:55:30
                       Indirect
```

3. Verify that the leaf and its multihomed ESI peer device are both advertising the EVPN Type 6 route for the multicast group.

   `user@leaf-2>` **show route table default-switch.evpn.0 match-prefix 6:*100003*225.0.1.[12]***

```
default-switch.evpn.0: 100334 destinations, 198153 routes (100334 active, 0
holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.1.3:10::100003::225.0.1.1::192.168.1.3/520
                   *[BGP/170] 04:55:09, localpref 100, from 192.168.0.4
                     AS path: I, validation-state: unverified
                    >  to 172.16.4.1 via ae1.0
                       to 172.16.4.3 via ae2.0
                    [BGP/170] 04:55:11, localpref 100, from 192.168.0.5
                     AS path: I, validation-state: unverified
                    >  to 172.16.4.1 via ae1.0
                       to 172.16.4.3 via ae2.0
6:192.168.1.3:10::100003::225.0.1.2::192.168.1.3/520
                   *[BGP/170] 04:55:07, localpref 100, from 192.168.0.4
                     AS path: I, validation-state: unverified
                    >  to 172.16.4.1 via ae1.0
```

```
                        to 172.16.4.3 via ae2.0
                      [BGP/170] 04:55:10, localpref 100, from 192.168.0.5
                        AS path: I, validation-state: unverified
                      >  to 172.16.4.1 via ae1.0
                        to 172.16.4.3 via ae2.0
6:192.168.1.4:10::100003::225.0.1.1::192.168.1.4/520
                    *[EVPN/170] 04:56:16
                        Indirect
6:192.168.1.4:10::100003::225.0.1.2::192.168.1.4/520
                    *[EVPN/170] 04:56:14
                        Indirect
```

Configuring PIM

1. To configure inter-VNI multicast routing at the spine, create a routing instance for a tenant (a leaf device) named VRF-1. Configure the following in the routing instance:

   - Add the IRB interfaces to the leaf devices.
   - Enable PIM and configure the local address for this spine as the rendezvous point (RP).
   - Enable PIM on the IRB interfaces.

   Spine 1:

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 interface irb.1
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface lo0.10
set routing-instances VRF-1 route-distinguisher 192.186.0.2:1
set routing-instances VRF-1 vrf-target target:100:1
set routing-instances VRF-1 protocols pim rp local address 10.0.1.242
set routing-instances VRF-1 protocols pim interface irb.1 family inet
set routing-instances VRF-1 protocols pim interface irb.1 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.2 family inet
set routing-instances VRF-1 protocols pim interface irb.2 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.3 family inet
set routing-instances VRF-1 protocols pim interface irb.3 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.4 family inet
set routing-instances VRF-1 protocols pim interface irb.4 mode sparse-dense
```

2. Configure multicast routing on another spine. Configure a corresponding VRF routing instance for the same tenant as in step 6.

   - Add the IRB interfaces toward the leaf devices.
   - Enable PIM and configure the RP address on spine 1 as the static RP.
   - Enable PIM on the IRB interfaces.

   Spine 2:

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 interface irb.1
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface lo0.10
set routing-instances VRF-1 route-distinguisher 192.168.0.3:1
set routing-instances VRF-1 vrf-target target:100:1
set routing-instances VRF-1 protocols pim rp static address 10.0.1.242
set routing-instances VRF-1 protocols pim interface irb.1 family inet
set routing-instances VRF-1 protocols pim interface irb.1 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.2 family inet
set routing-instances VRF-1 protocols pim interface irb.2 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.3 family inet
set routing-instances VRF-1 protocols pim interface irb.3 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.4 family inet
set routing-instances VRF-1 protocols pim interface irb.4 mode sparse-dense
```

Verifying PIM

1. On spine 1, check the PIM control plane on the RP, and verify that:

   - PIM joins are created from Type 6 routes that are generated by leaf devices.
   - IGMP reports are coming from non-IGMP snooping capable leaf devices.

   `user@spine-1>` **show pim join instance VRF-1 225.0.1.0/30 extensive**

```
Instance: PIM.VRF-1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 225.0.1.1
    Source: *
```

```
    RP: 10.0.1.242
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 15:18:24
    Downstream neighbors:
        Interface: irb.3
            10.0.3.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 15:17:51 Time since last Join: 15:17:51
        Interface: irb.2
            10.0.2.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 14:51:29 Time since last Join: 14:51:29
        Interface: irb.1
            10.0.1.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 05:31:09 Time since last Join: 05:31:09
            10.0.1.245 State: Join Flags: SRW Timeout: 199
            Uptime: 15:17:28 Time since last Join: 00:00:11
    Number of downstream interfaces: 3
    Number of downstream neighbors: 4

Group: 225.0.1.2
    Source: *
    RP: 10.0.1.242
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 15:18:24
    Downstream neighbors:
        Interface: irb.3
            10.0.3.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 15:17:51 Time since last Join: 15:17:51
        Interface: irb.2
            10.0.2.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 14:51:29 Time since last Join: 14:51:29
        Interface: irb.1
            10.0.1.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 05:31:09 Time since last Join: 05:31:09
            10.0.1.245 State: Join Flags: SRW Timeout: 199
            Uptime: 15:17:28 Time since last Join: 00:00:11
    Number of downstream interfaces: 3
    Number of downstream neighbors: 4
```

2. On the spine that is configured as the PIM DR, verify the multicast forwarding state from the spine to the tenant VRF. To do so:

   a. Enter **show pim interfaces instance** on all spines, and check the State column to see which IRB interface is a DR.

   `user@spine-1>` **show pim interfaces instance VRF-1**

```
Stat = Status, V = Version, NbrCnt = Neighbor Count,
S = Sparse, D = Dense, B = Bidirectional,
DR = Designated Router, DDR = Dual DR, DistDR = Distributed DR,
P2P = Point-to-point link, P2MP = Point-to-Multipoint,
Active = Bidirectional is active, NotCap = Not Bidirectional Capable

Name              Stat Mode IP V State       NbrCnt JoinCnt(sg/*g) DR address
irb.1              Up   SD    4 2 DR,NotCap        3 2/100         10.0.1.245
irb.2              Up   SD    4 2 DR,NotCap        3 0/0           10.0.2.245
irb.3              Up   SD    4 2 DR,NotCap        3 0/0           10.0.3.245
irb.4              Up   SD    4 2 DR,NotCap        3 98/0          10.0.4.245
pime.32769          Up   S     4 2 P2P,NotCap          0 0/0
irb.1               Up   SD    6 2 DR,NotCap           2 0/0
fe80:db8:10:0:1::254
irb.2               Up   SD    6 2 DR,NotCap           2 0/0
fe80:db8:10:0:2::254
irb.3               Up   SD    6 2 DR,NotCap           2 0/0
fe80:db8:10:0:3::254
irb.4               Up   SD    6 2 DR,NotCap           2 0/0
fe80:db8:10:0:4::254
```

   b. On the PIM DR, display the multicast forwarding state.

   `user@spine-2>` **show multicast route extensive instance VRF-1 group 225.0.1.0/30**

```
Instance: VRF-1 Family: INET

Group: 225.0.1.1
    Source: 10.0.1.5/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.3 irb.2
    Number of outgoing interfaces: 3
    Session description: Unknown
    Statistics: 69 kBps, 559 pps, 309165 packets
    Next-hop ID: 2109194
```

```
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: 360 seconds
    Wrong incoming interface notifications: 0
    Uptime: 00:09:13

