Solution Guide Software as a Service

Copyright © 2016, Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

**Chapter 1**

**Software as a Service** .................................................. 5

About This Solution Guide ................................................. 5
Understanding the Software as a Service Solution ................. 5
  Overview ............................................................................. 5
  Requirements .................................................................... 8
  Design Considerations ..................................................... 12
  Implementation ............................................................... 14

Example: Configuring the Software as a Service Solution ......... 15
Configuring an IP Fabric using Junos Space Network Director or OpenClos .... 88
  Using Network Director to Provision Leaf and Spine Devices .......... 90
  Using OpenClos to Provision Leaf and Spine Devices ................. 95
CHAPTER 1

Software as a Service

- About This Solution Guide on page 5
- Understanding the Software as a Service Solution on page 5
- Example: Configuring the Software as a Service Solution on page 15
- Configuring an IP Fabric using Junos Space Network Director or OpenClos on page 88

About This Solution Guide

The Software as a Service (SaaS) solution is an innovative architecture that is built for the cloud era. An SaaS network offers Web Services providers a way to grow their network operations exponentially to support multiple services running on as many as 120,000 servers. By using a Clos-based Layer 3 IP fabric infrastructure that provides resiliency and scale in a cost-efficient manner, Web Services providers can support their customers, scale their cloud-enabled business, and manage costs all at the same time.

This guide provides an overview of the SaaS solution, the solution requirements, design considerations, and how the solution was implemented by the Juniper Networks solutions team. It also provides an example of how to configure an SaaS network and verify that the solution is working as expected.

Understanding the Software as a Service Solution

- Overview on page 5
- Requirements on page 8
- Design Considerations on page 12
- Implementation on page 14

Overview

Web Services providers typically offer large-scale distributed Web applications, consumer applications, Software as a Service (SaaS), and Infrastructure as a Service (IaaS) to their customers. To offer these services effectively, these providers need to be mindful of ways to keep the network costs under control. Automation, programmability, and zero touch deployments maximize the speed of offering new services, minimize staffing levels, and support employees’ software expertise with Python, Ruby, and other programming languages in a Linux operational environment.
The solution described in this guide will help you understand the requirements for an SaaS network, the architecture required to build the network, how to configure each layer, and how to verify its operational state. Because of the cutting-edge nature of the SaaS solution, this architecture will also appeal to system integrators, infrastructure vendors, Web hosting companies, and multisystem operators (MSOs). The solution was built using the following key component features:

- **BGP / Layer 3 routing**—An SaaS network needs Layer 3 and specifically BGP to provide the proper scaling and performance required by this solution. Networks that rely on Layer 2 switching are restricted based on the number of VLANs that can be supported by the equipment in the network. However, BGP was designed to handle the scale of the global Internet routing table and can be repurposed to support the needs of a top-tier cloud provider. As shown in Figure 1 on page 6, you assign an autonomous system number to each device in the Clos-based IP fabric network. The devices peer with each other using external BGP and allow you to grow and scale your SaaS network efficiently.

  ![Figure 1: BGP Autonomous Systems in a Clos-Based IP Fabric](image1)

- **5-Layer IP fabric**—An SaaS network uses a robust set of layers to direct traffic and to offer multiple paths to the same destination. Instead of high availability options such as multichassis link aggregation groups (MC-LAGs) or Virtual Chassis, an SaaS solution uses multiple links at each layer and device. Such an architecture shifts some of the needs for resiliency to distributed applications in the network, and enhances the amount of resiliency and redundancy provided by the physical network infrastructure itself. As shown in Figure 2 on page 6, the five IP fabric layers include a core routing layer, a fabric layer, a spine layer, a leaf layer, and a compute layer.

  ![Figure 2: 5-Layer IP Fabric](image2)
When deployed, the five layers look like the network shown in Figure 3 on page 7. The core routing layer connects into a Data Center Interconnect (DCI), a virtual private network (VPN), or both to carry data center traffic from the fabric layer securely to other Web Services provider locations or data centers. The fabric layer joins the core routing layer to the spine layer, aggregates traffic from the spine layer, and provides connectivity between point of delivery (POD) modules that contain spine and leaf layer devices. The spine layer sits between the fabric and leaf layers to aggregate traffic and provide intra-POD connectivity, and the leaf layer connects servers to the rest of the IP fabric. The multiple paths available from each device at each layer provide robust connectivity and no single point of failure. Additionally, the fabric layer can connect to multiple PODs containing a spine and leaf layer.

**Figure 3: SaaS Solution - Clos-Based IP Fabric Topology**

- **Multiple server deployment options**—As shown in Figure 4 on page 8, the SaaS solution offers three choices for server connectivity:
  - **Unicast**—The servers connect to the leaf layer at Layer 2 through a server VLAN, and the server VLAN is routed at Layer 3 through an IRB interface on the leaf layer devices. The spine layer receives the aggregated prefixes from the leaf layer IRB interfaces.
  - **Anycast**—The servers run BGP natively and connect to the leaf layer devices at Layer 3. The servers share their address prefixes directly with the leaf layer devices, and the spine layer devices receive routes from the leaf layer devices. By configuring a single anycast IP address on each server, the Clos-based network offers multiple ways for the traffic to flow, multiple servers that can process the requests when they arrive, and multiple services that can be handled simultaneously.
  - **Hybrid**—This option offers a blend of the unicast and anycast choices by including an IRB interface at the leaf layer. BGP runs natively on the servers, and the leaf layer
devices share both the IP fabric routes and the IRB prefixes with the spine layer devices.

Figure 4: Server Models for the SaaS Solution

- **Automation**—The SaaS network configuration shown in this solution guide was partially created using OpenClos Python scripts. OpenClos is an open-source method to dynamically generate device configurations and is embedded within Network Director. With the SaaS solution contained in this guide, you can use either Network Director or native Python scripts to build an initial configuration. The solution also offers Junos® OS support for other programming languages, zero touch provisioning (ZTP), and REST APIs. Such tools allow the SaaS network to be built and supported effectively and efficiently by a lean operational staff.

**Requirements**

Web Services companies interested in deploying an SaaS solution have certain requirements that must be met in order to make the solution viable for their needs. The main categories into which these requirements fall include device capabilities, network services, automation, monitoring, class of service, security, and performance and scaling. Table 1 on page 9 explores these services and requirements.
Table 1: SaaS Solution Descriptions and Requirements

<table>
<thead>
<tr>
<th>Type</th>
<th>Components</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network devices</strong></td>
<td>The physical hardware required for networking infrastructure in the SaaS solution includes MX Series routers and QFX Series switches as follows:</td>
<td>• Switches deployed at the fabric, spine, and leaf layers must be able to boot in 10 seconds and reboot in 30 seconds.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Core routing layer</strong>—You can select any of the MX Series routers that meet your needs (MX80, MX104, MX240, MX480, MX960, MX2010, or MX2020). The MX480 router was used in the testing of this solution.</td>
<td>• Software can be upgraded in 90 seconds without in-service software upgrade (ISSU).</td>
</tr>
<tr>
<td></td>
<td>• <strong>Fabric layer</strong>—You can select any of the currently supported QFX Series switches based on your budget and scaling needs (QFX5100 or QFX10002). The QFX5100-24Q switch was used in the testing of this solution.</td>
<td>• Switches must support 10-Gigabit Ethernet and 40-Gigabit Ethernet third-party optical transceivers.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Spine layer</strong>—Both the QFX10002-72Q and QFX5100-24Q switches were used in the testing of this solution, but we recommend the QFX10002-72Q switches at this layer for their expanded capabilities, port density, and processing power.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Leaf layer</strong>—The QFX5100-48S, QFX5100-48T, and OCX1100 switches were used in the testing of this solution as leaf devices.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 1: SaaS Solution Descriptions and Requirements (continued)

<table>
<thead>
<tr>
<th>Type</th>
<th>Components</th>
<th>Requirements</th>
</tr>
</thead>
</table>
| **Network services**  | • IP fabric—The SaaS solution must be able to support up to a 5-stage IP fabric. Such fabrics take advantage of Layer 3 and BGP to support native cloud applications and services. | **BGP**<br>• All switches in the IP fabric must use external BGP with 32-bit autonomous system (AS) numbers.  
• BGP must be the only routing protocol used in the IP fabric.  
• Servers must run BGP, and up to 48 servers can have the same anycast address.  
• Access switches must automatically accept trusted BGP connections from new servers by default and without additional configuration.  
• BGP anycast must be supported across points of delivery (PODs). |
|                       | • Multiple traffic models—The network must support unicast, anycast, and hybrid (both unicast and anycast) traffic. This allows you to choose the traffic model that works best for your network. |                                                                    |
|                       | • IPv4 and IPv6 addressing—To allow for expansion and growth, the solution requires support for both IPv4 and IPv6. |                                                                    |
|                       | • Multispeed interfaces—As your network evolves, you need to migrate servers and access interfaces to higher speeds. To do this, the leaf layer must support Fast Ethernet (100 Mbps), Gigabit Ethernet (1 Gbps), and 10-Gigabit Ethernet (10 Gbps) using either copper or fiber cabling to connect to servers, and 40-Gigabit Ethernet (40 Gbps) uplink interfaces to connect to the rest of the IP fabric. |                                                                    |
|                       | • Traffic management—To provide maximum resiliency and redundancy, the solution provides variable oversubscription, 64-way equal-cost multipath (ECMP) routing, and user-defined ECMP, resilient hashing, and traffic profiles. |                                                                    |
| **5-stage IP fabric** |                                                                 | **Advertisements, connectivity, and filtering**<br>• The solution must use a 5-stage IP fabric topology.  
• Leaf switches in a 5-stage topology must aggregate all access switch IRB interfaces to the spine layer and support a minimum of 48-way ECMP routing.  
• The IP fabric must be configured to support a dual stack with IPv4 and IPv6 addresses. |
|                       |                                                                 | **Other protocols**<br>• Bidirectional Forwarding Detection (BFD) must be configured on all PTP interfaces with 250ms intervals and a multiplier of 3.  
• All switches must support LLDP to identify remote devices and ports.  
• The network must support jumbo frames up to 9000 bytes end-to-end in the IP fabric.  
• The network must support unicast, anycast, and hybrid traffic flows. |
### Table 1: SaaS Solution Descriptions and Requirements (continued)

<table>
<thead>
<tr>
<th>Type</th>
<th>Components</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Automation</strong></td>
<td>• <strong>Zero touch provisioning (ZTP)</strong>—When you physically connect a switch to the network and boot it with a default configuration, ZTP attempts to upgrade the Junos OS software automatically and autoinstall a configuration file from the network. This allows your network to be provisioned quickly over Layer 3 using revenue ports.</td>
<td>• The IP fabric must be generated using OpenClos and ZTP.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Indefinite retries</strong>—This enables the network to keep trying to establish connectivity until all required components are added to the network.</td>
<td>• Each switch must support the ability to execute Python scripts.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Programming language support</strong>—Junos OS software offers native programming languages, such as Python, to enable you to automate common operational and configuration tasks in your network.</td>
<td>• Trusted servers must use BGP to peer automatically with the leaf switches, but without requiring manual configuration changes on the leaf switches when new servers are connected.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OpenClos</strong>—A set of Python scripts, created by Juniper Networks staff and the open-source scripting community, that are designed to autoprovion an IP fabric. Many of the OpenClos scripts are also integrated with Network Director.</td>
<td>• The solution must be able to use the REST API to make Packet Forwarding Engine-level and configuration-level queries and changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NOTE:</strong> The SaaS solution can implement OpenClos either through the capabilities built into Network Director or through the native OpenClos Python scripts. For more information on OpenClos, see “Configuring an IP Fabric using Junos Space Network Director or OpenClos” on page 88.</td>
</tr>
<tr>
<td><strong>Monitoring and management</strong></td>
<td>• <strong>Port mirroring</strong>—Allows a switch to send copies of packets to either a local interface for local monitoring or to a remote server for remote monitoring.</td>
<td>• The solution must have the ability to mirror traffic locally on a switch, or use GRE encapsulation to send the mirrored traffic to a remote Linux server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Monitoring for all BGP paths</strong>—This includes BGP Monitoring Protocol version 3 (BMPv3), time-stamping, and peer down notifications.</td>
<td>• Each switch must be configured with BMPv3 and send information to a centralized server for BGP reporting. (We recommend using the BMP process available here: <a href="https://github.com/garberg/bmpd">https://github.com/garberg/bmpd</a>)</td>
</tr>
<tr>
<td></td>
<td>• <strong>SNMPv3</strong>—Provides support for Simple Network Management Protocol, which is used widely in the networking industry.</td>
<td>• Each switch must be configured with SNMPv3 and use the authentication and encryption options.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Remote logging</strong>—Offers system log message support, firewall filters, and control plane/data plane thresholds.</td>
<td>• All SNMP data must be exported to a centralized server for reporting purposes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Digital optical monitoring (DOM)</strong>—Provides support for temperature and power thresholds and reporting.</td>
<td>• DOM must be part of the information captured by SNMPv3, including detection and reporting of unacceptable power levels and temperatures.</td>
</tr>
</tbody>
</table>
Table 1: SaaS Solution Descriptions and Requirements (continued)

<table>
<thead>
<tr>
<th>Type</th>
<th>Components</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>• Control plane security—Must include the ability to use firewall filters and policers, and discard and log traffic.</td>
<td>• Each switch must be configured to deny all IP traffic destined to the control plane that is not essential to the operation of the switch.</td>
</tr>
<tr>
<td></td>
<td>• Storm control—Must be provided for broadcast, unknown unicast, and multicast (BUM) traffic.</td>
<td>• Any denied control plane traffic must be counted and logged to a remote server.</td>
</tr>
<tr>
<td></td>
<td>• Firewall filter support—Must be able to filter on IPv4, IPv6, Layer 2 fields, and Layer 3 fields.</td>
<td>• SSH traffic sent to the switch must be policed at a rate of 5 Mbps.</td>
</tr>
<tr>
<td></td>
<td>• Storm control must be configured on every leaf layer access switch and triggered at 50% of bandwidth capacity.</td>
<td>• Storm control must be configured on every leaf layer access switch and triggered at 50% of bandwidth capacity.</td>
</tr>
<tr>
<td></td>
<td>• Leaf layer access switches must support the following to allow and block traffic:</td>
<td>• Leaf layer access switches must support the following to allow and block traffic:</td>
</tr>
<tr>
<td></td>
<td>• IPv4 and IPv6 port-based firewall filters</td>
<td>• IPv4 and IPv6 port-based firewall filters</td>
</tr>
<tr>
<td></td>
<td>• IPv4 and IPv6 VLAN-based firewall filters</td>
<td>• IPv4 and IPv6 VLAN-based firewall filters</td>
</tr>
<tr>
<td></td>
<td>• IPv4 and IPv6 IRB-based firewall filters</td>
<td>• IPv4 and IPv6 IRB-based firewall filters</td>
</tr>
</tbody>
</table>

The SaaS solution also has the following performance and scale requirements:

- 3:1 oversubscription within a POD (48x10G downstream and 4x40G upstream)
- 32 BFD sessions with a 250ms interval and a multiplier of 3 (32x40GE spine layer)
- 52 BGP sessions on each leaf layer device (48 servers + 4 uplinks)
- 48-way ECMP in a leaf layer device (48 servers + anycast)
- 4-way 802.3ad on a leaf layer device
- Routing table and forwarding table requirements
  - 2048 IPv4 or IPv6 loopback addresses
  - 1024 IPv4 or IPv6 PTP networks
  - 32 IPv4 or IPv6 IRB networks

Design Considerations

There are two primary design concerns when implementing an SaaS network:

- **IBGP or EBGP Clos-based IP fabric**—The first decision to make in an SaaS environment is whether to use IBGP or EBGP. The very nature of an IP fabric requires having multiple, equal-cost paths. The design must consider how IBGP and EBGP handle the equal-cost multipath (ECMP) feature. By default, EBGP supports ECMP without enabling additional features. Conversely, IBGP requires the use of a BGP route reflector and the AddPath feature to fully support ECMP.

  EBGP offers a simpler and more elegant way to design an IP fabric. EBGP also facilitates traffic engineering by using local preference and autonomous system padding techniques. As shown in Figure 5 on page 13, each device in the IP fabric uses a different
autonomous system (AS) number, and each leaf device must peer with every spine device in the IP fabric.

Figure 5: AS Number Assignment in an IP Fabric

Designing IBGP in an IP fabric is a bit more complicated because IBGP requires that all devices must peer with every other device. To make the peering requirements easier, you can use inline BGP route reflectors in the spine layer of the network. However, standard BGP route reflection only reflects the best prefix and does not work well with ECMP. In order to enable full ECMP, you need to configure the BGP AddPath feature, which provides additional ECMP paths into the BGP advertisements between the route reflector and the clients.

NOTE: For more information about the BGP AddPath feature, see Understanding the Advertisement of Multiple Paths to a Single Destination in BGP.

Because EBGP supports ECMP in a more straightforward fashion and IBGP is more complicated, this solution guide focuses on the configuration and validation of an EBGP-based IP fabric.

- **Server reachability options**—In an EBGP-based IP fabric, there are three ways to connect from the leaf layer to the servers:
  - **Anycast**—The server-to-leaf connection is pure Layer 3, where you configure BGP on the physical interface of the leaf layer device.
  - **Hybrid**—The server-to-leaf connection happens at Layer 2 through use of VLANs. The BGP session is established between the integrated routing and bridging (IRB) interface and the Layer 3 connection at the server.
  - **Unicast**—The server-to-leaf connection happens at Layer 2 through the use of VLANs with the leaf layer device IRB interface used as a default gateway. You do not need to configure BGP between the servers and leaf layer devices.

Because the requirements for most Web Services providers focus on BGP and Layer 3 running at the server layer, this solution guide focuses on the configuration and validation of anycast in the IP fabric. However, for completeness sake, we also show the unicast and hybrid options as well.
Implementation

The following hardware equipment and software features were used to create the SaaS solution described in the example:

Core Routing

- The two core routers are MX480 routers.
- Configure both Router R1 and Router R2 as MPLS provider edge (PE) routers to connect upstream to the provider core.
- Configure all fabric layer-facing interfaces as customer edge (CE) links inside a VRF routing instance.
- Configure EBGP with 2-byte autonomous system (AS) numbers as the PE-to-CE routing protocol to advertise the anycast routes to the core.

Fabric

- There are four QFX5100-24Q switches in the fabric layer.
- Use EBGP peering with 4-byte AS numbers to reach the upstream routers through a routing instance.
- Use EBGP peering with 4-byte AS numbers to connect with the downstream spine devices.
- Anycast routes received from the spine devices are advertised to the routers by way of EBGP.
- Configure EBGP multipath and per-packet load balancing on all devices.
- Enable resilient hashing.
- Configure Bidirectional Forwarding Detection (BFD) for all BGP sessions.

Spine

- There are four QFX10002-72Q switches in the spine layer.
- Use EBGP peering with 4-byte AS numbers to connect with both the upstream fabric devices and the downstream leaf devices.
- Anycast routes received from the leaf devices are advertised to the fabric devices by way of EBGP.
- Configure EBGP multipath and per-packet load balancing on all devices.
- Enable resilient hashing.
- Configure BFD for all BGP sessions.

Leaf
There are two QFX5100-48S, two QFX5100-48T, and two OCX1100 switches in the leaf layer to provide flexibility for fiber and copper cabled networks.

Use EBGP peering with 4-byte AS numbers to connect with both the upstream spine devices and the downstream servers.

Anycast routes received from the servers are advertised to the spine devices by way of EBGP.

Configure EBGP multipath and per-packet load balancing on all devices.

Enable resilient hashing.

Configure BFD for all BGP sessions.

**Compute**

Configure IBM Flex blade servers with the VMware ESXi operating system.

A traffic generator was used to simulate the BGP sessions to the leaf devices, as well as server-based traffic.

Now that we have completed our overview of the SaaS solution, it is time to view the configuration and verification sections of the solution.

**Example: Configuring the Software as a Service Solution**

This example describes how to build large IP fabrics using the Juniper Networks QFX10002 and QFX5100 line of switches.

- Requirements on page 15
- Overview and Topology on page 16
- Configuring an IP Fabric for the Software as a Service Solution on page 19
- Configuring Additional Features for the SaaS Solution on page 45
- Verification on page 54

**Requirements**

Table 2 on page 15 lists the hardware and software components used in this example.

**Table 2: Solution Hardware and Software Requirements**

<table>
<thead>
<tr>
<th>Device</th>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core routers</td>
<td>MX480</td>
<td>Junos OS Release 14.2R2.8</td>
</tr>
<tr>
<td>Fabric devices</td>
<td>QFX5100-24Q</td>
<td>Junos OS Release 14.1X53-D35.3</td>
</tr>
<tr>
<td>Spine devices</td>
<td>QFX10002-72Q*</td>
<td>15.1X53-D32.2</td>
</tr>
</tbody>
</table>
Table 2: Solution Hardware and Software Requirements *(continued)*

<table>
<thead>
<tr>
<th>Device</th>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaf devices</td>
<td>QFX5100-48S, QFX5100-48T, and OCX1100**</td>
<td>Junos OS Release 14.1X53-D35.3</td>
</tr>
<tr>
<td>Servers</td>
<td>IBM Flex and IBMx3750</td>
<td>VMware ESXi 5.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vCenter 5.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Junos Space Release 15.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Network Director Release 2.5</td>
</tr>
</tbody>
</table>

* This solution has also been validated using QFX5100-24Q switches at the spine layer.

** The Juniper Networks OCX1100 switch is an open networking switch based on hardware specifications ratified by the Open Compute Project (OCP) Foundation.

Overview and Topology

The topology used in this example consists of a series of QFX10002, QFX5100, and OCX1100 devices, and two MX480 devices, as shown in Figure 6 on page 17.
In this example, the leaf layer uses a combination of four QFX5100-48S, QFX5100-48T and OCX1100 switches. The spine layer uses four QFX10002-72Q switches, and the fabric layer uses four QFX5100-24Q switches. The core layer uses two MX480 routers. A series of servers are attached to the leaf layer to serve as typical data center end hosts.

Table 3 on page 17 and Table 4 on page 18 list the IP addressing used in this example.

**Table 3: IPv4 Addressing**

<table>
<thead>
<tr>
<th>IPv4 Network Subnets</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server to leaf links</td>
<td>172.16.0.0/16</td>
</tr>
<tr>
<td>Leaf to spine links</td>
<td>192.168.11.0/24</td>
</tr>
</tbody>
</table>
Table 3: IPv4 Addressing (continued)

<table>
<thead>
<tr>
<th>IPv4 Network Subnets</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spine to fabric links</td>
<td>192.168.13.0/24</td>
</tr>
<tr>
<td>Fabric to core links</td>
<td>192.168.14.0/24</td>
</tr>
<tr>
<td>Loopback IPs (for all devices)</td>
<td>10.0.0.0/16</td>
</tr>
<tr>
<td>Anycast IP address</td>
<td>10.1.1.1/24</td>
</tr>
</tbody>
</table>

Table 4: IPv6 Addressing

<table>
<thead>
<tr>
<th>IPv6 Network Subnets</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anycast IP address</td>
<td>2001:db8:2000::/64</td>
</tr>
</tbody>
</table>

Table 5 on page 18 lists the AS numbering used in this example.

Table 5: BGP AS Numbering

<table>
<thead>
<tr>
<th>Device</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaf-0</td>
<td>420006000</td>
</tr>
<tr>
<td>Leaf-1</td>
<td>420006001</td>
</tr>
<tr>
<td>Leaf-2</td>
<td>420006002</td>
</tr>
<tr>
<td>Leaf-3</td>
<td>420006003</td>
</tr>
<tr>
<td>Spine-0</td>
<td>420005000</td>
</tr>
<tr>
<td>Spine-1</td>
<td>420005001</td>
</tr>
<tr>
<td>Spine-2</td>
<td>420005002</td>
</tr>
<tr>
<td>Spine-3</td>
<td>420005003</td>
</tr>
<tr>
<td>Fabric-0</td>
<td>420005501</td>
</tr>
<tr>
<td>Fabric-1</td>
<td>420005502</td>
</tr>
</tbody>
</table>
Table 5: BGP AS Numbering (continued)

<table>
<thead>
<tr>
<th>Device</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fabric-2</td>
<td>420005503</td>
</tr>
<tr>
<td>Fabric-3</td>
<td>420005504</td>
</tr>
<tr>
<td>Core-1 RED-vpn routing instance</td>
<td>420006501</td>
</tr>
<tr>
<td>Core-2 RED-vpn routing instance</td>
<td>420006502</td>
</tr>
<tr>
<td>Core-1</td>
<td>65000</td>
</tr>
<tr>
<td>Core-2</td>
<td>65000</td>
</tr>
</tbody>
</table>

Configuring an IP Fabric for the Software as a Service Solution

This example explains how to build out the leaf, spine, fabric, and core layers of an IP fabric for the Software as a Service (SaaS) solution. It includes the following sections:

- Configuring Leaf Devices for the IP Fabric on page 19
- Configuring Additional Leaf Device Elements on page 23
- Configuring Server Access Options on Leaf Devices on page 25
- Configuring Server Load Balancing Using Anycast on page 28
- Configuring Spine Devices for the IP Fabric on page 31
- Configuring Additional Spine Device Elements on page 33
- Configuring Fabric Devices on page 36
- Configuring Core Routers on page 40

Configuring Leaf Devices for the IP Fabric

Juniper Networks provides tools to help automate the creation of spine-and-leaf IP fabrics for SaaS environments. This solution includes two options to help with IP fabric creation: OpenClos and Junos Space Network Director.

