

Extreme Network OS IP Fabrics Configuration Guide, 7.3.0

Supporting Network OS 7.3.0

© 2018, Extreme Networks, Inc. All Rights Reserved.

Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries. All other names are the property of their respective owners. For additional information on Extreme Networks Trademarks please see www.extremenetworks.com/company/legal/trademarks. Specifications and product availability are subject to change without notice.

© 2017, Brocade Communications Systems, Inc. All Rights Reserved.

Brocade, the B-wing symbol, and MyBrocade are registered trademarks of Brocade Communications Systems, Inc., in the United States and in other countries. Other brands, product names, or service names mentioned of Brocade Communications Systems, Inc. are listed at www.brocade.com/en/legal/brocade-Legal-intellectual-property/brocade-legal-trademarks.html. Other marks may belong to third parties.

Notice: This document is for informational purposes only and does not set forth any warranty, expressed or implied, concerning any equipment, equipment feature, or service offered or to be offered by Brocade. Brocade reserves the right to make changes to this document at any time, without notice, and assumes no responsibility for its use. This informational document describes features that may not be currently available. Contact a Brocade sales office for information on feature and product availability. Export of technical data contained in this document may require an export license from the United States government.

The authors and Brocade Communications Systems, Inc. assume no liability or responsibility to any person or entity with respect to the accuracy of this document or any loss, cost, liability, or damages arising from the information contained herein or the computer programs that accompany it.

The product described by this document may contain open source software covered by the GNU General Public License or other open source license agreements. To find out which open source software is included in Brocade products, view the licensing terms applicable to the open source software, and obtain a copy of the programming source code, please visit <http://www.brocade.com/support/oscd>.

Contents

| | |
|---|-----------|
| Preface..... | 7 |
| Document conventions..... | 7 |
| Notes, cautions, and warnings..... | 7 |
| Text formatting conventions..... | 7 |
| Command syntax conventions..... | 8 |
| Extreme resources..... | 8 |
| Document feedback..... | 8 |
| Contacting Extreme Technical Support..... | 9 |
| About This Document..... | 11 |
| Supported hardware and software..... | 11 |
| What's new in this document..... | 11 |
| IP Fabrics..... | 13 |
| Overview of IP Fabrics..... | 13 |
| Physical topologies and scale..... | 14 |
| IP unnumbered interfaces..... | 16 |
| Multihoming..... | 17 |
| Additional considerations and limitations for IP Fabrics..... | 18 |
| BFD for IP Unnumbered ECMP Interfaces..... | 19 |
| Overview of BFD for IP unnumbered ECMP interfaces..... | 19 |
| Classical BFD..... | 19 |
| BFD for IP unnumbered ECMP interfaces..... | 19 |
| Considerations and dependencies..... | 20 |
| Configuring BFD for IP unnumbered ECMP interfaces..... | 20 |
| Multicast in IP Fabrics..... | 23 |
| IGMP snooping in IP Fabrics..... | 23 |
| IGMP snooping source and querier connected to same leaf node..... | 24 |
| IGMP snooping source and querier not connected to same leaf node..... | 24 |
| PIM-SM in IP Fabrics..... | 24 |
| RP at the spine in an IP Fabric with symmetric/asymmetric routing in the default VRF..... | 24 |
| RP outside the IP Fabric with symmetric/asymmetric routing in the default VRF..... | 26 |
| Controllerless Network Virtualization with BGP EVPN..... | 27 |
| Overview of controllerless network virtualization with BGP EVPN..... | 27 |
| BGP-based underlay architecture supporting BGP EVPN..... | 30 |
| eBGP-based BGP EVPN..... | 30 |
| iBGP-based BGP EVPN..... | 32 |
| BGP add path..... | 33 |
| BGP EVPN NLRI..... | 33 |
| MAC address learning..... | 34 |
| EVPN instances..... | 34 |
| BGP L2VPN EVPN address family..... | 35 |
| Automatic VXLAN tunnel endpoint discovery..... | 36 |
| BGP next-hop-unchanged..... | 37 |
| BGP retain route target..... | 37 |
| RIBout AS path check..... | 37 |

| | |
|---|-----------|
| BGP legacy features supported for the L2VPN EVPN address family..... | 37 |
| Ethernet Segment Identifiers for BGP routing..... | 38 |
| BGP dynamic neighbors..... | 39 |
| BGP dynamic neighbors for an IP Fabric..... | 39 |
| Configuring BGP dynamic neighbors for an IP Fabric..... | 42 |
| Multi-VRF for BGP EVPN..... | 43 |
| Static anycast gateway..... | 44 |
| Overview of static anycast gateway | 44 |
| Considerations and limitations for static anycast gateway | 44 |
| Configuring MAC static anycast gateway addresses..... | 45 |
| Configuring IP static anycast gateway addresses | 46 |
| Show commands for static anycast gateway | 46 |
| ARP and ND scaling enhancements | 47 |
| ARP and Neighbor Discovery..... | 47 |
| ARP and ND suppression..... | 47 |
| Conversational ARP and ND..... | 49 |
| Standards conformance and RFC support for BGP EVPN..... | 51 |
| Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay..... | 53 |
| Configuration overview..... | 53 |
| Configuring the leaf switches..... | 55 |
| Creating the VLANs..... | 55 |
| Configuring ARP and ND suppression..... | 56 |
| Configuring static anycast gateway MAC addresses..... | 56 |
| Configuring the interfaces..... | 57 |
| Configuring the VRF instances..... | 58 |
| Configuring BGP routing..... | 59 |
| Configuring the overlay gateway..... | 63 |
| Configuring the spine switches | 64 |
| (Optional) Configuring a BGP EVPN instance | 68 |
| (Optional) Configuring support for dual-homed servers..... | 70 |
| Configuring a superspine switch..... | 71 |
| Configuring interoperability with other vendors..... | 74 |
| Verifying the configuration..... | 75 |
| Verifying configurations on a leaf..... | 75 |
| Verifying configurations on a spine..... | 77 |
| Using traceroute for overlay tunnels..... | 78 |
| Configuring additional features for IP Fabrics..... | 78 |
| Configuring MAC learning of Layer 2 extension site through BGP..... | 78 |
| Disabling automatic VXLAN tunnel endpoint discovery by BGP..... | 79 |
| Configuring BGP next hop unchanged..... | 79 |
| Configuring BGP retain route target..... | 80 |
| Applying a BGP extended community filter for the L2VPN EVPN address family..... | 81 |
| Configuring BGP graceful restart for the L2VPN EVPN address family..... | 82 |
| Configuring BGP peer groups for the L2VPN EVPN address family..... | 83 |
| Configuring a route reflector client for the L2VPN EVPN address family..... | 84 |
| Disabling client-to-client reflection for the L2VPN EVPN address family..... | 85 |
| Disabling the BGP AS_PATH check function for the L2VPN EVPN address family..... | 86 |
| Enabling the BGP AS_PATH check function for the sender BGP speaker..... | 87 |
| Using BGP VRF route filters..... | 88 |

| | |
|---|------------|
| Configuring an IP Fabric Automatically..... | 89 |
| Overview of automatic configuration..... | 89 |
| Underlay configuration examples..... | 89 |
| Overlay configuration examples..... | 90 |
| Configuration requirements and considerations..... | 91 |
| Loopback IP address..... | 91 |
| Role (leaf or spine)..... | 91 |
| Interface range..... | 92 |
| RBridge ID..... | 92 |
| Local AS..... | 92 |
| Password..... | 92 |
| Overlay flag..... | 93 |
| LVTEP IP address..... | 93 |
| Verbose..... | 93 |
| Configuration method and examples..... | 93 |
| Appendix A: Supported BGP EVPN Commands..... | 99 |
| Configuration commands supporting BGP EVPN..... | 99 |
| RBridge ID configuration mode..... | 99 |
| BGP configuration mode..... | 99 |
| BGP address-family L2VPN EVPN configuration mode..... | 99 |
| EVPN instance configuration mode..... | 99 |
| VNI configuration mode..... | 100 |
| VRF configuration mode..... | 100 |
| Port-channel configuration mode..... | 100 |
| VXLAN overlay-gateway site configuration mode..... | 100 |
| Show commands supporting BGP EVPN..... | 100 |
| Appendix B: Sample BGP EVPN configuration files..... | 103 |
| Sample BGP EVPN superspine configuration using eBGP..... | 103 |
| Sample BGP EVPN spine configuration using eBGP..... | 103 |
| Sample BGP EVPN leaf configuration using eBGP..... | 105 |
| Sample BGP EVPN superspine configuration using iBGP..... | 106 |
| Sample BGP EVPN spine configuration using iBGP..... | 106 |
| Sample BGP EVPN leaf configuration using iBGP..... | 107 |
| Appendix C: Sample Topology Configuration Files..... | 109 |
| Leaf..... | 109 |
| Spine..... | 114 |
| SuperSpine..... | 115 |

Preface

- Document conventions..... 7
- Extreme resources..... 8
- Document feedback..... 8
- Contacting Extreme Technical Support..... 9

Document conventions


The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in Extreme technical documentation.


Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

NOTE
A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION
An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

 **CAUTION**
A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

 **DANGER**
A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used to highlight specific words or phrases.

| Format | Description |
|--------------------|---------------------------------------|
| bold text | Identifies command names. |
| | Identifies keywords and operands. |
| | Identifies the names of GUI elements. |
| <i>italic text</i> | Identifies text to enter in the GUI. |
| | Identifies emphasis. |
| | Identifies variables. |
| Courier font | Identifies document titles. |
| | Identifies CLI output. |

| Format | Description |
|--------|-------------------------------------|
| | Identifies command syntax examples. |

Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

| Convention | Description |
|--------------------|---|
| bold text | Identifies command names, keywords, and command options. |
| <i>italic text</i> | Identifies a variable. |
| [] | Syntax components displayed within square brackets are optional. |
| { x y z } | Default responses to system prompts are enclosed in square brackets. A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options. |
| x y | A vertical bar separates mutually exclusive elements. |
| < > | Nonprinting characters, for example, passwords, are enclosed in angle brackets. |
| ... | Repeat the previous element, for example, <i>member[member...]</i> . |
| \ | Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash. |

Extreme resources

Visit the Extreme website to locate related documentation for your product and additional Extreme resources.

White papers, data sheets, and the most recent versions of Extreme software and hardware manuals are available at www.extremenetworks.com. Product documentation for all supported releases is available to registered users at www.extremenetworks.com/support/documentation.

Document feedback

Quality is our first concern at Extreme, and we have made every effort to ensure the accuracy and completeness of this document. However, if you find an error or an omission, or you think that a topic needs further development, we want to hear from you.

You can provide feedback in two ways:

- Use our short online feedback form at <http://www.extremenetworks.com/documentation-feedback-pdf/>
- Email us at internalinfodev@extremenetworks.com

Provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

Contacting Extreme Technical Support

As an Extreme customer, you can contact Extreme Technical Support using one of the following methods: 24x7 online or by telephone. OEM customers should contact their OEM/solution provider.

If you require assistance, contact Extreme Networks using one of the following methods:

- [GTAC \(Global Technical Assistance Center\)](#) for immediate support
 - Phone: 1-800-998-2408 (toll-free in U.S. and Canada) or +1 408-579-2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact.
 - Email: support@extremenetworks.com. To expedite your message, enter the product name or model number in the subject line.
- [GTAC Knowledge](#) - Get on-demand and tested resolutions from the GTAC Knowledgebase, or create a help case if you need more guidance.
- [The Hub](#) - A forum for Extreme customers to connect with one another, get questions answered, share ideas and feedback, and get problems solved. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.
- [Support Portal](#) - Manage cases, downloads, service contracts, product licensing, and training and certifications.

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any action(s) already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

About This Document

| | |
|---|----|
| • Supported hardware and software | 11 |
| • What's new in this document | 11 |

Supported hardware and software

In those instances in which procedures or parts of procedures documented here apply to some devices but not to others, this guide identifies exactly which devices are supported and which are not.

Although many different software and hardware configurations are tested and supported by Extreme Networks, Inc. for Network OS, documenting all possible configurations and scenarios is beyond the scope of this document.

The following hardware platforms are supported by this release of Network OS:

- ExtremeSwitching VDX 2741
- ExtremeSwitching VDX 2746
- ExtremeSwitching VDX 6740
 - ExtremeSwitching VDX 6740-48
 - ExtremeSwitching VDX 6740-64
- ExtremeSwitching VDX 6740T
 - ExtremeSwitching VDX 6740T-48
 - ExtremeSwitching VDX 6740T-64
 - ExtremeSwitching VDX 6740T-1G
- ExtremeSwitching VDX 6940-36Q
- ExtremeSwitching VDX 6940-144S
- ExtremeSwitching VDX 8770
 - ExtremeSwitching VDX 8770-4
 - ExtremeSwitching VDX 8770-8

To obtain information about a Network OS version other than this release, refer to the documentation specific to that version.

What's new in this document

On October 30, 2017, Extreme Networks, Inc. acquired the data center networking business from Brocade Communications Systems, Inc. This document has been updated to remove or replace references to Brocade Communications, Inc. with Extreme Networks, Inc., as appropriate.

This document is released in conjunction with Network OS 7.3.0.

This document adds the following new chapters:

- [BFD for IP Unnumbered ECMP Interfaces](#) on page 19
- [Configuring an IP Fabric Automatically](#) on page 89

IP Fabrics

| | |
|---|----|
| • Overview of IP Fabrics..... | 13 |
| • Physical topologies and scale..... | 14 |
| • IP unnumbered interfaces..... | 16 |
| • Multihoming..... | 17 |
| • Additional considerations and limitations for IP Fabrics..... | 18 |

Overview of IP Fabrics

Data center networking architectures have evolved with the changing requirements of the modern data center and cloud environments.

The traffic patterns in data center networks are changing rapidly from north-south to east-west. Cloud applications are often multitiered and hosted at different endpoints connected to the network. The communication between these application tiers is a major contributor to the overall traffic in a data center.

These traffic patterns are the primary reasons that data center networks need to evolve into scale-out architectures. Scale-out architectures are built to maximize the throughput for east-west traffic. In addition to providing high east-west throughput, scale-out architectures provide a mechanism to add capacity to the network horizontally, without reducing the provisioned capacity between the existing endpoints. The de-facto industry standard for implementing scale-out architectures is using Clos topologies. These topologies include the 3-stage folded Clos (or leaf-spine topology) and the optimized 5-stage folded Clos. These topologies are described in [Physical topologies and scale](#) on page 14.

With Extreme IP Fabrics, support is provided for a Layer 3 Clos deployment for data center sites, where all the links in the Clos topology are Layer 3 links. An Extreme IP Fabric includes the networking architecture, the protocols used to build the network, turnkey automation features used to provision, manage, and monitor the networking infrastructure and the hardware differentiation with Extreme VDX switches.

Because the infrastructure is built on IP, advantages such as loop-free communication using industry-standard routing protocols, ECMP, very high solution scale, and standards-based interoperability are leveraged.

These are some of the key benefits of deploying a data center site with Extreme IP Fabrics:

- Highly scalable infrastructure: Because the Clos topology is built upon IP protocols, the scale of the infrastructure is very high. The port and rack scales are documented with descriptions of the Extreme IP Fabrics deployment topologies.
- Standards-based and interoperable protocols: Extreme IP Fabrics is built upon industry-standard protocols such as the Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF). These protocols are well understood and provide a solid foundation for a highly scalable solution. In addition, industry-standard overlay control and data plane protocols such as BGP EVPN and Virtual Extensible Local Area Network (VXLAN) are used to extend the Layer 2 domain and extend tenancy domains by enabling Layer 2 communications and virtual machine (VM) mobility.
- Active-active vLAG pairs: By supporting vLAG pairs on leaf switches, dual-homing of the networking endpoints is supported. This provides higher redundancy. Also, because the links are active-active, vLAG pairs provide higher throughput to the endpoints. vLAG pairs are supported for all 10-GbE, 40-GbE, and 100-GbE interface speeds, and up to 32 links can participate in a vLAG.
- Layer 2 extensions: In order to enable Layer 2 domain extension across the Layer 3 infrastructure, VXLAN encapsulation is leveraged. The use of VXLAN provides a very large number of Layer 2 domains to support large-scale multitenancy over the infrastructure. In addition, Extreme BGP EVPN network virtualization provides the control plane for the VXLAN, enabling automatic VTEP discovery and control-plane learning of remote MAC addresses and MAC-IP bindings, thus eliminating

broadcast, unknown unicast, and multicast (BUM) traffic. (For details, refer to [Controllerless Network Virtualization with BGP EVPN](#) on page 27.)

- Multitenancy at Layers 2 and 3: Extreme IP Fabrics provides multitenancy at Layers 2 and 3, enabling traffic isolation and segmentation across the fabric. Layer 2 multitenancy allows an extended range of up to 8000 Layer 2 domains to exist at each ToR switch, while isolating overlapping 802.1q tenant networks into separate Layer 2 domains. Layer 3 multitenancy with VRFs, multi-VRF routing protocols, and BGP EVPN allow large-scale Layer 3 multitenancy. Specifically, Extreme BGP EVPN Network Virtualization leverages BGP EVPN to provide a control plane for MAC address learning and VRF routing for tenant prefixes and host routes, which reduces BUM traffic and optimizes the traffic patterns in the network.
- Support for unnumbered interfaces: Using Extreme Network OS support for IP unnumbered interfaces, only one IP address per switch is required to configure the routing protocol peering. This significantly reduces the planning and use of IP addresses and simplifies operations.
- Turnkey automation: Extreme automated provisioning dramatically reduces the deployment time of network devices and network virtualization. Prepackaged, server-based automation scripts provision Extreme IP Fabrics devices for service with minimal effort.
- Programmable automation: Extreme server-based automation provides support for common industry automation tools such as Python, Puppet, and YANG model-based REST and NETCONF APIs. Prepackaged PyNOS scripting library and editable automation scripts execute predefined provisioning tasks, while allowing customization for addressing unique requirements to meet technical or business objectives when the enterprise is ready.
- Ecosystem integration: Extreme IP Fabrics integrates with leading industry solutions and products such as VMware vSphere, NSX, and vRealize. Cloud orchestration and control are provided through OpenStack and OpenDaylight-based Extreme SDN Controller support.

NOTE

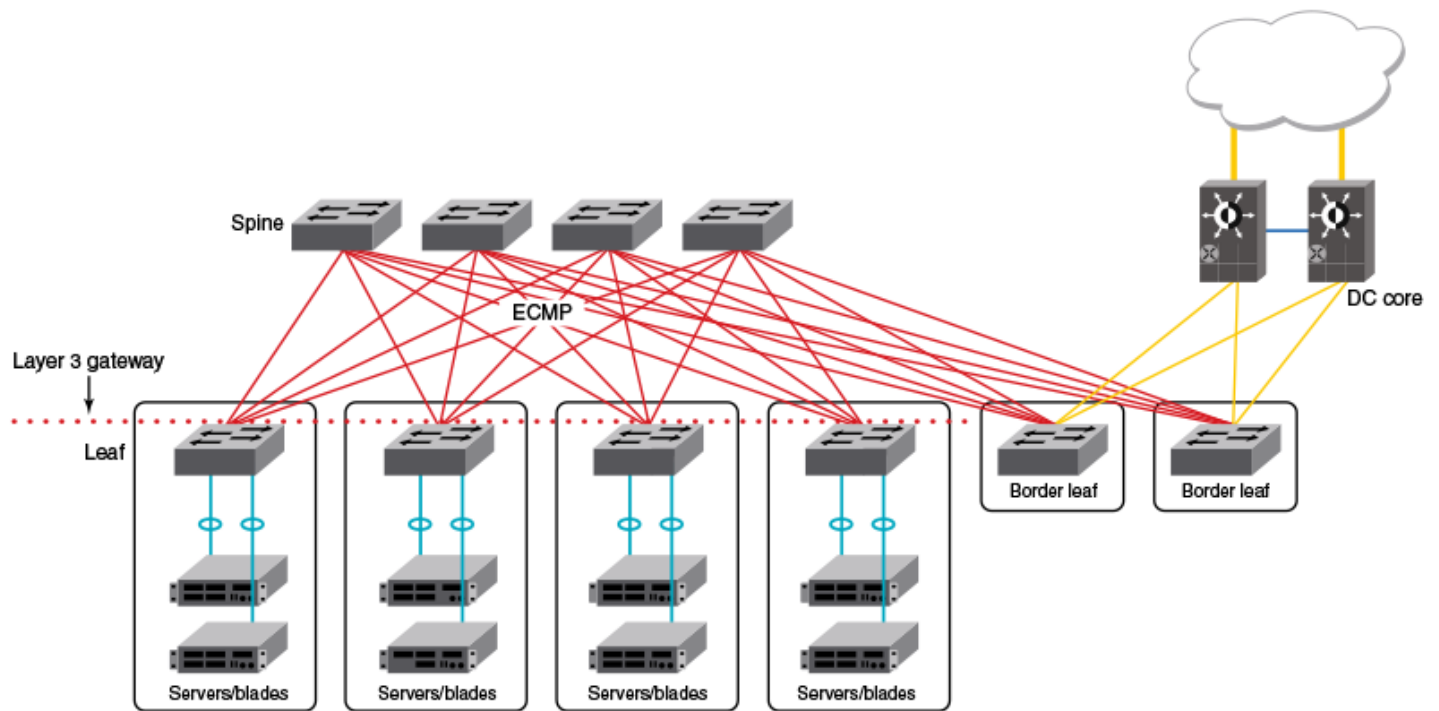
For supported features, refer to the *Extreme IP Fabrics Data Sheet*.

Physical topologies and scale

The basic IP Fabrics physical topologies represent the underlay network and offer different scaling factors.

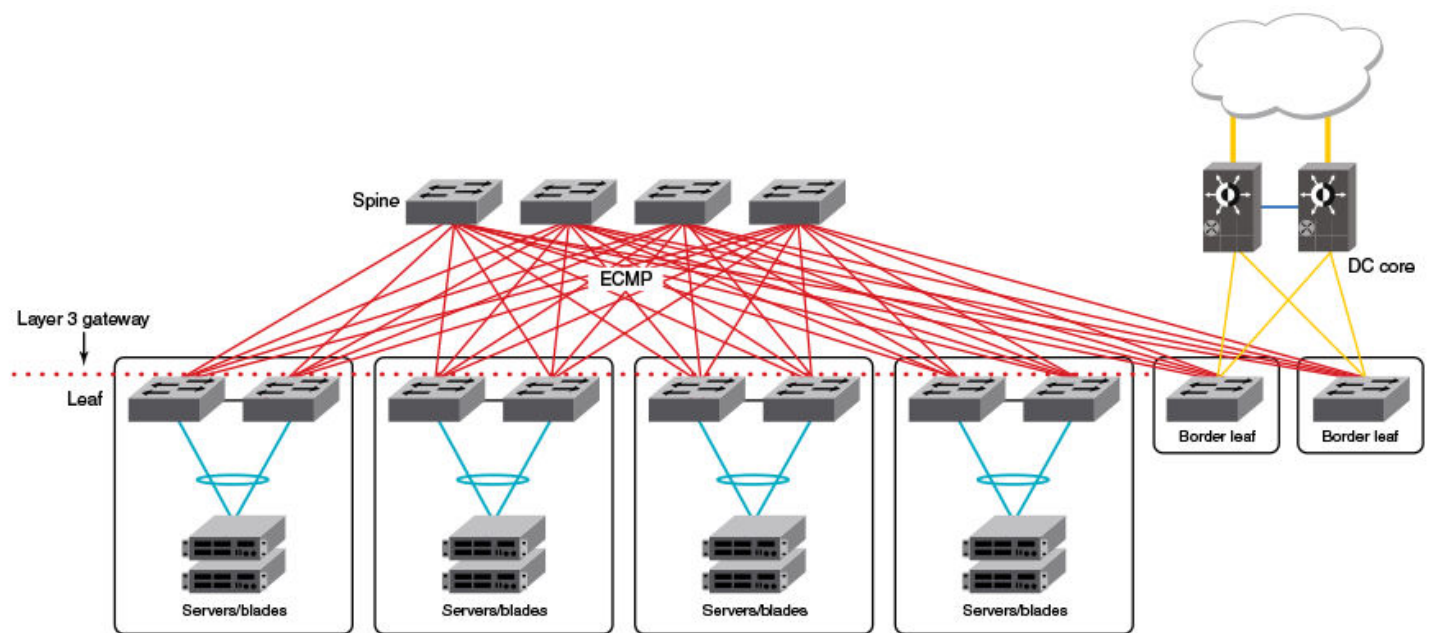
The following topology illustrates a 3-stage folded Clos topology with leaf nodes and spines, with connectivity to the DC core through border leaf nodes. These nodes provide external connectivity to the data center fabric and also provide connectivity to network services such as firewalls and load balancers.

FIGURE 1 A 3-stage folded Clos topology



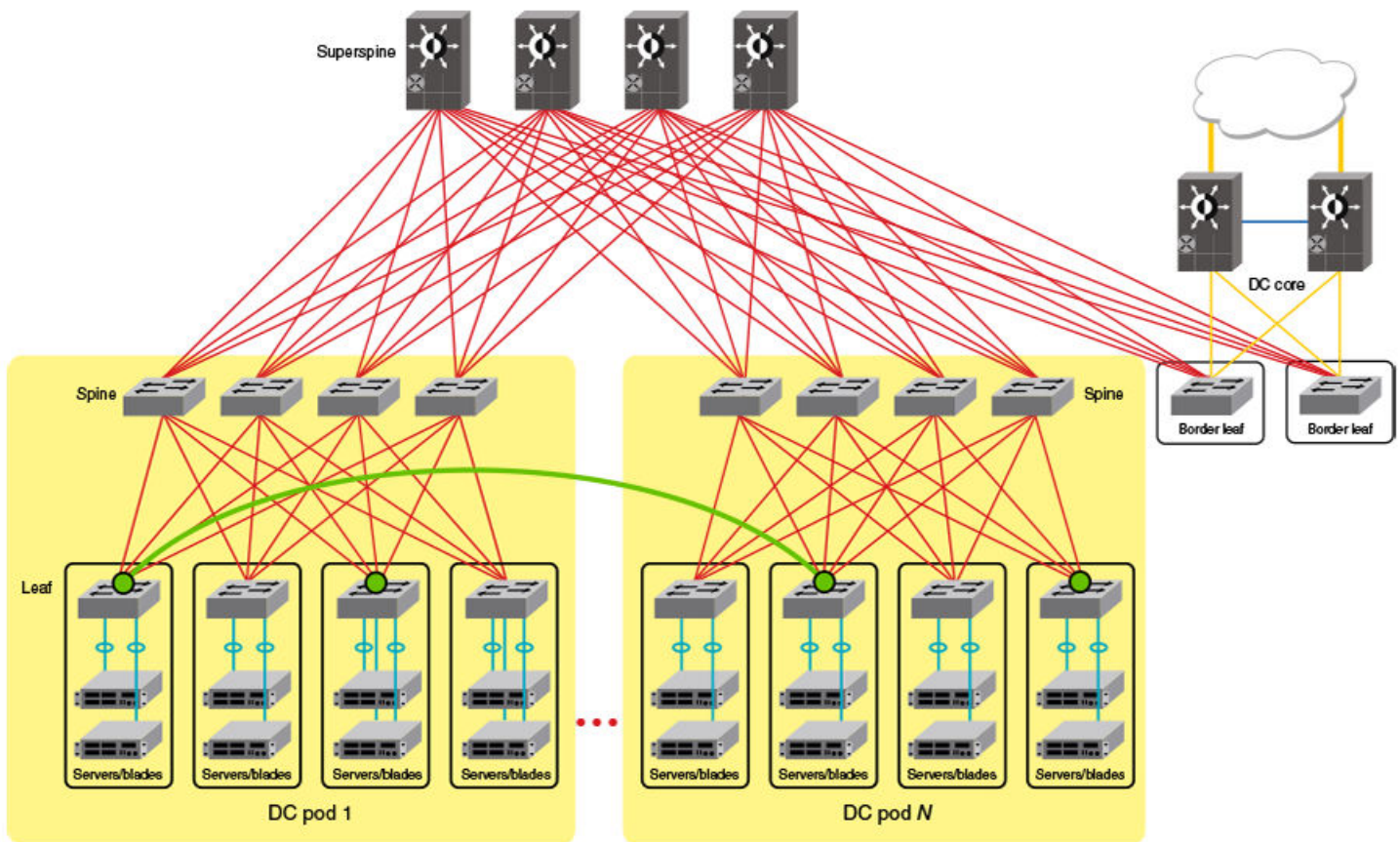
The following topology is similar to the previous topology, but with support for optional high availability (HA) provided through redundant servers or blades dual-homed by means of vLAGs. (Refer to [Multihoming](#) on page 17.)

FIGURE 2 A 3-stage folded Clos topology with vLAG-pair leaf nodes for redundant servers or blades



The following topology uses superspine nodes to interconnect multiple data center pods. Here the border leaf nodes connect directly to the superspine nodes.

FIGURE 3 Optimized 5-stage folded Clos topology



NOTE

For design considerations and scaling details, refer to the "Extreme Data Center Fabric Architectures" white paper. For details of BGP underlay topologies, refer to [Controllerless Network Virtualization with BGP EVPN](#) on page 27.

IP unnumbered interfaces

The IP unnumbered interfaces feature reduces the number of IPv4 addresses required within an IP Fabric, simplifying configuration and maintenance.

With standard /31 addressing, an IP Fabric with four spine nodes and 36 leaf nodes would require 144 links, resulting in 288 addresses. In addition, 40 loopback interfaces are required, for a total of 328 addresses. The solution is to use the IP unnumbered interfaces feature, whereby an IP address is borrowed from a "donor" interface. This is configured by means of the **ip unnumbered** command on a loopback interface and is applied to a physical port.

This reduces the burden on the hardware tables, and in the IP Fabric deployment simplifies the IP address assignment in the fabric. This reduces the number of required addresses to only the 40 /32 addresses required by the loopbacks. The IP addresses and MAC addresses of neighbors are discovered by means of LLDP.

For an example of how to configure this feature, refer to [Configuring the interfaces](#) on page 57. Note the following:

- ARP and Neighbor Discovery (ND) resolution is disabled on unnumbered interfaces. Refer to the [Controllerless Network Virtualization with BGP EVPN](#) on page 27 chapter for the role of these features.
- When eBGP is used in conjunction with the IP unnumbered interfaces feature, it is important to ensure that sufficient Time To Live (TTL) values are available for hello messages to establish separate neighbor adjacencies on leaf switches in an IP Fabric. To do so, use the **neighbor ebgp-multihop** command and set the number of maximum hops to **2**.

NOTE

For a simple three-stage IP Fabric, IP unnumbered interfaces can be used. For a five-stage IP Fabric, numbered interfaces are highly recommended, although IP unnumbered interfaces can be used in five-stage IP Fabric deployments if third-party devices are not included in the design. Extreme does not recommend using a mix of unnumbered and numbered interfaces within an IP Fabric.