Group: 225.0.1.2
    Source: 10.0.1.5/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.3 irb.2
    Number of outgoing interfaces: 3
    Session description: Unknown
    Statistics: 68 kBps, 554 pps, 307024 packets
    Next-hop ID: 2109194
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: 360 seconds
    Wrong incoming interface notifications: 0
    Uptime: 00:09:13
```

Multicast — Feature Summary

Table 12 on page 247 provides a history of the features described in this section and their support within this reference design.

Table 12: Multicast Feature Summary

Hardware	IGMPv2 Snooping	EVPN Type 6 SMET Routes	Inter-VNI Multicast with PIM Gateway	PIM to External Rendezvous Point (From Border)
QFX5100[1]	Not supported	Not supported	Not supported	Not supported
QFX5110-32Q, QFX5110-48S	18.1R3-S3	18.4R2	Not supported	Not supported
QFX5120-48Y	18.4R2	18.4R2	Not supported	Not supported

Table 12: Multicast Feature Summary *(continued)*

Hardware	IGMPv2 Snooping	EVPN Type 6 SMET Routes	Inter-VNI Multicast with PIM Gateway	PIM to External Rendezvous Point (From Border)
QFX5120-32C	19.1R2	19.1R2	Not supported	Not supported
QFX5200-32C[1], QFX5200-48Y[1]	Not supported	Not supported	Not supported	Not supported
QFX10002-36Q/72Q, QFX10008, QFX10016	18.1R3-S3	18.4R2	18.1R3-S3	17.3R3-S1
QFX10002-60C	20.2R1	20.2R1	20.2R1	20.2R1
MX204; MX240, MX480, MX960 with MPC7E; MX10003;	Not supported	Not supported	Not supported	Not supported

[1]Make sure that IGMP snooping is not enabled on these QFX switches. If IGMP snooping is inadvertently enabled, these switches might process EVPN Type 6 routes that are reflected to them.

RELATED DOCUMENTATION

Overview of Multicast Forwarding with IGMP Snooping in an EVPN-VXLAN Environment

Multicast Support in EVPN-VXLAN Overlay Networks

Multicast Optimization Design and Implementation

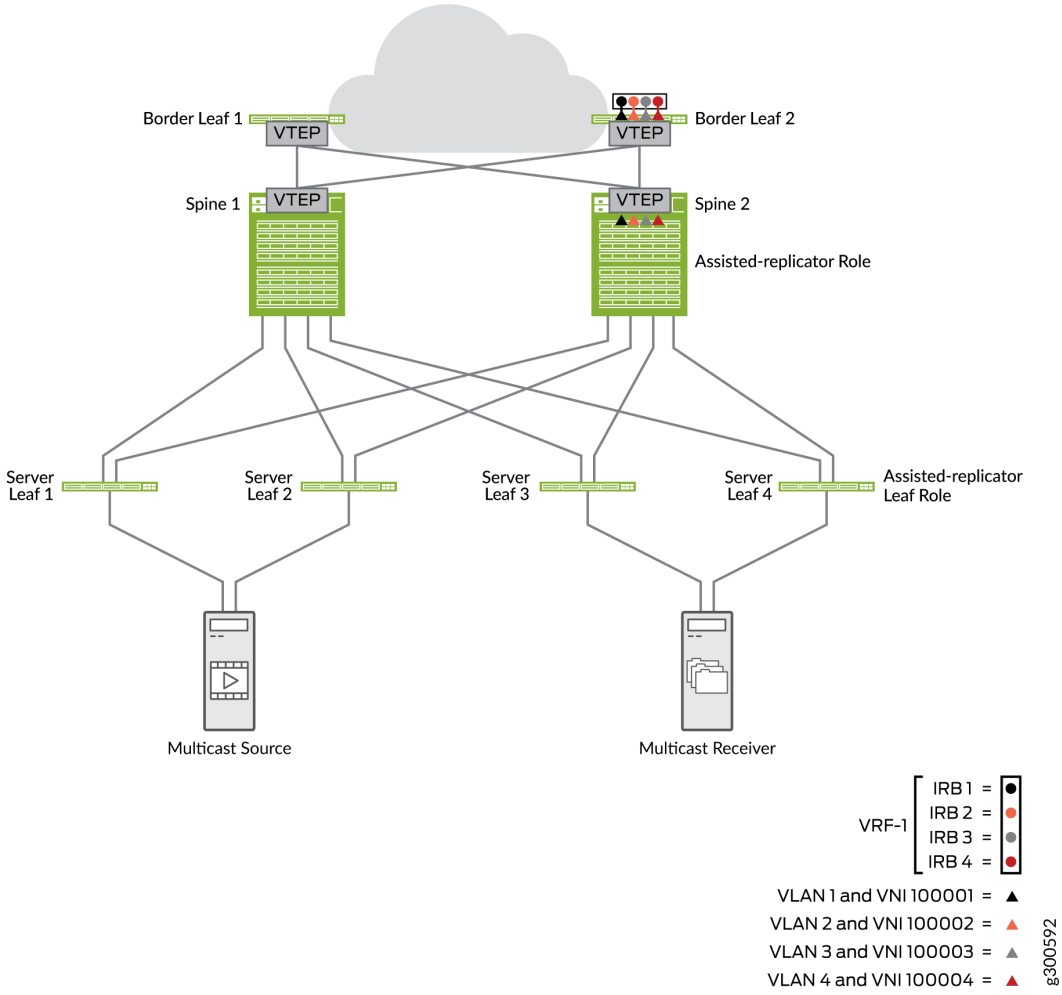
IN THIS SECTION

Juniper Networks supports the multicast optimization features discussed in this section in both centrally-routed and edge-routed bridging overlays.

This design assumes that an EVPN-VXLAN edge-routed bridging overlay is already running for IPv4 unicast traffic. (See "Edge-Routed Bridging Overlay Design and Implementation" on page 158 for information about configuring edge-routed bridging.) However, in terms of setting up the optimization features for multicast traffic, note that this design uses a centrally routed approach.

This section shows how to add centrally-routed multicast optimizations to the edge-routed bridging topology shown in Figure 15 on page 35.

Figure 60: Topology for Multicast Optimizations in an Edge-Routed Bridging Overlay



Multicast is configured as follows:

- Server leaf devices are set up in AR leaf role and for IGMP snooping.
- Spine devices are set up in AR replicator role.
- Border leaf devices are set up for multicast routing.

NOTE: If your multicast environment requires assisted replication to handle large multicast flows and multicast routing, we recommend any of the QFX10000 line of switches for the border leaf and spine roles. We do not recommend any of the MX Series routers included in this reference design as a border leaf in a multicast environment with large multicast flows.

For an overview of multicast optimizations, see the Multicast Optimization section in "Data Center Fabric Blueprint Architecture Components" on page 15.

The following sections show how to configure and verify multicast assisted replication:

Configuring the Server Leaf

We are configuring AR and IGMP snooping on the server leaf. When IGMP snooping is enabled on a device, SMET is also enabled on the device by default.

1. Enable IGMP snooping.

```
set protocols igmp-snooping vlan BD-1
set protocols igmp-snooping vlan BD-2
set protocols igmp-snooping vlan BD-3
set protocols igmp-snooping vlan BD-4
```

2. Enable AR in the leaf role. This causes the server leaf to only forward one copy of multicast traffic to the spine, which then performs replication of the multicast traffic.

   The **replicator-activation-delay** is the time, in seconds, the leaf waits before sending the replication to the AR replicator after receiving the AR replicator route from the replicator.

```
set protocols evpn assisted-replication leaf replicator-activation-delay 10
```

Configuring the Spine

We are configuring the spine as AR replicator device.

1. Configure IP addressing for the loopback interfaces. One address is used for the AR replicator role (192.168.102.2). The other address (192.168.2.2) is used for the VTEP tunnel.

```
set interfaces lo0 unit 0 family inet address 192.168.2.2/32 preferred
set interfaces lo0 unit 0 family inet address 192.168.102.2/32
```

2. Configure the spine to act as the AR replicator device.

```
set protocols evpn assisted-replication replicator inet 192.168.102.2
set protocols evpn assisted-replication replicator vxlan-encapsulation-source-ip ingress-replication-ip
```

3. Configure the loopback interface that is used in the VRF routing instance.

```
set interfaces lo0 unit 1 family inet
```

4. Configure a VRF routing instance.

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 route-distinguisher 192.168.2.2:1
set routing-instances VRF-1 vrf-target target:100:1
```

5. Configure VLANs to the border leaf.

```
set vlans BD-1 vlan-id 1
set vlans BD-1 vxlan vni 100001
set vlans BD-2 vlan-id 2
set vlans BD-2 vxlan vni 100002
set vlans BD-3 vlan-id 3
set vlans BD-3 vxlan vni 100003
set vlans BD-4 vlan-id 4
set vlans BD-4 vxlan vni 100004
```

6. Configure the EVPN protocol with VXLAN encapsulation.

```
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn extended-vni-list all
```

7. Configure the switch options, and specify that the loopback interface is the VTEP source interface.

```
set switch-options vtep-source-interface lo0.0
set switch-options vrf-target target:10458:0
set switch-options vrf-target auto
```

Configuring the Border Leaf

This section describes how to set up multicast routing on the border leafs.

> **NOTE:** We do not configure AR on the border leafs. In this network design, the two border leafs share a multihomed ESI, and one of the border leaf devices supports AR but the other does not. In this situation, we do not recommend configuring AR on the border leaf that supports this feature. However, if your network includes two border leafs that share a multihomed ESI, and both border leaf devices support AR, we support the configuration of AR on both border leafs.

1. Configure VLANs

```
set vlans BD-1 vlan-id 1
set vlans BD-1 l3-interface irb.1
set vlans BD-1 vxlan vni 100001
set vlans BD-2 vlan-id 2
set vlans BD-2 l3-interface irb.2
set vlans BD-2 vxlan vni 100002
set vlans BD-3 vlan-id 3
set vlans BD-3 l3-interface irb.3
set vlans BD-3 vxlan vni 100003
set vlans BD-4 vlan-id 4
set vlans BD-4 l3-interface irb.4
set vlans BD-4 vxlan vni 100004
```

2. Configure the EVPN protocol with VXLAN encapsulation.

```
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn extended-vni-list all
```

3. Configure the switch options and specify that the loopback interface is the VTEP source interface.

```
set switch-options vtep-source-interface lo0.0
set switch-options vrf-target target:10458:0
set switch-options vrf-target auto
```