OpenClos is a Python script library that enables you automate the design, deployment, and maintenance of a Layer 3 fabric built on BGP. To create an IP fabric that uses a spine-and-leaf architecture, the script generates device configuration files and uses zero touch provisioning (ZTP) to push the configuration files to the devices.

OpenClos functionality has also been built into Network Director 2.0 (and later), which allows you to provision spine-and-leaf Layer 3 fabrics using a GUI-based wizard.

For this example, the main configuration elements for the leaf devices were created using Network Director. For more information on using Network Director or OpenClos for this solution, see “Configuring an IP Fabric using Junos Space Network Director or OpenClos” on page 88.
CLI-Equivalent Configuration

The following commands show the resulting configuration created by the Network Director Layer 3 Fabric wizard (or OpenClos). This example is for the first leaf device (Leaf-0):

```plaintext
### System configuration ###
set system host-name cloud-saas-leaf-0
set system time-zone America/Los_Angeles
set system root-authentication encrypted-password [##hash##]
set system services ssh root-login allow
set system services ssh max-sessions-per-connection 32
set system services netconf ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set system syslog file default-log-messages any any
set system syslog file default-log-messages match "(requested 'commit' operation) | (copying configuration to juniper.save) | (commit complete) | (IfAdminStatus | (FRU power) | (FRU removal) | (FRU insertion) | (link UP) | transitioned | Transferred | transfer-file | (license add) | (license delete) | (package -X update) | (package -X delete) | (FRU Online) | (FRU Offline) | (plugged in) | (unplugged) | (OF_NODE | (OF_SERVER_NODE_GROUP | (OF_INTERCONNECT | (OF_DIRECTOR | QF_NETWORK_NODE_GROUP | (Master Unchanged, Members Changed) | (Master Changed, Members Changed) | (Master Detected, Members Changed) | (vc add) | (vc delete) | (Master detected) | (Master changed) | (Backup detected) | (Backup changed) | (interface vcp-)")
set system syslog file default-log-messages structured-data
set system extensions providers juniper license-type juniper deployment-scope commercial
set system extensions providers chef license-type juniper deployment-scope commercial
set system processes dhcp-service traceoptions file dhcp_logfile
set system processes dhcp-service traceoptions file size 10m
set system processes dhcp-service traceoptions level all
set system processes dhcp-service traceoptions flag all
set system processes app-engine-virtual-machine-management-service traceoptions level notice
set system processes app-engine-virtual-machine-management-service traceoptions flag all
### Leaf-to-spine interfaces ###
set interfaces et-0/0/48 mtu 9216
set interfaces et-0/0/48 unit 0 description facing_cloud-saas-spine-0
set interfaces et-0/0/48 unit 0 family inet mtu 9000
set interfaces et-0/0/48 unit 0 family inet address 192.168.11.1/31
set interfaces et-0/0/49 mtu 9216
set interfaces et-0/0/49 unit 0 description facing_cloud-saas-spine-1
set interfaces et-0/0/49 unit 0 family inet mtu 9000
set interfaces et-0/0/49 unit 0 family inet address 192.168.11.17/31
set interfaces et-0/0/50 mtu 9216
set interfaces et-0/0/50 unit 0 description facing_cloud-saas-spine-2
set interfaces et-0/0/50 unit 0 family inet mtu 9000
set interfaces et-0/0/50 unit 0 family inet address 192.168.11.33/31
set interfaces et-0/0/51 mtu 9216
set interfaces et-0/0/51 unit 0 description facing_cloud-saas-spine-3
set interfaces et-0/0/51 unit 0 family inet mtu 9000
set interfaces et-0/0/51 unit 0 family inet address 192.168.11.49/31
### Server-facing interfaces ###
set interfaces xe-0/0/0 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/0 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members SERVER
```
set interfaces xe-0/0/3 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/3 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/4 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/4 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/5 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/5 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/6 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/6 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/7 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/7 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/9 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/9 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/10 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/10 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/11 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/11 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/12 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/12 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/13 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/13 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/14 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/14 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/15 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/15 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/16 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/16 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/17 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/17 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/18 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/18 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/19 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/19 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/20 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/20 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/21 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/21 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/22 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/22 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/23 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/23 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/24 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/24 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/25 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/25 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/26 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/26 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/27 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/27 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/28 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/28 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/29 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/29 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/30 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/30 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/31 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/31 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/32 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/32 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/32 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/33 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/33 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/34 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/34 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/35 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/35 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/36 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/36 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/37 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/37 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/38 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/38 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/39 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/39 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/40 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/40 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/41 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/41 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/42 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/42 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/43 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/43 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/44 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/44 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/45 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/45 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/46 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/46 unit 0 family ethernet-switching vlan members SERVER
set interfaces xe-0/0/47 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/47 unit 0 family ethernet-switching vlan members SERVER

### Loopback and management interfaces ###
set interfaces lo0 unit 0 family inet address 10.0.16.1/32
set interfaces vme unit 0 family inet address 10.94.47.28

### IRB for the local VLAN "SERVERS" ###
set interfaces irb mtu 9216
set interfaces irb unit 1 description LOCAL_SERVERS
set interfaces irb unit 1 family inet mtu 9000
set interfaces irb unit 1 family inet address 172.16.64.1/27

### VLANs ###
set vlans SERVER vlan-id 1
set vlans SERVER 13-interface irb.1

### Static routes & routing options ###
set routing-options static route 10.94.63.252/32 next-hop 10.94.47.14
set routing-options static route 10.94.63.253/32 next-hop 10.94.47.14
set routing-options static route 0.0.0.0/0 next-hop 10.94.47.14
set routing-options forwarding-table export PFE-LB
set routing-options router-id 10.0.16.1
set routing-options autonomous-system 420006000

### BGP configuration ###
set protocols bgp log-updown
set protocols bgp import bgp-clos-in
set protocols bgp export bgp-clos-out
set protocols bgp graceful-restart
set protocols bgp group CLOS type external
set protocols bgp group CLOS mtu-discovery
set protocols bgp group CLOS bfd-liveness-detection minimum-interval 250
set protocols bgp group CLOS bfd-liveness-detection multiplier 3
set protocols bgp group CLOS bfd-liveness-detection session-mode single-hop
set protocols bgp group CLOS multipath multiple-as
set protocols bgp group CLOS neighbor 192.168.11.0 peer-as 420005000
set protocols bgp group CLOS neighbor 192.168.11.16 peer-as 420005001
set protocols bgp group CLOS neighbor 192.168.11.32 peer-as 420005002
set protocols bgp group CLOS neighbor 192.168.11.48 peer-as 420005003
### Routing policy ###
set policy-options policy-statement PFE-LB then load-balance per-packet
set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.16.0/28 orlonger
set policy-options policy-statement bgp-clos-in term loopbacks then accept
set policy-options policy-statement bgp-clos-in term server-L3-gw from route-filter 172.16.64.0/24 orlonger
set policy-options policy-statement bgp-clos-in term server-L3-gw then accept
set policy-options policy-statement bgp-clos-out term loopback from protocol direct
set policy-options policy-statement bgp-clos-out term loopback from route-filter 10.0.16.1/32 orlonger
set policy-options policy-statement bgp-clos-out term loopback then next-hop self
set policy-options policy-statement bgp-clos-out term loopback then accept
set policy-options policy-statement bgp-clos-out term server-L3-gw from protocol direct
set policy-options policy-statement bgp-clos-out term server-L3-gw from route-filter 172.16.64.1/27 orlonger
set policy-options policy-statement bgp-clos-out term server-L3-gw then next-hop self
set policy-options policy-statement bgp-clos-out term server-L3-gw then accept
### LLDP ###
set protocols lldp interface all
### SNMP and event-options ###
set snmp community public authorization read-write
set snmp trap-group openclos_trap_group
set snmp trap-group networkdirector_trap_group version v2
set snmp trap-group networkdirector_trap_group destination-port 10162
set snmp trap-group networkdirector_trap_group categories authentication
set snmp trap-group networkdirector_trap_group categories link
set snmp trap-group networkdirector_trap_group categories services
set snmp trap-group networkdirector_trap_group targets 10.94.63.253
set snmp trap-group space targets 10.94.63.252
set event-options policy target_add_test events snmpd_trap_target_add_notice
set event-options policy target_add_test then raise-trap

**Configuring Additional Leaf Device Elements**

To quickly configure the additional elements for the leaf devices, enter the following configuration statements on each device:

```
NOTE: The configuration shown here applies to device Leaf-0.
```

[edit]
set interfaces et-0/0/48 unit 0 family inet6 mtu 9000
set interfaces et-0/0/48 unit 0 family inet6 address 2001:db8:2001::1/126
set interfaces et-0/0/49 unit 0 family inet6 mtu 9000
set interfaces et-0/0/49 unit 0 family inet6 address 2001:db8:2001::2/126
set interfaces et-0/0/50 unit 0 family inet6 mtu 9000
set interfaces et-0/0/50 unit 0 family inet6 address 2001:db8:2001::3/126
set interfaces et-0/0/51 unit 0 family inet6 mtu 9000
set interfaces et-0/0/51 unit 0 family inet6 address 2001:db8:2001::4/126
set protocols bgp group CLOS-IPV6 bfd-liveness-detection minimum-interval 250
set protocols bgp group CLOS-IPV6 bfd-liveness-detection multiplier 3
set protocols bgp group CLOS-IPV6 multipath multiple-as
set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:1::2 peer-as 420005000
set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:2::2 peer-as 420005001
set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:3::2 peer-as 420005002
set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:4::2 peer-as 420005003
set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.22.0/28 orlonger
set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.23.0/28 orlonger
set routing-options graceful-restart

Step-by-Step Procedure

To configure the additional elements for the leaf devices:

1. Configure IPv6 on the spine-facing interfaces:

   [edit]
   user@Leaf-0# set interfaces et-0/0/48 unit 0 family inet6 mtu 9000
   user@Leaf-0# set interfaces et-0/0/48 unit 0 family inet6 address 2001:db8:2001:1::1/126
   user@Leaf-0# set interfaces et-0/0/49 unit 0 family inet6 mtu 9000
   user@Leaf-0# set interfaces et-0/0/49 unit 0 family inet6 address 2001:db8:2001:2::1/126
   user@Leaf-0# set interfaces et-0/0/50 unit 0 family inet6 mtu 9000
   user@Leaf-0# set interfaces et-0/0/50 unit 0 family inet6 address 2001:db8:2001:3::1/126
   user@Leaf-0# set interfaces et-0/0/51 unit 0 family inet6 mtu 9000
   user@Leaf-0# set interfaces et-0/0/51 unit 0 family inet6 address 2001:db8:2001:4::1/126

2. Configure IPv6 EBGP sessions with each spine device:

   [edit]
   user@Leaf-0# set protocols bgp group CLOS-IPV6 bfd-liveness-detection minimum-interval 250
   user@Leaf-0# set protocols bgp group CLOS-IPV6 bfd-liveness-detection multiplier 3
   user@Leaf-0# set protocols bgp group CLOS-IPV6 multipath multiple-as
   user@Leaf-0# set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:1::2 peer-as 420005000
   user@Leaf-0# set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:2::2 peer-as 420005001
   user@Leaf-0# set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:3::2 peer-as 420005002
   user@Leaf-0# set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:4::2 peer-as 420005003

3. Configure additional routing policy elements to enable reachability to the loopback interfaces of the fabric and core devices:

   [edit]
   user@Leaf-0# set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.22.0/28 orlonger
   user@Leaf-0# set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.23.0/28 orlonger

4. Configure graceful restart globally:

   [edit]
   user@Leaf-0# set routing-options graceful-restart
Configuring Server Access Options on Leaf Devices

Step-by-Step Procedure

The SaaS solution offers three methods for server connectivity, as shown in Figure 7 on page 25.

Figure 7: SaaS Server Access Options

Leaf devices can be configured to support server connectivity in three ways:

- **Anycast**—Using a Layer 3 interface, with a BGP session between the server and the physical interface of the leaf device.
- **Unicast**—Using a Layer 2 interface and VLAN, with no BGP session between the server and the leaf device. Servers use the IRB interface of the leaf device as their default gateway.
- **Hybrid**—Using a Layer 2 interface and VLAN, with a BGP session between the server and the IRB interface of the leaf device.

### Anycast Server Access Option

This method uses a Layer 3 interface, with a BGP session between the server and the physical interface of the leaf device.

To configure the anycast server access option:

1. **Configure the server-facing interfaces:**

   ```
   [edit]
   user@Leaf-0# set interfaces xe-0/0/0 unit 0 family inet address 172.16.71.2/30
   user@Leaf-0# set interfaces xe-0/0/1 unit 0 family inet address 172.16.72.2/30
   user@Leaf-0# set interfaces xe-0/0/46 unit 0 family inet6 address 2001:db8:2001:92::2/126
   user@Leaf-0# set interfaces xe-0/0/47 unit 0 family inet6 address 2001:db8:2001:93::2/126
   ```

2. **Configure EBGP sessions with the servers:**

   ```
   [edit]
   user@Leaf-0# set protocols bgp group Anycast bfd-liveness-detection minimum-interval 250
   user@Leaf-0# set protocols bgp group Anycast multipath multiple-as
   user@Leaf-0# set protocols bgp group Anycast allow all ## enables BGP autoprovisioning for additional connections
   user@Leaf-0# set protocols bgp group Anycast neighbor 172.16.71.1 local-address 172.16.71.2
   user@Leaf-0# set protocols bgp group Anycast neighbor 172.16.71.1 peer-as 420008501
   user@Leaf-0# set protocols bgp group Anycast neighbor 172.16.72.1 local-address 172.16.72.2
   user@Leaf-0# set protocols bgp group Anycast neighbor 172.16.72.1 peer-as 420008502
   ```
Step-by-Step Procedure

Unicast Server Access Option

This method uses a Layer 2 interface and VLAN, with no BGP session between the server and the leaf device. Servers use the IRB interface of the leaf device as their default gateway.

To configure the unicast server access option:

1. Configure the Layer 2 server-facing interfaces, and associate the interfaces to VLAN SERVER:

   ```
   [edit]
   user@Leaf-0# set interfaces xe-0/0/0 unit 0 family ethernet-switching interface-mode access
   user@Leaf-0# set interfaces xe-0/0/0 unit 0 family ethernet-switching vlan members SERVER
   user@Leaf-0# set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode access
   user@Leaf-0# set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members SERVER
   ```

2. Configure an IRB interface to act as the default gateway for the servers:

   ```
   [edit]
   user@Leaf-0# set interfaces irb mtu 9216
   user@Leaf-0# set interfaces irb unit 1 description LOCAL_SERVERS
   user@Leaf-0# set interfaces irb unit 1 family inet mtu 9000
   user@Leaf-0# set interfaces irb unit 1 family inet address 172.16.64.1/27
   ```

3. Configure VLAN SERVER to aggregate the server-facing Layer 2 interfaces and associate them with the IRB interface:

   ```
   [edit]
   user@Leaf-0# set vlans SERVER vlan-id
   user@Leaf-0# set vlans SERVER l3-interface irb.1
   ```
Step-by-Step Procedure

Hybrid Server Access Option

This method uses a Layer 2 interface and VLAN, with a BGP session between the server and the IRB interface of the leaf device.

To configure the hybrid server access option:

1. Configure the Layer 2 server-facing interfaces, and associate the interfaces to VLAN hybrid:
   
   ```
   [edit]
   user@Leaf-0# set interfaces xe-0/0/0 unit 0 family ethernet-switching vlan members hybrid
   user@Leaf-0# set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members hybrid
   ```

2. Configure an IRB interface to act as the peering point for BGP connections with the servers:
   
   ```
   [edit]
   user@Leaf-0# set interfaces irb mtu 9216
   user@Leaf-0# set interfaces irb unit 100 description Hybrid
   user@Leaf-0# set interfaces irb unit 100 family inet mtu 9000
   user@Leaf-0# set interfaces irb unit 100 family inet address 172.16.73.2/24
   ```

3. Configure VLAN hybrid to aggregate the server-facing Layer 2 interfaces, and associate them with the IRB interface:
   
   ```
   [edit]
   user@Leaf-0# set vlans hybrid vlan-id 100
   user@Leaf-0# set vlans hybrid l3-interface irb.100
   ```

4. Configure EBGP (with BFD) sessions with the servers:
   
   ```
   [edit]
   user@Leaf-0# set protocols bgp group Hybrid bfd-liveness-detection minimum-interval 350
   user@Leaf-0# set protocols bgp group Hybrid bfd-liveness-detection multiplier 3
   user@Leaf-0# set protocols bgp group Hybrid bfd-liveness-detection session-mode single-hop
   user@Leaf-0# set protocols bgp group Hybrid multipath multiple-as
   user@Leaf-0# set protocols bgp group Hybrid allow all ## enables BGP autoprovisioning for additional connections
   user@Leaf-0# set protocols bgp group Hybrid neighbor 172.16.73.1 local-address 172.16.73.2 peer-as 420006503
   user@Leaf-0# set protocols bgp group Hybrid neighbor 172.16.73.3 local-address 172.16.73.2 peer-as 420006504
   ```
Configuring Server Load Balancing Using Anycast

In data centers, a common way to increase the capacity, and availability, of applications and services is to duplicate them across multiple servers, and assign all servers the same address to take advantage of anycast’s inherent load balancing capability. Separation of applications can be achieved by running each application on a different group of servers, and assigning a unique anycast address to each server group.

Figure 8: SaaS Anycast Server Load Balancing

In the example shown in Figure 8 on page 28, the leaf device has multiple Layer 3 interfaces connected to multiple servers. Each server has a separate BGP session established with the leaf device, and all servers are using the same anycast IP address.

To support server load balancing for anycast traffic, leaf devices can be configured with three configuration elements:

- A per-flow balancing policy applied to the forwarding table (configured earlier in this example)
- ECMP, with resilient hashing
- BGP, using multipath

To configure load balancing across multiple servers:
1. Configure the server-facing interfaces:

[edit]
user@Leaf-0# set interfaces xe-0/0/2 unit 0 family inet address 172.16.152.1/30
user@Leaf-0# set interfaces xe-0/0/3 unit 0 family inet address 172.16.153.1/30
user@Leaf-0# set interfaces xe-0/0/4 unit 0 family inet address 172.16.154.1/30
user@Leaf-0# set interfaces xe-0/0/5 unit 0 family inet address 172.16.155.1/30
user@Leaf-0# set interfaces xe-0/0/6 unit 0 family inet address 172.16.156.1/30
user@Leaf-0# set interfaces xe-0/0/7 unit 0 family inet address 172.16.157.1/30
user@Leaf-0# set interfaces xe-0/0/8 unit 0 family inet address 172.16.158.1/30
user@Leaf-0# set interfaces xe-0/0/9 unit 0 family inet address 172.16.159.1/30
user@Leaf-0# set interfaces xe-0/0/10 unit 0 family inet address 172.16.160.1/30
user@Leaf-0# set interfaces xe-0/0/11 unit 0 family inet address 172.16.161.1/30
user@Leaf-0# set interfaces xe-0/0/12 unit 0 family inet address 172.16.162.1/30
user@Leaf-0# set interfaces xe-0/0/13 unit 0 family inet address 172.16.163.1/30
user@Leaf-0# set interfaces xe-0/0/14 unit 0 family inet address 172.16.164.1/30
user@Leaf-0# set interfaces xe-0/0/15 unit 0 family inet address 172.16.165.1/30
user@Leaf-0# set interfaces xe-0/0/16 unit 0 family inet address 172.16.166.1/30
user@Leaf-0# set interfaces xe-0/0/17 unit 0 family inet address 172.16.167.1/30
user@Leaf-0# set interfaces xe-0/0/18 unit 0 family inet address 172.16.168.1/30
user@Leaf-0# set interfaces xe-0/0/19 unit 0 family inet address 172.16.169.1/30
user@Leaf-0# set interfaces xe-0/0/20 unit 0 family inet address 172.16.170.1/30
user@Leaf-0# set interfaces xe-0/0/21 unit 0 family inet address 172.16.171.1/30
user@Leaf-0# set interfaces xe-0/0/22 unit 0 family inet address 172.16.172.1/30
user@Leaf-0# set interfaces xe-0/0/23 unit 0 family inet address 172.16.173.1/30
user@Leaf-0# set interfaces xe-0/0/24 unit 0 family inet address 172.16.174.1/30
user@Leaf-0# set interfaces xe-0/0/25 unit 0 family inet address 172.16.175.1/30
user@Leaf-0# set interfaces xe-0/0/26 unit 0 family inet address 172.16.176.1/30
user@Leaf-0# set interfaces xe-0/0/27 unit 0 family inet address 172.16.177.1/30
user@Leaf-0# set interfaces xe-0/0/28 unit 0 family inet address 172.16.178.1/30
user@Leaf-0# set interfaces xe-0/0/29 unit 0 family inet address 172.16.179.1/30
user@Leaf-0# set interfaces xe-0/0/30 unit 0 family inet address 172.16.180.1/30
user@Leaf-0# set interfaces xe-0/0/31 unit 0 family inet address 172.16.181.1/30
user@Leaf-0# set interfaces xe-0/0/32 unit 0 family inet address 172.16.182.1/30
user@Leaf-0# set interfaces xe-0/0/33 unit 0 family inet address 172.16.183.1/30
user@Leaf-0# set interfaces xe-0/0/34 unit 0 family inet address 172.16.184.1/30
user@Leaf-0# set interfaces xe-0/0/35 unit 0 family inet address 172.16.185.1/30
user@Leaf-0# set interfaces xe-0/0/36 unit 0 family inet address 172.16.186.1/30
user@Leaf-0# set interfaces xe-0/0/37 unit 0 family inet address 172.16.187.1/30
user@Leaf-0# set interfaces xe-0/0/38 unit 0 family inet address 172.16.188.1/30
user@Leaf-0# set interfaces xe-0/0/39 unit 0 family inet address 172.16.189.1/30
user@Leaf-0# set interfaces xe-0/0/40 unit 0 family inet address 172.16.190.1/30
user@Leaf-0# set interfaces xe-0/0/41 unit 0 family inet address 172.16.191.1/30
user@Leaf-0# set interfaces xe-0/0/42 unit 0 family inet address 172.16.192.1/30
user@Leaf-0# set interfaces xe-0/0/43 unit 0 family inet address 172.16.193.1/30
user@Leaf-0# set interfaces xe-0/0/44 unit 0 family inet address 172.16.194.1/30
user@Leaf-0# set interfaces xe-0/0/45 unit 0 family inet address 172.16.195.1/30

2. Configure ECMP:

[edit]
user@Leaf-0# set chassis maximum-ecmp 64
user@Leaf-0# set forwarding-options enhanced-hash-key ecmp-resilient-hash

3. Configure EBGP (with BFD) sessions with the servers:

[edit]
user@Leaf-0# set protocols bgp group LB-Anycast bfd-liveness-detection minimum-interval 250
user@Leaf-0# set protocols bgp group LB-Anycast bfd-liveness-detection session-mode single-hop
user@Leaf-0# set protocols bgp group LB-Anycast multipath
user@Leaf-0# set protocols bgp group LB-Anycast allow all  # enables BGP autoprovisioning for additional connections
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.152.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.153.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.154.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.155.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.156.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.157.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.158.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.159.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.160.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.161.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.162.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.163.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.164.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.165.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.166.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.167.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.168.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.169.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.170.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.171.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.172.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.173.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.174.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.175.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.176.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.177.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.178.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.179.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.180.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.181.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.182.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.183.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.184.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.185.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.186.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.187.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.188.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.189.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.190.2 peer-as 420006700
user@Leaf-0# set protocols bgp group LB-Anycast neighbor 172.16.191.2 peer-as 420006700
4. Configure routing policies to advertise the anycast IPv4 address (10.1.1.1) and anycast IPv6 address (2001:db8:2000::):

   [edit]
   user@Leaf-0# set policy-options policy-statement bgp-clos-out term LB-Anycast4-route from protocol bgp
   user@Leaf-0# set policy-options policy-statement bgp-clos-out term LB-Anycast4-route from route-filter 10.1.1.1/24 exact
   user@Leaf-0# set policy-options policy-statement bgp-clos-out term LB-Anycast4-route then accept
   user@Leaf-0# set policy-options policy-statement bgp-clos-out term LB-Anycast6-route from protocol bgp
   user@Leaf-0# set policy-options policy-statement bgp-clos-out term LB-Anycast6-route from route-filter 2001:db8:2000::/64 exact
   user@Leaf-0# set policy-options policy-statement bgp-clos-out term LB-Anycast6-route then accept

**Configuring Spine Devices for the IP Fabric**

As noted earlier, Juniper Networks provides tools (OpenClos and Junos Space Network Director) to help automate the creation of spine-and-leaf IP fabrics for SaaS environments.