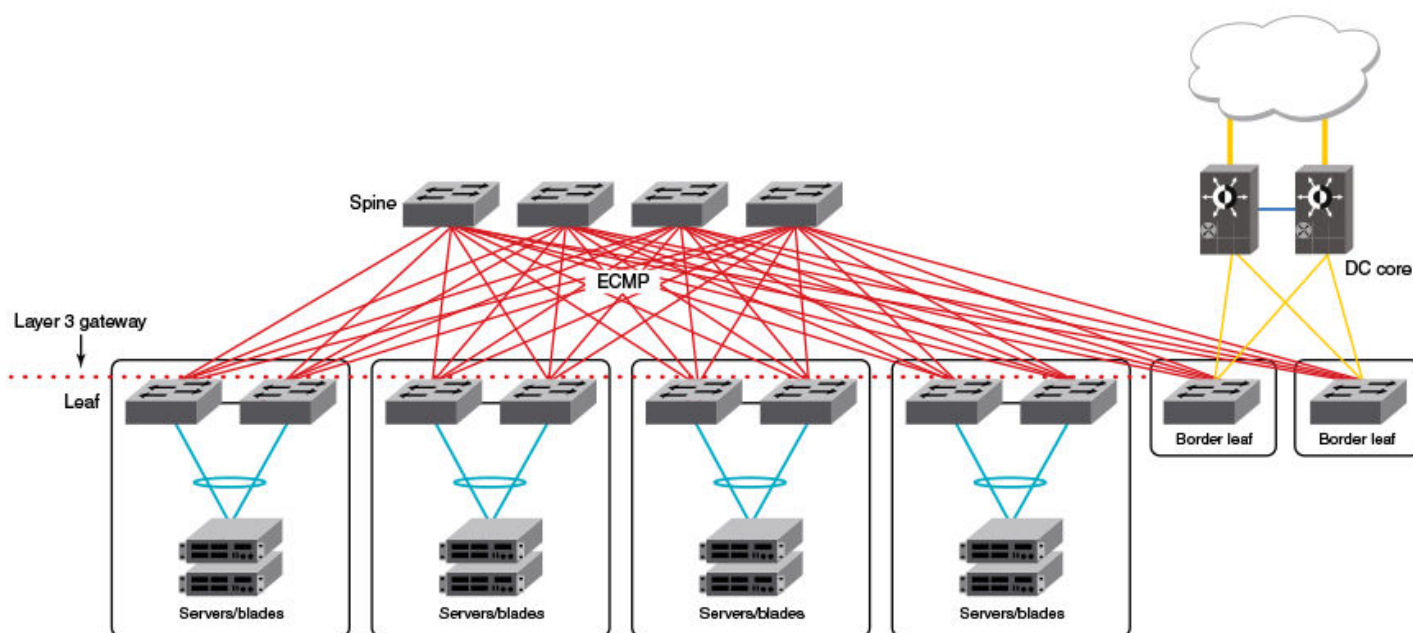
To understand and configure Bidirectional Forwarding Detection (BFD) on IP unnumbered equal-cost multi-path (ECMP) interfaces, refer to [BFD for IP Unnumbered ECMP Interfaces](#) on page 19.

Multihoming

Multihoming is an optional feature that supports high availability.

When servers must be multihomed (specifically, dual homed; refer to the following figure) to more than one leaf switch in an IP Fabric (as for optional HA support), the leaf switches must be configured as a vLAG pair. In this case, (1) RBridge IDs must be unique for each node in the pair, in order to form the local VCS Fabric; (2) the same loopback addresses must be configured on each node to support distributed VTEP if required; (3) both nodes must have the same VCS ID; and (4) a unique router ID (configured by means of the **ip router-id** command) must be configured on each RBridge, so that each RBridge can associate unique route identifiers.

FIGURE 4 A 3-stage folded Clos topology with redundant servers or blades



NOTE

For the details of vLAG configuration, refer to the "Link Aggregation" chapter in the *Extreme Network OS Layer 2 Switching Configuration Guide*.

Note the following considerations and limitations for this scenario:

- A pair of switches to which other servers create dual-homed connections forms a vLAG pair.
- A VXLAN tunnel is not established between the nodes in a vLAG pair.
- Distributed VXLAN gateway functionality is used to extend VXLAN tunnels to other leaf nodes.
- The Static Anycast Gateway and VRRP-E features are options for providing redundant gateway functionality.
- If two leaf nodes are connected by means of an external Layer 2 switch, this can result in a multihoming situation for hosts connected to that switch. This topology is not supported.

When two leaf nodes such as those shown in the previous figure operate in a vLAG pair, they form independent BGP sessions with other routers, including other nodes in the same IP Fabric. It is recommended that the same **local-as** value be configured for all switches belonging to the same vLAG pair. For leaf switches to accept BGP updates, **neighbor allowas-in** must be configured, to disable the AS_PATH check function from rejecting routes that contain the recipient BGP speaker's AS number (ASN).

As noted above, all switches in the vLAG pair must be configured with the same loopback VTEP IP address. This ensures that VTEP discovery does not form a VXLAN tunnel between the two nodes. (The tunnel source and destination IP addresses are the same.) This supports redundancy for the tunnels, as provided in the previous release.

Additional considerations and limitations for IP Fabrics

Consider the following considerations and limitations for IP Fabrics.

- IP Fabrics do not support fabric cluster mode.
- An IP Fabric can include a two-node vLAG pair or a standalone switch in logical chassis cluster mode.
- With the exception of dual homing and the resulting vLAG pair, TRILL is not supported.
- It may be necessary to change an RBridge ID from the default value, as discussed below.

When two or more VDX switches are physically connected, each switch, by default, has an RBridge ID of 1 and a VCS ID of 1. The switches will not try to form a vLAG pair until they have different RBridge IDs. Also by default, each switch is a standalone VCS Fabric in logical chassis cluster mode. Because each switch has the same RBridge ID within the same VCS, no VCS cluster formation will be attempted. You do not want a VCS cluster to be formed. If you want to give each switch a unique RBridge ID to simplify your switch management, you must also change its VCS ID to be unique within the IP Fabric to prevent a VCS Fabric from being formed.

NOTE

The only exception to the above behavior within an IP Fabric is when you want to configure multihoming. For more information, refer to [\(Optional\) Configuring support for dual-homed servers](#) on page 70.

BFD for IP Unnumbered ECMP Interfaces

- Overview of BFD for IP unnumbered ECMP interfaces..... 19
- Configuring BFD for IP unnumbered ECMP interfaces..... 20

Overview of BFD for IP unnumbered ECMP interfaces

Bidirectional Forwarding Detection (BFD) is used in IP networks to check the connectivity between platforms in an IP address pair. This feature, available only on router ports, is used in an IP Fabric deployment of a Clos architecture supporting a data center. Bidirectional Forwarding Detection for IP unnumbered Equal Cost Multi-Path interfaces (BFD for IP unnumbered ECMP interfaces) is supported for Border Gateway Protocol (BGP) in IP Fabric deployments.

Classical BFD

BFD supports the low-overhead, short-duration detection of failures in a path. As a result, BFD has been used extensively in IP networks to detect connectivity between the forwarding engines, including connectivity for a destination that is a single hop or a multihop away. Forwarding path failures detected by BFD are identified to the routing protocols so that nodes can converge and take necessary action to avoid traffic loss. By sending periodic packets, BFD checks to see whether the destination is reachable at any given time. If one of the two nodes running BFD does not receive BFD packets for the configured session for a specified interval, then the connection is deemed failed, and BFD informs the owner (client) of the connectivity loss. As networks may have different paths from a source to a destination, routing protocols choose the best possible paths and configure them in the hardware. The traffic uses these paths to reach the destination.

BFD provides a single mechanism for failure detection for any protocols (for the network layer, the link layer, tunnels, and so on) between two forwarding engines. The BFD packets are carried as the payload of whatever encapsulating protocol is appropriate for the medium and network. BFD eliminates the need for each protocol to implement a similar mechanism of its own, thereby reducing duplication in failure detection protocols.

BFD can provide failure detection on any kind of path between systems, including direct physical links, multihop routed paths, and tunnels. Multiple BFD sessions can be established between the same pair of systems when multiple paths between them are present in at least one direction, even if fewer paths are available in the other direction.

Identifying connectivity for each of the paths in an ECMP IP network can be complicated, as when IP unnumbered interfaces are used. An IP unnumbered interface helps to reduce the IP addresses used in the system. An unnumbered interface gets the IP address from another interface and uses this borrowed IP address just as if it were allocated to the interface.

NOTE

For further information on classical BFD, refer to the "BFD" chapter in the *Network OS Layer 3 Routing Configuration Guide*.

BFD for IP unnumbered ECMP interfaces

BFD for IP unnumbered ECMP interfaces, available only on router ports, is used in an IP Fabric deployment of a Clos architecture supporting a data center.

Identifying connectivity for each of the paths in an ECMP IP network can be complicated, as when IP unnumbered interfaces are used. An IP unnumbered interface helps to reduce the IP addresses used in the system. An unnumbered interface gets the IP address from another interface and uses this borrowed IP address just as if it were allocated to the interface.

Classical BFD is used in IP networks to check for connectivity between IP address pairs. IP Fabric deployments in a data center are built with a Clos network, with a leaf/spine topology. There can be multiple hierarchies that connect to data centers. Typically, ECMP links can be established between the leaf and spine nodes, providing different paths to the destination. ECMP towards the spine provides path redundancy for possible failures in the other links, thereby reducing the time that a given node is offline.

BFD for IP unnumbered ECMP interfaces is currently supported only for BGP. In IP Fabric deployments, BGP creates the session between the leaf and spine nodes. Layer 3 ECMP in the underlay network and multipath BFD are used by BGP to find and detect connectivity.

Considerations and dependencies

For classical BFD, the destination UDP port numbers are 3784 for a single hop and 4784 for a multihop. BFD for IP unnumbered ECMP interfaces uses a destination UDP port number of 6784. This UDP port number is currently reserved for micro BFD by the IETF; however, because the mechanism used in BFD for Layer 3 ECMP is similar to that used for micro BFD, the same UDP port number can be used.

The TTL value to be used is set to 255 by default. This value is changed in accordance with normal routing rules. The source port number is allocated with the same rules as are used for classical BFD. The destination MAC address is used in accordance with normal IP forwarding logic, and there is no relation to the usage-specific MAC address as specified in the micro BFD RFC.

The following conditions are assumed:

- Destination nodes are only one hop away.
- In a system, there are not two or more sessions for the same destination, one from each of the separate sources.
- BFD clients do not create both multipath and classical BFD sessions for the same destination.
- There is no support for multiple session types for the same destination.
- Underlying hardware provides the interface information related to which BFD packet has reached a node.
- Misconfiguration or incomplete configuration is not detected. The user must ensure that configurations are correct.

Configuring BFD for IP unnumbered ECMP interfaces

This task configures BFD for IP unnumbered ECMP interfaces, by means of the **neighbor bfd** command with the **multipath** keyword, and confirms the configuration.

The **multipath** keyword configures multipath BFD for a BGP neighbor, facilitating per-path connectivity status to the router information base (RIB). BGP obtains a status of DOWN when all the paths to the destinations are down, or a status of UP when the first session comes up. The **multipath** keyword can be added or deleted. In case a multipath BFD session is created and the keyword is removed, BGP tears down the existing session and creates a new session with the appropriate session type. When the **multipath** keyword is removed, the **bfd** keyword is retained and a legacy BFD session is created. To remove the BFD sessions entirely, both the **bfd** and **multipath** keywords must be removed.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Specify an RBridge ID.

```
device(config)# rbridge-id 122
```

3. Enable BGP routing and enter BGP router configuration mode.

```
device(config-rbridge-id-122)# router bgp
```

4. Specify a neighbor IP address, enable BFD, and specify multipath BFD.

```
device(config-bgp-router)# neighbor 10.0.6.1 bfd multipath
```

The above configuration uses the default interval, which is device dependent. Refer to the Usage Guidelines of the **neighbor bfd** command for details.

5. Use the **show bfd neighbors details** command to confirm the basic configuration.

```
device# show bfd neighbors details
OurAddr      NeighAddr      State  Int      RBridge-ID Session-Type
20.0.6.1      10.0.6.1        UP      Te1/0/1   RB1         SH/MH/MP

Local      State: UP  Diag: 0  Demand mode: 0 Poll: 0
Received State: UP  Diag: 0  Demand mode: 0 Poll: 0 Final: 0
Local      MinTxInt(ms): 1000  MinRxInt(ms): 1000  Multiplier: 5
Received MinTxInt(ms): 1000  MinRxInt(ms): 1000  Multiplier: 5
  Rx Count: 3806  Tx Count: 4308
  LD/RD: 10001/10001  Heard from Remote: Y
Current Registered Protocols: BGP
Uptime: 0 day 0 hour 0 min 0 sec 0 msec
```

6. Use the **show bfd neighbors session-type multipath details** command to confirm the multipath and session configuration.

```
device# show bfd neighbors session-type multipath details
OurAddr      NeighAddr      State  Int      RBridge-ID Session-Type
20.0.6.1      10.0.6.1        UP      Lo1       RB1         MP

Local      State: UP  Diag: 0  Demand mode: 0 Poll: 0
Received State: UP  Diag: 0  Demand mode: 0 Poll: 0 Final: 0
Local      MinTxInt(ms): 1000  MinRxInt(ms): 1000  Multiplier: 5
Received MinTxInt(ms): 1000  MinRxInt(ms): 1000  Multiplier: 5
  Rx Count: 3806  Tx Count: 4308
  LD/RD: 10001/10001  Heard from Remote: Y
Current Registered Protocols: BGP
Uptime: 0 day 0 hour 0 min 0 sec 0 msec
```


Multicast in IP Fabrics

- IGMP snooping in IP Fabrics.....23
- PIM-SM in IP Fabrics.....24

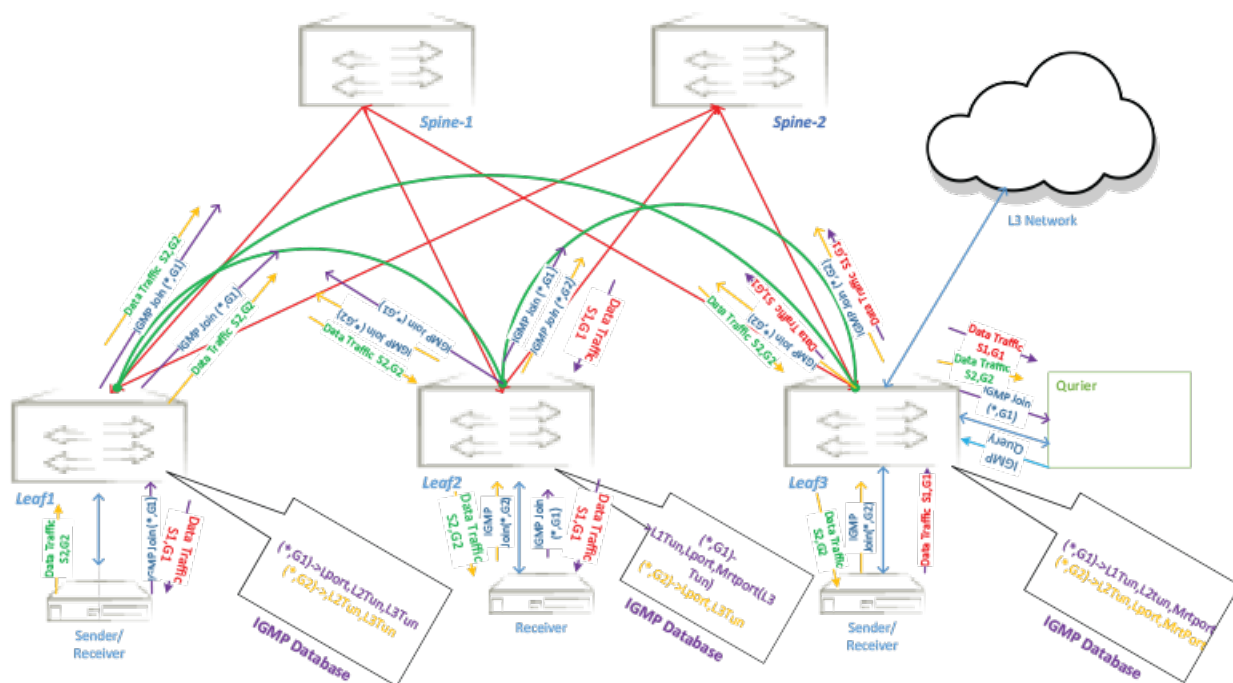
Many data center deployments are required to process multicast traffic efficiently, to use the interconnect links bandwidth and avoid flooding. IGMP snooping is used to achieve this for intra-VLAN multicast traffic. This chapter explains how IGMP snooping, as well as Protocol Independent Multicast-Sparse Mode (PIM-SM) in the default VRF, works in IP Fabrics.

IGMP snooping in IP Fabrics

Internet Group Management Protocol (IGMP) is supported in IP Fabrics.

IGMP snooping can be enabled on VLANs on leaf nodes. Each leaf node is aware of local receivers and sends reports to multicast router (mrrouter) querier-connected ports and all VXLAN tunnels. With this approach, each leaf is aware of its local receivers as well as those of other leafs. Restrict unknown multicast (RUM) can be enabled to restrict the unknown multicast traffic flooding to local edge ports. Traffic received on a VXLAN tunnel is not forwarded to other tunnels; instead, it is forwarded only to local edge ports. The following figure illustrates an IGMP snooping topology.

FIGURE 5 IGMP snooping in IP Fabrics



Behavior differs depending on whether the IGMP snooping source and the mrouter querier are connected to the same leaf node or not, as detailed below.

IGMP snooping source and querier connected to same leaf node

In this case, If there are hosts joined in to the multicast group on other leafs or edge ports, those hosts are known to the sender leaf as other the other leafs send reports to the querier that is on the sender leaf or is connected to the sender leaf. Consequently, traffic from the source is sent to all leafs that have interested receivers. Refer to (S1,G1) for data traffic and (*,G1) for control flows in the above figure.

For known multicast traffic (if there are local receivers or receivers are learned from other leafs, ingress traffic is replicated to the corresponding peer leaf's VXLAN tunnels and any local receiver ports. On receiving this traffic, the other leafs forward it to interested receivers. If there are no interested receivers, then the traffic is treated as unknown multicast.

Unknown traffic with RUM enabled is sent on the mrouter port if the querier is connected to the source leaf. Otherwise the traffic is dropped.

Unknown traffic without RUM enabled is flooded to all edge ports and VXLAN tunnels connected to the other leafs. On receiving this traffic the other leafs flood it to the all the edge ports.

IGMP snooping source and querier not connected to same leaf node

In this case, If there are hosts joined in to the multicast group on other leafs, those hosts are known to the sender leaf as other leafs send reports to the sender leaf over a VXLAN tunnel. Consequently, traffic coming from the source is subject to being forwarded to other leafs to which interested receivers are connected. Refer to (S2,G2) for data traffic and (*,G2) for control flows in the above figure.

If the sender leaf has only local receivers, the traffic is treated as known traffic and is sent to the local interested receivers and the mrouter port to which the querier is connected.

If the sender leaf does not have local or remote receivers, the traffic forwarding depends on the RUM configuration. If RUM is configured, traffic is sent on the mrouter port and does not reach the other leafs. If RUM is not configured, traffic is flooded to all edge ports and VXLAN tunnels connected to the other leafs. On receiving the traffic, the other leafs flood it to all the edge ports.

If the sender leaf has remote receivers, traffic is treated as known traffic and is sent to the local interested receivers and the mrouter port to which the querier is connected. This traffic reaches the other leafs if they have interested listeners.

PIM-SM in IP Fabrics

Protocol Independent Multicast-Sparse Mode (PIM-SM) is supported in IP Fabrics.

Because multicast protocols are not supported on multiple VRFs, Layer 3 Multicast in IP Fabrics works only in the default VRF. PIM can be enabled on all leaf nodes. A rendezvous point (RP) can be configured on spine nodes within or outside an IP Fabric.

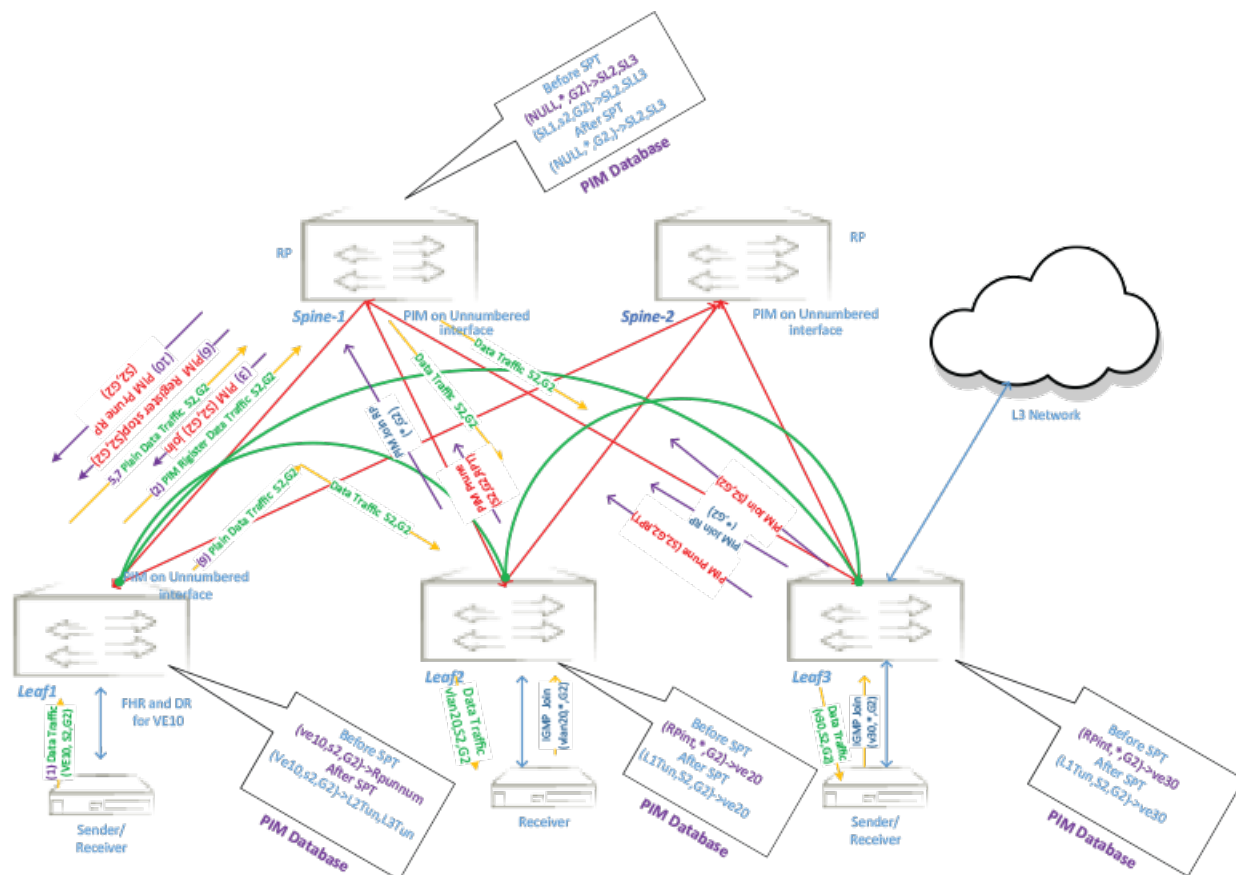
PIM can be configured on the VE interfaces on leaf nodes. PIM Hello messages are exchanged over VXLAN tunnels for the corresponding VE interfaces. One leaf can be elected as the designated router (DR), and multicast traffic can be routed on that DR. From there, traffic is switched to other leaf nodes over VXLAN tunnels.

The RP can be configured on IP Fabric spine nodes on a loopback interface. The RP can also be configured outside of the IP Fabric. PIM works in the default VRF with both symmetric and asymmetric unicast routing.

RP at the spine in an IP Fabric with symmetric/asymmetric routing in the default VRF

The following figure illustrates how PIM-SM operates at the spine in an IP Fabric with symmetrical/asymmetrical routing in the default VRF.

FIGURE 6 PIM-SM at the spine in the default VRF



PIM can be enabled on each leaf, and one designated router (DR) is elected. There may be no optimal path for the traffic from the spine to a leaf node, because the traffic comes to the DR and from there is routed or switched to other leaf nodes. Because the connected subnet is available on multiple leaf nodes, PIM should be enabled on all multicast virtual Ethernet (VE) interfaces on those nodes. Because the route path from spine to source is equal-cost multipath (ECMP) to the leaf nodes having the same subnet, (S,G) joins may go to a leaf other than the leaf to which the source is connected. The "other" leaf creates an (S,G) entry like an intermediate router; the traffic coming from the source (which is the ingress VLAN) is flooded on the first-hop router (FHR), comes to this leaf, and takes the (S,G) path.

For asymmetric traffic, PIM works by means of connected VEs across the leaf nodes. For symmetric traffic, PIM works by means of connectivity through the underlay network.

The following sequence summarizes the flow of asymmetric traffic from VE 10 to VE 20 (VE 10 is extended on Leaf1 and Leaf2). PIM must be enabled on both leaf nodes. The source is connected to Leaf1 on VE 10. Leaf1 is the DR. Receiver R1 is on VE 20 at Leaf2, and Leaf2 is the DR for 20.

1. Leaf2 creates (rp_if(unnumbered),*,G2) and sends (*,G2) toward the RP on Spine-1.
2. Spine-1 receives (*,G2) and creates a (*,G2) entry.
3. The source sends traffic, and Leaf1 receives it as a DR and sends the register traffic to Spine-1.
4. Spine-1 sends traffic to Leaf2 on the (*,G2) tree.

5. Spine-1 sends an (S,G2) Join message towards the interface where the source is connected. Here two routes are possible, through Leaf1 or Leaf2. If the (S,G2) join is sent toward Leaf1, then Leaf1 sends plain traffic to the RP (Spine-1). The RP sends a Register-Stop message to Leaf1, which stops the register traffic. If an (S,G2) join is sent towards Leaf2, as the source is in a directly connected network, Leaf2 creates an (S,G2) entry with the outgoing interface towards the RP. Because the source traffic is flooded on the VLAN and the PIM neighbor is connected on the tunnel port on Leaf1, the traffic comes to Leaf2, hits the (S,G2) entry, and is sent to the RP. The RP sends a Register-Stop message to Leaf1, stopping the register traffic. (This case is a traffic tromboning case.)
6. If a shortest-path tree (SPT) threshold is configured at Leaf2, Leaf2 creates an (S,G) entry with outgoing interface VE 20 (to which the source is connected) and sends an (RPT,S,G2) Prune message towards RP, which in turn sends the (S,G2) Prune towards Leaf1. The source traffic coming on VE 10 is sent to Leaf2 as an ingress VLAN flood. Leaf2 sends the traffic on (S,G2) towards receiver R1 on VE 20. Periodic Null registers are sent from Leaf1 to the RP. The above scenario should work seamlessly with the two-node VCS, assuming that the ingress VLAN flood of multicast traffic towards the distributed VXLAN tunnel is not duplicated.

The following sequence summarizes the flow of symmetric traffic from VE 10 to VE 30. (VE 10 is not extended on Leaf3.) Because VE 10 is extended on two leaf nodes, PIM must be enabled on both leaf nodes. The source is connected to Leaf1 on VE 10. Leaf1 is the DR. Receiver R1 is on VE 30 at Leaf3 and Leaf3 is the DR for VE 30.

1. Leaf2 creates (rp_if(unnumbered),*,G2) and sends (*,G2) toward the RP on Spine-1.
2. Spine-1 receives an (*,G2) Join message from Leaf3 and creates an (*,G2) entry.
3. The source sends the traffic and Leaf1 receives it; as a DR, Leaf1 sends the traffic to Spine-1.
4. Spine-1 sends the traffic to Leaf2 on the (*,G2) tree.
5. Spine-1 sends an (S,G2) join towards the interface through which the source is connected. Here there are two possible routes, through Leaf1 or Leaf2. If an (S,G2) Join is sent toward Leaf1, then Leaf1 sends plain traffic to the RP (Spine-1). Spine-1 sends a Register-Stop to Leaf1, which stops the register traffic. If an (S,G2) Join is sent towards Leaf2, as the source is in a directly connected network, Leaf2 creates an (S,G2) entry with the outgoing interface toward the RP. As the source traffic is flooded on the VLAN and the PIM neighbor is connected through the tunnel port on Leaf1, the traffic comes to Leaf2, hits the (S,G2) entry, and is sent to the RP. The RP in turn sends a Register-Stop message to Leaf1, which stops the register traffic. (This case is a traffic tromboning case.)
6. Spine-1 sends the traffic to Leaf3.
7. If an SPT threshold is configured on Leaf3, and VE 10 is not connected to the source, reachability is through the spine. Spine-1 or Spine-2 can be the next hop from the source on Leaf3. If the next hop is Spine-1 (the RP), Spine-1 keeps the outgoing interface of the existing receiver R2 and traffic is sent to Leaf3. If the next hop is Spine-2, Spine-2 creates the (S,G) entry and sends a Join towards Leaf1 or Leaf2. If an (S,G2) Join is sent toward Leaf1, then Leaf1 sends plain traffic to Spine-1 or Spine-2. If an (S,G2) Join is sent towards Leaf2, as the source is in a directly connected network, Leaf2 creates an (S,G2) entry with the outgoing interface toward Spine-1 or Spine-2. As the source traffic is flooded on the VLAN and the PIM neighbor is connected on a tunnel port on Leaf1, the traffic comes to Leaf2, hits the (S,G2) entry, and is sent to Spine-1 or Spine-2, which send the traffic to Leaf3. Periodic Null registers are sent from Leaf1 to the RP.

The above scenario should work seamlessly with the two-node VCS, assuming that the ingress VLAN flood of multicast traffic towards the distributed VXLAN tunnel is not duplicated.

RP outside the IP Fabric with symmetric/asymmetric routing in the default VRF

The sequence of events for the scenario above applies to the scenario where the RP is outside the IP Fabric.

Controllerless Network Virtualization with BGP EVPN

| | |
|--|----|
| • Overview of controllerless network virtualization with BGP EVPN..... | 27 |
| • BGP-based underlay architecture supporting BGP EVPN..... | 30 |
| • BGP EVPN NLRI..... | 33 |
| • MAC address learning..... | 34 |
| • EVPN instances..... | 34 |
| • BGP L2VPN EVPN address family..... | 35 |
| • Automatic VXLAN tunnel endpoint discovery..... | 36 |
| • BGP next-hop-unchanged..... | 37 |
| • BGP retain route target..... | 37 |
| • RIBout AS path check..... | 37 |
| • BGP legacy features supported for the L2VPN EVPN address family..... | 37 |
| • Ethernet Segment Identifiers for BGP routing..... | 38 |
| • BGP dynamic neighbors..... | 39 |
| • Multi-VRF for BGP EVPN..... | 43 |
| • Static anycast gateway..... | 44 |
| • ARP and ND scaling enhancements | 47 |
| • Standards conformance and RFC support for BGP EVPN..... | 51 |

Overview of controllerless network virtualization with BGP EVPN

Layer 2 extension mechanisms using VXLAN rely on “flood and learn” mechanisms. These mechanisms are very inefficient, making MAC address convergence longer and resulting in unnecessary flooding.

Also, in a data center environment with VXLAN-based Layer 2 extension mechanisms, a Layer 2 domain and an associated subnet might exist across multiple racks and even across all racks in a data center site. With traditional underlay routing mechanisms, routed traffic destined to a VM or a host belonging to the subnet follows an inefficient path in the network, because the network infrastructure is aware only of the existence of the distributed Layer 3 subnet, but is not aware of the exact location of the hosts behind a leaf switch.

With Extreme BGP EVPN network virtualization, network virtualization is achieved through creation of a VXLAN-based overlay network. Extreme BGP EVPN network virtualization leverages BGP EVPN to provide a control plane for the virtual overlay network. BGP EVPN enables control-plane learning for end hosts behind remote VXLAN tunnel endpoints (VTEPs). This learning includes reachability for Layer 2 MAC addresses and Layer 3 host routes.