4. Configure the IRBs.

```
set interfaces irb unit 1 virtual-gateway-accept-data
set interfaces irb unit 1 family inet address 10.0.1.239/24 preferred
set interfaces irb unit 1 family inet address 10.0.1.239/24 virtual-gateway-address 10.0.1.254
set interfaces irb unit 1 family inet6 nd6-stale-time 1200
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:239/112 preferred
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:239/112 virtual-gateway-address
  2001:db8::10:0:1:254
set interfaces irb unit 1 family inet6 address fe80:10:0:1::239/64 virtual-gateway-address fe80:10:0:1::254
set interfaces irb unit 1 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 2 virtual-gateway-accept-data
set interfaces irb unit 2 family inet address 10.0.2.239/24 preferred
set interfaces irb unit 2 family inet address 10.0.2.239/24 virtual-gateway-address 10.0.2.254
set interfaces irb unit 2 family inet6 nd6-stale-time 1200
set interfaces irb unit 2 family inet6 address 2001:db8::10:0:2:239/112 preferred
set interfaces irb unit 2 family inet6 address 2001:db8::10:0:2:239/112 virtual-gateway-address
  2001:db8::10:0:2:254
set interfaces irb unit 2 family inet6 address fe80:10:0:2::239/64 virtual-gateway-address fe80:10:0:2::254
set interfaces irb unit 2 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 2 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 3 virtual-gateway-accept-data
set interfaces irb unit 3 family inet address 10.0.3.239/24 preferred
set interfaces irb unit 3 family inet address 10.0.3.239/24 virtual-gateway-address 10.0.3.254
set interfaces irb unit 3 family inet6 nd6-stale-time 1200
set interfaces irb unit 3 family inet6 address 2001:db8::10:0:3:239/112 preferred
set interfaces irb unit 3 family inet6 address 2001:db8::10:0:3:239/112 virtual-gateway-address
  2001:db8::10:0:3:254
set interfaces irb unit 3 family inet6 address fe80:10:0:3::239/64 virtual-gateway-address fe80:10:0:3::254
set interfaces irb unit 3 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 3 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.0.4.239/24 preferred
set interfaces irb unit 4 family inet address 10.0.4.239/24 virtual-gateway-address 10.0.4.254
set interfaces irb unit 4 family inet6 nd6-stale-time 1200
set interfaces irb unit 4 family inet6 address 2001:db8::10:0:4:239/112 preferred
set interfaces irb unit 4 family inet6 address 2001:db8::10:0:4:239/112 virtual-gateway-address
  2001:db8::10:0:4:254
set interfaces irb unit 4 family inet6 address fe80:10:0:4::239/64 virtual-gateway-address fe80:10:0:4::254
set interfaces irb unit 4 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 4 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

5. Configure a VRF routing instance.

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 interface irb.1
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 route-distinguisher 192.186.0.1:1
set routing-instances VRF-1 vrf-target target:100:1
```

6. Configure PIM for multicast routing at the border leaf devices.

```
set routing-instances VRF-1 protocols pim rp local address 10.0.1.239
set routing-instances VRF-1 protocols pim interface irb.1 family inet
set routing-instances VRF-1 protocols pim interface irb.1 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.2 family inet
set routing-instances VRF-1 protocols pim interface irb.2 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.3 family inet
set routing-instances VRF-1 protocols pim interface irb.3 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.4 family inet
set routing-instances VRF-1 protocols pim interface irb.4 mode sparse-dense
```

Verifying Assisted Replication on the Server Leaf

The server leaf is in the role of AR leaf device. This means that it does not perform ingress replication. Instead, it forwards one copy of multicast traffic to the spine, which is configured as the AR replicator device.

1. Verify that the spines are in the role of Assisted Replicator and are receiving Type 3 routes. Address 192.168.102.1 is Spine 1, and address 192.168.102.2 is Spine 2.

   `user@server-leaf>` **show route table bgp.evpn.0 match-prefix 3:*100001*192.168.102.* extensive | match "3:192.168.2|ASSISTED-REPLICATION"**

```
3:192.168.2.1:10000::100001::192.168.102.1/248 IM (2 entries, 0 announced)
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.1
 Role AR-REPLICATOR
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.1
 Role AR-REPLICATOR
3:192.168.2.2:10000::100001::192.168.102.2/248 IM (2 entries, 0 announced)
```

```
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.2
Role AR-REPLICATOR
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.2
Role AR-REPLICATOR
```

user@server-leaf> **show route table bgp.evpn.0 match-prefix 3:*100001*192.168.102.1* extensive**

```
bgp.evpn.0: 156388 destinations, 299429 routes (156388 active, 0 holddown, 0
hidden)
3:192.168.2.1:10000::100001::192.168.102.1/248 IM (2 entries, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.2.1:10000
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION
192.168.102.1 Role AR-REPLICATOR
                Next hop type: Indirect, Next hop index: 0
                Address: 0x1a6b08f0
                Next-hop reference count: 4000
                Source: 192.168.2.1
                Protocol next hop: 192.168.102.1
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <Active Int Ext>
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 4:58:08    Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.2.1
                AS path: I
                Communities: target:32897:268535457 encapsulation:vxlan(0x8)
evpn-mcast-flags:0x4:extended-MH-AR
                Import Accepted
                Localpref: 100
                Router ID: 192.168.2.1
                Secondary Tables: default-switch.evpn.0
                Indirect next hops: 1
                        Protocol next hop: 192.168.102.1
                        Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
                                Next hop: 172.16.109.0 via ae3.0
                                Session Id: 0x0
                                192.168.102.1/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 1
```

```
                                    Nexthop: 172.16.109.0 via ae3.0
                                    Session Id: 0
         BGP    Preference: 170/-101
                Route Distinguisher: 192.168.2.1:10000
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION
192.168.102.1 Role AR-REPLICATOR
                Next hop type: Indirect, Next hop index: 0
                Address: 0x1a6b08f0
                Next-hop reference count: 4000
                Source: 192.168.2.2
                Protocol next hop: 192.168.102.1
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <NotBest Int Ex>
                Inactive reason: Not Best in its group - Cluster list length
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 5:14:41    Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.2.2
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.2.1
                Communities: target:32897:268535457 encapsulation:vxlan(0x8)
evpn-mcast-flags:0x4:extended-MH-AR
                Import Accepted
                Localpref: 100
                Router ID: 192.168.2.2
                Secondary Tables: default-switch.evpn.0
                Indirect next hops: 1
                        Protocol next hop: 192.168.102.1
                        Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
                                Next hop: 172.16.109.0 via ae3.0
                                Session Id: 0x0
                                192.168.102.1/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 1
                                        Nexthop: 172.16.109.0 via ae3.0
                                        Session Id: 0
```

2. Verify the spine that is set up as the AR device for the VLAN. 192.168.102.2 is the address of the AR device.

user@server-leaf> **show evpn multicast-snooping assisted-replication next-hops l2-domain-id 100001**

```
Instance: default-switch
AR Role: AR Leaf

  VN Identifier: 100001
    Load Balance Nexthop Index: 134135
      Load balance to:
      Nexthop Index    Interface        AR IP
      23119            vtep.32881       192.168.102.1
      21753      vtep.32770     192.168.102.2 (Designated Node)
```

Verifying Assisted Replication on the Spine

1. Verify that server leaf devices 1 through 4 are AR leaf devices. (The loopback addresses of server leaf devices 1 through 4 are 192.168.0.1, 192.168.0.2, 192.168.0.3, and 192.168.0.4, respectively.) The border leaf devices are not set up for assisted replication.

   user@spine> **show route table bgp.evpn.0 match-prefix 3:*100001*192.168.0.* extensive | match "3:192.168.0.|LEAF"| except "PMSI|Path"**

```
3:192.168.0.1:10000::100001::192.168.0.1/248 IM (1 entry, 1 announced)
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF
## Leaf 1
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF
3:192.168.0.2:10::100001::192.168.0.2/248 IM (1 entry, 1 announced)
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.2 AR-LEAF
## Leaf 2
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.2 AR-LEAF
3:192.168.0.3:10000::100001::192.168.0.3/248 IM (1 entry, 1 announced)
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.3 AR-LEAF
## Leaf 3
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.3 AR-LEAF
3:192.168.0.4:10000::100001::192.168.0.4/248 IM (2 entries, 1 announced)
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.4 AR-LEAF
## Leaf 4
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.4 AR-LEAF
3:192.168.0.10:10000::100001::192.168.0.10/248 IM (1 entry, 1 announced)  ## Border
 Leaf 1
3:192.168.0.11:10000::100001::192.168.0.11/248 IM (1 entry, 1 announced)  ## Border
 Leaf 2
```

   user@spine> **show route table bgp.evpn.0 match-prefix 3:*100001*192.168.0.1 extensive**

```
bgp.evpn.0: 362179 destinations, 504791 routes (347873 active, 14306 holddown,
 0 hidden)
3:192.168.0.1:10000::100001::192.168.0.1/248 IM (1 entry, 1 announced)
TSI:
Page 0 idx 0, (group overlay-bgp-rr type Internal) Type 1 val 0x1af46804
(adv_entry)
   Advertised metrics:
     Nexthop: 192.168.0.1
     Localpref: 100
     AS path: [4210000001] I
     Communities: target:32897:268535457 encapsulation:vxlan(0x8)
evpn-mcast-flags:0x1:snooping-enabled
```

```
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF

     Cluster ID: 192.168.2.10
     Originator ID: 192.168.0.1
Page 0 idx 1, (group overlay-bgp type Internal) Type 1 val 0x1af46510 (adv_entry)

   Advertised metrics:
     Nexthop: 192.168.0.1
     Localpref: 100
     AS path: [4210000001] I
     Communities: target:32897:268535457 encapsulation:vxlan(0x8)
evpn-mcast-flags:0x1:snooping-enabled
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF

     Cluster ID: 192.168.2.10
     Originator ID: 192.168.0.1
    Advertise: 0000001e
Path 3:192.168.0.1:10000::100001::192.168.0.1
from 192.168.0.1
Vector len 4.  Val: 0 1
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.0.1:10000
              PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1
 AR-LEAF
                Next hop type: Indirect, Next hop index: 0
                Address: 0x11bd0d90
                Next-hop reference count: 35023
                Source: 192.168.0.1
                Protocol next hop: 192.168.0.1
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <Active Int Ext>
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 18:34:04   Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.0.1
                Announcement bits (1): 1-BGP_RT_Background
                AS path: I
                Communities: target:32897:268535457 encapsulation:vxlan(0x8)
evpn-mcast-flags:0x1:snooping-enabled
                Import Accepted
                Localpref: 100
                Router ID: 192.168.0.1
                Secondary Tables: default-switch.evpn.0
                Indirect next hops: 1
```

```
                    Protocol next hop: 192.168.0.1
                    Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                    Indirect path forwarding next hops: 1
                            Next hop type: Router
                            Next hop: 172.16.101.1 via ae1.0
                            Session Id: 0x0
                            192.168.0.1/32 Originating RIB: inet.0
                              Node path count: 1
                              Forwarding nexthops: 1
                                    Nexthop: 172.16.101.1 via ae1.0
                                    Session Id: 0
```

Multicast Optimization— Feature Summary

Table 13 on page 262 provides a history of the features described in this section and their support within this reference design.