For this example, the main configuration elements for the spine devices were created using Network Director. For more information on using Network Director or OpenClos for this solution, see “Configuring an IP Fabric using Junos Space Network Director or OpenClos” on page 88.

**CLI-Equivalent Configuration**

The following commands show the resulting configuration created by the Network Director Layer 3 Fabric wizard (or OpenClos). This example is for the first spine device (Spine-0):

```plaintext
### System configuration ###
set system host-name cloud-saas-spine-0
set system time-zone America/Los_Angeles
set system root-authentication encrypted-password [##hash##]
set system services ssh root-login allow
set system services ssh max-sessions-per-connection 32
set system services netconf ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set system syslog file default-log-messages any
set system syslog file default-log-messages match "/(requested 'commit' operation) | (copying configuration to juniper.save) | (commit complete) | IfAdminStatus | (FRU power) | (FRU removal) | (FRU insertion) | (link UP) | transitioned | Transferred | transfer-file | (license add) | (license delete) | (package -X update) | (package -X delete) | (FRU Online) | (FRU Offline) | (plugged in) | (unplugged) | QF_NODE | QF_SERVER_NODE_GROUP | QF_INTERCONNECT | QF_DIRECTOR |
```
QF_NETWORK_NODE_GROUP | (Master Unchanged, Members Changed) | (Master Changed, Members Changed) | (Master Detected, Members Changed) | (vc add) | (vc delete) | (Master detected) | (Master changed) | (Backup detected) | (Backup changed) | (interface vcp-)

set system syslog file default-log-messages structured-data
set system extensions providers juniper license-type juniper deployment-scope commercial
set system extensions providers chef license-type juniper deployment-scope commercial
set system processes dhcp-service traceoptions file dhcp_logfile
set system processes dhcp-service traceoptions file size 10m
set system processes dhcp-service traceoptions level all
set system processes dhcp-service traceoptions flag all
set system processes app-engine-virtual-machine-management-service traceoptions level notice
set system processes app-engine-virtual-machine-management-service traceoptions flag all

### Spine-to-leaf interfaces ###
set interfaces et-0/0/0 mtu 9216
set interfaces et-0/0/0 unit 0 description facing_cloud-saas-leaf-0
set interfaces et-0/0/0 unit 0 family inet mtu 9000
set interfaces et-0/0/0 unit 0 family inet address 192.168.11.0/31
set interfaces et-0/0/0/1 mtu 9216
set interfaces et-0/0/0/1 unit 0 description facing_cloud-saas-leaf-1
set interfaces et-0/0/0/1 unit 0 family inet mtu 9000
set interfaces et-0/0/0/1 unit 0 family inet address 192.168.11.1/31
set interfaces et-0/0/0/2 mtu 9216
set interfaces et-0/0/0/2 unit 0 description facing_cloud-saas-leaf-2
set interfaces et-0/0/0/2 unit 0 family inet mtu 9000
set interfaces et-0/0/0/2 unit 0 family inet address 192.168.11.2/31
set interfaces et-0/0/0/3 mtu 9216
set interfaces et-0/0/0/3 unit 0 description facing_cloud-saas-leaf-3
set interfaces et-0/0/0/3 unit 0 family inet mtu 9000
set interfaces et-0/0/0/3 unit 0 family inet address 192.168.11.6/31
set interfaces et-0/0/0/4 mtu 9216
set interfaces et-0/0/0/4 unit 0 description facing_cloud-saas-leaf-4
set interfaces et-0/0/0/4 unit 0 family inet mtu 9000
set interfaces et-0/0/0/4 unit 0 family inet address 192.168.11.8/31
set interfaces et-0/0/0/5 mtu 9216
set interfaces et-0/0/0/5 unit 0 description facing_cloud-saas-leaf-5
set interfaces et-0/0/0/5 unit 0 family inet mtu 9000
set interfaces et-0/0/0/5 unit 0 family inet address 192.168.11.10/31

### Loopback and management interfaces ###
set interfaces lo0 unit 0 family inet address 10.0.16.9/32
set interfaces vme unit 0 family inet address 10.94.47.1/28

### Static routes & routing options ###
set routing-options static route 10.94.63.252/32 next-hop 10.94.47.14
set routing-options static route 10.94.63.253/32 next-hop 10.94.47.14
set routing-options static route 0.0.0.0/0 next-hop 10.94.47.14
set routing-options forwarding-table export PFE-LB
set routing-options router-id 10.0.16.9
set routing-options autonomous-system 420005000

### BGP configuration ###
set protocols bgp log-updown
set protocols bgp import bgp-clos-in
set protocols bgp export bgp-clos-out
set protocols bgp graceful-restart
set protocols bgp group CLOS type external
set protocols bgp group CLOS mtu-discovery
set protocols bgp group CLOS bfd-liveness-detection minimum-interval 250
set protocols bgp group CLOS bfd-liveness-detection multiplier 3
set protocols bgp group CLOS bfd-liveness-detection session-mode single-hop
set protocols bgp group CLOS multipath multiple-as
set protocols bgp group CLOS neighbor 192.168.11.1 peer-as 420006000
set protocols bgp group CLOS neighbor 192.168.11.3 peer-as 420006001
set protocols bgp group CLOS neighbor 192.168.11.5 peer-as 420006002
set protocols bgp group CLOS neighbor 192.168.11.7 peer-as 420006003
set protocols bgp group CLOS neighbor 192.168.11.9 peer-as 420006004
set protocols bgp group CLOS neighbor 192.168.11.11 peer-as 420006005

### Routing policy ###
set policy-options policy-statement PFE-LB then load-balance per-packet
set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.16.0/28 orlonger
set policy-options policy-statement bgp-clos-in term loopbacks then accept
set policy-options policy-statement bgp-clos-in term server-L3-gw from route-filter 172.16.64.0/24 orlonger
set policy-options policy-statement bgp-clos-in term server-L3-gw then accept
set policy-options policy-statement bgp-clos-in term reject then reject
set policy-options policy-statement bgp-clos-out term loopback from protocol direct
set policy-options policy-statement bgp-clos-out term loopback from route-filter 10.0.16.0/28 orlonger
set policy-options policy-statement bgp-clos-out term loopback then next-hop self
set policy-options policy-statement bgp-clos-out term loopback then accept
set policy-options policy-statement bgp-clos-out term server-L3-gw from protocol direct
set policy-options policy-statement bgp-clos-out term server-L3-gw from route-filter 172.16.64.0/24 orlonger
set policy-options policy-statement bgp-clos-out term server-L3-gw then next-hop self
set policy-options policy-statement bgp-clos-out term server-L3-gw then accept

### LLDP ###
set protocols lldp interface all

### SNMP and event-options ###
set snmp community public authorization read-write
set snmp trap-group networkdirector_trap_group version v2
set snmp trap-group networkdirector_trap_group destination-port 10162
set snmp trap-group networkdirector_trap_group categories authentication
set snmp trap-group networkdirector_trap_group categories link
set snmp trap-group networkdirector_trap_group categories services
set snmp trap-group networkdirector_trap_group targets 10.94.63.253
set snmp trap-group space targets 10.94.63.252
set event-options policy target_add_test events snmpd_trap_target_add_notice
set event-options policy target_add_test then raise-trap

Configuring Additional Spine Device Elements

To quickly configure additional elements for the spine devices, enter the following configuration statements on each device:

NOTE: The configuration shown here applies to device Spine-0.

```
[edit]
set interfaces et-0/0/0 unit 0 family inet6 mtu 9000
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:2001::2/126
set interfaces et-0/0/1 unit 0 family inet6 mtu 9000
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:2001::5::2/126
set interfaces et-0/0/2 unit 0 family inet6 mtu 9000
set interfaces et-0/0/2 unit 0 family inet6 address 2001:db8:2001::9::2/126
set interfaces et-0/0/3 unit 0 family inet6 mtu 9000
set interfaces et-0/0/3 unit 0 family inet6 address 2001:db8:2001::13::2/126
set interfaces et-0/2/0 unit 0 family inet address 192.168.13.13/30
set interfaces et-0/2/0 unit 0 family inet6 address 2001:db8:2001:24::1/126
```
set interfaces et-0/2/1 unit 0 family inet address 192.168.13.9/30
set interfaces et-0/2/1 unit 0 family inet6 address 2001:db8:2001:23::1/126
set interfaces et-0/2/2 unit 0 family inet address 192.168.13.5/30
set interfaces et-0/2/2 unit 0 family inet6 address 2001:db8:2001:22::1/126
set interfaces et-0/2/3 unit 0 family inet address 192.168.13.1/30
set interfaces et-0/2/3 unit 0 family inet6 address 2001:db8:2001:21::1/126
set protocols bgp group CLOS-IPV6 bfd-liveness-detection minimum-interval 250
set protocols bgp group CLOS-IPV6 bfd-liveness-detection multiplier 3
set protocols bgp group CLOS-IPV6 multipath multiple-as
set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:1::1 peer-as 420006000
set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:5::1 peer-as 420006001
set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:9::1 peer-as 420006002
set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:13::1 peer-as 420006003
set protocols bgp group FABRIC bfd-liveness-detection minimum-interval 250
set protocols bgp group FABRIC multipath multiple-as
set protocols bgp group FABRIC neighbor 192.168.13.2 peer-as 420005501
set protocols bgp group FABRIC neighbor 192.168.13.6 peer-as 420005502
set protocols bgp group FABRIC neighbor 192.168.13.10 peer-as 420005503
set protocols bgp group FABRIC neighbor 192.168.13.14 peer-as 420005504
set protocols bgp group FABRIC-IPV6 bfd-liveness-detection minimum-interval 250
set protocols bgp group FABRIC-IPV6 multipath multiple-as
set protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:21::2 peer-as 420005501
set protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:22::2 peer-as 420005502
set protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:23::2 peer-as 420005503
set protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:24::2 peer-as 420005504
set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.22.0/28 orlonger
set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.23.0/28 orlonger
set policy-options policy-statement bgp-clos-in term LB-Anycast4-route from protocol bgp
set policy-options policy-statement bgp-clos-in term LB-Anycast4-route from route-filter 10.1.1.1/24 exact
set policy-options policy-statement bgp-clos-in term LB-Anycast4-route then accept
user@Spine-0# insert policy-options policy-statement bgp-clos-in term LB-Anycast4-route before term reject
set policy-options policy-statement bgp-clos-in term LB-Anycast6-route from protocol bgp
set policy-options policy-statement bgp-clos-in term LB-Anycast6-route from route-filter 2001:db8:2001::64 exact
set policy-options policy-statement bgp-clos-in term LB-Anycast6-route then accept
user@Spine-0# insert policy-options policy-statement bgp-clos-in term LB-Anycast6-route before term reject
set chassis maximum-ecmp 64
set forwarding-options enhanced-hash-key ecmp-resilient-hash
set routing-options graceful-restart

Step-by-Step Procedure

To configure the additional elements for the spine devices:

1. Configure IPv6 on the leaf-facing interfaces:

   [edit]
   user@Spine-0# set interfaces et-0/0/0 unit 0 family inet6 mtu 9000
   user@Spine-0# set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:2001:1::2/126
   user@Spine-0# set interfaces et-0/0/1 unit 0 family inet6 mtu 9000
   user@Spine-0# set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:2001:5::2/126
   user@Spine-0# set interfaces et-0/0/2 unit 0 family inet6 mtu 9000
   user@Spine-0# set interfaces et-0/0/2 unit 0 family inet6 address 2001:db8:2001:9::2/126
   user@Spine-0# set interfaces et-0/0/3 unit 0 family inet6 mtu 9000
   user@Spine-0# set interfaces et-0/0/3 unit 0 family inet6 address 2001:db8:2001:13::2/126

2. Configure IPv4 and IPv6 on the fabric-facing interfaces:
Configure IPv6 EBGPsessions with each leaf device:

    [edit]
    user@Spine-0# set protocols bgp group CLOS-IPV6 bfd-liveness-detection minimum-interval 250
    user@Spine-0# set protocols bgp group CLOS-IPV6 bfd-liveness-detection multiplier 3
    user@Spine-0# set protocols bgp group CLOS-IPV6 multipath multiple-as
    user@Spine-0# set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:1::1 peer-as 420006000
    user@Spine-0# set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:5::1 peer-as 420006001
    user@Spine-0# set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:9::1 peer-as 420006002
    user@Spine-0# set protocols bgp group CLOS-IPV6 neighbor 2001:db8:2001:13::1 peer-as 420006003

Configure IPv4 and IPv6 EBGPsessions with each fabric device:

    [edit]
    user@Spine-0# set protocols bgp group FABRIC bfd-liveness-detection minimum-interval 250
    user@Spine-0# set protocols bgp group FABRIC multipath multiple-as
    user@Spine-0# set protocols bgp group FABRIC neighbor 192.168.13.2 peer-as 420005501
    user@Spine-0# set protocols bgp group FABRIC neighbor 192.168.13.6 peer-as 420005502
    user@Spine-0# set protocols bgp group FABRIC neighbor 192.168.13.10 peer-as 420005503
    user@Spine-0# set protocols bgp group FABRIC neighbor 192.168.13.14 peer-as 420005504
    user@Spine-0# set protocols bgp group FABRIC-IPV6 bfd-liveness-detection minimum-interval 250
    user@Spine-0# set protocols bgp group FABRIC-IPV6 multipath multiple-as
    user@Spine-0# set protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:21::2 peer-as 420005501
    user@Spine-0# set protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:22::2 peer-as 420005502
    user@Spine-0# set protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:23::2 peer-as 420005503
    user@Spine-0# set protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:24::2 peer-as 420005504

Configure additional route filters for the loopbacks term in the bgp-clos-in routing policy to enable reachability to the loopback interfaces of the fabric and core devices:

    [edit]
    user@Spine-0# set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.22.0/28 or longer
    user@Spine-0# set policy-options policy-statement bgp-clos-in term loopbacks from route-filter 10.0.23.0/28 or longer

Configure additional terms for the bgp-clos-in routing policy to advertise the anycast IPv4 address (10.1.1.1) and anycast IPv6 address (2001:db8:2000::):

    [edit]
    user@Spine-0# set policy-options policy-statement bgp-clos-in term LB-Anycast4-route from protocol bgp
user@Spine-0# set policy-options policy-statement bgp-clos-in term LB-Anycast4-route from route-filter 10.1.1.1/24 exact
user@Spine-0# set policy-options policy-statement bgp-clos-in term LB-Anycast4-route then accept
user@Spine-0# insert policy-options policy-statement bgp-clos-in term LB-Anycast4-route before term reject
user@Spine-0# set policy-options policy-statement bgp-clos-in term LB-Anycast6-route from protocol bgp
user@Spine-0# set policy-options policy-statement bgp-clos-in term LB-Anycast6-route from route-filter 2001:db8:2000::/64 exact
user@Spine-0# set policy-options policy-statement bgp-clos-in term LB-Anycast6-route then accept
user@Spine-0# insert policy-options policy-statement bgp-clos-in term LB-Anycast6-route before term reject

7. Configure ECMP:

    [edit]
    user@Spine-0# set chassis maximum-ecmp 64
    user@Spine-0# set forwarding-options enhanced-hash-key ecmp-resilient-hash

8. Configure graceful restart globally:

    [edit]
    user@Spine-0# set routing-options graceful-restart

---

Configuring Fabric Devices

**CLI Quick Configuration**

To quickly configure the fabric devices, enter the following configuration statements on each device:

```
[i]
set groups int-global interfaces <*> mtu 9192
set chassis fpc 0 pic 0 port 0 channel-speed 10g
set interfaces apply-groups int-global
set interfaces et-0/0/20 unit 0 family inet address 192.168.13.2/30
set interfaces et-0/0/20 unit 0 family inet6 address 2001:db8:2001:21::2/126
set interfaces et-0/0/21 unit 0 family inet address 192.168.13.18/30
set interfaces et-0/0/21 unit 0 family inet6 address 2001:db8:2001:25::2/126
set interfaces et-0/0/22 unit 0 family inet address 192.168.13.34/30
set interfaces et-0/0/22 unit 0 family inet6 address 2001:db8:2001:29::2/126
set interfaces et-0/0/23 unit 0 family inet address 192.168.13.50/30
set interfaces et-0/0/23 unit 0 family inet6 address 2001:db8:2001:33::2/126
set interfaces xe-0/0/0:0 unit 0 family inet address 192.168.14.5/30
set interfaces xe-0/0/0:0 unit 0 family inet6 address 2001:db8:2001:51::1/126
set interfaces xe-0/0/0:3 unit 0 family inet address 192.168.14.5/30
set interfaces xe-0/0/0:3 unit 0 family inet6 address 2001:db8:2001:52::1/126
set interfaces lo0 unit 0 family inet address 10.0.22.1/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:2003::1/128
set interfaces em0 unit 0 family inet address 10.94.191.64/24
set routing-options static route 10.94.63.252/32 next-hop 10.94.191.254
set routing-options static route 10.94.63.253/32 next-hop 10.94.191.254
set routing-options router-id 10.0.22.1
set routing-options autonomous-system 420005501
```
Step-by-Step Procedure

To configure the fabric devices:

1. Configure an interface group to set the MTU value for all interfaces:
   
   `[edit]`  
   user@Fabric-0# set groups int-global interfaces <*> mtu 9192

2. Configure IPv4 and IPv6 on the spine-facing interfaces:
3. Configure IPv4 and IPv6 on the core-facing interfaces:

    [edit]
    user@Fabric-0# set interfaces xe-0/0/0:0 unit 0 family inet address 192.168.14.1/30
    user@Fabric-0# set interfaces xe-0/0/0:0 unit 0 family inet6 address 2001:db8:2001:51::1/126
    user@Fabric-0# set interfaces xe-0/0/0:3 unit 0 family inet address 192.168.14.5/30
    user@Fabric-0# set interfaces xe-0/0/0:3 unit 0 family inet6 address 2001:db8:2001:52::1/126

4. Configure the loopback and management interfaces:

    [edit]
    user@Fabric-0# set interfaces lo0 unit 0 family inet address 10.0.22.1/32
    user@Fabric-0# set interfaces lo0 unit 0 family inet6 address 2001:db8:2003:1::1/128
    user@Fabric-0# set interfaces em0 unit 0 family inet address 10.94.191.64/24

5. Configure static routes and routing options:

    [edit]
    user@Fabric-0# set routing-options static route 10.94.63.252/32 next-hop 10.94.191.254
    user@Fabric-0# set routing-options static route 10.94.63.253/32 next-hop 10.94.191.254
    user@Fabric-0# set routing-options router-id 10.0.22.1
    user@Fabric-0# set routing-options autonomous-system 420005501

6. Configure IPv4 and IPv6 EBGP sessions with each spine device:

    [edit]
    user@Fabric-0# set protocols bgp group SPINE bfd-liveness-detection minimum-interval 250
    user@Fabric-0# set protocols bgp group SPINE multipath multiple-as
    user@Fabric-0# set protocols bgp group SPINE neighbor 192.168.13.1 peer-as 420005000
    user@Fabric-0# set protocols bgp group SPINE neighbor 192.168.13.17 peer-as 420005001
    user@Fabric-0# set protocols bgp group SPINE neighbor 192.168.13.33 peer-as 420005002
    user@Fabric-0# set protocols bgp group SPINE neighbor 192.168.13.49 peer-as 420005003
    user@Fabric-0# set protocols bgp group SPINE-IPV6 bfd-liveness-detection minimum-interval 250
    user@Fabric-0# set protocols bgp group SPINE-IPV6 multipath multiple-as
    user@Fabric-0# set protocols bgp group SPINE-IPV6 neighbor 2001:db8:2001:21::1 peer-as 420005000
    user@Fabric-0# set protocols bgp group SPINE-IPV6 neighbor 2001:db8:2001:25::1 peer-as 420005001
    user@Fabric-0# set protocols bgp group SPINE-IPV6 neighbor 2001:db8:2001:29::1 peer-as 420005002
    user@Fabric-0# set protocols bgp group SPINE-IPV6 neighbor 2001:db8:2001:33::1 peer-as 420005003
7. Configure IPv4 and IPv6 EBGP sessions with each core router:

   [edit]
   user@Fabric-0# set protocols bgp bfd-liveness-detection minimum-interval 250
   user@Fabric-0# set protocols bgp group CORE multipath multiple-as
   user@Fabric-0# set protocols bgp group CORE neighbor 192.168.14.6 description to-r2-PE
   user@Fabric-0# set protocols bgp group CORE neighbor 192.168.14.6 peer-as 420006502
   user@Fabric-0# set protocols bgp group CORE neighbor 192.168.14.2 description to-r1-PE
   user@Fabric-0# set protocols bgp group CORE neighbor 192.168.14.2 peer-as 420006501
   user@Fabric-0# set protocols bgp group CORE-IPV6 bfd-liveness-detection minimum-interval 250
   user@Fabric-0# set protocols bgp group CORE-IPV6 multipath multiple-as
   user@Fabric-0# set protocols bgp group CORE-IPV6 neighbor 2001:db8:2001:51::2 peer-as 420006501
   user@Fabric-0# set protocols bgp group CORE-IPV6 neighbor 2001:db8:2001:52::2 peer-as 420006502

8. Configure and apply routing policies to enable reachability to the loopback interfaces of the other devices in the IP fabric:

   [edit]
   user@Fabric-0# set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.16.0/28 orlonger
   user@Fabric-0# set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.22.0/28 orlonger
   user@Fabric-0# set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.23.0/28 orlonger
   user@Fabric-0# set policy-options policy-statement receive-loopbacks term loopbacks then accept
   user@Fabric-0# set policy-options policy-statement advertise-loopbacks term loopback from protocol direct
   user@Fabric-0# set policy-options policy-statement advertise-loopbacks term loopback from route-filter 10.0.22.0/28 orlonger
   user@Fabric-0# set policy-options policy-statement advertise-loopbacks term loopback then next-hop self
   user@Fabric-0# set policy-options policy-statement advertise-loopbacks term loopback then accept
   user@Fabric-0# set protocols bgp import receive-loopbacks
   user@Fabric-0# set protocols bgp export advertise-loopbacks

9. Configure per-flow load balancing and ECMP:

   [edit]
   user@Fabric-0# set policy-options policy-statement pfe-lb then load-balance per-packet
   user@Fabric-0# set routing-options forwarding-table export pfe-lb
   user@Fabric-0# set chassis maximum-ecmp 64
   user@Fabric-0# set forwarding-options enhanced-hash-key ecmp-resilient-hash