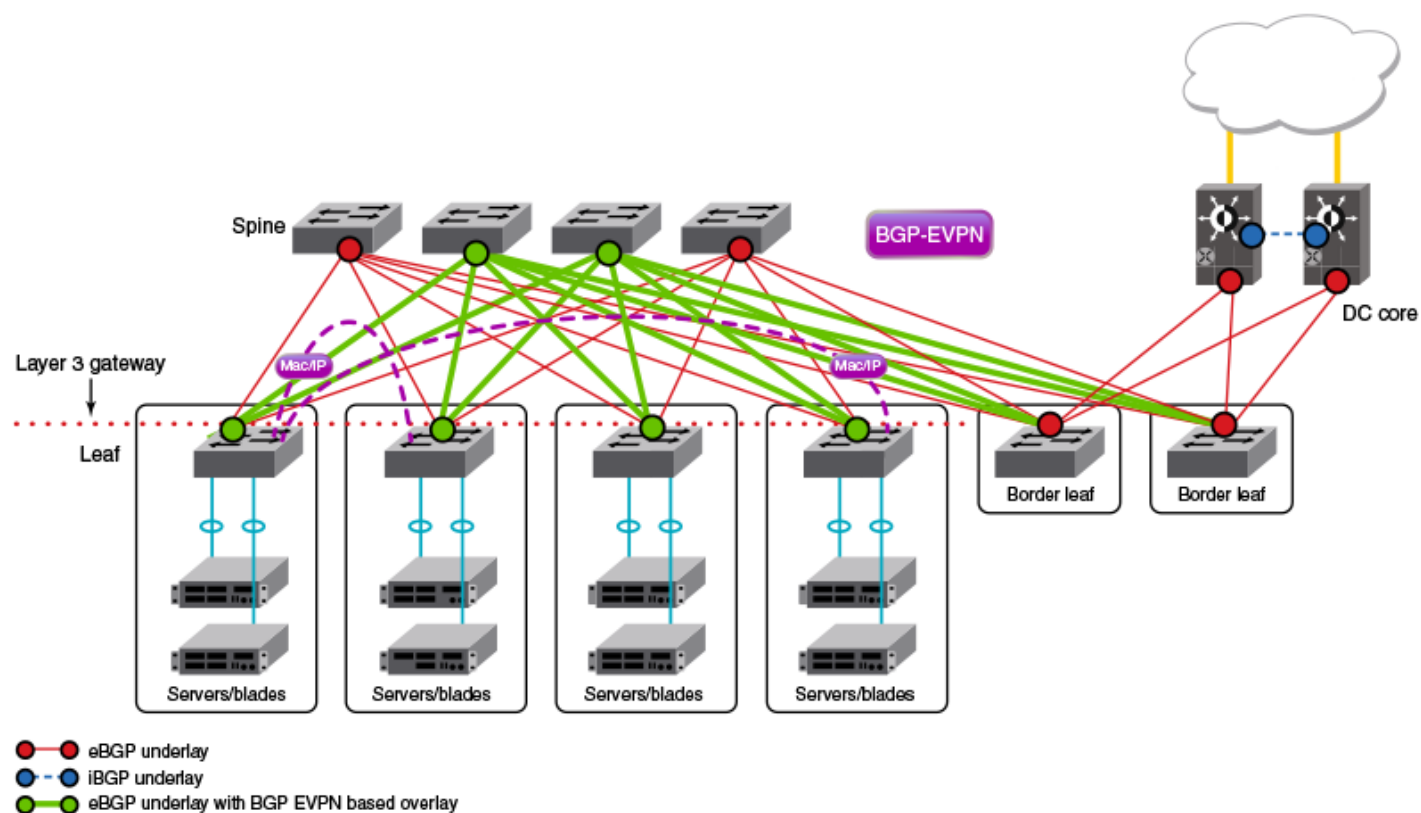
Some key features and benefits of Extreme BGP EVPN network virtualization are summarized as follows:

- Active-active vLAG pairs: node pairs for multiswitch port channel for dual homing of network endpoints are supported at the leaf. Both the switches in the vLAG pair participate in the BGP EVPN operations and are capable of actively forwarding traffic.
- Static anycast gateway: With static anycast gateway technology, each leaf is assigned the same default gateway IP and MAC addresses for all the connected subnets. This ensures that local traffic is terminated and routed at Layer 3 at the leaf. This also eliminates any suboptimal inefficiencies found with centralized gateways. All leaves are simultaneously active forwarders for all default traffic for which they are enabled. Also, because the static anycast gateway does not rely on any control plane protocol, it can scale to large deployments.

- **Efficient VXLAN routing:** With the existence of active-active leaf and the static anycast gateway, all traffic is routed and switched at the leaf. Routed traffic from the network endpoints is terminated in the leaf and is then encapsulated in VXLAN header to be sent to the remote site. Similarly, traffic from the remote leaf node is VXLAN-encapsulated and needs to be decapsulated and routed to the destination. This VXLAN routing operation into and out of the tunnel on the leaf switches is enabled in the Extreme platform ASICs. VXLAN routing performed in a single pass is more efficient than with competitive ASICs.
- **Data plane IP and MAC learning:** With IP host routes and MAC addresses learned from the data plane and advertised with BGP EVPN, the leaf switches are aware of the reachability information for the hosts in the network. Any traffic destined to the hosts takes the most efficient route in the network.
- **Layer 2 and Layer 3 multitenancy:** BGP EVPN provides control plane for VRF routing as well as for Layer 2 VXLAN extension. BGP EVPN enables a multitenant infrastructure and extends it across the data center site to enable traffic isolation between the Layer 2 and Layer 3 domains, while providing efficient routing and switching between the tenant endpoints.
- **Dynamic tunnel discovery:** With BGP EVPN, the remote VTEPs are automatically discovered. The resulting VXLAN tunnels are also automatically created. This significantly reduces Operational Expense (OpEx) and eliminates errors in configuration.
- **ARP/ND suppression:** As the BGP EVPN EVI leaves discover remote IP and MAC addresses, they use this information to populate their local ARP tables. Using these entries, the leaf switches respond to any local ARP queries. This eliminates the need for flooding ARP requests in the network infrastructure.
- **Conversational ARP/ND learning:** Conversational ARP/ND reduces the number of cached ARP/ND entries by programming only active flows into the forwarding plane. This helps to optimize utilization of hardware resources. In many scenarios, there are software requirements for ARP and ND entries beyond the hardware capacity. Conversational ARP/ND limits storage-in-hardware to active ARP/ND entries; aged-out entries are deleted automatically.
- **VM mobility support:** If a VM moves behind a leaf switch, with data plane learning, the leaf switch discovers the VM and learns its addressing information. It advertises the reachability to its peers, and when the peers receive the updated information for the reachability of the VM, they update their forwarding tables accordingly. BGP EVPN-assisted VM mobility leads to faster convergence in the network.
- **Simpler deployment:** With multi-VRF routing protocols, one routing protocol session is required per VRF. With BGP EVPN, VRF routing and MAC address reachability information is propagated over the same BGP sessions as the underlay, with the addition of the L2VPN EVPN address family. This significantly reduces OpEx and eliminates errors in configuration.
- **Open standards and interoperability:** BGP EVPN is based on the open standard protocol and is interoperable with implementations from other vendors. This allows the BGP EVPN-based solution to fit seamlessly in a multivendor environment

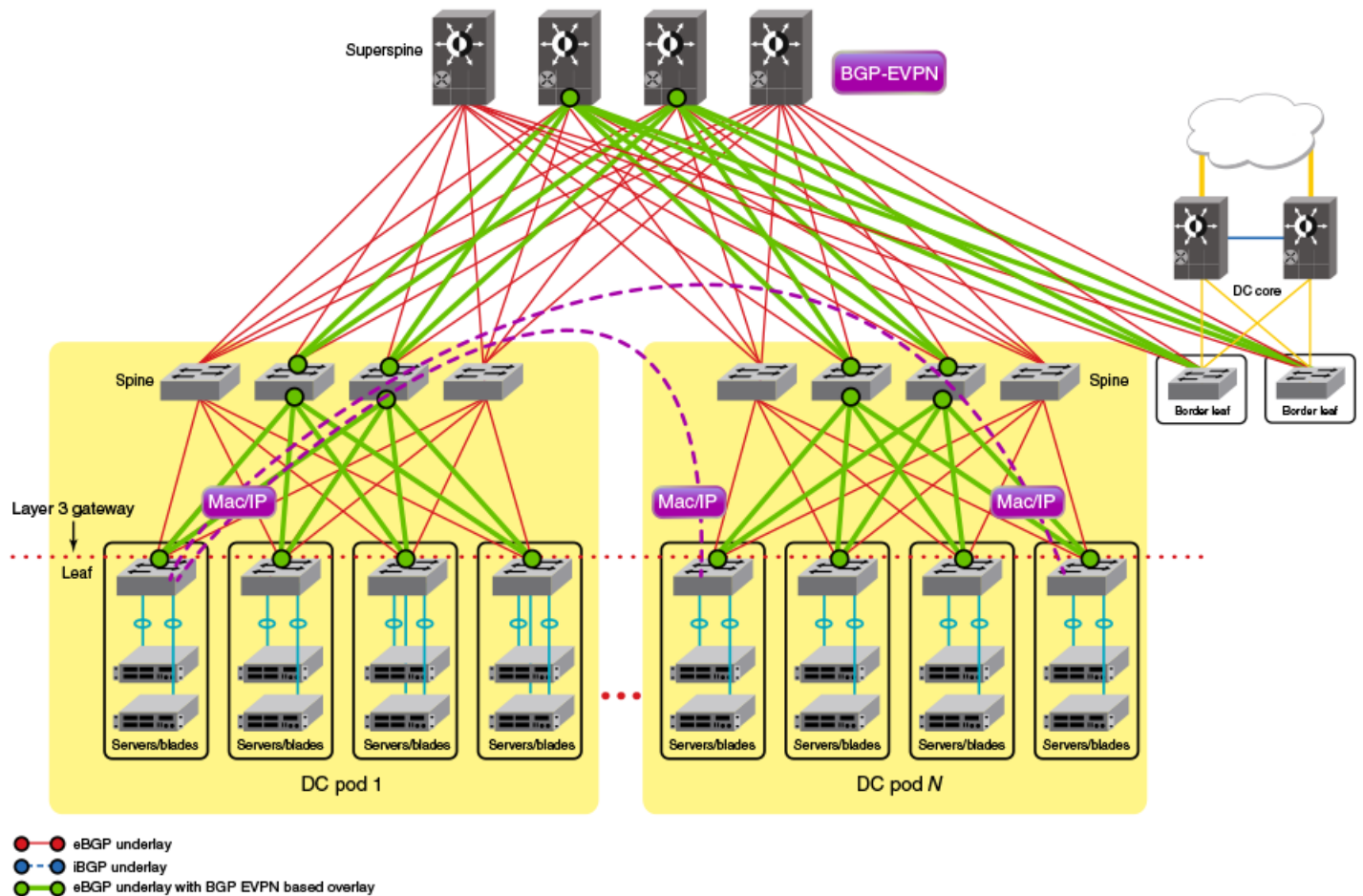
The following illustration depicts a 3-stage Clos topology that uses BGP EVPN. VXLAN tunnels are represented by dashed lines. This topology allows the provider to extend VLANs across racks with support from a Layer 3 underlay. Note that there is no controller, as VTEP autodiscovery is used. VTEPs are automatically discovered across the racks, and all MAC and IP address learning occurs automatically on the control plane. In this case, manual or controller intervention is not required. BGP EVPN can be extended to the border leaf nodes.

FIGURE 7 Overlay for a 3-stage Clos network without a controller



The following figure depicts a topology similar to that shown above, but with an optional optimized 5-stage folded Clos topology (superspine) for data center connectivity. Here the border leaf nodes connect directly to the superspine nodes. BGP EVPN can be extended to the border leaf nodes.

FIGURE 8 Overlay for an optimized 5-stage folded Clos topology without a controller



BGP-based underlay architecture supporting BGP EVPN

This section illustrates the basic BGP underlay architecture that is required to support the BGP EVPN overlay.

- [eBGP-based BGP EVPN](#) on page 30
- [iBGP-based BGP EVPN](#) on page 32

eBGP-based BGP EVPN

When eBGP is used for BGP EVPN in an IP Fabric, spine switches are configured to be in one autonomous system (AS). Each leaf switch or ToR pair is configured to be in a different AS from that of the spine switches, and advertises the routes by using local VTEP IP addresses as the next hop address.

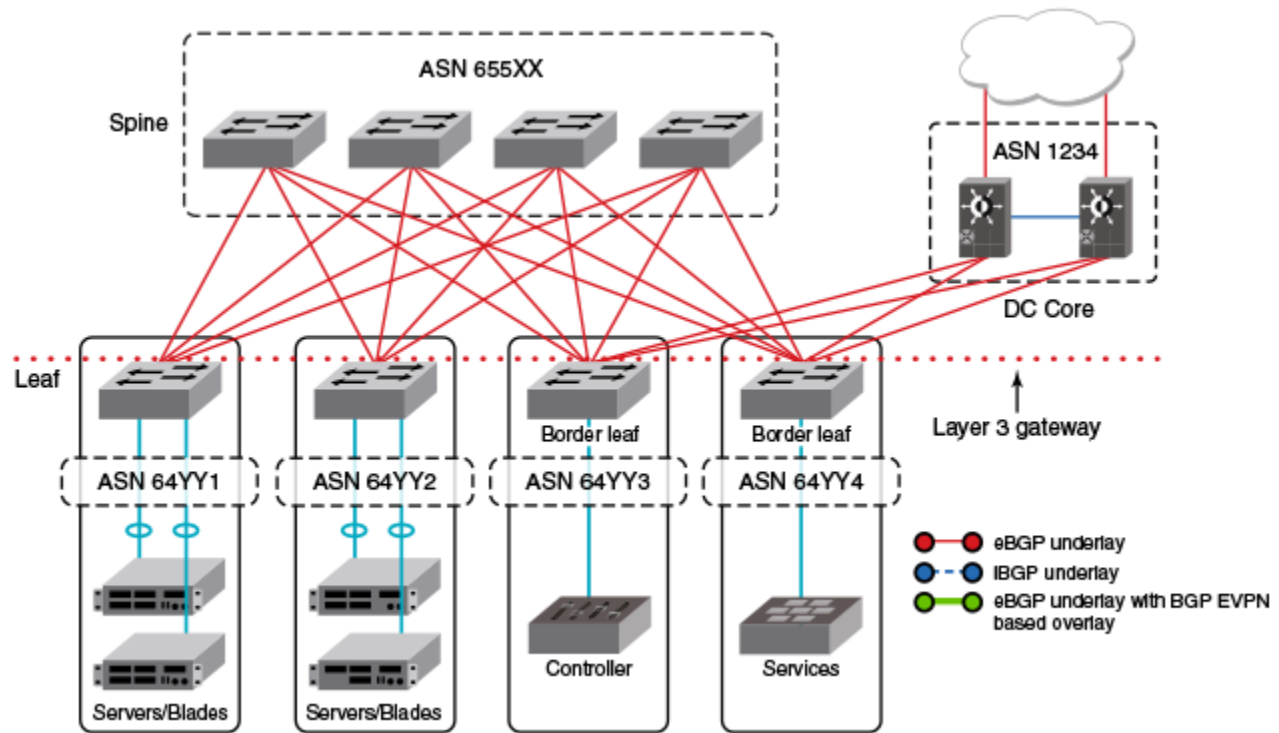
For the BGP EVPN address family, spine switches are configured to advertise these routes to other eBGP peers, leaving the next-hop attribute unchanged.

NOTE

The spines are configured with separate VCS IDs.

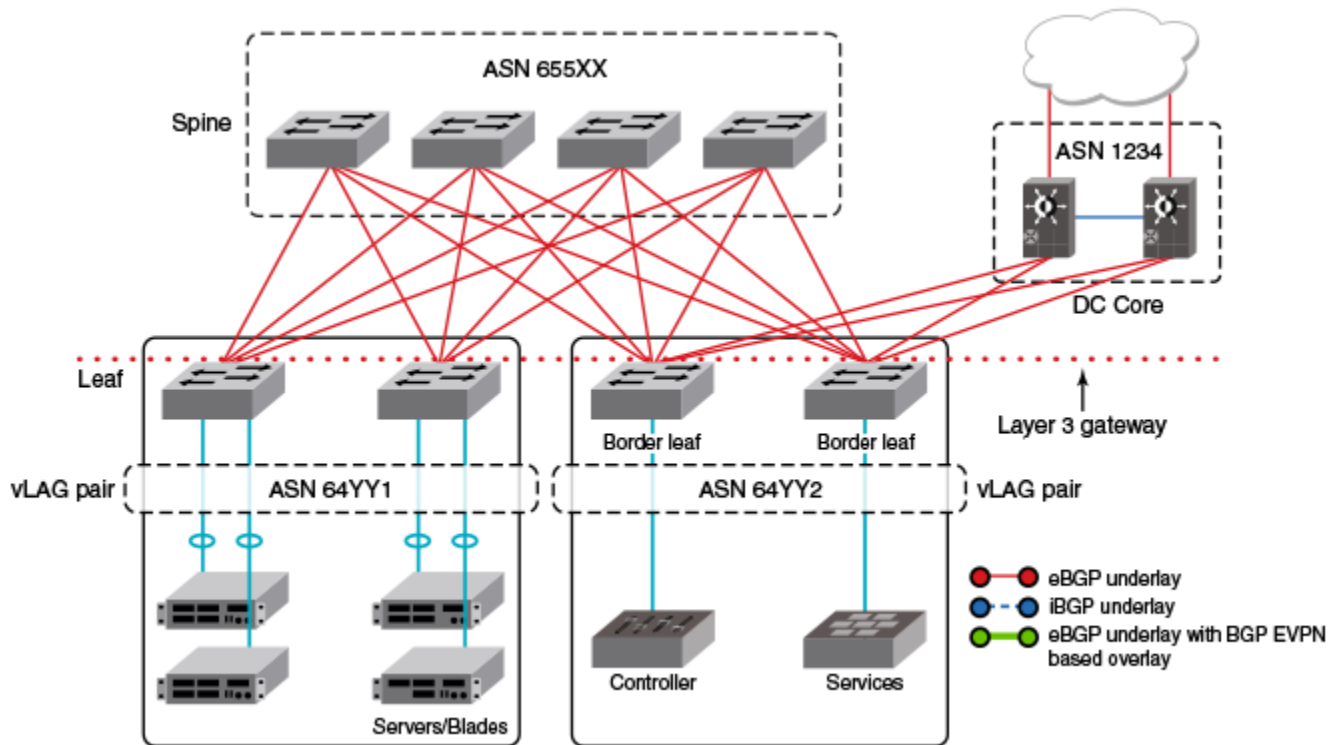
The following figure shows the eBGP underlay configuration in an IP Fabric where the spine switches are configured to be in one AS. Each leaf switch is configured to be in a separate AS.

FIGURE 9 eBGP underlay configuration with leaf switches in a separate autonomous system



The following figure shows the eBGP underlay configuration in an IP Fabric where the spine switches are configured to be in one AS. Leaf 1 and leaf 2 are configured to be in one AS, and leaf 3 and leaf 4 are configured to be in another AS.

FIGURE 10 eBGP underlay configuration with leaf switches in two separate autonomous systems

**NOTE**

When eBGP is used in conjunction with the IP unnumbered interfaces feature, it is important to ensure that sufficient Time To Live (TTL) values are available for hello messages to establish separate neighbor adjacencies on leaf switches in an IP Fabric. To do so, use the **neighbor ebgp-multihop** command and set the number of maximum hops to 2. Refer to the *Command Reference* for more information.

iBGP-based BGP EVPN

When iBGP is used for BGP EVPN in an IP Fabric, spine switches must be configured as route reflectors (RRs). This is because iBGP generally requires a switch to peer with every other device in the IP Fabric. When the spine switch is configured as an RR this situation is avoided.

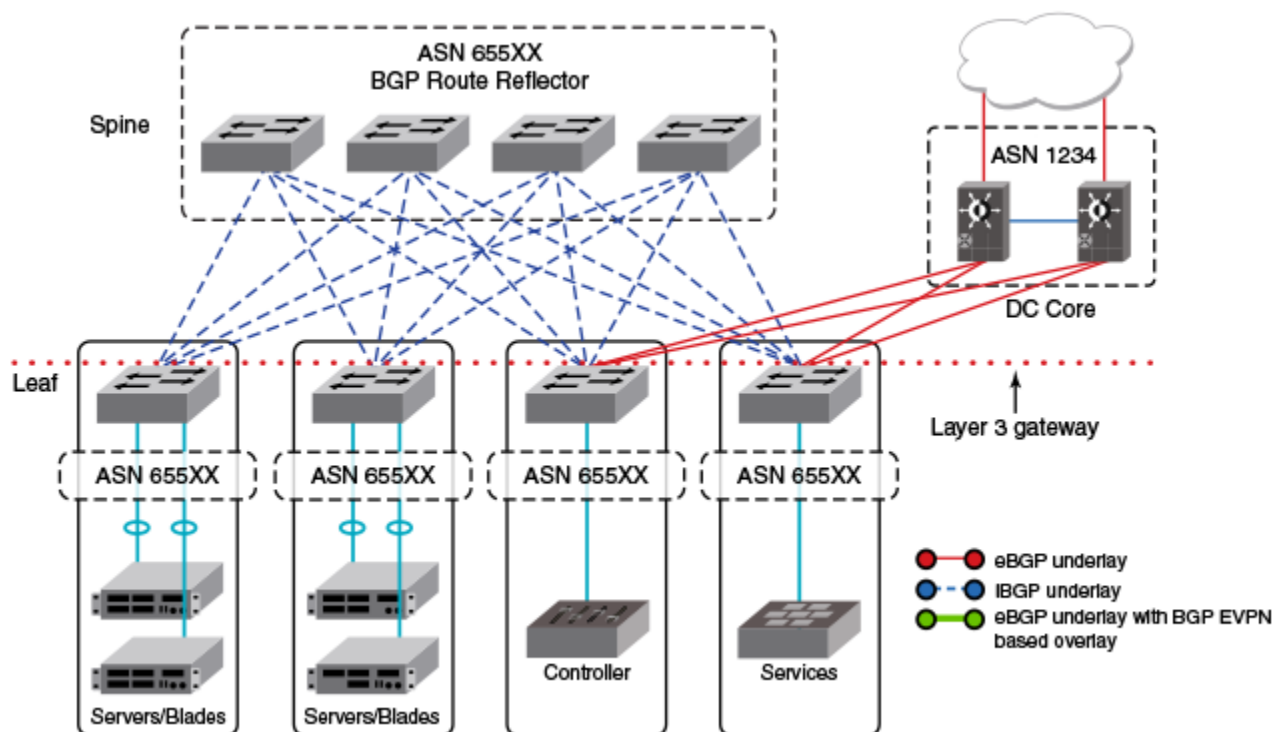
However, BGP RRs only reflect the best route. Only the single, best prefix is reflected to each leaf, which is configured as an RR client, even if multiple Equal-Cost Multipath (ECMP) routes exist.

NOTE

To enable faster convergence with ECMP routes, and ensure that a BGP RR sends multiple paths to the clients, the BGP add path feature must be configured in IPv4 or IPv6 address-family unicast configuration mode to advertise additional ECMP paths to the clients. To meet all IP Fabrics requirements when using iBGP for an IP Fabric, spine switches must support BGP RR and BGP add path. The entire IP Fabric can then be managed as a single ASN.

The following figure shows the iBGP underlay configuration for an IP Fabric. The entire IP Fabric is managed as a single ASN.

FIGURE 11 iBGP underlay configuration in an IP Fabric



BGP add path

The BGP add path feature provides the ability for multiple paths for the same prefix to be advertised without the new paths implicitly replacing the previous paths. Path diversity is achieved rather than path hiding.

When iBGP is used for BGP EVPN in an IP Fabric, spine switches are configured as route reflectors (RRs) so that a switch is not required to peer with every other device in the IP Fabric. Leaf switches are configured as RR clients.

BGP add path is supported for the BGP IPv4 and IPv6 unicast address families. It is not supported for the BGP L2VPN EVPN address family. BGP add path is not required for the L2VPN EVPN address family as a unique RD at each RBridge ensures that the same prefixes from multiple sources are not suppressed.

For more information on BGP add path, refer to the "BGP4" and "BGP4+" chapters in the *Extreme Network OS Layer 3 Routing Configuration Guide*.

BGP EVPN NLRI

BGP EVPN distributes Network Layer Reachability Information (NLRI) for a network. BGP EVPN NLRI includes both Layer 2 and Layer 3 reachability information for end hosts residing in the network, and advertises both the MAC and IP addresses of the EVPN VXLAN end hosts.

The following table shows BGP EVPN support for NLRI.

TABLE 1 BGP EVPN support for NLRI

| Support for NLRI | Description |
|---------------------------------|---|
| Auto-discovery (AD) per ES | An ESI can be associated with more than one EVPN instance. When an interface with an associated ESI and EVPN instance comes online, an auto-discovery (AD) route per ES route is originated and installed in the corresponding EVPN instance. The ESI can be distinguished locally through the IP address of the peer. |
| MAC and MAC IP | The MAC and MAC-IP addresses of the EVPN VXLAN end hosts are advertised to peers, and the distribution of these addresses through BGP EVPN reduces unknown unicast flooding in the VXLAN. |
| Ethernet Tag Routes | Multicast Ethernet Tag routes advertise VLAN membership to peers, indicating the type of multicast tunnel on which flooded traffic can be received for a VLAN. |
| ES-Routes | Multihoming support is provided through the advertisement of BGP Ethernet Segment Routes (ES-Routes). An Ethernet Segment (ES) is the set of links connected to the same multihomed host. An Ethernet Segment Identifier (ESI) is associated with each Ethernet segment and advertised in the Ethernet Segment Route (ES-Route). For BGP EVPN, BGP initiates an ES-Route in the EVPN routing table and advertises it to EVPN neighbors, thus distributing NLRI for the network. |
| Inclusive Multicast Route (IMR) | BGP EVPN uses this BGP Type 0x3 Route Type to advertise multicast labels for multicast tunnel end-point discovery. |
| Prefix Routes | IPv4 and IPv6 prefix routes belonging to tenants (VRFs) are exchanged providing inter-subnet connectivity within the data center. |

MAC address learning

Data plane (Layer 2) MAC address learning is enabled by default at the end points of the statically configured VXLAN tunnel..

Where BGP routing is used, MAC learning can be enabled by means of the **mac-learning protocol bgp** command. This delegates the responsibility for MAC learning to the BGP control-plane protocol. When VXLAN tunnel auto-discovery is enabled in BGP EVPN, default MAC learning is done by means of BGP.

For more information on MAC address learning, refer to [Configuring MAC learning of Layer 2 extension site through BGP](#) on page 78, as well as to the **mac-learning protocol bgp** command in the *Command Reference*.

EVPN instances

An EVPN instance (EVI) is an EVPN routing and forwarding instance spanning across the leaf nodes participating in that EVPN. EVIs are created using the **evpn-instance** command. Each EVI is identified by this configured name and is assigned an EVPN instance ID by the device.

Each EVI has a unique route distinguisher (RD) and one or more route targets (RT). RTs control the routes to be imported into and exported from the EVPN instance. An EVPN Routing table, containing information about the various routes associated with the EVI, is maintained for each EVI instance.

NOTE

Only one EVPN instance (EVI) is currently supported.

EVI can be configured with the following features:

- Route distinguisher (RD): Unique route distinguishers are assigned for an EVI. This value is automatically derived globally using the **rd auto** command so that each EVI has an associated RD that is unique across the entire fabric. RDs can also be manually configured for a specific virtual network identifier (VNI) under an EVI using the **rd (VNI)** command.
- Route targets: The Route Target (RT) extended community attribute, enabling logical grouping of EVPN routes that can be imported or exported, can be specified for an EVI. These route targets can be globally derived automatically using the **route-target (EVPN)** command. The **ignore-as** option for the **route-target (EVPN)** command should be used in instances where the

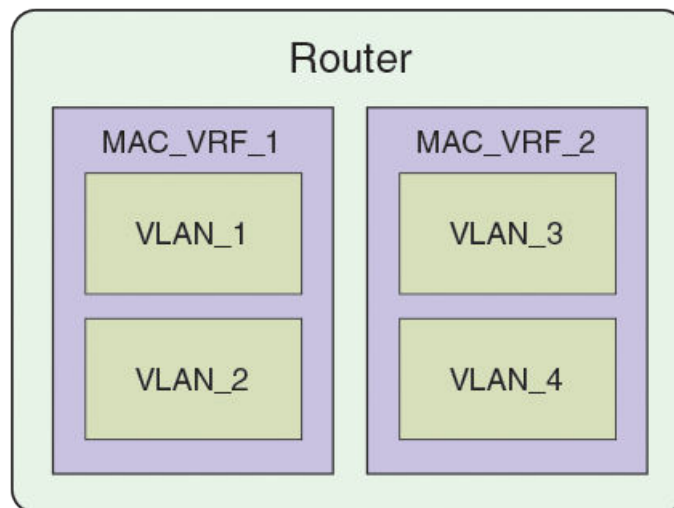
leaf nodes are under different autonomous systems and you want to import the routes from one leaf node to another. Route targets can also be manually configured for a specific virtual network identifier (VNI) under an EVI using the **route-target (VNI)** command. BGP EVPN uses these route targets to control the advertisement of EVPN routes.

- Virtual network identifiers (VNIs): VNIs can be added and removed for an EVI using the **vni add** and **vni remove** commands.
- Duplicate MAC detection timer: When the same host MAC address is learned by two different devices, continuous MAC moves are triggered by the traffic originating from these hosts. A duplicate MAC detection timer can be configured, using the **duplicate-mac-timer** command, to detect these continuous MAC moves. This timer specifies both a time interval and the maximum threshold of MAC moves that can occur within the configured time interval before the MAC address is treated as a duplicate address and further processing of updates for that MAC address are blocked.

For more information on the EVPN commands, refer to the *Extreme Network OS Command Reference*.

The following figure shows an RBridge with two configured EVIs, MAC_VRF_1 and MAC_VRF_2. Each EVI has a separate MAC-to-VLAN table.

FIGURE 12 RBridge with two configured EVIs



NOTE

Only one EVPN instance (EVI) is currently supported.

BGP L2VPN EVPN address family

The BGP L2VPN EVPN address-family configuration level provides access to commands that allow you to configure BGP EVPN. The BGP L2VPN EVPN address family uses components of BGP that are independent of the IPv4 and IPv6 unicast address families. Both iBGP and eBGP are supported.

The L2VPN EVPN address family supports the EVPN Subsequent Address Family Identifier (SAFI), an address qualifier that provides additional information about the Network Layer Reachability Information (NLRI) type for a given attribute.

The commands that you enter at this level apply only to the BGP L2VPN EVPN address family. BGP L2VPN EVPN address family capability can be negotiated along with the IPv4 or IPv6 address family. A separate BGP session is not required for the BGP EVPN address family.

To negotiate only the BGP L2VPN EVPN address family on an IPv4 neighbor, you must explicitly deactivate the IPv4 unicast address family using the **no neighbor activate** command.

To negotiate only the BGP L2VPN EVPN address family on an IPv6 neighbor, you must activate the neighbor only under the BGP L2VPN EVPN address family.

The following configuration options are allowed under BGP address-family L2VPN EVPN configuration mode:

```
device(config-bgp-evpn)# ?
```

Possible completions:

| | |
|-----------------------------|---|
| client-to-client-reflection | Configure client to client route reflection |
| graceful-restart | Enables the BGP graceful restart capability |
| neighbor | Specify a neighbor router |
| retain | Retain route targets |
| vtep-discovery | Enable VTEP discovery |

The following neighbor configuration options are allowed under BGP address-family L2VPN EVPN configuration mode:

```
device(config-bgp-evpn)# neighbor 10.1.1.1 ?
```

Possible completions:

| | |
|------------------------|--|
| activate | Allow exchange of route in the current family mode |
| allowas-in | Disables the AS_PATH check of the routes learned from the AS |
| enable-peer-as-check | Disable routes advertise between peers in same AS |
| maximum-prefix | |
| next-hop-unchanged | Next hop unchanged |
| route-map | Apply route map |
| route-reflector-client | Configure a neighbor as Route Reflector client |
| send-community | Send community attribute to this neighbor |

Automatic VXLAN tunnel endpoint discovery

VXLAN tunnel endpoint (VTEP) IP addresses are carried in every BGP EVPN route so that the leaf device receiving the BGP EVPN updates triggers VXLAN tunnel creation using this remote VTEP IP address. Remote VTEP discovery is enabled by default for BGP EVPN when the BGP L2VPN EVPN address family is enabled.

BGP devices can determine which VLANs are common between two devices and extend those VLANs over the VXLAN tunnel between the two devices. VTEP IP address is carried in BGP EVPN updates in next-hop network address field of the MP_REACH_NLRI attributes. BGP deletes VXLAN tunnels associated with remote VTEP when all of the routes from remote VTEP are withdrawn. If you want to configure a tunnel explicitly (statically) or in an NSX Controller deployment, the BGP Automatic VTEP feature should be disabled by means of the **no vtep-discovery** command.

BGP next-hop-unchanged

The BGP next-hop-unchanged feature is supported for the L2VPN EVPN address family, allowing BGP to send updates to eBGP peers with the next hop in the MP_REACH_NLRI attribute unchanged.