Table 13: Multicast Optimization Feature Summary

Hardware	IGMPv2 Snooping	EVPN Type 6 SMET Routes	Inter-VNI Multicast with PIM Gateway	Assisted Replication	PIM to External Rendezvous Point (From Border)
QFX5100[1]	Not supported	Not supported	Not supported	Not supported	Not supported
QFX5110-32Q, QFX5110-48S	18.1R3-S3	18.4R2	Not supported	Not supported	Not supported
QFX5120-48Y	18.4R2	18.4R2	Not supported	Not supported	Not supported
QFX5120-32C	19.1R2	19.1R2	Not supported	Not supported	Not supported
QFX5200-32C[1], QFX5200-48Y[1]	Not supported	Not supported	Not supported	Not supported	Not supported
QFX10002-36Q/72Q, QFX10008, QFX10016	18.1R3-S3	18.4R2	18.1R3-S3	18.4R2	17.3R3-S1
QFX10002-60C	20.2R1	20.2R1	20.2R1	20.2R1	20.2R1

Table 13: Multicast Optimization Feature Summary *(continued)*

Hardware	IGMPv2 Snooping	EVPN Type 6 SMET Routes	Inter-VNI Multicast with PIM Gateway	Assisted Replication	PIM to External Rendezvous Point (From Border)
MX204; MX240, MX480, MX960 with MPC7E; MX10003;	Not supported	Not supported	Not supported	Not supported	Not supported

[1]Make sure that IGMP snooping is not enabled on these QFX switches. If IGMP snooping is inadvertently enabled, these switches might process EVPN Type 6 routes that are reflected to them.

RELATED DOCUMENTATION

Multicast Support in EVPN-VXLAN Overlay Networks

Configuring VMTO

This section describes how to configure virtual machine traffic optimization (VMTO). For overview information about VMTO, see Ingress Virtual Machine Traffic Optimization for EVPN section in "Data Center Fabric Blueprint Architecture Components" on page 15.

To enable VMTO for EVPN:

- Enable VMTO on the spine:

```
set routing-instances VRF-1 protocols evpn remote-ip-host-routes
```

To verify that VMTO is working:

- Display the routing table.

```
host@spine> show route table VRF-1.inet.0 protocol evpn

VRF-1.inet.0: 45 destinations, 45 routes (45 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
10.0.37.1/32       *[EVPN/7] 1d 21:44:43
                    >  via irb.37
10.0.37.2/32       *[EVPN/7] 1d 15:26:27
```

VMTO — Release History

Table 14 on page 264 provides a history of all of the features in this section and their support within this reference design.

Table 14: VMTO Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train also support all features documented in this section.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section.
18.1R3-S3	All devices in the reference design that support Junos OS Release 18.1R3-S3 and later releases in the same release train also support all features documented in this section.

RELATED DOCUMENTATION

Ingress Virtual Machine Traffic Optimization

DHCP Relay Design and Implementation

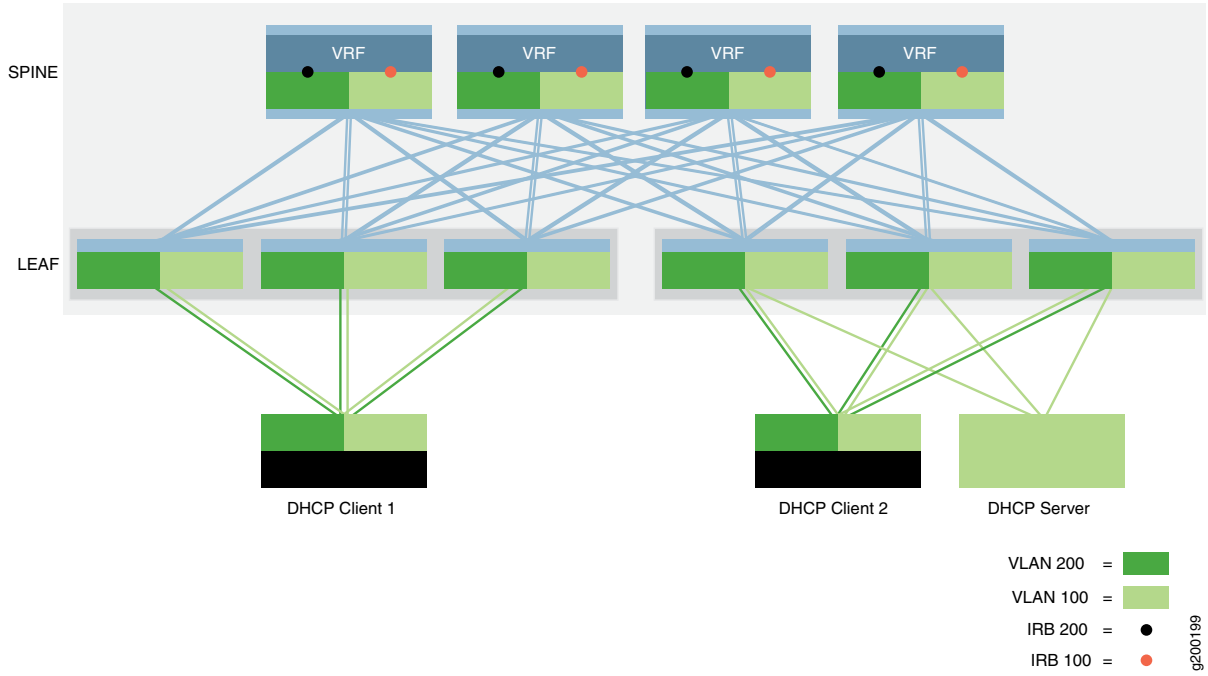
IN THIS SECTION

For an overview of the DHCP Relay implementation in this design, see the DHCP Relay section in “Data Center Fabric Blueprint Architecture Components” on page 15.

DHCP Relay was validated on centrally-routed bridging overlays only. See “Centrally-Routed Bridging Overlay Design and Implementation” on page 95.

Figure 61 on page 265 provides a sample DHCP server and client placement in this design.

Figure 61: DHCP Server and Client Setup

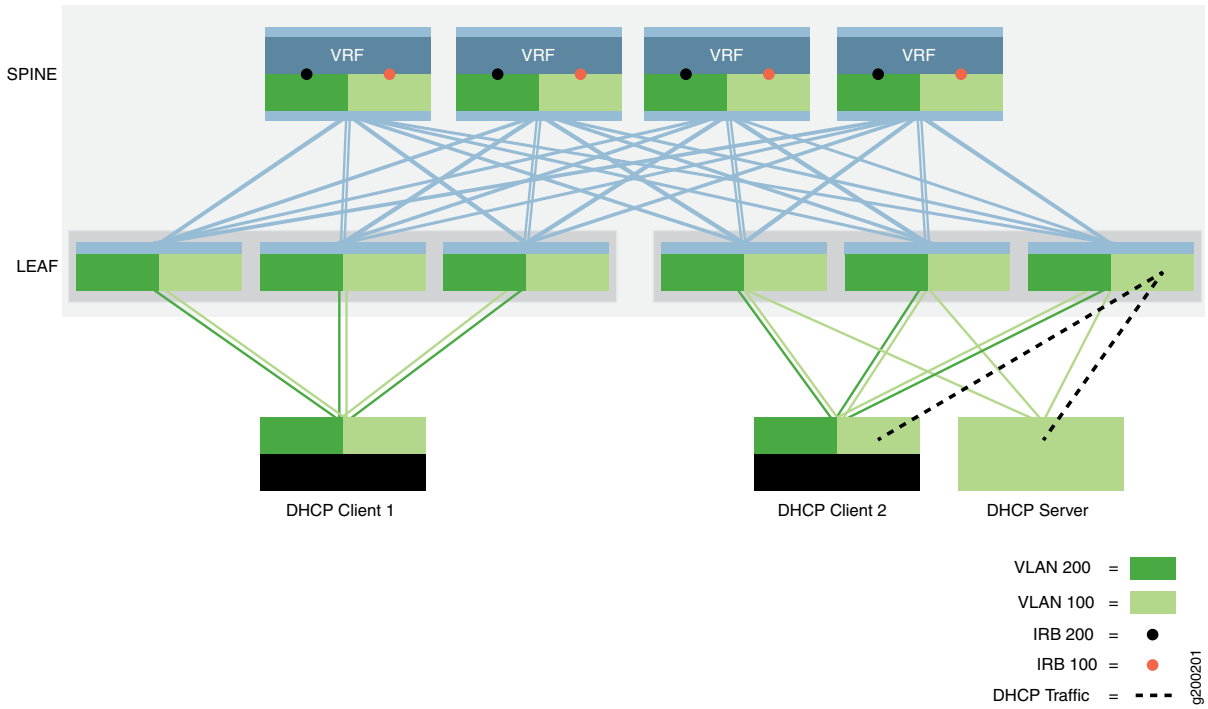


This section covers the following DHCP Relay scenarios:

Enabling DHCP Relay: DHCP Client and Server in Same VLAN and Connected to Same Leaf Device

Figure 62 on page 266 shows how DHCP traffic might traverse the overlay network when the DHCP client and server connect to access interfaces on the same leaf device and are in the same VLAN.