10. Configure LLDP:

    [edit]
    user@Fabric-0# set protocols lldp interface all

11. Configure SNMP and event options:

    [edit]
    user@Fabric-0# set snmp community public authorization read-write
    user@Fabric-0# set snmp trap-group networkdirector_trap_group version v2
    user@Fabric-0# set snmp trap-group networkdirector_trap_group destination-port 10162
    user@Fabric-0# set snmp trap-group networkdirector_trap_group categories authentication
    user@Fabric-0# set snmp trap-group networkdirector_trap_group categories link
    user@Fabric-0# set snmp trap-group networkdirector_trap_group categories services
To quickly configure the core routers, enter the following configuration statements on each router:

```
[edit]
set groups int-global interfaces <*> mtu 9192
set groups int-global interfaces <*> unit 0 family mpls
set chassis network-services enhanced-ip
set interfaces apply-groups int-global
set interfaces xe-1/0/0 unit 0 family inet address 192.168.14.18/30
set interfaces xe-1/0/0 unit 0 family inet6 address 2001:db8:2001:55::2/126
set interfaces xe-1/0/1 unit 0 family inet address 192.168.14.26/30
set interfaces xe-1/0/1 unit 0 family inet6 address 2001:db8:2001:57::2/126
set interfaces xe-1/0/2 unit 0 family inet address 192.168.14.2/30
set interfaces xe-1/0/2 unit 0 family inet6 address 2001:db8:2001:51::2/126
set interfaces xe-1/0/3 unit 0 family inet address 192.168.14.10/30
set interfaces xe-1/0/3 unit 0 family inet6 address 2001:db8:2001:53::2/126
set interfaces xe-1/1/3/1 unit 0 family inet address 192.168.15.5/30
set interfaces xe-1/1/3/2 unit 0 family inet address 192.168.15.9/30
set interfaces xe-1/1/3/3 unit 0 family inet address 192.168.15.1/30
set interfaces lo0 unit 0 family inet address 10.0.23.1/32
set interfaces fxp0 unit 0 family inet address 10.94.191.55/24
set routing-options static route 10.94.63.252/32 next-hop 10.94.191.254
set routing-options static route 10.94.63.253/32 next-hop 10.94.191.254
set routing-options router-id 10.0.23.1
set routing-options autonomous-system 65000
set protocols bgp multihop
set protocols bgp group to-r2-PE type internal
set protocols bgp group to-r2-PE local-address 10.0.23.1
set protocols bgp group to-r2-PE family inet-vpn unicast
set protocols bgp group to-r2-PE bfd-liveness-detection minimum-interval 250
set protocols bgp group to-r2-PE multipath
set protocols bgp group to-r2-PE neighbor 10.0.23.2
set protocols ospf traffic-engineering shortcuts
set protocols ospf area 0.0.0.0 interface all node-link-protection
set protocols ospf area 0.0.0.0 interface all bfd-liveness-detection minimum-interval 250
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols rsvp interface all link-protection
set protocols mpls interface all
set protocols mpls label-switched-path to-r2 backup
set protocols mpls label-switched-path to-r2 to 10.0.23.2
set protocols mpls label-switched-path to-r2 standby
set protocols mpls label-switched-path to-r2 link-protection
set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.16.0/28 orlonger
```
set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.22.0/28 orlonger
set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.23.0/28 orlonger
set policy-options policy-statement receive-loopbacks term loopbacks then accept
set policy-options policy-statement advertise-loopbacks term loopback from protocol direct
set policy-options policy-statement advertise-loopbacks term loopback from route-filter 10.0.23.0/28 orlonger
set policy-options policy-statement advertise-loopbacks term loopback then next-hop self
set policy-options policy-statement advertise-loopbacks term loopback then accept
set policy-options policy-statement pfe-lb then load-balance per-packet
set routing-options forwarding-table export pfe-lb
set forwarding-options hash-key family inet layer-3
set forwarding-options hash-key family inet layer-4
set protocols lldp interface all
set protocols lldp interface fxp0 disable
set snmp community public authorization read-write
set snmp trap-group networkdirector_trap_group version v2
set snmp trap-group networkdirector_trap_group destination-port 10162
set snmp trap-group networkdirector_trap_group categories authentication
set snmp trap-group networkdirector_trap_group categories link
set snmp trap-group networkdirector_trap_group categories services
set snmp trap-group networkdirector_trap_group targets 10.94.63.253
set snmp trap-group space targets 10.94.63.252
set event-options policy target_add_test events snmpd_trap_target_add_notice
set event-options policy target_add_test then raise-trap
set routing-instances RED-vpn instance-type vrf
set routing-instances RED-vpn interface xe-1/0/0.0
set routing-instances RED-vpn interface xe-1/0/1.0
set routing-instances RED-vpn interface xe-1/0/2.0
set routing-instances RED-vpn interface xe-1/0/3.0
set routing-instances RED-vpn route-distinguisher 65001:1
set routing-instances RED-vpn vrf-target target:65001:1
set routing-instances RED-vpn protocols bgp import receive-loopbacks
set routing-instances RED-vpn protocols bgp export advertise-loopbacks
set routing-instances RED-vpn protocols bgp bfd-liveness-detection minimum-interval 250
set routing-instances RED-vpn protocols bgp group FABRIC local-as 420006501
set routing-instances RED-vpn protocols bgp group FABRIC multipath multiple-as
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.1 description Fabric-sw01
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.9 description Fabric-sw02
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.29 peer-as 420005501
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.1 peer-as 420005501
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.9 description Fabric-sw02
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.9 peer-as 420005502
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.17 peer-as 420005503
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.29 peer-as 420005501
set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.25 peer-as 420005504
set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 local-as 420006501
set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 multipath multiple-as
set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:51::1 peer-as 420005501
set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:53::1 peer-as 420005502
set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:55::1 peer-as 420005503
group FABRIC-IPV6 neighbor 2001:db8:2001:57::1 peer-as 420005504

Step-by-Step Procedure

To configure the core routers:

1. Configure an interface group to apply an MTU value and the MPLS protocol family to all interfaces:

   [edit]
   user@R1# set groups int-global interfaces <*> mtu 9192
   user@R1# set groups int-global interfaces <*> unit 0 family mpls

2. Configure IPv4 and IPv6 on the fabric-facing interfaces:

   [edit]
   user@R1# set chassis network-services enhanced-ip
   user@R1# set interfaces apply-groups int-global
   user@R1# set interfaces xe-1/0/0 unit 0 family inet address 192.168.14.18/30
   user@R1# set interfaces xe-1/0/0 unit 0 family inet6 address 2001:db8:2001:55::2/126
   user@R1# set interfaces xe-1/0/1 unit 0 family inet address 192.168.14.26/30
   user@R1# set interfaces xe-1/0/1 unit 0 family inet6 address 2001:db8:2001:57::2/126
   user@R1# set interfaces xe-1/0/2 unit 0 family inet address 192.168.14.152/30
   user@R1# set interfaces xe-1/0/2 unit 0 family inet6 address 2001:db8:2001:51::2/126
   user@R1# set interfaces xe-1/0/3 unit 0 family inet address 192.168.14.16/30
   user@R1# set interfaces xe-1/0/3 unit 0 family inet6 address 2001:db8:2001:53::2/126

3. Configure the interfaces towards the neighboring core router and the Internet:

   [edit]
   user@R1# set interfaces xe-1/3/1 unit 0 family inet address 192.168.15.5/30
   user@R1# set interfaces xe-1/3/2 unit 0 family inet address 192.168.15.9/30
   user@R1# set interfaces xe-1/3/3 unit 0 family inet address 192.168.15.1/30

4. Configure the loopback and management interfaces:

   [edit]
   user@R1# set interfaces lo0 unit 0 family inet address 10.0.23.1/32
   user@R1# set interfaces fxp0 unit 0 family inet address 10.94.191.55/24

5. Configure static routes and routing options:

   [edit]
   user@R1# set routing-options static route 10.94.63.252/32 next-hop 10.94.191.254
   user@R1# set routing-options static route 10.94.63.253/32 next-hop 10.94.191.254
   user@R1# set routing-options router-id 10.0.23.1
   user@R1# set routing-options autonomous-system 65000

6. Configure an IBGP session with the neighboring core router:

   [edit]
   user@R1# set protocols bgp multihop
   user@R1# set protocols bgp group to-r2-PE type internal
   user@R1# set protocols bgp group to-r2-PE local-address 10.0.23.1
   user@R1# set protocols bgp group to-r2-PE family inet vpn unicast
   user@R1# set protocols bgp group to-r2-PE bfd-liveness-detection minimum-interval 250
   user@R1# set protocols bgp group to-r2-PE multipath
   user@R1# set protocols bgp group to-r2-PE neighbor 10.0.23.2

7. Configure OSPF:

   [edit]
   user@R1# set protocols ospf traffic-engineering shortcuts
8. Configure RSVP and MPLS, including an LSP to the neighboring core router:

```
[edit]
user@R1# set protocols rsvp interface all link-protection
user@R1# set protocols mpls interface all
user@R1# set protocols mpls label-switched-path to-r2 backup
user@R1# set protocols mpls label-switched-path to-r2 to 10.0.23.2
user@R1# set protocols mpls label-switched-path to-r2 standby
user@R1# set protocols mpls label-switched-path to-r2 link-protection
```

9. Configure routing policies to enable reachability to the loopback interfaces of the other devices in the IP fabric:

```
[edit]
user@R1# set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.16.0/28 orlonger
user@R1# set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.22.0/28 orlonger
user@R1# set policy-options policy-statement receive-loopbacks term loopbacks from route-filter 10.0.23.0/28 orlonger
user@R1# set policy-options policy-statement receive-loopbacks term loopbacks then accept
user@R1# set policy-options policy-statement advertise-loopbacks term loopback from protocol direct
user@R1# set policy-options policy-statement advertise-loopbacks term loopback from route-filter 10.0.23.0/28 orlonger
user@R1# set policy-options policy-statement advertise-loopbacks term loopback then next-hop self
user@R1# set policy-options policy-statement advertise-loopbacks term loopback then accept
```

10. Configure per-flow load balancing:

```
[edit]
user@R1# set policy-options policy-statement pfe-lb then load-balance per-packet
user@R1# set routing-options forwarding-table export pfe-lb
user@R1# set forwarding-options hash-key family inet layer-3
user@R1# set forwarding-options hash-key family inet layer-4
```

11. Configure LLDP:

```
[edit]
user@R1# set protocols lldp interface all
user@R1# set protocols lldp interface fxp0 disable
```

12. Configure SNMP and event options:

```
[edit]
user@R1# set snmp community public authorization read-write
user@R1# set snmp trap-group networkdirector_trap_group version v2
user@R1# set snmp trap-group networkdirector_trap_group destination-port 10162
user@R1# set snmp trap-group networkdirector_trap_group categories authentication
user@R1# set snmp trap-group networkdirector_trap_group categories link
user@R1# set snmp trap-group networkdirector_trap_group categories services
user@R1# set snmp trap-group networkdirector_trap_group targets 10.94.63.253
user@R1# set snmp trap-group networkdirector_trap_group targets 10.94.63.252
user@R1# set event-options policy target_add_test events snmpd_trap_target_add_notice
user@R1# set event-options policy target_add_test then raise-trap
13. Configure a VRF instance to provide connectivity to the fabric devices:
   [edit]
   user@R1# set routing-instances RED-vpn instance-type vrf

14. Add the fabric-facing interfaces to the routing instance:
   [edit]
   user@R1# set routing-instances RED-vpn interface xe-1/0/0.0
   user@R1# set routing-instances RED-vpn interface xe-1/0/1.0
   user@R1# set routing-instances RED-vpn interface xe-1/0/2.0
   user@R1# set routing-instances RED-vpn interface xe-1/0/3.0

15. Configure a route distinguisher and VRF target:
   [edit]
   user@R1# set routing-instances RED-vpn route-distinguisher 65001:1
   user@R1# set routing-instances RED-vpn vrf-target target:65001:1

16. Within the routing instance, configure IPv4 and IPv6 EBGP sessions with each fabric device:
   [edit]
   user@R1# set routing-instances RED-vpn protocols bgp import receive-loopbacks
   user@R1# set routing-instances RED-vpn protocols bgp export advertise-loopbacks
   user@R1# set routing-instances RED-vpn protocols bgp bfd-liveness-detection minimum-interval 250
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC local-as 420006501
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC multipath multiple-as
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.1
description Fabric-sw01
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.1
target peer-as 420005501
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.9
description Fabric-sw02
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.9
target peer-as 420005502
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.17
description Fabric-sw03
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.17
target peer-as 420005503
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.25
description Fabric-sw04
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC neighbor 192.168.14.25
target peer-as 420005504
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 local-as 420006501
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 multipath multiple-as
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:51::1 peer-as 420005501
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:53::1 peer-as 420005502
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:55::1 peer-as 420005503
   user@R1# set routing-instances RED-vpn protocols bgp group FABRIC-IPV6 neighbor 2001:db8:2001:57::1 peer-as 420005504
Configuring Additional Features for the SaaS Solution

This section describes how to configure additional elements for the SaaS solution. These features are optional, and the sample configurations below are intended as a general guide, to be customized as necessary to fit your environment.

- Configuring SNMPv3 on page 45
- Configuring BMPv3 on page 46
- Configuring Device (Routing Engine) Protection on page 46
- Configuring Remote Port Mirroring on page 51
- Configuring Storm Control on page 52
- Configuring Class of Service on page 52

Configuring SNMPv3

This solution uses SNMP version 3 (SNMPv3), which supports authentication and encryption. SNMPv3 uses the user-based security model (USM) for message security and the view-based access control model (VACM) for access control. USM specifies authentication and encryption. VACM specifies access-control rules.

To quickly configure SNMPv3, enter the following representative configuration statements on each device:

```bash
## SNMPv3 ##
set groups SNMP-config snmp v3 usm local-engine user snmp authentication-sha
authentication-key "$ABC123$"
set groups SNMP-config snmp v3 usm local-engine user snmp privacy-des privacy-key
"$ABC123$"
set groups SNMP-config snmp v3 vacm security-to-group security-model usm
security-name snmp group view-all
set groups SNMP-config snmp v3 vacm access group view-all default-context-prefix
security-model usm security-level privacy read-view view-all
set groups SNMP-config snmp v3 vacm access group view-all default-context-prefix
security-model usm security-level privacy notify-view view-all
set groups SNMP-config snmp v3 target-address ta1 address 10.94.63.250
timeout 1
retry-count 0
tag-list router1
target-parameters tp1
set groups SNMP-config snmp v3 target-address ta2 address 10.94.63.251
timeout 1
retry-count 0
tag-list router2
target-parameters tp1
set groups SNMP-config snmp v3 target-parameters tp1 parameters
message-processing-model v3
security-model usm
security-name snmp
notify-filter nf1
set groups SNMP-config snmp v3 target-parameters tp1 notify SPACE_TRAPS type trap
```

Copyright © 2016, Juniper Networks, Inc.
Configuring BMPv3

The BGP Monitoring Protocol (BMP) allows a Junos device to send BGP route information to a monitoring application on a separate device. This solution uses BMP version 3 (BMPv3).

To quickly configure BMPv3, enter the following representative configuration statements on each device:

```plaintext
## BMPv3 ##
set groups BMP-config routing-options bmp connection-mode active
set groups BMP-config routing-options bmp monitor enable
set groups BMP-config routing-options bmp route-monitoring pre-policy
set groups BMP-config routing-options bmp station-address 10.94.63.193
set groups BMP-config routing-options bmp station-port 11019
set groups BMP-config routing-options bmp statistics-timeout 30
set groups BMP-config routing-options bmp station BMP-Server
## APPLY GROUP ##
set apply-groups BMP-config
```

**NOTE:** For more information on BMPv3, see Configuring BGP Monitoring Protocol Version 3.

Configuring Device (Routing Engine) Protection

To protect a Junos device from unwanted traffic and attacks, you can apply an input filter to the loopback interface. This filter can also be a useful way to count inbound (wanted or unwanted) traffic.

To quickly configure a firewall filter for the loopback interface, enter the following representative configuration statements on each device:

```plaintext
## FIREWALL FILTER ##
set groups RE-Protect-config interfaces lo0 unit 0 family inet filter input FF-PROTECT-RE
```
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 01-DISCARD-ICMP-FIRST-FRAGMENT from first-fragment
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 01-DISCARD-ICMP-FIRST-FRAGMENT from protocol icmp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 01-DISCARD-ICMP-FIRST-FRAGMENT then count 01-DISCARD-ICMP-FIRST-FRAGMENT
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 01-DISCARD-ICMP-FIRST-FRAGMENT then discard
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 02-DISCARD-NEXT-FRAG from is-fragment
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 02-DISCARD-NEXT-FRAG from protocol icmp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 02-DISCARD-NEXT-FRAG then count 02-DISCARD-NEXT-FRAG
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 02-DISCARD-NEXT-FRAG then discard
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 03-ACCEPT-ICMP-POLICER from protocol icmp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 03-ACCEPT-ICMP-POLICER from icmp-type echo-request
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 03-ACCEPT-ICMP-POLICER from icmp-type echo-reply
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 03-ACCEPT-ICMP-POLICER from icmp-type unreachable
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 03-ACCEPT-ICMP-POLICER from icmp-type time-exceeded
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 03-ACCEPT-ICMP-POLICER then policer POLICER-ICMP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 03-ACCEPT-ICMP-POLICER then count 03-ACCEPT-ICMP-POLICER
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 03-ACCEPT-ICMP-POLICER then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 04-ACCEPT-BGP from source-prefix-list PL-BGP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 04-ACCEPT-BGP from protocol tcp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 04-ACCEPT-BGP from port bgp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 04-ACCEPT-BGP then count 04-ACCEPT-BGP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 04-ACCEPT-BGP then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 05-ACCEPT-BGP-VRF from source-prefix-list PL-BGP-VRF
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 05-ACCEPT-BGP-VRF from protocol tcp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 05-ACCEPT-BGP-VRF from port bgp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 05-ACCEPT-BGP-VRF then count 05-ACCEPT-BGP-VRF
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 05-ACCEPT-BGP-VRF then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 06-ACCEPT-PIM from protocol pim
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
 06-ACCEPT-PIM then count 06-ACCEPT-PIM
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
06-ACCEPT-PIM then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
07-ACCEPT-SNMP from source-prefix-list PL-SNMP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
07-ACCEPT-SNMP from protocol udp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
07-ACCEPT-SNMP from destination-port snmp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
07-ACCEPT-SNMP then policer POLICER-SNMP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
07-ACCEPT-SNMP then count 07-ACCEPT-SNMP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
07-ACCEPT-SNMP then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
08-ACCEPT-OSPF from source-prefix-list PL-OSPF
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
08-ACCEPT-OSPF from protocol ospf
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
08-ACCEPT-OSPF from protocol udp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
08-ACCEPT-OSPF then count 08-ACCEPT-OSPF
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
08-ACCEPT-OSPF then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
09-ACCEPT-LDP from source-prefix-list PL-LDP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
09-ACCEPT-LDP from protocol tcp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
09-ACCEPT-LDP from protocol udp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
09-ACCEPT-LDP from port ldp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
09-ACCEPT-LDP then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
09-ACCEPT-LDP then count 09-ACCEPT-LDP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
09-ACCEPT-LDP then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
10-ACCEPT-NTP from source-prefix-list PL-NTP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
10-ACCEPT-NTP from protocol udp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
10-ACCEPT-NTP from source-port ntp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
10-ACCEPT-NTP then policer POLICER-NTP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
10-ACCEPT-NTP then count 10-ACCEPT-NTP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
10-ACCEPT-NTP then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
11-ACCEPT-DNS from source-prefix-list PL-DNS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
11-ACCEPT-DNS from protocol udp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
11-ACCEPT-DNS from source-port domain
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
11-ACCEPT-DNS then policer POLICER-DNS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
11-ACCEPT-DNS then count 11-ACCEPT-DNS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
11-ACCEPT-DNS then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
12-ACCEPT-SSH from source-prefix-list PL-SSH-CLIENTS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
12-ACCEPT-SSH from protocol tcp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
12-ACCEPT-SSH from destination-port ssh
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
12-ACCEPT-SSH then count 12-ACCEPT-SSH
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
12-ACCEPT-SSH then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
13-ACCEPT-SSH-OUTGOING from source-prefix-list PL-SSH-DESTINATION
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
13-ACCEPT-SSH-OUTGOING from protocol tcp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
13-ACCEPT-SSH-OUTGOING from source-port ssh
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
13-ACCEPT-SSH-OUTGOING then count 13-ACCEPT-SSH-OUTGOING
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
13-ACCEPT-SSH-OUTGOING then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
14-ACCEPT-RADIUS from source-prefix-list PL-RADIUS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
14-ACCEPT-RADIUS from protocol udp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
14-ACCEPT-RADIUS from source-port 1812
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
14-ACCEPT-RADIUS from source-port 1813
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
14-ACCEPT-RADIUS then policer POLICER-RADIUS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
14-ACCEPT-RADIUS then count 14-ACCEPT-RADIUS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
14-ACCEPT-RADIUS then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
15-ACCEPT-TACACS from source-prefix-list PL-TACACS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
15-ACCEPT-TACACS from protocol tcp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
15-ACCEPT-TACACS from source-port tacacs
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
15-ACCEPT-TACACS from tcp-established
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
15-ACCEPT-TACACS then policer POLICER-TACACS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
15-ACCEPT-TACACS then count 15-ACCEPT-TACACS
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
15-ACCEPT-TACACS then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
16-ACCEPT-IGMP from protocol igmp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
16-ACCEPT-IGMP then count 16-ACCEPT-IGMP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term
16-ACCEPT-IGMP then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 17-ACCEPT-BFD from protocol udp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 17-ACCEPT-BFD from destination-port 3784
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 17-ACCEPT-BFD from destination-port 4784
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 17-ACCEPT-BFD then count 17-ACCEPT-BFD
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 17-ACCEPT-BFD then accept
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 18-ACCEPT-DHCP from protocol udp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 18-ACCEPT-DHCP from port dhcp
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 18-ACCEPT-DHCP then count 18-ACCEPT-DHCP
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 18-ACCEPT-DHCP then accept
####
## ADD ADDITIONAL TERMS HERE FOR ADDITIONAL TRAFFIC TYPES ##
## THAT SHOULD BE PERMITTED TO REACH THE RE ##
####
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 99-DISCARD-EVERYTHING-ELSE then count 99-DISCARD-EVERYTHING-ELSE
set groups RE-Protect-config firewall family inet filter FF-PROTECT-RE term 99-DISCARD-EVERYTHING-ELSE then discard
## PREFIX LISTS ##
set groups RE-Protect-config policy-options prefix-list PL-BGP apply-path "protocols bgp group <*> neighbor <*>"
set groups RE-Protect-config policy-options prefix-list PL-SNMP apply-path "snmp community <*> clients <*>"
set groups RE-Protect-config policy-options prefix-list PL-NTP apply-path "system ntp server <*>"
set groups RE-Protect-config policy-options prefix-list PL-DNS apply-path "system name-server <*>"
set groups RE-Protect-config policy-options prefix-list PL-BGP-VRF apply-path "routing-instances <*> protocols bgp group <*> neighbor <*>"
set groups RE-Protect-config policy-options prefix-list PL-RADIUS apply-path "system radius-server <*>"
set groups RE-Protect-config policy-options prefix-list PL-TACACS apply-path "system tacplus-server <*>"
set groups RE-Protect-config policy-options prefix-list PL-SSH-CLIENTS 172.16.0.0/12
set groups RE-Protect-config policy-options prefix-list PL-SSH-CLIENTS 192.168.0.0/16
set groups RE-Protect-config policy-options prefix-list PL-SSH-CLIENTS 10.10.0.0/16
set groups RE-Protect-config policy-options prefix-list PL-SSH-DESTINATION 0.0.0.0/0
set groups RE-Protect-config policy-options prefix-list PL-LDP 0.0.0.0/0
set groups RE-Protect-config policy-options prefix-list PL-OSPF 0.0.0.0/0
## APPLY FIREWALL FILTER TO LOOPBACK INTERFACE##
set groups RE-Protect-config interfaces lo0 unit 0 family inet filter input FF-PROTECT-RE
## APPLY GROUP ##
set apply-groups RE-Protect-config

NOTE: For more information on using firewall filters to protect a device, see Overview of Firewall Filters and Applying Firewall Filters to Interfaces.
Configuring Remote Port Mirroring

Port mirroring copies packets entering or exiting a port, or entering a VLAN, and sends the copies to either a local interface for local monitoring or to a remote monitoring station. Use port mirroring to send traffic to applications that analyze traffic for purposes such as monitoring compliance, enforcing policies, detecting intrusions, monitoring and predicting traffic patterns, correlating events, and so on.

There are two port mirroring instance types:

- Analyzer instance—useful for mirroring all traffic transiting an interface or VLAN.
- Port-mirroring instance—useful for controlling which types of traffic should be mirrored.

To quickly configure remote port mirroring, enter the following representative configuration statements on each device:

### Analyzer Instance
This option copies all inbound traffic arriving at interface xe-0/0/47 and mirrors it through a GRE-encapsulated tunnel to a remote analyzer at 10.100.10.1.

```
## OPTION 1: ANALYZER INSTANCE ##
set groups RemPortMon-Al-config forwarding-options analyzer RemPortMon-GRE input ingress interface xe-0/0/47.0
set groups RemPortMon-Al-config forwarding-options analyzer RemPortMon-GRE output ip-address 10.100.10.1
## APPLY GROUP ##
set apply-groups RemPortMon-Al-config
```

### Port-Mirroring Instance
This option copies only inbound traffic arriving at interface xe-0/0/47 from a host at 10.1.1.1, and mirrors it through a GRE-encapsulated tunnel to a remote analyzer at 10.100.10.1.