By default, eBGP BGP speakers change the next hop while sending the updates to eBGP neighbors. The BGP next-hop-unchanged feature overrides this behavior so that the next-hop address remains unchanged while updates are sent to eBGP peers. The BGP speaker is forced to retain the next hop address in the BGP updates received from neighbors. BGP next-hop-unchanged should be configured on the spine switch for each EVPN neighbor if the leaf switch is an eBGP peer. BGP next-hop-unchanged should be configured on the leaf switch for each EVPN neighbor if the spine switch is an eBGP peer. This means that eBGP BGP speakers will not change the next hop while sending updates to eBGP neighbors in an IP Fabric. Therefore, by configuring BGP next-hop-unchanged under the BGP L2VPN EVPN address family, changes to the next hop address in BGP EVPN updates are prevented.

NOTE

BGP next-hop-unchanged should be configured on spine nodes for all types of BGP deployments.

BGP retain route target

The BGP retain route target feature allows a spine node to accept all route targets (RTs).

Because spine switches do not have EVPN instances (EVIs) and VRFs configured, BGP EVPN routes are filtered as otherwise none of the RTs in the routes match the local route-target import configuration. When spine switches are configured as RRs, or establish eBGP peering with the leaf nodes, the BGP retain route target feature should be enabled so that the Spine switches do not do route-target filtering. This means that all route targets are retained and the route-target attributes in the EVPN routes are not modified or removed.

BGP retain route target is supported for the BGP L2VPN EVPN address family only and is used in BGP EVPN configurations. It is not supported for the IPv4 and IPv6 unicast address families.

RIBout AS path check

The RIBout AS path check feature enables the outbound AS_PATH check function so that a BGP sender speaker does not send routes with an AS path that contains the ASN of the receiving speaker. If the remote AS of the neighbor appears in the AS path segment of the NLRI, the NLRI is not added to RIBout of the peer.

When the AS path check is enforced for the sender using the **neighbor enable-peer-as-check** command, it saves the amount of RIBout memory used to store routes that would otherwise be stored in sender's RIBout, advertised to the peer and be discarded by the receiver, if the **neighbor allows-in** command is not configured. Convergence time is also improved.

BGP legacy features supported for the L2VPN EVPN address family

The following BGP features, already supported for IPv4 and IPv6 unicast address families, are supported for the BGP L2VPN EVPN address family and used in BGP EVPN configurations:

- **BGP extended community:** The BGP extended community feature filters routes based on a regular expression specified when a route has multiple community values in it. The BGP extended community feature is supported for the L2VPN EVPN address

family for both spine and leaf nodes. When a spine or leaf node receives BGP EVPN routes from other spine nodes, it checks the route-target extended community attribute of the routes. If the route-target is the same as the import target of the given EVPN instance on the local leaf node, the leaf node adds the route to the EVPN routing table. If the spine node is configured with the BGP retain route target feature, this checking is bypassed in the spine. If a static MAC address is redistributed to BGP, it is advertised with the “Sticky MAC” extended community attribute. The next hop router MAC address for prefix routes is advertised as the default gateway extended community attribute. The same check is performed for prefix routes; however the comparison is made against the RT configured for VRFs rather than EVIs.

- BGP graceful restart: BGP graceful restart (GR) can be configured for the L2VPN EVPN address family. When GR capability is negotiated, neighboring devices participate in the restart, helping to ensure that no route and topology changes occur in the network for the duration of the restart. GR helps retain the peer routes when a restart occurs during HA failover. When the GR feature is enabled for the L2VPN EVPN address family, existing sessions are not affected and require neighbor reset to negotiate the GR capability.
- BGP peer groups: BGP peer groups can be configured for the L2VPN EVPN address family so that neighbors with the same attributes and parameters can be grouped together.
- BGP route reflection: The BGP route reflection feature is supported for the L2VPN EVPN address family. When iBGP is used for BGP EVPN in an IP Fabric, spine switches are configured as route reflectors (RRs) and leaf switches are configured as route reflector clients. You can configure a leaf switch as a route reflector client from the spine switch using the **neighbor route-reflector-client** command. Each leaf switch should be configured so that they all belong in the same cluster.
- Client-to-client-reflection: The BGP client-to-client reflection feature is supported for the L2VPN EVPN address family. For iBGP, if the leaf switches are configured as route reflector clients and are connected in a full iBGP mesh, you can disable client-to-client reflection on the spine switch which is configured as an RR. By default, in an IP Fabric, the clients of a route reflector are not required to be fully meshed and the routes from a client are reflected to other clients.
- Disabling the BGP AS_PATH check: A device can be configured so that the AS_PATH check function for routes learned from a specific location is disabled, and routes that contain the recipient BGP speaker’s AS number are not rejected. When eBGP is configured for BGP EVPN in an IP Fabric, the AS_PATH check function can be disabled on a leaf switch so that route updates from the spine layer are not discarded by the leaf.

Ethernet Segment Identifiers for BGP routing

An Ethernet Segment is a set of Ethernet links that connect a multihomed device to a BGP router. Ethernet Segments are assigned a unique identifier, referred to as an Ethernet Segment Identifier (ESI). ESIs can be configured on port-channel interfaces.

Ethernet links that connect single-homed devices are not required to have an ESI value associated with them. For BGP EVPN, each BGP device advertises an Ethernet Segment Route (ES-Route) informing other BGP devices that it is connected to that ES. The other BGP devices can then detect if they are connected to the same ES. The user can use the **esi** command in port-channel configuration mode to configure the port-channel to derive the ESI value automatically by means of Link Aggregation Control Protocol (LACP). ESI value can be manually also configured on the port channel. This is needed for static port channels where LACP is not involved.

BGP dynamic neighbors

The BGP dynamic neighbors feature allows peering to a group of remote neighbors defined by a listen range. BGP neighbors can be created without statically configuring them.

Routing protocols are designed to exchange routing information and distribute it to all neighbors. Each protocol has its own way of detecting neighbors and establishing adjacency with them. By design, all IGP neighbors are a single hop away. However, this is not true for BGP. BGP neighbors can be a single hop away or n hops away. This implies that BGP neighbors are not dynamically discovered. Rather, an administrator must enable and configure each neighbor for the exchange of routing information. The greater the number of neighbors, the greater the administrative overhead. With BGP dynamic neighbors, when a device sees an incoming TCP session initiated from a remote neighbor with an IP address in the configured subnet listen range, a new BGP neighbor is dynamically created. The device that initiates this connection still has the neighbor statically configured. Each listen range can be configured as a subnet IP range. BGP dynamic neighbors are configured using a range of IP addresses and BGP peer groups.

Whenever the incoming TCP session falls under the configured subnet listen range, a new BGP neighbor is dynamically created as a member of that peer group. Once the listen range is configured, and the associated peer group is activated, dynamic BGP neighbor creation does not require any further configuration on the initial device. Newly created BGP dynamic peers automatically inherit the attributes associated with the peer group attached to the listen range, such as inbound policy, outbound policy, and so on.

You can set the total number of dynamic neighbors that can be formed in the system. When the global or listen range limit is increased, any new connection coming from the remote end that falls under the configured range is accepted. If the limit has already been reached and you reduce the global or peer-group limit, already established dynamic neighbors are not destroyed. Once the limit has been reached and new connection requests are received, the connections are rejected and the information is logged.

BGP dynamic neighbors are deleted when a session moves from the established state to the idle state.

NOTE

BGP dynamic neighbors supports only IPv4 BGP peering.

Consider the following when using BGP dynamic neighbors:

- A neighbor can be associated with a peer group so that the peer group properties are inherited by each individual neighbor.
- Individual dynamic neighbors cannot be configured with a separate set of policies as BGP dynamic neighbors are created on demand.
- BFD for dynamic neighbors is supported. When Bidirectional Forwarding Detection (BFD) detects that a link has gone down, the dynamic neighbor moves back to the idle state and the dynamic neighbor is deleted.
- Static neighbors are always given precedence over BGP dynamic neighbors.
- The **auto-shutdown-new neighbors** command is not supported for BGP dynamic neighbors.
- BTSH and MD5 passwords are not supported for BGP dynamic neighbors.

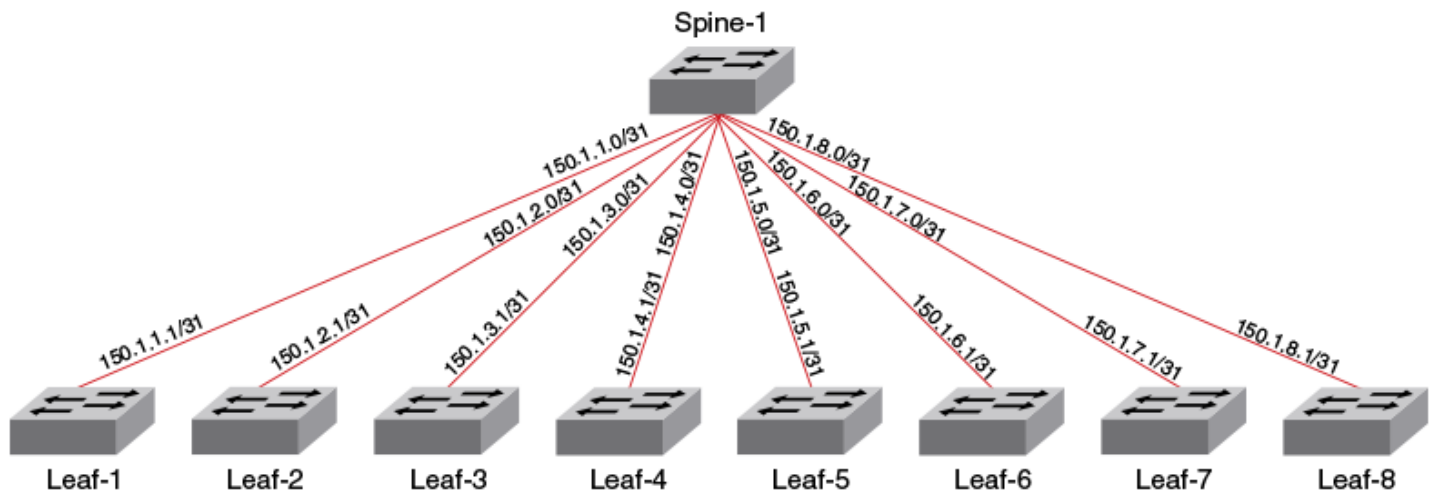
BGP dynamic neighbors for an IP Fabric

The BGP dynamic neighbors feature allows peering to a group of remote neighbors defined by a listen range. BGP neighbors can be created without statically configuring them.

In a typical data center deployment, a spine has a BGP session with each of the leaf nodes. Typically in this situation, the ratio of spine nodes to leaf nodes is 1 to 4. The greater the number of leafs, the greater the administrative overhead at the spine. Every time a new leaf node is added, specific configurations must be added for each spine node. Moreover, if the IP address used by a leaf for neighboring with the spine node changes, BGP peering information at all spines must be changed. Therefore, as the number of leaf node connections increases, a great deal of administrative work can be required.

The following figure illustrates a typical data center deployment where a spine has a BGP session with each of the individual leaf nodes.

FIGURE 13 BGP session between spine and leaf nodes



In order for the spine to establish sessions with each individual leaf node, as depicted in the previous figure, a great deal of configuration is needed at the spine level. The following configuration is required at the spine level to establish such a session with the leaf nodes.

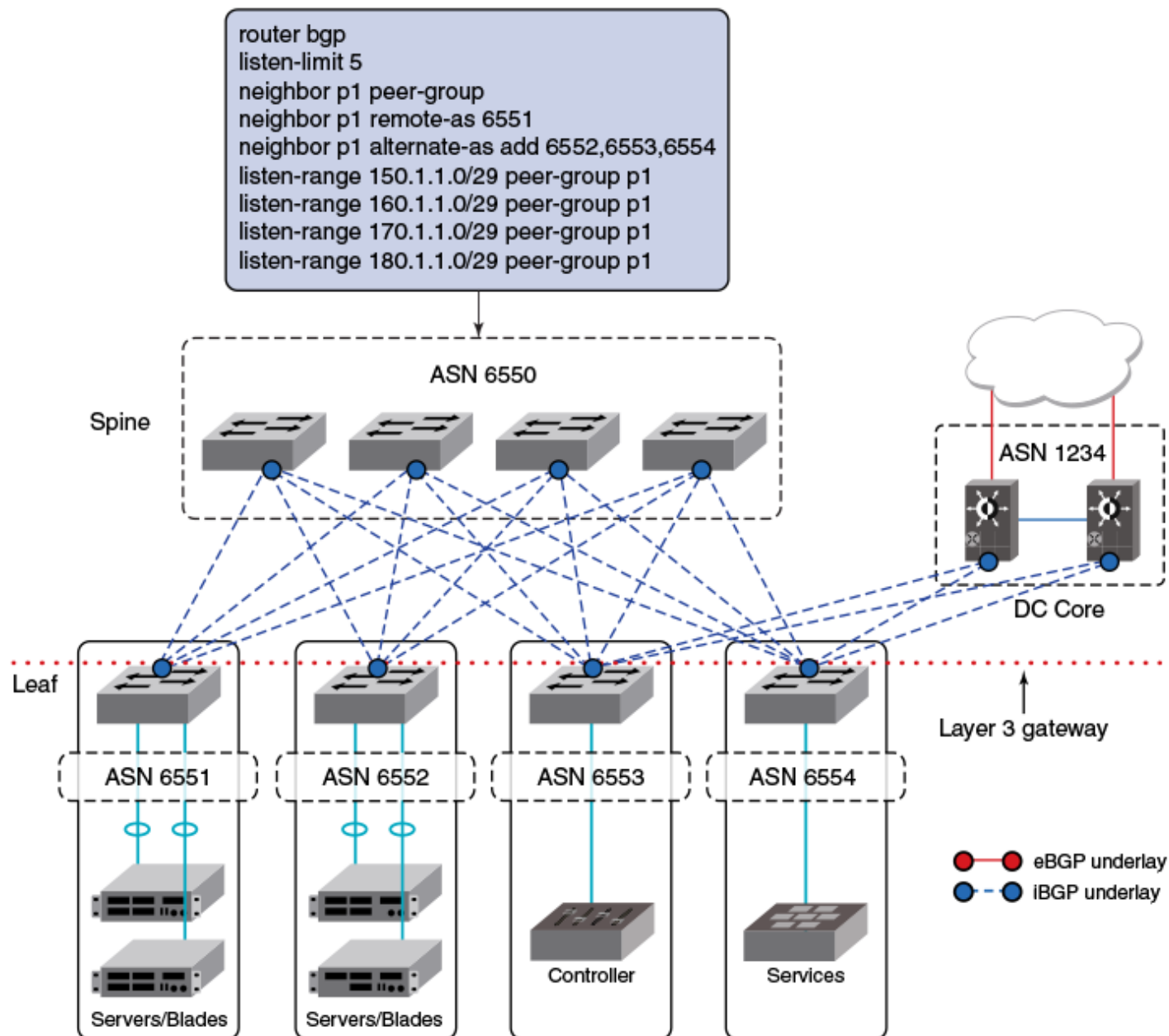
```
device# configure terminal
device(config)# rbridge-id 250
device(config-rbridge-id-250)# router bgp
device(config-bgp-router)# local-as 65500
device(config-bgp-router)# neighbor 150.1.1.1 remote-as 65550
device(config-bgp-router)# neighbor 150.1.2.1 remote-as 65550
device(config-bgp-router)# neighbor 150.1.3.1 remote-as 65550
device(config-bgp-router)# neighbor 150.1.4.1 remote-as 65550
device(config-bgp-router)# neighbor 150.1.5.1 remote-as 65550
device(config-bgp-router)# neighbor 150.1.6.1 remote-as 65550
device(config-bgp-router)# neighbor 150.1.7.1 remote-as 65550
device(config-bgp-router)# neighbor 150.1.8.1 remote-as 65550
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 150.1.1.1 activate
device(config-bgp-evpn)# neighbor 150.1.2.1 activate
device(config-bgp-evpn)# neighbor 150.1.3.1 activate
device(config-bgp-evpn)# neighbor 150.1.4.1 activate
device(config-bgp-evpn)# neighbor 150.1.5.1 activate
device(config-bgp-evpn)# neighbor 150.1.6.1 activate
device(config-bgp-evpn)# neighbor 150.1.7.1 activate
device(config-bgp-evpn)# neighbor 150.1.8.1 activate
```

With BGP dynamic neighbors, the previous configuration is reduced to the following configuration.

```
device# configure terminal
device(config)# rbridge-id 250
device(config-rbridge-id-250)# router bgp
device(config-bgp-router)# local-as 65500
device(config-bgp-router)# neighbor pgl peer-group
device(config-bgp-router)# neighbor pgl remote-as 65550
device(config-bgp-router)# listen-range 150.1.0.0/16 peer-group pgl limit 8
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor pgl activate
```

When BGP dynamic neighbors is enabled for a spine, a BGP subnet listen range (a subnet range address that the IP address of all connected leafs falls under) is configured. Leaf switches are added as BGP peers based on the listen range. The following figure illustrates the minimum configuration needed for a single spine to enable BGP dynamic neighbors.

FIGURE 14 Minimum configuration at the spine level for BGP dynamic neighbors



When eBGP peers have similar configurations, the remote AS number (ASN) cannot be configured as part of the peer group because each peer has its own AS number. For static eBGP peers, a remote ASN must first be configured and then attached to the peer group. For BGP dynamic neighbors, you can configure an alternate AS range; an AS range under which the eBGP peer can fall for the listen range the peer group is a part of. This alternate range is used only for BGP dynamic neighbors. The previous figure illustrates the configuration for the deployment of dynamic eBGP neighbors.

When an OPEN message is received from a particular peer with an AS number that falls under the range configured for the peer group, it is accepted and a session is formed.

For general information on BGP dynamic neighbors, refer to the *Extreme Network OS Layer 3 Routing Configuration Guide*.

Configuring BGP dynamic neighbors for an IP Fabric

The following task is configured on a spine and creates a peer group called "leaf-group" and associates it with the listen range of 150.1.0.0/16. The spine waits for the incoming TCP session initiated by a leaf node. When the TCP connection is accepted, the spine verifies whether the incoming TCP session falls under one of the configured listen ranges. If the verification is successful, a new BGP dynamic peer is created and associated with the peer group "leaf-group". This newly created BGP dynamic peer inherits the attributes associated with the peer group.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 122
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-122)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 65520
```

5. Enter the **neighbor peer-group** command, specifying a name for the peer group, to create a peer group.

```
device(config-bgp-router)# neighbor leaf-group peer-group
```

6. Enter the **listen-limit** command and specify a value to define the maximum number of BGP dynamic subnet range neighbors that can be created.

```
device(config-bgp-router)# listen-limit 30
```

7. Enter the **neighbor peer-group remote-as** command to specify an AS for the peer group.

```
device(config-bgp-router)# neighbor leaf-group remote-as 65550
```

8. Enter the **listen-range** command, specifying an IP address and mask, to associate a subnet range with the BGP peer group. Use the **limit** parameter and specify a value to set the maximum number of BGP dynamic neighbors that can be created.

```
device(config-bgp-router)# listen-range 150.1.0.0/16 peer-group leaf-group limit 20
```

9. Enter the **neighbor alternate-as** command with the **add** parameter, specifying a value, to set an alternate AS number for listen range neighbors in the configured peer group.

```
device(config-bgp-router)# neighbor leaf-group alternate-as add 1210
```

10. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

11. Enter the **neighbor peer-group activate** command to enable the exchange of information with the peer group.

```
device(config-bgp-evpn)# neighbor leaf-group activate
```

The following example creates a peer group called "leaf-group" and associates it with the listen range of 150.1.0.0/16. It also sets a limit on the number of dynamic BGP neighbors that can be created. An alternate AS number of listen range neighbors in the configured peer group is set.

```
device# configure terminal
device(config)# rbridge-id 122
device(config-rbridge-id-122)# router bgp
device(config-bgp-router)# local-as 65520
device(config-bgp-router)# neighbor leaf-group peer-group
device(config-bgp-router)# listen-limit 30
device(config-bgp-router)# neighbor leaf-group remote-as 65550
device(config-bgp-router)# listen-range 150.1.0.0/16 peer-group leaf-group limit 20
device(config-bgp-router)# neighbor leaf-group alternate-as add 1210
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor leaf-group activate
```

Multi-VRF for BGP EVPN

Multi-VRF must be configured in an IP Fabric to support asymmetric or symmetric routing. The link between the leaf and spine switches must be configured in the default VRF. For a nondefault VRF, traffic is routed at the leaf switch and is forwarded to a VXLAN tunnel.

For asymmetric routing, all VLANs are configured on every leaf switch and enabled for IP routing and forwarding. Traffic flows through different routes in different directions. This severely affects scaling with each switch carrying the MAC and MAC-IP routes for hosts that are not necessarily locally connected.

For symmetric routing, a VLAN is configured only at the leaf switch where a local host exists. A unique, common Layer 3 virtual network identifier (VNI) number is generated for all leaf nodes for the VRF instance using the **vni (VRF)** command. The ingress leaf switch performs a Layer 3 lookup and routes the packets towards the remote leaf switch over the common Layer 3 VNI. The inner MAC destination address (DA) of the packet is rewritten to that of the gateway MAC address advertised by the remote leaf switch and the packet is then encapsulated in a VXLAN tunnel and sent to the remote leaf switch. The remote leaf switch then terminates the VXLAN tunnel. Because the MAC DA of the inner packet is now that of the gateway MAC address advertised by the leaf switch, it performs Layer 3 lookup and the packet is routed to the locally attached host.

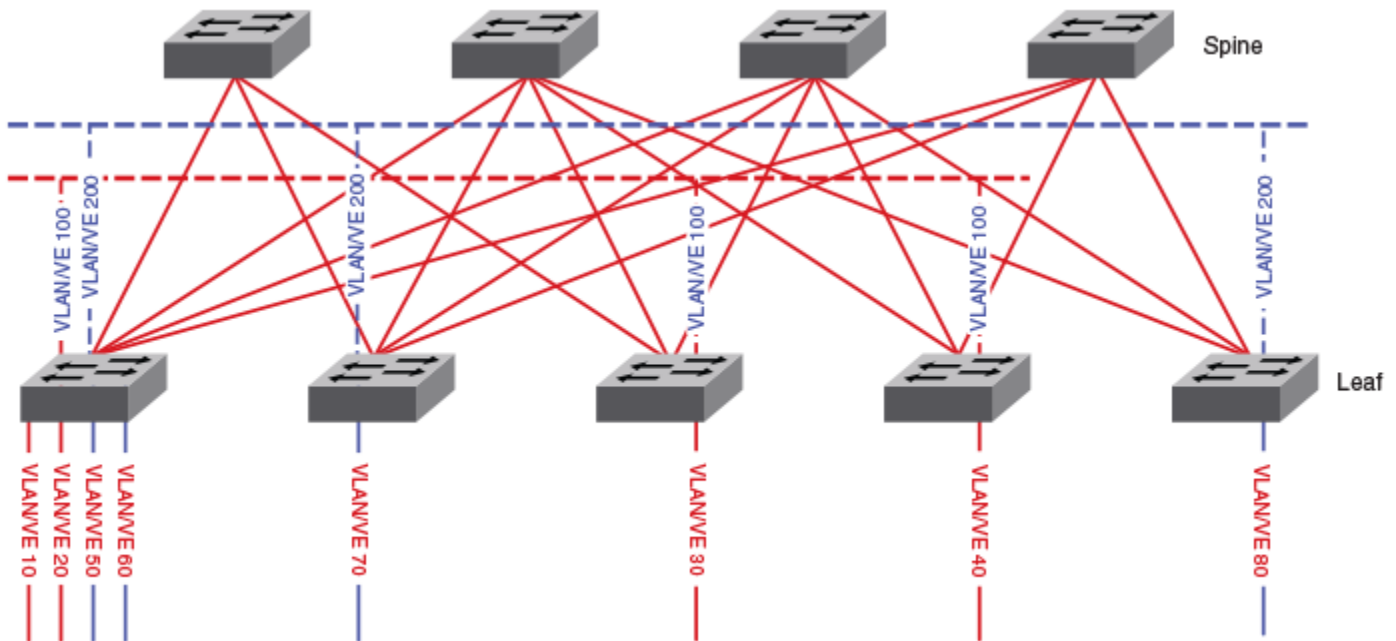
VRFs can be configured with the following features:

- Virtual network identifiers (VNIs): Layer 3 VNIs can be added and removed for a VRF instance using the **vni (VRF)** command. The VNI configuration under a VRF instance tells BGP to include the Layer 3 VNI and its associated MAC address in EVPN updates.
- Route distinguisher (RD): Unique route distinguishers are assigned for a VRF instance. RDs are manually configured for a VRF instance using the **rd (VRF)** command.
- Route targets: The route target (RT) extended community attribute, enabling logical grouping of EVPN routes that can be imported or exported, can be specified for a VRF instance. Route targets are manually configured for a specific VRF instance using the **route-target (VRF)** command. BGP EVPN uses these route targets to control the exchange of EVPN routes.
- Route filters: Filtering based on route-map policies can be added to the EVPN IPv4 and IPv6 prefix routes that are imported to or exported from VRFs. Refer to "BGP VRF route filters" in the "BGP4" and "BGP4+" chapters in this document.

For more information on the EVPN commands, refer to the *Extreme Network OS Command Reference*.

The following figure illustrates Multi-VRF topology using BGP EVPN. In this figure VRF red has VLAN 10, 20, 30, and 40. VLAN 100 is the Layer 3 VNI for VRF red. VRF blue has VLAN 50, 60, 70, and 80. VLAN 200 is the Layer 3 VNI for VRF blue.

FIGURE 15 Multi-VRF for BGP EVPN



The leaking of IPv4 and IPv6 prefix routes across VRFs is allowed using the BGP-EVPN control plane across leaf nodes. If you configure route-targets of the source VRF as import route-targets, routes originated at a leaf node in a given VRF can be imported into one or more VRFs by a node. Routes from a given VRF, including MAC IP (ARP) routes that were converted to /32 prefix routes and installed into the RIB, can be leaked into multiple VRFs. Refer to the **route-target (VRF)** command in the *Extreme Network OS Command Reference* for more information.

Static anycast gateway

This section discusses the static anycast gateway feature and illustrates example configurations.

Overview of static anycast gateway

Static anycast gateway enables you to configure identical IP/MAC gateway addresses to virtual Ethernet (VE) interfaces on leaf switches in an IP Fabric, increasing routing efficiency.

Static anycast gateway provides seamless virtual machine (VM) mobility across all of the leaf (ToR) switches. Even if hosts move among leaf switches, there is no need to reconfigure the default gateway. In this way, forwarding behavior is optimized.

Considerations and limitations for static anycast gateway

There are considerations and limitations that you need to be aware of when implementing static anycast gateway.

Static anycast gateway is supported only in an IP Fabric, and BGP EVPN must be enabled. In addition, when this feature is enabled in an IP Fabric, to prevent ARP/ND broadcast across the network, the user should enable ARP/ND suppression on the respective VLANs.

Static anycast gateway is not supported along with Fabric-Virtual-Gateway (FVG).

The following tables summarize support for VRRP and VRRP-E with static anycast gateway on the supported platforms.

TABLE 2 Support for static anycast gateway with VRRP and VRRP-E (VDX 6740 series)

| Static anycast gateway MAC address | VRRP (IPv4 or IPv6) | VRRP-E (IPv4 or IPv6) |
|------------------------------------|---------------------|-----------------------|
| Default | No | Yes |
| Nondefault | No | No |

TABLE 3 Support for static anycast gateway with VRRP and VRRP-E (VDX 6940 series)

| Static anycast gateway MAC address | VRRP (IPv4 or IPv6) | VRRP-E (IPv4 or IPv6) |
|------------------------------------|---------------------|-----------------------|
| Default | Yes | Yes |
| Nondefault | Yes | No |

Configuring MAC static anycast gateway addresses

To implement static anycast gateway, you need to specify gateway MAC addresses for IPv4 and IPv6 traffic on each RBridge.

NOTE

Depending on your needs, you can specify a MAC address for IPv4 traffic, for IPv6 traffic, or for both.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command to access RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. To specify a static anycast gateway MAC address for IPv4 traffic, enter the **ip anycast-gateway-mac** command, with one of the following options:

- To specify the default static anycast gateway MAC address for IPv4 traffic, enter the following command:

```
device(config-rbridge-id-1)# ip anycast-gateway-mac default-mac
```

NOTE

"Default" means that an automatically generated MAC address is used.

- To specify a nondefault static anycast-gateway MAC address for IPv4 traffic, enter a command such as the following:

```
device(config-rbridge-id-1)# ip anycast-gateway-mac 2222.2244.4444
```

- To specify a static anycast gateway MAC address for IPv6 traffic, enter the **ipv6 anycast-gateway-mac** command, with one of the following options:

- To specify the default static anycast gateway MAC address for IPv6 traffic, enter the following command:

```
device(config-rbridge-id-1)# ipv6 anycast-gateway-mac default-mac
```

- To specify a nondefault static anycast gateway MAC address for IPv6 traffic, enter a command such as the following:

```
device(config-rbridge-id-1)# ipv6 anycast-gateway-mac 2222.2266.6666
```

NOTE

The first three bytes (six digits) of a nondefault IPv4 static anycast gateway MAC address must be identical with the first three bytes of a corresponding IPv6 static anycast gateway MAC address.

Configuring IP static anycast gateway addresses

To implement static anycast gateway, you must specify IPv4 and IPv6 static anycast gateway addresses on a virtual Ethernet (VE) interface.

NOTE

Depending on your needs, you can specify an IPv4 address, an IPv6 address, or both.

- Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

- Enter the **rbridge-id** command to access RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

- Enter the **interface ve** command to access VE (RBridge) interface configuration mode.

```
device(config-rbridge-id-1)# interface ve 10
```

- To specify an IPv4 static anycast gateway address, enter the **ip anycast-address** command.

```
device(config-rbridge-ve-1)# ip anycast-address 2.2.2.2/24
```

- To specify an IPv6 static anycast gateway address, enter the **ipv6 anycast-address** command.

```
device(config-rbridge-ve-1)# ipv6 anycast-address 1234:10::10:100/64
```

Show commands for static anycast gateway

Use the following **show** commands to verify configurations of static anycast gateway.

TABLE 4 Show commands for static anycast gateway

| Command | Description |
|----------------------------------|---|
| show ip anycast-gateway | Displays IPv4 static anycast gateway details for all virtual Ethernet (VE) interfaces or for a specified VE interface. You can also filter by RBridge and by VRF. |
| show ipv6 anycast-gateway | Displays IPv6 static anycast gateway details for all VE interfaces or for a specified VE interface. You can also filter by RBridge and by VRF. |

ARP and ND scaling enhancements

This section discusses Address Resolution Protocol (ARP) and Neighbor Discovery (ND) features and illustrates example configurations.

ARP and Neighbor Discovery

When forwarding traffic, a device needs to know the destination's MAC address, because each IP packet is encapsulated in an ethernet frame. The MAC address is needed not only for the packet's final destination but also for a next hop towards the destination.

The technology by which a device gets the MAC address varies between IPv4 and IPv6, as follows:

- IPv4: Address Resolution Protocol (ARP)
- IPv6: Neighbor Discovery (ND)

IPv4 traffic

When the destination's IP address is known, to get the MAC address, a device first searches its ARP cache. A match for the IP address supplies the corresponding MAC address. Otherwise, the device broadcasts an ARP request. The network devices receive such ARP requests, and the host with a matching IP address sends an ARP reply that includes its MAC address.

IPv6 traffic

When the destination's IPv6 address is known, to get the MAC address, a device first searches its neighbor cache. A match for the IPv6 address supplies the corresponding MAC address. Otherwise, the device broadcasts a neighbor solicitation (NS) request. The network devices receive the NS request, and the host with a matching IPv6 address sends a neighbor advertisement (NA) reply that includes its MAC address.

ARP and ND scaling enhancements

In many network configurations, ARP and Neighbor Discovery (ND) traffic and caching consume significant resources, which can be optimized.

ARP and ND suppression

In a data center fabric, the ARP and ND suppression features can help reduce ARP and ND control traffic.

The default scenario (disablement of ARP and ND suppression) can lead to excess control traffic:

1. A device needs to forward traffic to an IP address, but the corresponding MAC address is not in its ARP or ND cache.
2. The device broadcasts an ARP or ND request throughout the IP Fabric.

When ARP and ND suppression are enabled, excess control traffic is reduced:

1. A device needs to forward traffic to an IP address, but the corresponding MAC address is not in its ARP or ND cache.
2. The device broadcasts an ARP or ND request.
3. The leaf BGP EVPN control plane intercepts the request and looks for a match in its local cache.
 - If there is a match, the control plane responds to the device (rather than broadcasting the original request to the entire IP Fabric).
 - Only if there is no match, does the leaf control plane broadcast the request to the entire IP Fabric.

NOTE

When the static anycast gateway feature is enabled in an IP Fabric, to prevent ARP/ND broadcast across the network, the user should enable ARP/ND suppression on the respective VLANs.

Supported platforms

The ARP and ND suppression features are supported on the following platforms:

- ExtremeSwitching VDX 6740 series
- ExtremeSwitching VDX 6940 series
- ExtremeSwitching VDX 2741 (not supported for IP Fabrics)
- ExtremeSwitching VDX 2746 (not supported for IP Fabrics)

Enabling ARP and ND suppression on a VLAN

Use this procedure to enable and disable ARP and ND suppression on a VLAN.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface vlan** command to access VLAN configuration mode.