Figure 62: DHCP Relay—Same Leaf Device and Same VLAN

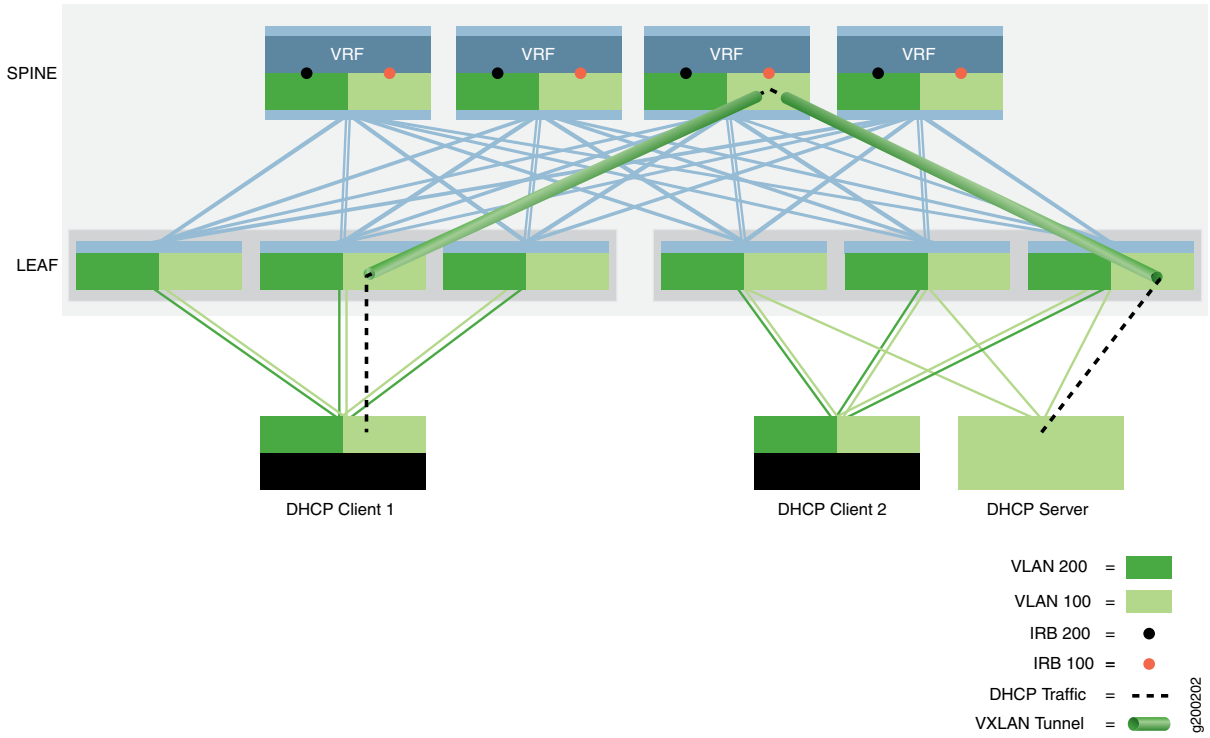


DHCP traffic is handled like any other intra-VLAN traffic in this scenario. It could also have been passed in **VNI_10000** on leaf device 10 or 11. No additional configuration of the leaf or spine devices is required to support this traffic exchange.

Enabling DHCP Relay: DHCP Client and Server in Same VLAN and Connected to Different Leaf Devices

Figure 63 on page 267 shows how DHCP traffic might be forwarded when the DHCP client and server connect to access interfaces on non-connected leaf devices that have interfaces in the same VLAN.

Figure 63: DHCP Relay—Different Leaf Device and Same VLAN

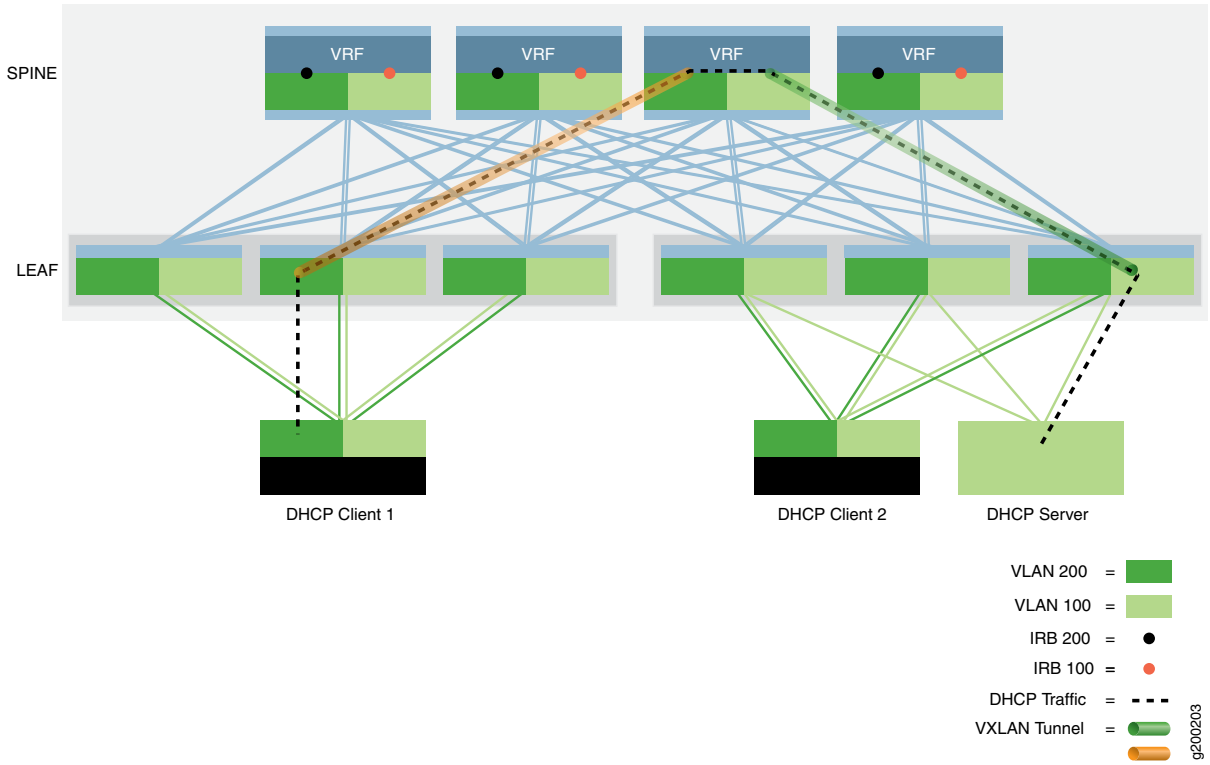


In this scenario, the DHCP traffic remains in **VNI_10000** and is passed over the topology using a VXLAN tunnel. No additional configuration of the leaf or spine devices is required to support this traffic exchange.

Enabling DHCP Relay: DHCP Client and Server in Different VLANs

Figure 64 on page 268 shows how DHCP traffic is forwarded when the DHCP client and server connect to access interfaces in different VLANs.

Figure 64: DHCP Relay—Different VLANs



When the DHCP client and server are in different VNI, DHCP traffic is forwarded to a spine device. The DHCP traffic is forwarded between VNIs via the IRB interfaces on the spine device and then passed to the destination DHCP client or server.

The IRB interfaces on the spine devices must be configured to support DHCP Relay.

To prepare the network to support DHCP relay when the DHCP client and server are in different VLANs:

1. Ensure the centrally-routed bridging overlay is configured. See "Centrally-Routed Bridging Overlay Design and Implementation" on page 95.

2. Enable DHCP relay in a routing instance with the forward only option. The forward only option ensures that DHCP packets are forwarded on the switch but that no DHCP server client bindings are created.

   Spine Device:

   ```
   set routing-instances VRF_1 forwarding-options dhcp-relay forward-only
   ```

3. Create and activate the DHCP relay server group. The DHCP relay server group include one or more DHCP servers—individually identified by IP address—and a user-defined name for the servers.

   In this reference design, one DHCP server—10.1.0.211—is assigned into a DHCP server group named *DHCP_SERVER_GROUP_1*.