```
## OPTION 2: PORT-MIRRORING INSTANCE ##
set groups RemPortMon-PM-config interfaces xe-0/0/47 unit 0 family inet filter input RemPortMon
set groups RemPortMon-PM-config firewall family inet filter RemPortMon term 1 from source-address 10.1.1.1/32
count count1
set groups RemPortMon-PM-config firewall family inet filter RemPortMon term 1 then port-mirror-instance RemPortMon-GRE
set groups RemPortMon-PM-config firewall family inet filter RemPortMon term 1 then accept
set groups RemPortMon-PM-config forwarding-options port-mirroring instance RemPortMon-GRE family inet output ip-address 10.100.10.1
## APPLY GROUP ##
set apply-groups RemPortMon-PM-config
```

**NOTE:** For more information on using remote port mirroring, see Understanding Port Mirroring and Configuring Port Mirroring for Remote Analysis.
Configuring Storm Control

Storm control helps to prevent network outages caused by broadcast storms. Storm control enables a device to monitor traffic levels and take a specified action when a specified traffic level (the storm control level) is exceeded. You can configure devices to drop broadcast and unknown unicast packets, shut down interfaces, or temporarily disable interfaces when the storm control level is exceeded.

To quickly configure storm control, enter the following representative configuration statements on each device:

```plaintext
## STORM CONTROL ##
set groups StrmCtrl-config forwarding-options storm-control-profiles storm-protect all
bandwidth-percentage 5
set groups StrmCtrl-config forwarding-options storm-control-profiles storm-protect
action-shutdown
set groups StrmCtrl-config interfaces xe-0/0/47.0 family ethernet-switching storm-control
storm-protect
## APPLY GROUP ##
set apply-groups StrmCtrl-config
```

NOTE: For more information on using storm control, see Understanding Storm Control and Configuring Storm Control to Prevent Network Outages.

Configuring Class of Service

Junos OS class of service (CoS) enables you to divide traffic into classes and set various levels of throughput and packet loss when congestion occurs.

To quickly configure CoS, enter the following representative configuration statements on each device:

```plaintext
## MULTIFIELD CLASSIFIERS ##
set groups CoS-config firewall family inet filter mf-classifier term App-1 from
source-address 10.1.1.1/32
set groups CoS-config firewall family inet filter mf-classifier term App-1 from protocol
tcp
set groups CoS-config firewall family inet filter mf-classifier term App-1 from source-port
ftp
set groups CoS-config firewall family inet filter mf-classifier term App-1 then loss-priority
high
set groups CoS-config firewall family inet filter mf-classifier term App-1 then forwarding-class App-1
set groups CoS-config firewall family inet filter mf-classifier term App-1 then policer
POLICER-APP1
set groups CoS-config firewall family inet filter mf-classifier term App-2 from
source-address 10.1.1.1/32
set groups CoS-config firewall family inet filter mf-classifier term App-2 from protocol
tcp
set groups CoS-config firewall family inet filter mf-classifier term App-2 from source-port
http
```
set groups CoS-config firewall family inet filter mf-classifier term App-2 then loss-priority low
set groups CoS-config firewall family inet filter mf-classifier term App-2 then forwarding-class App-2
set groups CoS-config firewall family inet filter mf-classifier term App-2 then policer POLICER-APP2
set groups CoS-config firewall family inet filter mf-classifier term App-3 from source-address 10.1.1.1/32
set groups CoS-config firewall family inet filter mf-classifier term App-3 from protocol tcp
set groups CoS-config firewall family inet filter mf-classifier term App-3 from source-port pop3
set groups CoS-config firewall family inet filter mf-classifier term App-3 then loss-priority low
set groups CoS-config firewall family inet filter mf-classifier term App-3 then forwarding-class App-3
set groups CoS-config firewall family inet filter mf-classifier term App-3 then policer POLICER-APP3
set groups CoS-config firewall family inet filter mf-classifier term accept-all-else then accept
## APPLY MF CLASSIFIERS TO INTERFACES ##
set groups CoS-config interfaces xe-0/0/46 unit 0 family inet filter input mf-classifier
set groups CoS-config interfaces xe-0/0/47 unit 0 family inet filter input mf-classifier
## FORWARDING CLASSES ##
set groups CoS-config class-of-service forwarding-classes class BE-data queue-num 0
set groups CoS-config class-of-service forwarding-classes class App-1 queue-num 1
set groups CoS-config class-of-service forwarding-classes class App-2 queue-num 2
set groups CoS-config class-of-service forwarding-classes class App-3 queue-num 5
set groups CoS-config class-of-service forwarding-classes class NC queue-num 3
## TRAFFIC CONTROL PROFILES ##
set groups CoS-config class-of-service traffic-control-profiles tcp-anycast scheduler-map sm-anycast
set groups CoS-config class-of-service traffic-control-profiles tcp-anycast guaranteed-rate percent 100
## FORWARDING CLASS SETS ##
set groups CoS-config class-of-service forwarding-class-sets fc-set-anycast class BE-data
set groups CoS-config class-of-service forwarding-class-sets fc-set-anycast class App-1
set groups CoS-config class-of-service forwarding-class-sets fc-set-anycast class App-2
set groups CoS-config class-of-service forwarding-class-sets fc-set-anycast class App-3
## APPLY COS ELEMENTS TO INTERFACES ##
set groups CoS-config class-of-service interfaces xe-* forwarding-class-set fc-set-anycast output-traffic-control-profile tcp-anycast
set groups CoS-config class-of-service interfaces xe-* rewrite-rules dscp dscp-remark
set groups CoS-config class-of-service interfaces et-* forwarding-class-set fc-set-anycast output-traffic-control-profile tcp-anycast
set groups CoS-config class-of-service interfaces et-* rewrite-rules dscp dscp-remark
## REWRITE RULES ##
set groups CoS-config class-of-service rewrite-rules dscp dscp-remark forwarding-class App-1 loss-priority low code-point 000001
set groups CoS-config class-of-service rewrite-rules dscp dscp-remark forwarding-class App-2 loss-priority low code-point 000010
set groups CoS-config class-of-service rewrite-rules dscp dscp-remark forwarding-class App-3 loss-priority low code-point 000011
set groups CoS-config class-of-service rewrite-rules dscp dscp-remark forwarding-class BE-data loss-priority low code-point 000000
## SCHEDULER MAP ##
set groups CoS-config class-of-service scheduler-maps sm-anycast forwarding-class App-1 scheduler App-1-sc

set groups CoS-config class-of-service scheduler-maps sm-anycast forwarding-class App-2 scheduler App-2-sc

set groups CoS-config class-of-service scheduler-maps sm-anycast forwarding-class App-3 scheduler App-3-sc

set groups CoS-config class-of-service scheduler-maps sm-anycast forwarding-class BE-data scheduler BE-data-sc

## SCHEDULERS ##

set groups CoS-config class-of-service schedulers App-1-sc transmit-rate percent 1
set groups CoS-config class-of-service schedulers App-1-sc shaping-rate percent 50
set groups CoS-config class-of-service schedulers App-1-sc buffer-size percent 20
set groups CoS-config class-of-service schedulers App-1-sc priority low
set groups CoS-config class-of-service schedulers App-2-sc transmit-rate percent 59
set groups CoS-config class-of-service schedulers App-2-sc shaping-rate percent 60
set groups CoS-config class-of-service schedulers App-2-sc buffer-size percent 10
set groups CoS-config class-of-service schedulers App-2-sc priority low
set groups CoS-config class-of-service schedulers App-3-sc transmit-rate percent 20
set groups CoS-config class-of-service schedulers App-3-sc shaping-rate percent 60
set groups CoS-config class-of-service schedulers App-3-sc buffer-size percent 30
set groups CoS-config class-of-service schedulers App-3-sc priority low
set groups CoS-config class-of-service schedulers BE-data-sc transmit-rate percent 20
set groups CoS-config class-of-service schedulers BE-data-sc priority low

## POLICERS ##

set groups CoS-config firewall policer POLICER-APP1 if-exceeding bandwidth-limit 5m
set groups CoS-config firewall policer POLICER-APP1 if-exceeding burst-size-limit 50k
set groups CoS-config firewall policer POLICER-APP1 then discard
set groups CoS-config firewall policer POLICER-APP2 if-exceeding bandwidth-limit 500k
set groups CoS-config firewall policer POLICER-APP2 if-exceeding burst-size-limit 50k
set groups CoS-config firewall policer POLICER-APP2 then discard
set groups CoS-config firewall policer POLICER-APP3 if-exceeding bandwidth-limit 50m
set groups CoS-config firewall policer POLICER-APP3 if-exceeding burst-size-limit 10k
set groups CoS-config firewall policer POLICER-APP3 then discard

## APPLY GROUP ##

set apply-groups CoS-config

---

### NOTE: For more information on using class of service, see [Traffic Management Feature Guide for QFX Series](#).###

---

**Verification**

Confirm that the SaaS IP fabric configuration is working properly.

- **Leaf: Verifying Interfaces** on page 55
- **Leaf: Verifying IPv4 EBGP Sessions** on page 56
- **Leaf: Verifying IPv6 EBGP Sessions** on page 56
- **Leaf: Verifying BFD Sessions** on page 57
- **Leaf: Verifying Server Access - Anycast** on page 57
- **Leaf: Verifying Server Access - Unicast** on page 58
Leaf: Verifying Server Access - Hybrid on page 59
Leaf: Verifying Server Load Balancing Using IPv4 Anycast - Packet Forwarding Engine Load Balancing on page 60
Leaf: Verifying Server Load Balancing Using IPv4 Anycast - BGP Load Balancing on page 62
Leaf: Verifying Server Load Balancing Using IPv4 Anycast - BFD Sessions on page 64
Leaf: Verifying IPv6 Anycast Reachability on page 65
Leaf: Verifying Server Load Balancing Using IPv6 Anycast - Packet Forwarding Engine Load Balancing on page 66
Spine: Verifying Interfaces on page 66
Spine: Verifying IPv4 EBGP Sessions on page 67
Spine: Verifying IPv6 EBGP Sessions on page 68
Spine: Verifying BFD Sessions on page 69
Spine: Verifying IPv6 Anycast Reachability on page 69
Fabric: Verifying Interfaces on page 70
Fabric: Verifying IPv4 EBGP Sessions on page 71
Fabric: Verifying IPv6 EBGP Sessions on page 72
Fabric: Verifying BFD Sessions on page 73
Fabric: Verifying IPv4 Anycast Reachability on page 73
Fabric: Verifying IPv6 Anycast Reachability on page 75
Core: Verifying Interfaces on page 76
Core: Verifying IPv4 BGP Sessions on page 77
Core: Verifying IPv6 EBGP Sessions on page 78
Core: Verifying BFD Sessions on page 79
Core: Verifying IPv4 Anycast Reachability on page 79
Core: Verifying Server Load Balancing Using IPv4 Anycast - Packet Forwarding Engine Load Balancing on page 80
Core: Verifying IPv6 Anycast Reachability on page 81
Additional Features: Verifying SNMP on page 82
Additional Features: Verifying BMP on page 84
Additional Features: Verifying Remote Port Mirroring on page 85
Additional Features: Verifying Class of Service on page 85

**Leaf: Verifying Interfaces**

**Purpose**
Verify the state of the spine-facing interfaces.
**Action**  Verify that the spine-facing interfaces (et-0/0/48 to et-0/0/51) are up:

```
user@Leaf-0> show interfaces terse et-*
Interface               Admin Link Proto    Local                 Remote
et-0/0/48               up    up    inet     192.168.11.1/31
et-0/0/48.0             up    up    inet6    2001:db8:2001:1::1/126
fe80::8aa2:5eff:fecc:cab3/64
et-0/0/49               up    up    inet     192.168.11.17/31
et-0/0/49.0             up    up    inet6    2001:db8:2001:2::1/126
fe80::8aa2:5eff:fecc:cab7/64
et-0/0/50               up    up    inet     192.168.11.33/31
et-0/0/50.0             up    up    inet6    2001:db8:2001:3::1/126
fe80::8aa2:5eff:fecc:cab/64
et-0/0/51               up    up    inet     192.168.11.49/31
et-0/0/51.0             up    up    inet6    2001:db8:2001:4::1/126
fe80::8aa2:5eff:fecc:cafb/64
```

**Meaning**  The spine-facing interfaces are functioning normally.

**Leaf: Verifying IPv4 EBGP Sessions**

**Purpose**  Verify the state of IPv4 EBGP sessions between the leaf and spine devices.

**Action**  Verify that IPv4 EBGP sessions are established:

```
user@Leaf-0> show bgp summary | match 192.168
192.168.11.0      420005000       9121       8816       0       3 1w1d 21:02:13
4/2053/2053/0        0/0/0/0
192.168.11.16     420005001      39280      37398       0       1 1w4d 14:28:02
4/2055/2055/0        0/0/0/0
192.168.11.32     420005002      30453      29419       0       2 1w2d 15:57:58
4/2054/2054/0        0/0/0/0
192.168.11.48     420005003      24204      23376       0       4 1w0d 16:37:50
2052/2054/2054/0     0/0/0/0
```

**Meaning**  The EBGP sessions are established and functioning correctly.

**Leaf: Verifying IPv6 EBGP Sessions**

**Purpose**  Verify the state of IPv6 EBGP sessions between the leaf and spine devices.
Action Verify that IPv6 EBGP sessions are established:

```
user@Leaf-0> show bgp summary | find 2001
2001:db8:2001:1::2         420005000       1208       1179       0      25
  9:21:16 Establ
    inet6.0: 0/0/0/0
2001:db8:2001:2::2         420005001       1938       1892       0      24
  15:01:17 Establ
    inet6.0: 0/0/0/0
2001:db8:2001:3::2         420005002        219        215       0      25
  1:41:16 Establ
    inet6.0: 0/0/0/0
2001:db8:2001:4::2         420005003       1209       1179       0      19
  9:21:12 Establ
    inet6.0: 0/0/0/0
```

Meaning The EBGP sessions are established and functioning correctly.

Leaf: Verifying BFD Sessions

Purpose Verify the state of BFD for the IPv4 BGP sessions between the leaf and spine devices.

Action Verify that BFD sessions are up:

```
user@Leaf-0> show bfd session | match 192.168
Detect   Transmit
192.168.11.0             Up        et-0/0/48.0    0.750     0.250        3
192.168.11.16            Up        et-0/0/49.0    0.750     0.250        3
192.168.11.32            Up        et-0/0/50.0    0.750     0.250        3
192.168.11.48            Up        et-0/0/51.0    0.750     0.250        3
```

Meaning The BFD sessions are established and functioning correctly.

Leaf: Verifying Server Access - Anycast

Purpose Verify the configuration of the anycast server access option.

Action 1. Verify that the server-facing interfaces are up:

```
user@Leaf-0> show interfaces terse | match 172.16
xe-0/0/0.0              up    up   inet     172.16.71.2/30
xe-0/0/1.0              up    up   inet     172.16.72.2/30
```

```
user@Leaf-0> show interfaces terse xe-0/0/46
Interface       Admin Link Proto    Local                 Remote
xe-0/0/46               up    up   inet  172.16.90.2/30
xe-0/0/46.0             up    up   inet6 2001:db8:2001:92::2/126
```

```
user@Leaf-0> show interfaces terse xe-0/0/47
```

### Interface Admin Link Proto Local Remote

<table>
<thead>
<tr>
<th>Interface</th>
<th>Admin</th>
<th>Link</th>
<th>Proto</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>xe-0/0/47</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>172.16.91.2/30</td>
<td>2001:db8:2001:93::2/126</td>
</tr>
<tr>
<td>xe-0/0/47.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>172.16.91.2/30</td>
<td>fe80::8aa2:5eff:fecc:cab2/64</td>
</tr>
</tbody>
</table>

2. Verify that EBGP sessions are established:

```
user@Leaf-0> show bgp summary | match 172.16
172.16.71.1       420008501      40788      46121       0       0 2w0d 3:52:00
0/2/2/0              0/0/0/0
172.16.72.1       420008502      40788      46118       0       0 2w0d 3:52:02
2/2/2/0              0/0/0/0
```

```
user@Leaf-0> show bgp summary | match 2001:9
2001:db8:2001:92::1        420008000       2115       2217       0       2
17:36:28 Establ inet6.0: 1/1/1/0
2001:db8:2001:93::1        420008001       5303       5561       0       1 1d
20:10:25 Establ inet6.0: 1/1/1/0
```

Meaning  The server-facing interfaces are up, the EBGP sessions are established, and anycast server access is functioning correctly.

### Leaf: Verifying Server Access - Unicast

#### Purpose
Verify the configuration of the unicast server access option.

#### Action
1. Verify that the server-facing interfaces are up:

```
user@Leaf-0> show interfaces terse | match xe- *
xe-0/0/0                up    up
xe-0/0/0.0              up    up   eth-switch
xe-0/0/1                up    up
xe-0/0/1.0              up    up   eth-switch
```

2. Verify that the IRB interface is up:

```
user@Leaf-0> show interfaces terse | match irb.1
irb                     up    up
irb.1                   up    up inet 172.16.64.1/27
```

3. Verify the VLAN configuration:

```
user@Leaf-0> show vlans detail SERVER
Routing instance: default-switch
VLAN Name: SERVER                       State: Active
Tag: 1
Internal index: 2, Generation Index: 2, Origin: Static
MAC aging time: 300 seconds
Layer 3 interface: irb.1
Interfaces:
   xe-0/0/0.0*, untagged, access
   xe-0/0/1.0*, untagged, access
```
Number of interfaces: Tagged 0, Untagged 2
Total MAC count: 2

Meaning  The server-facing and IRB interfaces are up, the VLAN is active, and unicast server access is functioning correctly.

Leaf: Verifying Server Access - Hybrid

Purpose  Verify the configuration of the hybrid server access option.

Action  1. Verify that the server-facing interfaces are up:

```
user@Leaf-0> show interfaces terse | match xe-*
exe-0/0/0                up    up
exe-0/0/0.0              up    up   eth-switch
exe-0/0/1                up    up
exe-0/0/1.0              up    up   eth-switch
```

2. Verify that the IRB interface is up:

```
user@Leaf-0> show interfaces terse | match irb
irb                     up    up
irb.100                 up    up   inet     172.16.73.2/24
```

3. Verify the VLAN configuration:

```
user@Leaf-0> show vlans detail hybrid
Routing instance: default-switch
VLAN Name: hybrid                         State: Active
Tag: 100
Internal index: 3, Generation Index: 3, Origin: Static
MAC aging time: 300 seconds
Layer 3 interface: irb.100
Interfaces:
  xe-0/0/0.0*,untagged,access
  xe-0/0/1.0*,untagged,access
Number of interfaces: Tagged 0, Untagged 2
Total MAC count: 2
```

4. Verify that EBGP sessions are established:

```
user@Leaf-0> show bgp summary | match 172.16
172.16.73.1       420006503     258960     281623       0       0 12w5d 21:55:50
0/2/2/0              0/0/0/0
172.16.73.3       420006504     258953     281624       0       0 12w5d 21:55:51
2/2/2/0              0/0/0/0
```

Meaning  The server-facing and IRB interfaces are up, the VLAN is active, the EBGP sessions are established, and hybrid server access is functioning correctly.
Purpose

For the server load balancing scenario, verify reachability and load sharing to the servers using the anycast IPv4 address.
**Action**
Verify that Packet Forwarding Engine load balancing is enabled to the anycast IPv4 address:

```
user@Leaf-0> show route forwarding-table destination 10.1.1.1
Routing table: default.inet
Internet:
   Destination        Type RtRef Next hop           Type Index    NhRef Netif
     10.1.1.1/32         user     0                    ulst   131070     6
```

Meaning  The forwarding table entry for route 10.1.1.1/32 has the type ulst (unilist, meaning the device can use multiple next hops), followed by the list of eligible next hops.

Purpose  Verify the state of IPv4 BGP load balancing between the leaf device and load-sharing servers.
Action

Verify that IPv4 EBGP sessions are established:

```
user@Leaf-0> show bgp summary | match 6700
172.16.152.2      420006700      39086      39952       0       0 1w5d 20:46:28
  5/48/48/0            0/0/0/0
172.16.153.2      420006700      39086      39962       0       0 1w5d 20:46:24
  5/48/48/0            0/0/0/0
172.16.154.2      420006700      39087      39963       0       0 1w5d 20:46:34
  5/48/48/0            0/0/0/0
172.16.155.2      420006700      39087      39962       0       0 1w5d 20:46:27
  5/48/48/0            0/0/0/0
172.16.156.2      420006700      39284      40190       0       0 1w5d 22:19:43
  5/48/48/0            0/0/0/0
172.16.157.2      420006700      39283      40191       0       0 1w5d 22:19:58
  5/48/48/0            0/0/0/0
172.16.158.2      420006700      39284      40198       0       0 1w5d 22:19:57
  5/48/48/0            0/0/0/0
172.16.159.2      420006700      39284      40198       0       0 1w5d 22:19:56
  5/48/48/0            0/0/0/0
172.16.160.2      420006700      39284      40198       0       0 1w5d 22:19:52
  5/48/48/0            0/0/0/0
172.16.161.2      420006700      39284      40208       0       0 1w5d 22:19:48
  5/48/48/0            0/0/0/0
172.16.162.2      420006700      39284      40208       0       0 1w5d 22:19:43
  5/48/48/0            0/0/0/0
172.16.163.2      420006700      39284      40302       0       0 1w5d 22:19:39
  5/48/48/0            0/0/0/0
172.16.164.2      420006700      39285      40297       0       0 1w5d 22:20:00
  5/48/48/0            0/0/0/0
172.16.165.2      420006700      39284      40287       0       0 1w5d 22:19:56
  5/48/48/0            0/0/0/0
172.16.166.2      420006700      39284      40283       0       0 1w5d 22:19:51
  5/48/48/0            0/0/0/0
172.16.167.2      420006700      39284      40269       0       0 1w5d 22:19:48
  5/48/48/0            0/0/0/0
172.16.168.2      420006700      39284      40210       0       0 1w5d 22:19:43
  5/48/48/0            0/0/0/0
172.16.169.2      420006700      39284      40199       0       0 1w5d 22:19:39
  5/48/48/0            0/0/0/0
172.16.170.2      420006700      39285      40208       0       0 1w5d 22:20:00
  5/48/48/0            0/0/0/0
172.16.171.2      420006700      39284      40200       0       0 1w5d 22:19:56
  5/48/48/0            0/0/0/0
172.16.172.2      420006700      39284      40199       0       0 1w5d 22:19:51
  5/48/48/0            0/0/0/0
172.16.173.2      420006700      39284      40198       0       0 1w5d 22:19:48
  5/48/48/0            0/0/0/0
172.16.174.2      420006700      39284      40198       0       0 1w5d 22:19:43
  5/48/48/0            0/0/0/0
172.16.175.2      420006700      39284      40197       0       0 1w5d 22:19:39
  5/48/48/0            0/0/0/0
172.16.176.2      420006700      39284      40194       0       0 1w5d 22:20:00
  5/48/48/0            0/0/0/0
172.16.177.2      420006700      39283      40183       0       0 1w5d 22:19:58
  5/48/48/0            0/0/0/0
172.16.178.2      420006700      39283      40192       0       0 1w5d 22:19:58
  5/48/48/0            0/0/0/0
172.16.179.2      420006700      39284      40181       0       0 1w5d 22:19:48
  5/48/48/0            0/0/0/0
172.16.180.2      420006700      39284      40181       0       0 1w5d 22:19:43
  5/48/48/0            0/0/0/0
```
<table>
<thead>
<tr>
<th>Device</th>
<th>IP Address</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>State</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/48/48/0</td>
<td>172.16.181.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:19:39</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.182.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:20:00</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.183.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:19:56</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.184.2</td>
<td>420006700</td>
<td>39283</td>
<td>0</td>
<td>0 1w5d 22:19:54</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.185.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:20:02</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.186.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:19:43</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.187.2</td>
<td>420006700</td>
<td>39285</td>
<td>0</td>
<td>0 1w5d 22:20:02</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.188.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:20:00</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.189.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:19:56</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.190.2</td>
<td>420006700</td>
<td>39283</td>
<td>0</td>
<td>0 1w5d 22:19:54</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.191.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:19:47</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.192.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:19:43</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.193.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:20:02</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.194.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:20:00</td>
</tr>
<tr>
<td>5/48/48/0</td>
<td>172.16.195.2</td>
<td>420006700</td>
<td>39284</td>
<td>0</td>
<td>0 1w5d 22:20:02</td>
</tr>
</tbody>
</table>

**Meaning**  
The EBGPs sessions are established and functioning correctly.

**Leaf: Verifying Server Load Balancing Using IPv4 Anycast - BFD Sessions**

**Purpose**  
Verify the state of BFD for the IPv4 BGP connections between the leaf device and load-sharing servers.
### Action
Verify that BFD sessions are up:

```
user@Leaf-0> show bfd session | match 172.16.1
```

<table>
<thead>
<tr>
<th>Address</th>
<th>State</th>
<th>Interface</th>
<th>Detect Time</th>
<th>Transmit Interval</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.16.152.2</td>
<td>Up</td>
<td>xe-0/0/2.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.153.2</td>
<td>Up</td>
<td>xe-0/0/3.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.154.2</td>
<td>Up</td>
<td>xe-0/0/4.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.155.2</td>
<td>Up</td>
<td>xe-0/0/5.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.156.2</td>
<td>Up</td>
<td>xe-0/0/6.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.157.2</td>
<td>Up</td>
<td>xe-0/0/7.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.158.2</td>
<td>Up</td>
<td>xe-0/0/8.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.159.2</td>
<td>Up</td>
<td>xe-0/0/9.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.160.2</td>
<td>Up</td>
<td>xe-0/0/10.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.161.2</td>
<td>Up</td>
<td>xe-0/0/11.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.162.2</td>
<td>Up</td>
<td>xe-0/0/12.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.163.2</td>
<td>Up</td>
<td>xe-0/0/13.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.164.2</td>
<td>Up</td>
<td>xe-0/0/14.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.165.2</td>
<td>Up</td>
<td>xe-0/0/15.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.166.2</td>
<td>Up</td>
<td>xe-0/0/16.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.167.2</td>
<td>Up</td>
<td>xe-0/0/17.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.168.2</td>
<td>Up</td>
<td>xe-0/0/18.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.169.2</td>
<td>Up</td>
<td>xe-0/0/19.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.170.2</td>
<td>Up</td>
<td>xe-0/0/20.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.171.2</td>
<td>Up</td>
<td>xe-0/0/21.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.172.2</td>
<td>Up</td>
<td>xe-0/0/22.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.173.2</td>
<td>Up</td>
<td>xe-0/0/23.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.174.2</td>
<td>Up</td>
<td>xe-0/0/24.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.175.2</td>
<td>Up</td>
<td>xe-0/0/25.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.176.2</td>
<td>Up</td>
<td>xe-0/0/26.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.177.2</td>
<td>Up</td>
<td>xe-0/0/27.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.178.2</td>
<td>Up</td>
<td>xe-0/0/28.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.179.2</td>
<td>Up</td>
<td>xe-0/0/29.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.180.2</td>
<td>Up</td>
<td>xe-0/0/30.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.181.2</td>
<td>Up</td>
<td>xe-0/0/31.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.182.2</td>
<td>Up</td>
<td>xe-0/0/32.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.183.2</td>
<td>Up</td>
<td>xe-0/0/33.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.184.2</td>
<td>Up</td>
<td>xe-0/0/34.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.185.2</td>
<td>Up</td>
<td>xe-0/0/35.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.186.2</td>
<td>Up</td>
<td>xe-0/0/36.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.187.2</td>
<td>Up</td>
<td>xe-0/0/37.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.188.2</td>
<td>Up</td>
<td>xe-0/0/38.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.189.2</td>
<td>Up</td>
<td>xe-0/0/39.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.190.2</td>
<td>Up</td>
<td>xe-0/0/40.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.191.2</td>
<td>Up</td>
<td>xe-0/0/41.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.192.2</td>
<td>Up</td>
<td>xe-0/0/42.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.193.2</td>
<td>Up</td>
<td>xe-0/0/43.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.194.2</td>
<td>Up</td>
<td>xe-0/0/44.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>172.16.195.2</td>
<td>Up</td>
<td>xe-0/0/45.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
</tbody>
</table>

### Meaning
The BFD sessions are established and functioning correctly.

### Purpose
Verify reachability to the servers using the anycast IPv6 address.

#### Leaf: Verifying IPv6 Anycast Reachability
**Action**  Verify that the leaf device has an entry for the 2001:db8:2000::/64 route:

```
user@Leaf-0> show route 2001:db8:2000::/64
```

inet6.0: 20 destinations, 26 routes (20 active, 0 holddown, 0 hidden)
Restart Complete
+ = Active Route, - = Last Active, * = Both

```
2001:db8:2000::/64   *[BGP/170] 1d 20:06:17, localpref 100, from
> to 2001:db8:2001:92::1 via xe-0/0/46.0
  to 2001:db8:2001:93::1 via xe-0/0/47.0
```

**Meaning**  The leaf device has reachability to the servers using the anycast IPv6 address.

**Leaf: Verifying Server Load Balancing Using IPv6 Anycast - Packet Forwarding Engine Load Balancing**

**Purpose**  Verify load sharing to the servers using the anycast IPv6 address.

**Action**  Verify that Packet Forwarding Engine load balancing is enabled to the anycast IPv6 address:

```
user@Leaf-0> show route forwarding-table destination 2001:db8:2000::/64
Routing table: default.inet6
Internet6:
  Destination          Type RtRef Next hop           Type Index NhRef Netif
  2001:db8:2000::/64   user     0                      ulst   131070     2
                     2001:db8:2001:92::1 ucst     2019     3 xe-0/0/46.0
                     2001:db8:2001:93::1 ucst     2017     3 xe-0/0/47.0
Routing table: __master.anon__.inet6
Internet6:
  Destination        Type RtRef Next hop           Type Index NhRef Netif
  default            perm     0                    rjct     1684
```

**Meaning**  The forwarding table entry for route 2001:db8:2000::/64 has the type `ulst` (unilist, meaning the device can use multiple next hops), followed by the list of eligible next hops.

**Spine: Verifying Interfaces**

**Purpose**  Verify the state of the leaf-facing and fabric-facing interfaces.
Action Verify that the leaf-facing interfaces (et-0/0/0 to et-0/0/3) and fabric-facing interfaces (et-0/2/0 to et-0/2/3) are up:

user@Spine-0> show interfaces terse | match et-

<table>
<thead>
<tr>
<th>Interface</th>
<th>Admin</th>
<th>Link</th>
<th>Proto</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>et-0/0/0</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>192.168.11.0/31</td>
<td>2001:db8:2001:1::2/126</td>
</tr>
<tr>
<td>et-0/0/0.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.11.0/31</td>
<td>2001:db8:2001:1::2/126</td>
</tr>
<tr>
<td>et-0/0/1</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.11.2/31</td>
<td>2001:db8:2001:5::2/126</td>
</tr>
<tr>
<td>et-0/0/1.0</td>
<td>up</td>
<td>up</td>
<td>inet6</td>
<td>192.168.11.2/31</td>
<td>2001:db8:2001:5::2/126</td>
</tr>
<tr>
<td>et-0/0/2</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.11.4/31</td>
<td>2001:db8:2001:9::2/126</td>
</tr>
<tr>
<td>et-0/0/2.0</td>
<td>up</td>
<td>up</td>
<td>inet6</td>
<td>192.168.11.4/31</td>
<td>2001:db8:2001:9::2/126</td>
</tr>
<tr>
<td>et-0/0/3</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.11.6/31</td>
<td>2001:db8:2001:13::2/126</td>
</tr>
<tr>
<td>et-0/0/3.0</td>
<td>up</td>
<td>up</td>
<td>inet6</td>
<td>192.168.11.6/31</td>
<td>2001:db8:2001:13::2/126</td>
</tr>
<tr>
<td>et-0/2/0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.13.13/30</td>
<td>2001:db8:2001:24::1/126</td>
</tr>
<tr>
<td>et-0/2/0.0</td>
<td>up</td>
<td>up</td>
<td>inet6</td>
<td>192.168.13.13/30</td>
<td>2001:db8:2001:24::1/126</td>
</tr>
<tr>
<td>et-0/2/1</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.13.9/30</td>
<td>2001:db8:2001:23::1/126</td>
</tr>
<tr>
<td>et-0/2/1.0</td>
<td>up</td>
<td>up</td>
<td>inet6</td>
<td>192.168.13.9/30</td>
<td>2001:db8:2001:23::1/126</td>
</tr>
<tr>
<td>et-0/2/2</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.13.5/30</td>
<td>2001:db8:2001:22::1/126</td>
</tr>
<tr>
<td>et-0/2/2.0</td>
<td>up</td>
<td>up</td>
<td>inet6</td>
<td>192.168.13.5/30</td>
<td>2001:db8:2001:22::1/126</td>
</tr>
<tr>
<td>et-0/2/3</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.13.1/30</td>
<td>2001:db8:2001:21::1/126</td>
</tr>
<tr>
<td>et-0/2/3.0</td>
<td>up</td>
<td>up</td>
<td>inet6</td>
<td>192.168.13.1/30</td>
<td>2001:db8:2001:21::1/126</td>
</tr>
</tbody>
</table>

Meaning The leaf-facing and fabric-facing interfaces are functioning normally.

Spine: Verifying IPv4 EBGP Sessions

Purpose Verify the state of leaf-facing and fabric-facing IPv4 EBGP sessions.
**Action**

Verify that IPv4 EBGP sessions are established:

```
user@Spine-0> show bgp summary
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Local AS</th>
<th>Remote AS</th>
<th>Local Pref</th>
<th>Remote Pref</th>
<th>Local Flap</th>
<th>Remote Flap</th>
<th>Last Changed</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.11.1</td>
<td>420006000</td>
<td>874</td>
<td>892</td>
<td>0</td>
<td>272</td>
<td>6:49:50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>192.168.11.3</td>
<td>420006001</td>
<td>197304</td>
<td>199046</td>
<td>0</td>
<td>3</td>
<td>9w0d 6:00:16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>192.168.11.5</td>
<td>420006002</td>
<td>98702</td>
<td>98275</td>
<td>0</td>
<td>1</td>
<td>4w3d 11:20:09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>192.168.11.7</td>
<td>420006003</td>
<td>205938</td>
<td>206525</td>
<td>0</td>
<td>1</td>
<td>9w2d 15:10:25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Meaning**

The EBGP sessions are established and functioning correctly.

---

**Spine: Verifying IPv6 EBGP Sessions**

**Purpose**

Verify the state of leaf-facing and fabric-facing IPv6 EBGP sessions.

**Action**

Verify that IPv6 EBGP sessions are established:

```
user@Spine-0> show bgp summary | find 2001
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Local AS</th>
<th>Remote AS</th>
<th>Local Pref</th>
<th>Remote Pref</th>
<th>Local Flap</th>
<th>Remote Flap</th>
<th>Last Changed</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001:db8:2001:1::1</td>
<td>420006000</td>
<td>1309</td>
<td>1343</td>
<td>0</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:2001:5::1</td>
<td>420006001</td>
<td>24813</td>
<td>24671</td>
<td>0</td>
<td>0</td>
<td>1w0d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:2001:9::1</td>
<td>420006002</td>
<td>24979</td>
<td>24671</td>
<td>0</td>
<td>0</td>
<td>1w0d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:2001:13::1</td>
<td>420006003</td>
<td>24965</td>
<td>24670</td>
<td>0</td>
<td>0</td>
<td>1w0d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Meaning**

The EBGP sessions are established and functioning correctly.
Spine: Verifying BFD Sessions

Purpose
Verify the state of BFD for the BGP sessions between the leaf and fabric devices.

Action
Verify that the BFD sessions are up:

user@Spine-0> show bfd session

<table>
<thead>
<tr>
<th>Address</th>
<th>State</th>
<th>Interface</th>
<th>Detect Time</th>
<th>Transmit Interval</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.11.1</td>
<td>Up</td>
<td>et-0/0/0.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.11.3</td>
<td>Up</td>
<td>et-0/0/1.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.11.5</td>
<td>Up</td>
<td>et-0/0/2.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.11.7</td>
<td>Up</td>
<td>et-0/0/3.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.13.2</td>
<td>Up</td>
<td>et-0/2/0.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.13.6</td>
<td>Up</td>
<td>et-0/2/2.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.13.10</td>
<td>Up</td>
<td>et-0/2/1.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.13.14</td>
<td>Up</td>
<td>et-0/2/0.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>2001:db8:2001:1::1</td>
<td>Up</td>
<td>et-0/0/0.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>2001:db8:2001:5::1</td>
<td>Up</td>
<td>et-0/0/1.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>2001:db8:2001:9::1</td>
<td>Up</td>
<td>et-0/0/2.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>2001:db8:2001:13::1</td>
<td>Up</td>
<td>et-0/0/3.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>2001:db8:2001:21::2</td>
<td>Up</td>
<td>et-0/2/0.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>2001:db8:2001:22::2</td>
<td>Up</td>
<td>et-0/2/2.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>2001:db8:2001:23::2</td>
<td>Up</td>
<td>et-0/2/1.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>2001:db8:2001:24::2</td>
<td>Up</td>
<td>et-0/2/0.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
</tbody>
</table>

16 sessions, 16 clients
Cumulative transmit rate 64.0 pps, cumulative receive rate 64.0 pps

Meaning
The BFD sessions are established and functioning correctly.

Spine: Verifying IPv6 Anycast Reachability

Purpose
Verify reachability to the servers using the anycast IPv6 address.

Action
Verify that the spine device has an entry for the 2001:db8:2000::/64 route:

user@Spine-0> show route 2001:db8:2000::/64

inet6.0: 29 destinations, 43 routes (26 active, 0 holddown, 9 hidden)
Restart Complete
+ = Active Route, - = Last Active, * = Both

AS path: 420006000 420008001 I, validation-state: unverified
  to 2001:db8:2001:1::1 via et-0/0/0.0
  > to 2001:db8:2001:5::1 via et-0/0/1.0
[BGP/170] 1d 21:15:42, localpref 100
AS path: 420006001 420008003 I, validation-state: unverified
  > to 2001:db8:2001:5::1 via et-0/0/1.0

Meaning
The spine device has reachability to the servers using the anycast IPv6 address.

Purpose
Verify load sharing to the servers using the anycast IPv6 address.

Action
Verify that Packet Forwarding Engine load balancing is enabled to the anycast IPv6 address:

```
user@Spine-0> show route forwarding-table destination 2001:db8:2000::/64
Routing table: default.inet6
Internet6: Destination          Type RtRef Next hop                  Type   Index    NhRef
Netif
2001:db8:2000::/64   user     0                           ulst   131072     2
2001:db8:2001:1::1          ucst     1749     4
et-0/0/0.0
2001:db8:2001:5::1          ucst     1774     4
et-0/0/1.0
```

Meaning
The forwarding table entry for route 2001:db8:2000::/64 has the type ulst (unilist, meaning the device can use multiple next hops), followed by the list of eligible next hops.

Fabric: Verifying Interfaces

Purpose
Verify the state of the core-facing and spine-facing interfaces.
**Action**  Verify that the core-facing interfaces (xe-0/0/0:0 and xe-0/0/0:3) and spine-facing interfaces (et-0/0/20 to et-0/0/23) are up:

```bash
user@Fabric-0> run show interfaces terse
```

<table>
<thead>
<tr>
<th>Interface</th>
<th>Admin</th>
<th>Link</th>
<th>Proto</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr-0/0/0</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pfe-0/0/0</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pfe-0/0/0.16383</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pfe-0/0/0.16383</td>
<td>up</td>
<td>up</td>
<td>inet6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pfh-0/0/0</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pfh-0/0/0.16383</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-0/0/0:0</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-0/0/0:0:0</td>
<td>up</td>
<td>down</td>
<td>inet</td>
<td>192.168.14.1/30</td>
<td></td>
</tr>
<tr>
<td>xe-0/0/0:1</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-0/0/0:2</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-0/0/0:3</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-0/0/0:3.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.14.5/30</td>
<td></td>
</tr>
<tr>
<td>et-0/0/20</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>et-0/0/20.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.13.2/30</td>
<td></td>
</tr>
<tr>
<td>et-0/0/21</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>et-0/0/21.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.13.18/30</td>
<td></td>
</tr>
<tr>
<td>et-0/0/22</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>et-0/0/22.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.13.34/30</td>
<td></td>
</tr>
<tr>
<td>et-0/0/23</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>et-0/0/23.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.13.50/30</td>
<td></td>
</tr>
<tr>
<td>bme0</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bme0.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>128.0.0.1/2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128.0.0.4/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128.0.0.16/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128.0.0.32/2</td>
</tr>
<tr>
<td>bme1</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsc</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>em0</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>em0.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>10.94.191.64/24</td>
<td></td>
</tr>
<tr>
<td>em1</td>
<td>up</td>
<td>down</td>
<td>inet</td>
<td>192.168.1.2/24</td>
<td></td>
</tr>
<tr>
<td>em2</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>em2.32768</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.1/2/24</td>
<td></td>
</tr>
<tr>
<td>gre</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ipip</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>irb</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jsrv</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jsrv.1</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>128.0.0.127/2</td>
<td></td>
</tr>
<tr>
<td>lo0</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lo0.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>10.0.22.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>127.0.0.1</td>
<td></td>
</tr>
<tr>
<td>lo0.16385</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lsi</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mtun</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pimd</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pime</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tap</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vme</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vtep</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Meaning**  The core-facing and spine-facing interfaces are functioning normally.

**Fabric: Verifying IPv4 EBGP Sessions**

**Purpose**  Verify the state of IPv4 EBGP sessions with the spine devices and core routers.
**Action**
Verify that IPv4 EBGP sessions are established with the spine devices (192.168.13.x) and core routers (192.168.14.x):

```
user@Fabric-0> run show bgp summary
```

```
Groups: 2 Peers: 6 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0         10720       2082          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn
State|#Active/Received/Accepted/Damped...
192.168.13.1      420005000      9999     10115       0       1 3d 0:26:41
9/2008/2008/0        0/0/0/0
192.168.13.17     420005001     178429     178780       0       0 7w6d 6:26:51
9/2008/2008/0        0/0/0/0
192.168.13.33     420005002     178508     178908       0       0 7w6d 6:25:51
7/1988/1988/0        0/0/0/0
192.168.13.49     420005003     174753     178484       0       0 7w6d 6:24:07
2057/2060/2060/0     0/0/0/0
192.168.14.2      420006501     177329     180326       0       0 7w6d 16:19:18
0/1069/1069/0        0/0/0/0
192.168.14.6      420006502     181694     180136       0       1 7w6d 16:17:17
0/1587/1587/0        0/0/0/0
```

**Meaning**
The EBGP sessions are established and functioning correctly.

**Fabric: Verifying IPv6 EBGP Sessions**

**Purpose**
Verify the state of IPv6 EBGP sessions with the spine devices and core routers.

**Action**

```
user@Fabric-0> show bgp summary | find 2001
```

```
2001:db8:2001:21::1        420005000      14277     14353       0       0 4d 0:26:41
14:36:16 Establ
inet6.0: 1/1/1/0
2001:db8:2001:25::1        420005001      14275     14354       0       0 4d 0:26:41
14:36:12 Establ
inet6.0: 1/1/1/0
2001:db8:2001:29::1        420005002      14272     14354       0       0 4d 0:26:41
14:36:08 Establ
inet6.0: 1/1/1/0
2001:db8:2001:33::1        420005003      14265     14354       0       0 4d 0:26:41
14:36:04 Establ
inet6.0: 1/1/1/0
2001:db8:2001:51::2        420006501      1911     1908        0       1
14:40:18 Establ
inet6.0: 2/2/2/0
2001:db8:2001:52::2        420006502      1914     1908        0       1
14:40:47 Establ
inet6.0: 2/3/3/0
```

**Meaning**
The EBGP sessions are established and functioning correctly.
**Fabric: Verifying BFD Sessions**

**Purpose**  Verify the state of BFD for the BGP sessions with the spine devices and core routers.

**Action**  Verify that BFD sessions are up:

```bash
user@Fabric-0> run show bfd session
```

<table>
<thead>
<tr>
<th>Address</th>
<th>State</th>
<th>Interface</th>
<th>Detect Time</th>
<th>Transmit Interval</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.13.1</td>
<td>Up</td>
<td>et-0/0/20.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.13.17</td>
<td>Up</td>
<td>et-0/0/21.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.13.33</td>
<td>Up</td>
<td>et-0/0/22.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.13.49</td>
<td>Up</td>
<td>et-0/0/23.0</td>
<td>6.000</td>
<td>2.000</td>
<td>3</td>
</tr>
<tr>
<td>192.168.14.2</td>
<td>Up</td>
<td>xe-0/0/0:0.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.14.6</td>
<td>Up</td>
<td>xe-0/0/0:3.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
</tbody>
</table>

**Meaning**  The BFD sessions are established and functioning correctly.

**Fabric: Verifying IPv4 Anycast Reachability**

**Purpose**  Verify reachability to the servers using the anycast IPv4 address.
**Action**  
Verify that the fabric device has an entry for the 10.1.1.1 route:

```bash
user@Fabric-0> run show route 10.1.1.1
```

inet.0: 2114 destinations, 10774 routes (2113 active, 0 holddown, 1 hidden)  
+ = Active Route, - = Last Active, * = Both

| 10.1.1.0/24              | *[BGP/170] 1w4d 05:30:04, localpref 100, from 192.168.13.17  
                          | AS path: 420005001 420006000 420006700 I, validation-state: unverified  
                          | to 192.168.13.1 via et-0/0/20.0  
                          | to 192.168.13.17 via et-0/0/21.0  
                          | > to 192.168.13.33 via et-0/0/22.0  
                          | to 192.168.13.49 via et-0/0/23.0  
                          | [BGP/170] 3d 00:29:37, localpref 100  
                          | AS path: 420005000 420006000 420006700 I, validation-state: unverified  
                          | > to 192.168.13.1 via et-0/0/20.0  
                          | [BGP/170] 1w4d 05:30:04, localpref 100  
                          | AS path: 420005002 420006000 420006700 I, validation-state: unverified  
                          | > to 192.168.13.33 via et-0/0/22.0  
                          | [BGP/170] 1w4d 05:30:03, localpref 100  
                          | AS path: 420005003 420006000 420006700 I, validation-state: unverified  
                          | > to 192.168.13.49 via et-0/0/23.0  
                          | [BGP/170] 3d 00:31:25, localpref 100  
                          | AS path: 420006501 420005503 420005001 420006000 420006700 I, validation-state: unverified  
                          | > to 192.168.14.2 via xe-0/0:0/0.0 |

**Meaning**  
The fabric device has reachability to the servers using the anycast IPv4 address.

**Purpose**  
Verify load sharing to the servers using the anycast IPv4 address.
**Action**  Verify that Packet Forwarding Engine load balancing is enabled to the anycast IPv4 address:

```plaintext
user@Fabric-0> run show route forwarding-table destination 10.1.1.1
Routing table: default.inet
Internet:

Destination        Type RtRef Next hop           Type Index    NhRef Netif
10.1.1.0/24         user     0                    ulst   131070     7
                    192.168.13.1 ucst  1735  6 et-0/0/20.0
                    192.168.13.17 ucst  1736  6 et-0/0/21.0
                    192.168.13.33 ucst  1737  5 et-0/0/22.0
                    192.168.13.49 ucst  1738  2054 et-0/0/23.0
```

```
Routing table: __juniper_services__.inet
Internet:

Destination        Type RtRef Next hop           Type Index    NhRef Netif
default            perm     0                    dscd     1667     2
```

```
Routing table: __master.anon__.inet
Internet:

Destination        Type RtRef Next hop           Type Index    NhRef Netif
default            perm     0                    rjct     1682     1
```

**Meaning**  The forwarding table entry for route 10.1.1.0/24 has the type ulst (unilist, meaning the device can use multiple next hops), followed by the list of eligible next hops.

**Fabric: Verifying IPv6 Anycast Reachability**

**Purpose**  Verify reachability to the servers using the anycast IPv6 address.
**Action**
Verify that the leaf device has an entry for the 2001:db8:2000::/64 route:

```
user@Fabric-0> show route 2001:db8:2000::/64
inet6.0: 24 destinations, 35 routes (24 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8:2000::/64 *[BGP/170] 1d 05:00:11, localpref 100
  AS path: 420005000 420006000 420008001 I, validation-state: unverified
  > to 2001:db8:2001:21::1 via et-0/0/20.0
  > to 2001:db8:2001:25::1 via et-0/0/21.0
  > to 2001:db8:2001:29::1 via et-0/0/22.0
  > to 2001:db8:2001:33::1 via et-0/0/23.0

[BGP/170] 1d 05:00:11, localpref 100
  AS path: 420005001 420006000 420008001 I, validation-state: unverified
  > to 2001:db8:2001:25::1 via et-0/0/21.0

[BGP/170] 1d 05:00:11, localpref 100
  AS path: 420005002 420006000 420008001 I, validation-state: unverified
  > to 2001:db8:2001:29::1 via et-0/0/22.0

[BGP/170] 1d 05:00:11, localpref 100
  AS path: 420005003 420006000 420008001 I, validation-state: unverified
  > to 2001:db8:2001:33::1 via et-0/0/23.0

[BGP/170] 14:42:59, localpref 100
  AS path: 420006502 420005503 420005000 420006000 420008001
  I, validation-state: unverified
  > to 2001:db8:2001:52::2 via xe-0/0/0:3.0
```

**Meaning**
The leaf device has reachability to the servers using the anycast IPv6 address.

**Core: Verifying Interfaces**

**Purpose**
Verify the state of the fabric-facing, internet-facing, and core-facing interfaces.
Action  Verify that the fabric-facing interfaces (xe-1/0/0 to xe-1/0/3), Internet-facing interfaces (xe-1/3/1 and xe-1/3/2), and the interface to the neighboring core router (xe-1/3/3) are up:

```
user@R1> show interfaces terse xe-*
```

<table>
<thead>
<tr>
<th>Interface</th>
<th>Admin</th>
<th>Link</th>
<th>Proto</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>xe-1/0/0</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/0/0.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.14.18/30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>multiservice</td>
</tr>
<tr>
<td>xe-1/0/1</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/0/1.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.14.26/30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>multiservice</td>
</tr>
<tr>
<td>xe-1/0/2</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/0/2.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.14.2/30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>multiservice</td>
</tr>
<tr>
<td>xe-1/0/3</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/0/3.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.14.10/30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>multiservice</td>
</tr>
<tr>
<td>xe-1/1/0</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/1/1</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/1/2</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/1/3</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/2/0</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/2/1</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/2/2</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/3/0</td>
<td>up</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/3/1</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/3/1.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.15.5/30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mpls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>multiservice</td>
</tr>
<tr>
<td>xe-1/3/2</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/3/2.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.15.9/30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mpls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>multiservice</td>
</tr>
<tr>
<td>xe-1/3/3</td>
<td>up</td>
<td>up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xe-1/3/3.0</td>
<td>up</td>
<td>up</td>
<td>inet</td>
<td>192.168.15.1/30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mpls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>multiservice</td>
</tr>
</tbody>
</table>

Meaning  The interfaces are up and functioning normally.

Core: Verifying IPv4 BGP Sessions

Purpose  Verify the state of IPv4 BGP sessions with the neighboring core router and fabric devices.
**Action**  Verify that IPv4 BGP sessions are established with the neighboring core router (10.0.23.2) and fabric devices (192.168.14.x):