```
device(config)# interface vlan 110
```

3. To enable ARP suppression, enter **suppress-arp**.

```
device(config-Vlan-110)# suppress-arp
```

To disable ARP suppression, enter **no suppress-arp**.

4. To enable ND suppression, enter **suppress-nd**.

```
device(config-Vlan-110)# suppress-nd
```

To disable ND suppression, enter **no suppress-nd**.

The following example enables ARP suppression on VLAN 110.

```
device# configure terminal
device(config)# interface vlan 110
device(config-Vlan-110)# suppress-arp
```

Enabling and disabling ARP learning

Use this procedure to enable ARP learning not only locally, but from all ARP requests. ARP learning decreases the time needed to populate the ARP cache.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command to access RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```


3. Enter the **interface ve** command to access VE configuration mode.

```
device(config-rbridge-id-1)# interface ve 110
```

4. To enable fabric learning, enter the **ip arp learn-any** command.

```
device(config-ve-110)# ip arp learn-any
```

To disable fabric learning, enter the **no ip arp learn-any** command.

ARP and ND suppression show and clear commands

There is a full range of ARP and ND suppression **show** and **clear** commands, listed here with descriptions.

TABLE 5 ARP and ND suppression show commands in the *Command Reference*

| Command | Description |
|--|--|
| show ip arp suppression-cache | Displays IPv4 ARP suppression information. |
| show ip arp suppression-statistics | Displays IPv4 ARP suppression statistics. |
| show ip arp suppression-status | Displays the IPv4 ARP suppression status. |
| show ipv6 nd suppression-cache | Displays IPv6 ND suppression information. |
| show ipv6 nd suppression-statistics | Displays IPv6 ND suppression statistics. |
| show ipv6 nd suppression-status | Displays the IPv6 ND suppression status. |

TABLE 6 ARP and ND suppression clear commands in the *Command Reference*

| Command | Description |
|---|--|
| clear ip arp suppression-cache | Clears the IPv4 ARP suppression cache. You can also clear the cache for a specified VLAN. |
| clear ip arp suppression-statistics | Clears the IPv4 ARP suppression statistical information. You can also clear statistics for a specified VLAN. |
| clear ipv6 nd suppression-cache | Clears the IPv6 ND suppression cache. You can also clear the cache for a specified VLAN. |
| clear ipv6 nd suppression-statistics | Clears the IPv6 ND suppression statistical information. You can also clear statistics for a specified VLAN. |

Conversational ARP and ND

Conversational ARP and ND reduce the number of cached ARP and ND entries by programming only active flows into the forwarding plane. This feature helps to optimize utilization of hardware resources.

In many use-case scenarios—especially in an IP Fabric—there are software requirements for ARP and ND entries beyond the hardware capacity. Conversational ARP and ND limit storage-in-hardware to active ARP and ND entries; aged-out entries are deleted automatically.

By default, the aging-out threshold is 300 seconds. (You can change the threshold to any integer value from 60 through 100,000 seconds, either before or during enablement.) Any entry that does not have at least one conversation before aging-out is deleted from the cache. Each conversation restarts the clock for that entry.

However, aging-out is also influenced by the enablement and disablement cycle:

1. When you enable conversational ARP and ND, a fast-aging policy of 30 seconds (not configurable) applies to all entries in the ARP and ND caches at that time.

2. From enablement of conversational ARP and ND, the aging-time value applies for existing entries with new traffic and for new entries.
3. Upon disablement, the conversational ARP and ND timers no longer apply. All current entries become permanent as do all new entries.

The following entries are not subject to conversational behavior:

- Static ARPs and NDs
- Next hops

Supported platforms

Conversational ARP and ND are supported on the following platforms:

- ExtremeSwitching VDX 6740 series
- ExtremeSwitching VDX 6940 series
- ExtremeSwitching VDX 2741 (not supported for IP Fabrics)
- ExtremeSwitching VDX 2746 (not supported for IP Fabrics)

Enabling and disabling conversational ARP and ND

Conversational ARP and ND can reduce the number of cached ARP and ND entries, optimizing utilization of hardware resources.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command to access RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. To specify an aging-time value other than the default 300 seconds, enter the **host-table aging-time conversational** command.

```
device(config-rbridge-id-1)# host-table aging-time conversational 600
```

To restore the default aging-time value of 300 seconds, enter the **no host-table aging-time conversational** command.

4. To enable conversational ARP and ND, enter the **host-table aging-mode conversational** command.

```
device(config-rbridge-id-1)# host-table aging-mode conversational
```

To disable conversational ARP and ND, enter the **no host-table aging-mode conversational** command.

The following example implements conversational ARP and ND. The current aging-time value applies.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# host-table aging-mode conversational
```

Conversational ARP and ND show and clear commands

Conversational ARP and ND **show** and **clear** commands are listed here with descriptions.

TABLE 7 Conversational ARP and ND show commands in the *Command Reference*

| Command | Description |
|----------------------|-------------------------|
| show arp slot | Displays the ARP cache. |

TABLE 7 Conversational ARP and ND show commands in the *Command Reference* (continued)

| Command | Description |
|--------------------------------|-------------------------------------|
| show ipv6 neighbor slot | Displays IPv6 neighbor information. |

TABLE 8 Conversational ARP and ND clear commands in the *Command Reference*

| Command | Description |
|---------------------------------------|--|
| clear arp | Clears the local ARP cache and then resends ARP requests to the local hosts. |
| clear arp no-refresh | Clears the ARP cache, without resending ARP requests to the local hosts. |
| clear ipv6 neighbor | Clears the IPv6 ND cache and then resends ND requests to the local hosts. |
| clear ipv6 neighbor no-refresh | Clears the ND cache, without resending ND requests to the local hosts. |

Standards conformance and RFC support for BGP EVPN

The following table lists the IETF RFCs and other draft documents that support the implementation of BGP EVPN in support of Extreme IP Fabrics.

TABLE 9 IETF BGP EVPN RFCs and other draft documents supporting Extreme IP Fabrics

| Document | URL | Description |
|--|---|--|
| RFC 7432: "BGP MPLS-Based Ethernet VPN" | http://tools.ietf.org/html/rfc7432 | Describes procedures for BGP MPLS-based Ethernet VPNs (EVPN). |
| "A Network Virtualization Overlay Solution using EVPN" | https://tools.ietf.org/html/draft-ietf-bess-evpn-overlay-01 | Describes how Ethernet VPN (EVPN) can be used as an Network Virtualization Overlay (NVO) solution and explores the various tunnel encapsulation options over IP and their impact on the EVPN control-plane and procedures. |
| "Integrated Routing and Bridging in EVPN" | https://tools.ietf.org/html/draft-ietf-bess-evpn-inter-subnet-forwarding-00 | Describes an extensible and flexible multi-homing VPN solution for intra-subnet connectivity among hosts/VMs over an MPLS/IP network. |
| "IP Prefix Advertisement in EVPN" | https://tools.ietf.org/html/draft-ietf-bess-evpn-prefix-advertisement-02 | Defines a new EVPN route type for the advertisement of IP prefixes and explains some use-case examples where this new route- type is used. |

Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay

- Configuration overview.....53
- Configuring the leaf switches.....55
- Configuring the spine switches64
- (Optional) Configuring a BGP EVPN instance68
- (Optional) Configuring support for dual-homed servers.....70
- Configuring a superspine switch.....71
- Configuring interoperability with other vendors.....74
- Verifying the configuration.....75
- Using traceroute for overlay tunnels.....78
- Configuring additional features for IP Fabrics.....78

Configuration overview

This chapter provides the steps to set up a basic Extreme IP Fabric topology at the spine and leaf tiers, as well as to configure optional multihomed servers.

NOTE

The following is not meant to represent a production deployment scenario, with a full variety of deployment options. It is meant to facilitate the understanding of the basic steps that are required.

NOTE

Extreme offers Extreme Workflow Composer, an open and programmable turnkey automation solution that enables organizations to:

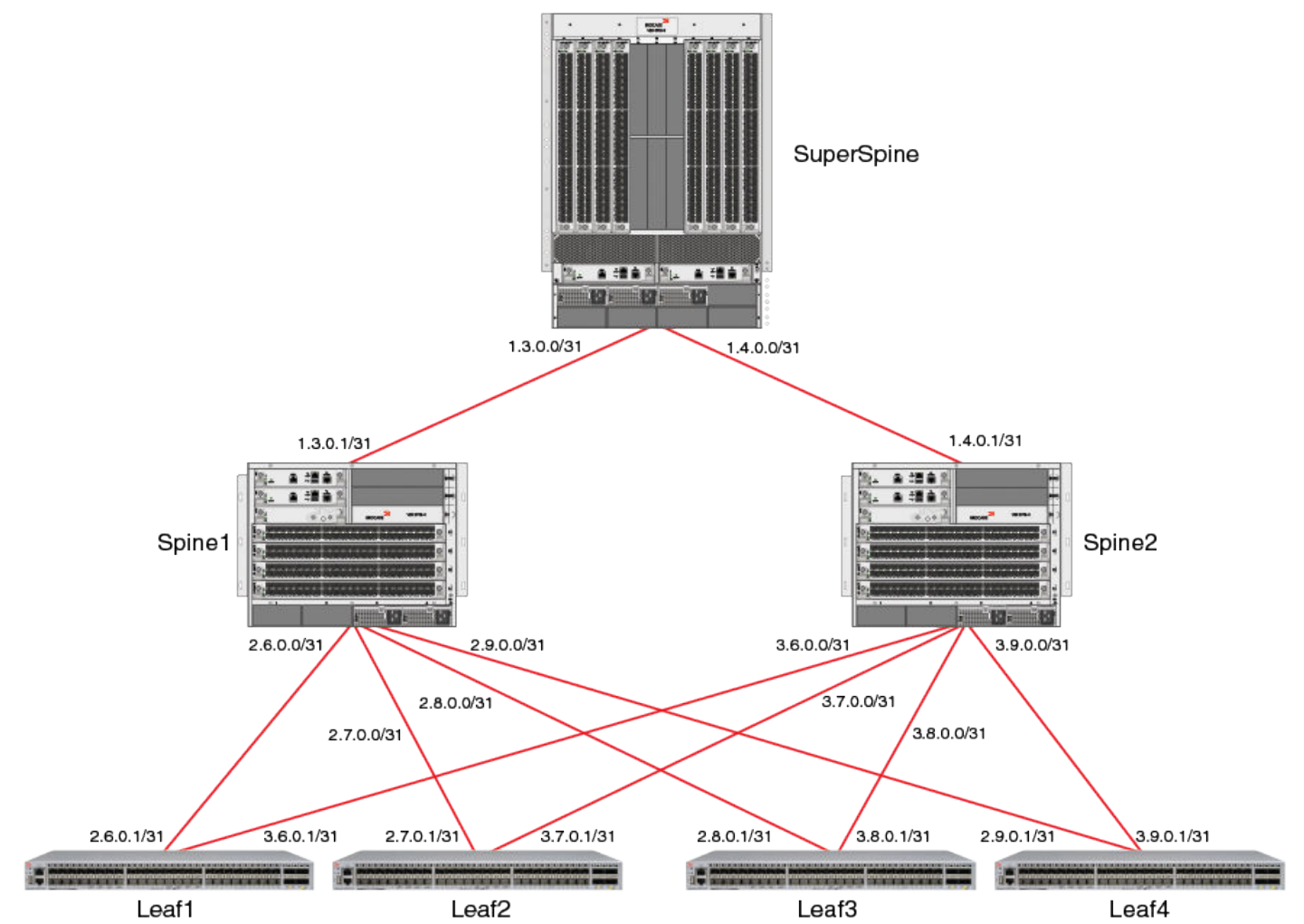
- Easily provision and validate IP Fabric infrastructure
- Customize automation to optimize IT operations

Extreme Workflow Composer’s pre-packaged, Python-based automation modules can provision and validate Extreme IP Fabrics and BGP-EVPN with minimal effort, while support for open interfaces and commonly used infrastructure programming tools enables simple and straightforward customization when needed.

For more information, refer to the *Extreme Workflow Composer User Guide*.

The following illustrates the topology and subnets of the nodes used in this example configuration.

FIGURE 16 IP Fabrics topology



NOTE
Although this configuration example uses BGP EVPN to implement the overlay network, BGP EVPN is not essential to the configuration of an IP Fabric.

Refer to the topology in [Multihoming](#) on page 17 for the basic design. The example topology consists of a pair of switches at the spine tier, four switches at the leaf tier, and an optional superspine switch. The following table summarizes the basic configurations at each tier for an example switch.

TABLE 10 Basic configurations at leaf, spine, and optional superspine tiers

| Leaf | Spine | Superspine |
|------------------------|---------------------|---------------------|
| VLANs | | |
| Static anycast gateway | | |
| Ethernet interfaces | Ethernet interfaces | Ethernet interfaces |
| Loopback interfaces | Loopback interfaces | Loopback interfaces |

TABLE 10 Basic configurations at leaf, spine, and optional superspine tiers (continued)

| Leaf | Spine | Superspine |
|--|--|--|
| (Optional) Port-channel interface with vLAG and Ethernet Segment Identifier to support dual-homed servers | | |
| Access/trunk ports | | |
| Routed, IP port-channel, or optional unnumbered IP interfaces | Routed, IP port-channel, or optional unnumbered IP interfaces | Routed, IP port-channel, or optional unnumbered IP interfaces |
| VRF instances (route distinguisher, VXLAN Network Identifier, route) | | |
| Address family (IPv4, IPv6, EVPN): route target | Address family (IPv4, IPv6, EVPN) | Address family (IPv4, IPv6, EVPN) |
| Virtual Ethernet interfaces (VRF forwarding, anycast address) | | |
| (Optional) EVPN instance (add VLANs) | | |
| Overlay gateway (Layer 2 extension, loopback address, add RBridge, map VLANs to VNIs) | | |
| (Optional) Configure interoperability with other vendors where Extreme extended VLANs must be remapped manually to their respective VNIs | | |
| | | (Optional) OSPF routing |
| Configure BGP and IGP (for iBGP-based networks) on all switches, to distribute Layer 3 reachability information; if eBGP is used, configure the neighbor allowas-in command, to reduce load on hardware processing; optionally enable Bidirectional Forwarding Detection (BFD) for BGP neighbors, for fast recovery in case of link failure | Configure BGP and IGP (for iBGP-based networks) on all switches, to distribute Layer 3 reachability information; optionally enable Bidirectional Forwarding Detection (BFD) for BGP neighbors, for fast recovery in case of link failure | Configure BGP and IGP (for iBGP-based networks) on all switches, to distribute Layer 3 reachability information; optionally enable Bidirectional Forwarding Detection (BFD) for BGP neighbors, for fast recovery in case of link failure |
| | BGP neighbor (remote AS, prevent route rejection, load balancing, recursive next-hop lookups, redistribute directly connected routes) | BGP neighbor (local AS, load balancing, recursive next-hop lookups, redistribute directly connected routes) |

Configuring the leaf switches

The following tasks configure the leaf switches.

Creating the VLANs

This task establishes the required VLANs, to support both VRFs and VXLAN Network Identifiers (VNIs).

1. In global configuration mode, create the VLANs required to support service, starting with leaf switch Leaf1.

```
Leaf1# configure terminal
Leaf1(config)# interface vlan 101-108,120,121-128,140,141-148,160,161-168,180
```

NOTE

VLANs 120, 140, 160, and 180 will support VRFs as VXLAN Network Identifiers (VNIs). VLANs must be created before they can be added to a switchport trunk.

2. Repeat Step 1 as appropriate for the remaining leaf switches.

3. Use the **show running-config interface vlan** command to verify the configuration.
Refer to [Verifying the configuration](#) on page 75 for examples of useful **show** commands.

Configuring ARP and ND suppression

ARP suppression is required to support static anycast gateway. This task configures both ARP and Neighbor Discovery (ND) suppression, which is done in VLAN configuration mode.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Specify an existing VLAN and enter VLAN configuration mode. (This example starts with VLAN 101.)

```
device(config)# interface vlan 101
```

3. To enable ARP suppression, enter the **suppress-arp** command.

```
device(config-Vlan-101)# suppress-arp
```

4. To enable ND suppression, enter the **suppress-nd** command.

```
device(config-Vlan-101)# suppress-nd
```

5. Repeat Step 2 through Step 4, as appropriate, for all remaining VLANs on which ARP or ND suppression is required.

Configuring static anycast gateway MAC addresses

This task configures static anycast gateway MAC addresses for IPv4 and IPv6 support.

It is recommended that you do not use this feature unless BGP EVPN is implemented. It is also recommended that ARP/ND suppression be enabled.

Within a dual-stack IP Fabric, the first three bytes (six digits) of a nondefault IPv4 static-anycast-gateway MAC address must be identical with those of the corresponding IPv6 MAC address. IPv4 and IPv6 addresses can also be used, as well as default MAC addresses. For more information about the static anycast gateway feature, refer to [Static anycast gateway](#) on page 44.

1. In RBridge ID configuration mode, specify IPv4 and IPv6 static anycast gateway MAC addresses, by using the **ip anycast-gateway-mac** and **ipv6 anycast-gateway-mac** commands, respectively.

```
Leaf1(config)# rbridge-id 1
Leaf1(config-rbridge-id-1)# ip anycast-gateway-mac 0000.abba.baba
Leaf1(config-rbridge-id-1)# ipv6 anycast-gateway-mac 0000.abba.abba
```

2. Repeat Step 1 as appropriate for the remaining leaf switches. The same MAC address must be used in all cases.
3. To verify the configuration, use the **show running-config rbridge-id ip anycast-gateway-mac** and the **show running-config rbridge-id ipv6 anycast-gateway-mac** commands. Refer to [Verifying the configuration](#) on page 75 for examples of useful **show** commands.

Configuring the interfaces

This task configures the physical interfaces, as well as the virtual (loopback) interfaces configured with a donor IP address in the optional IP Unnumbered address configuration. It also references an optional task where servers are dual-homed to a leaf node for high availability, as well as an optional task to configure the maximum transmission unit (MTU) to account for encapsulation overhead.

1. In interface subtype configuration mode, configure an Ethernet interface for point-to-point connectivity between leaf switch Leaf1 and the spine switches.

This example uses both IPv4 and IPv6 addresses, with /31 and /127 networks, respectively.

```
Leaf1(config-rbridge-id-1)# interface tengigabitethernet 1/0/1
Leaf1(config-if-te-1/0/1)# ip address 2.6.0.1/31
Leaf1(config-if-te-1/0/1)# ipv6 address 1000:2:6::1/127
Leaf1(config-if-te-1/0/1)# no shut
```

Use the **description** keyword to facilitate management and maintenance of the network.

Repeat the preceding configuration for the second Ethernet interface.

```
interface tengigabitethernet 1/0/5
ip address 3.6.0.1/31
ipv6 address 1000:3:6::1/12
no shut
```

2. In RBridge ID configuration mode, configure a loopback interface.

The loopback interface provides the source of the donor addresses used for the IP unnumbered feature. This example uses both IPv4 and IPv6 addresses, with /32 and /128 networks, respectively.

```
Leaf1(config)# rbridge-id 1
Leaf1(config-rbridge-id-1)# interface loopback 1
Leaf1(config-loopback-1)# ip address 6.0.0.6/32
Leaf1(config-loopback-1)# ipv6 address 1000:6::6/128
Leaf1(config-loopback-1)# no shut
```

3. (Optional) You can configure unnumbered IPv4 and IPv6 addresses, with an appropriate loopback address, as follows.

```
Leaf1(config-rbridge-id-1)# interface tengigabitethernet 1/0/1
Leaf1(config-if-te-1/0/1)# ip unnumbered loopback 1
Leaf1(config-if-te-1/0/1)# ipv6 unnumbered loopback 1
Leaf1(config-if-te-1/0/1)# no shut
```

4. (Optional) To accommodate encapsulation overhead on a SLX 9140, you can use the **mtu** command to configure a nondefault MTU both globally and on an interface, with the interface configuration taking precedence.

- The following example configures a nondefault MTU globally.

```
device# configure terminal
device(config)# mtu 2000
```

- The following example configures a nondefault MTU on an Ethernet interface.

```
Leaf1(conf-if-eth-0/1)# mtu 2000
```

5. (Optional) To ensure that sufficient Time To Live (TTL) values are available for hello messages to establish separate neighbor adjacencies on leaf switches when eBGP is used in conjunction with the IP unnumbered feature, it is important to set the number of available eBGP paths to 2. Refer to [Configuring BGP routing](#) on page 59.

6. In interface subtype configuration mode, configure an Ethernet interface for switchport trunk connectivity between leaf switch Leaf1 and the spine switches.

These trunks will carry the ranged VLANs.

```
Leaf1(config-rbridge-id-1)# interface tengigabitethernet 1/0/16
Leaf1(config-if-te-1/0/16)# switchport
Leaf1(config-if-te-1/0/16)# switchport mode trunk
Leaf1(config-if-te-1/0/16)# switchport trunk allowed vlan add 101-108
Leaf1(config-if-te-1/0/16)# switchport trunk allowed vlan add 121-128
Leaf1(config-if-te-1/0/16)# no shut
```

Repeat the preceding configuration for the second Ethernet switchport trunk interface.

```
Leaf1(config-rbridge-id-1)# interface tengigabitethernet 1/0/17
Leaf1(config-if-te-1/0/16)# switchport
Leaf1(config-if-te-1/0/16)# switchport mode trunk
Leaf1(config-if-te-1/0/16)# switchport trunk allowed vlan add 141-148
Leaf1(config-if-te-1/0/16)# switchport trunk allowed vlan add 161-168
Leaf1(config-if-te-1/0/16)# no shut
```

7. Repeat Step 1 through Step 7, as appropriate, for the remaining leaf nodes.
8. (Optional) Where servers are dual-homed to a leaf node, refer to [\(Optional\) Configuring support for dual-homed servers](#) on page 70.
9. To verify the configuration, use the **show running-config interface** command.
Refer to [Verifying the configuration](#) on page 75 for examples of useful **show** commands.

Configuring the VRF instances

This task establishes the VRF instances and applies them to virtual Ethernet (VE) interfaces.

1. Create VRF instances, and configure a route distinguisher and specify the Layer 3 VNI to be used for intersubnet forwarding.
This task illustrates the implementation of BGP EVPN Layer 3 multitenancy where BGP EVPN is used for the overlay. BGP EVPN must be enabled to support VXLAN Network Identifiers (VNIs).

- a) Create a VRF instance, in this example "red".

```
Leaf1(config-rbridge-id-1)# vrf red
Leaf1(config-vrf-red)#
```

- b) Configure a route distinguisher (RD) and VNI.

```
Leaf1(config-vrf-red)# rd 6.0.0.6:1
Leaf1(config-vrf-red)# vni 120
```

2. Enter address-family IPv4 unicast configuration mode, configure import and export route targets, and specify EVPN routes as targets.

```
Leaf1(config-vrf-red)# address-family ipv4 unicast
Leaf1(config-vrf-red-ipv4-unicast)# route-target import 3:3 evpn
Leaf1(config-vrf-red-ipv4-unicast)# route-target export 3:3 evpn
Leaf1(config-vrf-red-ipv4-unicast)# exit
Leaf1(config-vrf-red)#
```

3. Repeat the preceding configuration for IPv6.

```
Leaf1(config-vrf-red)# address-family ipv6 unicast
Leaf1(config-vrf-red-ipv6-unicast)# route-target import 3:3 evpn
Leaf1(config-vrf-red-ipv6-unicast)# route-target export 3:3 evpn
```

4. Repeat Step 1 through Step 3 for all remaining VRF instances, and return to interface subtype configuration mode.

5. In interface subtype configuration mode, configure the VE interfaces corresponding to the ranged VLANs (101-108, 121-128, 141-148, 161-168) used to support the VRF instances with anycast addresses. The anycast gateway address is used to configure the static anycast gateway MAC addresses for the hosts. This example also shows the optional configuration to support IPv6.

- a) Configure VE 101 to forward VRF "red".

```
Leaf1(config-rbridge-id-1)# interface ve 101
Leaf1(config-Ve-101)# vrf forwarding red
```

- b) Configure VE 101 to support IPv4 and IPv6 static anycast gateway addresses, and enable the interface.

```
Leaf1(config-Ve-101)# ip anycast-address 101.1.1.254/24
Leaf1(config-Ve-101)# ipv6 anycast-address 101:1:1::254/64
Leaf1(config-Ve-101)# no shut
```

6. Repeat Step 5 for VE interfaces 102-108, 121-128, 141-148, 161-168 and their respective VRF instances.
7. In interface subtype configuration mode, configure the VE interfaces 120, 140, 160, 180 to support their respective VRF instances without anycast addresses. These are Layer 3 VNIs and therefore do not need IP addresses.

- a) Configure VE 120 to forward VRF "red".

```
Leaf1(config-rbridge-id-1)# interface ve 120
Leaf1(config-Ve-120)# vrf forwarding red
```

- b) Enable the interface.

```
Leaf1(config-Ve-120)# no shut
```

8. To support IPv6 traffic over the L3 VNI configured above, then configure support for an IPv6 address on the interface.
- a) In RBridge ID configuration mode, enter interface subtype configuration mode and specify the VE interface used above.

```
Leaf1(config-rbridge-id-1)# interface ve 120
```

- b) Enter the **ipv6 address use-link-local-only** command.

```
Leaf1(config-rbridge-Ve-120)# ipv6 address use-link-local-only
```

9. Repeat Step 7 for VE interfaces 140, 160, and 180 to support their respective VRF instances.
10. Repeat Step 1 through Step 9 for the remaining leaf nodes, as appropriate.
11. To verify the configuration, use the **show running-config rbridge-id interface ve** command.

Configuring BGP routing

This task enables BGP routing and configures a variety of parameters.

ATTENTION

IPv4 neighbors are activated by default under the IPv4 address family. To negotiate only the BGP L2VPN EVPN address family on an IPv4 neighbor, you must explicitly deactivate the IPv4 unicast address family by using the **no neighbor activate** command. The BGP configuration examples in this document assume that the neighbors configured for Layer 2 EVPN carry both IPv4 and IPv6 unicast routes as well.

This example illustrates the optional redistribution of OSPF routes into BGP, to support the example spine configuration. This is required only if OSPF is used for the underlay network.

1. Enable BGP routing on an RBridge and enter BGP global configuration mode.

```
Leaf1(config-rbridge-id-1)# router bgp
```

2. Use the **neighbor** command to configure the following attributes.

- a) Configure a local AS number.

NOTE

A local AS number facilitates the merging of networks by allowing a switch in an acquired network to appear to belong to the former AS. In eBGP deployments, the local AS values must be different for each node.

```
Leaf1(config-bgp-router)# local-as 1
```

- b) Configure a remote AS number for both IPv4 and IPv6 addresses to support BGP sessions to the spine.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 remote-as 2
Leaf1(config-bgp-router)# neighbor 1000:2:6:: remote-as 2
Leaf1(config-bgp-router)# neighbor 3.6.0.0 remote-as 2
Leaf1(config-bgp-router)# neighbor 1000:3:6:: remote-as 2
```

3. (Optional) To ensure that sufficient Time To Live (TTL) values are available for hello messages to establish separate neighbor adjacencies on leaf switches when eBGP is used in conjunction with the IP unnumbered feature, it is important to set the number of allowed hops to 2, as in the following example.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 ebgp-multihop 2
Leaf1(config-bgp-router)# neighbor 3.6.0.0 ebgp-multihop 2
```

4. Enter address-family IPv4 unicast configuration mode and configure the following attributes.

- a) Use the **neighbor allowas-in** command to disable the AS_PATH check function for routes learned from a specified location.

```
Leaf1(config-bgp-router)# address-family ipv4 unicast
Leaf1(config-bgp-ipv4u)# neighbor 2.6.0.0 allowas-in 1
Leaf1(config-bgp-ipv4u)# neighbor 3.6.0.0 allowas-in 1
```

NOTE

This prevents BGP from rejecting routes that contain the recipient BGP speaker's AS number, where *number* here specifies the number of times that the AS path of a received route may contain the recipient BGP speaker's AS number and still be accepted.

- b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
Leaf1(config-bgp-ipv4u)# maximum-paths 8
```

- c) Enable BGP recursive next-hop lookups.

```
Leaf1(config-bgp-ipv4u)# next-hop-recursion
```

- d) Enable directly connected routes to be redistributed into BGP.

```
Leaf1(config-bgp-ipv4u)# redistribute connected
```

- e) (Optional) Enable OSPF routes to be redistributed into BGP.

```
Leaf1(config-bgp-ipv4u)# redistribute ospf
```

- f) Exit address-family IPv4 unicast configuration mode.

```
Leaf1(config-bgp-ipv4u)# exit
```

5. (Optional) Add a route map to filter only VTEP addresses.

- a) In global configuration mode, enter the **route-map** command to specify and configure a route map.

```
Leaf1(config)# route-map rm-allow-vtep-addresses permit 5
Leaf1(config-route-map-rm-allow-vtep-addresses/permit/5)# match ip address prefix-list pl-vtep-addresses
Leaf1(config-route-map-rm-allow-vtep-addresses/permit/5)# exit
```

- b) In global configuration mode, enter the **ip prefix-list** command to specify and configure a prefix list.

```
Leaf1(config)# ip prefix-list pl-vtep-addresses
```

- c) In global configuration mode, enter the **ip access-list** command to create a standard access list and enter ACL configuration mode, then use the **seq** command to specify a loopback address.

```
device(config)# ip access-list standard stdACL
device(conf-ipacl-std)# seq 5 permit 6.0.0.6/32
```

The above address must be a loopback address.

6. Enter address-family IPv6 unicast configuration mode and configure the following attributes.

- a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups, and use the **neighbor allows-in** command to disable the AS_PATH check function for routes learned from a specified location.

```
Leaf1(config-bgp-router)# address-family ipv6 unicast
Leaf1(config-bgp-ipv6u)# neighbor 1000:2:6:: activate
Leaf1(config-bgp-ipv6u)# neighbor 1000:2:6:: allows-in 1
Leaf1(config-bgp-ipv6u)# neighbor 1000:3:6:: activate
Leaf1(config-bgp-ipv6u)# neighbor 1000:3:6:: allows-in 1
```

- b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
Leaf1(config-bgp-ipv6u)# maximum-paths 8
```

- c) Enable BGP recursive next-hop lookups.

```
Leaf1(config-bgp-ipv6u)# next-hop-recursion
```

- d) Enable directly connected routes to be redistributed into BGP.

```
Leaf1(config-bgp-ipv6u)# redistribute connected
```

- e) (Optional) Enable OSPF connected routes to be redistributed into BGP, and exit to BGP global configuration mode.

```
Leaf1(config-bgp-ipv6u)# redistribute ospf
Leaf1(config-bgp-ipv6u)# exit
Leaf1(config-bgp-router)#
```

NOTE

This required only if OSPF is used for the underlay network.

7. (Optional) Configure neighbor Bidirectional Forwarding Detection (BFD), by means of the **neighbor bfd** command, to facilitate fast recovery in case of a link failure.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 bfd
Leaf1(config-bgp-router)# neighbor 3.6.0.0 bfd
```

NOTE

For more information, refer to the "BFD" chapter in the *Extreme Network OS Layer 3 Routing Configuration Guide*.

8. (Optional) Configure Bidirectional Forwarding Detection (BFD) session parameters for BGP globally, by means of the **bfd interval** command.