   Spine Device:

   ```
   set routing-instances VRF_1 forwarding-options dhcp-relay server-group DHCP_SERVER_GROUP_1
   10.1.0.211
   set routing-instances VRF_1 forwarding-options dhcp-relay active-server-group DHCP_SERVER_GROUP_1
   ```

4. Associate the server group with the IRB interfaces on the spine devices.

   In this topology, irb.100 and irb.200 relay DHCP traffic between VNI_10000 and VNI_20000.

   Spine Devices:

   ```
   set routing-instances VRF_1 forwarding-options dhcp-relay group RELAY_DHCP_SERVER_GROUP_1
   interface irb.100
   set routing-instances VRF_1 forwarding-options dhcp-relay group RELAY_DHCP_SERVER_GROUP_1
   interface irb.200
   ```

5. Confirm that the DHCP hosts received an IP address by reviewing the ARP table on the spine device.

   ```
   user@spine-1> show arp no-resolve
   MAC Address       Address         Interface        Flags
   0e:ad:11:00:00:0d 10.1.0.214 irb.100 [.local..260] none
   0e:ad:11:00:00:0e 10.1.0.215 irb.100 [.local..260] none
   0e:ad:11:00:00:10 10.1.1.219 irb.200 [.local..260] none
   0e:ad:11:00:00:0f 10.1.1.220 irb.200 [.local..260] none
   ```

DHCP Relay — Release History

Table 2 on page 66 provides a history of all of the features in this section and their support within this reference design.

Table 15: DHCP Relay Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train and serve as spine devices also support all features documented in this section. We have not validated DHCP Relay between VLANs in edge-routed bridging overlays for this reference design.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section.
18.1R3-S3	QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section.
17.3R3-S1	All spine devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section. We have not validated DHCP Relay between VLANs in edge-routed bridging overlays for this reference design.

RELATED DOCUMENTATION

Configuring DHCP and BOOTP Relay

Verifying and Disabling Proxy ARP and ARP Suppression for the Edge-Routed Bridging Overlay

IN THIS SECTION

Verifying Proxy ARP and ARP Suppression for the Edge-Routed Bridging Overlay

Proxy ARP and ARP suppression are enabled by default on all QFX Series switches that can act as leaf devices in an edge-routed bridging overlay. (For a list of these switches, see Table 1 on page 46.) As a result, there is no configuration needed to enable these features. To verify proxy ARP is working on supported devices and ARP suppression prevents other leaf devices from seeing the ARP requests, perform the following:

1. Select a remote end system entry to verify that proxy ARP is enabled on a supported leaf device. In this example, Leaf 10 is a QFX10002 switch.

   `user@leaf-10>` **show arp no-resolve expiration-time | match 10.1.4.201**

   ```
   02:0c:10:04:02:01 10.1.4.201      irb.500 [.local..9]      none 1157
   ```

2. Verify that the entry was learned from a remote leaf device. In this case, the ARP entry was learned from Leaf 4 and 5.

   `user@leaf-10>` **show evpn database mac-address 02:0c:10:04:02:01 extensive**

   ```
   Instance: default-switch

   VN Identifier: 50000, MAC address:: 02:0c:10:04:02:01
     Source: 00:00:00:00:00:00:51:10:00:01, Rank: 1, Status: Active
       Remote origin: 192.168.1.4  # Leaf 4
       Remote origin: 192.168.1.5  # Leaf 5
       Timestamp: Sep 11 00:37:32 (0x59b63d3c)
       State: <Remote-To-Local-Adv-Done>
       IP address: 10.1.4.201
         Remote origin: 192.168.1.4
         Remote origin: 192.168.1.5
       IP address: 2001:db8::10:1:4:201
         Flags: <Proxy>
         Remote origin: 192.168.1.4
         Remote origin: 192.168.1.5    History db:
         Time                 Event
        Sep 11 00:37:33 2017  Active ESI unchanged (00:00:00:00:00:00:51:10:00:01)

         Sep 11 00:37:33 2017  Updating output state (change flags 0x0)
         Sep 11 00:37:33 2017  Advertisement route cannot be created (no local
   state present)
         Sep 11 00:37:33 2017  Updating output state (change flags 0x0)
         Sep 11 00:37:33 2017  Advertisement route cannot be created (no local
   ```

```
source present)
      Sep 11 00:37:33 2017  IP host route cannot be created (No remote host
route for non-MPLS instance)
      Sep 11 00:37:33 2017  Updating output state (change flags 0x4000
<IP-Peer-Added>)
      Sep 11 00:37:33 2017  Creating MAC+IP advertisement route for proxy
      Sep 11 00:37:33 2017  IP host route cannot be created (No remote host
route for non-MPLS instance)
      Sep 11 00:37:33 2017  Clearing change flags <IP-Peer-Added>
```

3. Send an ARP Request and verify that the ARP reply is generated.

   `user@leaf-10>` **monitor traffic interface irb no-resolve**

```
verbose output suppressed, use <detail> or <extensive> for full protocol decode
Address resolution is OFF.
Listening on irb, capture size 96 bytes


00:43:01.881508  In arp who-has 10.1.4.201 tell 10.1.4.202
00:43:01.881569 Out arp reply 10.1.4.201 is-at 02:0c:10:04:02:01
00:43:02.081404  In arp who-has 10.1.4.201 tell 10.1.4.202

## The next entry is the MAC address from the operational
mode command issued in Step 2.
00:43:02.081466 Out arp reply 10.1.4.201 is-at 02:0c:10:04:02:01
```

4. Verify that ARP is suppressed in the remote leaf device. Note: There is no ARP request connecting Leaf 4 to any other leaf in the segment.

   `user@leaf-4>` **monitor traffic interface irb no-resolve**

```
verbose output suppressed, use <detail> or <extensive> for full protocol decode
Address resolution is OFF.
Listening on irb, capture size 96 bytes

^C
0 packets received by filter
0 packets dropped by kernel
```

Disabling Proxy ARP and ARP Suppression for the Edge-Routed Bridging Overlay

To override the default settings and disable proxy ARP and ARP suppression, you must configure the leaf device.

> **NOTE:** When disabling ARP suppression, be aware of the following implications
>
> - Even though it finds the MAC-IP address binding in its database, the device floods an ARP request through the Ethernet bridging domain. Therefore, we recommend that ARP suppression remains enabled.
> - Disabling the suppression of ARP packets on a PE device essentially disables the proxy ARP functionality. Therefore, we recommend that ARP suppression remains enabled.

To disable proxy ARP and ARP suppression, and verify this change, perform the following:

1. Configure the local leaf device (Leaf 10 in this example) to disable proxy ARP and ARP suppression.

   Leaf 10:

   ```
   set vlans VNI_50000 no-arp-suppression
   commit
   ```

2. Send an ARP Request and verify that ARP is not being suppressed.

   ```
   user@leaf-10> show arp no-resolve expiration-time | match 10.1.4.201
   ## Leaf 10 shows there is an ARP request for 10.1.4.201,
   which is connected to Leaf 4.
   02:0c:10:04:02:01 10.1.4.201      irb.500 [.local..9]      none 1066
   ```

   ```
   user@leaf-10> monitor traffic interface irb no-resolve
   verbose output suppressed, use <detail> or <extensive> for full protocol decode
   Address resolution is OFF.
   Listening on irb, capture size 96 bytes

   ## The ARP request from 10.1.4.202 intended for 10.1.4.201
   is ingress replicated and sent to all leaf devices.
   00:51:42.455797  In arp who-has 10.1.4.201 tell 10.1.4.202
   00:51:42.697881  In arp who-has 10.1.4.201 tell 10.1.4.202
   ```

```
00:51:42.855720  In arp who-has 10.1.4.201 tell 10.1.4.202
00:51:43.055561  In arp who-has 10.1.4.201 tell 10.1.4.202
```

NOTE: Notice that no ARP replies are being received by Leaf 10.

3. Verify that the ARP Request is flooded to all leaf devices.

```
user@leaf-4> monitor traffic interface irb no-resolve
verbose output suppressed, use <detail> or <extensive> for full protocol decode
Address resolution is OFF.
Listening on irb, capture size 96 bytes

## This next request is coming from Leaf 10.
00:50:58.867745  In arp who-has 10.1.4.201 tell 10.1.4.202
00:50:59.067914  In arp who-has 10.1.4.201 tell 10.1.4.202
00:50:59.268015  In arp who-has 10.1.4.201 tell 10.1.4.202
00:50:59.467997  In arp who-has 10.1.4.201 tell 10.1.4.202
00:50:59.668098  In arp who-has 10.1.4.201 tell 10.1.4.202
00:50:59.868231  In arp who-has 10.1.4.201 tell 10.1.4.202
```

Proxy ARP and ARP Suppression for an Edge-Routed Bridging Overlay — Release History

Table 16 on page 274 provides a history of all of the features in this section and their support within this reference design.

Table 16: Proxy ARP and ARP Suppression in an Edge-Routed Bridging Overlay Release History

Release	Description
19.1R2	QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train also support all features documented in this section.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section.

Table 16: Proxy ARP and ARP Suppression in an Edge-Routed Bridging Overlay Release History *(continued)*

Release	Description
18.1R3-S3	All devices in the reference design that support Junos OS Release 18.1R3-S3 and later releases in the same release train also support all features documented in this section.