```
user@R1> show bgp summary
```

<table>
<thead>
<tr>
<th>Table</th>
<th>Tot Paths</th>
<th>Act Paths</th>
<th>Suppressed</th>
<th>History</th>
<th>Damp</th>
<th>State</th>
<th>Pending</th>
</tr>
</thead>
<tbody>
<tr>
<td>bgp.l3vpn.0</td>
<td>2259</td>
<td>2058</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>inet.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

```

<table>
<thead>
<tr>
<th>Peer</th>
<th>AS</th>
<th>InPkt</th>
<th>OutPkt</th>
<th>OutQ</th>
<th>Flaps</th>
<th>Last Up/Dwn</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.23.2</td>
<td>65000</td>
<td>180863</td>
<td>178051</td>
<td>0</td>
<td>0</td>
<td>7w6d 17:16:45</td>
</tr>
<tr>
<td>192.168.14.1</td>
<td>420005501</td>
<td>180341</td>
<td>177345</td>
<td>0</td>
<td>0</td>
<td>7w6d 16:26:06</td>
</tr>
<tr>
<td>192.168.14.9</td>
<td>420005502</td>
<td>180245</td>
<td>176736</td>
<td>0</td>
<td>0</td>
<td>7w6d 16:05:09</td>
</tr>
<tr>
<td>192.168.14.17</td>
<td>420005503</td>
<td>180777</td>
<td>177397</td>
<td>0</td>
<td>0</td>
<td>7w6d 15:55:11</td>
</tr>
<tr>
<td>192.168.14.25</td>
<td>420005504</td>
<td>180882</td>
<td>177427</td>
<td>0</td>
<td>0</td>
<td>7w6d 15:50:38</td>
</tr>
</tbody>
</table>

**Meaning**  The BGP sessions are established and functioning correctly.

---

**Core: Verifying IPv6 EBGP Sessions**

**Purpose**  Verify the state of IPv6 EBGP sessions with the fabric device.

```
user@R1> show bgp summary | find 2001
```

<table>
<thead>
<tr>
<th>Peer</th>
<th>AS</th>
<th>InPkt</th>
<th>OutPkt</th>
<th>OutQ</th>
<th>Flaps</th>
<th>Last Up/Dwn</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001:db8:2001:51::1</td>
<td>420005501</td>
<td>1917</td>
<td>1922</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2001:db8:2001:53::1</td>
<td>420005502</td>
<td>1917</td>
<td>1923</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2001:db8:2001:55::1</td>
<td>420005503</td>
<td>1918</td>
<td>1925</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2001:db8:2001:57::1</td>
<td>420005504</td>
<td>1916</td>
<td>1924</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Meaning**  The EBGP sessions are established and functioning correctly.
Core: Verifying BFD Sessions

**Purpose**
Verify the state of BFD for the connections with the neighboring core router and fabric devices.

**Action**
Verify that BFD sessions are up:

```bash
user@R1> show bfd session
```

<table>
<thead>
<tr>
<th>Address</th>
<th>State</th>
<th>Interface</th>
<th>Detect</th>
<th>Transmit</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.23.2</td>
<td>Up</td>
<td></td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.14.1</td>
<td>Up</td>
<td>xe-1/0/2.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.14.9</td>
<td>Up</td>
<td>xe-1/0/3.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.14.17</td>
<td>Up</td>
<td>xe-1/0/0.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
<tr>
<td>192.168.14.25</td>
<td>Up</td>
<td>xe-1/0/1.0</td>
<td>0.750</td>
<td>0.250</td>
<td>3</td>
</tr>
</tbody>
</table>

5 sessions, 5 clients
Cumulative transmit rate 40.0 pps, cumulative receive rate 40.0 pps

**Meaning**
The BFD sessions are established and functioning correctly.

Core: Verifying IPv4 Anycast Reachability

**Purpose**
Verify reachability to the servers using the anycast IPv4 address.
**Action**  Verify that the core router has an entry for the 10.1.1.1 route:

```
user@R1> show route 10.1.1.1
inet.0: 28 destinations, 28 routes (26 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both
0.0.0.0/0     *[Static/5] 8w2d 06:32:04
> to 10.94.191.254 via fxp0.0
RED-vpn.inet.0: 2269 destinations, 10507 routes (2068 active, 0 holddown, 201 hidden)
+ = Active Route, - = Last Active, * = Both
10.1.1.0/24    *[BGP/170] 3d 00:36:07, localpref 100, from 192.168.14.17
   AS path: 420005503 420005001 420006000 420006700 I, validation-state: unverified
   > to 192.168.14.1 via xe-1/0/2.0
   > to 192.168.14.9 via xe-1/0/3.0
   > to 192.168.14.17 via xe-1/0/0.0
   > to 192.168.14.25 via xe-1/0/1.0
   [BGP/170] 3d 00:36:07, localpref 100
   AS path: 420005501 420005001 420006000 420006700 I, validation-state: unverified
   > to 192.168.14.1 via xe-1/0/2.0
   [BGP/170] 3d 00:36:07, localpref 100
   AS path: 420005502 420005001 420006000 420006700 I, validation-state: unverified
   > to 192.168.14.9 via xe-1/0/3.0
   [BGP/170] 3d 00:36:07, localpref 100
   AS path: 420005504 420005001 420006000 420006700 I, validation-state: unverified
   > to 192.168.14.25 via xe-1/0/1.0
   [BGP/170] 3d 00:36:07, localpref 100, from 10.0.23.2
   AS path: 420006502 420005501 420005001 420006000 420006700 I, validation-state: unverified
   > to 192.168.15.6 via xe-1/3/1.0, label-switched-path to-r2
   > to 192.168.15.2 via xe-1/3/3.0, label-switched-path
   Bypass->192.168.15.6
```

**Meaning**  The core router has reachability to the servers using the anycast IPv4 address.

**Core: Verifying Server Load Balancing Using IPv4 Anycast - Packet Forwarding Engine Load Balancing**

**Purpose**  Verify load sharing to the servers using the anycast IPv4 address.
**Action**  Verify that Packet Forwarding Engine load balancing is enabled to the anycast IPv4 address:

```
user@R1> show route forwarding-table destination 10.1.1.1
Routing table: default.inet
Internet:
Destination     Type RtRef Next hop           Type Index    NhRef Netif
default         user     0 5c:5e:ab:79:42:81  ucst      355    11 fxp0.0
default         perm     0                    rjct      36     1
```

```
Routing table: __master.anon__.inet
Internet:
Destination     Type RtRef Next hop           Type Index    NhRef Netif
default         perm     0                    rjct      520     1
```

```
Routing table: __juniper_services__.inet
Internet:
Destination     Type RtRef Next hop           Type Index    NhRef Netif
default         perm     0                    dscd      542     2
```

```
Routing table: RED-vpn.inet
Internet:
Destination     Type RtRef Next hop           Type Index    NhRef Netif
10.1.1.0/24      user     0                    ulst 1048584  2061
  192.168.14.1    ucst     632     5 xe-1/0/2.0
  192.168.14.9    ucst     602     5 xe-1/0/3.0
  192.168.14.17   ucst     606     5 xe-1/0/0.0
  192.168.14.25   ucst     607     5 xe-1/0/1.0
```

**Meaning**  The forwarding table entry for route 10.1.1.0/24 has the type ulst (unilist, meaning the device can use multiple next hops), followed by the list of eligible next hops.

**Core: Verifying IPv6 Anycast Reachability**

**Purpose**  Verify reachability to the servers using the anycast IPv6 address.
**Action**  
Verify that the core router has an entry for the 2001:db8:2000::/64 route:

```
user@R1> show route 2001:db8:2000::/64
```

RED-vpn.inet6.0: 17 destinations, 32 routes (17 active, 0 holddown, 0 hidden)  
+ = Active Route, - = Last Active, * = Both

```
2001:db8:2000::/64 *[BGP/170] 14:45:59, localpref 100
   AS path: 420005501 420005000 420006000 420008001 I,
   validation-state: unverified
   to 2001:db8:2001:55::1 via xe-1/0/0.0
   to 2001:db8:2001:57::1 via xe-1/0/1.0
   > to 2001:db8:2001:51::1 via xe-1/0/2.0
   to 2001:db8:2001:53::1 via xe-1/0/3.0
   [BGP/170] 14:45:58, localpref 100
   AS path: 420005502 420005000 420006000 420008001 I,
   validation-state: unverified
   > to 2001:db8:2001:53::1 via xe-1/0/3.0
   [BGP/170] 14:45:58, localpref 100
   AS path: 420005503 420005000 420006000 420008001 I,
   validation-state: unverified
   > to 2001:db8:2001:55::1 via xe-1/0/0.0
   [BGP/170] 14:45:58, localpref 100
   AS path: 420005504 420005000 420006000 420008001 I,
   validation-state: unverified
   > to 2001:db8:2001:57::1 via xe-1/0/1.0
```

**Meaning**  
The core router has reachability to the servers using the anycast IPv6 address.

**Additional Features: Verifying SNMP**

**Purpose**  
Verify that SNMP and system logging (syslog) are operating correctly.

**Action**  
1. Verify that SNMP is configured and operating as expected:

```
user@Leaf-0> show snmp v3
Local engine ID: 80 00 0a 4c 04 31 30 2e 39 34 2e 36 34 2e 31 35 34
Engine boots: 3
Engine time: 642968 seconds
Max msg size: 65507 bytes

Engine ID: local
User      Auth/Priv  Storage   Status
snmp      sha/des   nonvolatile active
JunosSpace md5/des   nonvolatile active

Group name  Security model Security name Storage type Status
view-all    usm    snmp               nonvolatile active

Access control:
Group    Context Security model/level Read Write Notify
view-all prefix model/level usm/privacy view-all view-all
```

SNMP Target:
<table>
<thead>
<tr>
<th>Address name</th>
<th>Address</th>
<th>Port</th>
<th>Parameters name</th>
<th>Parameters type</th>
<th>Storage type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA_SPACE</td>
<td>10.94.63.251</td>
<td>162</td>
<td>TP_SPACE</td>
<td>nonvolatile</td>
<td>active</td>
<td></td>
</tr>
<tr>
<td>ta1</td>
<td>10.94.63.250</td>
<td>162</td>
<td>tp1</td>
<td>nonvolatile</td>
<td>active</td>
<td></td>
</tr>
<tr>
<td>ta2</td>
<td>10.94.63.251</td>
<td>162</td>
<td>tp1</td>
<td>nonvolatile</td>
<td>active</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters name</th>
<th>Security name</th>
<th>Security model/level</th>
<th>Notify filter type</th>
<th>Storage type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP_SPACE</td>
<td>JunosSpace</td>
<td>usm/privacy</td>
<td>SPACE_T</td>
<td>nonvolatile</td>
<td>active</td>
</tr>
<tr>
<td>tp1</td>
<td>snmp</td>
<td>usm/authent</td>
<td>nf1</td>
<td>nonvolatile</td>
<td>active</td>
</tr>
</tbody>
</table>

**SNMP Notify:**

<table>
<thead>
<tr>
<th>Notify name</th>
<th>Tag</th>
<th>Type</th>
<th>Storage type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE_TRAPS</td>
<td>TAG_SPACE</td>
<td>trap</td>
<td>nonvolatile</td>
<td>active</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filter name</th>
<th>Subtree</th>
<th>Filter type</th>
<th>Storage type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>nf1</td>
<td>1</td>
<td>include</td>
<td>nonvolatile</td>
<td>active</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filter name</th>
<th>Subtree</th>
<th>Filter type</th>
<th>Storage type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE_TRAP_FILTER</td>
<td>1</td>
<td>include</td>
<td>nonvolatile</td>
<td>active</td>
</tr>
</tbody>
</table>

2. Login to the syslog server (in this example, a Linux server at 10.94.63.191) and verify that logs are being received:

```bash
user@ubuntu:/var/log/JUNOS$ tail -f syslog.log
Jun 27 06:33:33 leaf-0 - last message repeated 5 times
Jun 26 22:20:55 ubuntu kernel: [459605.509892] init: supervisor-config main process (17142) killed by TERM signal
Jun 27 06:34:03 leaf-0 - last message repeated 6 times
Jun 27 06:34:06 leaf-0 alarmd[1709] Alarm cleared: License color=.YELLOW, class=CHASSIS, reason=BGP Routing Protocol usage requires a license
Jun 27 06:34:06 leaf-0 craftd[1710] Receive FX craftd clear alarm message: color: 2 class: 50 object: 50 slot: 126 id=0 reason=47
Jun 27 06:34:06 leaf-0 craftd[1710] Minor alarm cleared, BGP Routing Protocol usage requires a license
Jun 27 06:34:06 leaf-0 alarmd[1709] Alarm set: License color=.YELLOW, class=CHASSIS, reason=BGP Routing Protocol usage requires a license
Jun 27 06:34:06 leaf-0 craftd[1710] Receive FX craftd set alarm message: color: 2 class: 50 object: 50 slot: 126 silent: 0 short_reason=bgp long_reason=BGP Routing Protocol usage requires a license id=0 reason=47
Jun 27 06:34:06 leaf-0 craftd[1710] Minor alarm set, BGP Routing Protocol usage requires a license
Jun 27 06:34:08 leaf-0 /kernel setsockopts: setting SO_RTBL_INDEX to 2
```

**Meaning**

SNMP and syslog are operating as expected.
**Additional Features: Verifying BMP**

**Purpose**  
Verify that BMP is operating correctly.

**Action**  
1. Verify that BMP is configured and operating as expected:

   ```
   user@Leaf-0> show bgp bmp
   Station name: BMP-Server
   Local address/port: --/--, Station address/port: 10.94.63.193/11019, active
   State: established Local: 10.94.63.154+58176 Remote: 10.94.63.193+11019
   Last state change: 4d 8:51:48
   Monitor BGP Peers: enabled
   Route-monitoring: pre-policy
   Hold-down: 600, flaps 3, period 300
   Priority: low
   Statistics timeout: 30
   Version: 3
   ```

2. Log into the BMP server (in this example, a Linux server at 10.94.63.193) and verify that BMP messages are being received:

   ```
   user@ubuntu--bmp:~$ tail -f /var/lib/docker/aufs/mnt/abc123/root/ryu_bmp.log
   2016 Jun 26 23:12:44 | 10.94.63.154 | BMPStatisticsReport (is_post_policy=False, len=132, peer_address='2001:4::2', peer_as=420005003, peer_bgp_id='10.0.16.10', peer_distinguisher=0, peer_type=0,stats=[{'type': 0, 'value': 0, 'len': 4}, {'type': 1, 'value': 0, 'len': 4}, {'type': 2, 'value': 0, 'len': 4}, {'type': 3, 'value': 0, 'len': 4}, {'type': 4, 'value': 0, 'len': 4}, {'type': 5, 'value': 0, 'len': 4}, {'type': 6, 'value': 0, 'len': 4}, {'type': 7, 'value': 2, 'len': 8}, {'type': 8, 'value': 2, 'len': 8}],timestamp=1467034691.0,type=1,version=3)
   ```

   ```
   2016 Jun 26 23:12:44 | 10.94.63.154 | BMPStatisticsReport (is_post_policy=False, len=132, peer_address='2001:92::1', peer_as=420008000, peer_bgp_id='192.0.0.45', peer_distinguisher=0, peer_type=0,stats=[{'type': 0, 'value': 0, 'len': 4}, {'type': 1, 'value': 0, 'len': 4}, {'type': 2, 'value': 0, 'len': 4}, {'type': 3, 'value': 0, 'len': 4}, {'type': 4, 'value': 0, 'len': 4}, {'type': 5, 'value': 0, 'len': 4}, {'type': 6, 'value': 0, 'len': 4}, {'type': 7, 'value': 100, 'len': 8}, {'type': 8, 'value': 100, 'len': 8}],timestamp=1467034691.0,type=1,version=3)
   ```

   ```
   2016 Jun 26 23:12:44 | 10.94.63.154 | BMPStatisticsReport (is_post_policy=False, len=132, peer_address='2001:93::1', peer_as=420008001, peer_bgp_id='192.0.0.46', peer_distinguisher=0, peer_type=0,stats=[{'type': 0, 'value': 0, 'len': 4}, {'type': 1, 'value': 0, 'len': 4}, {'type': 2, 'value': 0, 'len': 4}, {'type': 3, 'value': 0, 'len': 4}, {'type': 4, 'value': 0, 'len': 4}, {'type': 5, 'value': 0, 'len': 4}, {'type': 6, 'value': 0, 'len': 4}, {'type': 7, 'value': 100, 'len': 8}, {'type': 8, 'value': 100, 'len': 8}],timestamp=1467034691.0,type=1,version=3)
   ```

   ```
   2016 Jun 26 23:12:44 | 10.94.63.154 | BMPStatisticsReport (is_post_policy=False, len=132, peer_address='2001:92::1', peer_as=420006601, peer_bgp_id='172.16.90.1', peer_distinguisher=0, peer_type=0,stats=[{'type': 0, 'value': 0, 'len': 4}, {'type': 1, 'value': 0, 'len': 4}, {'type': 2, 'value': 0, 'len': 4}, {'type': 3, 'value': 0, 'len': 4}, {'type': 4, 'value': 0, 'len': 4}, {'type': 5, 'value': 0, 'len': 4}, {'type': 6, 'value': 0, 'len': 4}, {'type': 7, 'value': 2, 'len': 8}, {'type': 8, 'value': 2, 'len': 8}],timestamp=1467034691.0,type=1,version=3)
   ```

   ```
   2016 Jun 26 23:12:44 | 10.94.63.154 | BMPStatisticsReport (is_post_policy=False, len=132, peer_address='2001:92::1', peer_as=420006601, peer_bgp_id='172.16.90.1', peer_distinguisher=0, peer_type=0,stats=[{'type': 0, 'value': 0, 'len': 4}, {'type': 1, 'value': 0, 'len': 4}, {'type': 2, 'value': 0, 'len': 4}, {'type': 3, 'value': 0, 'len': 4}, {'type': 4, 'value': 0, 'len': 4}, {'type': 5, 'value': 0, 'len': 4}, {'type': 6, 'value': 0, 'len': 4}, {'type': 7, 'value': 2, 'len': 8}, {'type': 8, 'value': 2, 'len': 8}],timestamp=1467034691.0,type=1,version=3)
   ```
Meaning BMP is operating as expected.

Additional Features: Verifying Remote Port Mirroring

Purpose Verify that remote port mirroring is operating correctly.

Action 1. If using the analyzer instance option, verify that the analyzer configuration is operating as expected:

user@Leaf-0> show forwarding-options analyzer RemPortMon-GRE
Analyzer name          : RemPortMon-GRE
Mirror rate            : 1
Maximum packet length  : 0
State                  : up
Ingress monitored interfaces : xe-0/0/47.0
Destination IP        : 10.100.10.1

2. If using the port-mirroring instance option, verify that the port mirroring instance configuration is operating as expected:

user@Leaf-0> show forwarding-options port-mirroring detail
Instance Name: RemPortMon-GRE
Instance Id: 2
Input parameters:
Rate                  : 1
Run-length            : 0
Maximum-packet-length : 0
Output parameters:
Family              State     Destination          Next-hop
inet                up        10.100.10.1

Meaning Remote port monitoring is operating as expected.

Additional Features: Verifying Class of Service

Purpose Verify that CoS is operating correctly.

Action 1. Verify the forwarding class set configuration:

user@Leaf-0> show class-of-service forwarding-class-set
Forwarding class set: fc-set-anycast, Type: normal-type, Forwarding class set index: 25422
Forwarding class         Index
BE-data                  0
App-1                    4
App-2                    5
2. Verify the scheduler map configuration:

```
user@Leaf-0> show class-of-service scheduler-map sm-anycast
Scheduler map: sm-anycast, Index: 40394

Scheduler: BE-data-sc, Forwarding class: BE-data, Index: 64266
Transmit rate: 20 percent, Rate Limit: none, Buffer size: remainder,
Buffer Limit: none, Priority: low
Excess Priority: unspecified, Explicit Congestion Notification: disable
Drop profiles:
<table>
<thead>
<tr>
<th>Loss priority</th>
<th>Protocol</th>
<th>Index</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
<tr>
<td>Medium high</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
<tr>
<td>High</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
</tbody>
</table>

Scheduler: App-1-sc, Forwarding class: App-1, Index: 56031
Transmit rate: 1 percent, Rate Limit: none, Buffer size: 20 percent,
Buffer Limit: none, Priority: low
Excess Priority: unspecified
Shaping rate: 50 percent
Explicit Congestion Notification: disable
Drop profiles:
<table>
<thead>
<tr>
<th>Loss priority</th>
<th>Protocol</th>
<th>Index</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
<tr>
<td>Medium high</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
<tr>
<td>High</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
</tbody>
</table>

Scheduler: App-2-sc, Forwarding class: App-2, Index: 23262
Transmit rate: 59 percent, Rate Limit: none, Buffer size: 10 percent,
Buffer Limit: none, Priority: low
Excess Priority: unspecified
Shaping rate: 60 percent
Explicit Congestion Notification: disable
Drop profiles:
<table>
<thead>
<tr>
<th>Loss priority</th>
<th>Protocol</th>
<th>Index</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
<tr>
<td>Medium high</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
<tr>
<td>High</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
</tbody>
</table>

Scheduler: App-3-sc, Forwarding class: App-3, Index: 56030
Transmit rate: 20 percent, Rate Limit: none, Buffer size: 30 percent,
Buffer Limit: none, Priority: low
Excess Priority: unspecified
Shaping rate: 60 percent
Explicit Congestion Notification: disable
Drop profiles:
<table>
<thead>
<tr>
<th>Loss priority</th>
<th>Protocol</th>
<th>Index</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
<tr>
<td>Medium high</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
<tr>
<td>High</td>
<td>any</td>
<td>1</td>
<td>&lt;default-drop-profile&gt;</td>
</tr>
</tbody>
</table>
```

3. Verify that traffic is being classified and queued appropriately:

```
user@Leaf-0> show interfaces queue et-0/0/48
Physical interface: et-0/0/48, Enabled, Physical link is Up
Interface index: 694, SNMP ifIndex: 574
Forwarding classes: 16 supported, 8 in use
Egress queues: 12 supported, 8 in use
```
Queue: 0, Forwarding classes: BE-data
Queued:
| Packets | 0 | 0 pps |
| Bytes   | 0 | 0 bps |
Transmitted:
| Packets | 28141 | 0 pps |
| Bytes   | 14408192 | 0 bps |
| Tail-dropped packets | Not Available |
| RL-dropped packets | 0 | 0 pps |
| RL-dropped bytes | 0 | 0 bps |
| Total-dropped packets | 0 | 0 pps |
| Total-dropped bytes | 0 | 0 bps |

Queue: 1, Forwarding classes: App-1
Queued:
| Packets | 0 | 0 pps |
| Bytes   | 0 | 0 bps |
Transmitted:
| Packets | 47414945 | 236 pps |
| Bytes   | 24276451840 | 970576 bps |
| Tail-dropped packets | Not Available |
| RL-dropped packets | 0 | 0 pps |
| RL-dropped bytes | 0 | 0 bps |
| Total-dropped packets | 0 | 0 pps |
| Total-dropped bytes | 0 | 0 bps |

Queue: 2, Forwarding classes: App-2
Queued:
| Packets | 0 | 0 pps |
| Bytes   | 0 | 0 bps |
Transmitted:
| Packets | 109551690 | 242 pps |
| Bytes   | 56090465280 | 994536 bps |
| Tail-dropped packets | Not Available |
| RL-dropped packets | 0 | 0 pps |
| RL-dropped bytes | 0 | 0 bps |
| Total-dropped packets | 0 | 0 pps |
| Total-dropped bytes | 0 | 0 bps |

Queue: 3, Forwarding classes: NC
Queued:
| Packets | 0 | 0 pps |
| Bytes   | 0 | 0 bps |
Transmitted:
| Packets | 0 | 0 pps |
| Bytes   | 0 | 0 bps |
| Tail-dropped packets | Not Available |
| RL-dropped packets | 0 | 0 pps |
| RL-dropped bytes | 0 | 0 bps |
| Total-dropped packets | 0 | 0 pps |
| Total-dropped bytes | 0 | 0 bps |

Queue: 4, Forwarding classes: no-loss
Queued:
| Packets | 0 | 0 pps |
| Bytes   | 0 | 0 bps |
Transmitted:
| Packets | 0 | 0 pps |
| Bytes   | 0 | 0 bps |
| Tail-dropped packets | Not Available |
| RL-dropped packets | 0 | 0 pps |
| RL-dropped bytes | 0 | 0 bps |
| Total-dropped packets | 0 | 0 pps |
| Total-dropped bytes | 0 | 0 bps |

Queue: 5, Forwarding classes: App-3
Meaning  Traffic matching the App-1, App-2, and App-3 multifield classifiers is correctly being identified, queued, and scheduled.