```
device(config-bgp-router)# bfd interval 140 min-rx 125 multiplier 44
```

9. (Optional) To provide support for BFD for IP unnumbered ECMP interfaces, enable BFD and specify multipath BFD as in the following example.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 bfd multipath
```

Repeat as appropriate for all ECMP interfaces. Refer to [BFD for IP Unnumbered ECMP Interfaces](#) on page 19

10. (Optional) Enter address-family Layer 2 Virtual Private Network (VPN) Ethernet VPN (EVPN) configuration mode and configure the following attributes for IPv4 addresses.

- a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups. Use the **neighbor allows-in** command to disable the AS_PATH check function for routes learned from a specified location, and use the **neighbor encapsulation vxlan** command to enable VXLAN encapsulation.

```
Leaf1(config-bgp-router)# address-family l2vpn evpn
Leaf1(config-evpn)# neighbor 2.6.0.0 activate
Leaf1(config-evpn)# neighbor 2.6.0.0 encapsulation vxlan
Leaf1(config-evpn)# neighbor 2.6.0.0 allows-in 1
Leaf1(config-evpn)# neighbor 3.6.0.0 activate
Leaf1(config-evpn)# neighbor 3.6.0.0 encapsulation vxlan
Leaf1(config-evpn)# neighbor 3.6.0.0 allows-in 1
```

NOTE

This step is applicable only when BGP EVPN is used for the overlay.

- b) Enable BGP to send an update to an eBGP peer with the next-hop attribute unchanged.

```
Leaf1(config-evpn)# neighbor 2.6.0.0 next-hop-unchanged
Leaf1(config-evpn)# neighbor 3.6.0.0 next-hop-unchanged
```

11. In BGP router configuration mode, configure a BGP instance for IPv4 unicast address-family for VRF red and use the **redistribute connected** command to redistribute connected routes, and do the same for IPv6.

```
Leaf1(config-bgp-router)# address-family ipv4 unicast vrf red
Leaf1(config-bgp-ipv4u-vrf)# redistribute connected

Leaf1(config-bgp-router)# address-family ipv6 unicast vrf red
Leaf1(config-bgp-ipv6u-vrf)# redistribute connected
```

12. Repeat the above steps on the remaining leaf nodes as appropriate.
13. To verify the configuration, use the **show running-config rbridge-id router bgp**, the **show ip bgp summary**, the **show bgp evpn summary**, and the **show bfd neighbors** commands.

Configuring the overlay gateway

This task creates an overlay gateway instance and configures a variety of parameters.

1. In global configuration mode, create an overlay gateway instance and enter overlay-gateway configuration mode.

```
Leaf1(config)# overlay-gateway Leaf1
```

2. In overlay-gateway configuration mode, specify the type as **layer2-extension**.

```
Leaf1(config-overlay-gw-Leaf1)# type layer2-extension
```

3. Specify the IP interface as Loopback 1.

```
Leaf1(config-overlay-gw-Leaf1)# ip interface loopback 1
```

4. Attach the RBridge ID of the current leaf switch.

```
Leaf1(config-overlay-gw-Leaf1)# attach rbridge-id add 1
```

5. (Optional) To support dual-homing, you must do the following.

- a) Use the **add** keyword to specify the RBridge IDs of each of the two supporting nodes.

```
Leaf1(config-overlay-gw-Leaf1)# attach rbridge-id add 1-2
```

- b) In RBridge ID configuration mode, enter the **ip router-id** command to assign a unique router ID to the first node in the vLAG pair.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# ip router-id 101.29.29.1
```

- c) On the second node in pair, repeat Step 5b.

```
device# configure terminal
device(config)# rbridge-id 2
device(config-rbridge-id-2)# ip router-id 101.31.31.1
```

This is required because the same loopback address can be assigned to both the R Bridges in this configuration, and the two nodes must use different router IDs to distinguish the BGP sessions.

6. Specify the automatic mapping of VLANs to VNIs.

```
Leaf1(config-overlay-gw-Leaf1)# map vni auto
```

NOTE

This is the preferred method for ease of configuration. If the user does not want to use automatic VLAN VNI mapping, then manual VLAN-to-VNI mapping can be used. This manual mapping is required for Cisco interoperability. Refer to [Configuring interoperability with other vendors](#) on page 74.

7. Activate the overlay gateway

```
Leaf1(config-overlay-gw-Leaf1)# activate
```

8. Repeat Step 1 through Step 7, as appropriate, for the remaining leaf nodes.

Configuring the spine switches

This task configures underlay physical connectivity between the spine, leaf, and optional superspine switches, and also sets BGP neighbor and AS attributes.

1. In interface subtype configuration mode, configure an Ethernet interface for point-to-point connectivity between spine switch Spine1 and supported leaf switches.

NOTE

This example uses both IPv4 and IPv6 addresses, with /31 and /127 networks, respectively.

```
Spine1# configure terminal
Spine1(config)# rbridge-id 1
Spine1(config-rbridge-id-1)# interface tengigabitethernet 1/1/33
Spine1(config-if-te-1/1/33)# ip address 1.2.1.1/31
Spine1(config-if-te-1/1/33)# ipv6 address 1000:1:2:2::1/127
Spine1(config-if-te-1/1/33)# no shut
Spine1(config-if-te-1/1/33)# exit
Spine1(config-rbridge-id-1)#
```


2. Configure loopback interfaces, to be the donor interfaces used for optional IP unnumbered.

NOTE

This example uses both IPv4 and IPv6 addresses, with /32 and /128 networks, respectively.

```
Spine1(config-rbridge-id-1)# interface loopback 1
Spine1(config-Loopback-1)# ip address 2.0.0.2/32
Spine1(config-Loopback-1)# ipv6 address 1000:2::2/128
Spine1(config-Loopback-1)# no shut
Spine1(config-Loopback-1)# exit
Spine1(config-rbridge-id-1)#
```

3. (Optional) You can configure unnumbered IPv4 and IPv6 addresses, with an appropriate loopback address, as follows.

```
Leaf1(config-rbridge-id-1)# interface tengigabitethernet 1/1/33
Leaf1(config-if-te-1/1/33)# ip unnumbered loopback 1
Leaf1(config-if-te-1/1/33)# ipv6 unnumbered loopback 1
Leaf1(config-if-te-1/1/33)# no shut
```

4. Repeat Step 1 or Step 2 for all Ethernet interfaces supporting the leaf switches.
5. Enable BGP routing and enter BGP global configuration mode.

```
Spine1(config-rbridge-id-1)# router bgp
Spine1(config-bgp-router)#
```

6. Use the **neighbor** command to configure the following attributes.

- a) Configure a local AS number for the spine switch.

NOTE

A local AS number facilitates the merging of networks by allowing a switch in an acquired network to appear to belong to the former AS. In eBGP deployments, the local AS values must be different for each node.

```
Spine1(config-bgp-router)# local-as 2
```

- b) Configure a remote AS number for both IPv4 and IPv6 addresses to support BGP sessions to the superspine.

```
Spine1(config-bgp-router)# neighbor 1.2.1.0 remote-as 1
Spine1(config-bgp-router)# neighbor 1000:1:2:1:: remote-as 1
```

- c) Configure remote AS numbers for both IPv4 and IPv6 addresses to support BGP sessions to the leaf switches.

```
Spine1(config-bgp-router)# neighbor 2.6.0.1 remote-as 1
Spine1(config-bgp-router)# neighbor 1000:2:6:1:: remote-as 1
```

- d) Repeat Step 6a through Step 6c, as appropriate, for the remaining leaf interfaces. The following shows the results of the configuration.

```
neighbor 2.7.0.1 remote-as 3
neighbor 1000:2:7::1 remote-as 3
neighbor 2.8.0.1 remote-as 3
neighbor 1000:2:8::1 remote-as 3
neighbor 2.9.0.1 remote-as 3
neighbor 1000:2:9::1 remote-as 3
```

- e) (Optional) To provide support for BFD for IP unnumbered ECMP interfaces, enable BFD and specify multipath BFD as in the following example.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 bfd multipath
```

Repeat as appropriate for all ECMP interfaces. Refer to [BFD for IP Unnumbered ECMP Interfaces](#) on page 19

7. Enter address-family IPv4 unicast configuration mode and configure the following attributes.
 - a) (Optional) Use the **neighbor allowas-in** command to disable the AS_PATH check function for routes learned from a superspine.

NOTE

This prevents BGP from rejecting routes that contain the recipient BGP speaker's AS number, where *number* here specifies the number of times that the AS path of a received route may contain the recipient BGP speaker's AS number and still be accepted.

```
Spine1(config-bgp-router)# address-family ipv4 unicast
Spine1(config-bgp-ipv4u)# neighbor 1.2.1.0 allowas-in 1
```

- b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
Spine1(config-bgp-ipv4u)# maximum-paths 8
```

- c) Enable BGP recursive next-hop lookups.

```
Spine1(config-bgp-ipv4u)# next-hop-recursion
```

- d) (Optional) Enable directly connected routes to be redistributed into BGP.

```
Spine1(config-bgp-ipv4u)# redistribute connected
```

- e) (Optional) Enable OSPF routes to be redistributed into BGP.

```
Spine1(config-bgp-ipv4u)# redistribute ospf
```

- f) Exit address-family IPv4 unicast configuration mode.

```
Spine1(config-bgp-ipv4u)# exit
Spine1(config-bgp-router)#
```

8. Enter address-family IPv6 unicast configuration mode and configure the following attributes.

- a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups, and optionally use the **neighbor allows-in** command to disable the AS_PATH check function for routes learned from a superspine.

```
Spine1(config-bgp-router)# address-family ipv6 unicast
Spine1(config-bgp-ipv6u)# neighbor 1000:1:2:1:: activate
Spine1(config-bgp-ipv6u)# neighbor 1000:1:2:1:: allows-in 1
```

- b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
Spine1(config-bgp-ipv6u)# maximum-paths 8
```

- c) Enable BGP recursive next-hop lookups.

```
Spine1(config-bgp-ipv6u)# next-hop-recursion
```

- d) Enable directly connected routes to be redistributed into BGP.

```
Spine1(config-bgp-ipv6u)# redistribute connected
```

- e) Enable OSPF routes to be redistributed into BGP.

```
Spine1(config-bgp-ipv6u)# redistribute ospf
```

- f) Exit address-family IPv6 unicast configuration mode, and enter BGP configuration mode.

```
Spine1(config-bgp-ipv6u)# exit
Spine1(config-bgp-router)#
```

9. (Optional) Enable Bidirectional Forwarding Detection (BFD) for BGP neighbors, to facilitate fast recovery in case of a link failure.

```
Spine1(config-bgp-router)# neighbor 1.2.1.0 bfd
Spine1(config-bgp-router)# neighbor 1000:1:2:1:: bfd
```

Repeat Step 10 for all IPv4 and IPv6 neighbors.

10. (Optional) Enter address-family Layer 2 Virtual Private Network (VPN) Ethernet VPN (EVPN) configuration mode and configure the following attributes for IPv4 addresses.

- a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups, and use the **neighbor allows-in** command to disable the AS_PATH check function for routes learned from a specified location.

```
Spine1(config-bgp-router)# address-family l2vpn evpn
Spine1(config-evpn)# neighbor 1.2.1.0 activate
Spine1(config-evpn)# neighbor 1.2.1.0 allows-in 1
```

- b) Enable BGP to send an update to an eBGP peer with the next-hop attribute unchanged.

```
Spine1(config-evpn)# neighbor 1.2.1.0 next-hop-unchanged
```

- c) Repeat Step 10a through Step 10b, as appropriate, for the remaining IPv4 leaf interfaces. The following shows the results of the configuration.

```
neighbor 2.6.0.1 activate
neighbor 2.6.0.1 next-hop-unchanged
neighbor 2.7.0.1 activate
neighbor 2.7.0.1 next-hop-unchanged
neighbor 2.8.0.1 activate
neighbor 2.8.0.1 next-hop-unchanged
neighbor 2.9.0.1 activate
neighbor 2.9.0.1 next-hop-unchanged
```

11. Enter the **retain-route-target-all** under BGP EVPN address-family configuration mode to configure the route reflector (RR) to accept all route targets (RTs), preventing filtering on routes that do not match the local configuration on the spine.

```
Spine1(config-bgp-router)# address-family l2vpn evpn
Spine1(config-bgp-evpn)# retain route-target all
```

12. (Optional) Where iBGP is used, use the **neighbor update-source** command to default to the loopback address as a backup interface in case of a link failure.

```
Spine1(config-bgp-router)# neighbor 2.0.0.2 update-source loopback 1
Spine1(config-bgp-router)# neighbor 1000:2::2 update-source loopback 1
```

13. Repeat Step 1 through Step 12, as appropriate, for the remaining interfaces.

14. Repeat Step 1 through Step 13, as appropriate, for Spine2.

15. To verify the configuration, use the **show ip bgp summary**, the **show bgp evpn summary**, and the **show bfd neighbors** commands.

(Optional) Configuring a BGP EVPN instance

A BGP EVPN instance (EVI) can be configured and various commands can be executed in EVPN instance configuration mode, as shown in the following configuration example.

NOTE

BGP EVPN is not essential to the configuration of an IP Fabric, and is one among a variety of overlay solutions.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **evpn-instance** command and specify a name to configure an EVPN instance and enter EVPN instance configuration mode.

```
device(config-rbridge-id-1)# evpn-instance myinstance
```

4. Enter the **rd auto** command to enable auto-generation of the RD value.

```
device(config-evpn-instance-myinstance)# rd auto
```

5. Enter the **route-target** command using the **both** and **auto** parameters to configure auto-generation of the import and export route-target community attributes globally.

```
device(config-evpn-instance-myinstance)# route-target both auto
```

6. Enter the **vni** command with the **add value** parameter to add VLANs to the EVI.

```
device(config-evpn-instance-myinstance)# vni add 1-100
```

Ensure that only local VNIs are added, supporting local VLANs. With symmetric integrated routing and bridging (IRB), if a remote VNI is added that is not on the local switch, that VNI is treated as being used for Layer 2 extension and ARP entries are not programmed in hardware, resulting in potential performance problems. For example, if switch Switch-1 has only VNI 100 and another switch, Switch-2, has VNI 200, you would execute the command **vni add 100** on Switch-1 and the command **vni add 200** on Switch-2.

7. (Optional) Enter the **duplicate-mac-timer count** command with the **max-count interval** parameter to set the duplicate MAC detection timer interval and the maximum count.

```
device(config-evpn-instance-myinstance)# duplicate-mac-timer 22 max-count 5
```

The following example configures the EVPN instance "myinstance" so that the RD value is generated automatically. The import and export route-target community attributes are also generated automatically. VLANs are added to the EVI. The duplicate MAC detection timer is set to 22 seconds and the number of times a MAC move can be detected in the configured 22 second interval before the MAC is suppressed is set to 5.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# evpn-instance myinstance
device(config-evpn-instance-myinstance)# rd auto
device(config-evpn-instance-myinstance)# route-target both auto
device(config-evpn-instance-myinstance)# vni add 1-100
device(config-evpn-instance-myinstance)# duplicate-mac-timer 22 max-count 5
```

(Optional) Configuring support for dual-homed servers

This optional task illustrates the configuration of connectivity between a leaf node (in a pair of supporting nodes) and a server over a port-channel vLAG where high availability (HA) is required through multihoming. This configuration must also be applied to the peer (supporting) leaf node in the HA pair. Both nodes are considered a vLAG pair.

This task involves the following:

- Use the **vlag ignore-split** command where a vLAG spans more than one node. This minimizes packet loss if one of the nodes goes down, and also reduces vLAG failover downtime.
- The **switchport** commands set the Layer 2 characteristics of the interface.
- Ethernet Segment Identifiers (ESIs) are optionally used to identify the links connecting multiple ToR devices to a server in a multihomed BGP EVPN environment. As part of the port-channel configuration, this task uses the **esi** command to specify a nondefault Ethernet Segment Identifier (ESI) for the interface if LACP autodiscovery is not deployed. The default ESI value, for single homing, is 0. Refer to [Ethernet Segment Identifiers for BGP routing](#) on page 38.
- Ensure that spanning tree is disabled. (Spanning tree is disabled by default.)

1. From global configuration mode, configure a router ID for the first node in the supporting pair.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# ip router-id 101.29.29.1
device(config-rbridge-id-1)# exit
device(config)#
```

2. From global configuration mode, specify a port-channel interface and enter port-channel configuration mode.

```
device(config)# interface port-channel 1
device(config-Port-channel-1)#
```

3. Use the **vlag ignore-split** command to minimize packet loss.

```
device(config-Port-channel-1)# vlag ignore-split
```

4. Configure Layer 2 characteristics and specify a VLAN.

- a) Put the interface in Layer 2 mode.

```
device(config-Port-channel-1)# switchport
```

- b) Use the **switchport mode access** command to specify the mode as "access," enabling the port to be assigned to a VLAN.

```
device(config-Port-channel-1)# switchport mode access
```

- c) Use the **switchport access vlan** command to specify a VLAN. (Your requirements will vary.)

```
device(config-Port-channel-1)# switchport access vlan 500
```

5. (Optional) Use the **spanning-tree shutdown** command to ensure that spanning tree (disabled by default) is disabled.

```
device(config-Port-channel-1)# spanning-tree shutdown
```

- (Optional) To support optional BGP EVPN, you can use the **esi auto lacp** command to specify a nondefault Ethernet Segment Identifier that is automatically generated.

NOTE

Where ESI values are assigned manually, the same ESI number must not be repeated on other port channels. You can use the **esi** command to assign a value manually. However, beginning with Network OS 7.0.1, it is not mandatory to assign an ESI value to a port-channel, either automatically or manually. An ESI value of 0 is used.

```
device(config-Port-channel-1)# esi auto lacp
```

- Enable the port-channel interface.

```
device(config-Port-channel-1)# no shut
```

- Repeat Step 1 through Step 7 on the peer leaf node of the HA (vLAG) pair, using the **ip router-id** command to assign a unique router ID as in Step 1.

```
device# configure terminal
device(config)# rbridge-id 2
device(config-rbridge-id-2)# ip router-id 101.31.31.1
device(config-rbridge-id-2)# exit
```

Configuring a superspine switch

This task configures a superspine switch that can be used to interconnect one datacenter pod to another. This example uses OSPF as the routing protocol for the interconnect.

- In interface subtype configuration mode, configure an Ethernet interface for point-to-point connectivity between superspine switch SuperSpine and supported spine switches.

NOTE

This example uses both IPv4 and IPv6 addresses, with /31 and /127 networks, respectively.

```
SuperSpine# configure terminal
SuperSpine(config)# rbridge-id 1
SuperSpine(config-rbridge-id-1)# interface fortygigabitethernet 1/4/11
SuperSpine(config-if-fo-1/4/11)# ip address 1.4.0.0/31
SuperSpine(config-if-fo-1/4/11)# ipv6 address 1000:1::/127
SuperSpine(config-if-fo-1/4/11)# no shut
SuperSpine(config-if-fo-1/4/11)# exit
SuperSpine(config-rbridge-id-1)#
```

- Configure a loopback interface.

NOTE

This example uses both IPv4 and IPv6 addresses, with /32 and /128 networks, respectively.

```
SuperSpine(config-rbridge-id-1)# interface loopback 1
SuperSpine(config-Loopback-1)# ip address 1.0.0.1/32
SuperSpine(config-Loopback-1)# ipv6 address 1000:1::1/128
SuperSpine(config-Loopback-1)# no shut
SuperSpine(config-Loopback-1)# exit
SuperSpine(config-rbridge-id-1)#
```

3. (Optional) You can configure unnumbered IPv4 and IPv6 addresses, with an appropriate loopback address, as follows.

```
SuperSpine(config-rbridge-id-1)# interface fortygigabitethernet 1/4/11
SuperSpine(config-if-fo-1/4/11)# ip unnumbered loopback 1
SuperSpine(config-if-fo-1/4/11)# ipv6 unnumbered loopback 1
SuperSpine(config-if-fo-1/4/11)# no shut
```

4. Repeat Step 1 through Step 3, as appropriate, for all Ethernet interfaces supporting the spine switches.
5. Enable BGP routing and enter BGP global configuration mode.

```
SuperSpine(config-rbridge-id-1)# router bgp
SuperSpine(config-bgp-router)#
```

6. Configure a local AS for the superspine switch.

```
SuperSpine(config-bgp-router)# local-as 1
```

NOTE

A local AS number facilitates the merging of networks by allowing a switch in an acquired network to appear to belong to the former AS. In eBGP deployments, the local AS values must be different for each node.

7. Use the **neighbor** command to configure a remote AS number for both IPv4 and IPv6 addresses to support BGP sessions to the spine switches.

```
SuperSpine(config-bgp-router)# neighbor 1.2.1.1 remote-as 2
SuperSpine(config-bgp-router)# neighbor 1000:1:2:1:: remote-as 2
SuperSpine(config-bgp-router)# neighbor 1.3.1.1 remote-as 2
SuperSpine(config-bgp-router)# neighbor 1000:1:5:1:: remote-as 2
```

8. Enter address-family IPv4 unicast configuration mode and configure the following attributes.

- a) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
SuperSpine(config-bgp-ipv4u)# maximum-paths 8
```

- b) Enable BGP recursive next-hop lookups.

```
SuperSpine(config-bgp-ipv4u)# next-hop-recursion
```

- c) Enable directly connected routes to be redistributed into BGP.

```
SuperSpine(config-bgp-ipv4u)# redistribute connected
```

- d) Enable OSPF connected routes to be redistributed into BGP, and exit to BGP configuration mode.

```
Leaf1(config-bgp-ipv4u)# redistribute ospf
Leaf1(config-bgp-ipv4u)# exit
Leaf1(config-bgp-router)#
```


9. Enter address-family IPv6 unicast configuration mode and configure the following attributes.

- a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups.

```
SuperSpine(config-bgp-router)# address-family ipv6 unicast
SuperSpine(config-bgp-ipv6u)# neighbor 1000:1:2:1::1 activate
SuperSpine(config-bgp-ipv6u)# neighbor 1000:1:3:1::1 activate
```

- b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
SuperSpine(config-bgp-ipv6u)# maximum-paths 8
```

- c) Enable BGP recursive next-hop lookups.

```
SuperSpine(config-bgp-ipv6u)# next-hop-recursion
```

- d) Enable directly connected routes to be redistributed into BGP.

```
SuperSpine(config-bgp-ipv6u)# redistribute connected
```

- e) Enable OSPF connected routes to be redistributed into BGP, and exit to BGP configuration mode.

```
SuperSpine(config-bgp-ipv6u)# redistribute ospf
SuperSpine(config-bgp-ipv6u)# exit
SuperSpine(config-bgp-router)#
```

10. (Optional) Enter address-family Layer 2 Virtual Private Network (VPN) Ethernet VPN (EVPN) configuration mode and configure the following attributes for IPv4 addresses.

- a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups.

```
SuperSpine(config-bgp-router)# address-family l2vpn evpn
SuperSpine(config-evpn)# neighbor 1.2.1.0 activate
```

- b) Enable BGP to send an update to an eBGP multihop peer with the next-hop attribute unchanged.

```
SuperSpine(config-evpn)# neighbor 1.2.1.0 next-hop-unchanged
```

- c) Repeat Step 10a and Step 10b, as appropriate, for the remaining IPv4 leaf interfaces.

11. Enter the **retain-route-target-all** under BGP EVPN address-family configuration mode to configure the route reflector (RR) to accept all route targets (RTs), preventing filtering on routes that do not match the local configuration on the spine.

```
SuperSpine(config-bgp-router)# address-family l2vpn evpn
SuperSpine(config-bgp-evpn)# retain route-target all
```

12. (Optional) Configure neighbor Bidirectional Forwarding Detection (BFD), by means of the **neighbor bfd** command, to facilitate fast recovery in case of a link failure.

```
SuperSpine(config-bgp-router)# neighbor 1.2.1.1 bfd
SuperSpine(config-bgp-router)# neighbor 1000:1:2:1:: bfd
SuperSpine(config-bgp-router)# neighbor 1.3.1.1 bfd
SuperSpine(config-bgp-router)# neighbor 1000:1:3:1:: bfd
```

13. Repeat Step 1 through Step 9, as appropriate, for a peer superspine switch.

Configuring interoperability with other vendors

When BGP EVPN is used, the automatic mapping of VLANs to VNIs may not work with all vendors.

Some vendors do not use the range of available VNIs (1 through 8191) that Extreme does. As a result of the automatic mapping process that is initiated by the **map vlan vni auto** command in overlay-gateway configuration mode (entered by means of the **overlay-gateway** command), a vendor's VLANs that do not conform to the Extreme extended VLAN range must be remapped manually to their respective VNIs.

In addition, the route target (RT) and route distinguisher (RD) values are set automatically. Because such route targets use AS numbers in the administrator field, routes cannot be exchanged between leaf nodes belonging to different AS numbers. In such cases manual adjustments are required, as illustrated in this task.

1. From global configuration mode, enter overlay-gateway configuration mode and specify an overlay gateway.

```
device(config)# overlay-gateway gateway1
device(config-overlay-gw-gateway1)#
```

2. In overlay-gateway configuration mode, do the following.

- a) Use the **map vlan** command to map a VLAN to a VNI manually, as in the following example.

```
device(config-overlay-gw-gateway1)# map vlan 100 vni 10100
```

- b) Repeat Step 2a as appropriate for additional manual mappings.

3. In RBridge ID configuration mode, enter EVPN configuration mode.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# evpn-instance myinstance
device(config-evpn-instance-myinstance)#
```

4. In EVPN instance configuration mode, use the **route-target (EVPN)** command to specify auto-generation of the import and export route-target community attributes and ignore the ASN, and the **rd auto (EVPN)** command to enable auto-generation of a route distinguisher (RD) for an EVPN instance.

```
device(config-evpn-instance-myinstance)# route-target both auto ignore-as
device(config-evpn-instance-myinstance)# rd auto
```

5. In EVPN instance configuration mode, configure the VPN route distinguisher (RD) manually.

- a) Enter the **vni (EVPN)** command to specify the VNI and enter VNI instance configuration mode, where you use the **rd (VNI)** command and the **route-target (VNI)** command to specify manually the RD and RT, respectively.

```
device(config-evpn-instance-myinstance)# vni 10100
device(evpn-vni-10100)# rd 100:100
device(evpn-vni-10100)# route-target import 1:1
device(evpn-vni-10100)# route-target export 1:1
```

- b) Repeat Step 6a for all RDs and RTs that need to be mapped manually.

Verifying the configuration

A variety of **show** commands can be used to verify configurations on a leaf or spine.

Verifying configurations on a leaf

To verify the Ethernet configuration, use the **show running-config interface** command, as in the following example.

```
Leaf1# show running-config interface TenGigabitEthernet 1/0/16
interface TenGigabitEthernet 1/0/16
  switchport
  switchport mode trunk
  switchport trunk allowed vlan add 22-25
  switchport trunk tag native-vlan
  spanning-tree shutdown
  fabric isl enable
  fabric trunk enable
  no shutdown
```

To verify a VLAN configuration, use the **show running-config interface vlan** command, as in the following example.

```
Leaf1# show running-config interface vlan 25
interface Vlan 25
  suppress-nd
  suppress-arp
```

To verify a virtual Ethernet (VE) configuration, use the **show running-config interface rbridge-id interface ve** command, as in the following example.

```
Leaf1# show running-config rbridge-id 1 interface ve 25
rbridge-id 1
interface Ve 25
  vrf forwarding vrf2
  ipv6 anycast-address 2:25:1::254/64
  ip anycast-address 2.25.1.254/24
  no shutdown
```

To verify IPv4 or IPv6 static-anycast-gateway configurations, use the **show running-config rbridge-id ip anycast-gateway-mac** command or the **show running-config rbridge-id ipv6 anycast-gateway-mac** command, respectively, as in the following examples.

```
Leaf1# show running-config rbridge-id 1 ip anycast-gateway-mac
rbridge-id 1
ip anycast-gateway-mac 0000.abba.abba

Leaf1# show running-config rbridge-id 1 ipv6 anycast-gateway-mac
rbridge-id 1
ipv6 anycast-gateway-mac 0000.abba.abba
```

To verify the entire BGP configuration, use the **show running-config rbridge-id router bgp** command, as in the following examples.

```
Leaf1# show running-config rbridge-id 1 router bgp
rbridge-id 1
router bgp
  local-as 65006
  neighbor 1000:2:6:: remote-as 65002
  neighbor 1000:2:6:: password 2 $TDk1UHNkRC1afDg=
  neighbor 1000:3:6:: remote-as 65002
  neighbor 1000:3:6:: password 2 $TDk1UHNkRC1afDg=
  neighbor 2.6.0.0 remote-as 65002
  neighbor 2.6.0.0 password 2 $TDk1UHNkRC1afDg=
  neighbor 2.6.0.0 bfd
  neighbor 3.6.0.0 remote-as 65002
  neighbor 3.6.0.0 password 2 $TDk1UHNkRC1afDg=
  neighbor 3.6.0.0 bfd
  address-family ipv4 unicast
```

```

redistribute connected
neighbor 3.6.0.0 allowas-in 1
neighbor 2.6.0.0 allowas-in 1
maximum-paths 8
next-hop-recursion
!
address-family ipv4 unicast vrf vrf12
redistribute connected
maximum-paths 8
!
address-family ipv4 unicast vrf vrf13
redistribute connected
maximum-paths 8
!
<---output omitted--->
!
address-family ipv4 unicast vrf vrf8
redistribute connected
maximum-paths 8
!
address-family ipv4 unicast vrf vrf9
redistribute connected
maximum-paths 8
!
address-family ipv6 unicast
redistribute connected
neighbor 1000:3:6:: allowas-in 1
neighbor 1000:3:6:: activate
neighbor 1000:2:6:: allowas-in 1
neighbor 1000:2:6:: activate
maximum-paths 8
next-hop-recursion
!
address-family ipv6 unicast vrf vrf12
redistribute connected
maximum-paths 8
!
address-family ipv6 unicast vrf vrf13
redistribute connected
maximum-paths 8
!
<---output omitted--->
!
address-family ipv6 unicast vrf vrf8
redistribute connected
maximum-paths 8
!
address-family ipv6 unicast vrf vrf9
redistribute connected
maximum-paths 8
!
address-family l2vpn evpn
neighbor 3.6.0.0 activate
neighbor 3.6.0.0 allowas-in 1
neighbor 3.6.0.0 next-hop-unchanged
neighbor 2.6.0.0 activate
neighbor 2.6.0.0 allowas-in 1
neighbor 2.6.0.0 next-hop-unchanged

```

To view summary BGP information, use the **show ip bgp summary** command, as in the following example.

```

Leaf1# show ip bgp summary
BGP4 Summary
Router ID: 6.0.0.6   Local AS Number: 65006
Confederation Identifier: not configured
Confederation Peers:
Maximum Number of IP ECMP Paths Supported for Load Sharing: 8
Number of Neighbors Configured: 2, UP: 1
Number of Routes Installed: 17, Uses 2040 bytes

```

```

Number of Routes Advertising to All Neighbors: 19 (16 entries), Uses 960 bytes
Number of Attribute Entries Installed: 6, Uses 690 bytes
Neighbor Address  AS#      State   Time      Rt:Accepted Filtered Sent    ToSend
2.6.0.0           65002   CONN    1h23m33s   0         0       0      16
3.6.0.0           65002   ESTAB   23h28m32s  14        0       3      0

```

To view summary BGP EVPN information, use the **show bgp evpn summary** command, as in the following example.

```

Leaf1# show bgp evpn summary
BGP4 Summary
Router ID: 6.0.0.6   Local AS Number: 65006
Confederation Identifier: not configured
Confederation Peers:
Maximum Number of IP ECMP Paths Supported for Load Sharing: 1
Number of Neighbors Configured: 2, UP: 1
Number of Routes Installed: 4536, Uses 544320 bytes
Number of Routes Advertising to All Neighbors: 5652 (4536 entries), Uses 272160 bytes
Number of Attribute Entries Installed: 2327, Uses 267605 bytes
Neighbor Address  AS#      State   Time      Rt:Accepted Filtered Sent    ToSend
2.6.0.0           65002   CONN    1h23m39s   0         0       0     4536
3.6.0.0           65002   ESTAB   23h28m37s  3420      24     1116    0

```

To verify the BFD neighbors configuration, use the **show bfd neighbors** command, as in the following example.