RELATED DOCUMENTATION

EVPN Proxy ARP and ARP Suppression, and Proxy NDP and NDP Suppression

Configuring Layer 2 Port Security Features on Ethernet-Connected End Systems

IN THIS SECTION

This section shows how to configure the following Layer 2 port security features. For overview information about these features, see Layer 2 Port Security Features on Ethernet-Connected End Systems in “Data Center Fabric Blueprint Architecture Components” on page 15

Configuring Storm Control

In this sample configuration, storm control rate limits BUM traffic on server-facing aggregated Ethernet interfaces. If the amount of BUM traffic exceeds 6% of the available bandwidth on the interface, storm control drops it to prevent broadcast storms.

To enable storm control:

1. Create a storm control profile and specify the percentage of bandwidth available to BUM traffic.

   ```
   set forwarding-options storm-control-profiles STORM-CONTROL all bandwidth-percentage 6
   ```

2. Apply the storm control profile to an ingress Layer 2 interface. After you apply the profile to an interface, the interface resides in the default switch interface.

   ```
   set interfaces ae11 unit 0 family ethernet-switching storm-control STORM-CONTROL
   ```

Verifying Storm Control

To verify storm control activity, filter system log messages related to storm control:

```
user@leaf10> show log messages | match storm
```

```
Sep 27 11:35:34 leaf1-qfx5100 l2ald[1923]: L2ALD_ST_CTL_IN_EFFECT: ae11.0: storm
control in effect on the port
```

Configuring Port Security Using MAC Filtering

To configure MAC filtering, you create firewall filters in which you specify one or more of the supported match conditions. See https://www.juniper.net/documentation/en_US/junos/topics/concept/evpn-vxlan-security-monitor.html for a list of match conditions supported on QFX5110 switches and QFX10000 switches. You then apply the firewall filter to a Layer 2 interface.

To configure MAC filtering:

1. Create a firewall filter for an ingress interface.

```
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ from source-mac-address
    be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ from destination-mac-address
    ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from source-mac-address
    be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ from ether-type arp
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ from user-vlan-id 10
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ then accept
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ then count ARP-REQ-CNT
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from source-mac-address
    be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from destination-mac-address
    ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from ether-type ipv4
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from user-vlan-id 10
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST then accept
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST then count
    V4-BROADCAST-CNT-IN
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST from source-mac-address
    be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST from destination-mac-address
    ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST from ether-type ipv6
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST from user-vlan-id 10
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST then accept
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST then count
    V6-BROADCAST-CNT-IN
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from source-mac-address
    be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from destination-mac-address
    00:00:5e:00:00:04/48
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from source-port 1020
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from destination-port 1024
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from ip-source-address
    10.0.10.201/32
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from ip-destination-address
    10.0.12.201/32
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from ip-protocol tcp
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from user-vlan-id 10
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 then accept
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 then count
    V4-PKT-CNT-IN-TCP-FLAG-0x90
set firewall family ethernet-switching filter L2-INGRESS term DEF then accept
```

```
set firewall family ethernet-switching filter L2-INGRESS term DEF then count DEF_CNT_IN
```

2. Apply the firewall filter to the ingress of an access interface / Layer 2 interface.

```
set interfaces ae11 unit 0 family ethernet-switching filter input L2-INGRESS
```

3. Create a firewall filter for an egress interface.

```
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP from source-mac-address
   00:00:5e:00:00:04/48
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP from destination-mac-address
   be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP from ether-type arp
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP from user-vlan-id 10
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP then accept
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP then count ARP-REP-CNT
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST from source-mac-address
   be:ef:a4:03:00:0c/48
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST from destination-mac-address
   ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST from ether-type ipv4
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST from user-vlan-id 12
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST then accept
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST then count
   V4-BROADCAST-CNT-OUT
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST from source-mac-address
   be:ef:a4:03:00:0c/48
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST from destination-mac-address
   ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST from ether-type ipv6
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST from user-vlan-id 12
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST then accept
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST then count
   V6-BROADCAST-CNT-OUT
set firewall family ethernet-switching filter L2-EGRESS term DEF then accept
set firewall family ethernet-switching filter L2-EGRESS term DEF then count DEF_CNT_OUT
```

4. Apply the firewall filter to the egress interface.

```
set interfaces ae11 unit 0 family ethernet-switching filter output L2-EGRESS
```

Verifying MAC Filtering

1. Verify MAC filtering on the ingress interface.

   user@leaf10> **show firewall filter L2-INGRESS**

```
Filter: L2-INGRESS
Counters:
Name                                                Bytes              Packets
ARP-REQ-CNT                                           640                   10
DEF_CNT_IN                                       44038137                79032
V4-BROADCAST-CNT-IN                                     0                    0
V4-PKT-CNT-IN-TCP                                 7418880                57960
V6-BROADCAST-CNT-IN                               5370880                41960
```

2. Verify MAC filtering on the egress interface.

   user@leaf10> **show firewall filter L2-EGRESS**

```
Filter: L2-EGRESS
Counters:
Name                                                Bytes              Packets
ARP-REP-CNT                                            68                    1
DEF_CNT_OUT                                      17365964               146535
V4-BROADCAST-CNT-OUT                              4171264                32588
V6-BROADCAST-CNT-OUT                              3147264                24588
```

Configuring Analyzer-Based Port Mirroring

This section shows how to mirror ingress traffic on an underlay interface to another physical port.

The source and destination ports for mirrored traffic are on the same leaf or same spine.

1. Configure an analyzer to mirror ingress traffic on interface ae1.0.

```
set forwarding-options analyzer A1 input ingress interface ae1.0
```

2. Configure the destination interface for the mirrored packets.

```
set forwarding-options analyzer A1 output interface et-0/0/71.0
```

3. Configure the interface that connects to another switch (the uplink interface) to trunk mode and associate it with the appropriate VLAN.

```
set interfaces et-0/0/71 unit 0 family ethernet-switching interface-mode trunk
set interfaces et-0/0/71 unit 0 family ethernet-switching vlan members all
```

Verifying Port Mirroring

- To verify port mirroring:

```
host> show forwarding-options analyze
```

```
r
  Analyzer name                    : A1
  Mirror rate                      : 1
  Maximum packet length            : 0
  State                            : up
  ingress monitored interfaces     : ae1.0
  Output interface                 : et-0/0/71.0
```

Layer 2 Port Security Features — Release History

Table 17 on page 280 provides a history of all of the features in this section and their support within this reference design.

Table 17: Layer 2 Port Security Release History

Release	Description
19.1R2	• QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support MAC filtering, storm control, and port mirroring and analyzing. • QFX10002-60C switches running Junos OS Release 19.1R2 and later releases in the same release train support MAC filtering. These switches do not support storm control, and port mirroring and analyzing.
18.4R2	QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section.

Table 17: Layer 2 Port Security Release History *(continued)*

Release	Description
18.1R3-S3	All devices in the reference design that support Junos OS Release 18.1R3-S3 and later releases in the same release train also support all features documented in this section.

RELATED DOCUMENTATION

MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment

Remote Port Mirroring for EVPN-VXLAN Fabrics

3

CHAPTER

Testing and Scaling Summaries

Data Center Fabric Reference Design Scaling Summary for Junos OS Release 19.1R2

This section shows the scale numbers for QFX10002-60C and QFX5120-32C switches. We support these switches in the Data Center Fabric validated topology starting in Junos OS Release 19.1R2.

Scaling for Centrally-Routed Bridging Overlays

Table 18 on page 283 provides a summary of the scale numbers for QFX10002-60C switches acting as spine or border spine devices. Table 19 on page 284 outlines the scale numbers for QFX5120-32C switches acting as spine devices. We added these switches, which were running Junos OS Release 19.1R2, to the existing Data Center Fabric validated centrally-routed bridging overlay.

These numbers do not represent the maximum supported scale. They represent only the initial solution testing performed.

Table 18: Scaling with QFX10002-60C as Spine/Border Spine in Centrally-Routed Bridging Overlay

Attribute	Fabric	Per Spine/Border Spine (QFX10002-60C)	Per Leaf
Virtual switches	1	1	0
VLANS/bridge domains/VNIs	2,000	2,000	2,000
Ethernet segment identifiers (ESIs)	N/A	N/A	48[1]
IRB interfaces	2,000	2,000	N/A
OSPFv2 sessions on IRB interfaces	810	810	N/A
IS-IS sessions on IRB interfaces	204	204	N/A
BGP sessions on IRB interfaces	500	500	N/A

Table 18: Scaling with QFX10002-60C as Spine/Border Spine in Centrally-Routed Bridging Overlay *(continued)*

Attribute	Fabric	Per Spine/Border Spine (QFX10002-60C)	Per Leaf
ARP entries	15,000	15,000	N/A
ND scale (includes link local)	25,000	25,000	N/A
Number of VTEPs (leaf tunnels)	210	210	210
VRF instances	500	500	N/A
MAC addresses	25,000	25,000	24,000
Externally learned EVPN Type 5/IPVPN routes	76,000	76,000	N/A
Multicast *,G	N/A	N/A	N/A
Multicast S,G	N/A	N/A	N/A
IGMPv2 snooping	N/A	N/A	10,200

[1]Only QFX5110 switches were tested with 48 ESIs.