Related Documentation  
- Understanding the Software as a Service Solution on page 5
- Configuring an IP Fabric using Junos Space Network Director or OpenClos on page 88

Configuring an IP Fabric using Junos Space Network Director or OpenClos

Juniper Networks provides tools to help automate the creation of spine-and-leaf IP fabrics for SaaS environments. This example includes two options to help with IP fabric creation: OpenClos and Junos Space Network Director.

OpenClos is a Python script library that enables you automate the design, deployment, and maintenance of a Layer 3 fabric built on BGP. To create an IP fabric that uses a spine-and-leaf architecture, the script generates device configuration files and uses zero touch provisioning (ZTP) to push the configuration files to the devices.
OpenClos functionality has also been built into Network Director 2.0 (and later), which allows you to provision spine-and-leaf Layer 3 fabrics using a GUI-based wizard.

### Leaf Device Configuration

The Layer 3 Fabric wizard and OpenClos tools autogenerate the following configuration elements for leaf devices:

- System configuration (hostname, root password, services, syslog, and so on)
- Upstream interfaces to each spine device
- Downstream Layer 2 interfaces for server access
- Loopback interface
- IRB interface to provide gateway address for servers
- VLAN to aggregate server-facing interfaces
- Static routes and other routing options
- EBGP sessions to each spine device
- Routing policy
- LLDP
- SNMP and event options

### Spine Device Configuration

The Layer 3 Fabric wizard and OpenClos tools autogenerate the following configuration elements for spine devices:

- System configuration (hostname, root password, services, syslog, and so on)
- Downstream interfaces to each leaf device
- Loopback and management interfaces
- Static routes and other routing options
- EBGP sessions to each leaf device
- Routing policy
- LLDP
- SNMP and event options

The following sections describe how to use these automation tools to provision the leaf and spine layers of an IP fabric.

- [Using Network Director to Provision Leaf and Spine Devices on page 90](#)
- [Using OpenClos to Provision Leaf and Spine Devices on page 95](#)
Using Network Director to Provision Leaf and Spine Devices

This section describes how to use the Layer 3 Fabrics wizard in Junos Space Network Director 2.5 to provision the leaf and spine layers of an IP fabric.

NOTE: Network Director 2.0 provided initial support for Layer 3 fabrics, using QFX5100 switches. Network Director 2.5 adds support for QFX10002 switches.

The procedure below creates the leaf configuration shown in the section “Configuring Leaf Devices for the IP Fabric” on page 19, and creates the spine configuration shown in the section “Configuring Spine Devices for the IP Fabric” on page 31.

NOTE: More detailed information on creating Layer 3 fabrics using Network Director can be found at Creating Layer 3 Fabrics.

To autoprovision the main spine and leaf configuration elements using Network Director:

1. In the Views drop-down menu, select Logical View; then in the Tasks section, select Device Management > Manage Layer 3 Fabrics.

2. In the Manage Layer-3 Fabrics section, select the Create option.
3. On the Fabric Requirement page:
   a. Enter a name in the Fabric Name field.
   b. In the Spines section, ensure the Model field is set to QFX10002-72Q, and set the Initial Capacity and Max. Capacity fields to 4.

   **NOTE:** This procedure was originally created using QFX5100-24Q-2P switches as spine devices, thus their use in the screenshots below. To meet the SaaS solution’s current specifications, simply select QFX10002-72Q as the model for the spine switches.

   c. In the Leaves section, add two qfx5100-48t-6q and four qfx5100-48s-6q devices, and set the Max. Capacity field to 6.
   d. Click Next.

4. On the Devices page, confirm that the device listing is correct and click Next.
5. On the Configurations page, fill in the fields as shown below.

6. On the Cabling page, review the cabling plan for the devices in the Layer 3 fabric and click Next.

**NOTE:** The cabling plan displays the exact port numbers that you must use to connect the spine and leaf devices. Configurations created and deployed by Network Director will use these interface names.
7. On the ZTP Settings page, enter the appropriate data in the various fields, including the serial number or management interface MAC address for all spine devices, and click Next.

**NOTE:** Using these details, the spine devices will be autodiscovered using LLDP, once cabling is completed.

8. On the Review page, review the configuration settings for the Layer 3 fabric, and when you are ready to deploy the configuration files to the devices, click **Deploy**.
9. Monitor the progress of the deployment by using the ZTP Provisioning dialog box.

10. To verify the initial configuration and connectivity of the leaf and spine devices, go to the Views drop-down menu and select Logical View; then in the Tasks section, select View Inventory.
The leaf and spine devices should appear in the Device Inventory window with a connection state of **UP** and configuration state of **In Sync**.

**NOTE:** Again, note that this procedure was originally created using QFX5100-24Q-2P switches as spine devices, thus their appearance in the screenshots above. Using the steps in this procedure, the platform displayed for spine devices should be QFX10002-72Q.

**For More Information**

The resulting leaf configuration from this procedure can be found at “Configuring Leaf Devices for the IP Fabric” on page 19.

The resulting spine configuration from this procedure can be found at “Configuring Spine Devices for the IP Fabric” on page 31.

More detailed information on creating Layer 3 fabrics using Network Director can be found at Creating Layer 3 Fabrics.

**Using OpenClos to Provision Leaf and Spine Devices**

This section describes how to use OpenClos 3.0 to provision the leaf and spine layers of an IP fabric.

The procedure below creates the leaf configuration shown in the section “Configuring Leaf Devices for the IP Fabric” on page 19, and creates the spine configuration shown in the section “Configuring Spine Devices for the IP Fabric” on page 31.

**NOTE:** More detailed information on creating Layer 3 fabrics using OpenClos can be found at [https://github.com/Juniper/OpenClos](https://github.com/Juniper/OpenClos).

Before you begin, install OpenClos on a Linux server, as follows:

2. OpenClos should be installed in /var/tmp.

   user@ubuntu-Openclos:/var/tmp$ ls
   OpenClos-devR3.0

To autoprovision the main spine and leaf configuration elements using OpenClos:

1. Navigate to the closDefinition.yaml file.

   This file is used to include the IP fabric requirements.

   user@ubuntu-Openclos:/var/tmp$ cd OpenClos-devR3.0/jnpr/openclos/conf
   user@ubuntu-Openclos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/conf$

2. Open the closDefinition.yaml file and edit the ZTP settings, number of PODs, and Junos OS image locations.

   solution@ubuntu-Openclos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/conf$ more closDefinition.yaml

   ztp:
     # the image file should be placed under <install-dir>/jnpr/openclos/conf/ztp
     # if not placed under this dir, the file would not be accessible from http server
     # and ZTP process will be broken, can be overridden at each pod for Spine/Leaf
     # this field is optional
     #    junosImage : jinstall-qfx-5-14.1X53-D10.4-domestic-signed.tgz
     dhcpSubnet : 10.94.63.150/24
      # dhcpOptionRoute is the Gateway address for any out-of-band network including
      # management network, this will get configured using static route.
      # by default openclos would run on same subnet as devices.
      dhcpOptionRoute : 10.94.63.254

      # Following two fields are optional, if not provided start and end
      # includes complete dhcp subnet, example for 10.0.2.0/24
      dhcpOptionRangeStart : 10.94.63.150
      dhcpOptionRangeEnd : 10.94.63.170

   pods:
      # pod name or pod identifier
      labLeafSpine:
       spineCount : 4
       # possible options for spine deviceType are qfx5100-24q-2p, qfx10002-***, qfx10008-***
       # the image file should be placed under <install-dir>/jnpr/openclos/conf/ztp
       # if not placed under this dir, the file would not be accessible from http server
       # and ZTP process will be broken, these are optional, overrides global setting ztp.junosImage
       spineSettings :
          - deviceType : qfx10002-72q
            junosImage : jinstall-qfx-10-f-15.1X53-D32.2-domestic-signed.tgz
leafCount : 6
# possible options for leafDeviceType are qfx5100-96S, qfx5100-48s-6q

# for complete list refer to openclclo.yaml
# the image file should be placed under
<install-dir>/jnpr/openclclo/conf/ztp
leafSettings :
  - deviceType : qfx5100-48s-6q
    junosImage : jinstall-qfx-5-flex-15.1R3.6-domestic-signed.tgz
  - deviceType : OCX1100-48Sx
    junosImage : jinstall-ocx-11-flex-14.1X53-D35.3-domestic.tgz
# Number of uplink from leaf must be properly connected and up to
indicate
# the leaf device as "good". If the leaf device is not in "good" state
# it would not go through 2-stage ZTP/configuration process.
# Possible value is in between 2 and spineCount, inclusive both end.
# This field is optional, default value is max(2,
math.ceil(spineCount/2))
leafUplinkCountMustBeUp : 3
hostOrVmCountPerLeaf : 25
interConnectPrefix : 192.168.11.0/24
vlanPrefix : 172.16.64.0/24
loopbackPrefix : 10.0.16.0/24
# either managementPrefix or (managementStartingIP, managementMask)
is mandatory. Here is how it works:
# case 1: managementPrefix : 1.2.3.7/24
# from cidr notation of managementPrefix we know available
block is [1.2.3.0 - 1.2.3.255]
# from ip portion of managementPrefix we know starting ip is
1.2.3.7
# so the effective range is [1.2.3.7 -
1.2.3.7+spineCount+leafCount]
# case 2: managementStartingIP : 1.2.3.7
# managementMask : 24
# from cidr notation of 'managementStartingIP/managementMask'
we know available block is [1.2.3.0 - 1.2.3.255]
# from managementStartingIP we know starting ip is 1.2.3.7
# so the effective range is [1.2.3.7 -
1.2.3.7+spineCount+leafCount]
managementPrefix : 10.94.63.150/24
# managementStartingIP : 10.94.63.150
# managementMask : 24
spineAS : 420005000
leafAS : 420006000
# possible options for topologyType are threeStage, fiveStageRealEstate,
fiveStagePerformance
topologyType : threeStage
inventory : inventoryLabLeafSpine.json
# device default root password
# List of out of band networks, example - devices in management network
outOfBandAddressList :
  # - 10.94.185.18/32
  # - 10.94.185.19/32
  # - 172.16.0.0/12
  0.0.0.0/0
# Management network gateway address
# It overrides ztp:dhcpOptionRoute setting
outOfBandGateway : 10.94.63.254
# device default root password

devicePassword: <password>

# possible options for leafDeviceType are qfx5100-96S, qfx5100-48s-6q

# but we can use qfx5100-24q-2p with customized SKU
# following example assumes device has no expansion module, so there
# are 24 ports. First 16 ports are used as access port and
# remaining 8 ports are used as uplink ports

leafSettings:
  - deviceType: qfx5100-24q-2p
    uplinkPorts: ['et-0/0/[16-23]']
    downlinkPorts: ['et-0/0/[0-15]']

# Number of uplink from leaf must be properly connected and up to
# indicate
# the leaf device as "good". If the leaf device is not in "good" state
# it would not go through 2-stage ZTP/configuration process.
# Possible value is in between 2 and spineCount, inclusive both end.
# This field is optional, default value is max(2, math.ceil(spineCount/2))

3. Confirm that the devices to be used in the IP fabric are included in the
deviceFamily.yaml file.

This file includes details about devices to be used in the IP fabric. OpenClos 3.0 includes
native support for several devices, including the QFX5100, QFX10002, and OCX1100
switches.

user@ubuntu-Openclos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/conf$

more deviceFamily.yaml
# Device port usage based on device family and topology
# qfx5100-24q-2p ports could have 32 ports with two four-port expansion modules
# When used as Fabric, all ports are downlink ports
# When used as Spine in 3-Stage topology, all ports are used as downlink
# When used as Spine in 5-Stage topology, ports are split between uplink and
downlink

deviceFamily:
qfx5100-24q-2p:
  fabric:
    uplinkPorts:
      downlinkPorts: ['et-0/0/[0-23]', 'et-0/1/[0-3]', 'et-0/2/[0-3]']
    spine:
      uplinkPorts: ['et-0/0/[16-23]', 'et-0/1/[0-3]', 'et-0/2/[0-3]']
      downlinkPorts: 'et-0/0/[0-15]'
qfx10002-36q:
  fabric:
    uplinkPorts:
      downlinkPorts: 'et-0/0/[0-35]'
    spine:
      uplinkPorts: 'et-0/0/[18-35]'
      downlinkPorts: 'et-0/0/[0-17]'
qfx10002-72q:
  fabric:
    uplinkPorts:
      downlinkPorts: 'et-0/0/[0-71]'
    spine:
      uplinkPorts: 'et-0/0/[36-71]'
      downlinkPorts: 'et-0/0/[0-35]'
qfx10008:
fabric:
    uplinkPorts:
    downlinkPorts:
spine:
    uplinkPorts:
    downlinkPorts:
leaf:
    uplinkPorts:
    downlinkPorts:

qfx5100-48s-6q:
    leaf:
        uplinkPorts: 'et-0/0/[48-53]'
        downlinkPorts: ['xe-0/0/[0-47]', 'ge-0/0/[0-47]']
qfx5100-48t-6q:
    leaf:
        uplinkPorts: 'et-0/0/[48-53]'
        downlinkPorts: 'xe-0/0/[0-47]'
OCX1100-48SX:
    leaf:
        uplinkPorts: 'et-0/0/[48-53]'
        downlinkPorts: 'xe-0/0/[0-47]'
qfx5100-96s-8q:
    leaf:
        uplinkPorts: 'et-0/0/[96-103]'
        downlinkPorts: ['xe-0/0/[0-95]', 'ge-0/0/[0-95]']
qfx5200-32c-32q:
    spine:
        uplinkPorts: 'et-0/0/[32-53]'
        downlinkPorts: 'et-0/0/[00-31]'
    leaf:
        uplinkPorts: 'et-0/0/[00-07]'
        downlinkPorts: 'et-0/0/[08-35]'
ex4300-24p:
    leaf:
        uplinkPorts: 'et-0/1/[0-3]'
        downlinkPorts: 'ge-0/0/[0-23]'
ex4300-24t:
    leaf:
        uplinkPorts: 'et-0/1/[0-3]'
        downlinkPorts: 'ge-0/0/[0-23]'
ex4300-32f:
    leaf:
        uplinkPorts: ['et-0/1/[0-1]', 'et-0/2/[0-1]']
        downlinkPorts: 'ge-0/0/[0-31]'
ex4300-48p:
    leaf:
        uplinkPorts: 'et-0/1/[0-3]'
        downlinkPorts: 'ge-0/0/[0-47]'
ex4300-48t:
    leaf:
        uplinkPorts: 'et-0/1/[0-3]'
        downlinkPorts: 'ge-0/0/[0-47]'

# additional customization of port allocation based on topology
3Stage:
qfx5100-24q-2p:
    spine:
        uplinkPorts:
        downlinkPorts: ['et-0/0/[0-23]', 'et-0/1/[0-3]', 'et-0/2/[0-3]']
qfx10002-36q:
spine:
  uplinkPorts: 'et-0/0/[0-35]
qfx10002-72q:
  spine:
    uplinkPorts:
    downlinkPorts: 'et-0/0/[0-71]
qfx10008:
  spine:
    # assuming 8 x ULC-36Q-12Q28 used
    # if ULC-30Q28 is used, the port range would change to
    'et-*/0/[0-29]
    uplinkPorts:
    downlinkPorts: ['et-0/0/[0-35]', 'et-1/0/[0-35]', 'et-2/0/[0-35]', 'et-3/0/[0-35]', 'et-4/0/[0-35]', 'et-5/0/[0-35]', 'et-6/0/[0-35]', 'et-7/0/[0-35]'
leaf:
  # assuming 8 x ULC-60S-6Q used
  uplinkPorts: ['et-0/0/[60-65]', 'et-1/0/[60-65]', 'et-2/0/[60-65]', 'et-3/0/[60-65]', 'et-4/0/[60-65]', 'et-5/0/[60-65]', 'et-6/0/[60-65]', 'et-7/0/[60-65]'
  downlinkPorts: ['et-0/0/[0-59]', 'et-1/0/[0-59]', 'et-2/0/[0-59]', 'et-3/0/[0-59]', 'et-4/0/[0-59]', 'et-5/0/[0-59]', 'et-6/0/[0-59]', 'et-7/0/[0-59]'
5Stage:

lineCard:
  ULC-30Q28:
    uplinkPorts: 'et-0/0/[0-29]
  ULC-36Q-12Q28:
    uplinkPorts: 'et-0/0/[0-35]
  ULC-60S-6Q:
    uplinkPorts: 'et-0/0/[60-65]
    downlinkPorts: 'et-0/0/[0-59]'

4. Edit the `openclos.yaml` file to configure OpenClos application settings.

```
user@ubuntu-OpenClos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/conf$ more openclos.yaml
# Deployment mode
# ztpStaged: true/false, true indicates ZTP process goes through 2-stage
# device configuration. During leaf device boot-strap, it gets generic config,
# then OpenClos finds the topology and applies new topology.
# False indicates all leaf configs are generated based on cabling-plan and
# deployed to the device using ZTP process.
# ztpStagedAttempt: How many times OpenClos tries to connect to leaf
# to collect lldp data when it receives trap from that leaf.
# default is 5 times. 0 means no-op so it basically disables the 2-stage.
# ztpStagedInterval: How long OpenClos waits in between retries.
# default is 60 seconds. 0 means do not wait.
# ztpVcpLldpDelay: How long OpenClos waits between delete VCP on EX4300 and
# LLDP collection
# ztpStagedAttempt and ztpStagedInterval only take effect
# when ztpStaged is set to true.
deploymentMode :
  ztpStaged : true
  ztpStagedAttempt: 5
  ztpStagedInterval: 30
```
ztpVcpLldpDelay: 40

# Generated file/configuration location
# default value 'out' relative to current dir
# can take absolute path as '/tmp/out/
outputDir : /tmp/out/

# Database URL
# Please NOTE dbUrl is used by sqlite only. For all other databases, please see
# MySQL parameters below as an example.

# sqlite parameters
# for relative file location ./data/sqlite3.db, url is
# sqlite:///data/sqlite3.db
# absolute file location /tmp/sqlite3.db, url is sqlite:///tmp/sqlite3.db
dbUrl : sqlite:///data/sqlite3.db

# MySQL parameters
#dbDialect : mysql
#dbHost : localhost
#dbUser : root
#dbPassword : password
#dbName : openclos

# debug SQL and ORM
# "true" will enable logging all SQL statements to underlying DB
debugSql : false
debugRest : true

#device configuration will be stored by default in DB
# "file" will allow device configuration to store in DB and File
writeConfigInFile : true

# List of colors used in the DOT file to represent interconnects
DOT :
colors :
- blue
- green
- violet
- brown
- aquamarine
- pink
- cadetblue
ranksep : 5 equally

# HttpServer for REST and ZTP.
# To make ZTP work the port has to be 80. IpAddr specified here
# is used to populate dhcpd.conf for ZTP. If no address is provided
# REST will start at localhost
# If protocol is http:
#   - certificate is ignored.
#   - basic authentication is supported but disabled by default.
# If protocol is https:
#   - basic authentication is enabled by non-empty username and password.
#   - this openclos.yaml comes with a predefined username 'juniper' and
#     password is 'juniper' ($9$R9McrvxNbo3DWL3DikTQEcyc)
#   - if you need to use a different password, run "python crypt.py
#     <cleartext_password>" to generate a 2-way encrypted password.
#   and copy it to 'password' attribute
# certificate must be full path to the server cert file. it can be generated by running "openssl req -new -x509 -keyout server.pem -out server.pem -da y3 365 -nodes"
# you MUST change 'ipAddr' to the IP address of the REST server. REST server won't run if ipAddr is 0.0.0.0 in https mode
restServer:
  version : 1
  protocol : https
  ipAddr : 0.0.0.0
  port : 20443
  username : juniper
  password : $9$R9McrvxNboJDWJdikTQEcy
  certificate : ~/openclos.pem
restServer:
  version : 1
  protocol : http
  ipAddr : 10.94.63.190
  port : 20080

# Number of threads used to communicate with devices
report :
  threadCount : 20

# SNMP trap settings for OpenClos
# OpenClos uses traps to perform staged ZTP process
# target address is where OpenClos is running (same as httpServer:ipAddr)
# threadCount: Number of threads used to start 2-stage configuration for devices
snmpTrap :
  openclo2_trap_group :
    port : 20162
    target : 10.94.63.190
    threadCount : 10

# various scripts
# Note for release 1.0, the backup database script is engine specific
script :
  database:
    backup : script/backup_sqlite.sh

# CLI configuration
cli:
  # This is the text that would appear at each prompt
  prompt_text: "openclos"
  # prompt_style follows prompt_text and these together make the command-
  # prompt of the CLI
  # The cli code will add <space> after the prompt_style
  prompt_style: "#"
  # header is the text that appears when CLI is invoked, and CLI prompt-
  # is issued
  header: "Welcome to openclos - by Juniper Networks"
  # on_exit is the message that would appear when CLI session is terminated
  on_exit: "goodbye"

# Optional callback to control 2-stage configuration processing.
# callback can be a shell command or a shell script.
# if the callback exit code is 0, 2-stage configuration for the current leaf continues,
# if the callback exit code is not 0, 2-stage configuration for the current leaf aborts
#twoStageConfigurationCallback:

# generic plugin configuration
plugin:
  - name: overlay
game: jnpr.openclos.overlay

# Number of threads in the thread pool for committing configuration on
device
threadCount: 10

# Number of seconds. Controls how frequent to scan "commit job queue"
dispatchInterval: 10

5. After editing the above files, navigate to the sampleApplication.py script and run it to generate the device configurations and push them to the devices.

These configurations will be pushed to the devices using ZTP (if configured above).

```
user@ubuntu-Openclos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/conf$ cd ../tests
user@ubuntu-Openclos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/tests$ python sampleApplication.py
user@ubuntu-Openclos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/tests$
```

6. To review the generated configuration files, navigate to /tmp/out and then into the directory holding the configuration files you just created.

The directory name is autogenerated using the format <POD-ID_POD-NAME>. In this example, the directory name is aeb1ba8e-207a-4317-bfb3-08906333bde6-labLeafSpine.

```
user@ubuntu-Openclos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/conf$ cd /tmp/out
user@ubuntu-Openclos:/tmp/out$ cd aeb1ba8e-207a-4317-bfb3-08906333bde6-labLeafSpine
user@ubuntu-Openclos:/tmp/out/aeb1ba8e-207a-4317-bfb3-08906333bde6-labLeafSpine$
ls
2304a28c-6d42-4893-ad60-1a378ce65e30__Spine-03.conf
24d27de-e1bb-475e-a426-165a26f9d53a__Leaf-02.conf
3cadc8f1-82a4-433c-acde-3dca5c82c06__Leaf-03.conf
46ffc4da-3271-4c39-a9fb-2ef800f2ae0b__Leaf-05.conf
6d0b5cc1-fa72-4eff-83a3-9c6f35f91bd2__Spine-00.conf
89fd49f5-e878-4a29-81bd-a0fec3949354__Spine-01.conf
a69d50ef-b749-4eb8-aeeb-e7daa318036e__Spine-02.conf
ed6db70-dbca-4bcb-8fba-5b45e631ab62__Leaf-01.conf
f1eea9a7-cf5b-43af-a444-56a170598657__Leaf-00.conf
cablingPlan.dot
cablingPlan.json
```

---

**NOTE:** When using the scripts to create multiple sets of configuration files, it can become difficult to determine which directory holds which files, as the POD ID does not use an intuitive format. One way to distinguish one directory from another is to alter the POD name (in the closDefinition.yaml file) each time you create configuration files. Another method is to look at the timestamps to determine which directory was created most recently.

```
user@ubuntu-Openclos:/var/tmp/OpenClos-devR3.0/jnpr/openclos/conf$ cd /tmp/out
user@ubuntu-Openclos:/tmp/out$ cd aeb1ba8e-207a-4317-bfb3-08906333bde6-labLeafSpine
user@ubuntu-Openclos:/tmp/out/aeb1ba8e-207a-4317-bfb3-08906333bde6-labLeafSpine$
ls
2304a28c-6d42-4893-ad60-1a378ce65e30__Spine-03.conf
24d27de-e1bb-475e-a426-165a26f9d53a__Leaf-02.conf
3cadc8f1-82a4-433c-acde-3dca5c82c06__Leaf-03.conf
46ffc4da-3271-4c39-a9fb-2ef800f2ae0b__Leaf-05.conf
6d0b5cc1-fa72-4eff-83a3-9c6f35f91bd2__Spine-00.conf
89fd49f5-e878-4a29-81bd-a0fec3949354__Spine-01.conf
a69d50ef-b749-4eb8-aeeb-e7daa318036e__Spine-02.conf
ed6db70-dbca-4bcb-8fba-5b45e631ab62__Leaf-01.conf
f1eea9a7-cf5b-43af-a444-56a170598657__Leaf-00.conf
cablingPlan.dot
cablingPlan.json
```
dhcpd.conf

For More Information

The resulting leaf configuration from this procedure can be found at “Configuring Leaf Devices for the IP Fabric” on page 19.

The resulting spine configuration from this procedure can be found at “Configuring Spine Devices for the IP Fabric” on page 31.

More detailed information on creating Layer 3 fabrics using OpenClos can be found at https://github.com/Juniper/OpenClos.