```

Leaf1# show bfd neighbors
Flags: * indicates State is inconsistent across the cluster
OurAddr      NeighAddr      State      Int      Rbridge-id
=====
3.6.0.1      3.6.0.0        UP         Te 1/0/5    1

```

Verifying configurations on a spine

To verify the BGP configuration, use the **show ip bgp summary** command, as in the following example.

```

Spine2# show ip bgp summary
BGP4 Summary
Router ID: 3.0.0.3   Local AS Number: 65002
Confederation Identifier: not configured
Confederation Peers:
Maximum Number of IP ECMP Paths Supported for Load Sharing: 8
Number of Neighbors Configured: 5, UP: 5
Number of Routes Installed: 22, Uses 2640 bytes
Number of Routes Advertising to All Neighbors: 70 (16 entries), Uses 960 bytes
Number of Attribute Entries Installed: 7, Uses 805 bytes
Neighbor Address  AS#      State   Time      Rt:Accepted Filtered Sent    ToSend
1.3.0.0           65001   ESTAB   23h14m16s   4         0      13     0
3.6.0.1           65006   ESTAB   23h30m19s   3         0      14     0
3.7.0.1           65078   ESTAB   23h21m35s   3         0      15     0
3.8.0.1           65078   ESTAB   23h21m47s   3         0      14     0
3.9.0.1           65009   ESTAB   23h24m15s   3         0      14     0

```

To verify the BGP EVPN configuration, use the **show bgp evpn summary** command, as in the following example.

```

Spine2# show bgp evpn summary
BGP4 Summary
Router ID: 3.0.0.3   Local AS Number: 65002
Confederation Identifier: not configured
Confederation Peers:
Maximum Number of IP ECMP Paths Supported for Load Sharing: 1
Number of Neighbors Configured: 5, UP: 5
Number of Routes Installed: 4536, Uses 544320 bytes
Number of Routes Advertising to All Neighbors: 18144 (4536 entries), Uses 272160 bytes
Number of Attribute Entries Installed: 2200, Uses 253000 bytes
Neighbor Address  AS#      State   Time      Rt:Accepted Filtered Sent    ToSend
1.3.0.0           65001   ESTAB   23h14m37s   0         0     4536    0
3.6.0.1           65006   ESTAB   23h30m40s  1116      0     3420    0
3.7.0.1           65078   ESTAB   23h21m57s  1152      0     3384    0

```

| | | | | | | | |
|---------|-------|-------|-----------|------|---|------|---|
| 3.8.0.1 | 65078 | ESTAB | 23h22m 9s | 1152 | 0 | 3384 | 0 |
| 3.9.0.1 | 65009 | ESTAB | 23h24m36s | 1116 | 0 | 3420 | 0 |

To verify the BFD neighbors configuration, use the **show bfd neighbors** command, as in the following example.

```
Spine2# show bfd neighbors
Flags: * indicates State is inconsistent across the cluster
OurAddr      NeighAddr      State      Int
Rbridge-id
=====
3.6.0.0      3.6.0.1        UP         Te 1/1/5      1
```

Using traceroute for overlay tunnels

The tunnel traceroute feature provides underlay hop information and overlay reachability information in an overlay tunnel, for enhanced visibility of packet traversal between network virtualization edge (NVE) nodes in a Clos IP Fabrics network.

For details about this feature, see the "Traceroute for Overlay Tunnels" chapter in the *Extreme Network OS Layer 3 Routing Configuration Guide*.

Configuring additional features for IP Fabrics

The following optional tasks enhance the implementation and management of IP Fabrics. The configuration of BGP EVPN, although demonstrated as the overlay implementation shown in this guide, is not essential to the deployment of an IP Fabric.

Configuring MAC learning of Layer 2 extension site through BGP

The user can choose the way MAC learning is achieved for a Layer 2 extension tunnel.

BGP routing must be configured.

Data plane (Layer 2) MAC address learning is enabled by default at the remote site. However, this leads to scalability issues. Where BGP routing is used, do the following to enable MAC learning by means of BGP instead. This delegates the responsibility for MAC learning on a tunnel to the Layer 3 control-plane protocol, such as BGP EVPN.

1. Enter VXLAN overlay-gateway site configuration mode.

```
device# configure terminal
device(config)# overlay-gateway gateway1
device(config-overlay-gw-gateway1)# site mysite
device(config-overlay-gw-gateway1-site-mysite)#
```

2. Enter the **mac-learning protocol bgp** command.

```
device(config-overlay-gw-gateway1-site-mysite)# mac-learning protocol bgp
```

3. To disable BGP MAC learning and return to the default of Layer 2 learning, use the **no mac-learning protocol bgp** command.

```
device(config-overlay-gw-gateway1-site-mysite)# no mac-learning protocol bgp
```

Disabling automatic VXLAN tunnel endpoint discovery by BGP

Automatic VXLAN tunnel endpoint (VTEP) discovery by BGP is enabled by default and can be disabled by means of the following procedure.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

5. Enter the **no vtep-discovery** command to disable automatic VTEP discovery by BGP.

```
device(config-bgp-evpn)# no vtep-discovery
```

The following example disables automatic VTEP discovery by BGP.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# no vtep-discovery
```

Configuring BGP next hop unchanged

BGP next hop unchanged can be configured for the L2VPN EVPN address family, allowing BGP to send updates to eBGP peers with the next hop attribute unchanged. The **neighbor next-hop-unchanged** command should be configured for the Spine switch for each EVPN neighbor if the leaf switch is an eBGP peer. The **neighbor next-hop-unchanged** command should be configured for the leaf switch for each EVPN neighbor if the spine switch is an eBGP peer.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor ipv4 remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
```

6. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

7. Enter the **neighbor ip address activate** command to establish a BGP EVPN session with the eBGP neighbor.

```
device(config-bgp-evpn)# neighbor 10.1.1.1 activate
```

8. Enter the **neighbor ip address neighbor next-hop-unchanged** command to configure BGP next hop unchanged.

```
device(config-bgp-evpn)# neighbor 10.1.1.1 next-hop-unchanged
```

The following example establishes a BGP EVPN session with an eBGP neighbor and configures BGP next hop unchanged so that updates can be sent to the neighbor with the next hop attribute unchanged.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 10.1.1.1 activate
device(config-bgp-evpn)# neighbor 10.1.1.1 next-hop-unchanged
```

Configuring BGP retain route target

BGP retain route target can be configured for the L2VPN EVPN address family so that a route reflector (RR) accepts all route targets (RTs). For iBGP, enable this feature on a spine switch so that route-target filtering is not performed and all route targets are retained and route-target attributes in the EVPN routes are not modified or removed.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor ipv4 remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
```

6. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

7. Enter the **retain route-target all** command to configure the RR to accept all route targets RTs.

```
device(config-bgp-evpn)# retain route-target all
```


The following example configures a RR to accept all RTs.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# retain route-target all
```

Applying a BGP extended community filter for the L2VPN EVPN address family

A BGP extended community filter can be applied for the L2VPN EVPN address family.

BGP communities must already be defined. For more information on defining BGP communities, refer to the “BGP4” and “BGP4+” chapters in the *Extreme Network OS Layer 3 Routing Configuration Guide*.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **ip extcommunity-list** command and specify a number to set a BGP extended community filter.

```
device(config-rbridge-id-1)# ip extcommunity-list 1 permit rt 123:2
```

4. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

5. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

6. Enter the **neighbor ip-address remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.2.3 remote-as 1001
```

7. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

8. Enter the **neighbor ip-address activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp-evpn)# neighbor 10.1.2.3 activate
```

9. Enter the **neighbor ip-address route-map** command and specify the **in** keyword to apply a route map to incoming routes.

```
device(config-bgp-evpn)# neighbor 10.1.2.3 route-map in extComRmap
```

10. Enter the **neighbor ip-address route-map** command and specify the **out** keyword to apply a route map to outgoing routes.

```
device(config-bgp-evpn)# neighbor 10.1.2.3 route-map out sendExtComRmap
```

11. Enter the **neighbor ip-address send-community** command and specify the **both** keyword to enable the sending of standard and extended attributes in updates to the specified BGP neighbor.

```
device(config-bgp-evpn)# neighbor 10.1.2.3 send-community both
```

The following example applies a BGP extended community filter for the L2VPN EVPN address family. The steps for configuring a route map are not included in this example.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# ip extcommunity-list 1 permit rt 123:2
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.2.3 remote-as 1001
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 10.1.2.3 activate
device(config-bgp-evpn)# neighbor 10.1.2.3 route-map in extComRmap
device(config-bgp-evpn)# neighbor 10.1.2.3 route-map out sendExtComRmap
device(config-bgp-evpn)# neighbor 10.1.2.3 send-community both
```

Configuring BGP graceful restart for the L2VPN EVPN address family

Graceful restart can be configured for BGP EVPN in the L2VPN EVPN address family configuration mode, helping to retain peer routes and ensure that no route and topology changes occur in the network for the duration of a restart.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor ip address remote-as** command to specify the ASN in which the neighbor resides.

```
device(config-bgp-router)# neighbor 2001:db8:93e8:cc00::1 remote-as 2
```

6. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

7. Enter the **graceful-restart** command to enable the graceful restart feature.

```
device(config-bgp-evpn)# graceful-restart
```

8. Do any of the following:

- Enter the **graceful-restart** command and use the **purge-time** parameter to overwrite the default purge time value.

```
device(config-bgp-evpn)# graceful-restart purge-time 300
```

- Enter the **graceful-restart** command and use the **restart-time** parameter to overwrite the default restart time advertised to graceful restart-capable neighbors.

```
device(config-bgp-evpn)# graceful-restart restart-time 180
```

- Enter the **graceful-restart** command and use the **stale-routes-time** parameter to overwrite the default amount of time that a helper device will wait for an EOR message from a peer.

```
device(config-bgp-evpn)# graceful-restart stale-routes-time 100
```

The following example enables the graceful restart feature.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1
device(config-bgp-router)# neighbor 2001:db8:93e8:cc00::1 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# graceful-restart
```

The following example enables the graceful restart feature and sets the purge time to 300 seconds, overwriting the default value.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1
device(config-bgp-router)# neighbor 2001:db8:93e8:cc00::1
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# graceful-restart purge-time 180
```

The following example enables the graceful restart feature and sets the restart time to 180 seconds, overwriting the default value.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1
device(config-bgp-router)# neighbor 2001:db8:93e8:cc00::1
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# graceful-restart restart-time 180
```

The following example enables the graceful restart feature and sets the stale-routes time to 100 seconds, overwriting the default value.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1
device(config-bgp-router)# neighbor 2001:db8:93e8:cc00::1
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# graceful-restart stale-routes-time 100
```

Configuring BGP peer groups for the L2VPN EVPN address family

A peer group can be created and activated in the L2VPN EVPN address family configuration mode.

- Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor peer-group-name peer-group** command to create a peer group.

```
device(config-bgp-router)# neighbor mypeergroup1 peer-group
```

6. Enter the **neighbor peer-group-name remote-as** command to specify the ASN of the peer group.

```
device(config-bgp-router)# neighbor mypeergroup1 remote-as 11
```

7. Enter the **neighbor ipv6-address peer-group** command to associate a neighbor with the peer group.

```
device(config-bgp-router)# neighbor 2001:2018:8192::125 peer-group mypeergroup1
```

8. Enter the **neighbor ipv6-address peer-group** command to associate a second neighbor with the peer group.

```
device(config-bgp-router)# neighbor 2001:2018:8192::124 peer-group mypeergroup1
```

9. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

10. Enter the **neighbor peer-group-name activate** command to establish a BGP EVPN session with the peer group.

```
device(config-bgp-evpn)# neighbor mypeergroup1 activate
```

The following example creates a peer group, specifying two neighbors to belong to the peer group, and activates the peer group in L2VPN EVPN.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor mypeergroup1 peer-group
device(config-bgp-router)# neighbor mypeergroup1 remote-as 11
device(config-bgp-router)# neighbor 2001:2018:8192::125 peer-group mypeergroup1
device(config-bgp-router)# neighbor 2001:2018:8192::124 peer-group mypeergroup1
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor mypeergroup1 activate
```

Configuring a route reflector client for the L2VPN EVPN address family

A BGP peer can be configured as a route reflector (RR) client for the L2VPN EVPN address family. When iBGP is used for BGP EVPN in an IP Fabric, spine switches are configured as RRs and leaf switches are configured as RR clients. The following task configures an RR client.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor ipv4 remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 3
```

6. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

7. Enter the **neighbor ip address route-reflector-client** command to configure a specified neighbor to be a route reflector client.

```
device(config-bgp-evpn)# neighbor 10.1.1.1 route-reflector-client
```

The following example configures a neighbor with the IP address 10.1.1.1 to be a route reflector client in L2VPN EVPN configuration mode.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 3
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 10.1.1.1 route-reflector-client
```

Disabling client-to-client reflection for the L2VPN EVPN address family

For iBGP, if the leaf switches are configured as route reflector clients and are connected in a full iBGP mesh, you can disable client-to-client reflection on the spine switch that is configured as a route reflector (RR).

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor ipv4 remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 3
```

6. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

7. Enter the **no client-to-client-reflection** command to disable client-to-client reflection.

```
device(config-bgp-evpn)# no client-to-client-reflection
```

The following example disables client-to-client reflection in L2VPN EVPN configuration mode.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 3
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# no client-to-client-reflection
```

Disabling the BGP AS_PATH check function for the L2VPN EVPN address family

A device can be configured so that the AS_PATH check function for routes learned from a specific peer is disabled, and routes that contain the recipient BGP speaker's AS number are not rejected. For eBGP, with leaves in one AS and spines in another AS, the AS_PATH check function can be disabled for a leaf switch, so that route updates from the Spine layer, containing routes from the other leaves, are not discarded by the receiving leaf.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor ipv6-address remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 2001:db8:e0ff:783a::4 remote-as 2
```

6. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

7. Enter the **neighbor ipv6 address activate** command to establish a BGP EVPN session with the eBGP neighbor.

```
device(config-bgp-evpn)# neighbor 2001:db8:e0ff:783a::4 activate
```

8. Enter the **neighbor ipv6-address allowas-in** command and specify a **number** to disable the BGP AS_PATH check function, and specify the number of times that the AS path of a received route may contain the recipient BGP speaker's AS number and still be accepted.

```
device(config-bgp-evpn)# neighbor 2001:db8:e0ff:783a::4 allowas-in 1
```

This example specifies that the AS path of a received route may contain the recipient BGP speaker's AS number once and still be accepted in L2VPN EVPN.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 2001:db8:e0ff:783a::4 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 2001:db8:e0ff:783a::4 activate
device(config-bgp-evpn)# neighbor 2001:db8:e0ff:783a::4 allowas-in 1
```

Enabling the BGP AS_PATH check function for the sender BGP speaker

A device can be configured so that a BGP sender speaker does not send routes with an AS path that contains the ASN of the receiving speaker.

NOTE

This task enforces the outbound BGP AS_PATH check function for the BGP IPv4 unicast. This feature is also supported for the BGP address-family IPv6 unicast and BGP address-family L2VPN EVPN address families.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 1
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config-rbridge-id-1)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor ip-address remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
```

6. Enter the **address-family ipv4 unicast** command to enter BGP address-family IPv4 unicast configuration mode.

```
device(config-bgp-router)# address-family ipv4 unicast
```

7. Enter the **neighbor ip address enable-peer-as-check** command to enforce the outbound AS_PATH check function for the IPv4 address family.

```
device(config-bgp-ipv4u)# neighbor 10.1.1.1 enable-peer-as-check
```

The following example enables the outbound AS_PATH check function for the BGP IPv4 unicast address family for a particular peer.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
device(config-bgp-router)# address-family ipv4 unicast
device(config-bgp-ipv4u)# neighbor 10.1.1.1 enable-peer-as-check
```

The following example enables the outbound AS_PATH check function for the BGP IPv6 unicast address family for a particular peer.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 2001:2018:8192::125 remote-as 2
device(config-bgp-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 enable-peer-as-check
```

The following example enables the outbound AS_PATH check function for the L2VPN EVPN address family for a particular peer.

```
device# configure terminal
device(config)# rbridge-id 1
device(config-rbridge-id-1)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 10.1.1.1 enable-peer-as-check
```

A neighbor reset is required once this task is complete.

Using BGP VRF route filters

Filtering based on route-map policies can be added to the EVPN IPv4 and IPv6 prefix routes that are imported to or exported from VRFs.

The IPv4/IPv6 prefix routes of a given VRF instance in the BGP routing table can be imported from or exported to the EVPN routing table by means of the **ip-extcommunity-list** command. For details, refer to "BGP VRF route filters in the "BGP4" chapter of the *Extreme Network OS Layer 3 Routing Configuration Guide*.

Configuring an IP Fabric Automatically

| | |
|--|----|
| • Overview of automatic configuration..... | 89 |
| • Configuration requirements and considerations..... | 91 |
| • Configuration method and examples..... | 93 |

Overview of automatic configuration

Manually configuring the underlay and overlay components of an IP Fabric is a significant task. Fortunately, because most of the necessary commands are predefined and need no arguments from the user, this configuration can be automated.

This can be done by means of python scripts (executed natively or by means of a CLI on the switch) that take basic necessary arguments as user input and complete the remaining configuration automatically.

This section presents example configurations of the output of automation for both the underlay and overlay networks:

- Underlay configuration includes the loopback IP or router ID configuration, Layer 3 interface configuration for the links between leaf and spine nodes, and BGP configuration.
- Overlay configuration includes the BGP EVPN, EVPN instance, and VXLAN tunnel-related configuration.

Underlay configuration examples

The IP Fabric underlay configuration helps bring up BGP between the leaf and spine nodes. The configuration on each node in most of the deployments is as in the following examples.

Layer 3 interfaces configuration

All the Layer 3 interfaces pointing to the leaf and spine nodes, respectively, are configured as IP unnumbered interfaces with a loopback IP address as the donor address.

```
interface FortyGigabitEthernet 1/0/49
  ip unnumbered loopback 1
  no shutdown
!
interface FortyGigabitEthernet 1/0/50
  ip unnumbered loopback 1
  no shutdown
!

interface Loopback 1
  ip address 1.1.1.1/32
  no shutdown
!
```

BGP configuration

BGP auto neighbor discovery with Link Layer Discovery Protocol (LLDP) is used to autodiscover BGP neighbors and the configuration is shown below. Configurations are different on leaf and spine nodes.

BGP on leaf nodes

NOTE

Lines in **bold** indicate the difference in BGP configuration between the leaf and spine nodes.

```
router bgp
  local-as 65535
  neighbor lldp-grp peer-group
  neighbor lldp-grp accept-lldp-neighbors
  neighbor lldp-grp ebgp-multihop
  neighbor lldp-grp password 2 $PVNHITJVPWQ=
  address-family ipv4 unicast

  redistribute connected
  neighbor lldp-grp allowas-in 1
  neighbor lldp-grp enable-peer-as-check
  !
  address-family ipv6 unicast
```

BGP on spine nodes configuration

```
router bgp
  local-as 65534
  neighbor lldp-grp peer-group
  neighbor lldp-grp accept-lldp-neighbors
  neighbor lldp-grp ebgp-multihop
  neighbor lldp-grp password 2 $PVNHITJVPWQ=
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
```

Overlay configuration examples

An IP Fabric overlay configuration includes the BGP EVPN, EVPN instance, and VXLAN tunnel-related configuration. The basic configuration on each node is shown below.

BGP configuration

BGP with EVPN address family capabilities is configured on leaf and spine nodes as shown below, with the differences between leaf and spine nodes highlighted.

On leaf nodes:

```
router bgp
  address-family l2vpn evpn
  neighbor lldp-grp activate
  neighbor lldp-grp allowas-in 1
  neighbor lldp-grp enable-peer-as-check
  !
```

On spine nodes:

```
router bgp
  address-family l2vpn evpn
  retain route-target all
  neighbor lldp-grp activate
  neighbor lldp-grp next-hop-unchanged
  !
```

EVPN instance configuration (only on leaf nodes)

```
evpn-instance default
  route-target both auto
  rd auto
  !
```

```
ip anycast-gateway-mac default-mac
ipv6 anycast-gateway-mac default-mac
```

NOTE

Additionally, the required VLANs must be added to the EVPN instance configuration to bring the IP Fabric overlay up.

VXLAN tunnel configuration (only on leaf nodes)

```
overlay-gateway tunnel
type layer2-extension
ip interface Loopback 1
attach rbridge-id add 1
map vlan vni auto
activate
!
```

NOTE

In case of a dual-node or multi-node VCS, a second loopback IP address configuration is needed, to act as the address of the logical VTEP. The second address is used in the overlay-gateway configuration to specify the IP interface. Additionally, the required VLANs must be added into the EVPN instance to bring the overlay network up.

Configuration requirements and considerations

As already noted, a considerable amount of manual configuration would be required on each node for a network administrator or user to bring the IP Fabric underlay and overlay up.

However, because most of the commands are predefined and do not require arguments, the IP Fabric configuration can be automated, and the number of commands to be configured to bring the underlay and overlay up can be reduced to a single command to invoke a python script (ipfabric-config-automation.py) that takes the basic necessary arguments as input from the user and configures the rest automatically.

The minimal configuration arguments required are detailed in the following sections.

Loopback IP address

This is the loopback IP address, plus the router ID, used for the switch. It is used as donor IP address for all the IP unnumbered interfaces between leaf and spine nodes, and also as the virtual tunnel endpoint (VTEP) IP address for the overlay-gateway tunnel in a single-node VCS switch.

The loopback IP address can be specified by means of an **-a** or **--ip** option in the ipfabric-config-automation.py script. With this option, interface Lo 255 is configured with the provided IP address.

Role (leaf or spine)

Depending on the role of the switch, the BGP configuration varies slightly, as indicated in the underlay and overlay configuration section above.

The role can be specified by means of an **-r** or **--role** option in the ipfabric-config-automation.py script. Accepted values are **leaf** or **spine**.

NOTE

Only the **Loopback-IP** and **Role** options are mandatory arguments without which the script terminates. The rest of the configurations listed below are optional. If not specified, the respective default values of each option are used.

Interface range

This is the comma-separated values (CSV) list of interface numbers or interface range in `<slot/port>` or `<slot/port1-port2>` format (for example, 0/1-4, 1/7, 2/10-14). Interfaces 0/1, 0/2, 0/3, 0/4, 1/7, 2/10, 2/11, 2/12, 2/13, 2/14 are configured as IP unnumbered where BGP auto neighbor discovery is enabled.

This is an optional argument.

By default, all the available 40G and 100G interfaces on the switch are configured as Layer 3 IP unnumbered interfaces.

The interface range can be specified by means of the `-r` or `--range` option in the `ipfabric-config-automation.py` script.

RBridge ID

This is the *rbridge-id* of the switch on which the IP Fabric commands are to be executed.

This option has significance on multi-node VCS leaf switches because, in a multi-node VCS scenario, all the nodes in the VCS must be configured only from the primary node; this option helps to determine the *rbridge-id* of the target node where the IP Fabric commands are to be configured.

Consequently, in multi-node VCS scenario, the `ipfabric-config-automation.py` script should be executed multiple times from the primary node, once for each target node in the VCS with the respective *rbridge-id* value specified.

ATTENTION

An error is thrown if a node with a given *rbridge-id* is not part of the VCS or if the script is not executed from the primary nodes in the VCS.

This is an optional parameter. By default, the *rbridge-id* of the local node on which the script is being executed is used to configure the IP Fabric commands.

The *rbridge-id* can be specified by means of the `-b` or `--rb` option in the `ipfabric-config-automation.py` script.

NOTE

On multi-node VCS leaf nodes, the `ipfabric-config-automation.py` script/command should be executed once for each node in the VCS from the primary node by specifying the respective *rbridge-id* argument.

Local AS

This specifies the BGP local autonomous system (AS) parameter. The range is 1 through 4294967295.

This is an optional parameter. By default, if local-as configuration is already present in the switch, the same value is used in the script; otherwise, the following local-AS values are used based on the role:

- 65534 for spine nodes
- 65535 for leaf nodes

The local AS can be specified by means of the `-l` or `--local-as` option in the `ipfabric-config-automation.py` script.

Password

This specifies the BGP MD5 password to be used for the BGP sessions in the IP Fabric. Such a password is a prerequisite for BGP auto neighbor discovery sessions to form.

However, in the `ipfabric-config-automation.py` script, this is an optional parameter.

By default, the string "ipfabric" is used as the BGP MD5 password.

The BGP MD5 password can be specified by means of the **-p** or **--password** option in the `ipfabric-config-automation.py` script.

Overlay flag

This is an optional argument that determines whether an IP Fabric overlay needs to be configured.

By default, only an underlay is configured. If the overlay flag is enabled, an overlay gateway instance is created with the name "tunnel".

The overlay flag can be enabled by means of the **-o** or **--overlay** option in the `ipfabric-config-automation.py` script.

LVTEP IP address

This is a second loopback IP address (Lo 254), which is used as logical VTEP IP address in the overlay gateway configuration. This option is valid only on leaf switches.

The LVTEP IP address is needed only on the multinode VCS leaf switches when the overlay flag is enabled.

ATTENTION

An error is thrown if the argument is not specified in this scenario.

This is an optional argument for the single-node VCS leaf switches.

By default, Loopback IP/Router-id configured on Lo 255 is used as the VTEP IP in overlay-gateway configuration.

The LVTEP IP address can be specified by means of the **-t** or **--lvtep** option in the `ipfabric-config-automation.py` script.

Verbose

This is a debug option that specifies verbose script execution output. With this option, all the commands being configured on the switch by the script are listed in detail. This can be used for debugging and informative purposes. This option can be by means of the **-v** or **--verbose** keywords.

This is an optional argument. By default, verbose output is disabled.

Configuration method and examples

Automation is implemented on the switch by means of a python script, `ipfabric-config-automation.py`, which takes the IP Fabric arguments listed above as input, and then generates and configures all the appropriate underlay and overlay commands.

The python script is executed on the switch by means of the **execute-script** command or the **python** command in the NOS CLI:

- The **execute-script** command is a NOS CLI command in EXEC mode that can execute any of a predefined list of python scripts in the `/scripts` directory on the switch. This is the preferred method.
- The **python** command executes any python script, given a full path and arguments. Refer to the "Python Event-Management and Scripting" chapter in the *Network OS Management Configuration Guide*.

The syntax of this command is as follows.

```
execute-script ipfabric-config-automation.py -a <Loopback-IP> -r <role:leaf/spine> -i <L3 interface-range> -b
<rbridge-id> -l <local-as> -p <md5-password> -o -t <Logical-VTEP-IP>
```

The following is an example configuration line.

```
execute-script ipfabric-config-automation.py -a 1.1.1.1 -r leaf -i 0/1-4,1/7,2/10-14
```

The following is an example ipfabric-config-automation.py script with options.

```
device# execute-script ipfabric-config-automation.py --help

=====
IP-Fabric Configuration Script Triggered
=====

usage: ipfabric-config-automation.py [-h] -a IP -r {leaf,spine} [-i RANGE]
                                     [-b RBRIDGEID] [-l LOCALAS]
                                     [-p PASSWORD] [-o] [-t LVTEP] [-v]

Arguments:
  -h, --help                show this help message and exit

  -a IP, --ip IP            Loopback IPv4 Address/Router ID (Lo 255)
                           (Mandatory Argument)

  -r {leaf,spine}, --role {leaf,spine}
                           Role: leaf/spine (Mandatory Argument)

  -i RANGE, --range RANGE  L3 Interface(slot/port) or Range(slot/port1-port2)
                           separated by comma. For eg: "0/1-4,1/7,2/9-11"
                           (Optional Argument; Default:All 40G, 100G
interfaces)

  -b RBRIDGEID, --rb RBRIDGEID
                           rBridge-ID (Optional Argument; Default:rBrdige-ID
                           of the node on which the command is executed)

  -l LOCALAS, --local-as LOCALAS
                           BGP Local AS <1-4294967295> (Optional Argument;
                           Default:65534 for spine & 65535 for leaf nodes)

  -p PASSWORD, --password PASSWORD
                           BGP MD5 Password (Optional Argument;
                           Default:"ipfabric")

  -o, --overlay            Enable Overlay Configuration (Optional Argument;
                           Default:Overlay configuration disabled & Default
                           overlay-gateway instance name="tunnel")

  -t LVTEP, --lvtep LVTEP
                           Logical VTEP IPv4 Address (Lo 254) (Optional
                           Argument except in Multi-Node VCS case when
                           overlay flag is enabled; Default:Router ID
                           configured using -a or -ip option)

  -v, --verbose            Enable Verbose Output. Lists all the Commands
                           Being configured (Optional Argument;
                           Default:Verbose output disabled)
```

The following example configures an underlay network only.

```
device# execute-script ipfabric-config-automation.py -a 3.3.3.3 -r leaf -i 0/1

=====
IP-Fabric Configuration Script Triggered
=====

Input/Default Arguments:
  Loopback IP:3.3.3.3
  Role:leaf
  L3 Interface Range: All 40G & 100G Interfaces
  BGP Local-AS:65535
  BGP MD5 Password:ipfabric
```

```

RbridgeID:3
Overlay:False

Configuring IP Fabric ...

%% Loopback Interface Lo 255 configured successfully!
%% L3 Interfaces configured successfully!
%% BGP Underlay configured successfully!

=====
IP-Fabric Underlay Configured Successfully
=====
device#

```

The following example configures both an underlay and an overlay network.

```

device# execute-script ipfabric-config-automation.py -a 3.3.3.3 -r leaf -o

=====
IP-Fabric Configuration Script Triggered
=====

Input/Default Arguments:
  Loopback IP:3.3.3.3
  Role:leaf
  L3 Interface Range: All 40G & 100G Interfaces
  BGP Local-AS:65535
  BGP MD5 Password:ipfabric
  RbridgeID:3
  Overlay:True
  overlay-gateway instance name:tunnel
  Logical Vtep IP:3.3.3.3

Configuring IP Fabric ...

%% Loopback Interface Lo 255 configured successfully!
%% L3 Interfaces configured successfully!
%% BGP Underlay configured successfully!
%% BGP Overlay configured successfully!
%% EVPN Instance configured successfully!
%% Overlay Gateway configured successfully!

=====
IP-Fabric Underlay & Overlay Configured Successfully!!
=====
device#

```

The following example configures an underlay and overlay network on a leaf node with verbose output.

```

device# execute-script ipfabric-config-automation.py -a 3.3.3.3 -r leaf -o -v

=====
IP-Fabric Configuration Script Triggered
=====

Input/Default Arguments:
  Loopback IP:3.3.3.3
  Role:leaf
  L3 Interface Range: All 40G & 100G Interfaces
  BGP Local-AS:65535
  BGP MD5 Password:ipfabric
  RbridgeID:3
  Overlay:True
  overlay-gateway instance name:tunnel
  Logical Vtep IP:3.3.3.3

Configuring IP Fabric ...

CLI CONFIG:

configure terminal

```

```

overlay-gateway tunnel
no activate

configure terminal
rbridge-id 3
interface Loopback 255
no ip address
ip address 3.3.3.3/32
no shutdown

%% Loopback Interface Lo 255 configured successfully!

configure terminal
interface FortyGigabitEthernet 3/0/49
ip unnumbered loopback 255
no shutdown

interface FortyGigabitEthernet 3/0/50
ip unnumbered loopback 255
no shutdown

interface FortyGigabitEthernet 3/0/51
ip unnumbered loopback 255
no shutdown

interface FortyGigabitEthernet 3/0/52
ip unnumbered loopback 255
no shutdown

%% L3 Interfaces configured successfully!

router bgp
local-as 65535
neighbor lldp-grp peer-group
neighbor lldp-grp accept-lldp-neighbors
neighbor lldp-grp ebgp-multihop
neighbor lldp-grp password ipfabric
address-family ipv4 unicast
neighbor lldp-grp allowas-in 1
neighbor lldp-grp enable-peer-as-check
redistribute connected

%% BGP Underlay configured successfully!

router bgp
address-family l2vpn evpn
neighbor lldp-grp allowas-in 1
neighbor lldp-grp enable-peer-as-check
neighbor lldp-grp activate

%% BGP Overlay configured successfully!

configure terminal
rbridge-id 3
evpn-instance default
route-target both auto
rd auto
ip anycast-gateway-mac default-mac
ipv6 anycast-gateway-mac default-mac

%% EVPN Instance configured successfully!

configure terminal
overlay-gateway tunnel
type layer2-extension
ip interface Loopback 255
attach rbridge-id add 3
map vlan vni auto
activate

%% Overlay Gateway configured successfully!

```



```
=====
IP-Fabric Underlay & Overlay Configured Successfully!!
=====
device#
```

The following example configures an underlay and overlay network on a spine node with verbose output.