Table 19: Scaling with QFX5120-32C as Spine in Centrally-Routed Bridging Overlay

Attribute	Fabric	Per Spine (QFX5120-32C)	Per Leaf
Virtual switches	0	0	0
VLANS/bridge domains/VNIs	1,000	1,000	1,000
ESIs	N/A	N/A	48[1]
IRB interfaces	1,000	1,000	N/A

Table 19: Scaling with QFX5120-32C as Spine in Centrally-Routed Bridging Overlay *(continued)*

Attribute	Fabric	Per Spine (QFX5120-32C)	Per Leaf
OSPFv2 sessions on IRB interfaces	0	0	N/A
IS-IS sessions on IRB interfaces	0	0	N/A
BGP sessions on IRB interfaces	0	0	N/A
ARP entries	12,000	12,000	N/A
ND scale (includes link local)	16,000	16,000	N/A
Number of VTEPs (leaf tunnels)	210	210	210
VRF instances	250	250	N/A
MAC addresses	16,000	16,000	16,000
Externally learned EVPN Type 5/IPVPN routes	5,000	5,000	N/A
Multicast *,G	N/A	N/A	N/A
Multicast S,G	N/A	N/A	N/A
IGMPv2 snooping	N/A	N/A	10,200

[1]Only QFX5110 switches were tested with 48 ESIs.

Scaling for Edge-Routed Bridging Overlays

Table 20 on page 286 provides the scale numbers for QFX10002-60C switches acting as border leafs and QFX5120-32C switches acting as server leafs. We added these switches running Junos OS Release 19.1R2 to the Data Center Fabric validated edge-routed bridging overlay.

These numbers do not represent the maximum supported scale. They represent only the initial solution testing performed.

Table 20: Scaling with QFX10002-60C as Border Leaf and QFX5120-32C as Server Leaf in Edge-Routed Bridging Overlay

Attribute	Fabric	Per Border Leaf (QFX10002-60C)	Per Server Leaf (QFX5120-32C)
Virtual switches	1	1	0
VLANS/bridge domains/VNIs	2,000	2,000	1,000
ESIs	N/A	N/A	48[1]
IRB interfaces	2,000	2,000	1,000
OSPFv2 sessions on IRB interfaces	810	810	0
IS-IS sessions on IRB interfaces	204	204	0
BGP sessions on IRB interfaces	500	500	0
ARP entries	15,000	15,000	12,000
ND scale (incudes link local)	25,000	25,000	12,000
Number of VTEPs (leaf tunnels)	210	210	210
VRF instances	500	500	250
MAC addresses	25,000	25,000	16,000
Externally learned EVPN Type 5/IPVPN routes	76,000	76,000	5,000
Multicast *,G	N/A	N/A	N/A

Table 20: Scaling with QFX10002-60C as Border Leaf and QFX5120-32C as Server Leaf in Edge-Routed Bridging Overlay *(continued)*

Attribute	Fabric	Per Border Leaf (QFX10002-60C)	Per Server Leaf (QFX5120-32C)
Multicast S,G	N/A	N/A	N/A
IGMPv2 snooping	10,200	N/A	10,200

[1]Only QFX5110 switches were tested with 48 ESIs.

RELATED DOCUMENTATION

Data Center Fabric Reference Design Overview and Validated Topology | 44

Data Center Fabric Reference Design Scaling Summary for Junos OS Release 18.4R2-S2

This section shows the scale numbers for MX routers that we support in the Data Center Fabric validated topology starting in Junos OS Release 18.4R2-S2.

Scaling for Data Center Interconnect Using EVPN Type 5

Table 21 on page 287 provides scale numbers for an MX router that acts as a border leaf in a data center interconnect (DCI) environment that uses EVPN Type 5 routes.

These numbers do not represent the maximum supported scale. They represent only the initial solution testing performed for this guide.

Table 21: Data Center Fabric Reference Design Scaling Summary -DCI Using EVPN Type 5 in Junos OS Release 18.4R2-S2

Attribute	Single Border Leaf Device Maximum (MX480 with MPC7E)
VRF instances	2,000

Table 21: Data Center Fabric Reference Design Scaling Summary -DCI Using EVPN Type 5 in Junos OS Release 18.4R2-S2 *(continued)*

Attribute	Single Border Leaf Device Maximum (MX480 with MPC7E)
External routes/route prefixes	150,000
Number of VTEPs (leaf tunnels)	400

RELATED DOCUMENTATION

Data Center Fabric Reference Design Overview and Validated Topology | 44

Data Center Fabric Reference Design Scaling Summary for Junos OS Release 18.4R2

This section shows the scale tested in the Data Center Fabric validated topology for 18.4R2. These numbers do not represent the maximum supported scale. They represent only the initial solution testing performed for this guide.

Scaling for Centrally-Routed Bridging

Table 22 on page 288 summarizes the scale tested in the Data Center Fabric validated topology for centrally-routed bridging (CRB) in Junos 18.4R2. These numbers do not represent the maximum supported scale. They represent only the initial solution testing performed for this guide.

Table 22: Data Center Fabric Reference Design Scaling Summary -Centrally-Routed Bridging in Junos 18.4R2

Attribute	Total Fabric Maximum	Single Spine QFX10000 Device Maximum	Single Leaf Device Maximum
VLANS/Bridge Domains/VNIs	4,000	4,000	2,000
IRB interfaces	4,000	4,000	N/A

Table 22: Data Center Fabric Reference Design Scaling Summary -Centrally-Routed Bridging in Junos 18.4R2 *(continued)*

Attribute	Total Fabric Maximum	Single Spine QFX10000 Device Maximum	Single Leaf Device Maximum
BGP sessions on IRBs	500	500	N/A
ISIS Sessions on IRB	204	204	N/A
OSPFv2 sessions on IRBs	810	810	N/A
IGMPv2	10,200	10,200	10,200
ARP entries	40,000	40,000	N/A
ND scale	20,000	20,000	N/A
Number of VTEPs (leaf tunnels)[1]	210	210	210
VRF instances	1,000	1,000	N/A
MAC addresses	40,000	40,000	20,000
Multicast *,G	10,200	10,200	N/A
Multicast S,G	10,200	10,200	N/A
Externally learned EVPN/Type 5/IPVPN routes	100,000	100,000	N/A

[1]The number of VTEPs (leaf tunnels) shown above was the result of multidimensional testing. A higher scale of 500 was achieved in unidimensional testing.

Scaling for Edge-Routed Bridging

Table 23 on page 290 summarizes the scale tested in the Data Center Fabric validated topology for edge-routed bridging (ERB) in Junos 18.4R2. These numbers do not represent the maximum supported scale. They represent only the initial solution testing performed for this guide.

Table 23: Data Center Fabric Reference Design Scaling Summary - Edge-Routed Bridging in Junos 18.4R2

Attribute	Total Fabric Maximum	Single Border Leaf Device Maximum (QFX10K)	Single Leaf Device Maximum	
			QFX10K (Max)	QFX5K (Max)
VLANS/Bridge Domains/VNIs	4,000	4,000	4,000	1,000
IRB interfaces	4,000	4,000	4,000	1,000
OSPFv2 sessions on IRB	810	810	810 (on border leaf)	N/A
ISIS sessions on IRB	204	204	204 (on border leaf)	N/A
BGP sessions on IRB	500	500	500 (on border leaf)	N/A
ARP entries	40,000	40,000	40,000	12,000
ND Scale	20,000	20,000	20,000	12,000
Number of VTEPs (leaf tunnels)	210	210	210	210
VRF instances	1,000	1,000	1,000	250
MAC addresses	40,000	100,000	100,000	16,000
Externally learned EVPN/Type 5/IPVPN routes	100,000	100,000	100,000	250
Multicast - *,G	10,200	10,200	10,200	N/A
Multicast - S,G	10,200	10,200	10,200	N/A
IGMPv2	10,200	10,200	10,200	10,200

RELATED DOCUMENTATION

Data Center Fabric Reference Design Overview and Validated Topology | 44

Data Center Fabric Reference Design Scaling Summary for Junos OS Release 17.3R3-S3

Table 22 on page 288 summarizes the scale tested in the Data Center Fabric validated topology for centrally-routed bridging (CRB) in Junos 17.3R3-S3. These numbers do not represent the maximum supported scale. They represent only the solution testing performed for this guide. The results of the multidimensional testing are provided for reference purposes only.

Table 24: Data Center Fabric Reference Design Scaling Summary - Centrally-Routed Bridging in Junos 17.3R3-S3

Attribute	Total Fabric Maximum	Single Spine Device Maximum	Single Leaf Device Maximum
VLANS/Bridge Domains/VNIs	4,000	4,000	2,000
IRB interfaces	4,000	4,000	N/A
ARP entries	40,000	40,000	N/A
ND Scale	20,000	20,000	N/A
Number of VTEPs (leaf tunnels)[1]	110	110	110
VRF instances	1,000	1,000	N/A
MAC addresses	40,000	40,000	20,000
Externally learned EVPN/Type 5/IPVPN routes	16,000	16,000	N/A

[1]The number of VTEPs (leaf tunnels) shown above was the result of multidimensional testing. A higher scale of 500 was achieved in unidimensional testing.

RELATED DOCUMENTATION

Data Center Fabric Reference Design Overview and Validated Topology | 44