```
device# execute-script ipfabric-config-automation.py -a 3.3.3.3 -r spine -i 0/1 -o -v

=====
IP-Fabric Configuration Script Triggered
=====

Input/Default Arguments:
  Loopback IP:3.3.3.3
  Role:spine
  L3 Interface Range:0/1
  BGP Local-AS:65534
  BGP MD5 Password:ipfabric
  RbridgeID:3
  Overlay:True

Configuring IP Fabric ...

CLI CONFIG:

configure terminal
rbridge-id 3
interface Loopback 255
  no ip address
  ip address 3.3.3.3/32
  no shutdown

%% Loopback Interface Lo 255 configured successfully!

configure terminal
interface TenGigabitEthernet 3/0/1
  shutdown
  no ip address
  no switchport
  no ip unnumbered
  no channel-group
  ip unnumbered loopback 255
  no shutdown

%% L3 Interfaces configured successfully!

router bgp
  local-as 65534
  neighbor lldp-grp peer-group
  neighbor lldp-grp accept-lldp-neighbors
  neighbor lldp-grp ebgp-multihop
  neighbor lldp-grp password ipfabric

%% BGP Underlay configured successfully!

router bgp
  address-family l2vpn evpn
  retain route-target all
  neighbor lldp-grp next-hop-unchanged
  neighbor lldp-grp activate

%% BGP Overlay configured successfully!

=====
IP-Fabric Underlay & Overlay Configured Successfully!!
=====
device#
```

After `ipfabric-config-automation.py` is executed on all the nodes in the IP Fabric topology as above, the BGP underlay network can be validated by means of the **`show ip bgp summary`** command.

NOTE

The IP Fabric automation script relies on the BGP auto neighbor discovery feature to automate the BGP sessions. Currently, this feature is not supported on SLX platforms.

Appendix A: Supported BGP EVPN Commands

- Configuration commands supporting BGP EVPN..... 99
- Show commands supporting BGP EVPN.....100

Configuration commands supporting BGP EVPN

The following configuration commands support BGP EVPN.

For more details, refer to the *Command Reference*.

RBridge ID configuration mode

- `evpn-instance`

BGP configuration mode

- `address-family l2vpn evpn`

BGP address-family L2VPN EVPN configuration mode

- `client-to-client-reflection`
- `graceful-restart (BGP)`
- `neighbor activate`
- `neighbor allowas-in`
- `neighbor enable-peer-as-check`
- `neighbor maximum-prefix`
- `neighbor next-hop-unchanged`
- `neighbor route-map`
- `neighbor route-reflector-client`
- `neighbor send-community`
- `retain route-target all`
- `vtep-discovery`

EVPN instance configuration mode

- `duplicate-mac-timer`
- `evpn`
- `rd auto (EVPN)`
- `route-target (EVPN)`

-
- vni (EVPN)
- vni add
- vni remove

VNI configuration mode

- rd (VNI)
- route-target (VNI)

VRF configuration mode

- rd (VRF)
- route-target (VRF)
-
- vni (VRF)

Port-channel configuration mode

- esi

VXLAN overlay-gateway site configuration mode

- mac-learning protocol bgp

Show commands supporting BGP EVPN

The following show commands support BGP EVPN.

For more details, refer to the *Command Reference*.

- show bgp evpn dampened-routes
- show bgp evpn interface port-channel
- show bgp evpn interface tunnel
- show bgp evpn l2route detail
- show bgp evpn l2route next-hop
- show bgp evpn l2route summary
- show bgp evpn l2route type arp
- show bgp evpn l2route type auto-discovery
- show bgp evpn l2route type ethernet-segment
- show bgp evpn l2route type inclusive-multicast
- show bgp evpn l2route type mac
- show bgp evpn l2route type nd
- show bgp evpn l2route unreachable

- `show bgp evpn l3vni`
- `show bgp evpn neighbors`
- `show bgp evpn neighbors advertised-routes`
- `show bgp evpn neighbors advertised-routes detail`
- `show bgp evpn neighbors advertised-routes type`
- `show bgp evpn neighbors routes best`
- `show bgp evpn neighbors routes detail`
- `show bgp evpn neighbors routes not-installed-best`
- `show bgp evpn neighbors routes type`
- `show bgp evpn neighbors routes unreachable`
- `show bgp evpn neighbors routes-summary`
- `show bgp evpn routes`
- `show bgp evpn routes best`
- `show bgp evpn routes detail`
- `show bgp evpn routes local`
- `show bgp evpn routes next-hop`
- `show bgp evpn routes no-best`
- `show bgp evpn routes not-installed-best`
- `show bgp evpn routes rd`
- `show bgp evpn routes rd type`
- `show bgp evpn routes summary`
- `show bgp evpn routes type arp`
- `show bgp evpn routes type auto-discovery`
- `show bgp evpn routes type ethernet-segment`
- `show bgp evpn routes type inclusive-multicast`
- `show bgp evpn routes type ipv4-prefix`
- `show bgp evpn routes type ipv6-prefix`
- `show bgp evpn routes type mac`
- `show bgp evpn routes type nd`
- `show bgp evpn routes unreachable`
- `show bgp evpn summary`
- `show mac-address-table`
- `show mac-address-table count evpn`
- `show mac-address-table evpn`

Appendix B: Sample BGP EVPN configuration files

| | |
|--|-----|
| • Sample BGP EVPN superspine configuration using eBGP..... | 103 |
| • Sample BGP EVPN spine configuration using eBGP..... | 103 |
| • Sample BGP EVPN leaf configuration using eBGP..... | 105 |
| • Sample BGP EVPN superspine configuration using iBGP..... | 106 |
| • Sample BGP EVPN spine configuration using iBGP..... | 106 |
| • Sample BGP EVPN leaf configuration using iBGP..... | 107 |

Sample BGP EVPN superspine configuration using eBGP

The following example is a sample BGP EVPN superspine configuration using eBGP in an IP Fabric. The **neighbor remote-as** command is used so that the switches are configured to be in one autonomous system (AS). The switches are configured to advertise routes to other eBGP peers leaving the next-hop attribute unchanged using the **neighbor next-hop-unchanged** command.

NOTE

This configuration sample demonstrates BGP EVPN configuration using eBGP on the superspine switch in an IP Fabric. For full configuration details for configuring an IP Fabric involving all the steps required, refer to [Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay](#) on page 53. For more details on the commands used in this sample, refer to the *Extreme Network OS Command Reference*. For more information on eBGP-based BGP EVPN, refer to [eBGP-based BGP EVPN](#) on page 30.

```
router bgp
  local-as 1
  ! IPv4/IPv6 BGP Sessions to Spine switches
  neighbor 10.2.1.1 remote-as 2
  neighbor 10.3.1.1 remote-as 2
  address-family ipv4 unicast
    maximum-paths 8
    next-hop-recursion
  redistribute connected
  address-family l2vpn evpn
    retain route-target all
  ! Activate EVPN Session to Spine switches
  neighbor 10.2.1.1 activate
  neighbor 10.3.1.1 activate
  neighbor 10.2.1.1 next-hop-unchanged
  neighbor 10.3.1.1 next-hop-unchanged
```

Sample BGP EVPN spine configuration using eBGP

The following example is a sample BGP EVPN spine configuration using eBGP in an IP Fabric. There are two spine switches, spine 1 and spine 2. The spine switches are configured to be in one autonomous system (AS) using the **neighbor remote-as** command and are configured to advertise routes to other eBGP peers leaving the next-hop attribute unchanged using the **neighbor next-hop-unchanged** command.

NOTE

This configuration sample demonstrates BGP EVPN configuration using eBGP on two spine switches in an IP Fabric. For full configuration details for configuring an IP fabric involving all the steps required, refer to [Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay](#) on page 53. For more details on the commands used in this sample, refer to the *Extreme Network OS Command Reference*.

Spine 1

```
router bgp
  local-as 2
  !BGP Sessions to the SuperSpine
  neighbor 10.2.1.0 remote-as 1
  !IPv4/IPv6 BGP Sessions to the leaf switches
  neighbor 10.6.0.1 remote-as 3
  neighbor 10.7.0.1 remote-as 3
  neighbor 10.8.0.1 remote-as 3
  neighbor 10.9.0.1 remote-as 3
  address-family ipv4 unicast
    maximum-paths 8
    next-hop-recursion
    redistribute connected
    neighbor 10.2.1.0 allowas-in 1
  address-family l2vpn evpn
    retain route-target all
    neighbor 10.2.1.0 activate
    neighbor 10.2.1.0 allowas-in 1
    neighbor 10.2.1.0 next-hop-unchanged
    neighbor 10.6.0.1 activate
    neighbor 10.6.0.1 next-hop-unchanged
    neighbor 10.7.0.1 activate
    neighbor 10.7.0.1 next-hop-unchanged
    neighbor 10.8.0.1 activate
    neighbor 10.8.0.1 next-hop-unchanged
    neighbor 10.9.0.1 activate
    neighbor 10.9.0.1 next-hop-unchanged
```

Spine 2

```
router bgp
  local-as 2
  !BGP Sessions to the SuperSpine
  neighbor 10.3.1.0 remote-as 1
  !IPv4/IPv6 BGP Sessions to the leaf switches
  neighbor 10.6.1.1 remote-as 3
  neighbor 10.7.1.1 remote-as 3
  neighbor 10.8.1.1 remote-as 3
  neighbor 10.9.1.1 remote-as 3
  address-family ipv4 unicast
    maximum-paths 8
    next-hop-recursion
    redistribute connected
    neighbor 10.3.1.0 allowas-in 1
  address-family l2vpn evpn
    retain route-target all
    neighbor 10.3.1.0 activate
    neighbor 10.3.1.0 allowas-in 1
    neighbor 10.3.1.0 next-hop-unchanged
    neighbor 10.6.1.1 activate
    neighbor 10.6.1.1 next-hop-unchanged
    neighbor 10.7.1.1 activate
    neighbor 10.7.1.1 next-hop-unchanged
    neighbor 10.8.1.1 activate
    neighbor 10.8.1.1 next-hop-unchanged
    neighbor 10.9.1.1 activate
    neighbor 10.9.1.1 next-hop-unchanged
```


Sample BGP EVPN leaf configuration using eBGP

The following example is a sample BGP EVPN leaf configuration using eBGP in an IP Fabric. The AS_PATH check function is disabled for the leaf switch using the **neighbor allowas-in** command so that route updates from the spine layer are not discarded by the leaf.

NOTE

This configuration sample demonstrates BGP EVPN configuration using eBGP on a leaf switch in an IP Fabric. For full configuration details for configuring an IP fabric involving all the steps required, refer to [Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay](#) on page 53. For more details on the commands used in this sample, refer to the *Extreme Network OS Command Reference*.

```
evpn-instance default
  rd auto
  route-target both auto
  vni add 101-108
  vni add 121-128
  vni add 141-148
  vni add 161-168
  vni add 120
  vni add 140
  vni add 160
  vni add 180
!
router bgp
  local-as 3
  neighbor 10.6.0.0 remote-as 2
  neighbor 10.7.0.0 remote-as 2
  !
  address-family ipv4 unicast
    maximum-paths 8
    next-hop-recursion
    redistribute connected
    neighbor 10.6.0.0 allowas-in 1
    neighbor 10.7.0.0 allowas-in 1
  !
  address-family l2vpn evpn
    neighbor 10.6.0.0 activate
    neighbor 10.7.0.0 activate
    neighbor 10.6.0.0 allowas-in 1
    neighbor 10.7.0.0 allowas-in 1
    neighbor 10.6.0.0 next-hop-unchanged
    neighbor 10.7.0.0 next-hop-unchanged
  !
overlay-gateway Leaf6
  type layer2-extension
  ip interface Loopback 1
  attach rbridge-id add 57
  map vlan vni auto
  activate
```

Sample BGP EVPN superspine configuration using iBGP

The following example is a sample BGP EVPN superspine configuration using iBGP in an IP Fabric.

NOTE

This configuration sample demonstrates BGP EVPN configuration using iBGP on the superspine in an IP Fabric. For full configuration details for configuring an IP Fabric involving all the steps required, refer to [Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay](#) on page 53. For more details on the commands used in this sample, refer to the *Extreme Network OS Command Reference*. For more information on eBGP-based BGP EVPN, refer to the section [iBGP-based BGP EVPN](#) on page 32.

```
router ospf
  area 0
  redistribute connected
!
router bgp
  local-as 3
! IPv4/IPv6 BGP Sessions to Spine switches
  neighbor 10.4.1.1 remote-as 3
  neighbor 10.5.1.1 remote-as 3
address-family ipv4 unicast
  maximum-paths 8
  next-hop-recursion
  redistribute connected
address-family l2vpn evpn
  retain route-target all
! Activate EVPN Session to Spine switches
  neighbor 10.4.1.1 activate
  neighbor 10.5.1.1 activate
  neighbor 10.4.1.1 route-reflector-client
  neighbor 10.5.1.1 route-reflector-client
int fo 100/4/11
  ip address 10.4.0.0/31
  ipv6 address 1000:1:4::/127
  ip ospf area 0
  no shutdown
!
int fo 100/4/12
  ip address 10.5.0.0/31
  ipv6 address 1000:1:5::/127
  ip ospf area 0
  no shutdown
```

Sample BGP EVPN spine configuration using iBGP

The following example is a sample BGP EVPN spine configuration using iBGP in an IP Fabric. The spine switch is configured as a route reflector (RR) and is configured to accept received updates containing at least one route target. Neighboring leaf switches are configured as route reflector (RR) clients using the **route-reflector-client** command.

NOTE

This configuration sample demonstrates BGP EVPN configuration using iBGP on a spine switch in an IP Fabric. For full configuration details for configuring an IP Fabric involving all the steps required, refer to [Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay](#) on page 53. For more details on the commands used in this sample, refer to the *Extreme Network OS Command Reference*.

```
router ospf
  area 0
  redistribute connected
```

```

!
router bgp
  local-as 3
  cluster-id ipv4-address 10.4.4.4
!BGP Sessions to the SuperSpine
!IPv4/IPv6 BGP Sessions to the leaf switches
  neighbor 10.0.0.10 remote-as 3
  neighbor 10.0.0.10 update-source loopback 1
  neighbor 10.0.0.11 remote-as 3
  neighbor 10.0.0.11 update-source loopback 1
  neighbor 10.0.0.12 remote-as 3
  neighbor 10.0.0.12 update-source loopback 1
  neighbor 10.0.0.13 remote-as 3
  neighbor 10.0.0.13 update-source loopback 1
address-family ipv4 unicast
  maximum-paths 8
  next-hop-recursion
  redistribute connected
  client-to-client-reflection
  neighbor 10.0.0.10 route-reflector-client
  neighbor 10.0.0.11 route-reflector-client
  neighbor 10.0.0.12 route-reflector-client
  neighbor 10.0.0.13 route-reflector-client
address-family l2vpn evpn
  retain route-target all
  neighbor 10.4.1.0 activate
  neighbor 10.4.1.0 route-reflector-client
  neighbor 10.0.0.10 activate
  neighbor 10.0.0.11 activate
  neighbor 10.0.0.12 activate
  neighbor 10.0.0.13 activate
!

```

Sample BGP EVPN leaf configuration using iBGP

The following example is a sample BGP EVPN leaf configuration using iBGP in an IP Fabric. The leaf switch is configured to communicate with a neighbor through a specified loopback interface.

NOTE

This configuration sample demonstrates BGP EVPN configuration using iBGP on a leaf in an IP Fabric. For full configuration details for configuring an IP Fabric involving all the steps required, refer to [Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay](#) on page 53. For more details on the commands used in this sample, refer to the *Extreme Network OS Command Reference*.

```

evpn-instance default
  rd auto
  route-target both auto ignore-as
  vlan add 101-108
  vlan add 121-128
  vlan add 141-148
  vlan add 161-168
  vlan add 120
  vlan add 140
  vlan add 160
  vlan add 180
!
router bgp
  local-as 3
  neighbor 10.0.0.4 remote-as 3
  neighbor 10.0.0.4 update-source loopback 1
  neighbor 10.0.0.5 remote-as 3
  neighbor 10.0.0.5 update-source loopback 1
!
  address-family ipv4 unicast
    maximum-paths 8

```

```

    next-hop-recursion
    redistribute connected
    !
    address-family l2vpn evpn
    neighbor 10.0.0.4 activate
    neighbor 10.0.0.5 activate
    neighbor 10.0.0.4 next-hop-unchanged
    neighbor 10.0.0.5 next-hop-unchanged
    !
router ospf
    area 0
    redistribute connected
    !
overlay-gateway Leaf10
    type layer2-extension
    ip interface Loopback 1
    attach rbridge-id add 76
    map vlan vni auto
    activate
int te 76/0/1
    ip address 10.10.0.1/31
    ipv6 address 1000:4:10::1/127
    ip ospf area 0
    no shut
    !
int te 76/0/2
    ip address 10.10.0.2/31
    ipv6 address 1000:5:10::1/127
    ip ospf area 0
    no shut
    !
int te 76/0/16
    switchport
    switchport mode trunk
    switchport trunk allowed vlan add 101-108
    switchport trunk allowed vlan add 121-128
    no shut
    !
int te 76/0/17
    switchport trunk allowed vlan add 141-148
    switchport trunk allowed vlan add 161-168
    no shut
    !

```

Appendix C: Sample Topology Configuration Files

| | |
|-------------------|-----|
| • Leaf..... | 109 |
| • Spine..... | 114 |
| • SuperSpine..... | 115 |

Leaf

The following configuration file is Leaf1.cfg.

```
int vlan 101-108
int vlan 120
int vlan 121-128
int vlan 140
int vlan 141-148
int vlan 160
int vlan 161-168
int vlan 180
rbridge-id 1
  switch-attributes host-name Leaf1
ip anycast-gateway-mac 0000.ABBA.BABA
ipv6 anycast-gateway-mac 0000.ABBA.ABBA
vrf red
  rd 6.0.0.6:1
  vni 120
  address-family ipv4 unicast
    route-target import 2:2 evpn
    route-target export 2:2 evpn
  address-family ipv6 unicast
    route-target import 2:2 evpn
    route-target export 2:2 evpn
!
vrf blue
  rd 6.0.0.6:2
  vni 140
  address-family ipv4 unicast
    route-target import 2:2 evpn
    route-target export 2:2 evpn
  address-family ipv6 unicast
    route-target import 2:2 evpn
    route-target export 2:2 evpn
!
vrf green
  rd 6.0.0.6:3
  vni 160
  address-family ipv4 unicast
    route-target import 2:2 evpn
    route-target export 2:2 evpn
  address-family ipv6 unicast
    route-target import 2:2 evpn
    route-target export 2:2 evpn
!
vrf yellow
  rd 6.0.0.6:4
  vni 180
  address-family ipv4 unicast
    route-target import 2:2 evpn
    route-target export 2:2 evpn
  address-family ipv6 unicast
    route-target import 2:2 evpn
```

```

    route-target export 2:2 evpn
!
!!!!Loopback interface
int loopback 1
  ip address 6.0.0.6/32
  ipv6 address 1000:6::6/128
  no shut
!
int ve 101
  vrf forwarding red
  ip anycast-address 101.1.1.254/24
  ipv6 anycast-address 101:1:1::254/64
  no shutdown
!
int ve 102
  vrf forwarding red
  ip anycast-address 102.1.1.254/24
  ipv6 anycast-address 102:1:1::254/64
  no shutdown
!
int ve 103
  vrf forwarding red
  ip anycast-address 103.1.1.254/24
  ipv6 anycast-address 103:1:1::254/64
  no shutdown
!
int ve 104
  vrf forwarding red
  ip anycast-address 104.1.1.254/24
  ipv6 anycast-address 104:1:1::254/64
  no shutdown
!
int ve 105
  vrf forwarding red
  ip anycast-address 105.1.1.254/24
  ipv6 anycast-address 105:1:1::254/64
  no shutdown
!
int ve 106
  vrf forwarding red
  ip anycast-address 106.1.1.254/24
  ipv6 anycast-address 106:1:1::254/64
  no shutdown
!
int ve 107
  vrf forwarding red
  ip anycast-address 107.1.1.254/24
  ipv6 anycast-address 107:1:1::254/64
  no shutdown
!
int ve 108
  vrf forwarding red
  ip anycast-address 108.1.1.254/24
  ipv6 anycast-address 108:1:1::254/64
  no shutdown
!
int ve 121
  vrf forwarding blue
  ip anycast-address 121.1.1.254/24
  ipv6 anycast-address 121:1:1::254/64
  no shutdown
!
int ve 122
  vrf forwarding blue
  ip anycast-address 122.1.1.254/24
  ipv6 anycast-address 122:1:1::254/64
  no shutdown
!
int ve 123
  vrf forwarding blue
  ip anycast-address 123.1.1.254/24
  ipv6 anycast-address 123:1:1::254/64

```

```

    no shutdown
    !
int ve 124
    vrf forwarding blue
    ip anycast-address 124.1.1.254/24
    ipv6 anycast-address 124:1:1::254/64
    no shutdown
    !
int ve 125
    vrf forwarding blue
    ip anycast-address 125.1.1.254/24
    ipv6 anycast-address 125:1:1::254/64
    no shutdown
    !
int ve 126
    vrf forwarding blue
    ip anycast-address 126.1.1.254/24
    ipv6 anycast-address 126:1:1::254/64
    no shutdown
    !
int ve 127
    vrf forwarding blue
    ip anycast-address 127.1.1.254/24
    ipv6 anycast-address 127:1:1::254/64
    no shutdown
    !
int ve 128
    vrf forwarding blue
    ip anycast-address 128.1.1.254/24
    ipv6 anycast-address 128:1:1::254/64
    no shutdown
    !
int ve 141
    vrf forwarding green
    ip anycast-address 141.1.1.254/24
    ipv6 anycast-address 141:1:1::254/64
    no shutdown
    !
int ve 142
    vrf forwarding green
    ip anycast-address 142.1.1.254/24
    ipv6 anycast-address 142:1:1::254/64
    no shutdown
    !
int ve 143
    vrf forwarding green
    ip anycast-address 143.1.1.254/24
    ipv6 anycast-address 143:1:1::254/64
    no shutdown
    !
int ve 144
    vrf forwarding green
    ip anycast-address 144.1.1.254/24
    ipv6 anycast-address 144:1:1::254/64
    no shutdown
    !
int ve 145
    vrf forwarding green
    ip anycast-address 145.1.1.254/24
    ipv6 anycast-address 145:1:1::254/64
    no shutdown
    !
int ve 146
    vrf forwarding green
    ip anycast-address 146.1.1.254/24
    ipv6 anycast-address 146:1:1::254/64
    no shutdown
    !
int ve 147
    vrf forwarding green
    ip anycast-address 147.1.1.254/24
    ipv6 anycast-address 147:1:1::254/64

```

```

    no shutdown
    !
int ve 148
    vrf forwarding green
    ip anycast-address 148.1.1.254/24
    ipv6 anycast-address 148:1:1::254/64
    no shutdown
    !
int ve 161
    vrf forwarding yellow
    ip anycast-address 161.1.1.254/24
    ipv6 anycast-address 161:1:1::254/64
    no shutdown
    !
int ve 162
    vrf forwarding yellow
    ip anycast-address 162.1.1.254/24
    ipv6 anycast-address 162:1:1::254/64
    no shutdown
    !
int ve 163
    vrf forwarding yellow
    ip anycast-address 163.1.1.254/24
    ipv6 anycast-address 163:1:1::254/64
    no shutdown
    !
int ve 164
    vrf forwarding yellow
    ip anycast-address 164.1.1.254/24
    ipv6 anycast-address 164:1:1::254/64
    no shutdown
    !
int ve 165
    vrf forwarding yellow
    ip anycast-address 165.1.1.254/24
    ipv6 anycast-address 165:1:1::254/64
    no shutdown
    !
int ve 166
    vrf forwarding yellow
    ip anycast-address 166.1.1.254/24
    ipv6 anycast-address 166:1:1::254/64
    no shutdown
    !
int ve 167
    vrf forwarding yellow
    ip anycast-address 167.1.1.254/24
    ipv6 anycast-address 167:1:1::254/64
    no shutdown
    !
int ve 168
    vrf forwarding yellow
    ip anycast-address 168.1.1.254/24
    ipv6 anycast-address 168:1:1::254/64
    no shutdown
    !
int ve 120
    vrf forwarding red
    ipv6 address use-link-local-only
    no shut
    !
int ve 140
    vrf forwarding blue
    ipv6 address use-link-local-only
    no shut
    !
int ve 160
    vrf forwarding green
    ipv6 address use-link-local-only
    no shut
    !
int ve 180

```



```

vrf forwarding yellow
    ipv6 address use-link-local-only
no shut
!
evpn-instance myinstance
rd auto
route-target both auto ignore-as
vlan add 101-108
vlan add 121-128
vlan add 141-148
vlan add 161-168
vlan add 120
vlan add 140
vlan add 160
vlan add 180
!
router bgp
local-as 3
neighbor 2.6.0.0 remote-as 2
neighbor 1000:2:6:: remote-as 2
neighbor 3.6.0.0 remote-as 2
neighbor 1000:3:6:: remote-as 2
!
address-family ipv4 unicast
maximum-paths 8
next-hop-recursion
redistribute connected
neighbor 2.6.0.0 allowas-in 1
neighbor 3.6.0.0 allowas-in 1
!
address-family ipv6 unicast
maximum-paths 8
next-hop-recursion
redistribute connected
neighbor 1000:2:6:: activate
neighbor 1000:3:6:: activate
neighbor 1000:2:6:: allowas-in 1
neighbor 1000:3:6:: allowas-in 1
!
address-family l2vpn evpn
neighbor 2.6.0.0 activate
neighbor 3.6.0.0 activate
neighbor 2.6.0.0 allowas-in 1
neighbor 3.6.0.0 allowas-in 1
neighbor 2.6.0.0 next-hop-unchanged
neighbor 3.6.0.0 next-hop-unchanged
!
address-family ipv4 unicast vrf red
redistribute connected
address-family ipv4 unicast vrf blue
redistribute connected
address-family ipv4 unicast vrf green
redistribute connected
address-family ipv4 unicast vrf yellow
redistribute connected
address-family ipv6 unicast vrf red
redistribute connected
address-family ipv6 unicast vrf blue
redistribute connected
address-family ipv6 unicast vrf green
redistribute connected
address-family ipv6 unicast vrf yellow
redistribute connected
overlay-gateway Leaf6
type layer2-extension
ip interface Loopback 1
attach rbridge-id add 1
map vlan vni auto
activate
int te 1/0/1
ip address 2.6.0.1/31
ipv6 address 1000:2:6::1/127

```

```

    no shut
    !
int te 1/0/5
    ip address 3.6.0.1/31
    ipv6 address 1000:3:6::1/127
    no shut
    !
int te 1/0/16
    switchport
    switchport mode trunk
    switchport trunk allowed vlan add 101-108
    switchport trunk allowed vlan add 121-128
    no shut
    !
int te 1/0/17
    switchport
    switchport mode trunk
    switchport trunk allowed vlan add 141-148
    switchport trunk allowed vlan add 161-168
    no shut
    !

```

Spine

The following configuration file is Spine1.cfg.

```

rbridge-id 1
    switch-attributes host-name Spine1
    !!!!!Loopback interface
int loopback 1
    ip address 2.0.0.2/32
    ipv6 address 1000:2::2/128
    no shut
    !
router bgp
    local-as 2
!BGP Sessions to the SuperSpine
    neighbor 1.2.1.0 remote-as 1
    neighbor 1000:1:2:1:: remote-as 1
!IPv4/IPv6 BGP Sessions to the leaf switches
    neighbor 2.6.0.1 remote-as 3
    neighbor 1000:2:6::1 remote-as 3
    neighbor 2.7.0.1 remote-as 3
    neighbor 1000:2:7::1 remote-as 3
    neighbor 2.8.0.1 remote-as 3
    neighbor 1000:2:8::1 remote-as 3
    neighbor 2.9.0.1 remote-as 3
    neighbor 1000:2:9::1 remote-as 3
address-family ipv4 unicast
    maximum-paths 8
    next-hop-recursion
    redistribute connected
    neighbor 1.2.1.0 allowas-in 1
address-family ipv6 unicast
    maximum-paths 8
    next-hop-recursion
    redistribute connected
    neighbor 1000:1:2:1:: activate
    neighbor 1000:1:2:1:: allowas-in 1
    neighbor 1000:2:6::1 activate
    neighbor 1000:2:7::1 activate
    neighbor 1000:2:8::1 activate
    neighbor 1000:2:9::1 activate
address-family l2vpn evpn
    retain route-target all
    neighbor 1.2.1.0 activate
    neighbor 1.2.1.0 allowas-in 1
    neighbor 1.2.1.0 next-hop-unchanged

```

```

neighbor 2.6.0.1 activate
neighbor 2.6.0.1 next-hop-unchanged
neighbor 2.7.0.1 activate
neighbor 2.7.0.1 next-hop-unchanged
neighbor 2.8.0.1 activate
neighbor 2.8.0.1 next-hop-unchanged
neighbor 2.9.0.1 activate
neighbor 2.9.0.1 next-hop-unchanged
!
int te 1/1/33
ip address 1.2.1.1/31
ipv6 address 1000:1:2:1::1/127
no shutdown
!
int te 1/1/34
ip address 1.2.2.1/31
ipv6 address 1000:1:2:2::1/127
no shutdown
!
int te 1/1/35
ip address 1.2.3.1/31
ipv6 address 1000:1:2:3::1/127
no shutdown
!
int te 1/1/36
ip address 1.2.4.1/31
ipv6 address 1000:1:2:4::1/127
no shutdown
!
int te 1/1/37
ip address 1.2.5.1/31
ipv6 address 1000:1:2:5::1/127
no shutdown
!
int te 1/1/1
ip address 2.6.0.0/31
ipv6 address 1000:2:6::0/127
no shutdown
!
int te 1/1/4
ip address 2.7.0.0/31
ipv6 address 1000:2:7::0/127
no shutdown
!
int te 1/1/3
ip address 2.8.0.0/31
ipv6 address 1000:2:8::0/127
no shutdown
!
int te 1/1/2
ip address 2.9.0.0/31
ipv6 address 1000:2:9::0/127
no shutdown
!

```

SuperSpine

The following configuration file is SuperSpine.cfg.

```

rbridge-id 1
switch-attributes host-name SuperSpine
!!!!Loopback interface
int loopback 1
ip address 1.0.0.1/32
ipv6 address 1000:1::1/128
no shut
!
router bgp

```

```

local-as 1
! IPv4/IPv6 BGP Sessions to Spine switches
neighbor 1.2.1.1 remote-as 2
neighbor 1.3.1.1 remote-as 2
neighbor 1000:1:2:1::1 remote-as 2
neighbor 1000:1:3:1::1 remote-as 2
address-family ipv4 unicast
maximum-paths 8
next-hop-recursion
redistribute connected
address-family ipv6 unicast
maximum-paths 8
next-hop-recursion
redistribute connected
neighbor 1000:1:2:1::1 activate
neighbor 1000:1:3:1::1 activate
address-family l2vpn evpn
retain route-target all
! Activate EVPN Session to Spine switches
neighbor 1.2.1.1 activate
neighbor 1.3.1.1 activate
neighbor 1.2.1.1 next-hop-unchanged
neighbor 1.3.1.1 next-hop-unchanged
int te 1/1/1
ip address 1.2.1.0/31
ipv6 address 1000:1:2:1::/127
no shutdown
!
int te 1/5/1
ip address 1.3.1.0/31
ipv6 address 1000:1:3:1::/127
no shutdown
!

```